Front end Tool Contour for Quality Assurance Regression System for IP Teams

Ms. Rani T N, Dr. Kiran Bailey
Student, M. Tech in VLSI Design and Embedded Systems, BMS College of Engineering, Bengaluru, India
Assistant Professor, Dept of Electronics and Communication Engineering, BMS College of Engineering, Bengaluru, India

Abstract: Increase in size and complexity of today's IP (Intellectual Property) have intensified the challenges of design and verification. Meeting these challenges requires advanced technologies and methodologies that ensure the highest design quality. Design Automation are the software tools and methods employed to aid in chip design. This paper describes the integration of few Front end Design Tools (Linting, CDC Checker, Power Management) and Verification/Validation Tools (Simulating Tool) for RTL quality checks in a common Front end simulation environment and running each of these tools on the design considered through the simulation environment. Each of these tools were run through the Front end simulation environment on the design considered and the reported results were analyzed, debugged and the designs quality is improved. Using many advanced algorithms and analysis techniques the DA platform provides designers with insight about the design, early in the process at RTL thus allowing designers to find bugs quickly and easily, significantly improving the quality of the most complex designs and enabling first-pass silicon success.

Keywords: Clock Domain Crossing, Design Automation, Linting, Unified Power Format.

I. INTRODUCTION

RTL is expressed usually in Verilog/VHDL. RTL Quality checks, Functional/Logical Verification is performed at this stage to ensure the RTL designed is clean and matches the idea. Creation and Validation of RTL includes writing, debug, HDL translation, lint and validation of the RTL etc. Taking advantage of advanced techniques and rapidly deploying them across project teams is essential. Platform tools expertise and applying best practices based on proven methodologies is required. Verification enables using pre-defined and pre-verified building block, which can effectively reduce the productivity gap. It can be block based design approach or platform based design approach [1].

Design Automation is a category of software tools for designing electronic systems such as integrated circuits. The tools work together in a design flow that chip designers use to design and analyze entire semiconductor chips. Since a modern semiconductor chip can have billions of components, EDA tools are essential for their design [5].

II. METHODOLOGY

RTL design considered here for the purpose of analysis through design automation is 32 bit booth multiplier which contains several blocks such as general purpose registers, adders, multipliers etc. RTL static checks such as Linting, CDC and Low Power checks, and then dynamic check by simulating the design using Simulating tool were performed [3]. The tools used to perform these checks were not run standalone on the design but are run through the front end simulation environment, in which all these tools are integrated. The results were then analyzed and the RTL quality was improved.

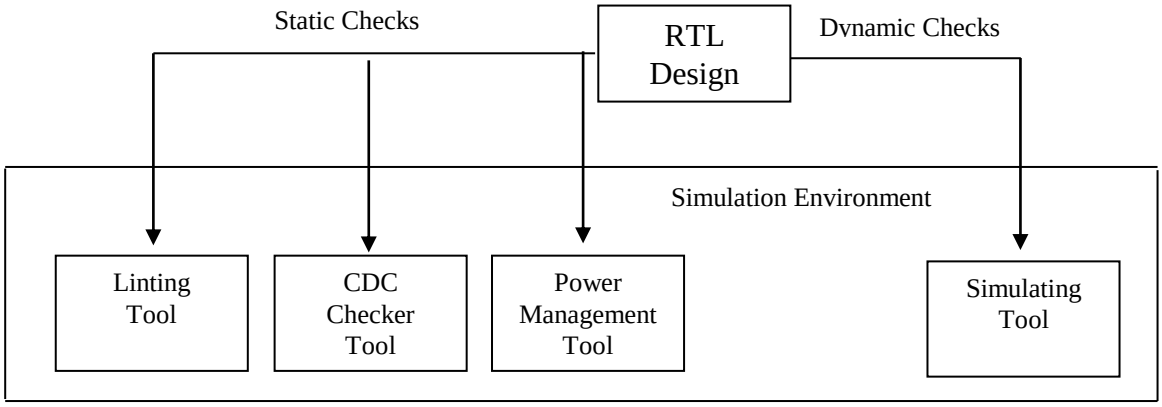

Fig 1: Checks performed on the Design

A. Static Checks

These are the checks performed on the RTL without providing any of the inputs. Checks done include **Linting** which checks HDL designs against various coding standards and design rules to check the designs for errors, **Clock Domain Crossing (CDC)** checks analyses the RTL for synchronization violations, **Low Power Checks** analyzes RTL and UPF against a set of rules. **UPF (Unified Power Format)** is a standard set of Tcl-like commands used to specify the power intent of a design.

