
Testchip Validation Automation

Major Project Report

Submitted in partial fulfillment of the requirements

For the Degree of

Master in Technology

In

Electronics and Communication Engineering

(Communication Engineering)

by

Swati Sharma

(14MECC26)



Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2016

Testchip Validation Automation

Major Project Report

Submitted in partial fulfillment of the requirements

For the Degree of

Master in Technology

In

Electronics and Communication Engineering

(Communication Engineering)

by

Swati Sharma

(14MECC26)



Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2016

Declaration

This is to certify that

1. The thesis comprises my original work towards the degree of Master of Technology in Communication Engineering at Nirma University and has not been submitted elsewhere for a degree.
2. Due acknowledgement has been made in the text to all other material used.

Swati Sharma
(14MECC26)

Disclaimer

“The content of this thesis does not represent the technology, opinions, beliefs, or positions of STMicroelectronics Pvt. Ltd., its employees, vendors, customers, or associates.”



Certificate

This is to certify that the Major Project entitled “**Testchip validation automation**” submitted by **Swati Sharma(14MECC26)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Communication Engineering, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.

External Project Guide:

Mr. Sanjiv Bansal

Group Manager

STMicroelectronics India Limited,
Greater Noida.



Certificate

This is to certify that the Major Project entitled "Test chip pattern validation automation tool" and "Memory block integration and assembly automation tool" submitted by Swati Sharma (14MECC26), towards the partial fulfillment of the requirements for the degree of Master of Technology in Communication Engineering of Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our 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 our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date:

Place: Ahmedabad

Guide

Program Coordinator

Dr Sachin Gajjar

Dr D. K. Kothari

Associate Professor, EC

Prof, EC

Director

Dr P. N. Tekwani

Head of Electrical and Electronics Department

Abstract

As the technology tends to continuously shrink from small scale integration (SSI) to very large scale integration (VLSI), design for testability is also included more seriously into the ASIC flow. With increase in complex system, testability is an increasing concern in almost every application and in every area of application development. Test engineers put more efforts in addressing the issue of testability at the device, board, and system levels deliver more consistently reliable and cost effective products to the market place. This means building test capabilities in every phase of development and deployment, including design verification, hardware and software integration. Also, ICs are made through block production, therefore, it is necessary to test them before actual manufacturing so as to reduce cost of manufacturing.

The aim of test pattern validation tool is to reduce total cycle time of test cases validation. Pattern validation environment is useful to launch various test cases in order to verify characteristics of Testchip SoC which consists of subsystems like memory block, standard cell block. For reliable and efficient testing, the tool needs to work fast and for every possible testcase, without fail and it is the utmost priority of the design engineer.

Standard cell block design tool is used to test the standard cell libraries for different types of cells for their functionality and delays in various possible PVT (process, voltage and temperature) corners. The libraries that need same inputs from input are better to be combined so that the tool can work faster. This control, whether to combine libraries or not, should be with the user or tester. This feature adds on to the efficiency of the tool.

There has been sufficient rise in the number of critical paths in digital designs due to gradually shrinking CMOS. Conventional techniques for evaluation of digital design have reduced the gap between CAD and Si design. But these technologies are restrained to limited process, voltage and temperature (PVT) corners and consumes a lot of time. The solution is presented by the CAD correction factor tool.

Distribution request tracker reporting tool is used to get the details of the products that has been

requested earlier from the database, through web service. This reporting from the database should be as fast as possible. Therefore, there was a need for optimization of the tool to make it work faster and accurate.

Acknowledgement

I would like to express my gratitude and sincere thanks to Dr. P.N. Tekwani, Head of Electrical Engineering Department and Dr. D. K. Kothari Coordinator M.Tech Communication Engineering program for allowing me to undertake this thesis work and for his guidelines during the review process.

I am deeply indebted to my external guide group manager Sanjiv Bansal and internal guide Dr. Sachin Gajjar for their constant guidance and motivation. I would also like to thank line manager Alok Mishra for his help and support. Without their experience and insights, it would have been very difficult to do quality work.

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

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

-Swati Sharma

14MECC26

Contents

Declaration	i
Disclaimer	ii
Certificate	iii
Certificate	iv
Abstract	v
Acknowledgement	vii
List of Figures	xii
Abbreviations	1
Abbreviation Notation and Nomenclature	1
1 Introduction	2
1.1 Background	2
1.2 Motivation	3

1.3	Objective	3
1.4	Problem Statement	4
1.5	Organisation of the Thesis	4
2	Test Pattern Validation Tool	5
2.1	Principle Involved	5
2.1.1	Elmore Delay Model [3]	6
2.2	Working of the tool	8
2.2.1	Input files	8
2.2.2	Output files	13
3	Automation	14
3.1	Need for automation	14
3.2	Load Sharing Facility (LSF)[1]	15
3.3	LSFKit[2]	21
4	Standard Cell Block Design Tool	23
4.1	Introduction	23
4.2	Standard Cell Libraries	24
4.3	Structure and Working	27
4.3.1	ALLCELL	29
4.3.2	FDD	30
4.3.3	Retention Block	30

4.4	Work done	31
4.4.1	Dependency of the execution of tool on home area of user	31
4.4.2	To combine libraries to increase efficiency	34
5	CAD Characterization Flow (CCF) Tool	36
5.1	Introduction	36
5.1.1	Background	36
5.1.2	Motivation	37
5.2	CAD Characterization Flow Tool	37
5.2.1	Built in self Characterizer (BISC) Module	39
5.2.2	Working	40
5.3	Work done	42
6	Distribution Record tracker (DRT) reporting Feature	45
6.1	Introduction	45
6.1.1	Submitted	46
6.1.2	Assigned	46
6.1.3	Pending approval	46
6.1.4	Approved	46
6.1.5	Failed	47
6.1.6	Cancelled	47
6.2	DRT Reporting feature	47

6.3	Work done	48
6.4	Result	48
7	Conclusion and Future work	52
7.1	Conclusion	52
7.2	Future Work	53
	References	54

List of Figures

2.1	Simple lumped RC model of an interconnect line, where R and C represent the total line resistance and capacitance, respectively.	6
2.2	Distributed RC ladder network model consisting-of N equal segments	6
2.3	A general RC tree network consisting of several branches	7
2.4	Block diagram of test pattern validation tool	8
2.5	Block diagram for netlist generation	10
2.6	Performance, power and area trade-off in a testchip	11
2.7	Contents of the dump file	12
3.1	LSF Terminologies	16
3.2	Job life cycle	17
3.3	Queue usage model in LSF	18
3.4	Queue usage model in LSF	21
4.1	Example of look-up tables in a Standard cell library	26
4.2	Block diagram of test circuit architecture	28
4.3	Basic circuitry for cells functionality testing	29

4.4	ALLCELL library block	30
4.5	Flow diagram to show working of tool to combine libraries	35
5.1	Setup for BISC module	39
5.2	Waveforms at different stages of the access time measurement setup	41
5.3	CCF Tool Flow	43
5.4	Flow diagram for CCF wrapper	44
6.1	DRT reporting user interface	49
6.2	Flow diagram showing fetching of request details	50
6.3	Result for all the customer projects starting with word 'test'	51

Abbreviation Notation and Nomenclature

Soc	Silicon on Chip
IC	Integrated Chip
LSF	Load Sharing Facility
ASIC	Application Specific Integrated Circuit
FDSOI	Fully depleted Silicon on Insulator
CAD	computer-aided design
CCF	CAD Characterization Factor
RTL	Register Transfer Logic
.cir	Circuit Simulation File
.spi	Spice Parasitic Information
PNR	Placement and Route
PVT	Process, Voltage, Temoerature
VCD	Graphic data System
VCD	Value Change Dump
SDF	Standard Delay Format
CUT	Cell Under Test
DRT	Distribution Request Tracker

Chapter 1

Introduction

With the advancement of technology, the size of integrated circuits are decreasing at a very fast pace. The reduced size is causing challenges in testing and manufacturing the ICs. Therefore, there is a need for optimal validation tools so that the actual behavior of the IC doesn't deviate much from the expected behavior. Also, There is a need for automation of these tools because human errors can cause hinderance to the reliability of any tool.

1.1 Background

Test reuse is the factor be considered, without loosing the track of low-level information, both structural (netlist) and physical (layout). To verify logic designs, one of the major approaches is simulation, which requires the generation and application of input patterns (tests) that check for correct behavior. Simulation is the most widely used and practical verification technique for large logic circuits. Exhaustive simulation using all possible input combinations as tests is one possibility, at least for combinational logic, but the number of tests needed for adequate coverage of design errors tends to be excessive (exponential).

Automation can itself solve many problems while testing ICs. These may be avoiding human errors, making more reliable products, faster testing and lesser human efforts.

1.2 Motivation

The first tool, on which I worked, is a test pattern validation tool. This tool is used to check the testchips for functional and timing qualification. Tool matches the output of the test patterns with pre-defined outputs and gives whether tests has been passed or failed. With new test patterns with large memory usage and CPU utilization, automation tool that has been currently used is facing some problems, now these issues need to be addressed.

Another tool is standard cell block design tool. It is used to check the functionality of standard cell libraries with different combinational, sequential and tristate buffer cells. Issue with this tool comes very rarely when the home area of the user is full. This obstructs the generation of output files even in the project area. This needs to be addressed for seamless functioning of the tool. Also, to increase the performance of the tool, standard cell libraries need to be combined.

And CAD correction factor (CCF) tool is used to calculate memory access time mismatch between CAD design and the fabricated Si design which is introduced because of measurement system used to calculate access time of a memory. There were many loopholes in the design of this tool which were not letting the tool work properly for each and every technology. These were needed to be resolved for making the tool reliable. Also, there was need of a wrapper for the tool to make tool work independently, with minimal human interference and making it automatic in true sense.

1.3 Objective

- To remove dependency of test pattern validation tool from LSFKit and launch test patterns directly on load sharing facility (LSF).[1]
- To remove the dependency of the standard cell block design tool from the home area of the user even when tool is running from the project area.
- To combine the libraries in standard cell block design tool into a single one for better efficiency of the tool
- To remove bugs from CAD characterization (CCF) tool.
- To create wrapper for CCF tool.

1.4 Problem Statement

Failure of test pattern validation tool because of unknown reasons while firing test patterns on LSFkit needs to be worked upon and this is done using removing LSFKit from the tool.

Also, Standard cell block design tool doesn't create required files in project work area because of insufficient space in the home area of the user. This was caused because LSFkit creates a temporary file called .lsfkit in the home area which doesn't able to generate due to insufficient space in home area. This problem was resolved by removing the dependency of standard cell block design tool on the LSFKit. Alongwith that, for tool to work faster, there is need to combine standard cell libraries according to their reference libraries and the sum of combinational cells present in the libraries to be combined.

1.5 Organisation of the Thesis

In this thesis, first chapter gives the brief introduction of all the tools and tasks that I have accomplished during my internship. Second chapter describes testchip pattern validation tool called Squirrel Pattern, which is used to launch various test cases in order to verify characteristics of testchips. Third chapter provides information about load sharing facility (LSF), which is a network of distributed computers organized into a seamless system, to which multiple jobs can be submitted. It is used for run testing tools like Squirrel Pattern to work faster and efficiently. In the next chapter, Standard Cell Block Design tool is explained. It is a tool to test standard cell libraries for functional testing of standard cells and leakage measurement of these libraries on the whole. Fifth chapter describes CAD characterization factor tool which is used to find the difference between the CAD and SPICE (silicon fabricated) memory. The chapter also describes BISC module. Sixth chapter gives details about Distribution Request Tracker (DRT) reporting tool which is used to fetch information about DRT requests from database using Java and SQL. Last chapter concludes the work done and the future work.

Chapter 2

Test Pattern Validation Tool

Pattern validation environment is useful to launch various test cases in order to verify characteristics of Testchip SoC which consists of subsystems like memory block, standard cell block. It is a platform where N number of test cases can be fired in parallel on N number of valid conditions with a number of valid input options. It will create a status report file which will contain information regarding all simulated test cases. The aim is to reduce total cycle time of test cases validation.

The tool helps in achieving functional and timing qualification of the testchip. For functional qualification, no delay of interconnects and gates are taken into account, only outputs are matched with pre-defined outputs. For delay qualification, Elmore Delay Model is used to calculate the estimated delay due to the interconnects.

2.1 Principle Involved

An interconnect can be disentangled into a lumped RC system if the duration of flight over the interconnect line is much shorter than duration for complete cycle of the signal that is, the time taken by the signal to set out starting with one point then onto the next is shorter than time period of the signal. The least complex model which can be considered to show the parasitic resistance and capacitance of the interconnect consists of one lumped resistance and one lumped capacitance which is shown in 2.1.

The transient behavior of an interconnect can be better represented using the RC ladder network,

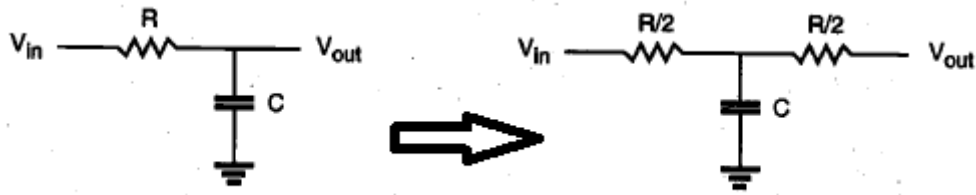


Figure 2.1: Simple lumped RC model of an interconnect line, where R and C represent the total line resistance and capacitance, respectively.

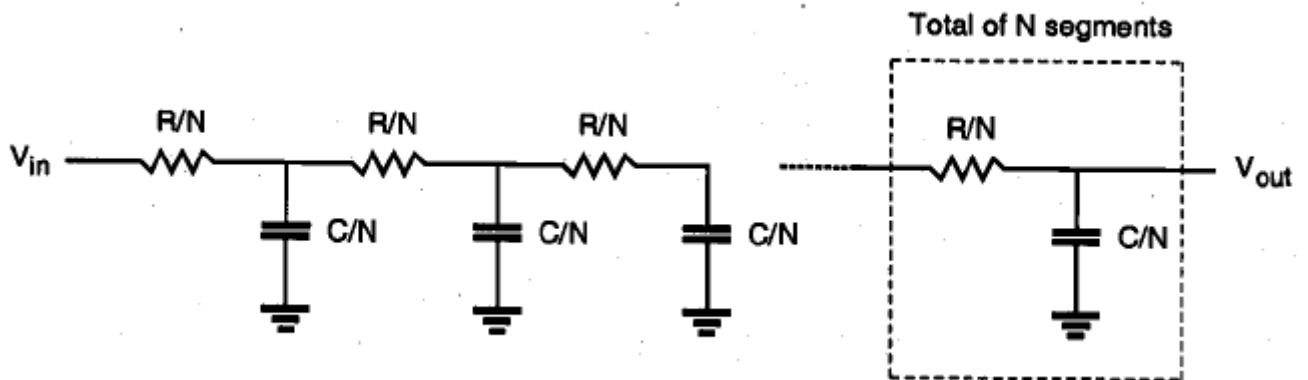


Figure 2.2: Distributed RC ladder network model consisting-of N equal segments

shown in 2.1. Here, each RC-segment consists of a series resistance (R/N), and a capacitance (C/N) in parallel, connected between the node and the ground. The accuracy of the model gets increased with increasing N, where the transient behavior tends to approach that of a distributed RC line.

