IP (Intellectual Property) CAD Validation Solution using Crossfire

Thesis Report

submitted in partial fulfillment of the requirements
for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Anshul Aneja
(801532005)

Under the supervision of:

Dr. Seema Bawa
Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004
June 2017

Acknowledgement

This research work would be incomplete without acknowledging the people who supported and guided me for the successful completion of this task.

First of all I wish to acknowledge the benevolence of God who gave me courage and strength to face the challenges and to overcome the obstacles that occurred while working on this task.

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 work.

I wish to express my profound gratitude to **Dr. Seema Bawa**, Professor, Computer Science & Engineering Department, Thapar University, Patiala 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.

I am grateful to **Dr. Maninder Singh**, Hon'ble Head of Computer Science & Engineering Department, Thapar University, Patiala for his kind support and providing basic infrastructure and healthy research environment.

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

- Anshul Aneja

801532005

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"IP (Intellectual Property) CAD Validation Solution using Crossfire"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science & Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of **Dr. Seema Bawa** and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Anshul Aneja
801532005
ME (CSE)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

22/07/2017

Dr. Seema Bawa
Professor
Computer Science & Engineering Department
Thapar University, Patiala

Abstract

Developments in the innovation business has made our every day life simpler than at any other time. Gadgets like cell phones, touch sensors, desktops and numerous different parts of advanced world are being more proficient and more astute every day. Each one of these gadgets contains silicon chip which is intended to perform a specific task. Intellectual Property (i.e. IP) inside a silicon chip is a portable design that can be re-utilized. An IP is a block of basic functionality that is licensed to various vendors/organizations when utilized in different silicon chip designs.

An Intellectual Property contains various views like CDL (Circuit Design Language), GDS (Graphical Design System), LEF (Library Exchange Format), etc. So as to make a gadget's working right, one needs to approve IP which thusly prompts towards validation of IP CAD Views or the Library Views. Validation is an exceptionally pivotal part in the evolution of an IP. IP Validation is possible done using two different methodologies i.e. either through Manual Validation or through Automated Validation. Manual IP CAD Validation is executed by an individual with no assistance of any tool/software. However, due to some confinements like less precise outcome, less reliability and time consuming process, we prefer automated IP validation which is executed with the assistance of a tool/software.

IP CAD validation is a vital process that helps the user or designer to detect critical defects prior to be use inside a real SoC design. This calls for the need of an IP CAD validation solution that precisely confirms the accuracy and adequacy of the IP design. This is where the IP Validator (IPV) comes into the picture. IPV is a Unified Framework that encapsulates all checks for an efficient and optimized IP Validation. It also includes Crossfire tool (from Fractal Technologies) as one of its component. In this thesis, a versatile IP CAD Validation Solution has been proposed that provides an escalated validation of different CAD views while hiding all the complexities from the user and in this way expanding the convenience.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IP	Intellectual Property
SoC	System-on-Chip
CAD	Computer Aided Design
EDA	Electronic Design Automation
GDS	Graphical Design System
CDL	Circuit Design Language
LEF	Library Exchange Format
IPV	IP Validator
DRAM	Dynamic Random-Access Memory
SRAM	Static Random-Access Memory
ROM	Read Only Memory
USB	Universal Serial Bus
PLL	Phase-Locked Loop
DRC	Design Rule Check
ASCII	American Standard Code for Information Interchange
HDL	Hardware Description Language
GUI	Graphical User Interface
TCL	Tool Command Language
HTML	Hyper Text Markup Language
CSV	Comma Separated Values
PDF	Portable Document Format
RTL	Register Transfer Level

Chapter-1

Introduction

1.1 Motivation

The main aspects contributing to the success of an IP (Intellectual Property) is the quality of IP views and the other major factor being the time to reach the market. Arteris, Inc. (a multinational firm that builds up the on-chip interconnect technology utilized as a part of SoC semiconductor designs for different types of electronic gadgets) conducted a survey [1] related to top design challenges that contributed to delayed delivery of SoC. As per the survey reports, quality problems and bugs contributed to 8% and debugging designs contributed to alarming 11%. Therefore, an enhanced CAD quality will diminish these ratings to a greater extent. Finding and fixing IP quality issues at some later stages is costly in terms of both time span and money. Additionally issues that are recognized at later stage are costly to settle as well as harm the manufacturer's brand image. Using Figure 1.1, Arteris presented the loss in revenue that the company may suffer on being late to the market.

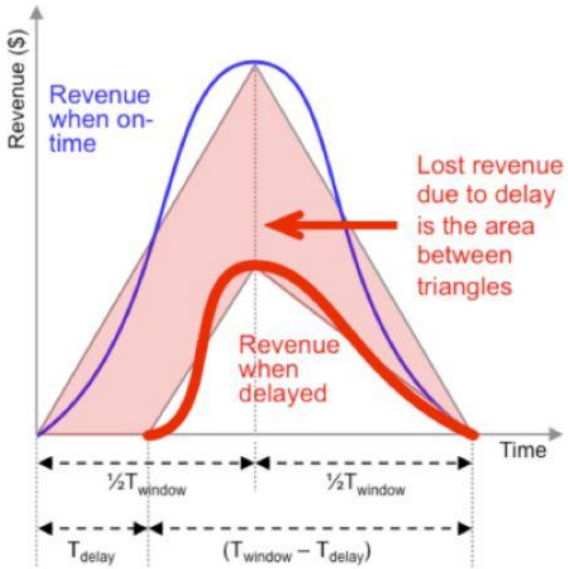


Figure-1.1 Anticipating the incomes over the lifetime of the SoC will be instrumental in ascertaining the expenses of being late. (Source: Arteris)

A well defined validation strategy help the designers to deliver an on-time and exceedingly dependable product and furthermore diminishing the general process duration. The trivial technique of checking views through the RTL2GDS flow had a gigantic process duration. [2] Also, checking the compatibility of the CAD view with different EDA tools took a significant amount of time. Thus the need of a common

platform was always there. For the solution of this problem R Kannavara in [3] proposed the idea of a unified bound IP validation solution and furthermore it highlights the exceptional issues that should be resolved to engage such a unified validation solution to be effective.

1.2 Objectives

The major objective behind this thesis are as follows:

a) To enhance the previous IP Validation approach being utilized at ST Microelectronics.
b) To assess the existing validation methodology.
c) To propose and deploy the improved validation solution for providing client (i.e. the designer) support.

1.3 Expectations from the New Solution

Functional (User):

a) Validation of a single view and automated enabling of checks with respect to the CAD view provided as input.
b) Automatic setup creation and enablement of check if an entire library or multiple views are provided as input for validation.
c) A unified solution with a complete coverage of checks. Encapsulating all EDA tool based checks inside a single framework.
d) More flexibility to user in terms of configuration of checks and the level of validation required.
e) Incremental handling of failed checks. So that designer can re-run only the failed checks with a single command.
f) The solution should be able to handle multiple IPs simultaneously and classification checks as per taxonomy.
g) Flexibility to the user to add any new check if required.

Technical:

a) Improved check algorithms in terms of run time.
b) More use of data structure than intermediate files.
c) New architecture should be flexible enough to use any other In-house or Third party tool (other than Crossfire).

1.4 Thesis Outline

The organization of this thesis is as follows:

Chapter 2 (Literature Survey) describes the fundamentals required for this thesis. It describes the Significance of Validation Phase in the Library development Flow. It describes the review of the existing approaches for the Validation process. Pros and cons of the Validation Approaches has been mentioned as well.

Chapter 3 (Proposed Solution: IPV) describes the layered architecture of the framework and its significance.

Chapter 4 (Design and Implementation) describes the whole development of IPV including the prerequisites, user interface and the output directory structure.

Chapter 5 (Results and Analysis) describes the various experiments done on IPV and the analysis of result is outlined.

Chapter 6 (Conclusion and Future Scope) describes the conclusion of the thesis. Future directions in this area is also outlined in this particular chapter.

Chapter-2

Literature Review

2.1 Introduction to Library Views

As discussed previously, Library is collection of IPs and an IP contain various cells and each cell has views which are the representation of the cell. All the cells have : Layout, Abstract, Schematic, Symbolic and Timing view. A cell is delivered as a set of all view and each view is utilized by different EDA tools in a given electronic design. Main library views are explained below :

2.1.2 Categorization of Libraries

Standard Cell Library
As per the name the standard cell library comprises of the various basic components and that's the reason they are also known as Core Library. Different standard functions are already implemented inside a Core Library like Invert, AND, OR etc.

Memory Library
These comprise different types of memories like DRAM (Dynamic random-access memory), SRAM(Static random-access memory), and ROM (Read only memory) etc.

Mixed Signal and Analog Library
Mixed Signal and Analog Library utilizes the CORE Library for their implementation. Various different examples of Mixed Signal and Analog Library are U.S.B. (Universal serial bus) PLL (Phase-locked loop) etc.

2.1.3 Library Views

As specified in the above paragraph, Library is an accumulation of IPs and an IP contain different cells and every cell has views that are the portrayal of a cell. A cell is conveyed as an arrangement of views and every view is utilized by an alternate EDA tool. [4]

Symbolic View

Symbolic View are the sketch (pictorial) portrayal 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. Labels in the symbol are used to add to the documentation of the design, selection box selects the complete area for the cells. 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. Symbolic view of a NOT fate is shown below:

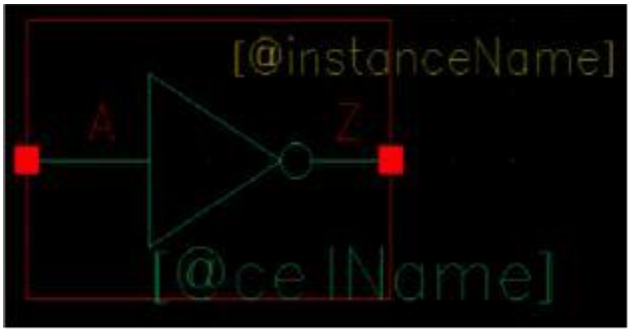


Figure-2.1: Symbol View representation of a NOT gate (Source: STMicroelectronics)

SLIB

SLIB stands for Symbolic Library SLIB le is the textual representation of symbolic view as shown in figure 2.2:

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-2.2: Example of .SLIB (Source: STMicroelectronics)

Schematic View

Schematic view is a simple portrayal of an electronic circuit. These views represents the different components of the circuit as simple standard symbols, and the signal and power connections between the devices. It shows the representation of the cell at transistor level. It represents component instances, wires and pins.

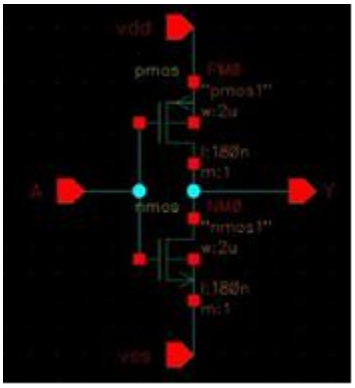


Figure-2.3: Schematic View of an Inverter (Source: STMicroelectronics)

Circuit Description Language (CDL)

CDL file is the text representation of the schematic view as shown in the figure-5. A CDL file provides the following data:

a) Transistor level connections
b) Various device parameters such as width, area, length, device name.
c) Transistor level data related to ground and power pins

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-2.4: Example of .CDL (Source: STMicroelectronics)

Layout View

It is the clear physical portrayal 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). Layout view must pass a sequence of checks during Verification process, The most important checks are Layout vs Schematic (LVS) and Design Rule Check (DRC). The parameters for such check are given by chip manufactures. Another name of Layout View mask design, IC mask layout.

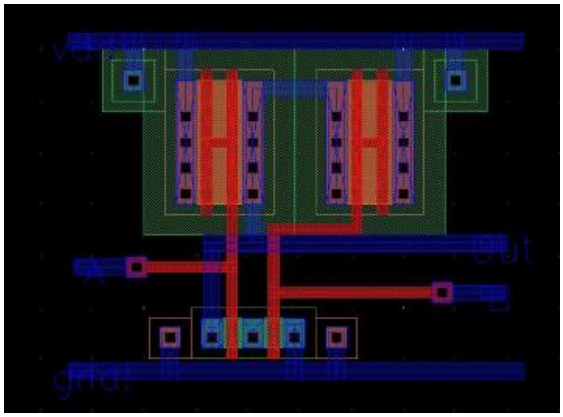


Figure-2.5: Layout View (Source: STMicroelectronics)

Graphical Design System (GDS)

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

Abstarct 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 obstruction area (i.e. non-routing area) between the layers.

