"Certification of Macros and Memories for VLSI in Nanometer Technology"

A Major Project Report

Submitted in Partial Fulfillment of the Requirements

For the Degree of

Master of Technology

IN

Electronics & Communication Engineering

(VLSI Design)

By

Jani Rachna Dhaval

(05MEC006)

Department of Electronics & Communication Engineering
Institute of Technology
Nirma University of Science & Technology
Ahmedabad-382481
May 2007

"Certification of Macros and Memories for VLSI in Nanometer Technology"

A Major Project Report

Submitted in Partial Fulfillment of the Requirements

For the Degree of

Master of Technology

IN

Electronics & Communication Engineering

(VLSI Design)

By

Jani Rachna Dhaval
(05MEC006)

Under the Guidance of

Mr. Shiv Om Sharma
Senior Design Engineer
S.T.Microelectronics,
Greater Noida

Prof. Amisha Naik
Institute of Technology,
Nirma University,
Ahmedabad

Department of Electronics & Communication Engineering
Institute of Technology
Nirma University of Science & Technology
Ahmedabad-382481
May 2007

ACKNOWLEDGEMENT

I would like to express my sincere thanks to Mr. Jwalant Joshipura, Section Manager, Testing Certification and Design Support (TCD) group, of S.T. Microelectronics, Noida, for providing me an opportunity to carry out my project work at the TCD group.

I would like to express my profound gratitude to my project guide Mr. Shiv Om Sharma, Team Leader, Library Certification Team of S.T. Microelectronics, Noida, for his outstanding support and guidance throughout the course of the Project work.

I would like to express my sincere thanks to Ms. Swati Mattoo and Mr. Apurva Chaure for their constant help during my learning phase. I am especially grateful for their limitless patience and encouragement during trying times.

I would like to express my heart-felt thanks to my internal project guide Mrs. Amisha Naik for her kind guidance and support during the project.

I would also like to express my sincere thanks to the Course Coordinator Dr. N.M. Devashrayee for his support and encouragement at every moment of the Major Project.

My sincere thanks to all my team members of the Library Certification Team especially, Ms. Lipika Parwani, Mr. Varinder Kumar, Mr. RipuTapan Singh, Mr. Suresh Godavarthy, Mr. Rishi Sharma, Mrs. Sonia Sharma, Ms. Neha Agrawal, Mr. Sachin Gupta for their time to time help and guidance during the course of the project.

I would also like to express my sincere thanks to the professors at the Institute of Technology, Nirma University, Ahmedabad especially Prof. N.P.Gajjar, Prof. D.A. Pujara, Prof. Usha S. Mehta and Prof. M. A. Upadhayay for providing me very good foundations which have proved useful throughout my studies and project work.

I would like to thank Ms. Niti Awasthi, Ms. Renuka Patel, Ms. Krupaxi Patel and for their kind support during the project work.

Finally, I would like to thank my family members for their constant encouragements and support.

(Jani Rachna Dhaval)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In the domain of deep submicron (DSM) and nanometer ASIC technologies (180 nm and below), the traditional separation between logical (synthesis) and physical (place and route) design methods often causes a problem—designs cannot meet their realistic timing objectives; creating the well known “timing closure problem”. Timing closure is now considered the biggest area of difficulty for ASIC performance-oriented designs. The underlying reason is that circuit delays are dominated by net delays, which are influenced by the placement of the cells. The traditional fanout-based wireload models, for estimating interconnect delay during synthesis, are considered inaccurate and are the key factor causing the lack of timing predictability between post synthesis and post layout results. It is evident that synthesis and placement technologies must merge to create properly placed and routable designs that meet realistic performance goals.

This thesis described Certification of different libraries. Hereby Certification flow and issues faced during Certification are discussed.

Certification of libraries is defined as the process of certifying an Intellectual Property (I.P.) through various design flows. Inputs to the certification process are the cell views available in libraries. These cell views are schematic, layout, symbol and abstract of the various combinational and sequential logic blocks. To make logical design CORE library is also required. The process starts by generating a gate level netlist using the cells available in the library to be certified. Next, the place and route flow operations like floorplanning, placement, clock tree synthesis (CTS), post CTS optimization, routing and post-route optimization are performed on this netlist to generate the layout at each and every level of flow. Ultimately, the post-routed design is fed to the finish design flow, wherein, the gdscad file (GDS-II) is generated along with some other files which are useful for the signoff flow of the design. By Signoff, one can do the operations like Formality Verification, DRC, LVS, Delay Calculation, Timing Analysis, Back-annotation and many more. The aim of certification is to verify that the libraries when used won’t create violations regarding timing closure, area constraints.

Nomenclature

Different companies might use different terminology for designs and their components. To understand Astro, it is needed to understand how these terms are used at Synopsys:

Term	Definition
Abstract	Abstract is a view which has pin, pin text information and blockage information. In Avanti/Astro flow, an abstract is referred to as **FRAM**.
Block	The meaning of block depends on the context in which the term is used. • In the logical hierarchy, a block is also called a module, logic block, or sub-block. • In the physical hierarchy, a block is also called a cluster, physical block, or floorplan block.
Blockage	Rectangular areas in which cells cannot be placed. A soft blockage allows cells to be added for optimization; a hard blockage does not allow cells to be added for optimization.
Cluster	Partitioning of cells in a design on a physical chip, used to guide the layout tools and optimization commands. Clusters in the hierarchy for the top level must match the hierarchy for the logic modules in the top level. Clusters are sometimes called physical blocks or floorplan blocks. Clusters have hard boundaries.
Core Area	The area where standard cells are placed.
Die Area	The silicon area of the chip.
DRC	The **DRC** performs physical verification of integrated circuit design. It performs dimensions and spacing checks based on process design rules ensuring successful fabrication. Design rule checking differs between the logic and physical domains. • Logic domain: Timing design rule constraints that must be met before further optimization for skew and insertion delay; examples are fanout, transition, and capacitance.

- Physical domain: Wire spacing and so forth.

Filler Cell	Filler cell is defined as a physical-only cell used by silicon vendors to fill open areas in the rows to make sure all power nets are connected. Usually this is done after placement is complete.
Floorplanning	In floorplanning, the estimation of sizes and setting of initial relative locations of the various blocks in the ASIC is done. The floorplan is the physical description of an ASIC.
GDS-II	A **GDS-II** (Graphic Design Station) file is a binary file with the layout information.
Hot Spot	A place in the floorplan that is highly congested with high switching activity leading to more power dissipation.
Layout	Layout is the Physical description of the cell (or macros) with all the layer information. In Avanti flow, a layout is referred to as **CEL.**
LEF	**LEF** is a textual representation of abstract view. It contains site information at library level and cell level information.
Legalized Cell	A placed cell that does not overlap other cells or blockages and is placed on a legal site location. Different die technologies have different rules that specify which cells are legal and where.
Legalized Placement	A placement in which no row contains more cells than the number of sites available. All cells are placed on legal locations, snapped to the site array grid.
.lib	.lib contains the text representation of a unidata with some additional library information. **TIM** : .lib representation in Avanti, **TLF**: .lib representation in **SE.**
LVS	**LVS** ensures equivalence between layout and schematic in terms of devices and connectivity.
Netlist	Netlist is defined as the description of the logic cells and their interconnection. The netlist is the logical description of an ASIC.
Pin	The input and output of cells within a design (such as gates and flip-flops). The ports of a sub-design are pins within the parent design.

Placement	Placement defines the location of the logic cells within the flexible blocks and sets aside space for interconnection to each logic cell.
pLib	pLib is Physical library which contains library information.
Port	The inputs and outputs of a design. Port direction is designated as input, output, or inout.
RC Correlation	The process of comparing RC parameters before and after routing and then tuning RC parameter values to reflect the true wire delays.
Schematic	Schematic is the representation of a cell at the transistor level. **CDL** is the text description of the Schematic. E.g., A **SPICE** code for an inverter.
sLib	sLib is the text representation of the symbol view.
Spare Cell	A logic gate added to a design to allow quick logic changes to be made without replacing all mask layers of the chip. Spare cells exist in the netlist and the input PDEF. Spare cells are not the same as filler cells.
SPF	Standard Parasitic Format describes the interconnect delay and loading due to parasitic resistance and capacitance. There are three forms of SPF: regular SPF, reduced SPF and detailed SPF.
Symbol	Symbol is a pictorial representation of a cell.
Unidata	Unidata is a view which contains timing and constraints information, input capacitance information, logical information, **SDF** (Standard Delay Format) and vital mapping for each cell in the library.
Unit Tile Cell	The smallest unit of placement. The width of the unit tile is the metal-2 pitch. The height is the standard cell height, unless you are using double-, triple-, and multiple-height cells. Each standard cell library contains a unit tile cell. There is no need to redefine the unit tile cell when using double-, triple-, and multiple-height cells.
View	A View is defined as a representation of a cell in various levels of abstraction.

About ST Microelectronics

STMicroelectronics is a global independent semiconductor company and is a leader in developing and delivering semiconductor solutions across the spectrum of microelectronics applications. An unrivaled combination of silicon and system expertise, manufacturing strength, Intellectual Property (IP) portfolio and strategic partners positions the Company at the forefront of System-on-Chip (SOC) technology and its products play a key role in enabling today's convergence trends.

ST is one of the world's largest semiconductor companies. In 2004, ST's net revenues were US$8,760 million and net earnings were US$601 million.

According to the most recent data from independent sources, ST is the world's leading supplier of application-specific analog ICs overall with number one rankings in various segments within this field, including wireless ASICs, computer peripheral ASICs and automotive ASSPs. ST is also the leader in MPEG-2 decoder ICs, and ASICs/ASSPs overall, including a number one position in digital consumer ASSPs. Additionally, ST is ranked at number two for discrete products, and in the memory market, ST holds third rank in NOR Flash ICs. In application segments overall: ST is number one for ICs in set-top boxes; at number two in smart cards, at number three in automotive; and at number four in wireless.

The Company's products are manufactured and designed using a broad range of fabrication processes and proprietary design methods. To complement this depth and diversity of process and design technology, the Company also possesses a broad intellectual property portfolio that it has used to enter into cross-licensing agreements with many other leading semiconductor manufacturers.

ST has developed a worldwide network of strategic alliances, including product development with key customers, technology development with customers and other semiconductor manufacturers, and equipment and CAD development alliances with

major suppliers. By augmenting its rich portfolio of proprietary technologies and core competencies with complementary expertise from a variety of carefully chosen strategic partners, ST has developed an unsurpassed capability to offer leading-edge solutions to customers in all segments of the electronics industry.

The Company currently offers over 3,000 main types of products to more than 1,500 customers, including Alcatel, Bosch, DaimlerChrysler, Ford, Hewlett-Packard, IBM, Motorola, Nokia, Nortel Networks, Philips, Seagate Technology, Siemens, Sony, Thomson and Western Digital. Approximately two-thirds of ST's revenue is derived from differentiated products, a combination of dedicated, semi-custom and programmable products designed to suit a specific customer or a specific application and therefore having high system content. This result reflects ST's exceptionally early recognition of the importance of system-on-chip technology, which is a key for addressing the fast growing market for convergence products, and the success of the strategies it developed to ensure its leading position in this key emerging field.

The ST group was formed in June 1987 as a result of the merger between SGS Microelettronica of Italy and Thomson Semiconducteurs of France. In May 1998, the company changed its name from SGS-THOMSON Microelectronics to STMicroelectronics.

Since its formation, the Company has significantly broadened and upgraded its range of products and technologies and has strengthened its manufacturing and distribution capabilities in Europe, North America, and the Asia Pacific region. This capacity expansion is an ongoing process with the upgrading of existing facilities and the creation of new 8-inch, sub-micron fabs around the world. ST currently has five 8-inch fabs in operation in: Rousset (France); Agrate Brianza, R2 (Italy); Crolles (France); Phoenix (Arizona); Catania (Italy); and Singapore. Furthermore, a new 12-inch manufacturing facility is currently under construction in Catania; and the company is now ramping up production from a 12-inch pilot line called Crolles2, in partnership with Philips and Freescale Semiconductor. The Crolles2 operation is also host to the joint

development program between the three companies to develop leading-edge CMOS process technology down to the 32nm node, in conjunction with TSMC for process alignment.

Corporate Headquarters, as well as the headquarters for Europe and for Emerging Markets, are in Geneva. The Company's U.S. Headquarters are in Carrollton (Dallas, Texas); those for Asia/Pacific are based in Singapore; and Japanese operations are headquartered in Tokyo.

To guarantee continued technological development and consistently offer customers true leading-edge products, ST each year invests a significant proportion of its sales in R&D and capital expenditures. ST has maintained an average R&D spend of close to 17% of revenue over the last three years, and an average of more than 15% of revenue from the date of the company's creation. In 2004, ST spent US$1,532 million in R&D, or 17.5% of its net revenues. Additionally, ST filed 711 patent applications in 2004, maintaining its track record as one of the industry's most prolific inventors. The new inventions protected with these filings covered a wide range of technologies, products and applications, in line with the broad range supplier mission of the Company. ST is also active in numerous collaborative research projects worldwide as well as playing a key role in Europe's advanced technology research programs such as MEDEA+, and its predecessors, MEDEA and JESSI.

ST's technical, marketing and manufacturing strengths are matched and further enhanced by an unswerving commitment to Total Quality and Environmental Management (TQEM) that has earned prestigious awards around the world. Since 1991, the Company's sites have received more than 70 awards, of which more than 40 were for environmental issues.

In addition to ST's previous honors for quality over the last few years - which include the Malaysian Prime Minister Quality Award, the Malta Quality Award and the European Quality Award - STMicroelectronics Asia Pacific was the winner of the 1999 Singapore Quality Award for Business Excellence; and the Company's US

subsidiary STMicroelectronics Inc, was awarded the 1999 Malcolm Baldrige National Quality Award, the highest level of national recognition for quality that a US company can receive. And in 2000, ST's Moroccan operations received the Morocco National Quality Award.

ST's commitment to environmental responsibility has resulted in substantial reductions in the consumption of energy, water, paper and hazardous chemicals, increased recycling of waste products and a significant cut in CO2 emissions. In 1999, ST received the United States Environmental Protection Agency's Climate Protection Award for its outstanding accomplishments in protecting the Earth's climate. In 2000, ST was ranked first in environmental management among 14 semiconductor companies by Innovest Strategic Value Advisors and received the only AAA ranking in eco-efficiency. Also in 2002, ST received the Seal of Sustainability from the Sustainable Business Institute and ST's back-end Malta plant received the Management Award for Sustainable Development as part of the European Awards for the Environment 2002, organized by the European Commission Directorate-General Environment. And most recently, ST was awarded the 'Best Industrial Renewable Energy Partnership' as part of the European Commission's 'Campaign for Take-Off for Renewable Energy Awards 2003.

Since December 8, 1994, when ST completed its initial public offering, the Company has been quoted on the New York Stock Exchange (NYSE: STM) and on Euronext Paris (NXT: STM) and since June 1998 also listed in Milan on Borsa Italiana (MIB: STM). The Company has around 900 million outstanding shares, over 69% of which are publicly traded on the various stock exchanges. The balance of the shares is held by STMicroelectronics Holding II B.V., a company whose shareholders are CASSA DEPOSITI E PRESTITI and Finmeccanica of Italy, and a French consortium comprising Areva and France Telecom.

Chapter -1

Introduction

CHAPTER 1

INTRODUCTION

1.1 Introduction To Certification:

Today's Systems on Chips (SoCs) are requested to be more and more complex whereas the associated Time-To-Market is getting shorter and shorter. Such SoCs are typically made up of building blocks, some of them actually providing innovative functions, others addressing more classical ones. The design data for such building blocks is provided in the form of a Library Database. This database can be reused in various SoCs. The overall process of system design begins with identifying the system requirements. They are the required functions, performance, power, cost, reliability and development time for the system.

The level of circuit performance which can be reached within a certain design time strongly depends on the efficiency of the design methodologies, as well as on the design styles. The choice of the particular design style for a VLSI product depends on the performance requirements, the technology being used, the expected lifetime of the product and the cost of the project.

The technology always impacts the design process. The design flow starts with laying down the specifications, RTL is generated and Macro and Memory Libraries are generated. These Macro and Memory libraries contain information regarding

- Cell characteristics (Cell and pin names, area, delay arcs and pin loading).
- Design rule Constraints (DRC).
- Operating Conditions.
- Wire Load Models.

All this information is used

- To implement the design function.
- To calculate timing values and path delays.
- To calculate power consumption.

The operating conditions specify the characteristics like operating temperature variations, supply voltage variations and process variation. The wire-load models estimate the effect of wire length and fan-out on resistance, capacitance and area of nets.

All these characteristics are used to make a physical description of the design. The physical specification is used to define how a particular part has to be constructed to yield a specific structure and hence a behavior.

The actual transistor sizes and physical capacitances are calculated to a high degree of accuracy. This information is mapped back to the structural description to place more accurate delays on gates and nets. This back-annotation is crucial to obtain accurate performance estimations.

All these comprise the process of certification, wherein the logical and physical characteristics of the design are verified for equivalence using the various ASIC design flow operations such as floorplanning, placement, clock tree synthesis (CTS), post CTS optimization, routing and post-route optimization followed by the signing-off of the design.

1.2 IO Libraries:

IO Libraries consists of a group of cells called I/O buffers. Those cells are already designed in a specific process technology. Which contain Physical/ logical / timing/ electrical information about the cells. I/O buffers are designed to interface the off-chip signals to inside chip environment and vice-versa and to provide Power/Ground Signals to the Chip. I/O's are placed on the periphery of the chip .Any signal which comes from off-chip environment (external voltages are at a typical voltage of 2.5V, 3.3V or 5V) into the chip, must be checked by I/O for any discrepancy in its behavior other than defined by the core for that particular signal. If I/O finds any signal defying the behavior expected from it, it modifies the signal so as to ensure proper functioning of chip. I/O's also act as protection devices for the core. I/O also scans the signal which is going from core to off-chip world.

1.3 Memories:

In the present-day SoC, approximately 50% to 60% of the SoC area is occupied by memories. Even in the modern microprocessors, more than 30% of the chip is occupied by embedded cache. So certification of these libraries is very important.

These contain Memories of Various Architectures like SRAM, DRAM, ROM. The basic building blocks are already designed in a specific process technology.

1.4 Analog and Mixed Signal Libraries:

These contain IPs providing some innovative functions like PLL, DAC, USB, high speed I/Os. The primary design issue in analog circuits is the precise specifications of various parameters. For SoC design, the design of analog circuit must meet the specifications of a significantly large number of parameters to ensure that the analog behavior of these circuits will be within the useful range after manufacturing. These can be implemented as full custom hand crafted cells or in a semi-custom manner using the CORE libraries.

1.5 CORE Libraries:

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

Chapter – 2

Place and Route Flow

CHAPTER 2

PLACE AND ROUTE FLOW

2.1 Overview:

The Certification is a process of applying various ASIC design flow operations on the design, to verify whether the design is fit to be used in the particular design environment.

The operations include generation of design, binding the netlist with the standard cells, floorplanning, placement and routing, clock tree synthesis, parasitic extraction, Design Rule Check (DRC), Layout Vs Schematic (LVS) and many more.

Purpose of library certification is to meet library CAD view quality demands w.r.t. cell-based design flow, while meeting cycle time constraints and following correct certification processes. Develop new methods to improve the existing methodology and provide support to customers.