2.1.1 Elmore Delay Model [3]

Consider an RC tree network, as shown in 2.1.1 Note that firstly, there are no resistor loops in the given circuit, secondly, all the capacitors in the RC tree are connected between a node and the ground, and lastly, there is only one input node in this circuit. Also, there is a particular resistive path, between the input node and every other node of the circuit. On observing the given RC tree network, we can consider the given path definitions :

- Let P_i denotes the unique path from the input node to any other node i , $i = 1, 2, 3, \dots, N$.

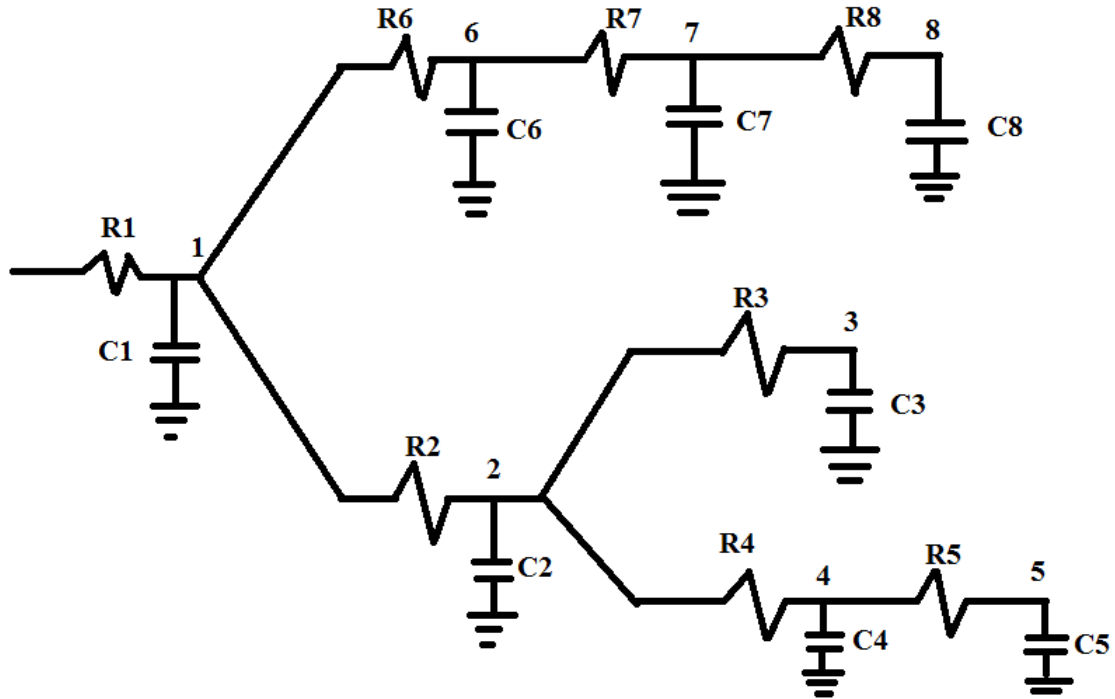


Figure 2.3: A general RC tree network consisting of several branches

- Let P_{ij} denotes that portion of the path between input and the node i , which is common to the path between the input and node j .

Assuming that the input signal is a step pulse at time $t = 0$, the Elmore delay at node i of this RC tree is given by the following expression.

$$\tau_{Di} = \sum_{j=1}^N C_j \sum_{k \in P_{ij}} R_k$$

Note that, although this delay is still a mere approximation for the actual signal propagation delay from the input node to node i , it provides a fairly simple and quite accurate means of predicting the behavior of the RC line. The procedure to calculate the delay at any node in the circuit is very straightforward.

For the circuit in 2.1.1, delay at node 7 to travel from input node can be calculated using the above formula as

$$\tau_{D7} = R_1C_1 + R_1C_2 + R_1C_3 + R_1C_4 + R_1C_5 + (R_1 + R_6)C_6 + (R_1 + R_6 + R_7)C_7 + (R_1 + R_6 + R_7)C_8$$

Similarly, for node 5, delay can be calculated as:

$$\tau_{D5} = R_1C_1 + (R_1 + R_2)C_2 + (R_1 + R_2)C_3 + (R_1 + R_2 + R_4)C_4 + (R_1 + R_2 + R_4 + R_5)C_5 + R_1C_6 + R_1C_7 + R_1C_8$$

In any case, there is a huge distinction between the transient reaction qualities of the simplistic lumped RC model and the more practical distributed RC ladder network model. The lumped RC representation of the interconnect line causes an overestimation of the propagation delay time. Elmore delay model has applicable only on one time-constant model, for more than one time constants, Asymptotic Wavelength Evaluation (AWE) is better to be used.

2.2 Working of the tool

2.2.1 Input files

For pattern validation of testchip, each test case needs compiled or non-compiled database. To be more specific RTLs (register transfer logic), NETLISTs and LIBRARYs are used in compiled format. Whereas, SDFs(Standard delay format file), VCD (value change dump files) and configuration file.

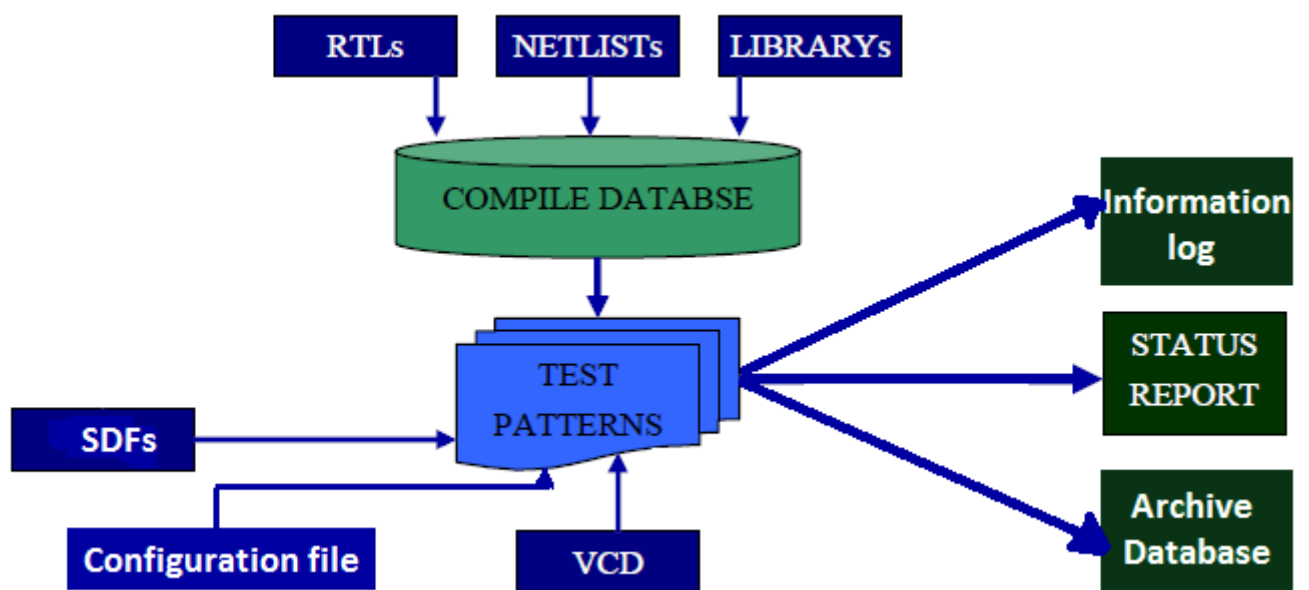


Figure 2.4: Block diagram of test pattern validation tool

1. **RTL:** RTL (register transfer logic) simulation simulates the code directly, so there is no timing information. There is no need to compile the code for RTL simulation. RTL supports only two languages, VHDL and Verilog (in modelsim). The simulation is pretty quick, because it is just a source code. In simple words, an RTL is the description of the hardware by the code that is only functional description, no delay are accounted for in RTL. This implies that the Verilog code describes how data is transformed as it passes from one register to another. The transforming of the data is performed by the combinational logic that exists between the registers.

RTL can be considered as a source, this is the logical or functional description of what should happen to the digital logic to perform the intended operations. In RTL, stimulus sources (inputs) and the timing boundaries (clocks) are specified. One can simulate the RTL code to validate logically if the design will work in the expected manner. To get from here to anything that can really run in an FPGA or ASIC, you have to "compile" (synthesize) that logical code.

Design created by the Register-Transfer Level specifies the characteristics of a circuit by operations and the transfer of data between the registers. An explicit clock is used. RTL design contains exact timing bounds, operations are scheduled to occur at certain times.

2. **Netlist:** A netlist is a textual portrayal of all segments of a circuit. Segments are usually gates, in this way, a Netlist is for the most part an association of gates. A netlist can likewise be an association of resistors, transistors or capacitors, which is a netlist that is utilized as a part of analog simulation tool as SPICE. A netlist is composed by hand, but usually it's the yield of a procedure called synthesis. For this situation the netlist then corresponds to a depiction of a configuration composed utilizing a hardware description language when composed as a part of a RTL style. The netlist is then expected to perform the same capacity as the corresponding HDL code.

The netlist made by the synthesis tool is then sent as an input to layout tools for delivering design of the chip. Amid this procedure, the netlist may get changed, yet practically it stays equivalent to its comparing HDL code. The netlist which is composed by the layout tool, after the layout has been made, is generally called post-format netlist. The distinction between the pre-design and the post-design netlist is the presence of 'clock tree buffers' in the post-design netlist.

Gate level simulation is a simulation of the compiled netlist. This contains timing information and because it uses compiled code the source can be anything. A limitation is that simulation is very slow.

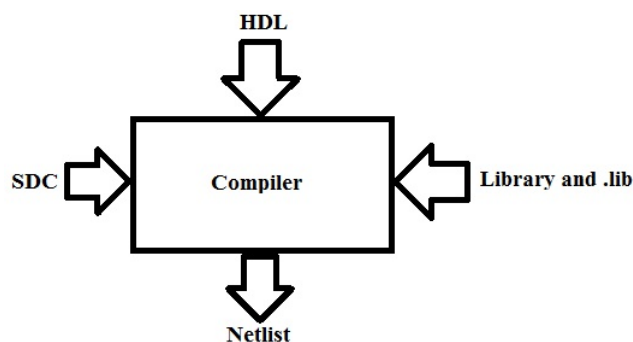


Figure 2.5: Block diagram for netlist generation

SDC (Synopsys Design Constraints) file contains the information related to timing delays, power consumed and area, and conditions applied on them.

.lib file is an ASCII representation of the timing and power parameters associated with any cell whether it is a combinational cell, a sequential cell or a buffer. It contains look-up tables for voltage and power look-up tables with information regarding power levels. It has the cell delays in a tabular form. It is used to verify that the given design meets its functional and timing requirements.

- Libraries:** Standard cell library contains primitive or basic cells required for digital design, however, more complex cells that are specially optimized can also be included. They give timing and power parameters obtained by simulating a cell under various conditions. It is a collection of low-level electronic logic functions such as AND, OR, INVERT, flip-flops, latches, and buffers. These cells are realized as fixed-height, variable-width full-custom cells. The key aspect with these libraries is that they are of a fixed height, which enables them to be placed in rows, easing the process of automated digital layout. The cells are typically optimized full-custom layouts, which minimize delays and area. It is used to calculate I/O delays and interconnect delays. A broad portfolio of high-speed, high-density and low-power standard cell libraries by synopsys is used by the tool, providing a complete standard cell platform solution for a wide variety of system-on-chip (SoC) designs.

The library usually contains multiple implementations of the same logic function, differing in area and speed. This variety enhances the efficiency of automated synthesis, place and route (SPR) tools. Indirectly, it also gives the designer greater freedom to perform implementation trade-offs (area vs. speed vs. power consumption). A complete group of standard-cell descriptions is usually called a technology library. Technology libraries are developed and distributed

by the foundry operators.

The standard cell libraries provide three separate architectures, high-speed (HS), high-density (HD) and ultra high-density (UHD), to optimize circuits for performance, power and area trade-offs.

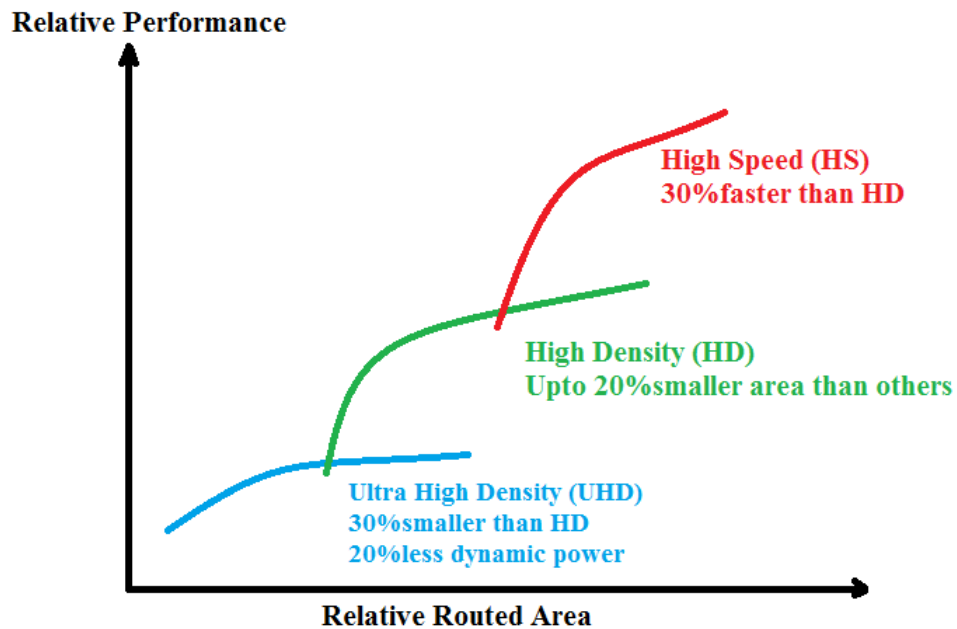


Figure 2.6: Performance, power and area trade-off in a testchip

4. **SDFs:** Abbreviated as SDF, standard delay format file tells how to present the circuit delays. SDF is spoken to in IEEE detail. It is an ASCII format and incorporates:

- Timing imperatives: way, skew, period, total, and difference
- Delays: module path, gadget, interconnect, and port
- Incremental and absolute delays
- Timing environment: expected operating timing environment
- Conditional and unconditional module path delays and timing checks
- Design/instance-specific or type/library-specific data
- Timing checks: setup, hold, recovery, evacuation, skew, width, period, and no change
- Scaling, environmental, and technology parameters

SDFs can be utilized at any stage as a part of the design flow for a precise and tool dependency free representation of circuit delays. In physical design flow, SDF documents are utilized for post-layout simulation and back-annotation. The STA device usually provides the SDF by writing it. This will give both interconnect delay and the cell delay. STA (standard timing analysis) is a simulation method of computing the expected timing of a digital circuit without requiring a simulation of the full circuit.