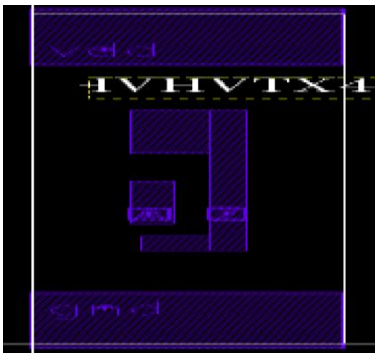


Figure-2.6: Abstract View (Source: STMicroelectronics)

Library Extension Format (LEF)

LEF file is nothing but an ASCII portrayal of the Abstract view. LEF file comprises of the following data:

a) Name of Cell
b) No. of I/O Pins in a Cell
c) Pin definitions
d) Pin directions
e) Cell size
f) Obstruction layer definition
g) Pin locations

Figure-2.7: Example of .LEF (Source: STMicroelectronics)

2.2 Library Evolution Flow

The traditional library development flow presented in the literature focus only on the design phase including FE and BE designs [5], [6], [7], [8], [9]. However as shown in figure-2.8, the library evolution flow is constituted of four fundamental stages. The primary stage is to define a detailed specification of library for the required technology. The next stage comprises of designing of all the cells of the library. The third stage is to get all the fundamental views (like schematic and layout) that are developed during the design stage of the library evolution flow to different CAD views for enabling various different CAD flows. The last and the final stage is the validation of the packaged library based on several criteria prior to the delivery to the concerned customers.

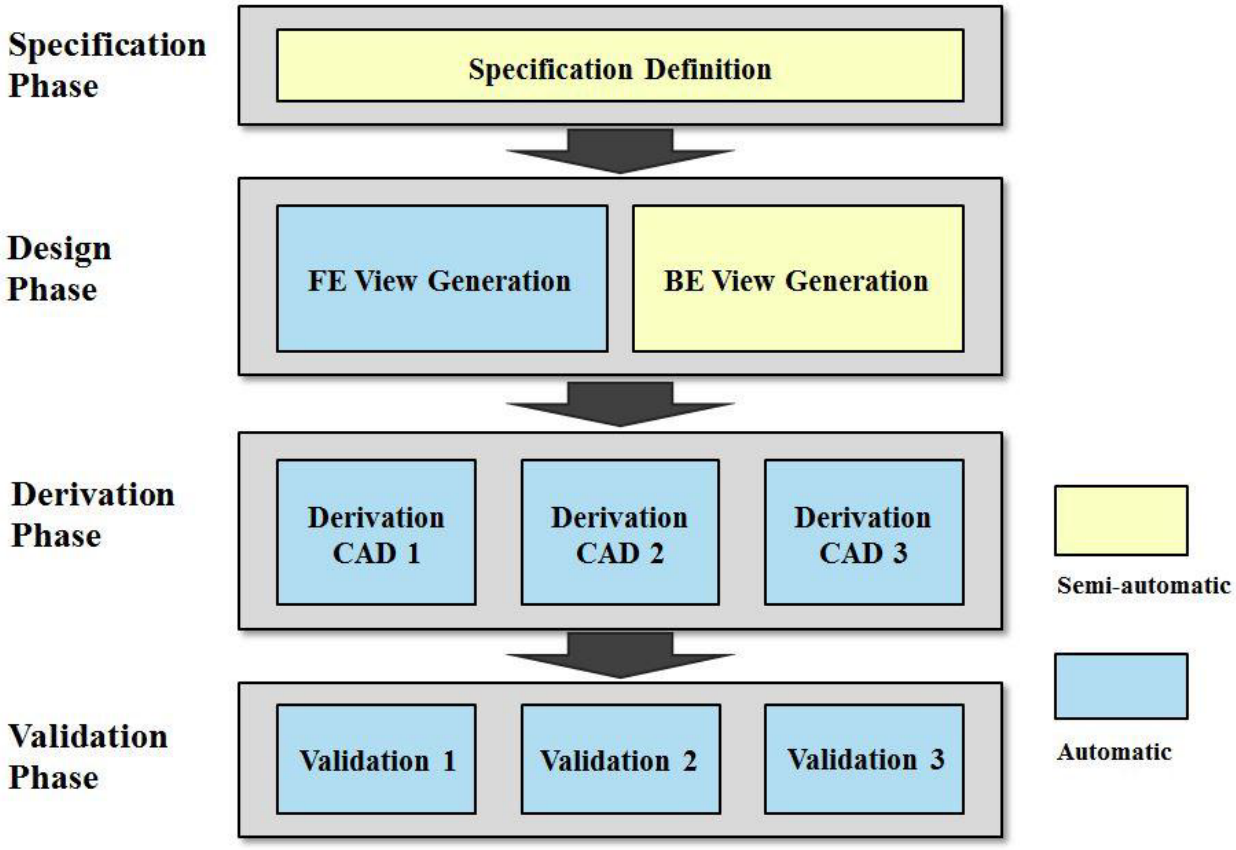


Figure-2.8: Library evolution flow (Source: STMicroelectronics)

2.2.1 Specification Phase

The specification stage is fundamentally essential for library advancement since it goes for gathering all required data. the determination for library improvement must cover an extensive variety of data. Additionally, the data must be gathered from a few data proprietors. In this way, the detail stage is a cooperative work. It comprises of gathering the data and making a determination with the gathered data. For instance, we require a

rundown of library perspectives and computer aided design apparatuses' data for library improvement. Yuan [10] highlighted the association with different functional working groups like CAD developers reliability and process engineering and the final marketing people accumulating the final environmental specifications and considerations for the library development flow

2.2.2 Design Phase

The design stage is to outline all library cells. This stage grants to produce every crucial view for FE and BE outlines of the framework. In particular, as per the cell detail, the cell is composed and after that its physical view is made by utilizing a format editorial manager. Be that as it may, much exertion and time ought to be given to format outline since it decides the execution of the cells.

In the wake of completing physical outline, library perspectives, for example, behavioral model for FE configuration must be delivered. This plan step includes creating behavioral model and timing model. Behavioral model speaks to the basic portrayal and the usefulness of the cell in Hardware Description Language (HDL, for example, Verilog. Despite what might be expected, the planning model containing the planning attributes of the cell can be created by two steps. The initial step is to describe the cell keeping in mind the end goal to concentrate its planning data, for example, postponement and imperatives. The second step is to make a planning model by embodying the got timing data in Liberty design. [11]

2.2.3 Derivation Phase

The derivation stage permits producing different CAD views from the major perspectives keeping in mind the end goal to totally bolster clients' CAD flows since some of them require their own particular semantics because of the absence of institutionalization endeavors.[12] Subsequently, all required CAD perspectives ought to be made by the determination stage to fulfill all clients' outline flows.

2.2.4 Validation Phase

The nature of IPs and libraries is straightforwardly identified not only with their reuse but also with reconciliation as well as the design effectiveness of SoC [13]. Therefore, they should be validated properly prior to delivery to the concerned customers. For this reason, Lin [14] first discussed about the prerequisites of a high quality standard cell library, for example, the appropriate functionality of a cell, its design having no rule violations related to an ideal design and its precise timing execution. Then, he stratified

the general errors in five different categories: inaccuracy , incompleteness, functional errors, inconsistency and design rule violation. As he commented, that errors easily creep in during the library evolution flow. In this way, this stage is really needed for the development of a highly reliable library.

Helena Zheng, highlights the main challenges that are faced in the process of successful IP integration [15]. There is a difference between Verification, Validation and testing of SoC designs [16]. Larry Cooke, explains the reason because of which we still don't have the quality of IP that we expect [17].

A great deal of research work has been done in the past for the improvement of IP quality by enhancing the validation approach used for IP/Library validation. The table below describes some of the related works done in past:

S.No.	Paper Title	Description	Reference No.
1.	Towards a Unified Framework for Pre-Silicon Validation	It highlights the need for a unified pre-silicon validation framework.	[3]
2.	A System to Validate and Certify Soft and Hard IP	They described a software tool named IPscreen that aims to both validate and certify IPand can be extended to all kinds of fl ows, checks and commercial tools.	[18]
3.	Automation of IP qualification and IP exchange	They described the quality risks when a third party IP is integrated. It introduces IP quality aspects and describes the IP qualification platform which is able to check quality characteristics of digital soft IP.	[19]
4.	VSIA Quality Metrics for IP and SoC	They described Quality Evaluation Spreadsheet (QES) that contains the metrics are designed for rapid assessment by the IP provider and the IP integrator	[20]
5.	A Practical IP Quality Measurement Framework	They proposed a framework using the XML schema that can be implemented as a quality measurement program within ASIC design flow that will help the designers to reach a consensus on which quality attributes are important for a given IP block.	[21]
6.	A Qualification Platform for Design Reuse	They proposed an IP qualification methodology for an automated quality check that also incorporates current standards that reduced the IP qualification time to a greater extent.	[22]

7.	Automated IP Quality Qualification for Efficient System-on-chip Design	They proposed an automated qualification platform to evaluate IP quality. The platform is based on the XML Schema technology, so as to be easily extended to accommodate new qualification criteria, changes of design flow and tools version updates.	[13]
8.	Benchmark Circuits Improve the Quality of a Standard Cell Library	He categorized the general errors in five different categories: inaccuracy , incompleteness, functional errors, inconsistency and design rule violation.	[14]
9.	Can IP Quality be Objectively Measured?	They introduced a QIP metrics QIP that the needs of the marketplace by providing an objective means to measure IP, reducing the integration time of quality IP, and providing an measurable method to guide IP vendors in the development of and to showcase the quality of their IP.	[23]
10.	IPQ: IP Qualification for Efficient System Design	They introduced IPQ that aims at assuring the design capability and improving the design efficiency for SoC design, and thus it meets essential subsidy criteria of the EkompaSS initiative.	[24]
11.	Measurement of IP Qualification Costs and Benefits	They proposed antomatic qualification framework in which several qualification methods, like completeness check, coding guidelines and design rule checks, verification and synthesis qualification are implemented.	[25]
12.	Pre-Silicon Security Verification and Validation: A Formal Perspective	They outlined equivalence checking techniques to guarantee that there are no malicious implants in the IP blocks.	[26]

Table-2.1: Related research work

2.2.4.1 Validation Process Parameters

The Validation stream ensures the customer that the IP under test is precisely affirmed. The parameters which are checked in the midst of Validation process are portrayed underneath:

A. Completeness

a) Keeping as a main priority the ultimate objective to ensure the Completeness of the Library, Inside the library package the presence of a particular view is checked. Also, the meta data of views is cross-checked with the record known as vc.bbview (vc.bbview is a ST specific indexation file that is present in every IP/Library. This file contains the information about all the different views present inside the package).
b) Therefore, the completeness for a IP/Library can be ensured by checking whether all the required views are present on the specified paths.

B. Correctness

a) Correctness of the views present in a Library is checked based on the characteristics of a library view as mentioned in the specification.
b) Correctness is ensured by checking the vital characteristic values of a view. After assuring the existence of a library view the characteristic values are cross verified to ensure that the IP/Library has been designed as per required rules.

C. Compatibility

a) The IP/library package must provide the designer with a complete list of views of library available inside the package to design a complete flow. While utilizing these views in a design flow there should be no compatibility problems with the EDA tools. This simply implies that the syntax of the views must be accurate.
b) To ensure the view compatibility, syntax of each view is checked by parsing the view with specific EDA tools. In case of there is any compatibility issue, then it is detected at the initial stages only.
c) Therefore, the compatibility of an IP/Library can be ensured by parsing the views with different EDA tools.

D. Consistency

a) The consistency among the various library views is ensured.
b) It is guaranteed that data is reliable between different views. For example, the cell names used in a particular view are same as the cell names present in another view.

2.3 Different Validation Approaches

2.3.1 Manual Validation

Manual validation is the most established and most thorough kind of validation technique. Manual IP validation requires an analyzer to perform manual test operations on the IP/Library without the assistance of test automation.

A person performing manual IP validation should possess the following qualities:

a) Observant
b) Creative
c) Open-minded
d) Skillful
e) To be patient
f) Resourceful

Benefits with Manual Validation:

a) Manual Validation is the eye ball testing.
b) Automation can't supplant human instinct, surmising, and inductive thinking.
c) This is approach is more dependable than the automated one (as in many situations it won't cover all the cases).
d) It requires less time and cost to start profitable validation using this approach.
e) Computerized Testing can't change course amidst a trial to analyze something that had not been already considered.

Problems with Manual Validation:

a) Running test manually is very time consuming job. Validation of a single IP may take upto 4-5 man days.

2.3.2 Automated Validation

Automation is the use of strategies and tools to diminish the requirement of human involvement or interaction in repetitive or redundant tasks. Automated validation, when done effectively can have many points of interest and be extremely valuable to the organization. There are however a few pitfalls or hindrances of test mechanization that one should know about.

Benefits with Manual Validation:

a) **Repeatable:** We can test how the product responds under rehashed execution of similar operations.
b) **Extensive:** We can manufacture a suite of tests that covers each element in your application.
c) **Cost Lessening:** As the quantity of assets for relapse test are diminished.
d) **Reliable:** Tests perform exactly similar operations each time they are run, in this way disposing of human mistake.
e) **Programmable:** We can program complex tests that bring out concealed data from the application.
f) **Quick:** Automated run tests altogether are speedier than human clients.

Problems with Automated Validation:

a) Capability is required to compose the automated test scripts.
b) Troubleshooting the test script is real issue. On the off chance that any bug is available in the test script, in some cases it might prompt savage outcomes.