B. Dynamic Checks

These are the checks performed on the RTL by providing inputs to the design. The behavior of the design is analyzed by simulating the design after providing the required inputs using simulation tool which is also used to debug the design.

C. Front end Simulation Environment

Front end Simulation Environment is designed for use to compile the design, build, run tests, regressions and also debug the design. It also determines, summarizes and reports tests results and also perform test run directory clean up. It acts as a wrapper developed around the tools to aid and ease the tool flow and analyze it's results.

III. RTL DESIGN AND VERIFICATION TOOLS

A. Linting Tool

Linting tool is a design checker tool that comes with prepackaged rules to check Verilog or System Verilog designs against various coding standards and design rules to check the designs for errors that may cause problems in the downstream simulation, synthesis, and equivalence checking flows. The tool provides rules for specific domains including CDC, Multi-Power Planes (MPP), Security Domain Checks (SDC) and others. Users can also implement User Defined Rules (UDRs).

Linting is a two step flow which includes **analyse** stage in which compilation of list of HDL files are done and stored them as library on the disk and then in the **elaborate** stage the tool loads all the libraries and additional design collateral such as UPF (Unified Power Format) files, collects all the violations generated during the creation of the loaded libraries, and performs all the checks on the full design an reports violations.

The tool loads UDR's, parses the configured files and waiver files, runs the tool and checks the UDR's, and generates reports as files, which can even be viewed through GUI. Linting rules can either be Built in or User Defined. Built in Rules are the warning's generated from the RTL compiler which

covers language rules, synthesis rules and coding style. User Defined Rules depends on the project requirements and covers coding style, connectivity, naming convention, macro rules, SVA etc. The violation of these rules are flagged by n-digit rule IDs.

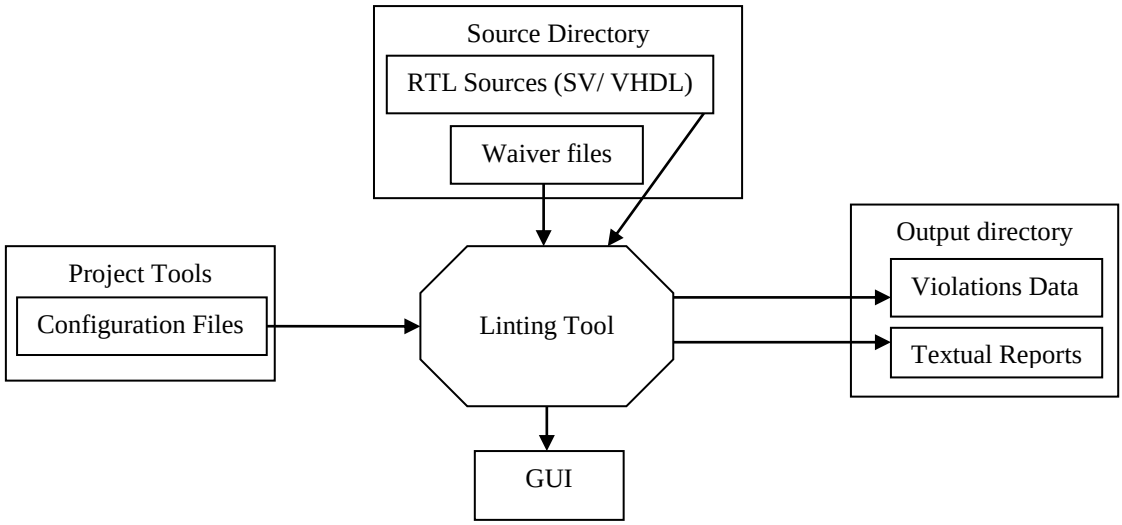

Fig 2: Lint Methodology

B. Clock Domain Crossing Checker Tool

In digital electronic design CDC, is the traversal of a signal in a synchronous digital circuit from one clock domain into another. Different clock domains have clocks which have a different frequency, a different phase or both. Either way the relationship between the clock edges in the two domains cannot be relied upon.