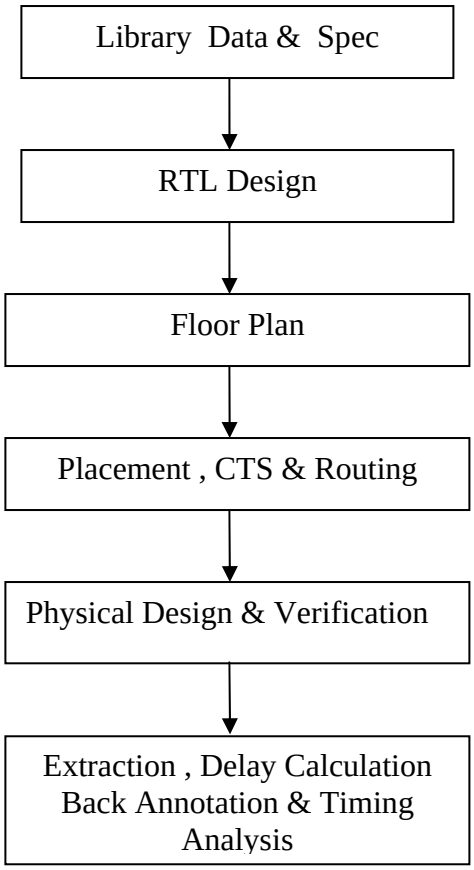


Fig. 2.1.1 Certification Execution Process

2.2 Methods of Certification:

Depending upon the updates done in the new versions of the library, the certification can be classified in four different categories:

- **Method Zero Certification:** Method 0 is the most preliminary method of certification. The method 0 is applied when there are updates in Release notes, Known Problems and Solutions document (KPS) or UserManual.

  Also, at times the report of the product is submitted depending on the logs during the development stage checked by certification team.
- **Method One Certification:** Method-1 certification is used when there are updates in .lib file but the changes made in .lib file are not reflected in the layout and post-layout calculations. This could be related to the timing updates. Since .def and .lef files are not available, wireload models are used to generate .dspf file. This is then used in delay calculation in order to generate sdf file which will be then used for back-annotation using ncsim/ncverilog. Tests run during this method are:
  - Read sdf to check the cross-compatibility of stf and map file.
  - Pre-layout back-annotation for Verilog and Vital models.
  - Tag Checks.
- **Method Two Certification:** Method-2 certification is used when there are back-end updates in the standard cell library. Thus, during the Method-2 the Design Rule Check (DRC) and Layout Versus Schematic (LVS) checks are performed.
- **Method Four Certification:** Method-4 certification is the most commonly used method when the library is to be certified for the first time. This means during this method, the entire flow is to be exercised.

2.3 Design Entry:

RTL design and Netlist can be created using plugin **“Fusion”** or manually. Connectivity of pins has to be taken care as per the manual for special connections. This provision is provided within **“Special Connections”** feature of Fusion.

2.4 PAR Flow:

The physical design of ASICs is normally divided into System partitioning, Floorplanning, Placement and routing.

System partitioning:

Microelectronic systems typically consist of many functional blocks. If a functional block is too large to fit in one ASIC, we may have to split, or **partition,** the function into pieces using goals and objectives that we need to specify.

• Goal Partition a system into a number of ASICs

• Objectives Minimize the number of external connection between the ASICs.

Floorplanning:

The input to a floorplanning tool is a hierarchical netlist that describes the interconnection of the blocks, the logic cells within the blocks; and the logic cell connectors. The netlist is a logical description of the ASIC; the floorplan is a physical description of an ASIC. Floorplanning is thus a mapping between the logical description and the physical description.

The goals of floorplanning are:

• arrange the blocks on a chip

• decide the location of the I/O pads

• decide the location and number of the power pads

• decide the type of power distribution

• decide the location and type of clock distribution.

Placement:

The goal of a placement tool is to arrange all the logic cells within the flexible blocks on a chip. Ideally, the objectives of the placement step are to

• Guarantee the router can complete the routing step

• Minimize all the critical net delays

• Make the chip as dense as possible

• Minimize power dissipation

• Minimize cross talk between signals

• Minimize the total estimated interconnect length

• Meet the timing requirements for critical nets

• Minimize the interconnect congestion

Routing:

Routing is of two kinds - Global routing and detailed routing

Global routing

A global router does not make any connections, it just plans them. One can typically global route the whole chip before detail routing the whole chip. There are two types of areas to global route: inside the flexible blocks and between blocks.

The goal of global routing is to provide complete instructions to the detailed router on where to route every net.

The objectives of global routing are

• Minimize the total interconnect length

• Maximize the probability that the detailed router can complete the routing

• Minimize the critical path delay

Detailed routing

The global routing step determines the channels to be used for each interconnect. Using this information the detailed router decides the exact location and layers for each interconnect.

The goal of detailed routing is to complete all the connections between the logic cells. The most common objective is to minimize one or more of the following:

- The total interconnect length and area.
- The number of layer changes that the connections have to make
- The delay of critical paths minimizing the number of layer changes corresponds to minimizing the number of vias that add parasitic resistance and capacitance to a connection.

2.4.1 Avanti Design Flow:

Avanti Design flow is used for place and routing. Which is shown below:

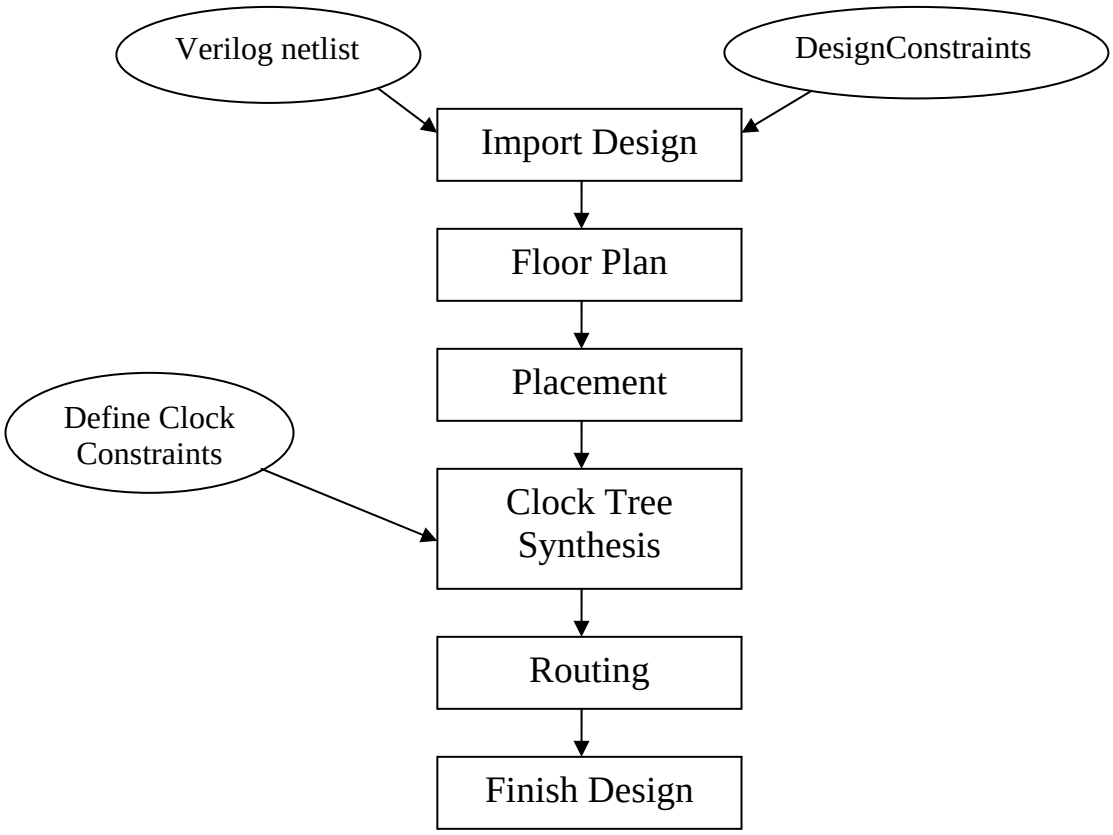


Fig. 2.4.1.1 Avanti Design Flow

2.4.1.1 Design Setup:

This first step performs:

- design library creation if non existent
- addition of RefLibs in case of library creation
- Attach TLUplus files to library
- generates dont't use and delay cell list in line with Ref Libraries in COLLECT_DATA.
- import the verilog netlist .
- flattening (expand)
- creation of CEL view (for P&R task)
- bind the netlist to cell
- preserve hierarchy
- create initial floorplan
- load converted tdf file (optional)

- load converted bond file (optional)
- make a Power/Ground Connection for All Cells
- load timing constraints file
- performs timing check (file ttiming_check.rpt in REPORT directory)
- do Timing Analysis with zero Wireload
- save the resulting cell in <TopCellName>_fp

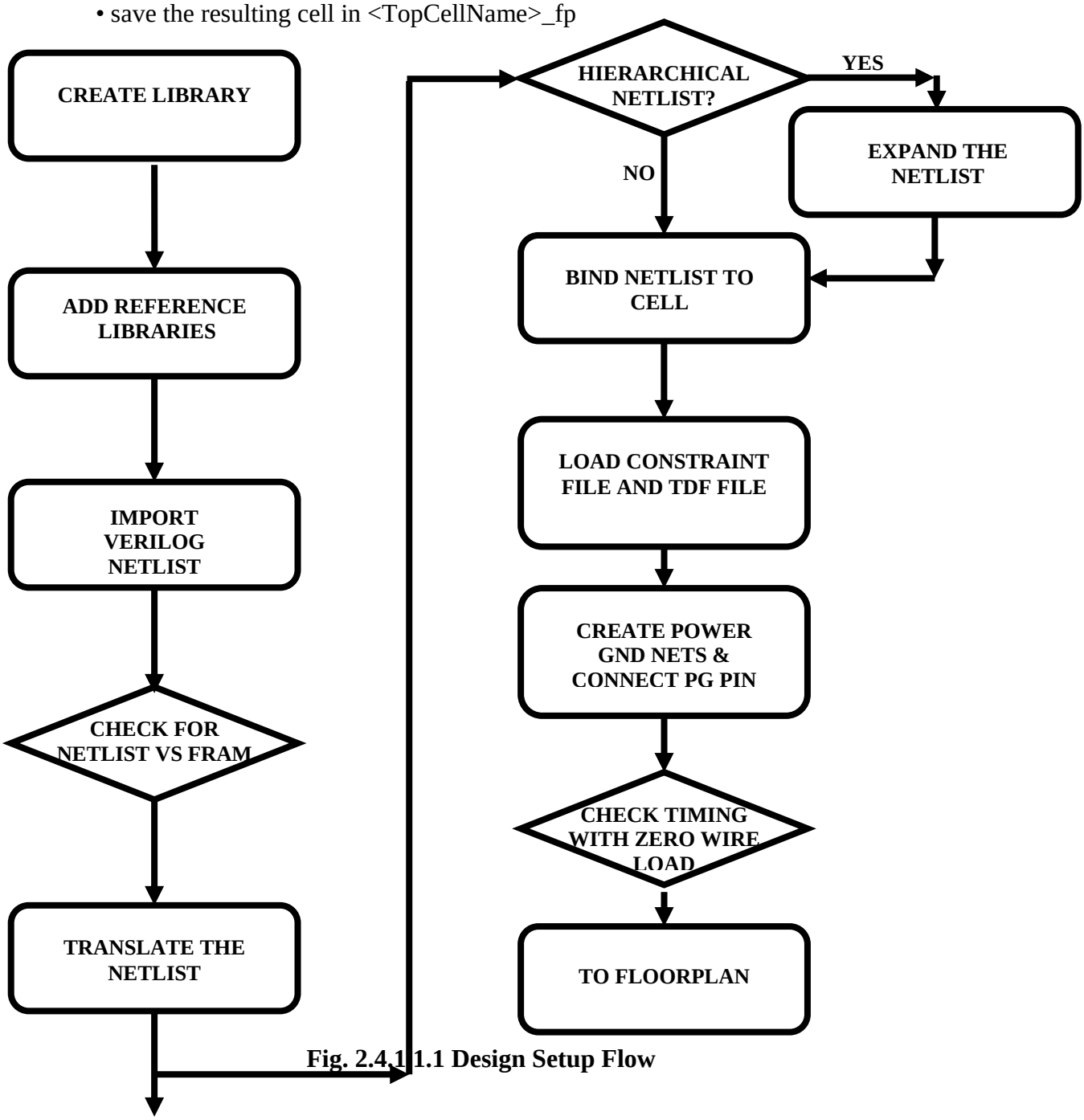


Fig. 2.4.1.1.1 Design Setup Flow

2.4.1.2 Floorplan:

Floor planning is the exercise of arranging blocks of layout within a chip to minimize area or maximize speed. It can be called, allotment of the silicon real estate to different blocks in the design. The input to the floor planning is typically the dimension of the chip, the area constraints. It is almost always necessary to place hard macros manually.

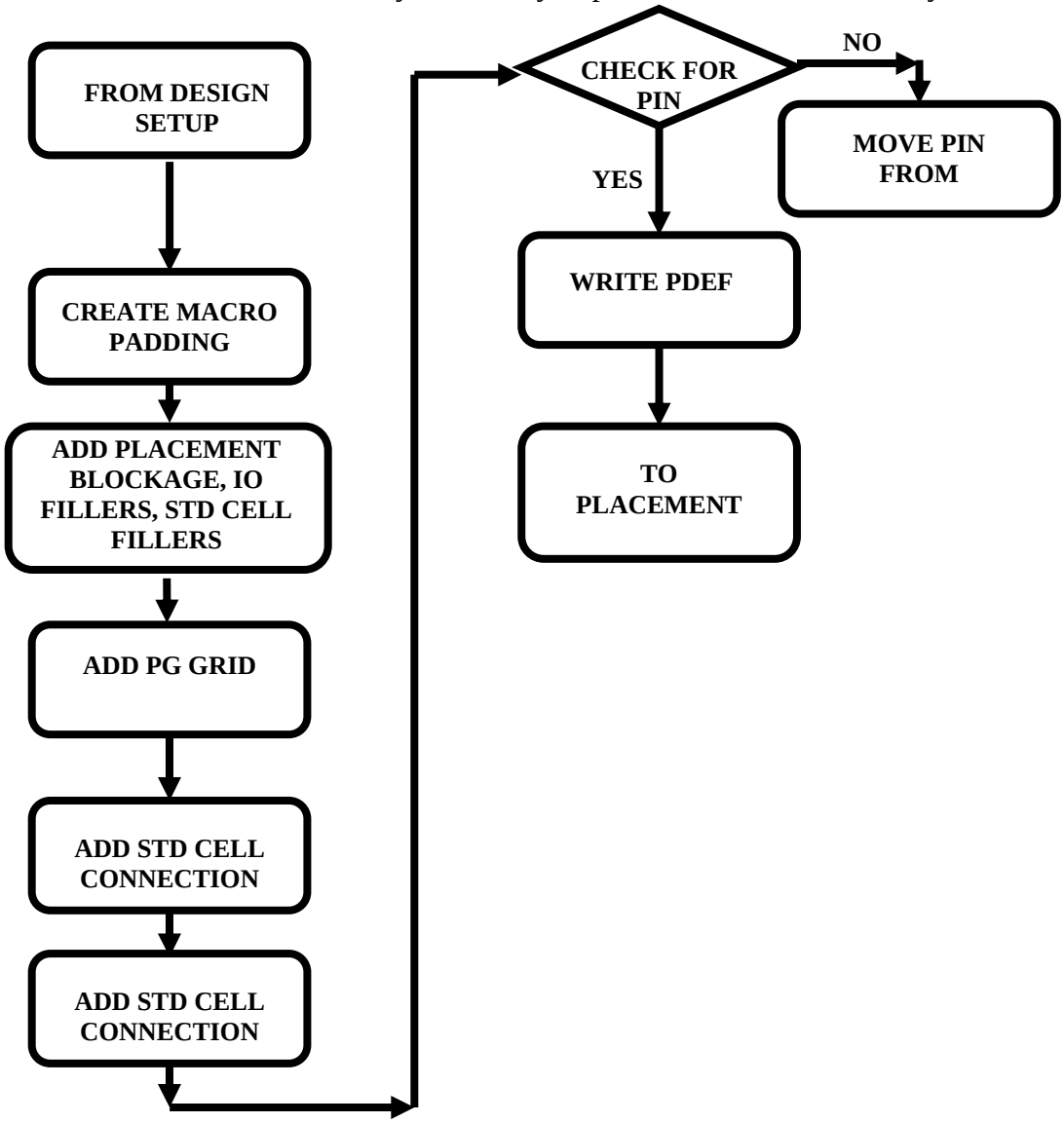


Fig 2.4.1.2.1 Floorplan Flow

The Floorplan executes following sequence:

- pins for block are snapped on wiretracks.
- add IOFILLER (padFiller) for a top + PG connect IOfiller
- add placement blockage around each Macro Cell

- add Power and Ground Grid
- pre-route the standard cell rails.
- performs PGConnect of Pads and IOFillers for a top
- check and repair the placement of pin for a block

2.4.1.3 Placement:

Placement is the task of placing Logic Cells or Gates in some pre-allocated rows to minimize area or cycle time. Cells are placed adjacent to each other so as to minimize the interconnections. Placement reads the PDEF, for placing the modules.

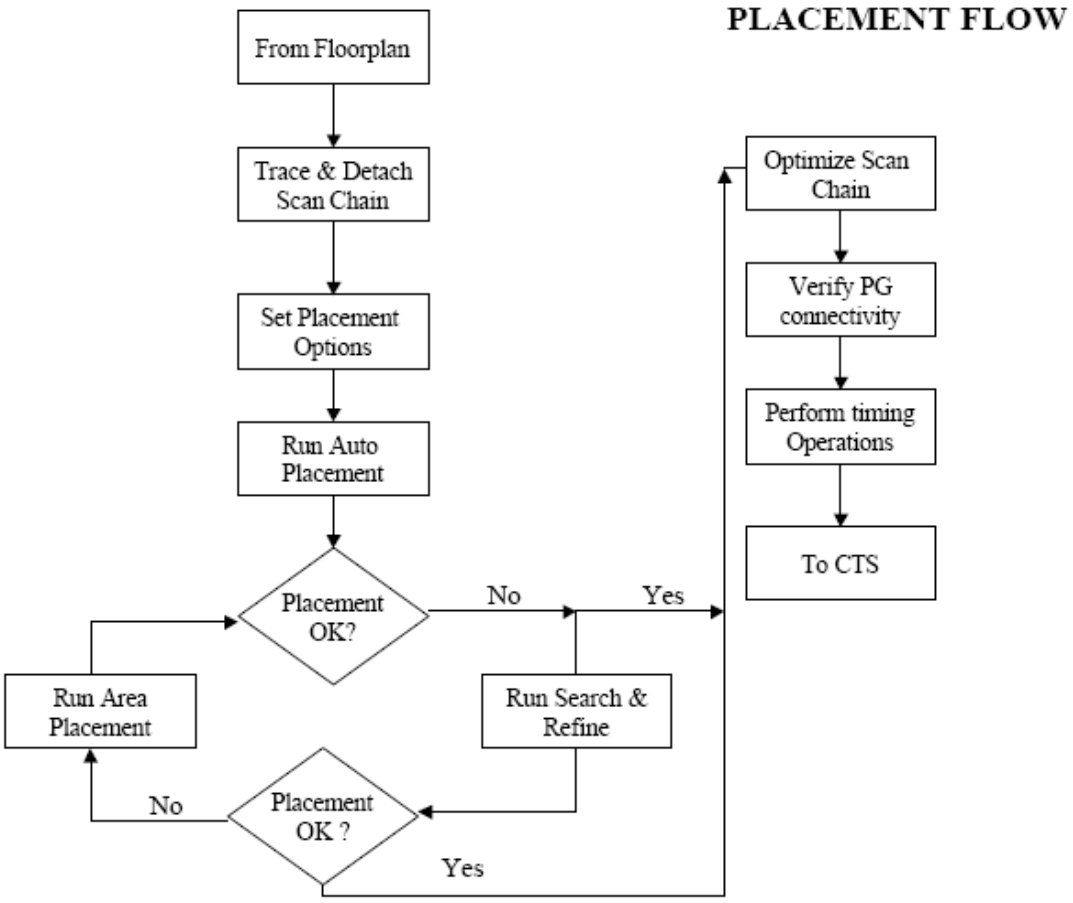


Fig. 2.4.1.3.1 Placement Flow

2.4.1.4 Clock Tree Synthesis:

Method constructs a tree of clock buffers with some suitable geometry such that modules those communicate with each other receive well-defined clocks. Gated CTS processes the active gated clock net from root to the clock pin. Given the root net Gated CTS traverses through all the gated subnets, determines the structure of buffer trees & performs CTS to optimize skew. Need clock constraints (min & max insertion delay, max transition, max skew, Buffers & Inverters to be inserted),as an input to CTS.