A. Standalone Validation

Standalone Validation ensures that a given IP is aligned with Design Platform Requirement.

Problems with Standalone Validation:

a) The IPs are validated independently and thus sometimes it doesn't work well when they are integrated at a later stage.
b) The IPs validated using this approach are not very reliable.
c) Standalone validation has limitations with reference in scope of validation checks which ensure IP CAD compliance.

B. Integration Validation

Integration Validation ensures that IPs or Library which are a major part of Design Platform the design of SoC. The conventional approach of validation, Integration Validation, of validating hard and soft IPs for CAD view compliance on requested platforms. After validation, IPs are declared CAD platform compliant to enable SOC design. In the process of RTL2GDS flow, the IPs are taken in a sample or test SoC. Once

the RTL setup is created, a few simulations are done. This confirms the correctness of the IP/Library in terms of flow.

Problems with Integration Validation:

a) It has limitation of large cycle time and large utilization of hardware and manpower resources.
b) Average cycle time of validation by conventional approach is 7 man days per IP on two platforms.

C. Hybrid Validation

The above limitations in both validation approaches led to the invention of an alternate approach of this. This paper proposes a novel approach for an IP CAD validation, called Hybrid validation. Hybrid validation approach is a combination of both standalone and integration validation. This approach allows standalone validation of IP but with following features:

Benefits with Hybrid Validation:

a) Ease the debug
b) Increase the usability
c) Increase coverage
d) Remove checks that are failing "systematically" or with Ambiguous results.

2.4 Previous Validation Solution

The previous solution used at STMicroelectronics has been renamed as Inhouse tool due to confidential purposes. Inhouse tool provides GUI which is a framework built on TCL. Libraries are loaded on the Inhouse tool 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 Inhouse tool. Each tuple of library, plugin corresponds to a new tab in Inhouse tool window. To run the Inhouse tool in GUI mode user has to click on each Checks that he wants to run/execute. To run Inhouse tool 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. Time of execution varies based on the total number of cells in the library and cell size.

2.4.1 Architecture of Inhouse Tool

The previous validation environment comprises of three layers embedded in a framework providing user interface as depicted in figure-2.9. It utilized the hybrid approach of validation. The three layers were:

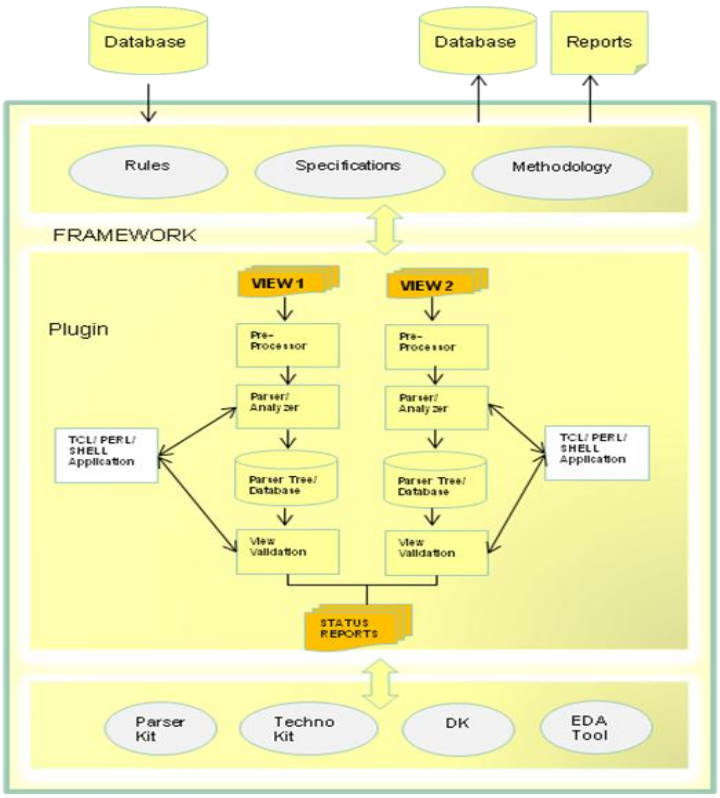


Figure-2.9 Previous solution architecture

a) The Plug-ins.
   i. These plugins were nothing but the different checks grouped together based on taxonomies.
   ii. Plugins were embedded in a framework to provide an interface to the user.

b) The Rules for Static Checks.
   i. Conditions about which checks should be run based on the under test library provided as input.

c) The kits like Techno Kits and Parser Kits
   i. The techno kit contains the information about the different technologies say 28nm, 65nm etc.
   ii. The parser kits provided standard parsers for parsing information from different type of library views.

2.4.2 Inputs to Inhouse Tool

a) **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:
   i. Command used to load Library
   ii. Command used to load auxiliary Library
   iii. Command to run specific Check/Task

b) **Tool & Plugin Information:** On the basis of the technology of library (65nm, 42 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).[4] 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.

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

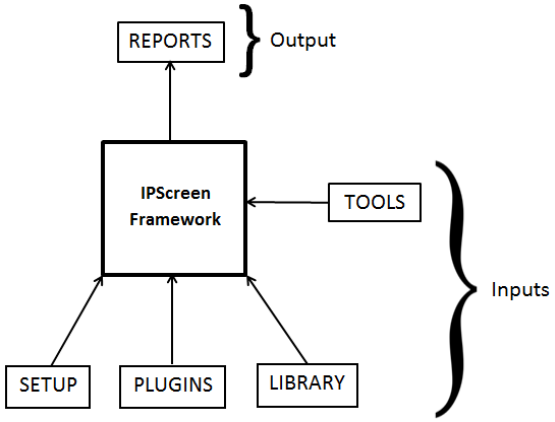


Figure-2.10 Block Diagram of INHOUSE TOOL

Figure 2.10 shows the input given by the INHOUSE TOOL 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 file, HTML.

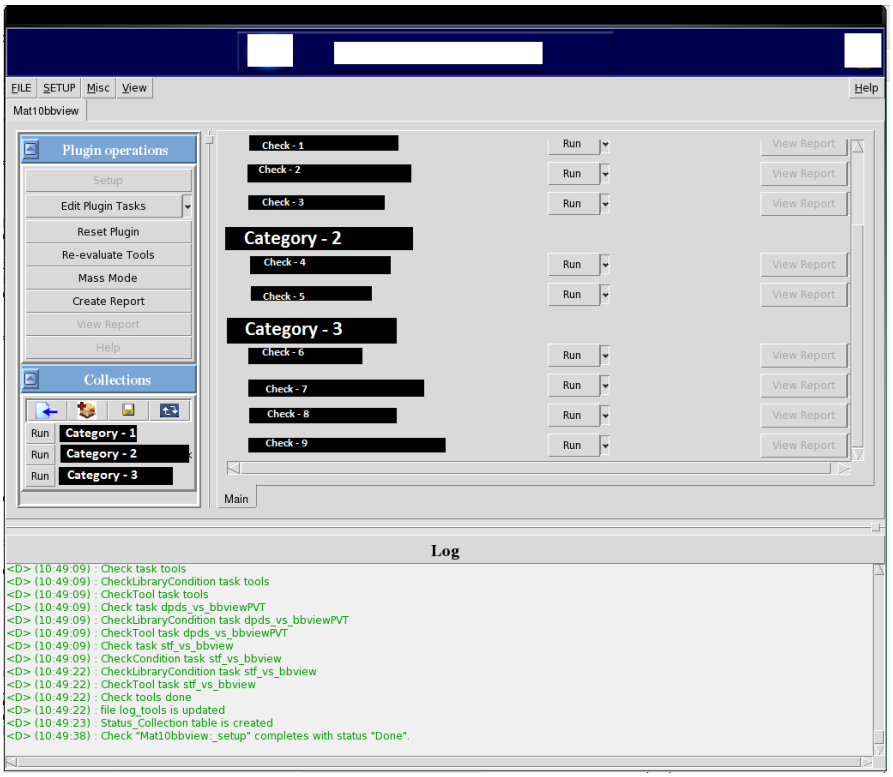


Figure-2.11 GUI of previous solution

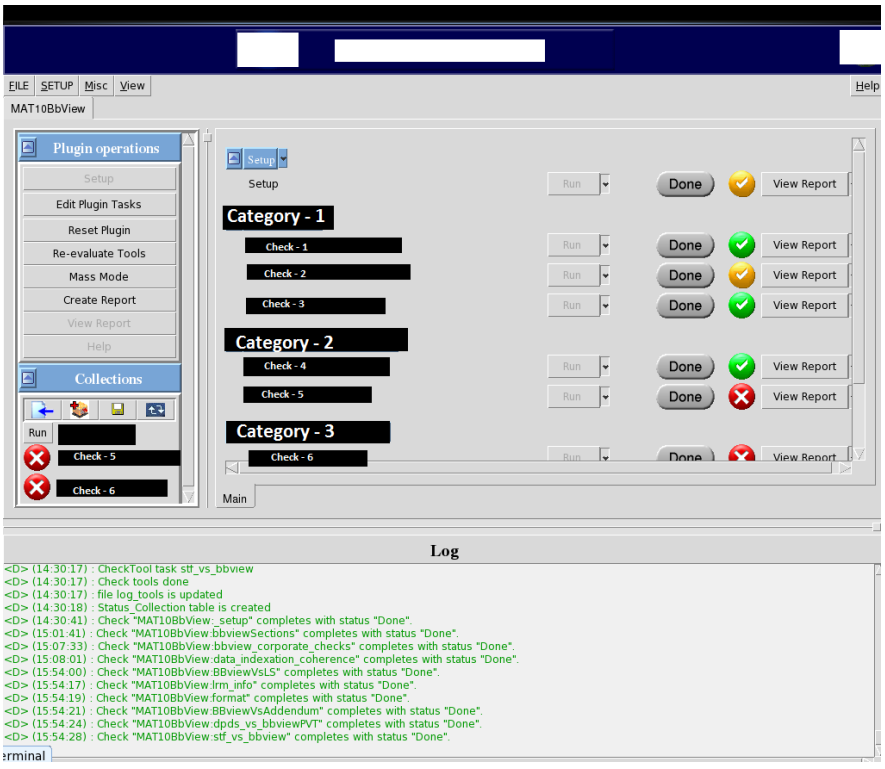


Figure-2.12 Execution of checks in previous solution

2.4.3 Problems with Previous Architecture

a) Validation of a single CAD view was not possible.
b) Every time intermediate data files were generated when an input library was provided for validation. This data was then used by checks. A lot of space was utilized by these intermediate data files.
c) Moreover, the time spent on reading these intermediate files by various different checks was also significant.
d) Very less flexibility for the user in terms of checks and the level of validation required.
e) It doesn't provide a complete coverage of checks. Thus, EDA tool based checks were required to be run independently by the user/designer.

Chapter-3

Proposed Solution: IP Validator (IPV)

IPV is a Unified Framework that encapsulates all checks for an efficient and optimized IP Validation. It also includes Crossfire tool (from Fractal Technologies) as one of its component.

3.1 IPV Architecture

The IPV framework comprises of four layers:

a) Layer1 – ST specific Data and IP Data from and Technology Kits is read and processed for setup generation.
b) Layer2 - This layer comprise of the Crossfire specific setup files.
c) Execution Layer - Execution of checks based on the configuration files.
d) Reporting Layer - This layer collect results for all the performed checks and generate the final reports in HTML as well as CSV and PDF formats.

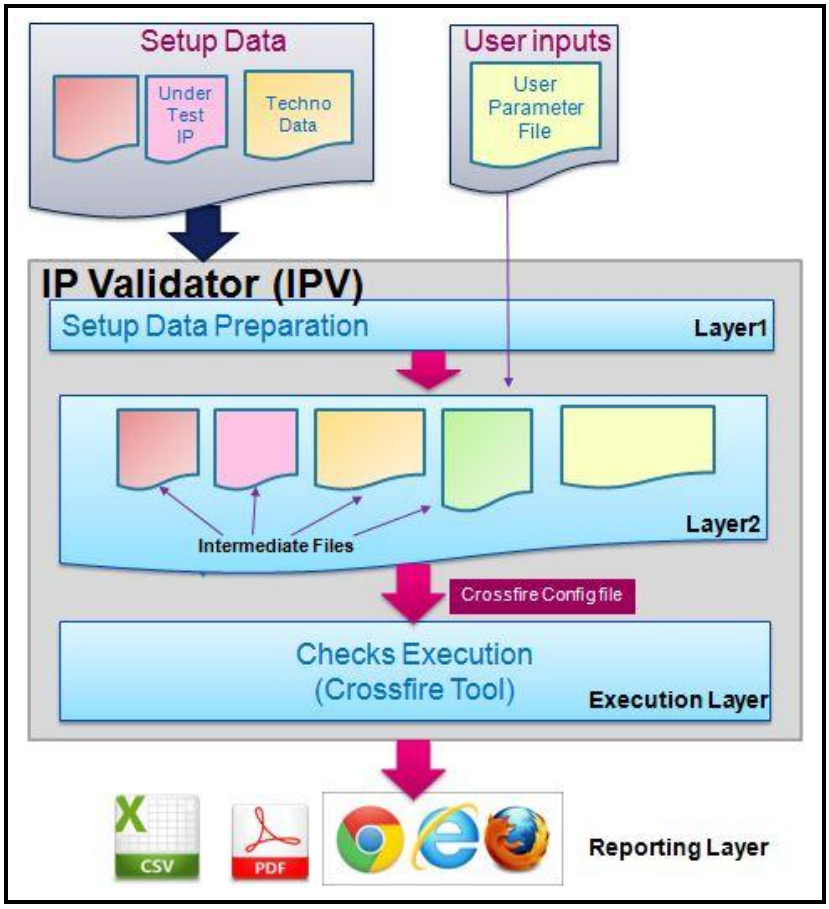


Figure-3.1 Multi-layered architecture of IP Validator

The general principle of using layered structure is to reduce the design complexity. The layered structure enhances modularity, scalability and flexibility of the IPV. All the layers contain different interfaces to interact with each other. Every layer can be independently evolved without impacting the other layers. Also, if we make changes in a layer without impacting the interface, the system remains functional.