5. **VCD file:** Verilog can produce a file called a value change dump (VCD) file that contains information about value changes on selected variables in a design. VCD file contains time ordered value changes for the signal for simulation and is used for comparison and debugging. With the VCD feature, one can save the value changes of variables in any portion of the design hierarchy during any specified time interval. One can also save these results globally, without having to explicitly name all signals involved. It is used to test and save results of simulation

As shown in the following figure, the VCD file starts with the header information (the date, the version number of Verilog used for the simulation, and the timescale used). It also describes the names of the signals in the file, followed by an abbreviation code (1 character) that will be used everywhere later on to refer to the signal (to save space in the file). In the remainder of the file it contains 1 time instant per line of text, and only the changing signals that occur at that time point. (hence the name CHANGE dump).

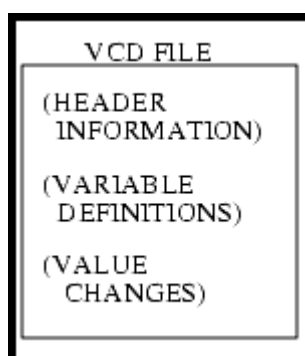


Figure 2.7: Contents of the dump file

Next, the file lists the definitions and the type of variables being dumped, followed by the actual value changes at each time increment. Only the variables that change value during a time increment are listed. Value changes for non-real variables are specified by 0, 1, X, or Z values. Value changes for real variables are specified by real numbers. Strength information and memories are not dumped.

6. **Configuration file:** Configuration file is useful to provide location of above files and relevant specification through variables. It is a mandatory input and the only input which is given as a command line argument to the pattern validation tool. Configuration file with absolute or relative path will be used as an input parameter to avoid hard-coded values inside the tool. It could be in terms of supported files and tool working directory and many other points. is used by the pattern validation tool to fetch required directory/file path, tool work directory, execution commands, supported categories/message groups/switches, error message strings, lsf commands and database archival information through different variables. Based on variable name, its interpretation is hardcode by the tool. This section gives idea about usage of different variables considering different design aspects. Tool uses various template files which are described in this section.

2.2.2 Output files

1. **Status Report file:** Pattern validation tool will provide status report file for each test case with supported categories.
2. **Information log:** Tool will also provide user with a runtime log as well as on screen log file which would be displayed on the terminal. Run-time log is a file created in the work directory which will consist of warning and error messages along with status information of the tool.
3. **Archive database:** Tool has the feature to provide archive database (if required by user) in zip tar format at the specified location. Tar file is generated after validating archived database with at least one run. The information provided by user in the config file decides whether to provide database in a compressed format like zip or tar format or to keep it uncompressed.

Chapter 3

Automation

Load sharing facility is used by the pattern validation to automatize it. Firstly, let me start with the need for automation which leads to the use of LSF through an intermediate tool called LSFKit[2]

3.1 Need for automation

The number of test cases can vary from one to hundreds depending upon different specifications like timing, voltage, power in different input files. Among all possible inputs for test case validation, few are common for all test cases and need to generate once. But inputs like VCD vary for each test case. Each test cases needs to be validated in one or many categories based on user specification. Above all, each test case needs to be fired on LSF for compilation, elaboration and simulation which result also needs to be evaluated with reference string to decide PASS or FAIL status manually.

After having a close look at the process, it is found that the complete process follows repeated effort while building environment build up and its execution and lacking core part or parallel launching of test cases on LSF server.

This repetitive nature of activity with little variation makes pattern validation a strong candidate for automation, where designer needs to intervene only for configuration and input test case validation file setup.

Test pattern validation automation is required because,

- Huge manual effort is required in test pattern generation, validation and delivery preparation
- Manual pattern validation is more prone to errors because of human intervention
- With frequently changing design during development phase, it is very difficult to test for a number of test cases manually
- By taking minimal non-redundant information from the user in the form of configuration file and set-up, reuse of data is possible
- Better utilization of LSF resources through LsfKit can make the tool more efficient in terms of accuracy and speed.

3.2 Load Sharing Facility (LSF)[1]

LSF, also known as compute farm is a network of distributed computers organized into a seamless system, to which multiple jobs can be submitted. There are various load balancing application software which enable this system.

It is a suite of distributed resource management products that:

- **Connects computers into a cluster (or grid):** In LSF computers are connected into a cluster so that task carried on by each computer is independent of each other but still they can be given any job and jobs can be shifted to any other computer if required.
- **Monitors loads of systems:** According to memory and CPU requirement and number of jobs to be run on the systems (computers) in LSF, jobs are managed and fired in different queues according to their requirements.
- **Distributes, schedules, and balances workload:** It schedules jobs and distributes workload on the queues. Jobs can be scheduled so that they can be fired only after the completion of the first job.
- **Controls access and load by policies:** Load is managed according to the requirements of CPU or time and memory. For example, a job with large duration will be fired in a long queue and a job with extensive memory usage is placed in bigmem queue.

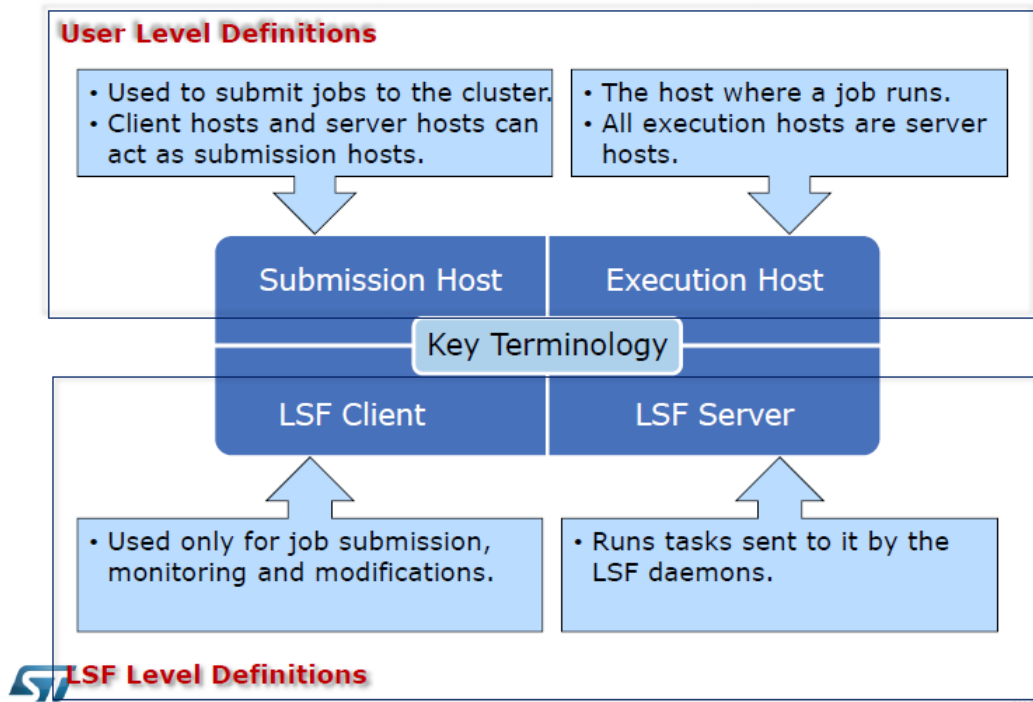


Figure 3.1: LSF Terminologies

- **Analyzes the workload:** Even if we fire a job in a particular queue, according to its resource requirement it is automatically shifted to another queue.
- **Provides transparent access to all available resources:** Any of the computer can be used to launch job irrespective of its position in the cluster.

Working of LSF

LSF include many terminals and each of them works as a submission host or an execution host.

- **Submission host:** It is used to submit jobs to the cluster of computers. Client hosts and server hosts can act as a submission host.
- **Execution host:** It is the terminal where all jobs are run.
- **LSF client:** Used only for job submission, monitoring of jobs and their modifications. This part is known as LSFKit. Jobs are handled here. It is an intermediate level before jobs are actually run and their execution is completed. It also shows the current status of each job that is fired.
- **LSF server:** It runs tasks sent to it by the LSF daemons.

About Compute Farm - Cluster concepts

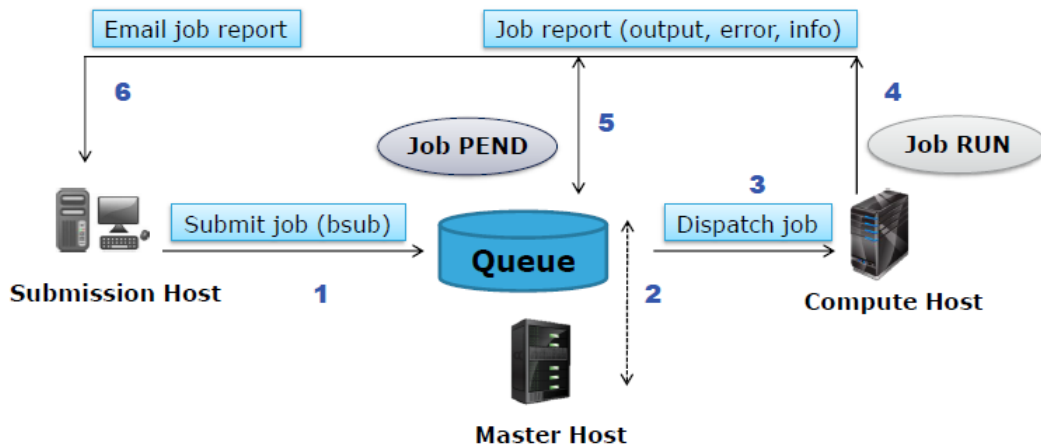


Figure 3.2: Job life cycle

Steps involved in job submission

- Job is submitted to LSF from a submission host. An LSF client or an LSF server can act as a submission host.
- Job is placed in one of the LSF queues, according to their resource requirements and passed to the scheduler to schedule the job for an organized execution of all jobs.
- Then, LSF master dispatches the job to execution host with an open job slot.
- Execution host on receiving the job, begin execution of the job.
- During execution of job, master host checks the status of the job whether it is running, done or exit state. And then, it sends the status of job to submission host.
- On completion of execution of job, results are send to the LSF client. Results can be sent even by an email to the client system.

LSF Queue Specifications

According to the resource requirements like memory, CPU utilization (in terms of speed) and number of jobs to be submitted, jobs are put in one of the LSF queues

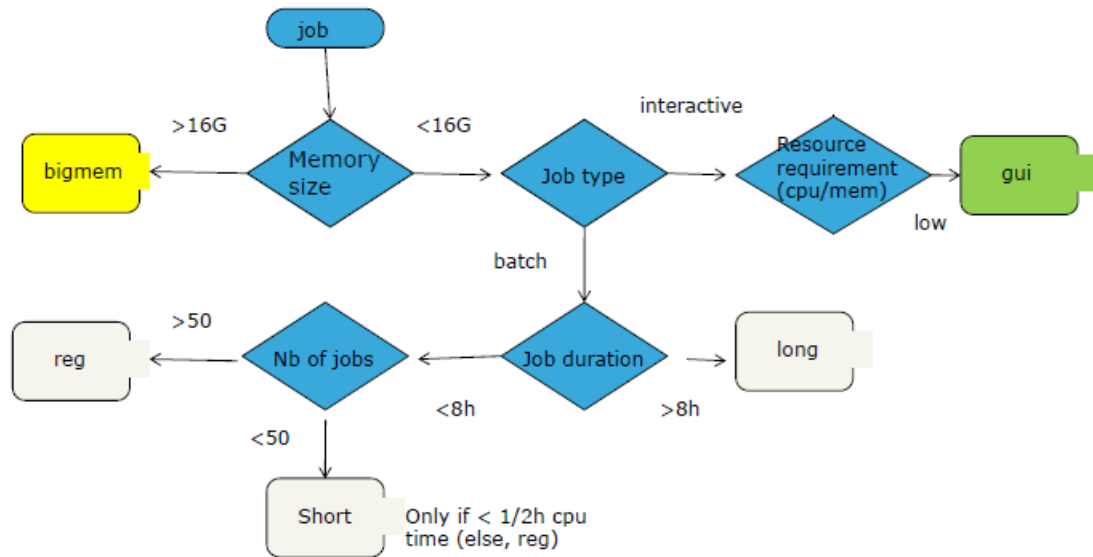


Figure 3.3: Queue usage model in LSF

- **gui queue:** gui queue is meant for interactive jobs, those which can give details related to the job, and intermediate results of the job. Jobs in gui queue are neither CPU intensive nor memory intensive. A specific host pool is dedicated to this queue (servers with gui LSF resources).
- **bigmem queue:** This queue is meant for the jobs which require memory utilization more than 16GB. LSF servers for bigmem queue have greater or equal to 256GB RAM. A specific host pool is available for bigmem queues also. For the usage of bigmem queue, specific authorization is to be requested. This is to make sure users are aware this is reserved for high memory jobs.
- **Batch queues:** Where more than one jobs are fired at once or require large time. There are three types of batch queues. These three batch queues share the same host pool. This pool includes servers with highest CPU factor. These queues are:
 - *short queue:* This queue is meant for jobs which requires less than 30 minutes of CPU time. Jobs running in short queue that require time more than 30 minutes are automatically killed.
 - *reg queue:* It is meant for many jobs that are to be launched in parallel.
 - *long queue:* This queue is for jobs which require more than 30 minutes of CPU time and not many parallel jobs are required to be launched.

The master host collects all resource usage from all slave hosts. Users submit jobs with the resource

requirements to Platform LSF. The master decides where to dispatch the job for execution based on the resource required and current availability of the resource.

The LSF system uses built-in and configured resources to track resource availability and usage. Jobs are scheduled according to the resources available on individual hosts. Jobs submitted through the LSF system will have the resources they use monitored while they are running. This information is used to enforce resource limits and load thresholds as well as fair-share scheduling. LSF collects information such as:

- Total CPU time consumed by all processes in the job
- Total resident memory usage in MB of all currently running processes in a job
- Total virtual memory usage in MB of all currently running processes in a job
- Currently active process group ID in a job
- Currently active processes in a job

LSF commands

- **bsub** is the command to submit a job through LSF. If no queue is specified while submitting the job, bsub command will use long as the default queue. When a job is submitted, a job file is created. A unique job id is assigned for each job. The job is executed when a host respecting resource requirement becomes available.

– For batch jobs:

```
% bsub -P (project name) -q reg (my job)
```

– For interactive jobs:

```
% bsub -Ip -P (project name) -q reg (my job)
```

– To submit a job that requires 18 GB of memory:

```
% bsub q bigmem -Ip -R "rusage[mem=18000]" P (project name) (my job)
```

The job will run on server where at least 18 GB memory is free and can be reserved. LSF will reserve this memory and start the job.