The TreeSynthesis.cmd script performs:

- High Fanout Net Synthesis (non gated CTS by default)
- Timing analysis pre-cts
- Clock Tree synthesis (gated CTS by default)
- Pre-route clock skew report
- Route clock + global route/track assign for signals
- Post-route skew report

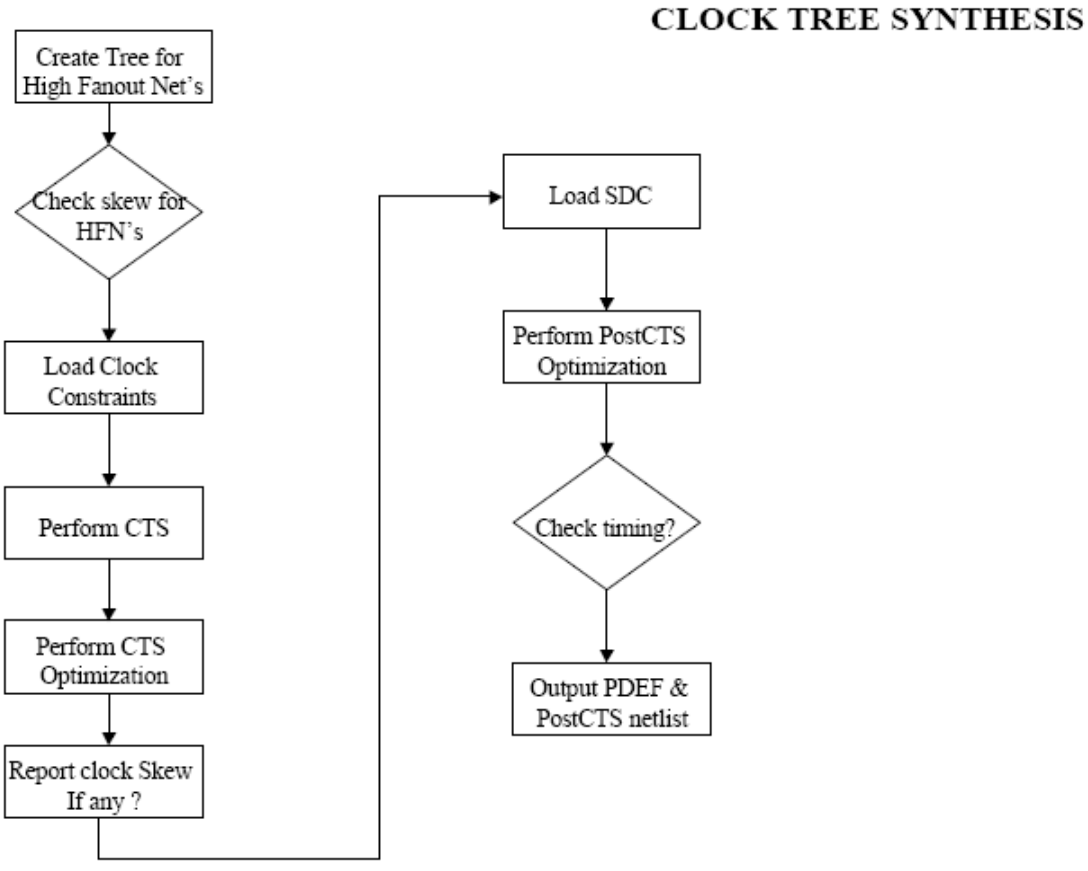


Fig.2.4.1.4.1 Clock Tree Synthesis

2.4.1.5 Routing:

A router takes a module placement and a list of connections and connects the module with wires. **Global Routing** maps general pathways through the design for each unrouted net. The global router uses a two dimensional array of GR cells to model the demand & capacity of Global Routing. **Detail Routing** uses the general pathway suggested by global routing & track assignments to place paths & contacts in order to route the nets.

The Route Design executes following sequences:

- pre-route the padring, based on the PG Connect commands
- buffers insertion on nets with length > threshold defined in Project_variables
- Add Diodes protection on pins for a block
- Route the clock tree nets
- route the design
- post-route timing report
- postRoute clock skew report

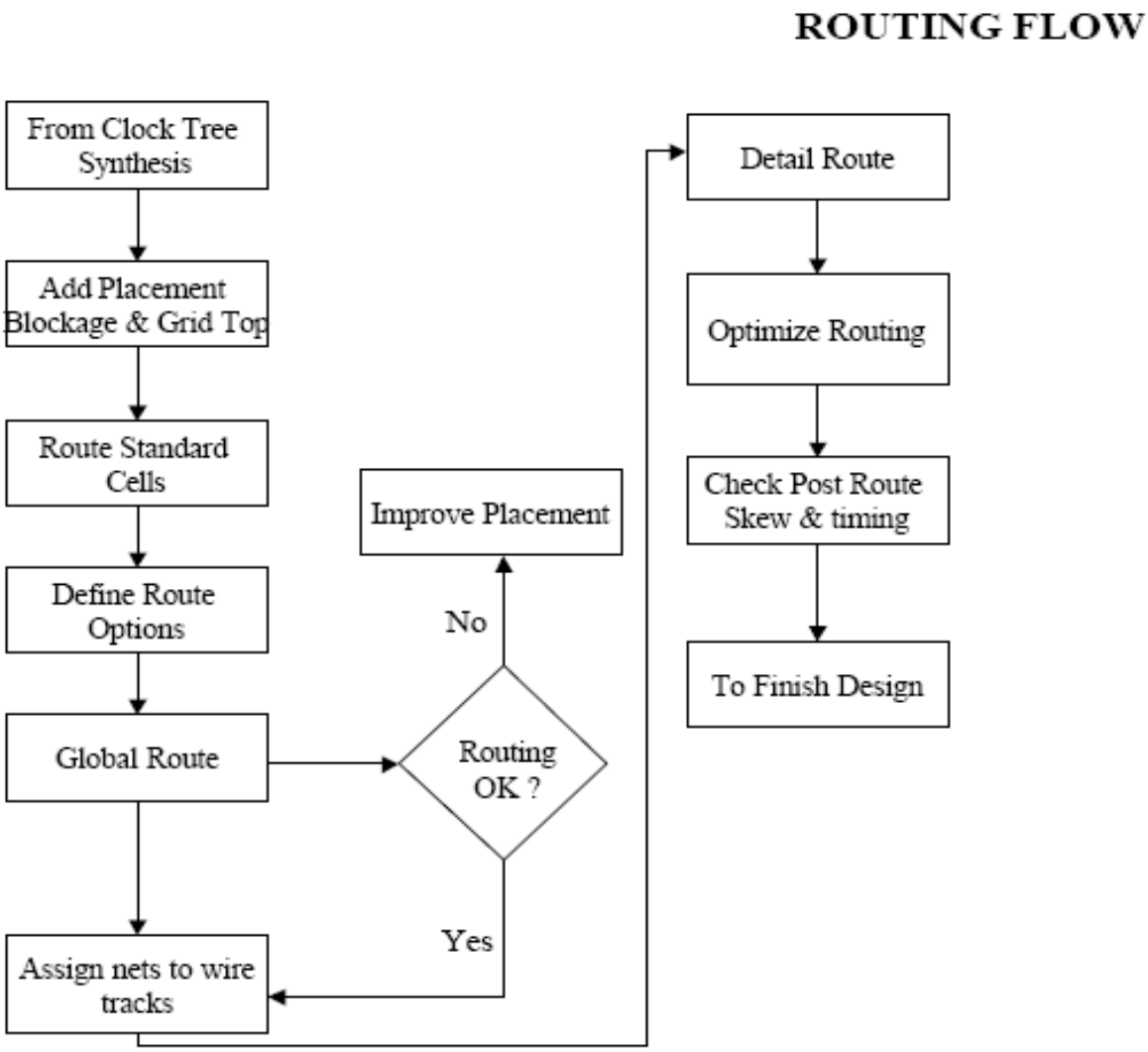


Fig. 2.4.1.5.1 Routing Flow

2.4.1.6 Finish Design:

This is the last step of PAR flow. This operation generates verilog netlist for timing analysis and LVS. It also stream out GDSII and defout. It verifies power and ground connections.

FLOW FOR FINISH DESIGN

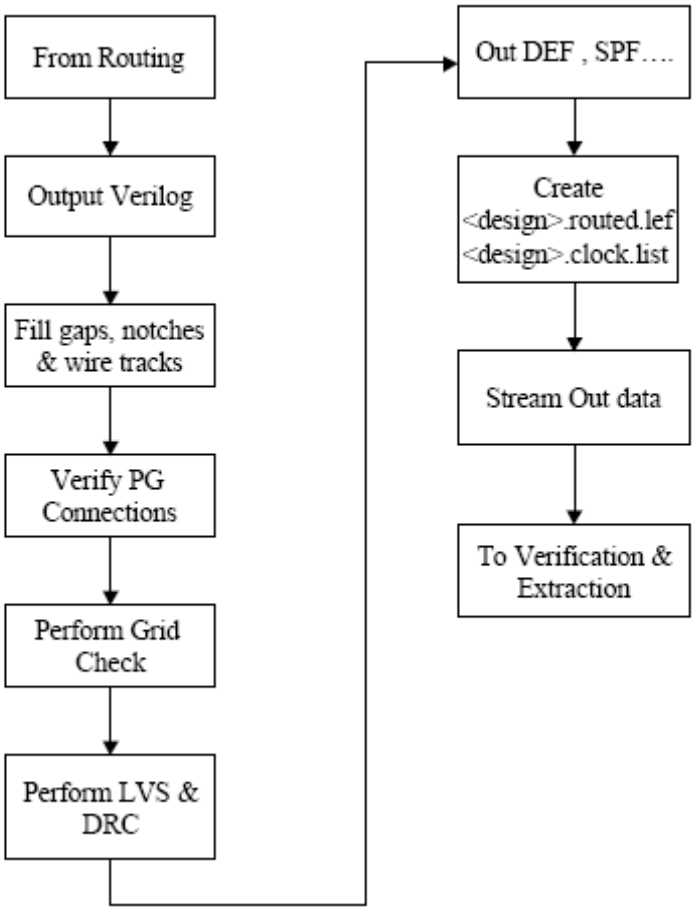


Fig. 2.4.1.6.1 Finish Design Flow

2.4.2 Benefits of the Astro:

Astro addresses ultradeep submicron (UDSM) effects during design processes. Its unique technology eliminates iterations, by concurrently accounting for all physical effects throughout the design process.

- Astro increases clock speeds and provides fast design completion.
- Astro improves productivity, by providing placement change controls such as incremental overlap removal and incremental placement. It also ensures the lowest possible clock skew with its early timing capability and ability to predict congestion impact.
- Design data created with the previous generation of Synopsys place and route tools is fully compatible with Astro.
- Astro follows the latest, advanced manufacturing process rules, thereby improving design reliability. It handles antenna repair, via optimization, metal fill, and metal slotting automatically. Effects such as conformal dielectrics, copper dishing, and shallow trench isolation are incorporated into the Astro parasitic extraction engine.
- Astro shortens design cycles. Its new high-performance algorithms and distributed routing capabilities enable you to get designs out the door quickly.

2.4.3 Magma PAR Flow:

The Magma flow includes a family of products that performs RTL synthesis (the Blast Create and Blast RTL tools) in addition to timing optimization and analysis, floorplanning, power planning, place and route, scan-based analysis and synthesis, and RC extraction technologies (the Blast Fusion tool). The system integrates these technologies with a common, open data model. There are no translations among the engines that implement these technologies, so intertool communication is fast and performance is excellent.

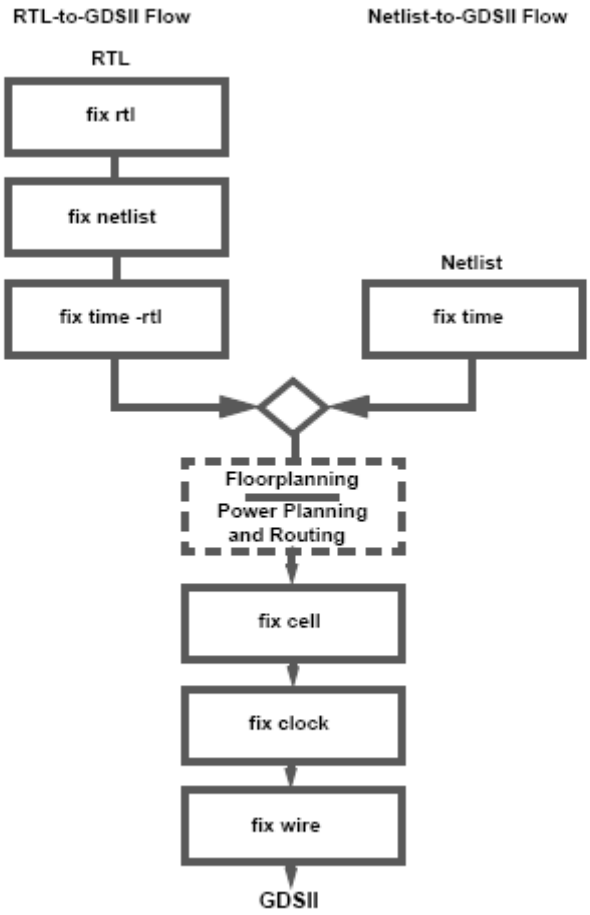


Fig. 2.4.2.1 Magma Design Flow

The Magma design flow includes a complete RTL-to-GDSII chip design and implementation system. Blast Create provides RTL-to-netlist implementation; Blast Fusion is the netlist-to-GDSII design and implementation system.

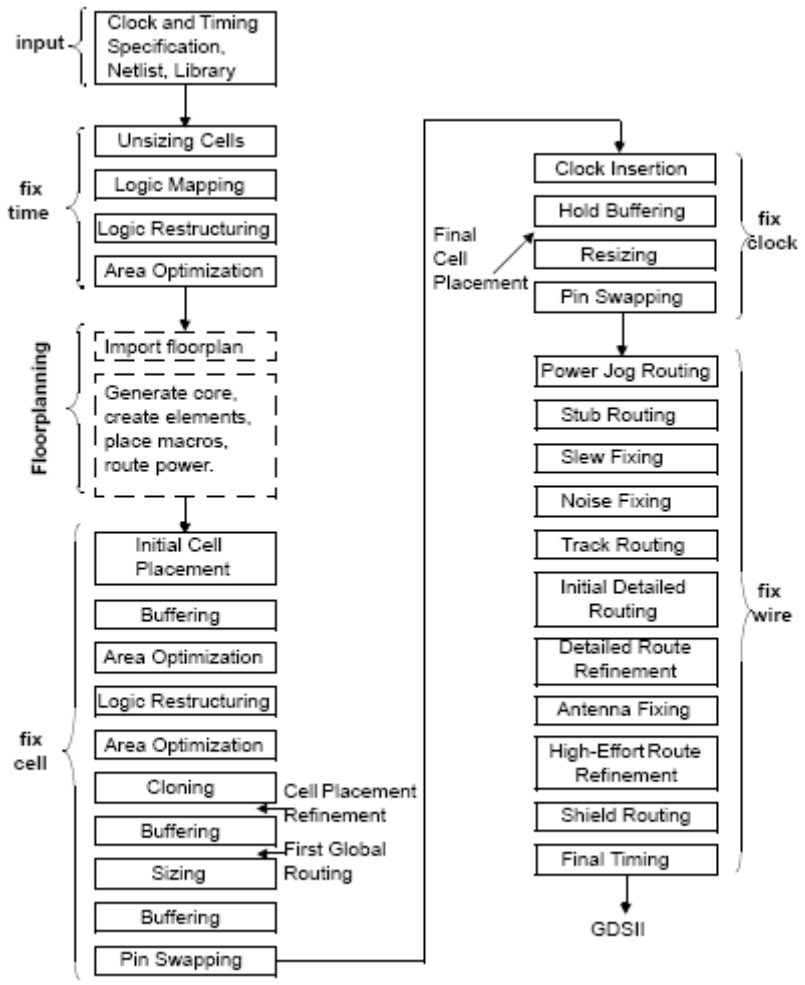


Fig. 2.4.2.2 Blast Fusion Design Flow

2.4.3.1 Optimization Features:

Blast Fusion includes the following optimization features:

- FixedTiming methodology, which allows Blast Fusion to determine the design's optimal timing prior to detailed physical implementation
- SuperCell abstraction, which replaces each logic function with automatically abstracted SuperCell models with variable sizes and fixed delay
- Gain trimming
- Logic restructuring
- Buffering to handle heavy loads
- Cloning and trimming
- Incremental static timing analysis

• Incremental parasitic extraction

2.4.3.2 Placement Features:

Blast Fusion includes the following placement features:

• High-capacity placement for handling large, complex, multimillion-gate designs

• Comprehensive congestion analysis based on placement, optimization, and routing Constraints.

• Full-chip, concurrent macro and cell placement

• Top-down and bottom-up methodologies for optimal I/O pin placement

• Scan chain reordering

• Interactive and user-defined I/O placement, including staggered I/Os

• Multi-height cell placement with power and ground rail sharing

2.4.3.3 Routing Features:

Blast Fusion includes the following routing features:

• Virtual gridless router, which blends the speed and capacity of a gridded router with the flexibility of a shape-based router

• Variable width and space routing

• Antenna avoidance

• Automatic metal slotting

• Stacked vias

• Custom via rule support

• Congestion-based capacitance estimation

• N-layer routing

• Top-level chip assembly routing and block-level standard cell routing

• Built-in polygon-based DRC engine

• Complete Engineering Change Order (ECO) support

• Power routing and support for various power mesh styles

• Automatic ring routing and wire tapping

• Interactive bus routing

• Scan chain routing

2.4.3.4 Clock Tree Features:

Blast Fusion includes the following advanced clock tree features:

• Balanced H-trees
• Bottom-up clustering algorithm
• Zero skew and useful skew
• User-controlled buffer specification
• Interclock skew and insertion delay minimization
• Multiple clock domains
• Automatic gated clock checks, wire sizing, shielding, wire spacing, and obstruction avoidance.

2.4.4 Benefits of the Magma Design Tool:

The Magma design tools provide the following benefits:

- Faster time to market

Eliminate the time-consuming iteration cycles that can delay the completion of your critical chip design project. You can create complex, high-quality deep submicron designs in less time.

- Better chip performance

Design chips with greater speed, smaller area, and lower power consumption. Powerful optimization technology coupled with the Magma embedded timing analyzer ensures the highest performance for your designs.

- Greater design capacity

Synthesize several million gates at a time—an order of magnitude larger than possible using traditional synthesis tools. The innovative Magma SuperCell technology and unified data model use memory efficiently, enabling increased capacity.

- Enhanced quality and reliability

The Magma single unified data model architecture allows automatic detection and active avoidance of potential signal integrity problems. The Blast Noise product provides automatic detection and avoidance of crosstalk noise and delay, and corrects potential signal electromigration (EM) problems. Additionally, both Blast Create and Blast Fusion provide automatic correction of potential antenna problems that can lead to lower chip yields.