The first layer, Layer1, mainly comprises of the Data Assembling step, in which all the inputs from different sources like Technology and IP Data are processed and correlated. IPV second layer, also known as the Layer2, is responsible for translation of ST specific inputs to Crossfire Tool readable format. Here also, the configuration files have been kept modular in order to facilitate updates in the files with no impact on each other. The next layer is check execution layer of the solution having Crossfire tool as its major component. This layer enables the execution of checks. Finally the fourth layer gather results for all the performed checks, waive any errors that are not useable for the user/designer, categorize the reports on the basis of checks, views, cell and generate the final reports in HTML as well as CSV and PDF formats.

3.2 Importance of IPV

IPV structure provides an efficient validation strategy for different kind of IPs. As compared with the previous validation strategy that needed MAT9 or either MAT10 IPs for validation, IPV outfits the IP Designer with ability to approve or validate view at any possible phase during the development cycle, paying little attention to level of improvement or the structure of library. Utilizing IPV, issues can be distinguished at the initial stage only and consequently decreasing the feedback cycle as depicted in figure-3.2.

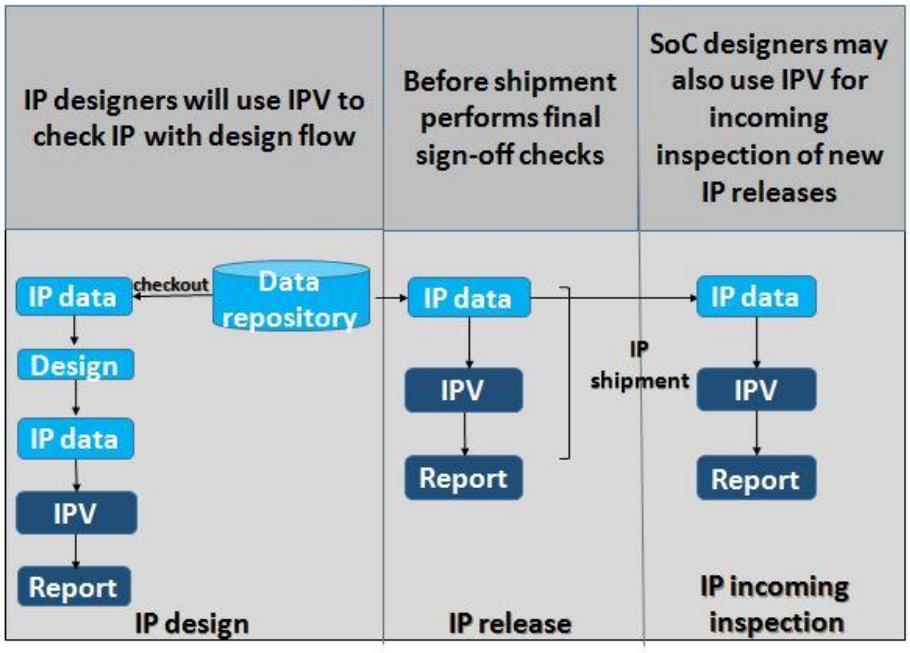


Figure-3.2: IPV at different stages of IP design flow

IPV is a completely unified platform as it combines checks based on different EDA tools in a single place that provides profitability upgrade, and streamlined flow for validation with a much large and enhanced checklist. With IPV, the user/designer gets the flexibility to design any new checks or including a custom view that is not supported by the framework. Moreover, any new and unavailable EDA based check that is not previously available inside the framework can also be included within this framework only.

Every existing check can be arranged by the user as per the need, i.e. the user/designer utilizing the IPV can configure the options and refine all the available checks as per the need. This solution keeps the Quality - Run time exchange off which can be tampered by the user/designer as wanted. If the user/designer needs a quick run time, so some check options can be tampered with a particular true objective to approve only few cells available inside each CAD view.

Another great purpose of utilizing IPV is the setup once generated can be used again and again to approve the view , in the long run diminishing the run time needed to validate required IP views.

3.3 IPV Execution Modes

The diagram below describes the different execution modes of the IP Validator:

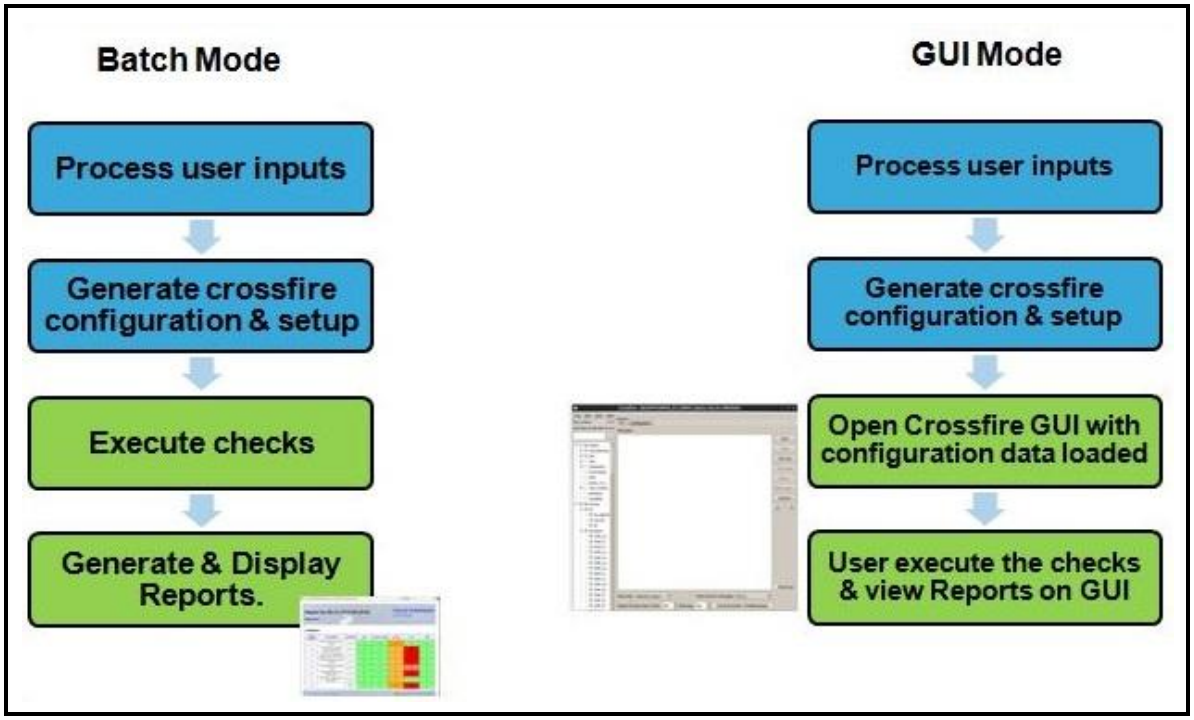


Figure-3.3: Different execution modes of IPV

Chapter-4

Design and Implementation

As the entire implementation work was done at ST Microelectronics, so due to the company policies some of the command and check names have been rephrased.

4.1 Prerequisite

4.1.1 Technical Requirements

Platform	Linux
RAM required	4 GB+
Programming Language Used	Python
	Python v2.5.XX - 2.7.XX
File Formats (for input and intermediate configuration files)	CSV (comma separated values) format, YAML(YAML Ain't Markup Language) format

Table-4.1: Technical requirements

4.2 User Interface

A total of four commands with various different options is provided to the user for execution using IP Validator. The commands are:

a) **command1**
b) **command2**
c) **command3**
d) **command4**

All the above mentioned commands with different command line options are described below:

A. command1

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

command1

-libraryName <library name value> -libraryPath <Library path value>

-ipStyle {MACRO, MEMORY}

-config <user parameter config file path>

[-milestone {PRELIM, INTERMEDIATE, FINAL, BE, FE}]

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* and MEMORY only. This is MANDATORY.

-config:

User specified checks and parameters configuration file. This is MANDATORY.

-[milestone]:

Current milestone of IP. The milestones supported are PRELIM, INTERMEDIATE, FINAL, BE and FE. The default value of milestone is FINAL. This is OPTIONAL.

[-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 View Format Mapping file. This is OPTIONAL.

B. command2

This command is used to generate the user configuration template file (based on IP Type).
The user configuration file is required for command1 (for checks execution). In this file, user can provide parameters related to tool Setup & Checks.
This command operates only in BATCH Mode.

command2

-ipStyle {MACRO, MEMORY} [-filePath <Output filename or directory>]

Where,

-ipStyle*:*

Refers to valid Type of IP. Current release is limited to *MACRO* only. This is MANDATORY.

[-milestone]:

Current milestone of IP. The milestones supported are PRELIM, INTERMEDIATE, FINAL, BE and FE. The default value of milestone is FINAL. This is OPTIONAL.

[-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 “analog_user_setup”.This is OPTIONAL.

C. command3

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, along with the format ID name. This file is given as an Input with *–viewFormatMap* argument to the command1 for generating format IDs based on the Deliverable and View available in the vc.bbview.
This option can be used while creating format ID corresponding to new Deliverable and View or to run the checks on a single view.

command3 [-filePath <Output filename or directory>]

Where,

[-filePath]:

It is User specific Output filename/directory/filepath. By default it will generate a View Format Mapping file with default name in the Current Working Directory.

Currently, the default name is “viewFormatMapping.csv”.This is OPTIONAL.

D. command4

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

command4

[-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 with the name as fnx_database.db). This is OPTIONAL.

[-setup]:

The path of IP setup Configuration file (.cfg file). This is OPTIONAL.

4.3 IP Validator Flow

The diagram below describes the top-level flow of the IP Validator:

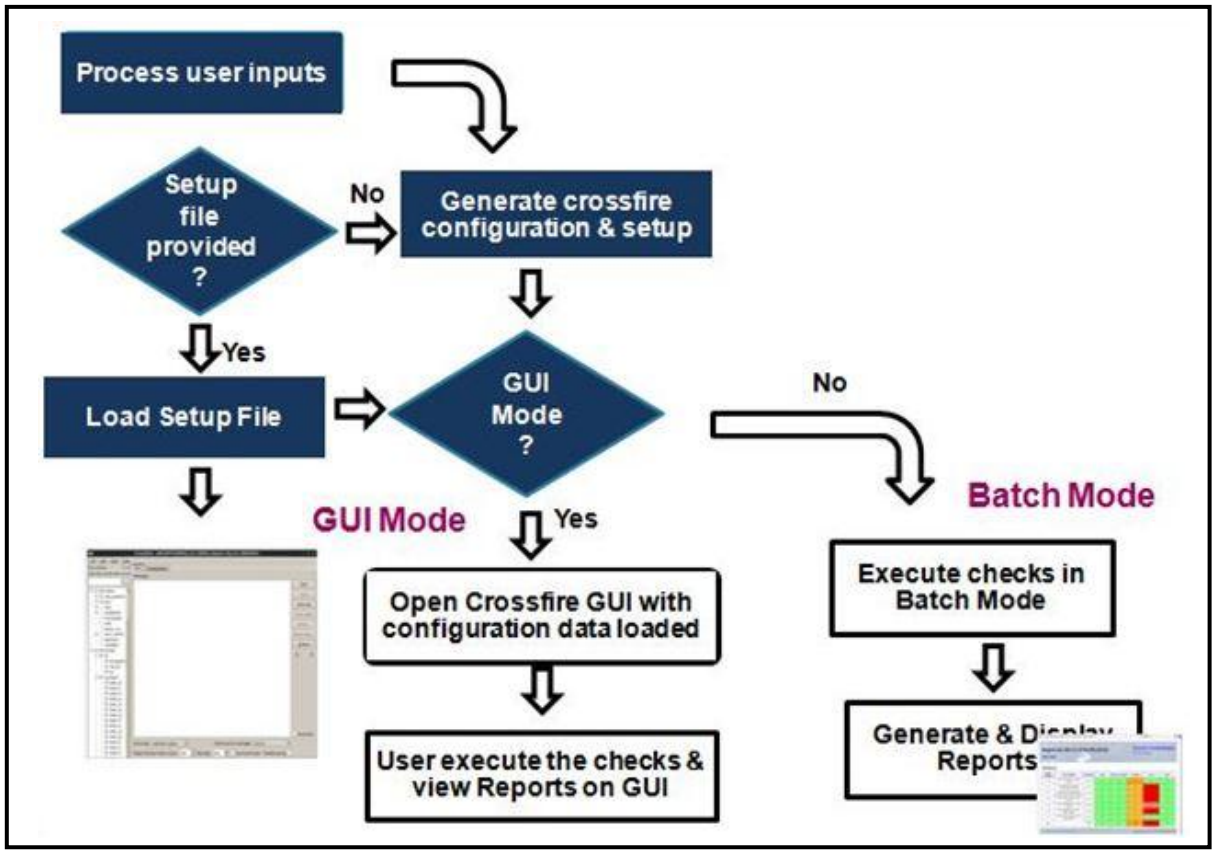


Figure-4.1 IPV flow diagram

4.4 Run Area Directory Structure

A. **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.
B. **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. command1, command2, command3 and command4). It does not include any information related to the checks executed.
C. **crossfire.log:** This is the log file generated by the Crossfire Tool.
D. **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 command4 mainly used to analyze the checks and generate HTML report.