CDC analysis is generally divided into two parts. Structural analysis and Functional analysis. **Structural analysis** is mainly but not limited to automatic identification of clock domains, recognition of synchronizers in the RTL design. **Functional analysis** is for verifying any re-convergence and data correlation issues.

CDC verification is important because it can catch missing synchronization, re convergence issues and can detect metastability issues, to name a few. These checks are done on RTL. Using Prime Time based scripts one can achieve similar functionality by doing clock domain crossing checks at Gate Level netlist, which is late in the design cycle. By running these CDC checks early in the design cycle, we can improve the overall RTL to GDSII flow.

C. Power Management Tool

The ability to address potential low power design issues early in the design cycle is critical to achieve high productivity in the design process. We get more optimized designs as we address these issues during the RTL code development.

The tool analyzes and provides RTL guiding towards lower power implementation by provides low power rules and design techniques to address multiple voltage domains, and power-down regions that may be present in a typical low power design. Supports power intent written in UPF, CPF, and SGDC formats. The IEEE 1801 Standard for Design and Verification of Low Power Integrated Circuits, also known as the Unified Power Format (UPF), consists of a set of commands used to specify the design intent for multi-voltage electronic systems. Using the UPF commands, you can specify the supply network, switches, isolation, retention, and other aspects pertinent to the power management of the

design. This single set of commands can be used for verification, analysis, and implementation of your design [4].

The power elements that are used to specify the design intent for low power designs are divided into the following sections: **Power Architectural Elements** which include Library Requirements for Low Power Designs such **as** Level-Shifter Cells, Isolation Cells, Power Switch Cells, Always-On Logic Cells, Retention Register Cells. Secondly it's **Power Distribution Elements** which include Supply Ports, Supply Nets, Power Switch. Lastly it's **Power State Tables (PST)** which lists the allowed combinations of voltage values and states of the power switches for all power domains in the design. It defines the legal combination of states that can exist simultaneously during the operation of the design. The methodology remains same for the static checker tools, with change of configuration files.

D. Simulating Tool

Simulating tool specifically designed to simulate and debug designs. There are three main steps in debugging the design, which are. 1. Compiling the Verilog/VHDL source code. 2. Running the Simulation. 3. Viewing and debugging the generated waveforms. The tool first compiles the verilog source code into object files, which are nothing but C source files. Simulating tool can compile the source code into the object files without generating assembly language files. It then invokes a C compiler to create an executable file. We use this executable file to simulate the design. One can use the command line to execute the binary file which creates the waveform file.

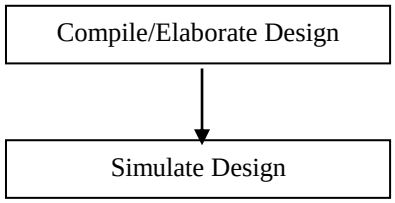

Fig 3: Two step simulation flow

IV. FRONT END SIMULATION ENVIRONMENT

The environment allows all the front end checks to be done. Front end checks including static checks, run tests and debug them. It provides all the requirements required to tape an RTL. It provides complete front end design kit which includes tools, environments, utilities, licences, required for an RTL design, validation and take the collateral across IP teams and organizations. The environment also emphasizes re-use: re-use of RTL, of the environment configuration, and of tests. As such, it is particularly well-suited for SoC designs, where re-use is paramount.