– To redirect output of LSF job to specific file:

```
% bsub -P (project name) -o /prj/xyz//out.log -q long (my job)
```

- **bjobs** for tracking a running job. It gives many features of the job, these are JobId, status, queue, submission host, execution host, job name, and submit time.

```
% bjobs -l (job Id)
```

- **bkill** To kill a currently running or pending job from the LSF.

```
% bkill (job Id)
```

3.3 LSFKit[2]

Earlier, LSF was not able to handle the jobs on its own, so LSFKit was used as an intermediate tool to handle the jobs. So, jobs used to be sent to the LSFKit which further sends the jobs to the LSF. Even, all the queries made to know about the jobs are sent to the LSFKit which further queries LSF. LSFKit is a an ST specific tool which is used to submit jobs on LSF, get current status of the job, gets output of the job, sends the result of the job to an email address when required and many other tasks like this to handle the jobs on LSF.

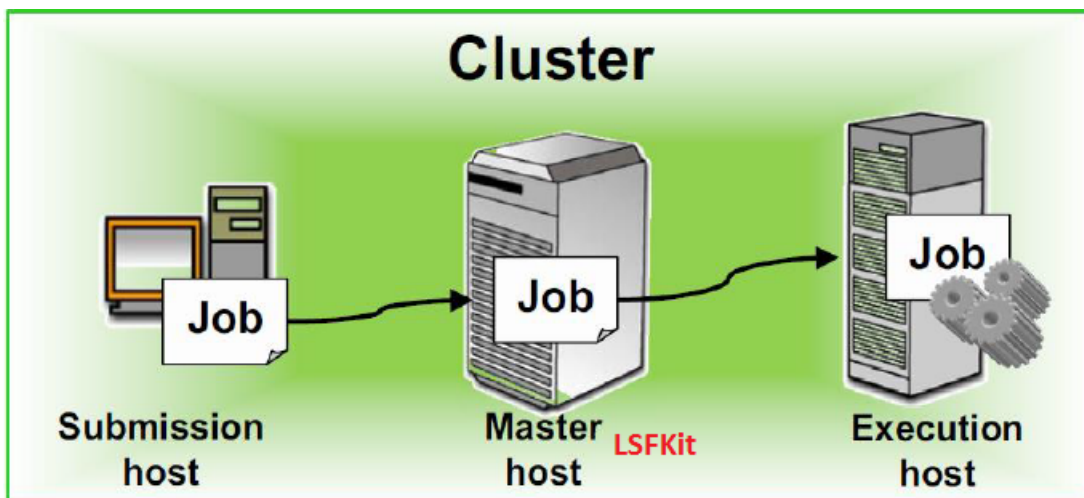


Figure 3.4: Queue usage model in LSF

LSF pain areas addressed by LSFKit are:

- Bsub command that is used to submit job on LSF is not able to submit when huge number of jobs are there at a time
- If some tool crashes or hangs, then LSF will keep on saying the state of job as RUN that is the job is Ok and currently running.
- Need to requeue, when some standard error messages are seen on lsf log or err files or License Issues
- Exec host down, LSF , at times comes to know after Delta delay(may be huge), in this time 1000s of jobs get dispatched on the same host
- LSF when requeues , does not take care of not dispatching on the same earlier host

- Job in unknown state
- Jobs exited without making any logs
- Jobs consume very High memory , while nothing is running

These were the areas which used to be addressed by LSFKit. But as the time advanced, LSF has been modified and now LSF has become capable enough to handle these issues and others like receiving jobs, status check and many others from the user on its own. LSFkit causes errors which can be resolved if jobs are handles directly by the LSF.

Errors that occur due to the use of LSFKit are:

- Child process sometimes get exited abnormally
- Job sent for submission to LSFKit is not sent to LSF for job execution
- Sometimes, command failed and command not found messages are given
- Job submission is rejected when 32-bit Linux login server is not used to use 32-bit Linux queue and 64-bit Linux login server is not used to use 64-bit Linux queue

Due to these errors experienced by the use of LSFKit, there is an urgent requirement to remove the dependency of the 'test pattern validation automation tool' to be removed and use LSF directly.

Chapter 4

Standard Cell Block Design Tool

It is a tool to test standard cell libraries for functional testing of standard cells and leakage measurement of these libraries on the whole. There is a particular set of reference libraries with pre-defined rules which are used to test new libraries.

4.1 Introduction

Standard cell block design tool is used to test the libraries of standard cell with already existing reliable reference libraries. These standard cell blocks include combinational cells, sequential cells, tristate buffers and CBUF cells (clock gating cells).

Purpose of standard cell is:

- Functional testing of standard cell libraries
- Leakage measurement of these libraries on the whole.

Following these test specifications we provide an input.csv to the STD cell generation tool which in turn generates out a RTL for us. Input csv (comma separated version) file contains information related to libraries that are to be tested like number of Cells in Library, number of combination cells in the library, number of sequential cells in the library, number of tristate cells in the library, cells not to be tested, number of repetition of cells, reference library name combinational, reference combinational cells, reference library name flip flop, reference flip flop, cell height, dedicated supply, leakage comparison library, leakage value of library, etcetera.

4.2 Standard Cell Libraries

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. A technology library consists of all the information about functioning and characteristics of every cell provided in the a semiconductor vendors library. A cell is a component that performs a basic function. It is more of a digital design concept, boolean and basic functions like AND, OR etcetera. A view is a particular representation of a cell. Each cell may have a layout view, schematic view, a symbolic view, a timing view etc. A cell is delivered as a set of view & each view is used by a different tool in a given design flow. A technology library (.lib) describes the structure, function, timing and environment of the ASIC technology being used.

The technology library consists of the following information for synthesis:

- Mapping: Functional information for each cell.
- Optimization: Area and timing information for each cell.
- DRC: Design rule check constraints on cells.

The library files are defined at various PVT (process, voltage and temperature) conditions. These PVT conditions basically specify the conditions on which the cells of the library are characterized. At the top of the .lib, these PVTs are defined in comments.

Each cell in the library contains a variety of attributes describing the function, timing and other information related to each cell. Some of these attributes are at cell level and some are at pin level. There are some attributes used to define the timing also. These attributes are used by the ASIC design tools (From RTL to GDS) in the entire ASIC design flow. Depending upon the value of these attributes the various types of library cells, eg., standard cell or IO or memory are differentiated.

```
cell(HS65_LHS_XNOR2X12){
area : 6.76;
#Area defines the cell area as a floating point number.
cell_leakage_power : 1.48811e-06;
#This is the average leakage power of the cell. The units are already specified.
drc_blockgroup : "stdcell";
#For standard cell : stdcell
#For IOs : periphery
```

#For Memories : block

Combinational	For combo cells, when cell's output are a function of cell inputs. eg. NAND,NOR
Storage	For cells that have an internal state and can store information. eg. RAM,ROM
Interface	For cells that are used to connect and amplify signals. Eg. Bider cells, tristate cells,i/p & o/p buffers
Routing	For cells that contain only supply or dummy pins. Eg. VDD/VSS cells, feed through cells, IO corners.

Cell characteristics may include cell information like its name, pin names of the cell, pin loading (or tolerable power and voltage at each pin) and delay arcs. They also include, marginal conditions that must be satisfy to make the cell work well functionally for example, maximum transition time of nets. To synthesize an RTL to produce netlist, two files are required- one is RTL itself and compiled library (in .db format).

Before a library can be used to generate netlist, it needs to be compiled into .db format To compile .lib files to generate .db library file, a tool called "Synthesis", provided by Synopsys is used. Program called dc_shell is a command line interface to Synthesis. to compile .lib to .db, following commands are run:

```
% dc_shell
```

```
dc_shell >read_lib example.lib
```

```
dc_shell >write_lib example
```

```
dc_shell >quit
```

dc_shell is also used to synthesize RTL to create gate level netlist.

Library file is sectioned into two parts: a header section which defines attributes to be used by every cell in the given library and the cell section which has description of each cell like pin attributes like names, functionality, area, timing, power etc. In these libraries, to assign delays observed by the cells under various conditions lookup tables are used. This is called non-linear CMOS delay model. Two components of delay are

- Dcell (Delay due to internal circuitry of the cell)
- Dconnect (delay due to interconnects, it is zero if we don't specify a value explicitly for it)

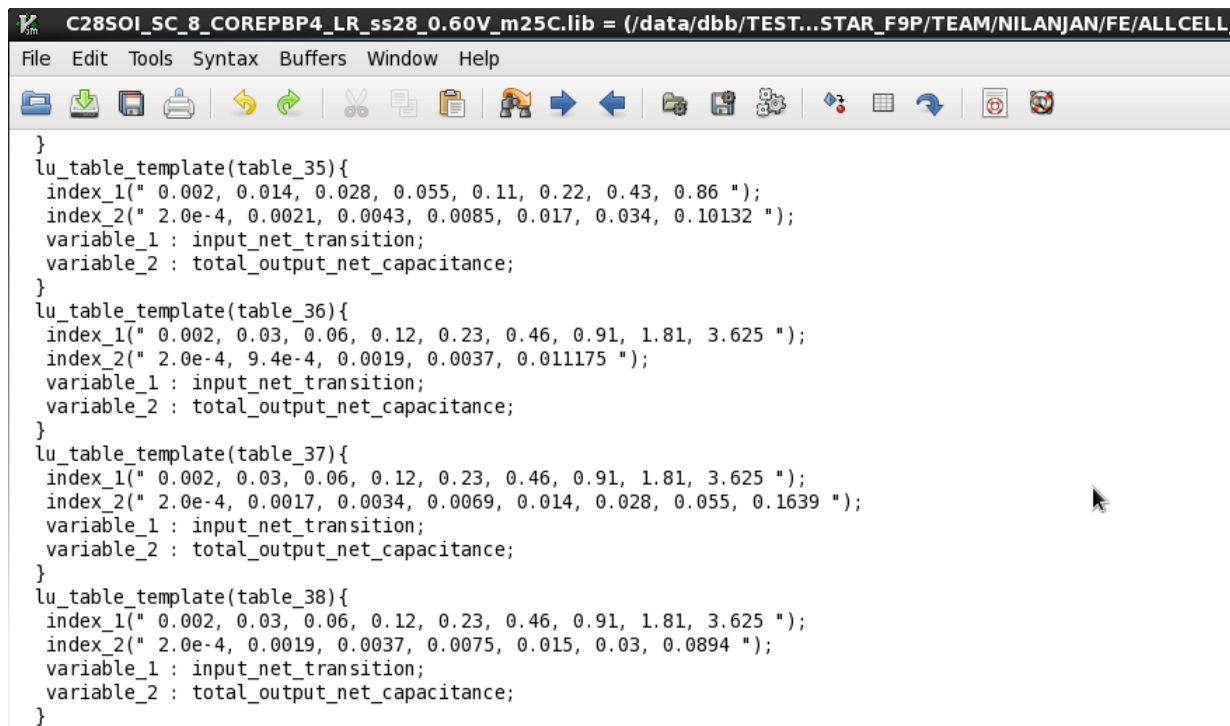
Dcell can be computed in two ways

- as a sum of two delays: in this the sum of both propagation time and transition time is considered.

Both of these values are found out from the delay look-up tables

- as a single delay (cell_rise, cell_fall) which also read from the delay look-up tables

The non-linear timing model can use a 1-D, 2-D or 3-D look-up tables. In the given project, we have used 2-D delay look-up tables. For a 2-D table, one of the axes must be input transition time and the other one can have many optional values among which the one that is most used is output net capacitive load. The units of look-up tables can be anything that is consistent with the data in the look-up tables. For capacitance it is generally microFarad and for input transition time, it is generally, ps or ns. In a look-up table, leftmost value is generally transition time of fastest cell cell, that is with no load and the rightmost value represents transition time of weakest cell with maximum load.



```

}
lu_table_template(table_35){
index_1(" 0.002, 0.014, 0.028, 0.055, 0.11, 0.22, 0.43, 0.86 ");
index_2(" 2.0e-4, 0.0021, 0.0043, 0.0085, 0.017, 0.034, 0.10132 ");
variable_1 : input_net_transition;
variable_2 : total_output_net_capacitance;
}
lu_table_template(table_36){
index_1(" 0.002, 0.03, 0.06, 0.12, 0.23, 0.46, 0.91, 1.81, 3.625 ");
index_2(" 2.0e-4, 9.4e-4, 0.0019, 0.0037, 0.011175 ");
variable_1 : input_net_transition;
variable_2 : total_output_net_capacitance;
}
lu_table_template(table_37){
index_1(" 0.002, 0.03, 0.06, 0.12, 0.23, 0.46, 0.91, 1.81, 3.625 ");
index_2(" 2.0e-4, 0.0017, 0.0034, 0.0069, 0.014, 0.028, 0.055, 0.1639 ");
variable_1 : input_net_transition;
variable_2 : total_output_net_capacitance;
}
lu_table_template(table_38){
index_1(" 0.002, 0.03, 0.06, 0.12, 0.23, 0.46, 0.91, 1.81, 3.625 ");
index_2(" 2.0e-4, 0.0019, 0.0037, 0.0075, 0.015, 0.03, 0.0894 ");
variable_1 : input_net_transition;
variable_2 : total_output_net_capacitance;
}
}

```

Figure 4.1: Example of look-up tables in a Standard cell library

For a complete chip design, input output pads have to be placed on the pins of the standard cells. This can be achieved by reading gate level netlist and using 'insert_pads' command for dc_shell. For this, it is required to have pad cells in the standard cell library. The circuitry present on a chip has to be

connected to other circuits. Function of input output pads are used to connect core circuit signals to the outer world. They may also consist of special circuits for protection from electrostatic discharge (ESD) because gates of input MOS transistors are directly exposed to input pads and high voltage to these gates may permanently damage the MOS transistors. Output pads don't need ESD protection because gates of output pads are not directly exposed to pads. Output pads should be capable of consuming large power. Following are the points that are to be taken care while creating the library groups:

- Separate netlist can be generated for these groups, hence those libraries which require dedicated power supply should be kept in separate group.
- Libraries which cannot be routed together should be kept in separate groups.

4.3 Structure and Working

[4] Standard cell block is used for the functional testing and delay measurement of the standard cell libraries on the whole. Following these test specifications we provide an input.csv to the STD cell generation tool which in turn generates out a RTL for us. The behavior of these standard cell libraries are viewed by comparing it with behavior of reference libraries.

The Libraries with combinational, sequential or tristate cells are functionally validated with respect to reference units. The output from libraries to be tested is compared with the output of reference cells. The segregation of Libraries in different groups is of special importance. Taking after are the focuses that are to be taken into consideration while making the library groups: - Separate netlist can be generated for these groups, hence those libraries which require dedicated power supply should be kept in Separate group. - Libraries which cannot be routed together should be kept in separate groups.

Reference Units: The reference unit consists of reference flip flops, reference multiplexer, reference buffers and reference AND or OR gates.

Basic cells for testing are COMBO, SEQ, TRI and CBUF. The main difference between these units are the types of cell under tests (CUTs)

- **COMBO:** This contains cells which include AND, OR, INVERTERS, BUFFERS etc and these cells are called as CUT (cell under test).