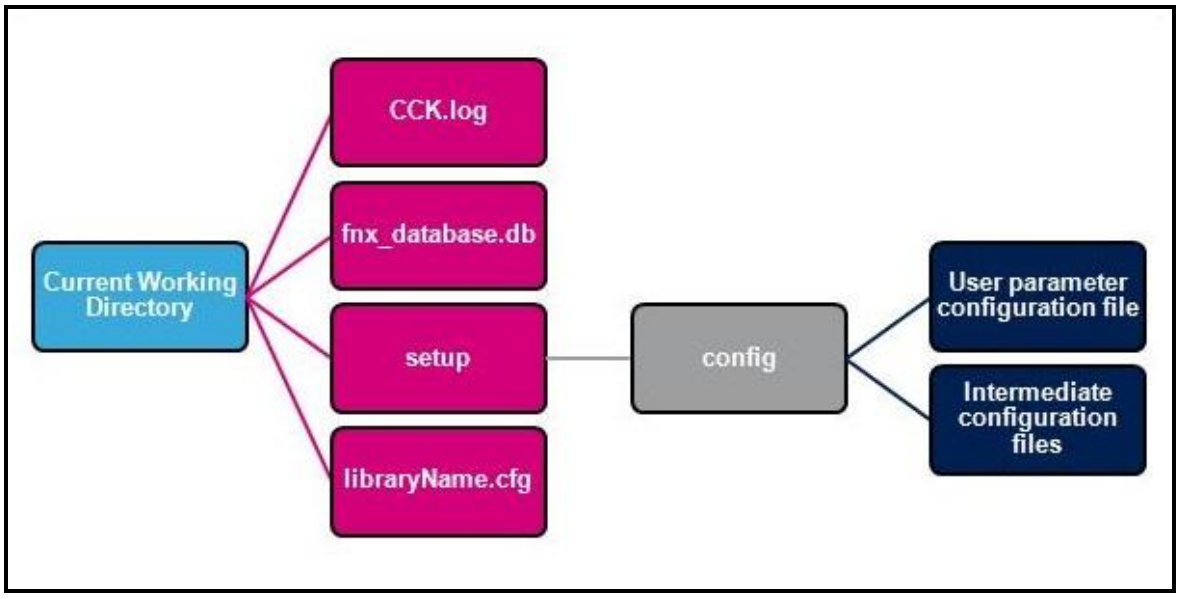


Figure-4.2: IPV run area directory structure

4.5 IP Validator Demo Run

The very first step for the validation of an IP/Library is to generate a user parameter template file using the **command2** as listed above, based on the type of IP/Library. The figure below depicts the generation of user parameter file using the required command.

Figure-4.3 Usage of command2

The next step is the generation of the viewFormatMapping file using the **command3** as described above. This step is optional as it is required only when we need to validate a IP/Library that contains some additional views that are not previously supported by IP Validator.

Figure-4.4 Usage of command3

The next immediate step is to start the process of setup creation using the **command1**. The figures below depicts the usage of **command1** for the generation of setup file.

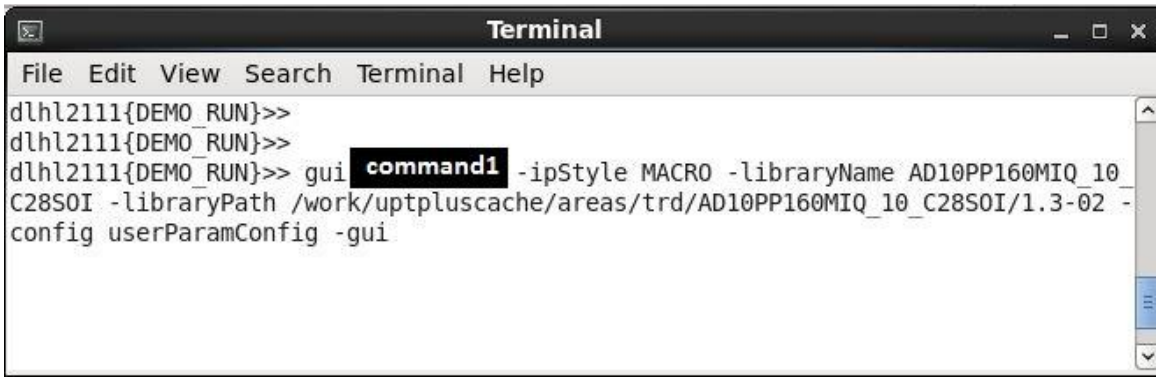


Figure-4.5 Usage of command1 (w/o viewFormatMap option)

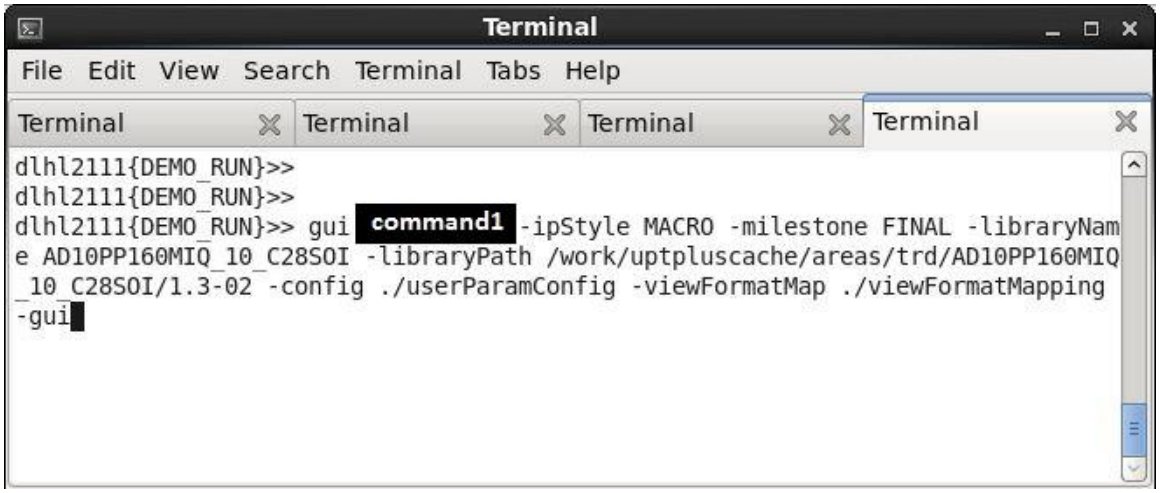


Figure-4.6 Usage of command1

On successful generation of the setup file, the Crossfire GUI is launched as visible in figure below or if gui option is not provided at the command line, then the execution of checks start using the command line mode only .

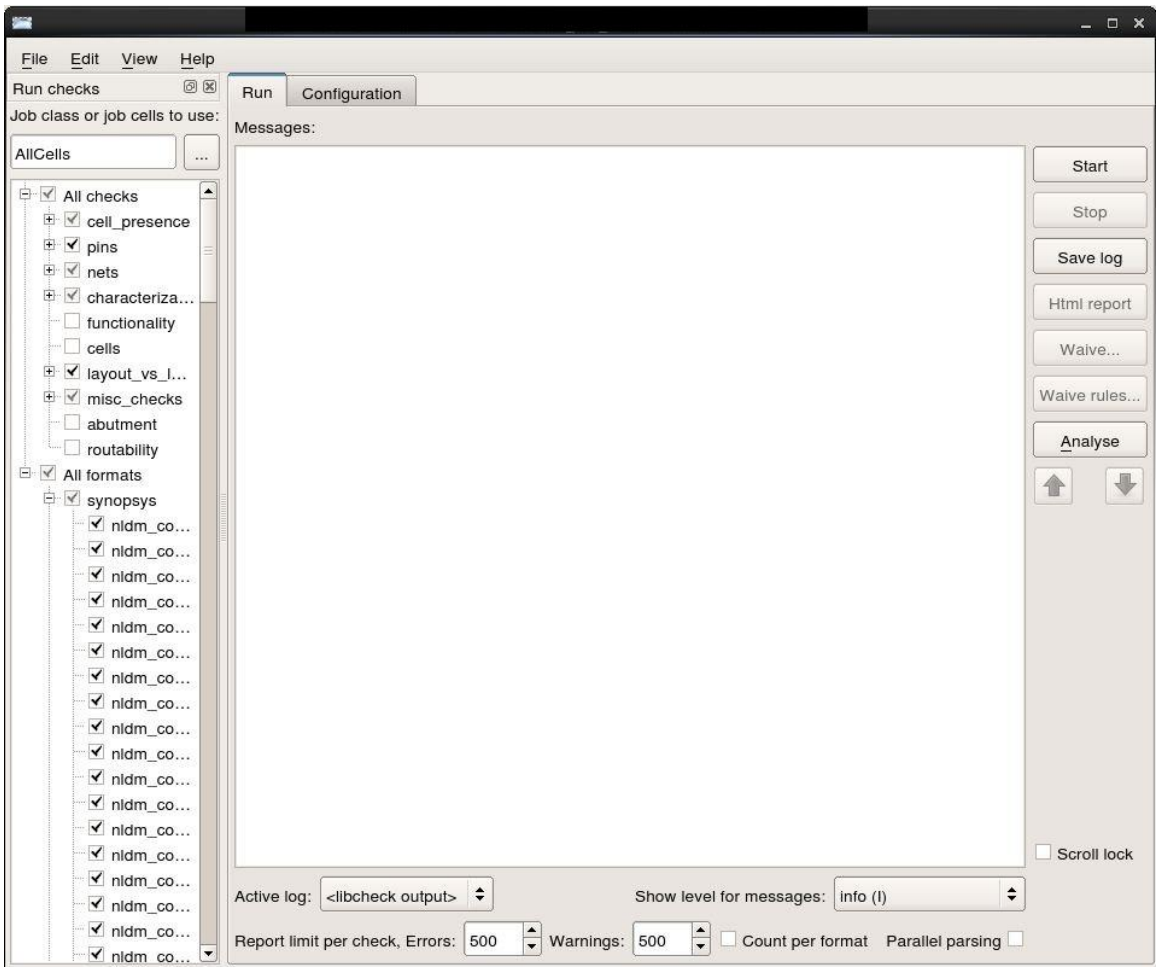


Figure-4.7 Crossfire GUI launched by validateIP command

Once the Crossfire GUI is launched the user can modify any of the check parameters if required and then click on Run to start the execution of checks. On completion of the check execution various different types of reports can be generated and that too with different level of granularity.