Chapter – 3

Signoff Flow

CHAPTER 3

SIGNOFF FLOW

3.1 Introduction:

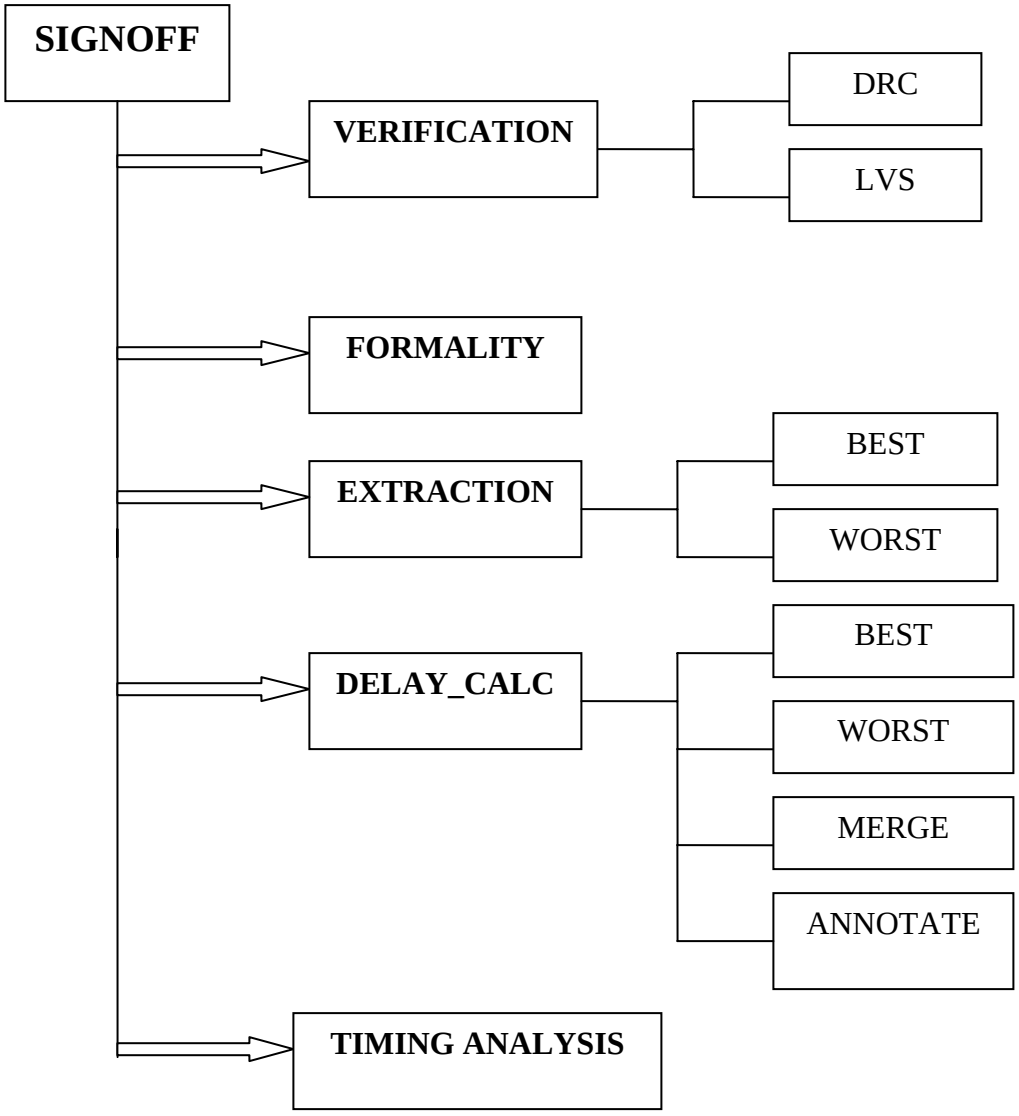


Fig.3.1 Signoff Flow

One purpose of a sign-off checklist is to ensure that certain checks were made during design, simulation and verification so that the final files meet certain criteria. Another objective of the checklist is to ensure that all necessary design, simulation and

verification files have been created and that installation script and required documentation have been developed.

The Sign-Off Flow is an environment to perform design Sign-Off verification. This flow consists of sequence of tasks to be performed by the user in a hierarchical manner. A script is provided to automate each task and generate reports. User has to carefully analyze each report.

Tasks that have to be run for design sign-off are:

- Netlist and Layout Verification Tasks
- Timing Verification Tasks

Among the tasks to be run are netlist and layout verification tasks:

FORMALITY: Formal proof between the verilog netlists before and after design implementation.

VERIFICATION: GDSII verification with calibre DRC/LVS and generation of the CDL netlist.

Timing verification tasks need to be run in following sequence:

EXTRACTION: Lef/Def extraction with StarRCXT.

DELAY_CALC: Verilog netlist annotated with parasitic into PrimeTime for delay calculation (max_capacitance, max_transition, max_fanout). SDF Worst, Best, merging and verilog SDF annotation are the used for simulation.

Table 3.1.1 The tools to be used for SignOff Flow are stated below:

Formal Verification	Synopsys Formality
Design rule check (DRC)	Mentor Graphics Calibre
Layout vs. Schematic	Mentor Graphics Calibre
Parasitic Extraction	Synopsys Star-RCXT
Post-layout delay calculation	Synopsys Prime Time
Post-layout Gate level timing analysis	Synopsys Prime Time
Back Annotation	ncVerilog , modelsim

Fig. 3.1.2 Netlist & Layout Verification Tasks:

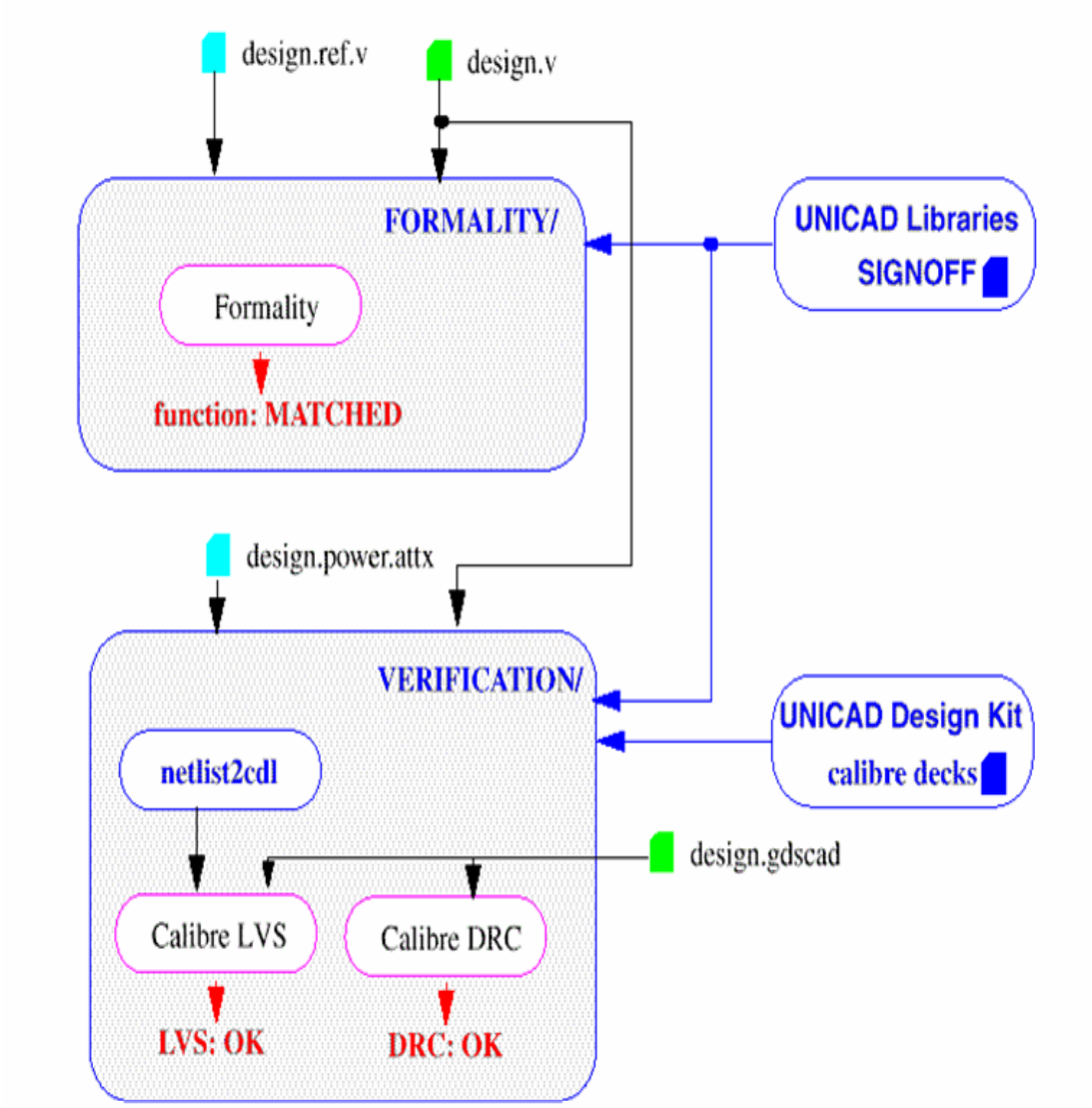
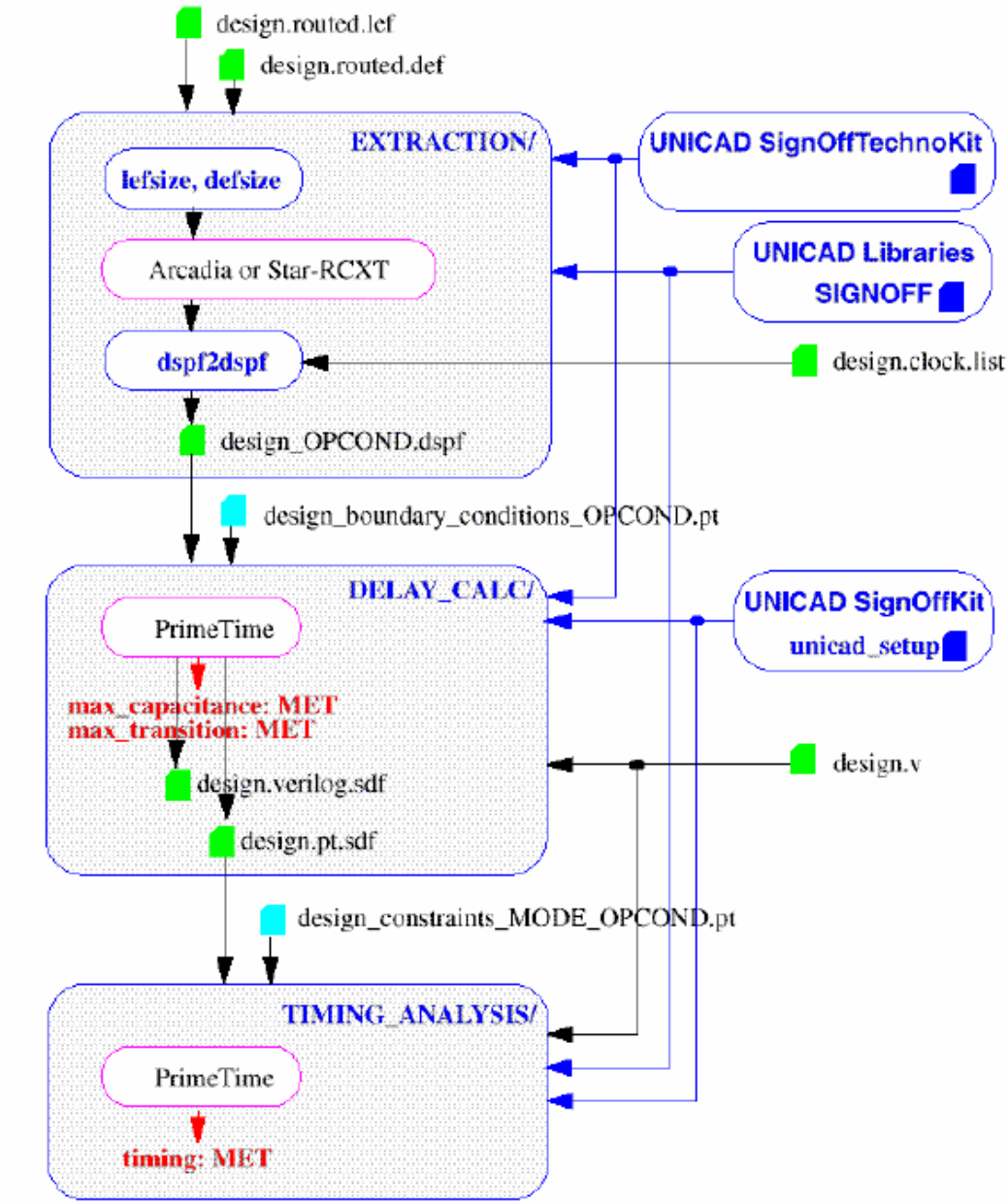


Fig. 3.1.3 Delay Calculation Tasks:



3.2 Formality:

Formality is an application that uses formal techniques to prove or disprove the functional equivalence of two designs or two cell libraries. Formality is used to compare a gate-level netlist to its register transfer level (RTL) source or to a modified version of that gate-level netlist. After the comparison, Formality reports whether the two designs or cell libraries are functionally equivalent. The Formality tool can significantly reduce your design cycle by providing an alternative to simulation for regression testing.

The techniques Formality uses are static and do not require simulation vectors. Consequently, for design verification you only need to provide a functionally correct, or golden design (called the reference design), and a modified version of the design (called the implementation). By comparing the implementation against the reference design, one can determine whether the implementation is functionally equivalent to the reference design. Cell library verification is similar except that each cell in the implementation library is compared against each cell in the reference library one cell at a time. As designs become larger and more complex and require more simulation vectors, regression testing with traditional simulation tools becomes a bottleneck in the design flow. Formality gives two distinct advantages: significantly reduced verification times and complete verification. Reduced verification times occur because Formality requires no input vectors. Reducing gate-level simulation time means one can spend more time verifying the initial golden RTL design to get better functional coverage.

The following list summarizes the Formality features:

- Proves two designs or cell libraries are functionally equivalent, faster than verification using event-driven simulators.
- Provides complete verification (not vector-dependent).
- Performs RTL-to-RTL, RTL-to-gate, and gate-to-gate design verifications.
- Performs Verilog-to-DB, Verilog-to-Verilog, DB-to-DB, Verilog-to-SPICE, and DB-to-SPICE cell library verifications.
- Offers diagnostic capabilities to help you locate and correct functional discrepancies between designs.
- Reads synthesizable VHDL, Verilog, Synopsys internal .db format, and EDIF netlists. “Performs automatic hierarchical verification.

- Uses existing Synopsys Design Compiler technology libraries.
- Saves and restores designs and verification sessions.
- Offers a graphical user interface (GUI) and a shell command-line interface (fm_shell).
- Verifies a wide range of design transforms or modifications, including pipeline retiming and reencoded finite state machines.
- Offers schematic views and isolated cone of logic views that support location of design discrepancies.

3.3 DRC:

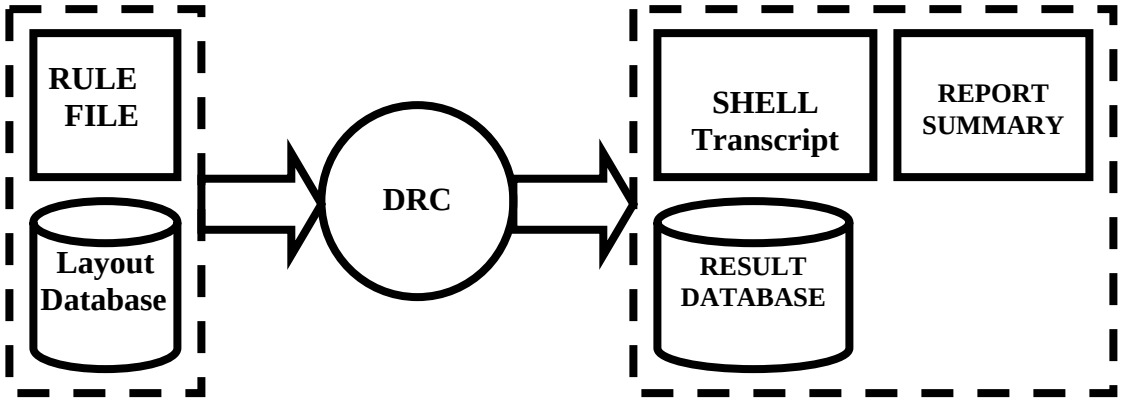


Fig. 3.3.1 DRC Data Flow

The Calibre DRC/DRC-H tools perform physical verification of integrated circuit layout designs in flat, hierarchical configurations. The inputs to Calibre DRC are an SVRF rule file and a layout database. The outputs are a run transcript and a DRC Results Database, with an optional DRC Summary Report.

3.4 LVS:

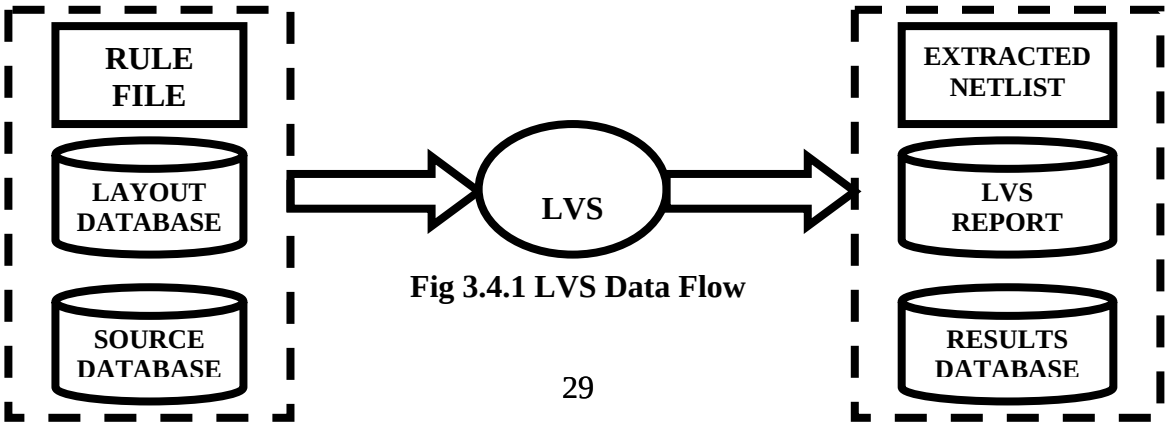


Fig 3.4.1 LVS Data Flow

LVS comparison occurs after device recognition in LVS applications. The LVS outputs that always occur include a run transcript and an LVS Report. Calibre LVS applications compare electrical circuits from the specified source netlist and layout geometry.

3.5 Extraction:

Star-RCXT is a software tool that extracts parasitics from connected databases that represent integrated circuit layout designs. One can use Star-RCXT to generate netlists to conduct a timing, clock, noise, or power analysis. one can also extract parasitics like resistors, capacitors, and inductors from a layout design database.

Star-RCXT performs these extraction tasks:

- Produces accurate, full-chip extraction for noise, signal electromigration, voltage (IR) drop, and timing verification.
- Does selective extraction and netlisting for critical path analysis.
- Provides a complete, production-proven solution for different design types, such as ASIC, full custom, microprocessor, memory, and analog designs.
- Offers consistent interconnect modeling for physical design and optimization. Efficient post-layout analysis enables fast timing convergence and rapid time to market.
- Includes advanced process effects such as copper interconnect, local interconnect, silicon on insulator (SOI), and in-die process variation.
- Ensures inductance extraction for multi-million gate SoC designs with automatic modeling of single or multiple current-return paths.
- Creates process characterization files for Star-RCXT which can also be obtained from major foundries.
- Integrates into existing design flows.
- Provides hierarchical and distributed processing.