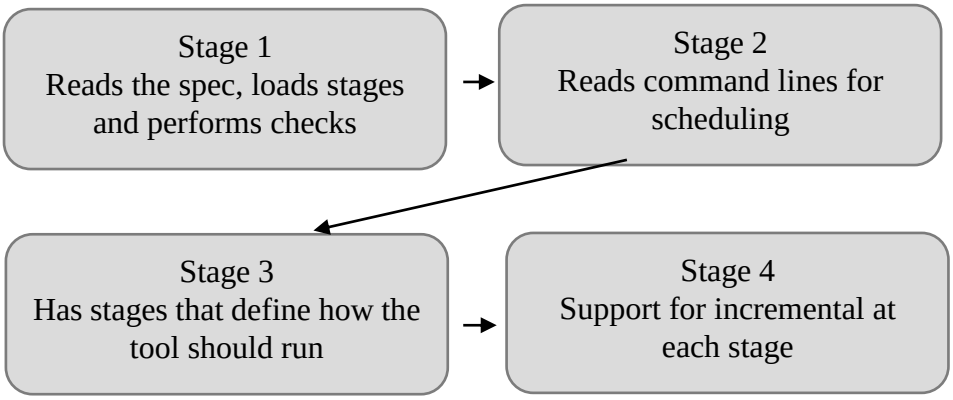

Fig 4: Stages involved in a front end simulation environment to run the tools

V. RESULTS

After Linting the design through the Front end Simulation Environment, the violations and warnings obtained for the Verilog design ChipTop.v is as shown in Fig 5.
The violations generated are in .xml format, which can viewed through spreadsheet and be dumped to GUI and then debugged.

```
/ChipTop_violations.xml
Lint status FAILED
Violations were found:
severity: Error , count: 116 violations , waived: 0, not waived: 116
severity: Warning , count: 158 violations , waived: 0, not waived: 158
```

Fig 5: Lint Status reporting Violations and Warnings for the design ChipTop.v

In fig 6, As the design had only one clock, there are no CDC violations. Minimum to two clock domains are required to report CDC violations. The report gives the complete summary of the clocks used in the design.

```
Clock Group Summary for 'ChipTop'
=================================
Total Number of Clock Groups        : 1
 1. User-Specified                  :(0)
 2. Inferred                        :(1)
    2.1 Primary                     : 1
    2.2 Undriven                    : 0
    2.3 Blackbox                    : 0
    2.4 Gated Mux                   : 0
    2.5 Gated Combo                 : 0
 3. Ignored                         :(0)
```

Fig 6: CDC Violations Summary for the Design ChipTop.v

Fig 7, reports the violation of the UPF design rule while designing UPF. The violation is that declared port is not assigned as state.

```
24
25  # VSS
26  create_supply_port VSS
27  create_supply_net  VSS   -domain TOP
28  create_supply_net  VSS   -domain INST -reuse
29  create_supply_net  VSS   -domain GPRS -reuse
30  create_supply_net  VSS   -domain MULT -reuse
31  create_supply_net  VSS   -domain GENPP -reuse
32  connect_supply_net VSS   -ports VSS
33
```

```
# ADD PORT STATE INFO
#####################

add_port_state VSS    -state {OFF 0}

add_port_state VDD    -state {HV  0.90}

add_port_state VDDI   -state {HV  0.90}\
                      -state {LV  0.75}

add_port_state inst_sw/out -state {HV  0.90}\
                           -state {LV  0.75}\
                           -state {OFF off}
```

Fig 7: UPF viewed through GUI to view the error (left). Debugging of the UPF error by invoking the editor (right)

Fig 8 shows the waveform of the design after simulating it by providing proper inputs