Chapter-5

Results and Analysis

5.1 Comparative Analysis

5.1.1 Functional Analysis

S. No.	Evaluation Parameter	Previous Solution	IP Validator
a)	Ease of use (in terms of user inputs)	Moderate	High
b)	Loading of Plugins	Required	Not required
c)	Reporting Feature		
d)	Automated setup generation	Partial	Fully automated setup generation
e)	Granularity of validation (per-cell or per-view)	Low	High

Table-5.1: Functional Analysis

5.1.2 Timing Analysis

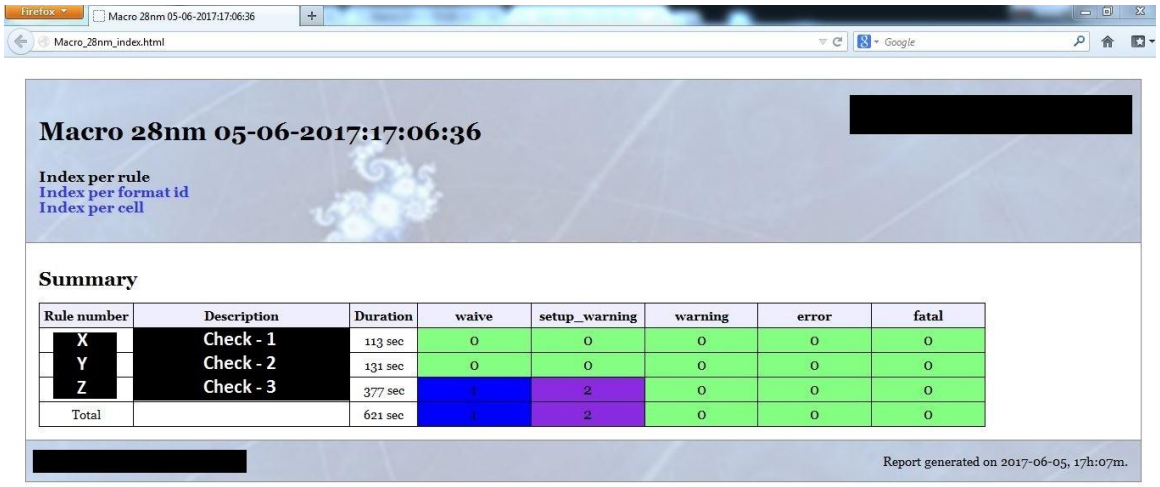
The new validation solution was tested with six different inputs i.e.

a) Macro 28nm
b) Macro 40 nm
c) Macro 65 nm
d) Memory 28 nm
e) Memory 40 nm, and
f) Memory 65 nm

for three different checks. The execution time obtained are visible in figures 24 - 29, that were obtained from different reports as follows:

Final report for Macro 28nm:

Input: Macro 28nm library



Macro 28nm 05-06-2017:17:06:36

Macro_28nm_index.html

Macro 28nm 05-06-2017:17:06:36

[redacted]

Index per rule
Index per format id
Index per cell

Summary

Rule number	Description	Duration	waive	setup_warning	warning	error	fatal
X	Check - 1	113 sec	0	0	0	0	0
Y	Check - 2	131 sec	0	0	0	0	0
Z	Check - 3	377 sec	1	2	0	0	0
Total		621 sec	1	2	0	0	0

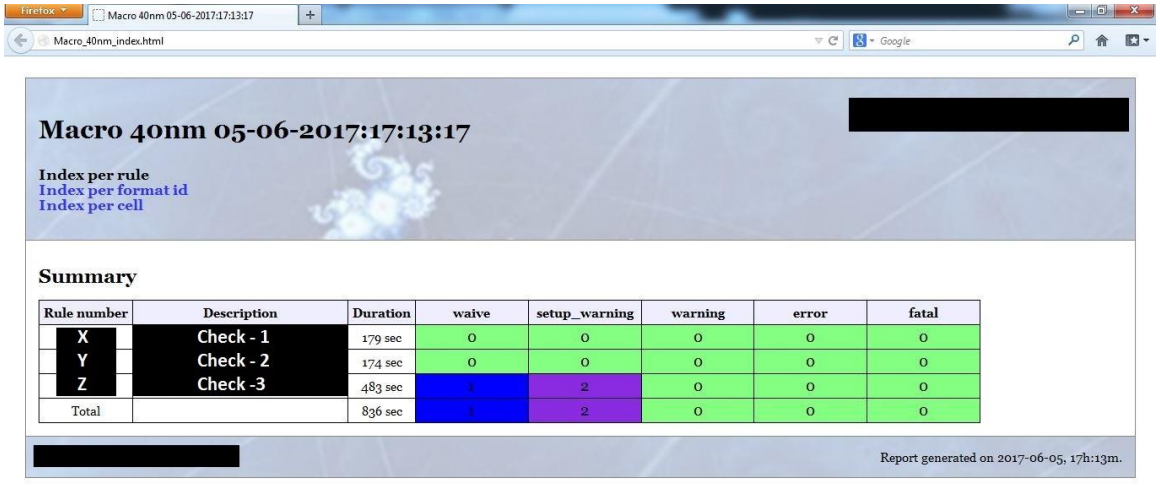
[redacted]

Report generated on 2017-06-05, 17h:07m.

Figure-5.1: Validation report for Macro 28nm as input

Final report for Macro 40nm:

Input: Macro 40nm library



Macro 40nm 05-06-2017:17:13:17

Macro_40nm_index.html

Macro 40nm 05-06-2017:17:13:17

[redacted]

Index per rule
Index per format id
Index per cell

Summary

Rule number	Description	Duration	waive	setup_warning	warning	error	fatal
X	Check - 1	179 sec	0	0	0	0	0
Y	Check - 2	174 sec	0	0	0	0	0
Z	Check -3	483 sec	1	2	0	0	0
Total		836 sec	1	2	0	0	0

[redacted]

Report generated on 2017-06-05, 17h:13m.

Figure-5.2: Validation report for Macro 40nm as input

Final report for Macro 65nm:

Input: Macro 65nm library

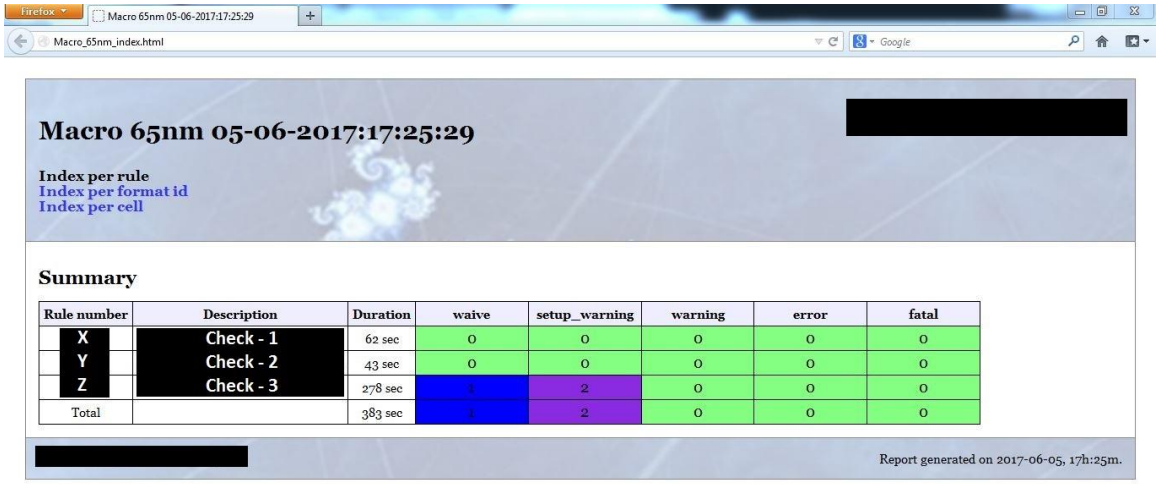


Figure-5.3: Validation report for Macro 65nm as input

Final report for Memory 28nm:

Input: Memory 28nm library

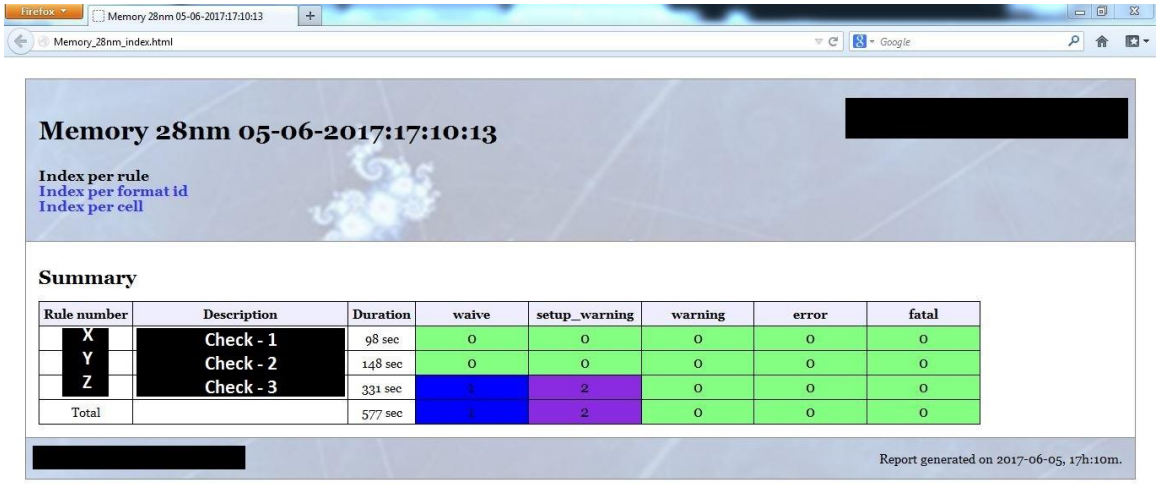
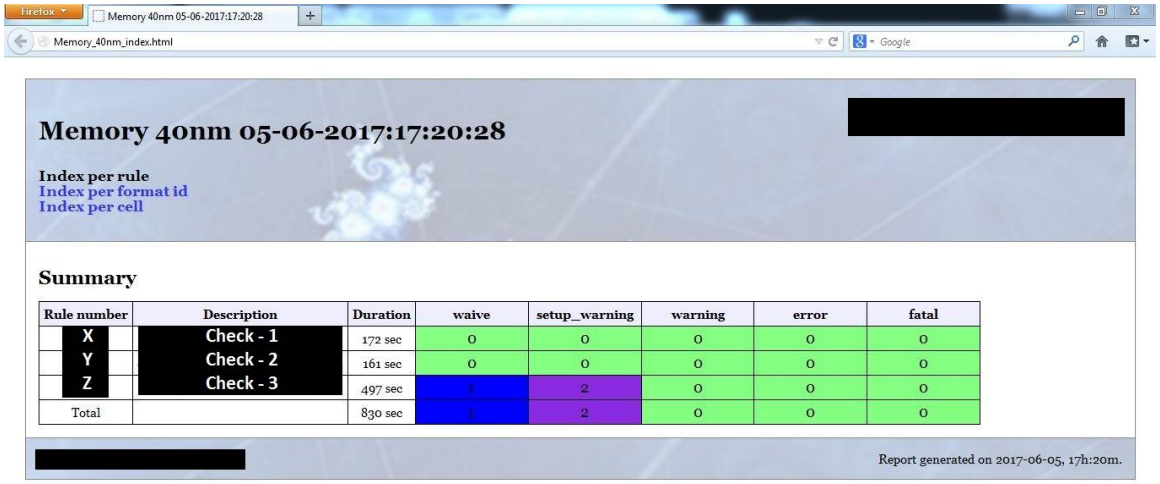


Figure-5.4: Validation report for Memory 28nm as input

Final report for Memory 40nm:

Input: Memory 40nm library



Memory 40nm 05-06-2017:17:20:28

Index per rule
Index per format id
Index per cell

Summary

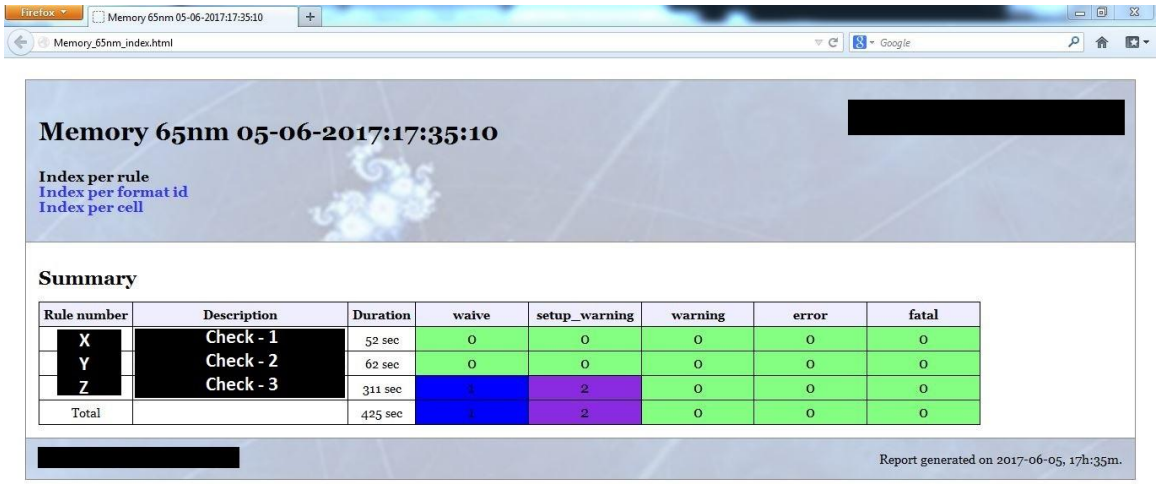
Rule number	Description	Duration	waive	setup_warning	warning	error	fatal
X	Check - 1	172 sec	0	0	0	0	0
Y	Check - 2	161 sec	0	0	0	0	0
Z	Check - 3	497 sec	1	2	0	0	0
Total		830 sec	1	2	0	0	0

Report generated on 2017-06-05, 17h:20m.