3.6 Delay Calculation:

After breaking down a design into a set of timing paths, PrimeTime calculates the delay along each path. The total delay of a path is the sum of all cell and net delays in the path. The method of delay calculation depends on whether chip layout has been completed. Before layout, the chip topography is unknown, so PrimeTime must estimate the net delays using wire load models. After layout, an external tool can accurately determine the delays and write them to a Standard Delay Format (SDF) file.

With report_delay_calculation command one can generate a detailed report on how PrimeTime calculates the delay for a specified cell or net timing arc. This information is useful for debugging a timing problem or verifying the timing data in a technology library. The reported delay calculation takes into account the operating conditions and wire load models.

For initial static timing analysis, PrimeTime estimates net delays based on a wire load model. Actual delays depend on the physical placement and routing of the cells and nets in the design.

A floorplanner or router can provide more detailed and accurate delay information, which you can provide to PrimeTime for a more accurate analysis. This process is called delay back-annotation. Back-annotated information often is provided in a Standard Delay Format (SDF) file. Worst and Best SDF are merged into a single SDF. SDF annotation is performed on the verilog netlist using NC-VERILOG and ensure that post-layout back annotated simulations can be performed.

3.7 Timing Calculation:

Timing analysis is typically an iterative process. You begin with a gate-level design description and analyze the timing using wire load models to estimate the net delays. Then, after placement and routing, one can back-annotate the design description with detailed delay information provided by the layout tools, allowing PrimeTime to do a more accurate, layout-specific analysis. PrimeTime is a full-chip, gate-level static timing analysis tool targeted for complex, multimillion-gate designs. It offers an unsurpassed combination of

speed, capacity, ease of use, and compatibility with industry-standard data formats and workflows. To check a design for violations, PrimeTime breaks the design down into a set of timing paths, calculates the signal propagation delay along each path, and checks for violations of timing constraints inside the design and at the input/output interface.
PrimeTime supports a wide range of advanced timing analysis features, including the following:

- Multiple clocks and clock frequencies
- Multicycle path timing exceptions
- False path timing exceptions and automatic false path detection
- Transparent latch analysis and time borrowing
- Simultaneous minimum/maximum delay analysis for setup and hold constraints
- Analysis with on-chip variation of process, voltage, and temperature (PVT) conditions
- Case analysis (analysis with constants or specific transitions applied to specified inputs).
- Mode analysis (analysis with module-specific operating modes, such as read mode or write mode for a RAM module)
- Bottleneck analysis (reporting of cells that cause the most timing violations)
- What-if analysis without modifying the original netlist, using inserted buffers, resized cells, and modified nets.
- Analysis of crosstalk effects between physically adjacent nets using the PrimeTime SI (signal integrity) option.

3.8 Back-Annotation:

The sole point of entry of timing information into a VITAL compliant model is through the timing generics. With the exception of the use of VITAL_Timing and VITAL_Memory routines, all timing calculations are performed outside of the VHDL model, and external timing information is passed to the model through the back-annotation timing generics. Back-annotation is the process of updating the back-annotation timing generics with the external timing information. Signal delays which are used to model

negative timing constraints are computed in the negative constraint calculation stage of simulation; their calculation is not part of the back-annotation process.

The rules governing the back-annotation of timing values into a VITAL compliant model and the mapping of SDF constructs to back-annotation timing generics define the semantics assumed by models which adhere to the Level 0 specification.

Chapter – 4

Analysis of Results

CHAPTER 4

ANALYSIS OF RESULTS

4.1 Macro Libraries:

- **Inputs**:

This section explains input contents required for certification flow. Complete flow is explained by taking different reference libraries like

IO90LPHVT_REF_COMPENSATION_1V8_50A,

IO90LPHVT_REF_COMPENSATION_2V5_50A,

IO90LPHVT_REF_COMPENSATION_3V3_50A,

IO90LPHVT_REF_SSTL_COMPENSATION_1V8_50A,

CORE90LPHVT,

PR90M7.

All four compensation libraries contain single core macro cell. CORE90LPHVT is a Standard Cell library for 90 nanometer CMOS090LP VLSI digital design platform. This library contains 715 SSI combinational and sequential cells, designed to work at 1.0V (+10%/- 10%), 1.20V(+10mv /-10mv) and 1.40V(+10%/-10%) with temperature range from -40 up to 125 C. This library is used to provide logic. PR90M7 is a standard-cell library for CMOS090 VLSI digital designs, dedicated to be used for place & route. It contains 8 special cells (Antenna Protection, Clocktree, and Prober) designed to work at 1.0V (+10%/-10%) and 1.2V (+10%/-10%) with temperature range from -40 up to 125 C.

4.1.1 Design Entry and Netlist generation:

All libraries which need to be certified should be copied in workspace where one has to run complete certification flow. Then with help of fusion script and Ipsceen tool Verilog netlist is generated.

Fusion Script and Ipscreen tool generates following three files:

1. RTL Netlist – Netlist
2. Standard design constraint file - <design name>.sdc
3. Verilog Netlist file – <design name>.ref.v

Fig .4.1.1 RTL Netlist

```
Module CERT_COMP (clk,rst,dataout);
input clk,rst;
output dataout;
reg dataout;
wire [72:1] a;
wire [108:1] g;
COUNTER_2 #(72) inst_COUNTER_2 (.a(a),.clk(clk),.rst(rst),.dataout(dataout));
COUNTER_1 #(108) inst_COUNTER_1 (.count(g),.clk(clk),.rst(rst));
IOREF_COMPENSATION_1V8inst1_IOREF_COMPENSATION_1V8
 (.ACCURATE(g[1]),.COMPEN(g[2]),.COMPOK(a[1]),.COMPTQ(g[3])
,.FREEZE(g.NASRC4(a[6]),.NASRC5(a[7]),.NASRC6(a[8]),.RASRC0(g[6]),
.RASRC1(g[7]), …
IOREF_COMPENSATION_1V8inst2_IOREF_COMPENSATION_1V8
(.ACCURATE(g[14]),
.COMPEN(g[15]),.COMPOK(a[10]),.COMPTQ(g[16]),.FREEZE(g[17]),.GNDBGCOMP(g[
18]),.NASRC0(a[11]),.NASRC1(a[12]),.NASRC2(a[13]),.NASRC3(a[14]),.NASRC4(a[15]),
.NASRC5(a[16]),.NASRC6(a[17]),. …

IOREF_COMPENSATION_2V5inst1_IOREF_COMPENSATION_2V5
(.ACCURATE(g[27]),.COMPEN(g[28]),.COMPOK(a[19]),.COMPTQ(g[29]),.FREEZE(g[3
0]),.GNDBGCOMP(g[31]),.NASRC0(a[20]),.NASRC1(a[21]),…..));

IOREF_COMPENSATION_2V5inst2_IOREF_COMPENSATION_2V5
(.ACCURATE(g[40]),.COMPEN(g[41]),.COMPOK(a[28]),.COMPTQ(g[42]),.FREEZE(g[4
3]),.GNDBGCOMP(g[44]),.NASRC0(a[29]),.NASRC1(a[30]),. …));
IOREF_COMPENSATION_3V3inst1_IOREF_COMPENSATION_3V3 (.A0SRC(a[37]),
.A1SRC(a[38]),.A2SRC(a[39]),.A3SRC(a[40]),.A4SRC(a[41]),.A5SRC(a[42]),.A6SRC(a[43
]),.ACCURATE(g[53]),.CHIPSLEEP(g[54]),…..));
```

```
IOREF_COMPENSATION_3V3inst2_IOREF_COMPENSATION_3V3 (.A0SRC(a[46]),
.A1SRC(a[47]),.A2SRC(a[48]),.A3SRC(a[49]),.A4SRC(a[50]),.A5SRC(a[51]),.A6SRC(a[52
]),.ACCURATE(g[68]),.CHIPSLEEP(g[69]),….));
IOREF_SSTL_COMPENSATION_1V8
inst1_IOREF_SSTL_COMPENSATION_1V8 (.ACCURATE(g[83]),.COMPEN(g[84]),
.COMPOK(a[55]),.COMPTQ(g[85]),.FREEZE(g[86]),
.GNDBGCOMP(g[87]),.NASRC0(a[56]),.NASRC1(a[57]),.NASRC2(a[58]),.NASRC3(a[59
]),.NASRC4(a[60]),.NASRC5(a[61]),.NASRC6(a[62]),.RASRC0(g[88]),.RASRC1(g[89]),
.RASRC2(g[90]),.RASRC3(g[91]),.RASRC4(g[92]),.RASRC5(g[93]),.RASRC6(g[94]),
.REXT(a[63]), .TQ(g[95]));
IOREF_SSTL_COMPENSATION_1V8 inst2_IOREF_SSTL_COMPENSATION_1V8
(.ACCURATE(g[96]),.COMPEN(g[97]),.COMPOK(a[64]),.COMPTQ(g[98]),.FREEZE(g[9
9]),.GNDBGCOMP(g[100]),.NASRC0(a[65]),.NASRC1(a[66]),…..));
endmodule
 module COUNTER_1 (count, clk,rst);
 parameter width=8;
 input clk,rst;
 output [width-1:0] count;
 reg [width-1:0] temp;  reg [width-1:0] count;
 always @ (posedge clk )
 begin
   if (rst==0)
   begin
   count=0;   temp=0;
   end
   else
   begin
     temp=temp+1 ;  count=temp;
   end
 end
  endmodule
```

Fig 4.1.2 Verilog Netlist :

```
Module COUNTER_2 (a, clk,rst,dataout);
 parameter width=9;
 input [width-1:0] a;
 input clk,rst;
 output dataout;  reg dataout;
  always @ (posedge clk )
 begin
   if (rst==1)
   dataout<=0;
   else
   dataout<= | (a);
 end
 endmodule
module CERT_COMP ( clk, rst, dataout );
  input clk, rst;
  output dataout;
  wire   n4, n1, n2;
  wire   [72:1] a;
  wire   [108:1] g;
  COUNTER_2_width72 inst_COUNTER_2 ( .a(a), .clk(clk), .rst(n1), .dataout );
  COUNTER_1_width108 inst_COUNTER_1 ( .count(g), .clk(clk), .rst(n1) );
  IOREF_COMPENSATION_1V8inst1_IOREF_COMPENSATION_1V8
(.ACCURATE(g[1]), .COMPEN(g[2]), .COMPOK(a[1]), .COMPTQ(g[3]), .FREEZE(g[4]),
.GNDBGCOMP(g[5]),.NASRC0(a[2]),.NASRC1(a[3]),.NASRC2(a[4]),
.NASRC3(a[5]),.NASRC4(a[6]),.NASRC5(a[7]),.NASRC6(a[8]),……
IOREF_COMPENSATION_1V8inst2_IOREF_COMPENSATION_1V8
(.ACCURATE(g[14]),.COMPEN(g[15]),.COMPOK(a[10]),.COMPTQ(g[16]),
.FREEZE(g[17]),.GNDBGCOMP(g[18]), .NASRC0(a[11]), .NASRC1(a[12]), …….) );
  IOREF_COMPENSATION_2V5inst1_IOREF_COMPENSATION_2V5
(.ACCURATE(g[27]),.COMPEN(g[28]),.COMPOK(a[19]),.COMPTQ(g[29]),
```

```
.FREEZE(g[30]),.GNDBGCOMP(g[31]),.NASRC0(a[20]),.NASRC1(a[21]),
.NASRC2(a[22]),  .NASRC3(a[23]), .NASRC4(a[24]), .NASRC5(a[25]), .NASRC6(a[26]),
.RASRC0(g[32]), .RASRC1(g[33]), .RASRC2(g[34]), ……) );
 IOREF_COMPENSATION_2V5inst2_IOREF_COMPENSATION_2V5
(.ACCURATE(g[40]),.COMPEN(g[41]),.COMPOK(a[28]),.COMPTQ(g[42]),
.FREEZE(g[43]),.GNDBGCOMP(g[44]),.NASRC0(a[29]),.NASRC1(a[30]),
.NASRC2(a[31]),
      .NASRC3(a[32]), .NASRC4(a[33]), .NASRC5(a[34]), .NASRC6(a[35]),….) );
IOREF_COMPENSATION_3V3inst1_IOREF_COMPENSATION_3V3
(A0SRC(a[37]),.A1SRC(a[38]),.A2SRC(a[39]),.A3SRC(a[40]),.A4SRC(a[41]),
.A5SRC(a[42]),.A6SRC(a[43]),.ACCURATE(g[53]),.CHIPSLEEP(g[54]),
.COMPEN(g[55]),.COMPOK(a[44]),.COMPTQ(g[56]),.FREEZE(g[57]),
.GNDBGCOMP(g[58]),.RA0SRC(g[59]),.RA1SRC(g[60]),.RA2SRC(g[61]),
.RA3SRC(g[62]), ….    ) );
 IOREF_COMPENSATION_3V3inst2_IOREF_COMPENSATION_3V3
(.A0SRC(a[46]),.A1SRC(a[47]),.A2SRC(a[48]),.A3SRC(a[49]),.A4SRC(a[50]),
.A5SRC(a[51]),.A6SRC(a[52]),.ACCURATE(g[68]),.CHIPSLEEP(g[69]),
.COMPEN(g[70]), .COMPOK(a[53]), .COMPTQ(g[71]), .FREEZE(g[72]), ….          ) );
 IOREF_SSTL_COMPENSATION_1V8 inst1_IOREF_SSTL_COMPENSATION_1V8
 (.ACCURATE(g[83]),  .COMPEN(g[84]),  .COMPOK(a[55]),  .COMPTQ(g[85]),
.FREEZE(g[86]),.GNDBGCOMP(g[87]),.NASRC0(a[56]),.NASRC1(a[57]),
.NASRC2(a[58]),… ) );
 IOREF_SSTL_COMPENSATION_1V8  inst2_IOREF_SSTL_COMPENSATION_1V8  (
.ACCURATE(g[96]),.COMPEN(g[97]),.COMPOK(a[64]),.COMPTQ(g[98]),
.FREEZE(g[99]),.GNDBGCOMP(g[100]),.NASRC0(a[65]),.NASRC1(a[66]),
.NASRC2(a[67]),.NASRC3(a[68]),.NASRC4(a[69]),.NASRC5(a[70]),.NASRC6(a[71]),…));
  BFHVTX12 U1 (..A(rst), .Z(n1) );
 IVHVTX0H U2 (.A(n4), .Z(n2) );
 IVHVTX12 U3 (.A(n2), .Z(dataout) );
Endmodule
```

Fig 4.1.3 Standard Design Constraints

```
#####################################################
#  Created by Design Compiler write_sdc on Mon Sep 25 11:32:41 2006
#####################################################
set sdc_version 1.4
create_clock -period 10 -waveform {0 5} [get_ports {clk}]
set_input_delay 5 -max -clock "clk" [get_ports {rst}]
set_input_delay 1 -min -clock "clk" [get_ports {rst}]
set_output_delay 5 -max -clock "clk" [get_ports {dataout}]
set_output_delay 0.2 -min -clock "clk" [get_ports {dataout}]
set_clock_uncertainty  0.1 [get_clocks {clk}]
set_input_transition -max 5 [get_ports {rst}]
set_input_transition -min 0.2 [get_ports {rst}]
set_load -pin_load 5 [get_ports {dataout}]
set_load -min -pin_load 5 [get_ports {dataout}]
```

4.1.2 PAR Flow:

Then Astro PAR flow is used for placement and routing.

- **Input Files Expected**

  Verilog netlist of the design : ***<design>.ref.v***

  Synopsys Design Constraint file for timing constraints : ***<design>.sdc***

  Clock Specification in attx format: : ***<design>.clock.attx***

  Muti-supply format in attx format: : ***<design>.power.attx***

design.ref.v and design.sdc files are generated by Ipscreen tool. design.clock.attx file is generated by CTEDIT task. Synopsys kit is required for that purpose. To support multisupply flow user has to create <design>.power.attx file for specifying particular gnd/power connection. If no <Design Name>.powerAttx file is given, default connections will be done according to default.power.attx file which is available in COLLECT_DATA.

4.1.2.1 Flow Setup:

In order to run the flow, the user must create/use a seed.tcl file, where a number of settings for the design is specified.

- **Workspace Setup :**

A utility called avtGenerateTasks reads in the seed.tcl file in the SETUP directory and creates directories described below the argument to pass to this utility are the following

[-seed] to specify seed.tcl -default = SETUP/seed.tcl

[- task] to specify the task environment - default = ASTRO

[- work] to specify directory name - default = PLACEANDROUTE for ASTRO task

This generates the design workspace with the following directories:

.AVANTI/

-<WORKSPACE>/

-VERILOGMODELS

-IMPORT

-EXPORT

Besides existing **LIBRARIES** and **SETUP** directories.

-LIBRARIES: This directory contains links to the master libraries directory.

-SETUP: Contain source *seed.tcl* file.

-AVANTI*: this directory contains the Astro workspace(s) directory, the IMPORT and EXPORT directories.*

•**<WORKSPACE>**: This is the directory where Astro will be run. Its content is detailed in the following chapter

•**IMPORT**: directory where the user will store input data for the Place&Route flow

•**EXPORT**: directory where the Place&Route flow will store data for HANDOFF and SIGNOFF task

Pdef and verilog File for Physical Compiler (post Floorplan or CTS)

GDS file for DRC/LVS in cad layers

DEF file to interface with cadence and/or extraction tools

Lef file with the definition of Vias for SignOff Extraction

Hierarchical Verilog netlist for Timing Analysis

- **IMPORT DESIGN :**

Before the step of NetlistIn, constraints files (<design>_constraints_<mode>_Max.pt, <design>_constraints_<mode>_Min.pt,<design>_boundary_conditions_Max.pt,<design>_ boundary_conditions_Min.pt) should be linked to the .sdc file in (**AVANTI/IMPORT/).** The design to be certified is imported first .This is done by clicking on STM icon and the option Import design.

Timing constraints to different signals used in front-end design (Verilog or VHDL) is need to be propagated to the back-end .This is given as SDC format to Synopsys Design Compiler.

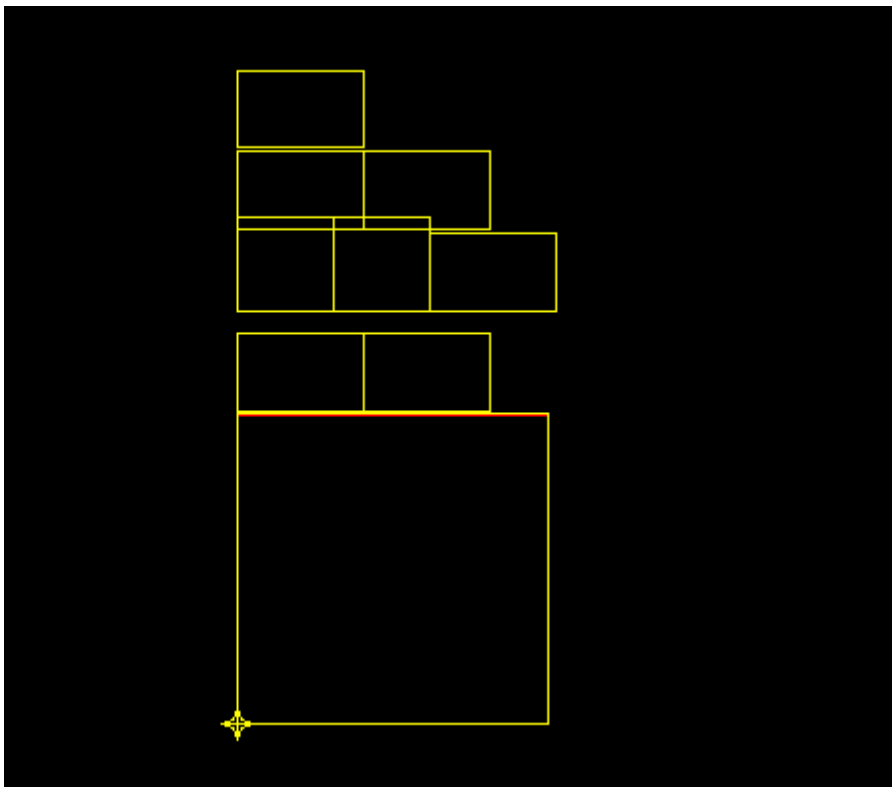


Fig. 4.1.2.1.1 Layout of Macro after Import Design

- **FLOORPLANNING :**

The next step is to do Floorplanning .Before running the **floorplan**, update the width and height of the floorplan such that sufficient area is left for subsequent routing of the cells. Macro should be placed in minimum two orientations R0 and R90 to test the power-Planning & routing related issues. After floorplanning check the grid view for connections of power-pins.Check digital power and ground are properly tapped .