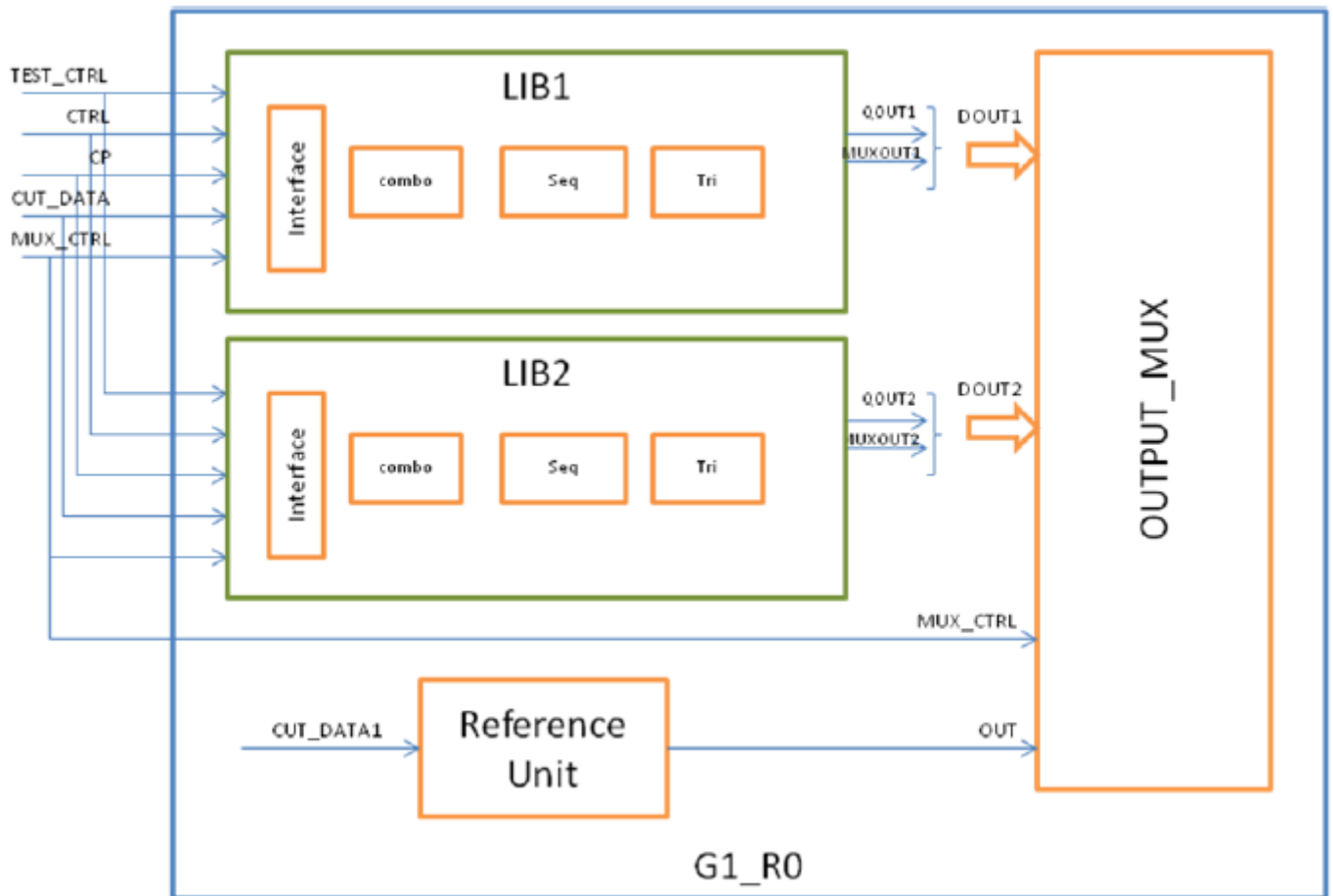


Figure 4.2: Block diagram of test circuit architecture

- **SEQ:** In SEQ, the cells are flip flops or we can say sequential cells. Same clock input is provided to the CUT and the reference flip-flop. Rest of the components are same as in combinational cells
- **TRI:** This block contains tristate cells as CUTs. The tristate cells has an enable input E which enables the tristate cell.
- **CBUF:** The clock gating cells are kept in CBUF unit. An essential clock gating cell has 3 inputs, E(Enable), TE(test enable), and CP (clock input). E and TE do both are used to enable the clock spread through the cell. Both these inputs are required in the cell since clock gating empower must be top-controlled in test mode other than being inside created in functional mode.

Shown in the figure is the basic unit inside std. cell block used for functional testing. Main components of this unit are: cell under test [CUT], reference cell, Outmux.

We have Library instances and Interface unit. The library units (std. cell architecture) can be of following types:

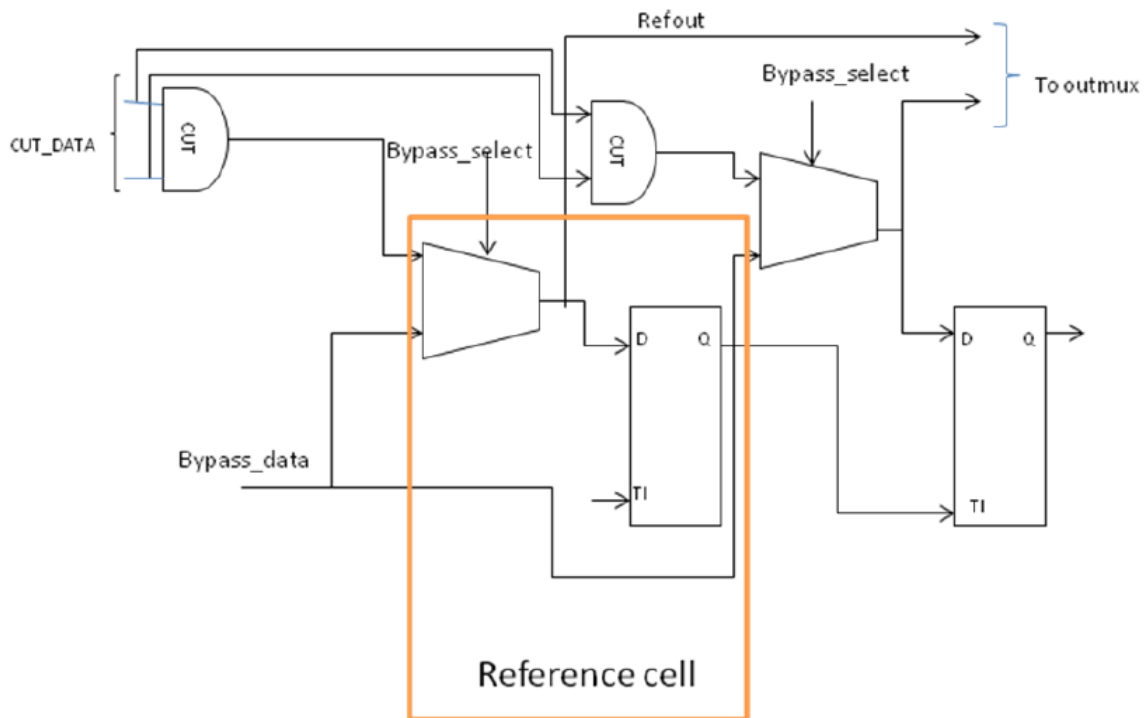


Figure 4.3: Basic circuitry for cells functionality testing

- ALLCELL library unit
- FDD
- RETENTION

4.3.1 ALLCELL

In ALLCELL architecture there is provision for testing of every types of cells by segregating cells in blocks such as COMBO, SEQ, TRI and CBUF.

The basic cells for ALLCELL are COMBO, SEQ, TRI, and CBUF. The main difference b/w these units are the type of CUTS (cell under test). This contains cells which include AND, OR, INVERTS, BUFFERS etc and these cells are called as CUT (cell under test).

Presently all the Cuts share the inputs. Outputs from these cuts are applied to the D0 input of the bypass mux. Bypass mux, as the name suggests, is given a bypass input at D1. Through this input we can test the D-Q path of ref flop to which the output of bypass mux is applied. Reference cell comprises of a bypass mux and a flop. Ref flops are connected in a scan chain such that, Q output of each flop is

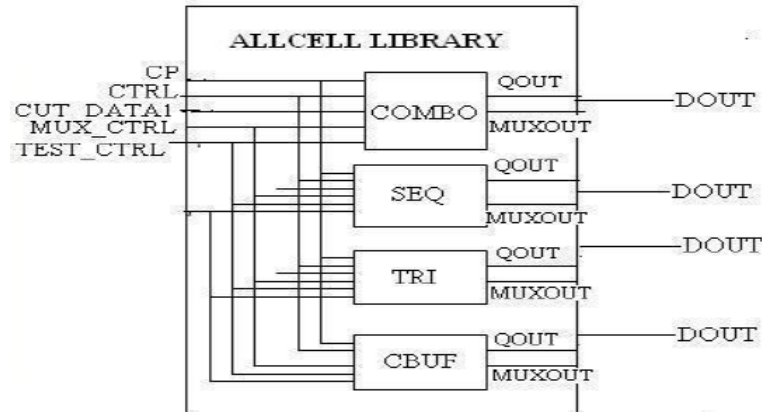


Figure 4.4: ALLCELL library block

connected to TI Input of next flop. Reference cell comprises of a bypass mux and a flop. Ref flops are connected in a scan chain such that, Q output of each flop is connected to TI Input of next flop. The no. of ref flops for each CUT depends upon type of block i.e. [allcell, FDD or retention].

4.3.2 FDD

FDD block was originally used for flops that have dual edge triggering capability. Now this block is also used for flops with some other special features and critical than the normal flop. Library is instantiated huge no. of times (1024) on a testchip. The dual-edge triggered flip-flop exhibits low delay and small power consumption because the clocking is done at half the clock frequency as compared to single-edge triggered storage element. For these and some other special flops, there is requirement of large no. of instantiation. Also, the library is very small (24 -36). Hence, we cannot use SEQ block of these cells.

In FDD architecture the whole library is present in the place of CUT of ALLCELL block. Thus a whole library shares a Ref cell which takes the multiplexed output from the library. This block is then repeated as per the requirements.

4.3.3 Retention Block

Retention block is used to test Retention flops which retain their data. Architecture and connectivity wise this block is just same as FDD block. Separate block other than FDD is used for these flops

because the SLEEP pins of these flops have to be routed through always on cells.

4.4 Work done

4.4.1 Dependency of the execution of tool on home area of user

The space available in the home area of the user must be sufficient to run the standard cell block design tool even if the tool is run in the work area separated from the home area. Major task was to find the reason of this dependency of tool on home area. This was done by understanding the source code of the tool and to find which command or the intermediate tool is responsible for this dependency that is some tool is creating some file in home area and when sufficient space is not available in the home area, this file is not created in home area. This results in abnormal behavior of the tool like all the required output files are not created.

On going through the source code, it was found that LSFKit was used to fire jobs on LSF to check libraries. LSFKit creates a temporary file .lsfkit is created in the home directory of the user. To remove this dependency of the tool on the home area, dependency of the tool on LSFkit is removed and jobs are fired directly on the LSF.

The diff file for difference between the old code of the tool and the modified one is as follows:

```
diff -bBir /sw/st_division/ccds/SquirrelStdCell/2.1/bin/SquirrelStdCell.csh 2.2/bin/SquirrelStdCell.csh
123,124c123,127
<echo myBsub -q $queue -J $Jobname_count >${WORKDIR}/my_line_1_part1
<paste $WORKDIR/my_line_1_part1 ${WORKDIR}/my_line_1 >
$WORKDIR/.${Jobname_count}.runme
—
>echo bsub -q $queue -J $Jobname_count >${WORKDIR}/my_line_1_part1
>echo ”>& ${WORKDIR}/tmpfileid” >${WORKDIR}/my_line_1_part3
># echo myBsub -q $queue -J $Jobname_count >${WORKDIR}/my_line_1_part1
>paste ${WORKDIR}/my_line_1_part1 ${WORKDIR}/my_line_1 >${WORKDIR}/my_line_combined
>paste ${WORKDIR}/my_line_combined ${WORKDIR}/my_line_1_part3
>${WORKDIR}/.${Jobname_count}.runme
```

134,145c137,157

```
>set idstatuschk = "FAIL"
```

```
>while($idstatuschk != "PASS")
```

```
>touch "${WORKDIR}/tmpfile" "${WORKDIR}/tmpfileid"
```

```
>sleep 2
```

```
>myBjobs -J $Jobname_count—grep -v LSFKIT_STATUS >"${WORKDIR}/tmpfile"
```

```
>sleep 2
```

```
>${SQUIRRELSTDCELL_TOOL}/lib/perl/findjobid.pl "${WORKDIR}/tmpfile" "${WORKDIR}/tmpfileid"
```

```
$usr
```

```
>sleep 2 >set id = 'cat "${WORKDIR}/tmpfileid"—grep ID—awk -F "=" '{print $2}''
```

```
>rm -rf "${WORKDIR}/tmpfile" "${WORKDIR}/tmpfileid"
```

```
>sleep 2
```

```
>set idstatuschk = "${SQUIRRELSTDCELL_TOOL}/lib/perl/checkid.pl $id'
```

```
—
```

```
>rm -f ${WORKDIR}/my_line_1_part1 >&/dev/null
```

```
>rm -f ${WORKDIR}/my_line_combined >&/dev/null
```

```
>set id = 'grep -e "<[0-9][0-9][0-9]>" ${WORKDIR}/tmpfileid — awk '{print $2}' — sed s/textgreater//g — sed s/</g'
```

```
>echo "For job: $job_entry" >${WORKDIR}/LSF_SQUIRRELSTDCELL_OUT.log
```

```
>cat ${WORKDIR}/tmpfileid >>${WORKDIR}/LSF_SQUIRRELSTDCELL_OUT.log
```

```
>
```

```
>Changes made in version 2.2
```

```
># set idstatuschk = "FAIL"
```

```
># while($idstatuschk != "PASS")
```

```
># touch "${WORKDIR}/tmpfile" "${WORKDIR}/tmpfileid"
```

```
># sleep 2
```

```
># myBjobs -J $Jobname_count—grep -v LSFKIT_STATUS >"${WORKDIR}/tmpfile"
```

```
># sleep 2
```

```
># ${SQUIRRELSTDCELL_TOOL}/lib/perl/findjobid.pl "${WORKDIR}/tmpfile" "${WORKDIR}/tmpfileid"
```

```
$usr
```

```
># sleep 2
```

```
># set id = 'cat "${WORKDIR}/tmpfileid"—grep ID—awk -F "=" '{print $2}''
```

```
># rm -rf "${WORKDIR}/tmpfile" "${WORKDIR}/tmpfileid"
># sleep 2
># set idstatuschk = "${SQUIRRELSTDCELL_TOOL}/lib/perl/checkid.pl $id"
>rm -f "${WORKDIR}/tmpfileid >&/dev/null
>
diff -bBir /sw/st_division/ccds/SQUIRRELSTDCELL_Flow/2.1/lib/shell/checkjob.csh 2.2/lib/shell/checkjob.csh
4,8c3,5
<myBjobs $jobno >& ${jobno}_stat.txt
<set mess = "${SQUIRRELPATTERN_HOME}/lib/perl/parse_bjob_file.pl ${jobno}_stat.txt $jobno"
<rm -f ${jobno}_stat.txt >&/dev/null
<rm -f $2
<echo $mess >$2
—
>set mess = 'bjobs $jobno'
>set stat='echo "$mess" — awk '($9 = $jobno) {print $11}''
>echo $stat >$2
9a7,17
>
>
>
>
>
>
>Changes made in version 2.2
>#myBjobs $jobno >& ${jobno}_stat.txt
>#set mess = "${SQUIRRELPATTERN_HOME}/lib/perl/parse_bjob_file.pl ${jobno}_stat.txt $jobno"
>#rm -f ${jobno}_stat.txt >&/dev/null
>#rm -f $2
>#echo $mess >$2
diff -bBir /SQUIRRELSTDCELL_Flow/2.1/setup/setup.csh 2.2/setup/setup.csh
27c27
<setenv SQUIRRELPATTERN_HOME /SquirrelPattern/v1.0_beta14/
—
```



```
>setenv SQUIRRELPATTERN_HOME /SquirrelPattern/v1.0_beta15/
```