Figure-5.5: Validation report for Memory 40nm as input

Final report for Memory 65nm:

Input: Memory 65nm library



Memory 65nm 05-06-2017:17:35:10

Index per rule
Index per format id
Index per cell

Summary

Rule number	Description	Duration	waive	setup_warning	warning	error	fatal
X	Check - 1	52 sec	0	0	0	0	0
Y	Check - 2	62 sec	0	0	0	0	0
Z	Check - 3	311 sec	1	2	0	0	0
Total		425 sec	1	2	0	0	0

Report generated on 2017-06-05, 17h:35m.

Figure-5.6: Validation report for Memory 65nm as input

Comparison of execution time obtained with previous solution:

Check-1

S. No.	IP/Library Technology	IP/Library Type	Check Duration Previous Solution (in seconds)	Check Duration IP Validator (in seconds)
a)	28 nm	Macro	184	113
b)	28 nm	Memory	124	98
c)	40 nm	Macro	239	179
d)	40 nm	Memory	203	172
e)	65 nm	Macro	78	62
f)	65 nm	Memory	57	52

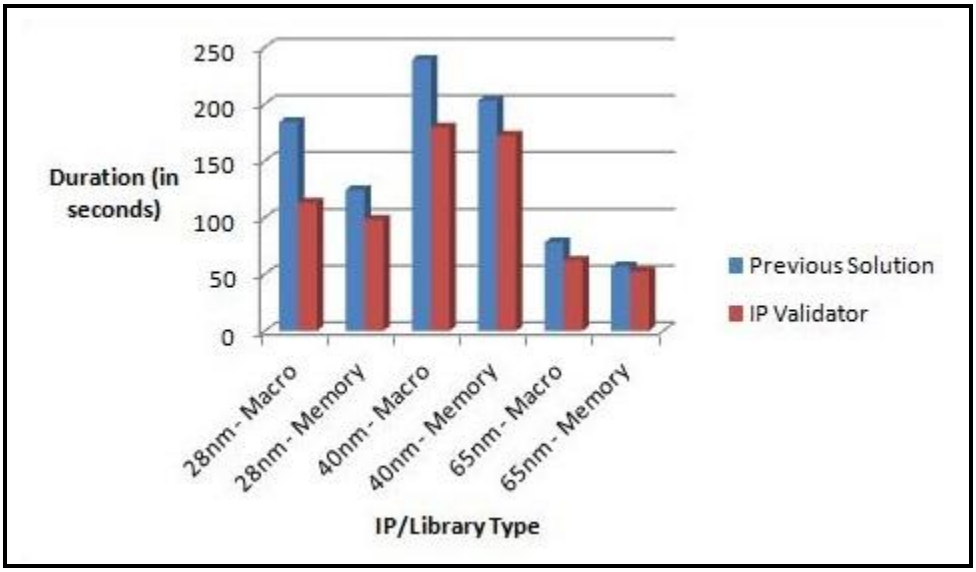


Figure-5.7 Timing comparison chart b/w Previous Solution & IPV (check-1)

As per the figure 5.7, the timing benefit was large for inputs: 28nm Macro and 40nm Macro. However, the gain was average for inputs: 28nm Memory, 40nm Memory. Finally, there was negligible gain in case of 65nm Macro and 65nm Memory.

Check-2

S. No.	IP/Library Technology	IP/Library Type	Check Duration Previous Solution (in seconds)	Check Duration IP Validator (in seconds)
a)	28 nm	Macro	179	131
b)	28 nm	Memory	153	148
c)	40 nm	Macro	169	174
d)	40 nm	Memory	197	161
e)	65 nm	Macro	79	43
f)	65 nm	Memory	84	62

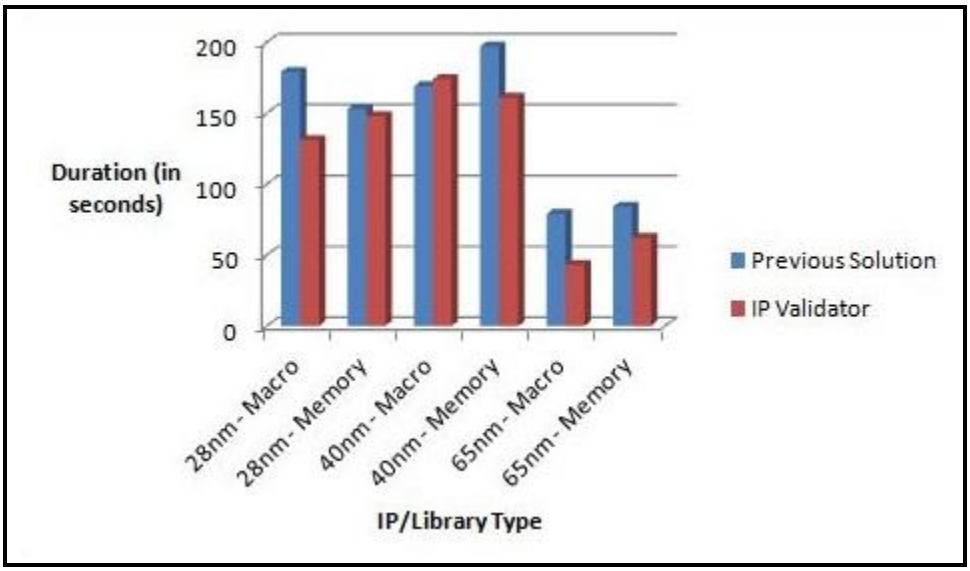


Figure-5.8 Timing comparison chart b/w Previous Solution & IPV (check-2)

According to the figure 5.8, the timing benefit was large for inputs: 28nm Macro and 40nm Memory. However, the new solution took more time for 40nm Macro as compared to the time taken by the previous validation solution. There were marginal gains in case of 28nm Memory and 40nm Macro.

Check-3

S. No.	IP/Library Technology	IP/Library Type	Check Duration Previous Solution (in seconds)	Check Duration IP Validator (in seconds)
a)	28 nm	Macro	492	377
b)	28 nm	Memory	412	331
c)	40 nm	Macro	512	483
d)	40 nm	Memory	523	497
e)	65 nm	Macro	365	278
f)	65 nm	Memory	389	311

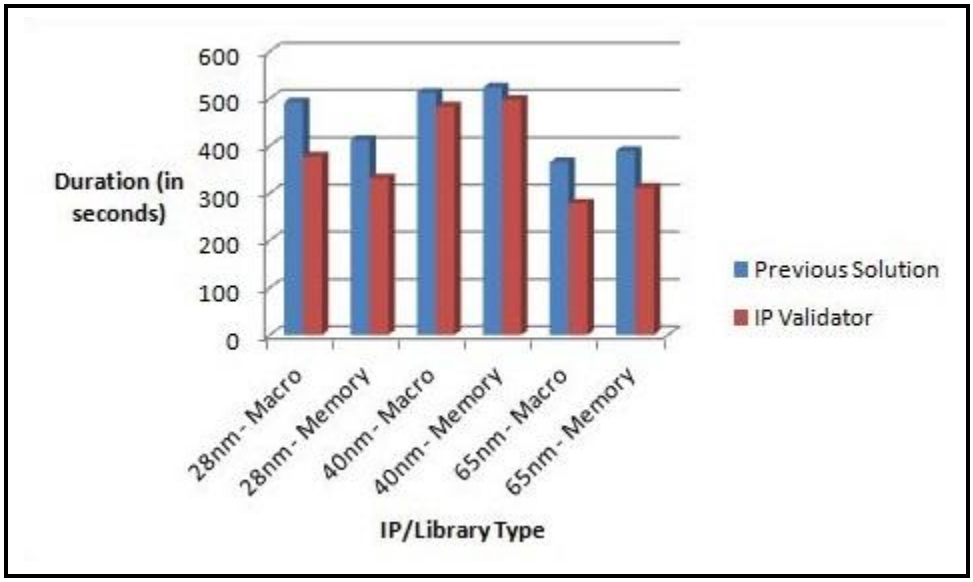


Figure-5.9 Timing comparison chart b/w Previous Solution & IPV (check-3)

As per the above results, the timing gain was average for inputs: 28nm Macro, 28nm Memory, 65nm Macro and 65nm Memory. For the rest of the inputs the gain was negligible.

Chapter-6

Conclusion and Future Scope

6.1 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 IPV.

Earlier, a considerable number of transitional files were utilized for the execution of checks. However, with IPV all these transitional files have been replaced by storing the required information in data structures. Using this upgraded validation solution, the designer gets ample amount of freedom. The designer has the ability to validate even single view. The new solution, comes with the ability to generate various reports like based on cells checked, or views or checks executed. The designer is provided with a graphical user interface (GUI) so that he or she is able to configure each and every check as per the requirement. The execution time of checks required for validation of an IP or library with the new solution relies on upon the parameter configuration of required checks.

6.2 Future Scope

Currently, this solution is supporting only one IP or library at a time. However, this solution can be easily extended in future to deal with more than one IP at a particular time. The checks can be further classified 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.

References

[1] 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

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

[3]: Raghudeep Kannavara, "Towards a Unified Framework for Pre-Silicon Validation", 2013

[4] STMicroElectronics Internal Document , Library Views

[5] J. Dreesen, "Standard cell development flow," in Euro ASIC, 1990.

[6] J. W. Chong and R. G. Forbes, "Design of basic CMOS cell library," IEEE Processings CHAE Jung Kyu – Doctoral thesis - 2014 105 G Circuits, Devices and Systems, vol. 139, no. 2, pp. 256-260, Apr. 1992.

[7] A. B. Jambeck, A. R. Mohd Noor Beg and M. R. Ahmad, "Standard cell library development," in International Conference on Microelectronics, 1999.

[8] J. D. Djigbenou, T. V. Nguyen, C. W. Ren and D. S. Ha, "Development of TSMC 0.25μm standard cell library," in IEEE SoutheastCon, 2007.

[9] D. Bekiaris, A. Papanikolaou, G. Stamelos, D. Soudris, G. Economakos and K. Pekmestzi, "A standard-cell library suite for deep-deep sub-micron CMOS technologies," in International Conference on Design & Technology of Integrated Systems in Nanoscale Era, 2011.

[10] J. K. Yuan, "Key to a successful cell library development: design methodology," in IEEE ASIC Seminar and Exhibit, 1989.

[11] "Liberty User Guides and Reference Manual Suite Version 2012.06," [Online]. Available: http://www.opensourceliberty.org.

[12] J. P. Moreau, J. Borel and D. Samani, "European trends in library development," IEEE Mirco, vol. 12, no. 4, pp. 43-53, Aug. 1992.

[13] L. W. Wang and H. W. Luo, "Automated IP quality qualification for efficient system-on-chip design," in *International Conference on Electronic Packaging Technology & High Density Packaging*, 2012.

[14] R. B. Lin, I. H. Chou and C. M. Tsai, "Benchmark circuits improve the quality of a standard cell library," in *Asia and South Pacific Design Automation Conference*, 1999.

[15] Design And Reuse. (2017). IP Verification : What are the main challenges to successful IP integration?. [online] Available at: https://www.design-reuse.com/articles/3249/ip-verification-what-are-the-main-challenges-to-successful-ip-integration.html

[16] AnySilicon. (2017). Verification, Validation, Testing of ASIC/SOC designs - What are the differences? - AnySilicon. [online] Available at: http://anysilicon.com/verification-validation-testing-asicsoc-designs-differences/

[17] EETimes. (2017). Why We Don't Have IP Quality Yet | EE Times. [online] Available http://www.eetimes.com/author.asp?section_id=36&doc_id=1265541

[18] Bernard Laurent and Thierry Karger, "A System to Validate and Certify Soft and Hard IP", 2003 Design, Automation and Test in Europe Conference and Exhibition, Pages: 208 - 213 suppl., DOI: 10.1109/DATE.2003.1253830

[19] Vörg, A. and Rosenstiel, W. (2004). Automation of IP qualification and IP exchange. *Integration, the VLSI Journal*, 37(4), pp.323-352.

[20] Mark Birnbaum and Charlene C. Johnson, "VSIA Quality Metrics for IP and SoC", Proceedings of the IEEE 2001. 2nd International Symposium on Quality Electronic Design, Pages: 279 - 283, DOI: 10.1109/ISQED.2001.915243

[21] ZHOU Meng, Gao Minglun1 and Philip C H CHAN, "A Practical IP Quality Measurement Framework", 2007 7th International Conference on ASIC, Pages: 1313 - 1316, DOI: 10.1109/ICASIC.2007.4415878

[22] R. Seepold, N. Martínez Madrid, A. Vorg, W. Rosenstiel, M. Radetzki, P. Neumann and J. Haase, "A Qualification Platform for Design Reuse", Proceedings of Design, Automation and Test in Europe Conference and Exhibition 2012

[23] Kathy Werner, "Can IP Quality be Objectively Measured?", Proceedings Design, Automation and Test in Europe Conference and Exhibition, Pages: 330 - 331 Vol.3, DOI: 10.1109/DATE.2004.1269264

[24] H. J. Brand, S. Rulke and M. Radetzki, "IPQ: IP qualification for efficient system design," *International Symposium on Signals, Circuits and Systems. Proceedings, SCS 2003. (Cat. No.03EX720)*, 2004, pp. 478-482.