In case, power ground pins are present within the macro(not abutted with the boundary) then check that the pins are present in atleast two orientations for more flexibility. **By default, the PnR flow creates a single power and ground supply. For macros** which have multiple power supplies, create the power straps after the placement-flow is done. Also the power.attx file should be prepared accordingly.

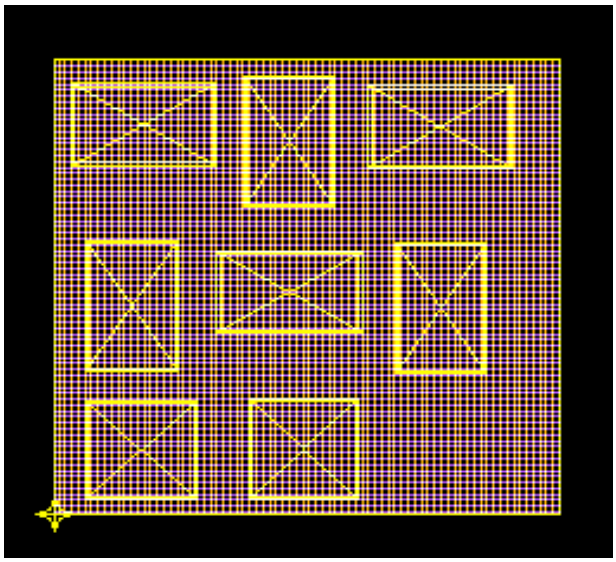


Fig. 4.1.2.1.2 Layout of Macros after Floorplan

- **PLACE DESIGN**

Performs placement in three steps:

- Pre-placement Optimization.
- Placement.
- Post-placement Optimization.

- **TREE SYNTHESIS**

The next task would be the generation of clock trees. The main objective of Clock Tree Generation is to avoid the clock skew which would create synchronization failure between different entities in the design. This is done in CTEDIT directory in Avanti.

- **ROUTE DESIGN**

The next task would be to route the entities in the design.

•pre-route the padring, based on the PGConnect commands

•Add Diodes protection on pins for a block

•Route the clock tree nets

•route the design

- **POSTROUTE DESIGN**

Main objectives of postroute design given below:

•post-route Timing Optimization

•Crosstalk correction

•Antenna repair

•Add Filler Cell

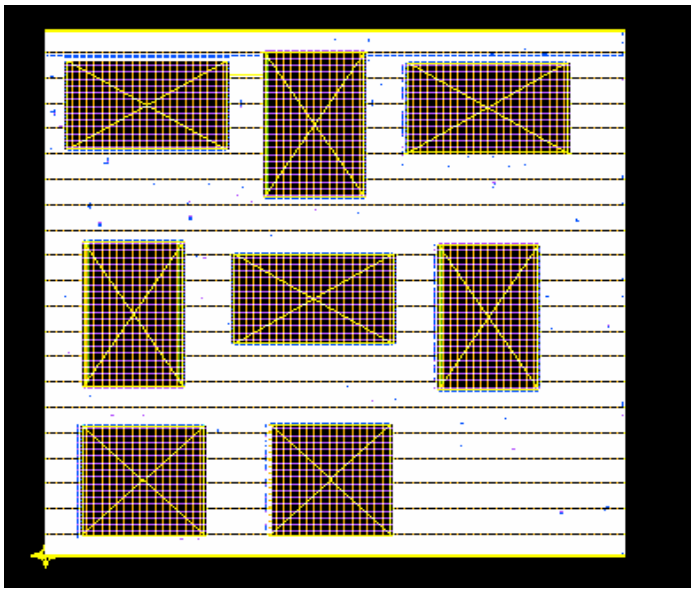


Fig. 4.1.2.1.3 Layout of Macros Post Routing

- **FINISH DESIGN :**

Finish Design option generate three files:

Verilog Netlist

GDSII

DEF

Fig 4.1.2.1.4 Gate level netlist (CERT_COMP.v)

```
Module CERT_COMP (dataout , rst , clk );
output dataout ;
input  rst ;
input  clk ;
wire [108:1] g ;
wire [72:1] a ;
COUNTER_1_width108inst_COUNTER_1 (.count(g),.clk(clk),.rst( rstASTttcNet304
) ) ;
COUNTER_2_width72 inst_COUNTER_2 (.clk ( clk ) , .rst ( rstASTttcNet304 ) ,
 .dataout ( n4 ) , .a ( a ) ) ;
IOREF_COMPENSATION_2V5  inst1_IOREF_COMPENSATION_2V5  (.RASRC6
(g[38] ) , .NASRC5 ( a[25] ), .COMPEN ( g[28] ) , .TQ ( g[39] ) ,.COMPTQ ( g[29]) ,
.FREEZE ( g[30] ) , .REXT ( a[27] ) , .ACCURATE ( g[27] ) ,  …. ));
IOREF_COMPENSATION_1V8  inst2_IOREF_COMPENSATION_1V8  (.NASRC0
( a[11] ), .NASRC1 ( a[12] ) , .NASRC2 ( a[13] ) , .NASRC3 ( a[14] ) , .NASRC4 (
a[15] ) , .NASRC5 ( a[16] ) , .NASRC6 ( a[17] ) , .RASRC0 ( g[19] ) ,… ) ) ;
IOREF_COMPENSATION_1V8  inst1_IOREF_COMPENSATION_1V8  (.NASRC0
( a[2] ), .NASRC1 ( a[3] ) , .NASRC2 ( a[4] ) , .NASRC3 ( a[5] ) , .NASRC4 ( a[6] ,
  .NASRC5 ( a[7] ) , .NASRC6 ( a[8] ) , .RASRC0 ( g[6] ) , .RASRC1 ( g[7]),  …. ) ) ;
BFHVTX4 count_regx81xASTttcInst6 (.A ( g[82] ) , .Z ( g_82_ASTttcNet6 ) ) ;
BFHVTX4 count_regx66xASTttcInst7 (.A ( g[67] ) , .Z ( g_67_ASTttcNet7 ) ) ;
BFHVTX12 U3ASTttcInst8 (.A ( dataoutASTttcNet8 ) , .Z ( dataout ) ) ;
BFHVTX6 U2ASTipoInst889 (.A ( n2 ) , .Z ( n2ASTipoNet234 ) ) ;
BFHVTX12 toplevelASTttcInst1162 (.A( rstASTttcNet474 ),.Z ( rstASTttcNet304 ) )
```

```
IVHVTX12 U3 (.A ( n2ASTipoNet234 ) , .Z ( dataoutASTttcNet8 ) ) ;
IVHVTX0H U2 (.A ( n4 ) , .Z ( n2 ) ) ;
BFHVTX12 toplevelASTttcInst1718 (.A ( rst ) , .Z ( rstASTttcNet474 ) ) ;
IOREF_SSTL_COMPENSATION_1V8
inst2_IOREF_SSTL_COMPENSATION_1V8 (.RASRC3( g[104] ),.NASRC1(a[66]),
.RASRC1 ( g[102] ) , .RASRC4 ( g[105]),.RASRC0 (g[101] ),.COMPOK(a[64] ),
.RASRC2(g[103]),.COMPTQ(g[98]),.RASRC5(g[106]),…. ) ) ;
IOREF_SSTL_COMPENSATION_1V8
inst1_IOREF_SSTL_COMPENSATION_1V8 (
.RASRC3 ( g[91] ) , .NASRC1 ( a[57] ) , .RASRC1 ( g[89] ) , .RASRC4 ( g[92] ) ,
.RASRC0 ( g[88] ) , .COMPOK ( a[55] ) , .RASRC2(g[90]) ,.COMPTQ(g[85]),…) ) ;
IOREF_COMPENSATION_3V3 inst2_IOREF_COMPENSATION_3V3 (.A4SRC (
a[50] ) , .A3SRC ( a[49] ) ,.RA3SRC ( g[77] ), .RA4SRC ( g[78] ) , .A6SRC ( a[52]),
.A5SRC ( a[51] ) , .COMPOK ( a[53] ) , .A2SRC ( a[48] ) , .A1SRC ( a[47] ),
.A0SRC ( a[46] ) , .RA0SRC ( g[74] ) , .RA1SRC ( g[75] ) , .RA2SRC (g[76])… ) ) ;
IOREF_COMPENSATION_3V3 inst1_IOREF_COMPENSATION_3V3 (.A4SRC (
a[41]) , .A3SRC ( a[40] ) , .RA3SRC ( g[62] ) , .RA4SRC ( g[63] ) ,.A6SRC ( a[43] ),
 .A5SRC ( a[42] ) , .COMPOK ( a[44] ) , .A2SRC ( a[39] ) , .A1SRC ( a[38] ),
 .A0SRC ( a[37] ) , .RA0SRC ( g[59] ) , .RA1SRC ( g[60]),.RA2SRC(g[61]),… ) ) ;
IOREF_COMPENSATION_2V5 inst2_IOREF_COMPENSATION_2V5 (.RASRC6
( g[51] ),.NASRC5 ( a[34] ) , .COMPEN ( g[41] ) , .TQ ( g[52] ) , .COMPTQ ( g[42] )
.FREEZE (g[43]),.REXT (a[36]),.ACCURATE (g[40]) , .COMPOK ( a[28]),… ) ) ;
Endmodule
```

Fig. 4.1.2.1.5 CERT_COMP.Routed.def

```
VERSION 5.5 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;
DESIGN CERT_COMP ;
TECHNOLOGY cmos090_7M2T ;
```

```
UNITS DISTANCE MICRONS 1000 ;
DIEAREA ( 0 0 ) ( 1199800 1200640 ) ;
ROW STD_ROW_0 unit 0 1160880 N DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_1 unit 0 1164800 FS DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_2 unit 0 1168720 N DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_3 unit 0 1172640 FS DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_4 unit 0 1176560 N DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_5 unit 0 1180480 FS DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_6 unit 0 1184400 N DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_7 unit 0 1188320 FS DO 4285 BY 1 STEP 280 0 ;
ROW STD_ROW_8 unit 0 1192240 N DO 4285 BY 1 STEP 280 0 ;
……..
TRACKS Y 140 DO 4288 STEP 280 LAYER M1 ;
TRACKS Y 140 DO 4288 STEP 280 LAYER M2 ;
TRACKS X 140 DO 4285 STEP 280 LAYER M2 ;
TRACKS X 140 DO 4285 STEP 280 LAYER M1 ;
TRACKS X 140 DO 4285 STEP 280 LAYER M3 ;
TRACKS Y 140 DO 4288 STEP 280 LAYER M3 ;
TRACKS Y 140 DO 4288 STEP 280 LAYER M2 ;
TRACKS Y 140 DO 4288 STEP 280 LAYER M4 ;
…….
GCELLGRID X -3920 DO 307 STEP 3920 ;
GCELLGRID Y -3360 DO 307 STEP 3920 ;
VIAS 70 ;
 - AVNT_via1_HVFat_550_230_ALL_2_1
   + PATTERNNAME AVNT_via1_HVFat-550-230-3
   + RECT M1 ( -275 -70 ) ( 275 70 )
   + RECT M2 ( -230 -115 ) ( 230 115 )
   + RECT VIA1 ( -225 -65 ) ( -95 65 )
   + RECT VIA1 ( 95 -65 ) ( 225 65 ) ;
 - AVNT_via1_HVFat_230_550_ALL_1_2
```

```
   + PATTERNNAME AVNT_via1_HVFat-230-550-I1
   + RECT M1 ( -115 -230 ) ( 115 230 )
   + RECT M2 ( -70 -275 ) ( 70 275 )
   + RECT VIA1 ( -65 -225 ) ( 65 -95 )
   + RECT VIA1 ( -65 95 ) ( 65 225 )
- CONT1_380_380_ALL_2_2
   + PATTERNNAME CONT1-380-380-I3
   + RECT PO ( -190 -190 ) ( 190 190 )
   + RECT M1 ( -190 -190 ) ( 190 190 )
   + RECT CONT ( -190 -190 ) ( -70 -70 )
   + RECT CONT ( 70 -190 ) ( 190 -70 )
   + RECT CONT ( -190 70 ) ( -70 190 )
   + RECT CONT ( 70 70 ) ( 190 190 )
- AVNT_via5Fat_6040_2360_ALL_7_3
   + PATTERNNAME AVNT_via5Fat-6040-2360-J7F
   + RECT M5 ( -3020 -1130 ) ( 3020 1130 )
   + RECT M6 ( -2970 -1180 ) ( 2970 1180 )
   + RECT VIA5 ( -2940 -1100 ) ( -2580 -740 )
   + RECT VIA5 ( -2020 -1100 ) ( -1660 -740 )
   + RECT VIA5 ( -1100 -1100 ) ( -740 -740 )
   + RECT VIA5 ( -180 -1100 ) ( 180 -740 )
   + RECT VIA5 ( 740 -1100 ) ( 1100 -740 )
- AVNT_via5Fat_2360_1440_ALL_3_2
   + PATTERNNAME AVNT_via5Fat-2360-1440-I7
   + RECT M5 ( -1180 -670 ) ( 1180 670 )
   + RECT M6 ( -1130 -720 ) ( 1130 720 )
   + RECT VIA5 ( -1100 -640 ) ( -740 -280 )
   + RECT VIA5 ( -180 -640 ) ( 180 -280 )
   + RECT VIA5 ( 740 -640 ) ( 1100 -280 )
- AVNT_via5Fat_6040_520_ALL_7_1
   + PATTERNNAME AVNT_via5Fat-6040-520-7F
```

```
   + RECT M5 ( -3020 -210 ) ( 3020 210 )
   + RECT M6 ( -2970 -260 ) ( 2970 260 )
   + RECT VIA5 ( -2940 -180 ) ( -2580 180 )
   + RECT VIA5 ( -2020 -180 ) ( -1660 180 )
- AVNT_via5Fat_6040_5120_ALL_7_6
   + PATTERNNAME AVNT_via5Fat-6040-5120-M7F
   + RECT M5 ( -3020 -2510 ) ( 3020 2510 )
   + RECT M6 ( -2970 -2560 ) ( 2970 2560 )
   + RECT VIA5 ( -2940 -2480 ) ( -2580 -2120 )
   + RECT VIA5 ( -2020 -2480 ) ( -1660 -2120 )
   + RECT VIA5 ( -1100 -2480 ) ( -740 -2120 )
- AVNT_via4Fat_870_2790_ALL_3_9
   + PATTERNNAME AVNT_via4Fat-870-2790-P7
   + RECT M4 ( -390 -1395 ) ( 390 1395 )
   + RECT M5 ( -435 -1350 ) ( 435 1350 )
   + RECT VIA4 ( -385 -1345 ) ( -255 -1215 )
   + RECT VIA4 ( -65 -1345 ) ( 65 -1215 )
   + RECT VIA4 ( 255 -1345 ) ( 385 -1215 )
```

4.1.3 Signoff Flow :

SoGenerateTasks utility is used to setup the environment by generating scripts and setup files for the tools used in the flow.

The command to be applied for the environment set up is as follows:

cd <workspace>/SETUP

SoGenerateTasks –seed seed.tcl

4.1.3.1 Formality:

The Formality tool is used to formally prove that the back-end netlist is consistent with the front-end netlist.

Input Files:

<design>.ref.v

<design>.v

The commands to run formality tool is available in run.csh file:

- SIGNOFF/FORMALITY/RUN/run.csh

Following file is used to view the verification result.

- vi FORMALITY.run.log

4.1.3.2 Verification:

In this task, Calibre tool is used to perform physical checks (DRC+LVS) on the layout in GDSII format (<design>.gdscad). This .gdscad file is generated by Apollo tool. The CDL netlist is generated before running Calibre LVS.

Input Files Required:

SIGNOFF/IMPORT/

· <design>.v

· <design>.power.attx

· <design>.gdscad

Output Files:

SIGNOFF/EXPORT/<design>.cdl

CDL

Translating the physical Verilog netlist with netlist2cdl generates the CDL netlist. The power attribute file (design.power.attx) is mandatory only if some of the netlist require different power connections, otherwise the file can be left empty to perform the flow.

Commands to generate the CDL netlist are:

Go to run directory

- cd SIGNOFF/VERIFICATION/RUN

Source script for generating CDL netlist from run.csh

- source VERIFICATION.CDL.run.csh

Reviews following log file in RUN directory and ensure no error or warning are present

- vi VERIFICATION.cdl.run.log

DRCCAD

Calibre is used to perform physical checks on the layout(DRC).

Sequence of commands to perform DRC:

Go to run directory

- cd SIGNOFF/VERIFICATION/RUN

Source script for running DRC from run.csh

- source VERIFICATION.DRCCAD.run.csh

Reviews log file (in RUN directory) and drc_summary and ensure that 'DRC results generated' is 0.

- vi VERIFICATION.drccad.run.log
- vi ../DRCCAD/drc_summary

In case of any issues, sourcing Calibre GUI and loading drc_results.db can do debugging. Following command should be run for loading results in Avanti(Apollo Main Window).

- st_loadCalibreDB

LVSCAD

Calibre is used to verify that layout generated is equivalent to the physical CDL netlist.

Sequence of commands to perform LVS:

Go to run directory

- cd SIGNOFF/VERIFICATION/RUN

Source script for running LVS from run.csh

- source VERIFICATION.LVSCAD.run.csh

Reviews log file (in RUN directory) and lvs_summary and ensure that 'LVS results generated' is 0.

- vi VERIFICATION.lvscad.run.log
- vi ../LVSCAD/compare.rep

4.1.3.3 Extraction:

The StarRCXT tool is used to extract parasitic resistances and capacitors from the placed and routed data base (LEF/DEF). Within the LEF/DEF flows, sizing on macro shapes (contained on LEF) and on interconnect wiring (defined on DEF) is applied.

There are two modes of extraction: Best and Worst

Input Files Required:

SIGNOFF/IMPORT/

- <design>.clock.list : Contains the list of clock nets
- <design>.routed.def : is the def out from the place and routed flow
- <design>.routed.lef : contains the definition of cells used in the <design>.routed.def and not present in the library or technology kit.

Output Files:

- **SIGNOFF/EXPORT/**
  - <design>_Best.dspf
  - <design>_Worst.dspf
  - <design>_Best.setload.dctcl
  - <design>_Worst.setload.dctcl

Go to run directory

- cd SIGNOFF/EXTRACTION/RUN

Source script for running EXTRACTION from run.csh

- source EXTRACTION_STARRCXT.BEST.run.csh
- source EXTRACTION_STARRCXT.WORST.run.csh

Reviews log file (in RUN directory) and ensure that no error is present.