4.4.2 To combine libraries to increase efficiency

Different standard cell libraries, to be checked for functional correctness and delay measurement, use different inputs and outputs. For this, libraries with similar inputs and outputs can use same input output pads. To increase the efficiency of the tool, it would be better if the libraries which need same I/O pads, that is the libraries with same reference library can be combined for better performance of the tool. For this, three conditions need to fulfilled, these are:

- User should allow the tool to combine libraries
- Libraries to be combined should be in the same group
- The sum of combinational cells in libraries to be combined should be less than or equal to 1024

For this, the tool has been modified by modifying its source code. The source code is written in such a way that, if there are some libraries that are in the same group, the tool would ask the user if he wants the tool to combine these libraries. If user would provide, a 'Y' or 'y', the tool would add combinational cells present in every library and if the sum of combinational cells in these libraries is less than 1024 (2^{10}), libraries would be combined. Whole flow is shown in figure 4.4.2. When these libraries are combined, the resultant library needs to follow a particular format for structure of libraries. Firstly, look-up tables for every libraries are written, by following there order in each library. similarly, for power look-up tables. At the end, cells in every library is added to the resultant library. The resultant library needs to have .lib extension.

The information like library group, total number of cells, total number of combinational, sequential, tristate buffer cells, cells not to be tested, reference libraries and other information related to standard cell libraries are present in input csv file, whose path is given as an input to the tool. Along with that, path of the folder which contains all the libraries to be checked for functional validity and leakage measurement. For the above requirement, I have written the source code in PERL and C Shell. In PERL, extensive usage of hashes and arrays is there, which makes the parsing of input csv file much easier.

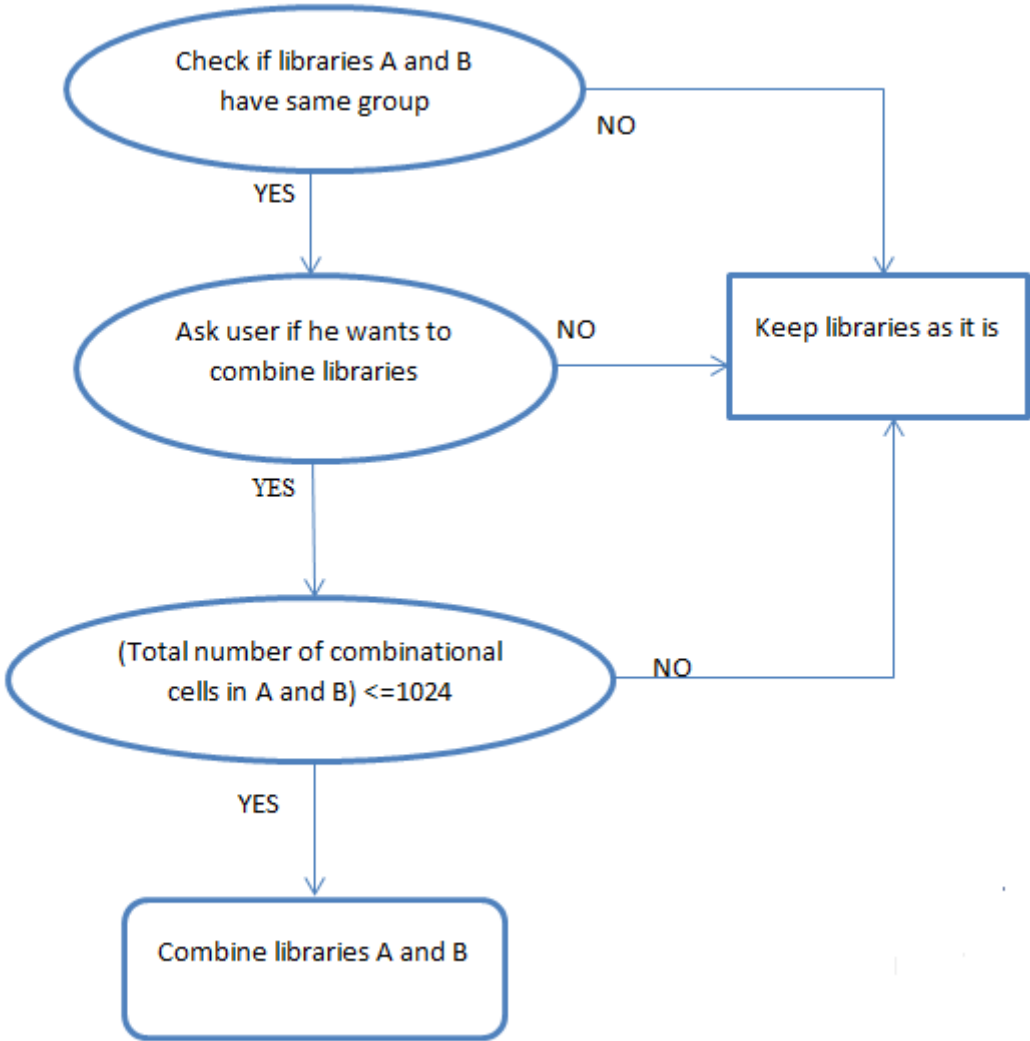


Figure 4.5: Flow diagram to show working of tool to combine libraries

Chapter 5

CAD Characterization Flow (CCF) Tool

5.1 Introduction

With the advancement of technology, the level of complexity in SOC design keeps on increasing which leads to paramount increase in the number of critical paths in the digital design. Ensuing advancements in verilog and mixed mode simulation have incredibly helped in far reaching assessment of computerized outlines based upon such advances and have lessened the crevice amongst CAD and Silicon comes about impressively. In any case, the large scale perceivability of outlines that these systems are restricted to, controls them to constrained PVT corners as well as increases simulation time manifold. The arrangement introduced here manages such constraints and endeavors to reflect upon partial system simulation techniques.

5.1.1 Background

The estimation of access time of a memory IP acknowledged in testchip vehicle in view of 28nm CMOS FDSOI has been singled out as a viable experiment where because of noteworthy inaccuracies recognized in delay estimations, a prerequisite was activated to infiltrate minutely into the measurement design and distinguish every single dispersed node and the electrical paths associating them and inspect them by applying particular stimuli. Static time analysis method is extraordinarily constrained by the limited extent of procedure corners which are the hybrids of manufacturing reliability of MOS-

FETs, working voltages and junction temperatures. Further, the delay estimation of critical paths must be demonstrated with their identical capacitive delays rather than the actual RC delays. This transformation, aside from changing the physical postponement profile also includes simulation time. The visibility of the outline is on a full scale level i.e. portrayal for a little area is impossible.

While there are strictly analog hardware circuit simulators, prevalent simulators often incorporate both analog and event driven digital simulation capacities, they are called mixed mode simulators. A whole mixed signal analysis can be driven from one incorporated schematic. All the advanced models in mixed signal analysis give precise detail of propagation time and rise/fall time delays. Mixed mode simulation techniques addresses every such downside of Static Time Analysis, however an across the broad characterization of an equivalent circuit would need to go through many analog to digital stages before giving back the real measurable parameters subsequently as a result representing monstrous runtimes. The visibility as yet being on macro level, the arrangement of stimuli appointed to be connected on a singular module would fundamentally be acquired on the configuration limits, bringing about a larger than average bank of input patterns.

5.1.2 Motivation

It was for the above reason, a prerequisite was identified to enter into the bigger hierarchical blocks and make independent stimuli profiles. The proposed arrangement which lies amongst HDL and mixed mode is basically created with the objective to give quick, proficient and precise method for extracting results. In addition it likewise covers a substantial scope of corners or more all particular way can be separated independently making flat hierarchical analysis possible. With the increase in technology complexities, detailed analysis using the conventional approaches is highly time consuming as, to pin point the issue, they use complete spice simulation, which are large in size. For faster analysis we need to have path based simulation methods which enable us to extract spice netlist of any desired path.

5.2 CAD Characterization Flow Tool

At the point when assessing delay values basic way with standard IPs and other rationale pieces utilizing variable jolts designs adapted according to innovation attributes of CMOS28FDSOI, the CAD

information yielded huge deviations from the outcomes obtained from Post silicon trials. The tool developed can be used to simulate any path of any hierarchy inside a complex chip using true spice on wide range of operating conditions. It provides accurate path error of the testchips on all the operating conditions yielding in better CAD to Silicon Correlation of the Memory parameters being tested which was not possible to do exhaustively on all operating conditions using conventional approaches. This tool provides a better Digital to Analog interface as the exact electrical behavior of the desired path can be extracted using analog simulation without much intervention. Also, the tool can be used for analysis of, Sensor data , Temperature sensitivity, voltage sensitivity, RC Impact and impact of device sensitivity. The generic nature of the tool will make it a vital part of the design flow. The underlying functioning of the process will remain unchanged irrespective of changing technologies.

The best case corner examination is combined with greatest voltage and the most pessimistic scenario corner is combined with the base voltage with the aim of merging the confound results over intermediate process corners whose deviations would be described by virtue of it. The two procedure corners chosen for this trial are the routine alternatives for approximating the mismatch amongst CAD and silicon. The mismatch elements acquired from these outcomes were likewise appointed to different process corners and the estimation was corresponded with Si results. The corresponding delay variation from Best case worked with 0.80V and worst case worked with 1.20V surpassed 500ps which was a huge variable of the real delay esteem. This enhanced technique other than ramping up the general accuracy of the pre-silicon trials would likewise give plenteous experiments on which the actual Silicon information could be accepted. However the large scale visibility nature of the STA would require that the trials are keep running from the chip limit with a plenteous boosts set. Variation from the expanded number of recognized procedure corners likewise expanded the simulation time drastically. Mixed mode analysis methods also posed similar constraints. Further examination yielded a requirement to confine the extent of trials to lower design hierarchy and considering just the particular basic way under test and reduce the whole stimuli set to a set number of patterns. The fractional design simulation would dispense with all redundant hierarchies of command and their way parts. To lessen testing time, memories are self described with an embedded on-chip Built in self Characterizer (BISC block).

5.2.1 Built in self Characterizer (BISC) Module

Using this block, access time of memory, setup time and maximum operating frequency measurements are done with quite good accuracy. BISC module has following components:

- Pulse Generator:** As shown in figure 5.2.1, it takes the clock input of the memory and output data of the memory, which is mostly, the fastest and the slowest output signal of the memory. The difference between the arrival of clock input and the output of memory is considered to be the access time of the memory. Pulse generator creates a pulse whose width is directly proportional to the access time of the memory. Output of pulse generator is sent to access time characterizer.
- Access time characterizer (ATC):** ATC which investigations the pulse width sent by the pulse generator conveys a computerized code which gives the estimation of the access time of the memory for the corresponding output.

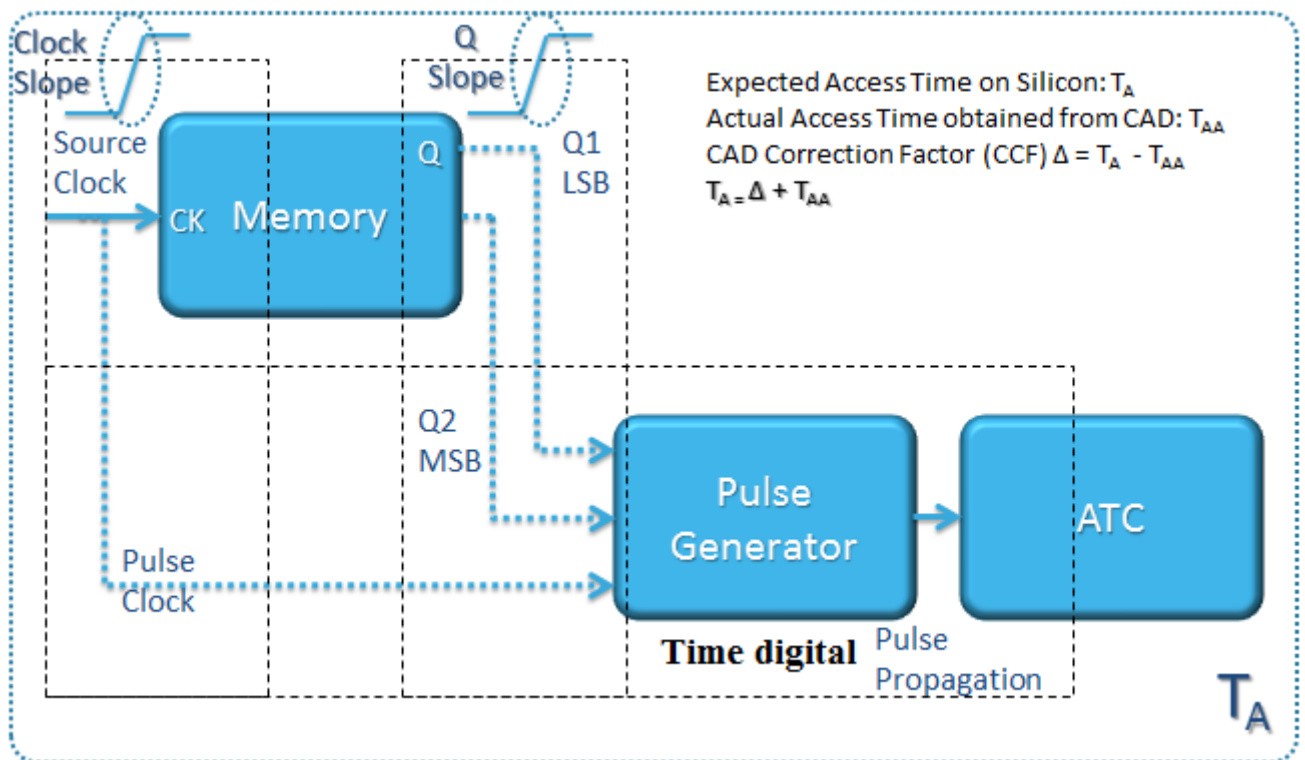


Figure 5.1: Setup for BISC module

5.2.2 Working

When measuring access time of memory, figure 5.2.1 is used. The whole circuit utilized for measurement is partitioned from the point of beginning of information and clock path into sub-circuits which can be portrayed by their own parasitic characteristics, for example, their wireload properties, data input capacitance, signal transition abilities and output load. These conveyed electrical paths interface key node points in the configuration and eventually follow the complete path of input signal to the characterizer. The diverse critical path contribution among various nodes recognized and included to compute the mismatch factor.