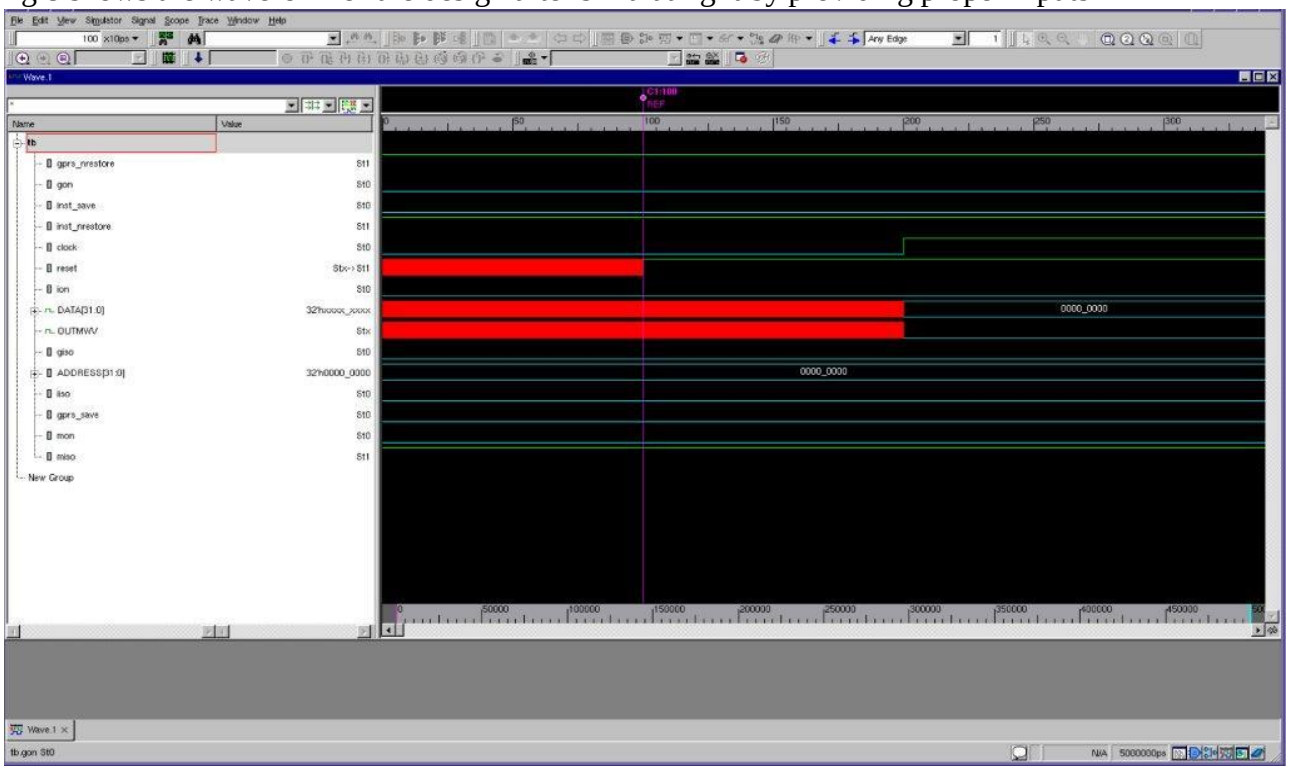

Fig 8: Waveform view of the Simulated Design viewed through GUI

VI. CONCLUSION

The field of applications enabled by semiconductor technology is growing at a rapid rate, ranging from very high-performance microprocessors and signal processors, to a broad array of low-power portable devices, to micro sense/communicate/actuate networks of chips that are driven by very low per-unit cost and extremely low operating power. Meeting these challenges requires advanced technologies and methodologies that ensure the highest design quality is important.

ACKNOWLEDGMENT

The satisfaction of the successful completion of the task would be incomplete without mentioning all those whose guidance, encouragement and cooperation worth the effort with success. I would like to take this opportunity to thank them all. I feel myself honored to place my warm salutation to **Bms College of Engineering.** I hereby express my profound gratitude to my guide **Dr. Kiran Bailey** for constant advice. Lastly, I thank my parents for their co-operation and helping me financially and morally during the course. I also thank my friends and one and all who helped me directly or indirectly for the successful completion of the work.

REFERENCES

[1]https://www.testandverification.com/DVClub/19_Nov_2012/5%20%20Broadcom%20-%20Geoff%20Barrett%20(Challenge).pdf

[2] 'ASIC Design Flow Tutorial Using Synopsys Tools' by Hima Bindu Kommuru, Hamid Mahmoodi- Nano-Electronics & Computing Research Lab ,School of Engineering ,San Francisco State University ,San Francisco, CA ,Spring 2009

[3] 'Synopsis Compilation', User Guide, L-2016.06-SP1, September 2016.

[4] Synopsys Low Power Verification Tools Suite' User Guide, Version K-2015.09-SP1, December 2015

[5] https://en.wikipedia.org/wiki/Electronic_design_automation

Rani T N Student, M. Tech in VLSI Design and Embedded Systems, BMS College of Engineering, Bengaluru, India

Dr. Kiran Bailey Assistant Professor, Dept of Electronics and Communication Engineering, BMS College of Engineering, Bengaluru, India