- vi EXTRACTION_STARRCXT.BEST.run.log
- vi EXTRACTION_STARRCXT.WORST.run.log

4.1.3.4 Delay Calculation:

Prime Time is used to perform delay calculation in order to generate two SDF (Standard Delay Format) files: one for Static Timing Analysis (STA) and one for Simulation. Worst and Best SDF are merged into a single SDF. SDF annotation is performed on the Verilog netlist using NC-VERILOG and ensure the post-layout back annotated simulations can be performed. Boundary constraints are required for delay calculations.

Go to run directory

- cd SIGNOFF/DELAY_CALCULATION/RUN

Source script for running DELAY_CALCULATION from run.csh

Reviews log file (in RUN directory) and ensure that no error is present.

- vi DELAY_CALCULATION.run.log

4.2 Memory:

This section describes certification flow by taking memory as reference library. All steps remain same as already discussed in previous libraries. So, only new issues are discussed faced during this certification.

LIBRARY	**H9I_WEBGEN_ST_SPUHD-10297**
TECHNOLOGY	**hcmos9i**

Netlist (CERT_MEM.ref.v) and constraint (CERT_MEM.sdc) file are generated with help of IPSCREEN tool.

Fig. 4.2.1 CERT_MEM.ref.v

```
module COUNTER_1_width10_DW01_inc_10_0 ( A, SUM );
  input [9:0] A;
  output [9:0] SUM;
  wire   \carry[9] , \carry[8] , \carry[7] , \carry[6] , \carry[5] ,     \carry[4] , \carry[3] ,
\carry[2] , n1;
  .. ……..
  endmodule
module COUNTER_1_width10 ( count, clk, rst );
  output [9:0] count;
  input clk, rst;
  wire   N23, N24, N25, N26, N27, N28, N29, N30, N31, N32, n12, n13, n14, n15,
        n16, n17, n18, n19, n20, n21;
  wire   [9:0] temp;
.....
  FD1QLLP temp_regx0x ( .CP(clk), .D(n21), .Q(temp[0]) );
  FD1QLLP temp_regx1x ( .CP(clk), .D(n20), .Q(temp[1]) );
```

```
  FD1QLLP temp_regx2x ( .CP(clk), .D(n19), .Q(temp[2]) );
  FD1QLLP temp_regx3x ( .CP(clk), .D(n18), .Q(temp[3]) );
  FD1QLLP temp_regx4x ( .CP(clk), .D(n17), .Q(temp[4]) );
  FD1QLLP temp_regx5x ( .CP(clk), .D(n16), .Q(temp[5]) );
  FD1QLLP temp_regx6x ( .CP(clk), .D(n15), .Q(temp[6]) );
  FD1QLLP temp_regx7x ( .CP(clk), .D(n14), .Q(temp[7]) );
  FD1QLLP temp_regx8x ( .CP(clk), .D(n13), .Q(temp[8]) );
  FD1QLLP temp_regx9x ( .CP(clk), .D(n12), .Q(temp[9]) );
   ...……..
endmodule
module CERT_MEM ( clk, rst, dataout );
  input clk, rst;
  output dataout;
  wire   [10:1] ibw1;
  wire   [127:1] ibw2;
  wire   [127:1] ibw3;
  wire   [11:1] ibw4;
  wire   [8:1] ibw5;
  wire   [9:1] ibw6;
  wire   [32:1] ibw7;
  wire   [13:1] ibw8;
  wire   [23:1] ibw9;
  wire   [127:1] obw1;
  wire   [4:1] a;
  wire   [8:1] obw2;
  wire   [32:1] obw3;
  wire   [23:1] obw4;
  wire   [12:1] g;
   ……..
  COUNTER_1_width10 inst_c1 ( .count(ibw1), .clk(clk), .rst(rst) );
  COUNTER_1_width127_1 inst_c2 ( .count(ibw2), .clk(clk), .rst(rst) );
```

```
  COUNTER_1_width127_0 inst_c3 ( .count(ibw3), .clk(clk), .rst(rst) );
  COUNTER_1_width11 inst_c4 ( .count(ibw4), .clk(clk), .rst(rst) );
  COUNTER_1_width8 inst_c5 ( .count(ibw5), .clk(clk), .rst(rst) );
  COUNTER_1_width9 inst_c6 ( .count(ibw6), .clk(clk), .rst(rst) );
  COUNTER_1_width32 inst_c7 ( .count(ibw7), .clk(clk), .rst(rst) );
  COUNTER_1_width13 inst_c8 ( .count(ibw8), .clk(clk), .rst(rst) );
  COUNTER_1_width23 inst_c9 ( .count(ibw9), .clk(clk), .rst(rst) );
  COUNTER_2_width127 inst_c10 ( .a(obw1), .clk(clk), .rst(rst), .dataout(a[1]) );
  COUNTER_2_width8 inst_c11 ( .a(obw2), .clk(clk), .rst(rst), .dataout(a[2]) );
  COUNTER_2_width32 inst_c12 ( .a(obw3), .clk(clk), .rst(rst), .dataout(a[3]) );
  COUNTER_2_width23 inst_c13 ( .a(obw4), .clk(clk), .rst(rst), .dataout(a[4]) );
  COUNTER_2_width4 inst_COUNTER_2 ( .a(a), .clk(clk), .rst(rst), .dataout(dataout) );
  COUNTER_1_width12 inst_COUNTER_1 ( .count(g), .clk(clk), .rst(rst) );
  SPUHD9Igp_1024x127m8_b inst1_SPUHD9Igp_1024x127m8_b ( .A(ibw1), .D(ibw2),
       .M(ibw3), .Q(obw1), .CK(clk), .CSN(g[1]), .OEN(g[2]), .WEN(g[3]) );
  SPUHD9Igp_2048x8m16 inst1_SPUHD9Igp_2048x8m16 ( .A(ibw4), .D(ibw5),
.Q(obw2),    .CK(clk), .CSN(g[4]), .OEN(g[5]), .WEN(g[6]) );
  SPUHD9Igp_288x32m8 inst1_SPUHD9Igp_288x32m8 ( .A(ibw6), .D(ibw7), .Q(obw3),
       .CK(clk), .CSN(g[7]), .OEN(g[8]), .WEN(g[9]) );
  SPUHD9Igp_8128x23m16 inst1_SPUHD9Igp_8128x23m16 ( .A(ibw8), .D(ibw9), .Q(
       obw4), .CK(clk), .CSN(g[10]), .OEN(g[11]), .WEN(g[12]) );
endmodule
```

Fig. 4.2.2 CERT_MEM.sdc

```
#######################################################
#  Created by Design Compiler write_sdc on Tue Jan  2 16:02:53 2007
#######################################################
set sdc_version 1.4
create_clock -period 10 -waveform {0 5} [get_ports {clk}]
set_input_delay 1 -max -clock "clk" [get_ports {rst}]
set_input_delay 0.2 -min -clock "clk" [get_ports {rst}]
```

```
set_output_delay 1 -max -clock "clk" [get_ports {dataout}]
set_output_delay 0.2 -min -clock "clk" [get_ports {dataout}]
set_clock_uncertainty  0.1 [get_clocks {clk}]
set_wire_load_mode "enclosed"
set_input_transition -max 1 [get_ports {rst}]
set_input_transition -min 0.2 [get_ports {rst}]
set_load -pin_load 1 [get_ports {dataout}]
set_load -min -pin_load 1 [get_ports {dataout}]
```

4.2.1 PAR Flow:

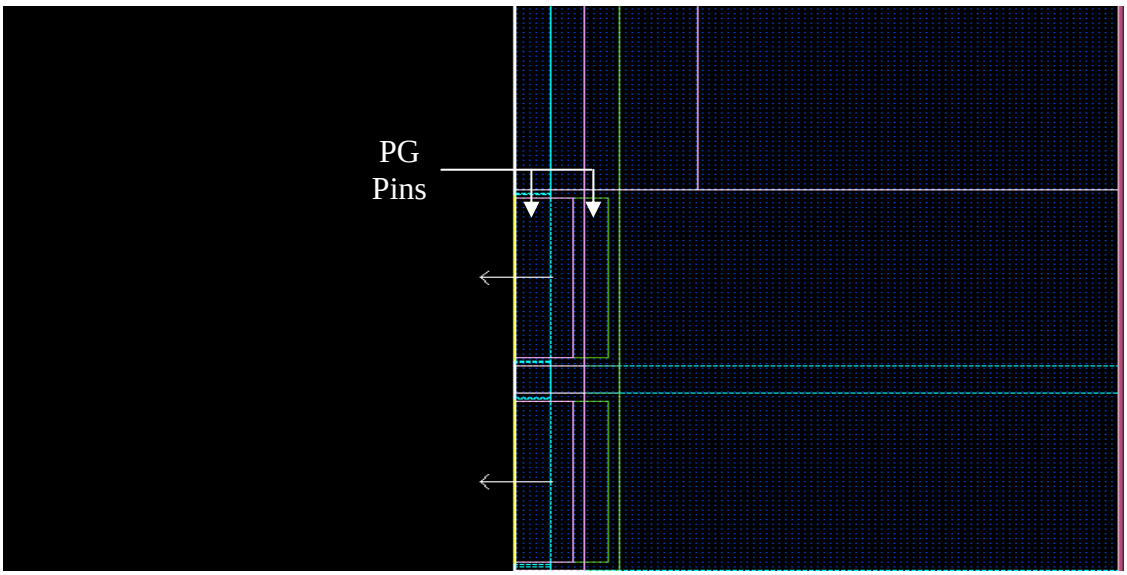


Fig.4.2.1.1 Abstract view of SPUHD9gp_2048*8m16

Metal 3: ———

Metal 4: ———

For the Placement and Routing, Astro flow used is same as one used for MACRO Libraries. Different bugs were entered for problem pertaining to DRCs.

- Found an error of VIA misalignment during DRC.

Bug Details:-

MISSING VIA BLOCKAGE in ABSTRACT.

PG pins are available in metal3 as well as metal 4. Connectivity should be checked in

both metal layers. Metal 3 PG pins and metal 4 PG pins are connected with help of VIA. Now VIA Blockage is not present in the abstract view (.fram) of memory cuts. The VIAs between power pin vdd/gnd in M3/M4 are not in the abstract. As a result, during pre-routing, router does not see the VIA. The pre-route connect the two pin with stacked VIA M3/M4 => DRC error of VIA misalignment.

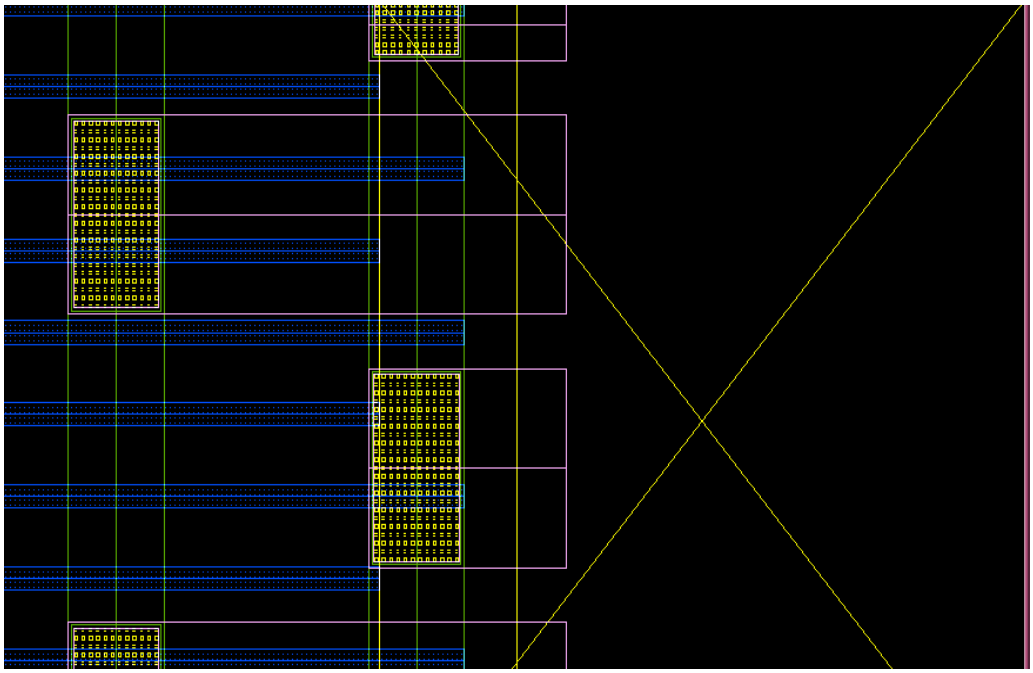


Fig. 4.2.1.2 After PG Routing

- For the cut: SPUHD9Igp_8128x23m16, there is a drc error for the following rule: R34.2.c & it indicates: Min. M3 space if at least one metal line width and length are > 10um 0.60um

- **Bug Details:-**

  On stretching the Metal3 pin for connection to power stripe, this length of M3 pin extends beyond 10um & results in a DRC error with adjacent metal3 rail running inside the memory.

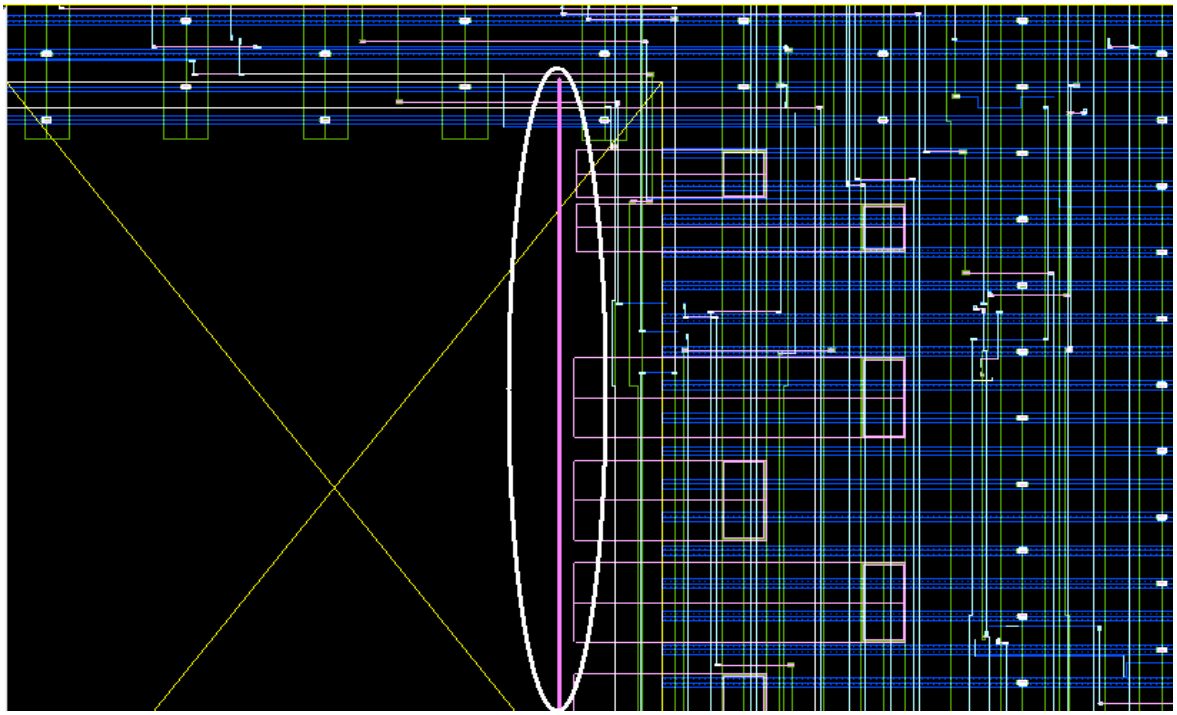


Fig. 4.2.1.3 After PG Routing

PAR flow generates three output files which are inputs for signoff flow.

Fig. 4.2.1.4 CERT_MEM.v

```
module COUNTER_1_width32_DW01_inc_32_0 (A , SUM );
input  [31:0] A ;
output [31:0] SUM ;
……..
ENLLP U5 (.Z ( SUM[31] ) , .B ( n1 ) , .A ( \carry[31] ) ) ;
HA1LL U1_1_7 (.S ( SUM[7] ) , .CO ( \carry[8] ) , .B ( \carry[7] ) , .A ( A[7] ) ) ;
HA1LL U1_1_6 (.S ( SUM[6] ) , .CO ( \carry[7] ) , .B ( \carry[6] ) , .A ( A[6] ) ) ;
HA1LL U1_1_5 (.S ( SUM[5] ) , .CO ( \carry[6] ) , .B ( \carry[5] ) , .A ( A[5] ) ) ;
HA1LL U1_1_4 (.S ( SUM[4] ) , .CO ( \carry[5] ) , .B ( \carry[4] ) , .A ( A[4] ) ) ;
HA1LL U1_1_3 (.S ( SUM[3] ) , .CO ( \carry[4] ) , .B ( \carry[3] ) , .A ( A[3] ) ) ;
HA1LL U1_1_2 (.S ( SUM[2] ) , .CO ( \carry[3] ) , .B ( \carry[2] ) , .A ( A[2] ) ) ;
HA1LL U1_1_1 (.S ( SUM[1] ) , .CO ( \carry[2] ) , .B ( A[0] ) , .A ( A[1] ) ) ;
IVLLX05 U6 (.Z ( n1 ) , .A ( A[31] ) ) ;
HA1LL U1_1_15 (.S ( SUM[15] ) , .CO ( \carry[16] ) , .B ( \carry[15] ) , .A ( A[15] ) ) ;
```

```
HA1LL U1_1_14 (.S ( SUM[14] ) , .CO ( \carry[15] ) , .B ( \carry[14] ) , .A ( A[14] ) ) ;
HA1LL U1_1_13 (.S ( SUM[13] ) , .CO ( \carry[14] ) , .B ( \carry[13] ) , .A ( A[13] ) ) ;
HA1LL U1_1_12 (.S ( SUM[12] ) , .CO ( \carry[13] ) , .B ( \carry[12] ) , .A ( A[12] ) ) ;
HA1LL U1_1_11 (.S ( SUM[11] ) , .CO ( \carry[12] ) , .B ( \carry[11] ) , .A ( A[11] ) ) ;
HA1LL U1_1_10 (.S ( SUM[10] ) , .CO ( \carry[11] ) , .B ( \carry[10] ), .A ( A[10] ) ) ;
……
HA1LL U1_1_24 (.S ( SUM[24] ) , .CO ( \carry[25] ) , .B ( \carry[24] ) , .A ( A[24] ) ) ;
endmodule
module COUNTER_1_width32 (clk_G1B8I28ASTHIRNet324 ,
   clk_G1B8I18ASTHIRNet355 , clk_G1B8I21ASTHIRNet371 ,
   clk_G1B8I24ASTHIRNet396 , clk_G1B8I17ASTHIRNet438 ,
   clk_G1B8I7ASTHIRNet476 , rst , clk , count , clk_G1B8I10ASTHIRNet497 ,
   clk_G1B8I26ASTHIRNet509 , clk_G1B8I13ASTHIRNet512 ,
   clk_G1B8I6ASTHIRNet560 );
input  clk_G1B8I28ASTHIRNet324 ;
input  clk_G1B8I18ASTHIRNet355 ;
input  clk_G1B8I21ASTHIRNet371 ;
input  clk_G1B8I24ASTHIRNet396 ;
input  clk_G1B8I17ASTHIRNet438 ;
input  clk_G1B8I7ASTHIRNet476 ;
input  rst ;
……
module CERT_MEM (dataout , rst , clk );
output dataout ;
input  rst ;
input  clk ;
……
wire [32:1] ibw7 ;
wire [9:1] ibw6 ;
wire [8:1] ibw5 ;
wire [11:1] ibw4 ;
```

```
wire [127:1] ibw3 ;
wire [127:1] ibw2 ;
wire [10:1] ibw1 ;
…….
BFLL count_regx20xASTfhInst1887 (.Z ( ibw2_21_ASTfhNet2186 )
   , .A ( ibw2_21_ASTfhNet1341 ) ) ;
BFLL count_regx122xASTfhInst1888 (.Z ( ibw3_123_ASTfhNet2187 )
   , .A ( ibw3_123_ASTfhNet1263 ) ) ;
BFLL count_regx3xASTfhInst1889 (.Z ( ibw9_4_ASTfhNet2188 ) , .A ( ibw9[4] ) ) ;
BFLL count_regx124xASTfhInst1890 (.Z ( ibw3_125_ASTfhNet2189 )
   , .A ( ibw3_125_ASTfhNet1303 ) ) ;
BFLL count_regx24xASTfhInst1891 (.Z ( ibw7_25_ASTfhNet2190 )
   , .A ( ibw7_25_ASTfhNet1280 ) ) ;
BFLL count_regx75xASTfhInst1892 (.Z ( ibw2_76_ASTfhNet2191 )
   , .A ( ibw2_76_ASTttcNet78 ) ) ;
BFLL count_regx28xASTfhInst1893 (.Z ( ibw7_29_ASTfhNet2192 )
   , .A ( ibw7_29_ASTfhNet1275 ) ) ;
BFLL count_regx116xASTfhInst1894 (.Z ( ibw2_117_ASTfhNet2193 )
   , .A ( ibw2_117_ASTttcNet263 ) ) ;
BFLL count_regx21xASTfhInst1895 (.Z ( ibw7_22_ASTfhNet2194 ) , .A ( ibw7[22] ) ) ;
BFLL count_regx3xASTfhInst1896 (.Z ( ibw5_4_ASTfhNet2195 )
   , .A ( ibw5_4_ASTfhNet1933 ) ) ;
BFLL count_regx25xASTfhInst1897 (.Z ( ibw7_26_ASTfhNet2196 )
   , .A ( ibw7_26_ASTfhNet2019 ) ) ;
BFLL count_regx0xASTfhInst1898 (.Z ( ibw8_1_ASTfhNet2197 ) , .A ( ibw8[1] ) ) ;
BFLL count_regx27xASTfhInst1899 (.Z ( ibw3_28_ASTfhNet2198 )
   , .A ( ibw3_28_ASTfhNet1276 ) ) ;
BFLL count_regx81xASTfhInst1900 (.Z ( ibw2_82_ASTfhNet2199 )
   , .A ( ibw2_82_ASTttcNet84 ) ) ;
BFLL count_regx76xASTfhInst1901 (.Z ( ibw2_77_ASTfhNet2200 ) , .A ( ibw2[77] ) ) ;
BFLL count_regx123xASTfhInst1902 (.Z ( ibw3_124_ASTfhNet2201 )
```

```
   , .A ( ibw3_124_ASTfhNet1212 ) ) ;
…….
Endmodule
```