[25] A. Vorg, M. Radetzki and W. Rosenstiel, "Measurement of IP qualification costs and benefits," *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, 2004, pp. 996-1001 Vol.2.

[26] Xiaolong Guo, R. G. Dutta, Yier Jin, F. Farahmandi and P. Mishra, "Pre-silicon security verification and validation: A formal perspective," *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, 2015, pp. 1-6.

List of Publications

[1] Anshul Aneja, Krunal Patanwadia and Jyoti Kumar, "IP CAD Validation Solution Using Crossfire", in ST Techweek Conference (2017).
[Communicated]

Video URL

https://youtu.be/MsZIAq9TLO8

Thesis File

ORIGINALITY REPORT

%2	%2	%1	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.wikiteka.com Internet Source	%1
2	Kannavara, Raghudeep. "Towards a unified framework for pre-silicon validation", IISA 2013, 2013. Publication	%1
3	www.ukessays.com Internet Source	<%1
4	cieonline.co.uk Internet Source	<%1

EXCLUDE QUOTES ON

EXCLUDE BIBLIOGRAPHY ON

EXCLUDE MATCHES < 6 WORDS

(ANSHUL ANEJA)

22/June/2017

(DR. SEEMA BAWA)

Appendix-A

Comparison of Various IP Validation Tools

The layered approach used in the new solution named as IPV (IP Validator) utilizes the Crossfire validation tool which is a third party component. The Crossfire tool can be replaced by any other tool as well. But the tool should satisfy a specific set of requirements to fit in as one of the layer of the IPV. A detailed analysis was performed before Crossfire was selected as the final choice. The table below represents the detailed analysis of the various validation tools on a specific set of parameters:

Parameter	Crossfire	Validation Tool 2	Validation Tool 3
Automated setup creation	✓	✓	✓
Integration of checks based on Third party EDA tool based in cross-fire	✓	✗	✓
Integration of ST specific checks as is	✓	✗	✗
Integration of ST specific checks using tool's internal data structure	✓	✗	✗
Integration of checks for ST specific views	✓	✗	✗
Parameterization of checks per IP profile	✗	✓	✗
Parameterization of checks per Technology	✗	✓	✓
Addition of Message codes(Error/Warn/Info)	✓	✗	✓
Addition of new checks	✓	✗	✓
Integrating ST Custom checks Reports with Validation Tool's Reports	✓	✗	✓
Report in text/CSV/HTML	✓	✓	✓
Granular reporting	✓	✓	✗
Report comparison with previous reports	✗	✓	✗
Re-run any single check	✓	✓	✓

By clicking on the error in report, tool should take directly to the failing view	✓	✗	✓
Configurable within existing checks i.e. Allowing user to add or delete checks	✓	✗	✗
Single validation tool should give flexibility to user to validate either Taxonomy based or view based	✗	✗	✗
Checks with multiple level of granularity can be run in Unix (command line)	✓	✓	✗
CLI equivalent mode in GUI supported at multi level of granularity	✓	✓	✗

Appendix-B

Crossfire - as IP Validation Solution

1. About Crossfire

Crossfire reports confounds or displaying mistakes for Libraries and IP that can truly defer an IC configuration extend.
Library and IP respectability checking has turned into an obligatory stride for a "best in class" profound submicron plan because of the accompanying difficulties:

- Time-to-market
- Large number of different views
- Complexity of different views
- Loss of design time

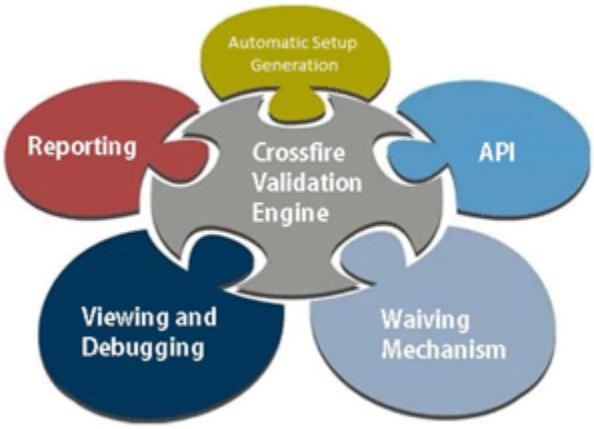


Figure: Crossfire Validation Schema

Crossfire helps CAD groups and IC creators accomplishing a high caliber of plan information in a brief timeframe. Crossfire guarantees that the data spoken to over the different perspectives is steady and does not contain irregularities.

2. Crossfire Usability Features

- Powerful progressive setup dialect supporting full scale capacities
- GUI based debugging mechanism
- Graphical yield sifting (zoom in on cells/designs)
- False error suppression mechanism
- CSV and HTML reports
- Automatic setup creation

- Rich set of APIs available in different programming languages
- Generic setups
- Parallel parsing of large number of views
- Parallel execution of checks

3. Crossfire Integration Features

Programming interface for making database autonomous checks, accessible in: Perl, TCL and Python. Existing client approval scripts can be coordinated. Visualization results from client scripts (double tap opens message).

4. Crossfire Interview

Crossfire tool provides a GUI named interview. This GUI can be used to see the view database contents. This view database is generated once the parsing of all the available views is done by the Crossfire tool. The designer can see the library name for that particular view. Also, the names of the cells available inside that view. The designer also gets the ability to see the pin information regarding that particular view.

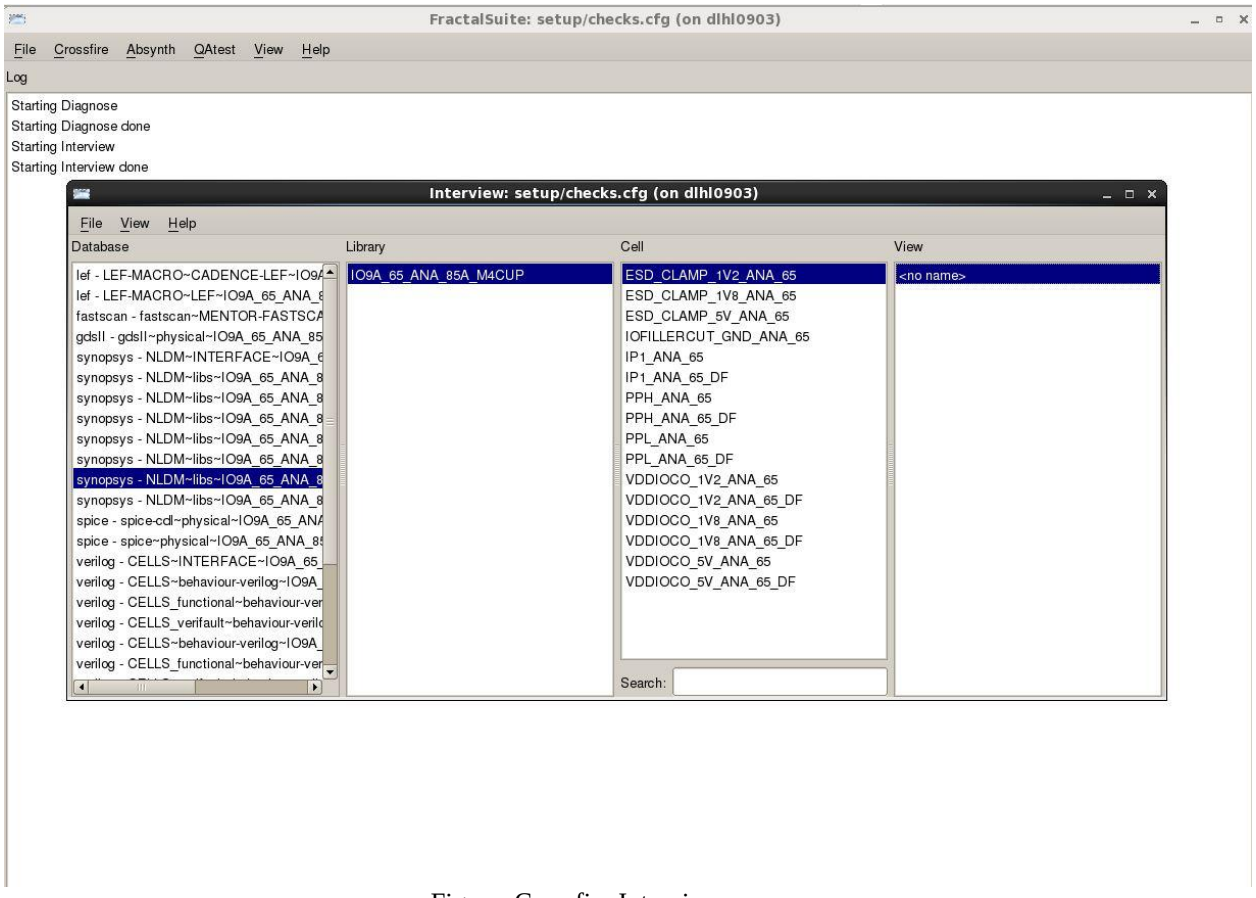


Figure: Crossfire Interview

5. Crossfire Diagnose

Diagnose is the Crossfire GUI intended for clients that desire to just break down Crossfire results. The setup and test definition areas of Crossfire are totally protected from the client. The client can see, report, channel, defer and investigate the produced Crossfire messages.

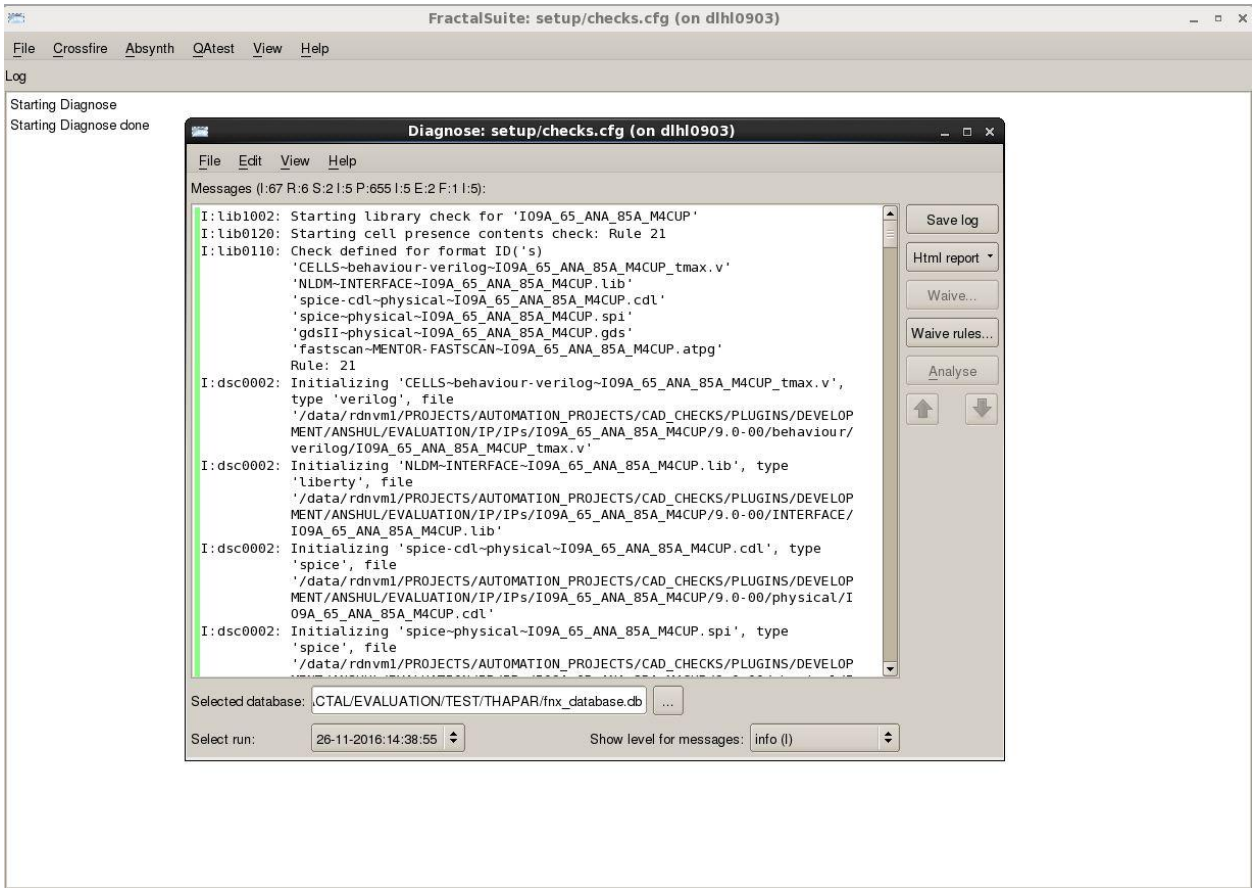


Figure: Crossfire Diagnose