- **Total Mismatch factor:** It is the path from memory output pin (Q) to input of the ATC. It is basically the summation of the three autonomous components specified underneath.
- **Balancing Factor:** It is the balancing factor accounted for path from memory pin to pulse generator input, compared against the clock pulse path.
- **Path Factor:** It is the section of path from pulse generator to the access time characterizer.
- **Converter Factor:** From pulse generator input to its output.

Value for these factors are found at different process (P), voltage (V) and temperature (T) conditions. The source clock path comprises of the input clock path to the memory which keeps running more than two buffers and ends at the junction point where the net leading to pulse generator branches off. The declination in edge slope would fundamentally stay under irrelevant deviations independent of the quantity of cells in the way and restricting the cells improves simulation time. The memory clock simulation extracts and approximates the definite slope at the clock data. It is better for useful resource management that a single ATC is functions for various memories, with inputs at the ATC optimally multiplexed. Therefore, access time at CAD level is not accurate and differs from that at SPICE level.

Assessment of signal starts with the node which further branches off to two signals, one to the memory input and another to the pulsegen input. The clock, applied from periphery of test vehicle, propagates through various intermediate combinational logics which modify the clock signal which reaches the input of memory. The intermediate logics are generally buffers to amplify the clock signal.

The memory is considered to be a black box, only its output is used to calculate the access time, its internal design is out of scope of the access time mismatch assessment. Next level assessment is made at the pulse generator input. The clock pulse from point of memory input to the pulse generator input is made to pass through some intermediate buffers to balance clock mismatch with the data output of the memory.

The final level of assessment is done on the path from pulse generator to the access time characterizer. The pulse rises up out of the output node of the converter and spreads through the pulse propagation path towards the characterizer and the true duration of the pulse gets rotted when it goes through an intermediate node.

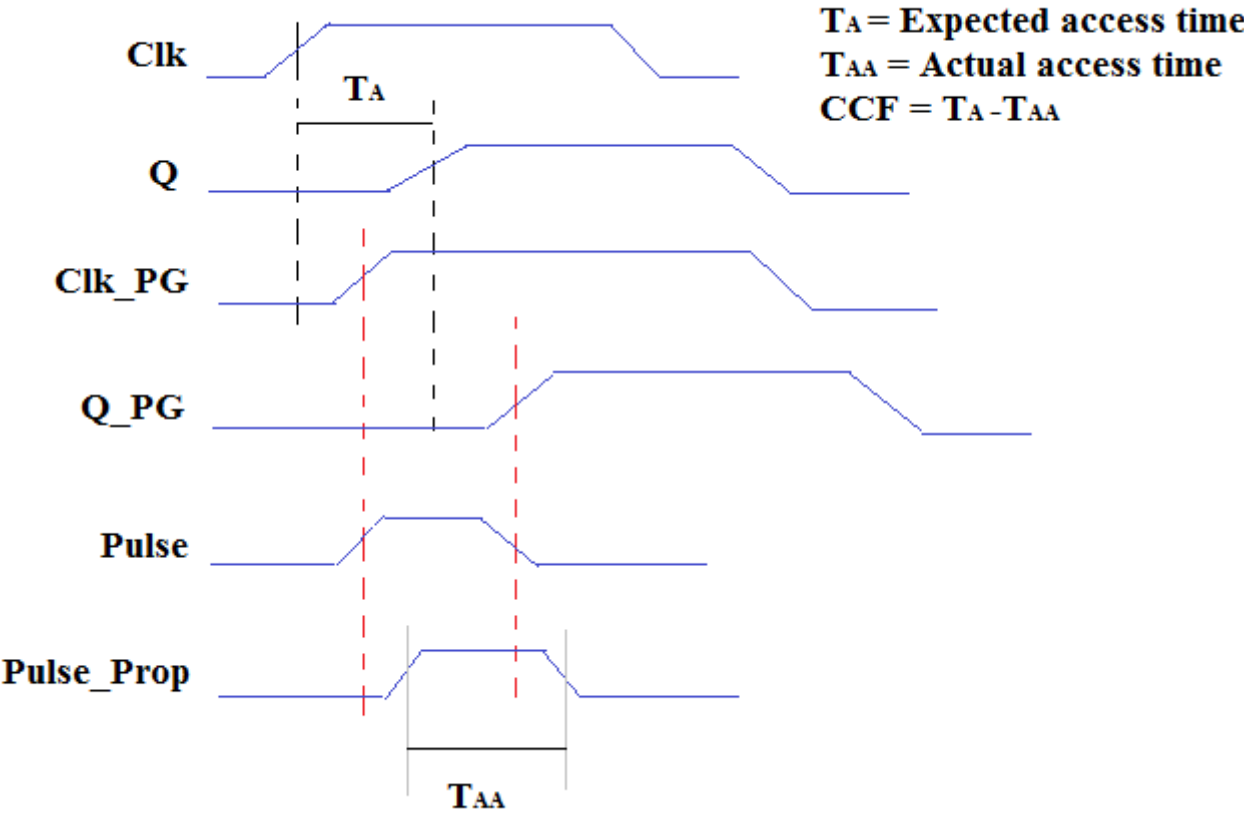


Figure 5.2: Waveforms at different stages of the access time measurement setup

In figure 5.2.2, Clk is the input to the memory and Q is the output of the memory. When clock and Q travels through the intermediate paths, to reach pulse generator, parasitic effects of the path introduces delay in the pulse. Because of this, we get a shifted Q and Clk at the input of pulse generator. Time shifted Clk pulse is referred as Clk_PG and delayed Q is denoted by Q_PG. So, the width of pulse

obtained from pulse generator is not proportional to the actual access time of memory. Now, the pulse obtained by the pulse generator is sent to the access time characterizer. For different intellectual properties (IPs)(memories), we have a corresponding pulse generator and the output of each pulse generator is multiplexed and sent to ATC. Different paths taken by different pulse generator outputs experience different lags. And the actual path of the ATC from pulse generator accounts for delay.

Thus the pulse which reaches ATC input is not the same as the pulse generated by pulse generator. The access time measured by ATC is close to the expected access time of the memory but it is not exactly same as that the expected one. Therefore, to estimate the expected access time, we need to find the difference between the expected access time and access time obtained by the ATC. This difference between expected and the actual access time is termed as CAD characterization factor (CCF).

CCF tool is used to find to find the effect of all these delays explained above and compile the result for each and every CUT in the memory and for various process, voltage and temperature conditions tat is PVT corners. CCF tool works in the flow given by figure 5.2.2. For each cut in a memory, spice files for 5 paths, given below, must be present so as to complete the simulation these are:

- CK_SourceClk.spice : For the path where clock is fed to the memory
- Q_LSB.spice : Path for Q(LSB) from output of memory to input of pulse generator
- Q_MSB.spice : Path for Q(MSB) from output of memory to input of pulse generator
- CK_Q_PulsePropagation.spice : It accounts for internal of pulse generator
- PulseClk.spice : It is for path from output of pulse generator to input of ATC

5.3 Work done

Before running this tool, there are certain steps which need to be followed by the tester to make the environment ready for the execution of the tool. These are, copying the above spice files in directory for each CUT, the present working directory should have DATA directory in it, pulsegen.spi which contains RC extraction of pulsegen should be present for each CUT, STANDARDCELL.spi for defining the inverter pairs used to balance clock and Q towards pulse generator should be present for every CUT. All these things are to be done manually by the tester. To avoid this, there was a vital requirement for the creation of a wrapper to automate the whole process. This is done by me by writing code for

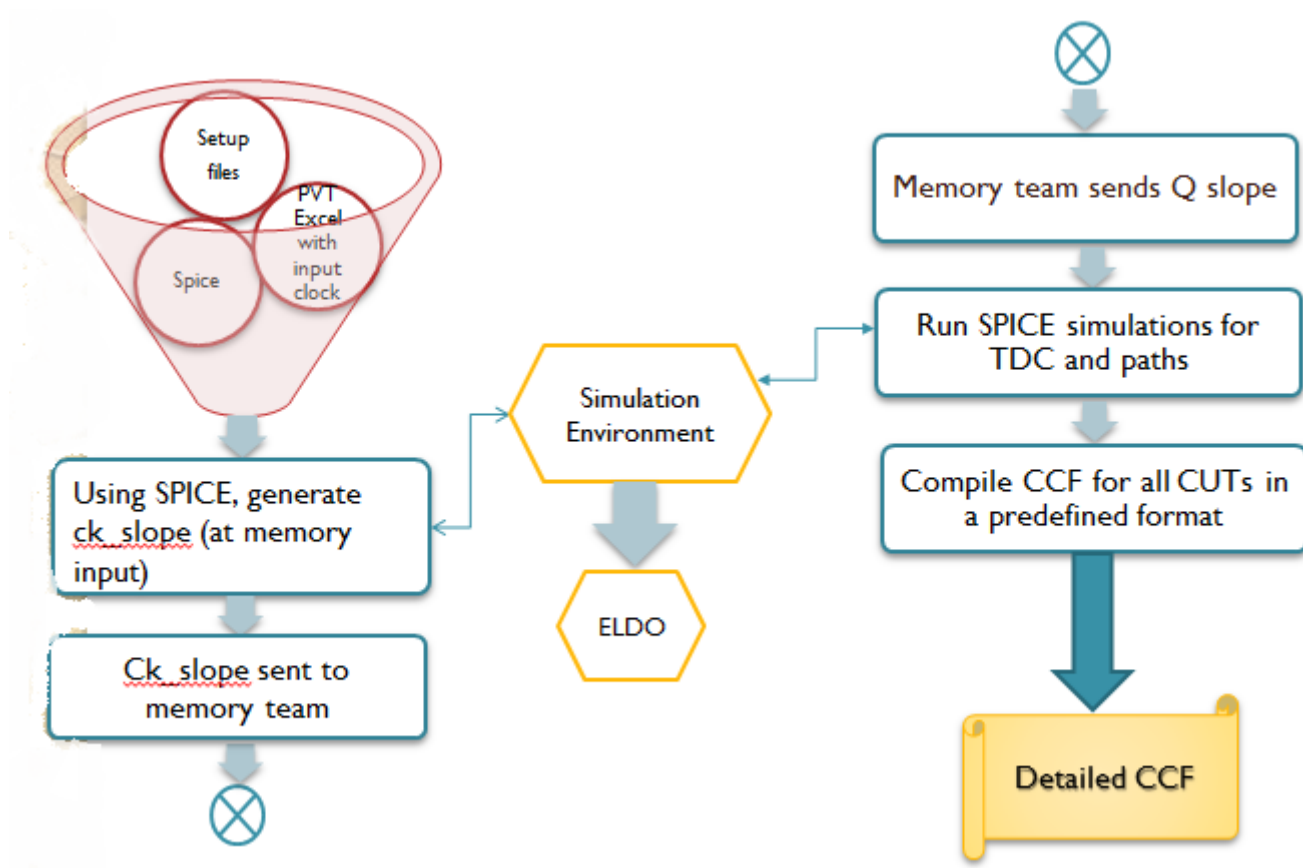


Figure 5.3: CCF Tool Flow

CCF_Wrapper. The flow diagram for its implementation is given in figure 5.3

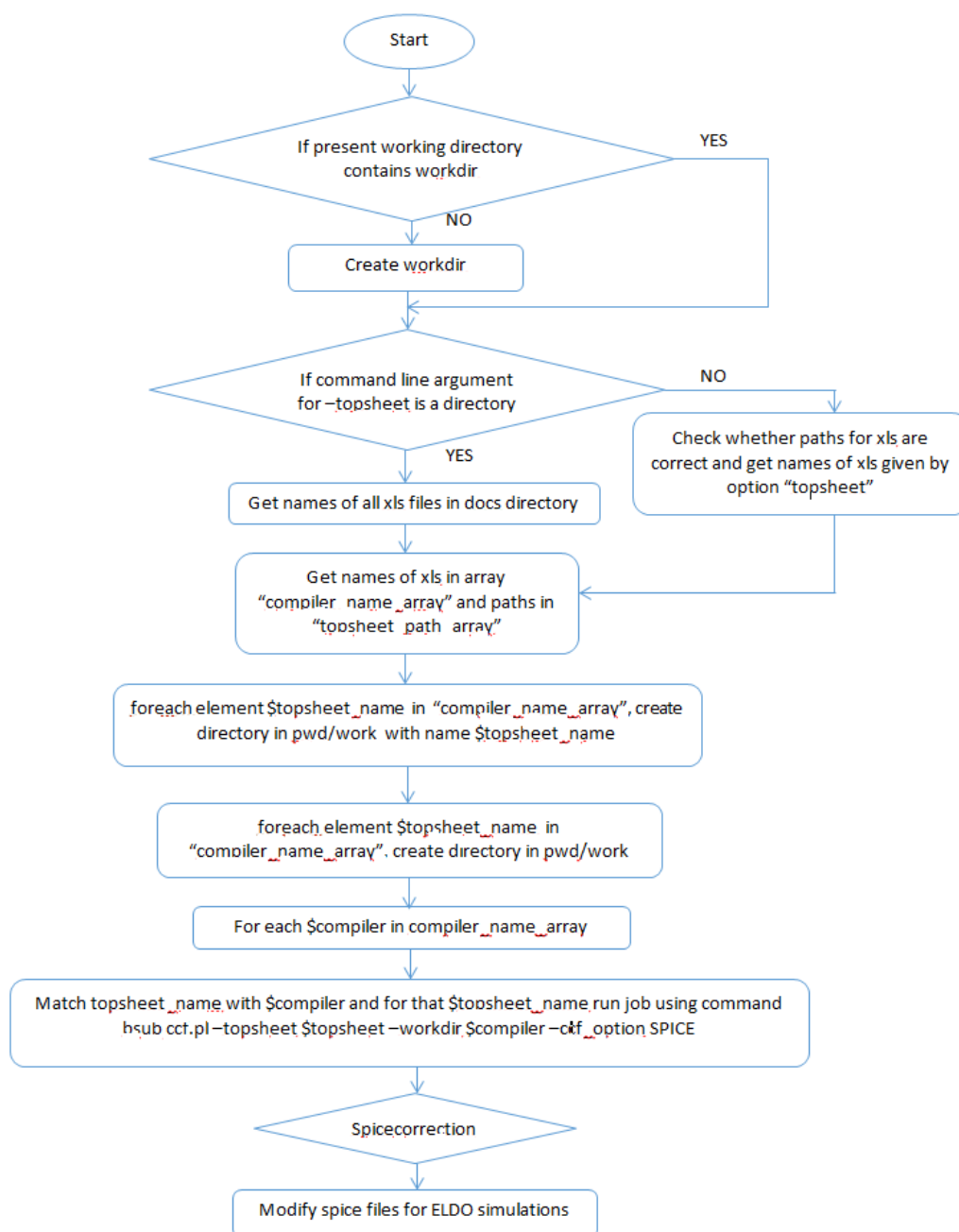


Figure 5.4: Flow diagram for CCF wrapper

Chapter 6

Distribution Record tracker (DRT) reporting Feature

6.1 Introduction

DRT stands for Distribution Request Tracker. The TRnD department creates libraries which can be reused for present and future projects. Other divisions in ST or external customers who need these libraries for their projects make a request for these libraries on a web interface known as DRT (Distribution Request Tracking). Distribution team takes care to process all the DRT requests. There are several types of methods used to process different kind of DRT. At present DRTs are being processed by little manual intervention. But this causes wastage of time, depends upon availability of person, involve high risk of manual mistakes and produce inefficient outputs. The objective is to make the things automated to reduce the manual mistakes and produce efficient outputs. They are as follows states are involved request.

- Submitted
- Assigned
- Pending Approval
- Approved

- Failed
- Cancelled

6.1.1 Submitted

It is a state where requester request for IPs to deliver. Whenever requester access the Distribution Request page with his/her own credentials, clicks the Request link under New label at left side of the page shown in below screenshot, then requester selects type of request which are already given in dropdown list. When requester clicks next button one form will be open with appropriate information to be entered which are needed for Distribution Team. Distributed intellectual are placed by the requester with respect to UPT (unicad product tracking) deliverable labels corresponding with the project name, cc list, downloader list, other required information and user comment.

6.1.2 Assigned

Once requester submits request a mail will be received by distribution team under dedicated mail box. It is just like notification to Distribution Team as new request arrived to process. When request arrives operator analyzes then someone from our team will be responsible to handle the request till end of the process.

6.1.3 Pending approval

When a requester put the request in the DRT , it is now decision based by approval system that this request is in pending approval state. The basic meaning of the state is request is made and approvers are now responsible to approve this request. Every DRT has to passed through this state.

6.1.4 Approved

When the request is correct then it is approved by the approver of that particular request. If the DRT is in this state means the request is ready to be processed.

6.1.5 Failed

This state can only come after approved state. Whenever there is any problem faced while processing the request then DRT is moved to Failed stage. Then the recovery is done after taking some decision.

6.1.6 Cancelled

Whenever there is some issue related to DRT from requester side then the DRT is cancelled and move to cancelled stage. An when the approver reject the request then it is moved to cancelled stage.

6.2 DRT Reporting feature

It may be required by the user that he/she wants to get the DRT requests made earlier. He may give the request ID or certain search parameters obeyed by some requests to get more than one requests. For that, the reporting tool will first get the search parameters from the reporting user interface page as shown in figure 6.2. For example, we can give requester name, action and media and according to that, the tool will report all the requests made by that requester which has action as provided in the reporting page and media through which they were sent.

From this reporting page, a jsp code is called. It fetches all the details provided on the reporting page and calls a java servlet. This java servlet takes all these details and store them in appropriate data structures for java. Details provided in each field is stored in an arraylist which is again part of a hashmap. From there, another java class is called which searches in the mySQL database management system for the requests that fulfils the search parameters of fields provided. For this, dynamic SQL queries are made using java and after making connection to the database, these queries are run on SQL in the background. From there, the requests obeying search parameters are captured and then again through jsp code, it is displayed on the new web page. From there, they can be obtained in xls or csv file as required by the user.

6.3 Work done

My responsibility was to write the code in java to query from MySQL to get the details of the requests which fulfil the search parameters, fetch the details from there and store them into a hashmap of arraylists which are data structure classes defined in java. The code was to be written in such a way that if the request Id is provided, then the search in database is made only on the basis of request Id and all other search features are ignored. And if request Id is not provided, then search is made according to the search features provided by the user.

This can be shown in figure 6.3

6.4 Result

The tool which is used to get the list of all the distribution requests that were requested earlier, was about 20 to 30 seconds to fetch large data from the database. To reduce the time to fetch the data related to the requests, optimized java code and SQL queries are written which has reduced the time to just 6 seconds to fetch the data which earlier used to take 30 seconds. The output of the report page is shown in a tabular form, as given in figure 6.4 which can be converted to csv file and downloaded.

This is QA Database & Server



Search

Product Line:

Deliverable:
Cmos

Customer Project:

Requester:

Action:
PUBLISHDP
UNPUBLISH_DP
UNPUBLISH_PRODUCT
CHANGE_SCOPE
SEND_PRODUCT

From Date(Submitted): (mm/dd/yyyy)
03/02/2016

Till Date(Submitted): (mm/dd/yyyy)
05/02/2016

State:
SUBMITTED
ASSIGNED
APPROVED
PROCESSED
CANCELLED

Media:
FTP
SendFile
SOL
UOL
Service_Repository

DRT Id:



Figure 6.1: DRT reporting user interface

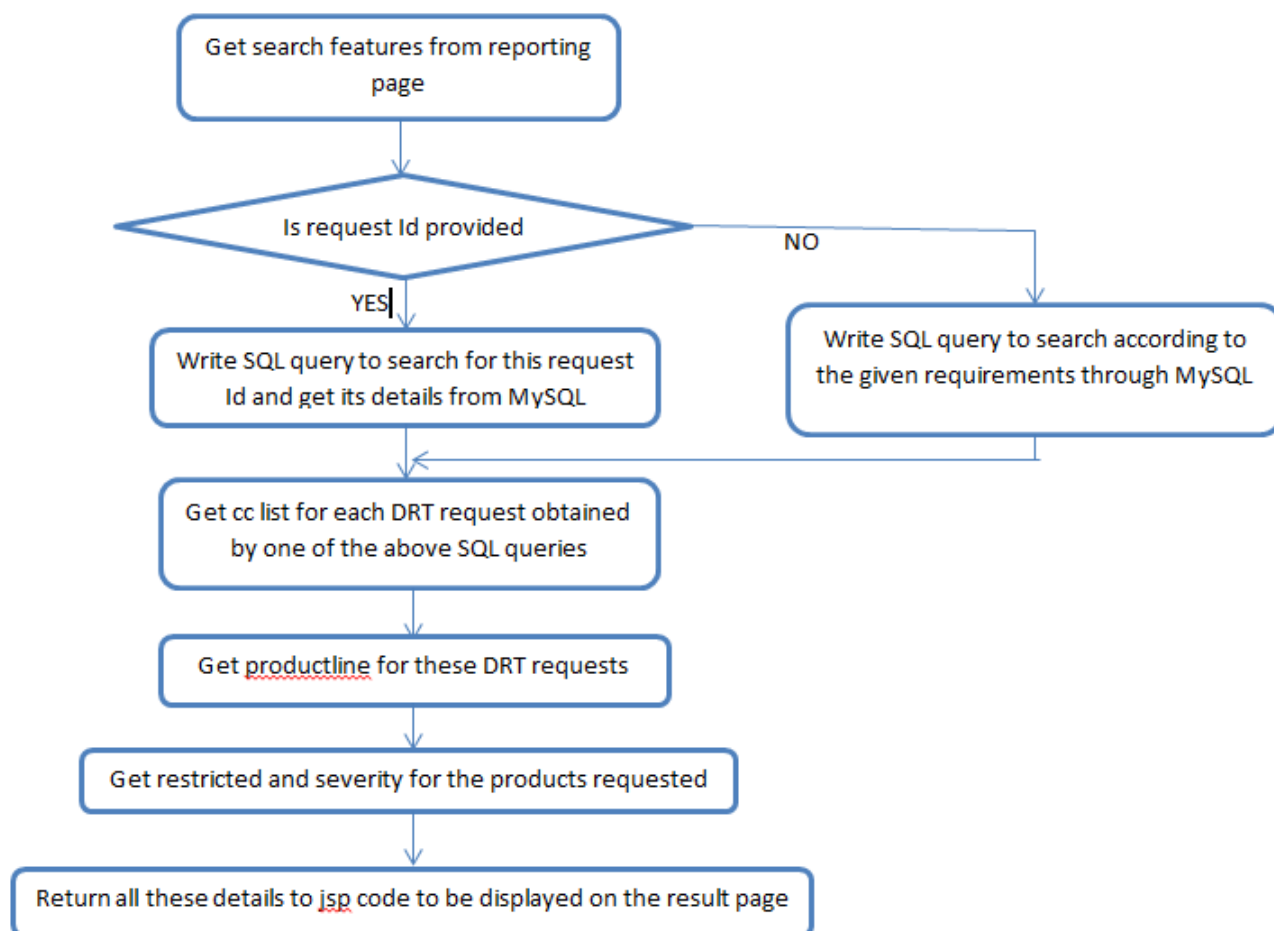


Figure 6.2: Flow diagram showing fetching of request details

STMicroelectronics

showing 33 search results

Request Id	State	Action	Customer Project	Media	Requester	Downloaders	CCList	
13022								
13021	PENDING APPROVAL	SEND_PRODUCT	test	FTP		class gain	temas	
13020								
13019								
13018								
13017								
13016	PENDING APPROVAL	SEND_PRODUCT	test	FTP		class gain	temas	
13015	PENDING APPROVAL	SEND_PRODUCT	test	FTP		class gain	temas	
13014	PENDING APPROVAL	SEND_PRODUCT	test	FTP		class gain	temas	

Figure 6.3: Result for all the customer projects starting with word 'test'

Chapter 7

Conclusion and Future work

7.1 Conclusion

This chapter says about the challenges that I faced while working on this project alongwith what I have learnt and achieved by internship at STMicroelectronics.

The first challenge was to learn programming langauges in which the tools are written. For that, I had to learn Tcl and Cshell for test pattern validation tool. For standard cell block design tool, I needed to learn PERL. The knowledge of PERL that I got while working on standard cell block design tool was used and further enhanced while working on CCF tool. For DRT reporting tool, I had to learn java, java servlets, and SQL.

Another challenge was to read and understand what the tool, which I supposed to mend, is actually doing. It was quite difficult to understand the code written by other. As explained by my guide, the code was functionally reachable but that was not the optimal code and was written in a haphazard way. So, for the first tool that I mended, my guide helped me in understanding the code. After building the concept, I was able to understand the code for all other tools on my own.

The intermediate tool (LSFKit) used for automation of test pattern validation tool through load sharing facility, no longer serves best reliable results. In spite of serving the purpose of automation, it is causing problems like not submitting jobs to load sharing facility (LSF), not providing status of the job, not providing output after completion of the job. So, there was an intense need to replace this tool as it is

obsolete now. This was done by using load sharing facility (LSF). For this, test patterns simulation jobs are fired directly on LSF which handles jobs better than LSFKit. Reliability on the pattern validation tool has been increased.

Also, a temporary and hidden file (.lsfkit) is generated in the home area of the user because of use of LSFKit for automation. When home area doesn't have sufficient space to allow creation of .lsfkit file, LSFKit can't work properly and thus required output is not obtained in project area even though there is no direct link of the tool with home area of the user. To remove this dependency of the home area space available, dependency of Standard cell block design tool on LSFKit has been removed.

Efficiency of the Standard cell generation tool can be increased if some of the libraries can be combined to a single libraries. The libraries that can be combined are provided in the input csv file. But, the reference libraries of the combined libraries have to be same and the total number of cells that a combined library can have should be less than or equal to 1024 (2^{10}). This is so because, the multiplexers used to test libraries have 10 inputs so there can be at most 1024 combinations of the input. Then, the .lib files for libraries are to be changed so that the new library can be used to be tested as a whole. As a move to reduce human dependency while getting the access time of the memory, wrapper of CCF tool was designed. It has been tested on different memory ICs and is being used by the Testchip team of STMicroelectronics.

By using Java and SQL, DRT reporting tool has been optimized to give large number of results in significantly small time.

7.2 Future Work

For getting any product from STMicroelectronics' repository called UPT (Unicad product tracker), any person from STMicroelectronics itself or from a client company needs to create and submit a DRT request for that product. To make the these products available to the requester, these requests are required to be approved by some approvers which may be one of the product managers or project managers or to be auto-approved. This whole process is carried out by the DRT approver system. DRT approver system needs to be modified for different conditions to approve any request. This is going to be the future work for me.

Bibliography

- [1] “LSF documentation by STMicroelectronics”, [online].Available: [http : //www.st.com/docs/computefarm/lfs.php](http://www.st.com/docs/computefarm/lfs.php) 11 Nov 2015
- [2] “LSFKit documentation by STMicroelectronics”, [online]. Available: [http : //www.st.com/docs/computefarm/lfskit.php](http://www.st.com/docs/computefarm/lfskit.php) 3 Sept 2015
- [3] Sung-MO Kang and Yusuf Leblebici, “CMOS Digital Integrated Circuits”*Analysis and Design*, 2nd ed. New York: McGraw-Hill, pp. 234-249 17 Oct 2015
- [4] *Eldo Squirrel Std Cell User Manual*, STMicroelectronics 30 Nov 2015
- [5] “Shell commands ”, [online].Available: [http : //www.webopedia.com/TERM/S/shell.html](http://www.webopedia.com/TERM/S/shell.html) 30 Oct 2015
- [6] “Shell Tutorials by Indiana University”, [online].Available: [https : //kb.iu.edu/d/acva](https://kb.iu.edu/d/acva) 11 Nov 2015
- [7] “Linux commands”, [online].Available: [http : //linuxcommand.org/lts0010.php](http://linuxcommand.org/lts0010.php) 20 Nov 2015
- [8] “Perldoc”, [online].Available: [http : //perldoc.perl.org](http://perldoc.perl.org) 15 Dec 2015
- [9] “Tutorials Point for PERL”, [online].Available: [http : //www.tutorialspoint.com/perl](http://www.tutorialspoint.com/perl) 15 Dec 2015
- [10] “Stack overflow”, [online].Available: [http : //stackoverflow.com](http://stackoverflow.com) 18 Dec 2015
- [11] “Perl Maven”, [online].Available: [http : //perlmaven.com](http://perlmaven.com) 18 Dec 2015
- [12] “Perl Monks”, [online].Available: [http : //www.perlmonks.org](http://www.perlmonks.org) 18 Dec 2015
- [13] “W3School for SQL tutorials”, [online].Available: [http : //www.w3schools.com/sql/](http://www.w3schools.com/sql/) 25 April 2016
- [14] “W3School for Java tutorials”, [online].Available: [http : //www.w3schools.in/java/](http://www.w3schools.in/java/) 29 April 2016
- [15] “Tutorials Point for Servlets”, [online].Available: [http : //www.tutorialspoint.com/servlets/](http://www.tutorialspoint.com/servlets/) 17 April 2016