Fig.4.2.1.5 CERT_MEM.routed.def

```
# DEF OUT API
# User Name : certif18
# Date : Thu Jan 11 14:42:28 2007
#
VERSION 5.3 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;
DESIGN CERT_MEM ;
TECHNOLOGY hcmos9i ;
UNITS DISTANCE MICRONS 1000 ;
DIEAREA ( 0 0 ) ( 2706000 2460820 ) ;

ROW STD_ROW_0 CORE 0 2366930 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_1 CORE 0 2371850 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_2 CORE 0 2376770 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_3 CORE 0 2381690 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_4 CORE 0 2386610 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_5 CORE 0 2391530 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_6 CORE 0 2396450 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_7 CORE 0 2401370 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_8 CORE 0 2406290 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_9 CORE 0 2411210 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_10 CORE 0 2416130 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_11 CORE 0 2421050 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_12 CORE 0 2425970 FS DO 6600 BY 1 STEP 410 4920 ;
```

```
ROW STD_ROW_13 CORE 0 2430890 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_14 CORE 0 2435810 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_15 CORE 0 2440730 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_16 CORE 0 2445650 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_17 CORE 0 2450570 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_18 CORE 0 2455490 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_19 CORE 0 2184890 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_20 CORE 0 2189810 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_21 CORE 0 2194730 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_22 CORE 0 2199650 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_23 CORE 0 2204570 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_24 CORE 0 2209490 FS DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_25 CORE 0 2214410 N DO 6600 BY 1 STEP 410 4920 ;
ROW STD_ROW_26 CORE 0 2219330 FS DO 6600 BY 1 STEP 410 4920 ;
…….
AVNT_via1_HH-720-210-ALL-2-1 ( 1381085 * ) AVNT_via1_HH-720-210-ALL-2-1 ( *
1996085 )
 NEW M1 ( 1381085 1995880 ) AVNT_via1_VH-310-700-ALL-1-2
 NEW M2 ( 2291900 1566815 ) AVNT_via2-720-310-ALL-2-1
 NEW M3 ( 2291695 1566815 ) ( 2297805 * )
 NEW M3 ( 2297600 1566815 ) AVNT_via3-960-550-ALL-2-1
 NEW M1 ( 2291900 1566815 ) AVNT_via1_VH-720-290-ALL-2-1
 NEW M2 ( 2134460 1390925 ) AVNT_via2-720-310-ALL-2-1
 NEW M3 ( 2125645 1390925 ) ( 2134665 * )
 NEW M2 ( 2125850 1390925 ) AVNT_via2-720-310-ALL-2-1
 NEW M2 ( 2125645 1389695 ) ( * 1390925 )
 NEW M1 ( 2134870 1390925 ) AVNT_via1_VH-720-290-ALL-2-1
 + USE SIGNAL ;
 - inst_COUNTER_1/n24 ( inst_COUNTER_1/U37 Z )  (
inst_COUNTER_1/count_regx1x D )  ( inst_COUNTER_1/temp_regx1x D )
 + ROUTED M1 ( 1560255 1085065 ) AVNT_via1_VH-720-290-ALL-2-1
```

```
 NEW M2 ( 1560255 1085065 ) ( * 1093675 )
  AVNT_via1_HH-720-210-ALL-2-1 ( 1479895 * )
 NEW M1 ( 1474565 1129550 ) AVNT_via1_VH-310-700-ALL-1-2
 NEW M2 ( 1474770 1129345 ) AVNT_via2-720-310-ALL-2-1
 NEW M3 ( 1474565 1129345 ) ( 1476045 * )
  AVNT_via3-960-550-ALL-2-1 ( * 1093675 ) AVNT_via3-960-550-ALL-2-1
AVNT_via2-720-310-ALL-2-1
  AVNT_via1_HH-720-210-ALL-2-1 ( 1479895 * )
 NEW M1 ( 1479950 1065385 ) ( 1480305 * )
 NEW M1 ( 1479950 1065385 ) AVNT_via1_HH-720-210-ALL-2-1
 NEW M2 ( 1479950 1065385 ) ( 1479895 * ) ( * 1093675 )
 NEW M1 ( 1479690 1093675 ) AVNT_via1_HH-720-210-ALL-2-1
 + USE SIGNAL ;
END NETS
END DESIGN
```

4.2.2 Signoff flow:

Signoff flow is same as it is mentioned in above macro flow. And all results are clean. So here Delay Analysis and Timing Analysis are discussed.

4.2.2.1 Delay Analysis:

Table 4.2.2.1.1 Delay Analysis for memory

Net Type	Total	RC N/W	Not Annotated
Internal nets	4133	4133	0
Boundary/port nets	3	3	0
Total	4136	4136	0

Check Not Annotated column. In this column result should be zero. If it is non zero, then there must be an error which can be read from the log file.

4.2.2.2 Timing Analysis:

In timing analysis setup and hold violations should be checked and corrected first. This can be done by giving different constraints in sdc file.

create_clock -period 8 -waveform {0 4} [get_ports {clk}]

set_clock_uncertainty 0.5 [get_clocks {clk}]

Table 4.2.2.2.1 Analysis Coverage for memory (Uncertainty 0.5 ns)

Type of Check	Total	Met	Violated	Untested
setup	1117	557 (51%)	560 (49%)	0 (0%)
hold	1117	54 (5%)	1063 (95%)	0 (0%)
min_period	4	4 (100%)	0 (0%)	0 (0%)
min_pulse_width	1506	1506 (100%)	0 (0%)	0 (0%)

Here setup violations are 49% and hold violations are 95% .

create_clock -period 8 -waveform {0 4} [get_ports {clk}]

set_clock_uncertainty 0.1 [get_clocks {clk}]

Table 4.2.2.2.2 Analysis Coverage for memory (Period – 8 ns)

Type of Check	Total	Met	Violated	Untested
setup	1117	583 (52%)	534 (48%)	0 (0%)
hold	1117	1117 (0%)	0 (0%)	0 (0%)
min_period	4	4 (100%)	0 (0%)	0 (0%)
min_pulse_width	1506	1506 (100%)	0 (0%)	0 (0%)

Here by decreasing uncertainty hold violations are reduced to zero.

create_clock -period 10 -waveform {0 5} [get_ports {clk}]

set_clock_uncertainty 0.1 [get_clocks {clk}]

Table 4.2.2.2..3 Analysis Coverage for memory (Period – 10 ns)

Type of Check	Total	Met	Violated	Untested
setup	1117	786 (70%)	331 (30%)	0 (0%)
hold	1117	1117 (100%)	0 (0%)	0 (0%)
min_period	4	4 (100%)	0 (0%)	0 (0%)
min_pulse_width	1506	1506 (100%)	0 (0%)	0 (0%)

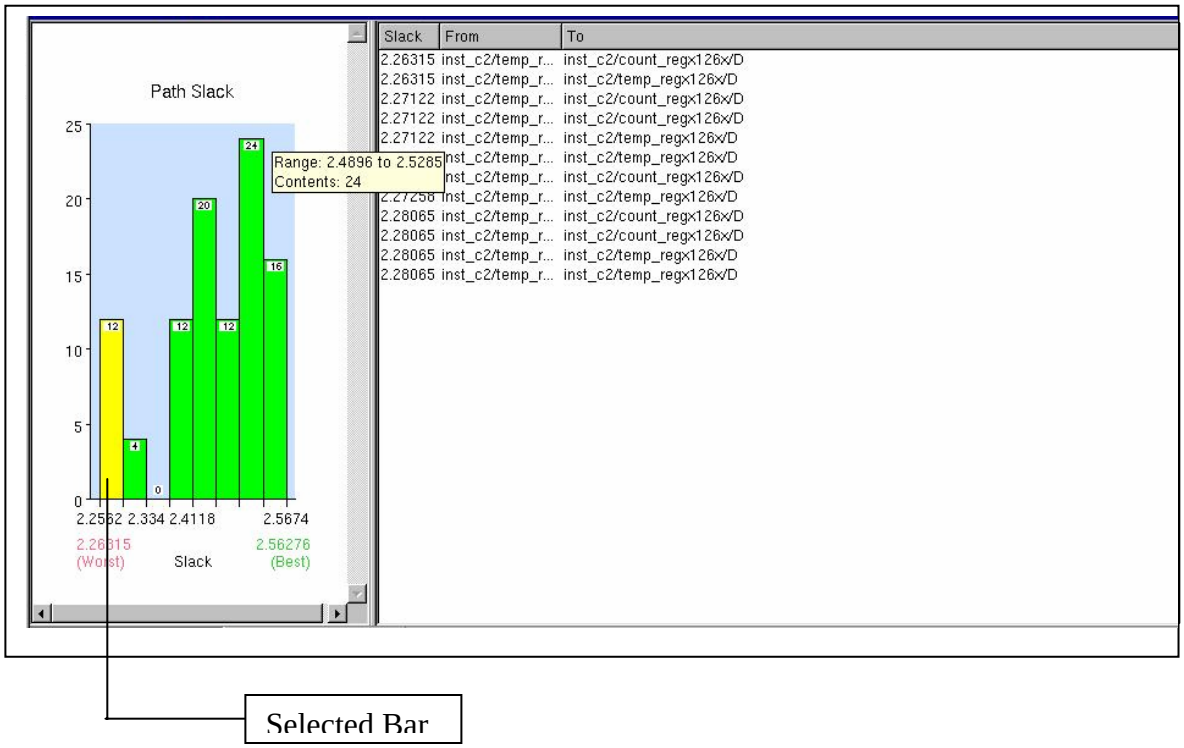


Fig. 4.2.2.2.1 Path Slack

Here, by increasing the clock period, setup violations are reduced to 30%, this can further be reduced to zero by increasing the clock period to 12 ns.

4.3 STD Cell Libraries:

LIBRARY	**CORE65HV50**
LIBRARY	**SHIFT65_HV_50**
LIBRARY	**SHIFT65_HV_LP50**
TECHNOLOGY	**cmos065lp_7m4x0y2z**

These libraries are to be certified by using Magma flow as described in Section-2. Netlist (CERT_MEM.ref.v) and constraint (CERT_MEM.sdc) file are generated with help of IPSCREEN tool.

Fig. 4.3.1 cert_65hv_gate.v

```
module adder_DW01_add_0 (CI,CO,A,B,SUM );
 input CI;
 input [7:0] A;
 input [7:0] B;
 output CO;
 output [7:0] SUM;
 wire n1;
 wire n10;
 wire n11;
 ..........................................................
wire xpog_arrayx2xx5x;
 wire xpog_arrayx2xx6x;
 HV65_50_IVX4 U0(
  .A(CI),
  .Z(xg_arrayx0xxx1x));
 HV65_50_NAND2X4 U0_1_0(
  .A(A[0]),
  .B(B[0]),
  .Z(xg_arrayx0xx0x) );
 HV65_50_NAND2X4 U0_1_1(
```

```
  .A(A[1]),
  .B(B[1]),
  .Z(xg_arrayx0xx1x) );
 HV65_50B_NAND2X4 U0_1_2(
  .A(A[2]),
  .B(B[2]),
  .Z(xg_arrayx0xx2x) );
module cert_65hv (CLK,CLK_LOGIC_1,RESET,SYNCH,PORT_OUT );
 input CLK,CLK_LOGIC_1,RESET;
 output SYNCH;
 output [7:0] PORT_OUT;
 wire CS;
 wire CS0;
 wire CS1;
 wire CS2;
 wire CS3;
 wire CS4;
 wire RW;
 wire SYNOPSYS_UNCONNECTED_1;
 wire [15:0] ADDRESS;
 wire [7:0] DATA;
 addr_decoder AD0(
  .CS(CS),
  .CS0(CS0),
  .CS1(CS1),
  .CS2(CS2),
  .CS3(CS3),
  .CS4(CS4),
.ADDRESS({ADDRESS[15],ADDRESS[14],ADDRESS[13],ADDRESS[12],ADDRESS[
11],ADDRESS[10],ADDRESS[9],ADDR
ESS[8],ADDRESS[7],ADDRESS[6],ADDRESS[5],ADDRESS[4],ADDRESS[3],ADDRE
```

```
SS[2],ADDRESS[1],ADDRESS[0]}) );
 spare_bloc I_spare_bloc(  .dummy_spare(SYNOPSYS_UNCONNECTED_1) );
 core_seq_bench P0(
  .CLK(CLK),
  .CS(CS0),
  .RESET(RESET),
  .RW(RW),
  .ADDRESS({ADDRESS[1],ADDRESS[0]}),
.DATA({DATA[7],DATA[6],DATA[5],DATA[4],DATA[3],DATA[2],DATA[1],DATA[
0]}) );
 core_logic_bench P1(
  .CLK(CLK),
  .CLK_LOGIC(CLK_LOGIC_1),
  .CS(CS1),
  .RESET(RESET),
  .RW(RW),
.ADDRESS({ADDRESS[4],ADDRESS[3],ADDRESS[2],ADDRESS[1],ADDRESS[0]}),
.DATA({DATA[7],DATA[6],DATA[5],DATA[4],DATA[3],DATA[2],DATA[1],DATA[
0]}) );
 softram_8x4 P2(
  .CLK(CLK),
  .CS(CS2),
  .RW(RW),
  .ADDRESS({ADDRESS[1],ADDRESS[0]}),
.DATA({DATA[7],DATA[6],DATA[5],DATA[4],DATA[3],DATA[2],DATA[1],DATA[
0]}) );
 outport P3(
  .CLK(CLK),
  .CS(CS3),
  .RW(RW),
  .SYNCH(SYNCH),
```

```
.DATA({DATA[7],DATA[6],DATA[5],DATA[4],DATA[3],DATA[2],DATA[1],DATA[
0]}),
.DATA_OUT({PORT_OUT[7],PORT_OUT[6],PORT_OUT[5],PORT_OUT[4],PORT_O
UT[3],PORT_OUT[2],PORT_OUT[1],PORT_OUT[0]}) );
 progROM P4(
  .CLK(CLK),
  .CS(CS4),
.ADDRESS({ADDRESS[8],ADDRESS[7],ADDRESS[6],ADDRESS[5],ADDRESS[4],A
DDRESS[3],ADDRESS[2],ADDRESS[1],ADDRESS[0]}),
.DATA({DATA[7],DATA[6],DATA[5],DATA[4],DATA[3],DATA[2],DATA[1],DATA[
0]}) );
 cpu_start_address512 U0(
  .CLK(CLK),
  .CS(CS),
  .RESET(RESET),
  .RW(RW),
.ADDRESS({ADDRESS[15],ADDRESS[14],ADDRESS[13],ADDRESS[12],ADDRESS[
11],ADDRESS[10],ADDRESS[9],ADDRESS[8],ADDRESS[7],ADDRESS[6],ADDRESS
[5],ADDRESS[4],ADDRESS[3],ADDRESS[2],ADDRESS[1],ADDRESS[0]}),
.DATA({DATA[7],DATA[6],DATA[5],DATA[4],DATA[3],DATA[2],DATA[1],DATA[
0]}) );
Endmodule
```

Fig. 4.3.2 cert_65hv.sdc

```
create_clock -name CLK -period 6 -waveform { 0 3 } [get_ports {CLK}]
create_clock -name CLK_LOGIC_1 -period 12 -waveform { 0 6 } [get_ports
{CLK_LOGIC_1}]
set sdc_version 1.5
set_clock_latency -max  0.6 [get_clocks {CLK}]
set_clock_latency -max  1.2 [get_clocks {CLK_LOGIC_1}]
set_clock_latency -min  0.3 [get_clocks {CLK}]
```

```
set_clock_latency -min  0.6 [get_clocks {CLK_LOGIC_1}]
set_clock_transition -fall -max  0.8813 [get_clocks {CLK_LOGIC_1}]
set_clock_transition -fall -max  0.8813 [get_clocks {CLK}]
set_clock_transition -fall -min  0.06647 [get_clocks {CLK_LOGIC_1}]
set_clock_transition -fall -min  0.06647 [get_clocks {CLK}]
set_clock_transition -rise -max  1.279 [get_clocks {CLK_LOGIC_1}]
set_clock_transition -rise -max  1.279 [get_clocks {CLK}]
set_clock_transition -rise -min  0.07253 [get_clocks {CLK_LOGIC_1}]
set_clock_transition -rise -min  0.07253 [get_clocks {CLK}]
set_clock_uncertainty  0.6 [get_clocks {CLK}]
set_clock_uncertainty  1.2 [get_clocks {CLK_LOGIC_1}]
```

4.3.1 PAR Flow:

For Placement and Routing Magma tool is used.

Flow Setup:

In order to run the flow, the user must create/use a seed.tcl file, where a number of settings for the design is specified.

- **Workspace Setup :**

magGenerateTasks [-dir <dirName>] [-force] [-tasks<taskName>]

This generates the design workspace with the following directories:

$G_WORKSPACE/

MAGMA/

EXPORT/

IMPORT/

LIBIMPORT/

PLACEANDROUTE/

PT_TIMING_ANALYSIS/

SHIFT65_HV_50 and SHIFT65_HV_LP50 libraries have two supplies vdd and vdd1.For SHIFT65_HV_50 vdd1 pin is available in metal1 which is shown below.

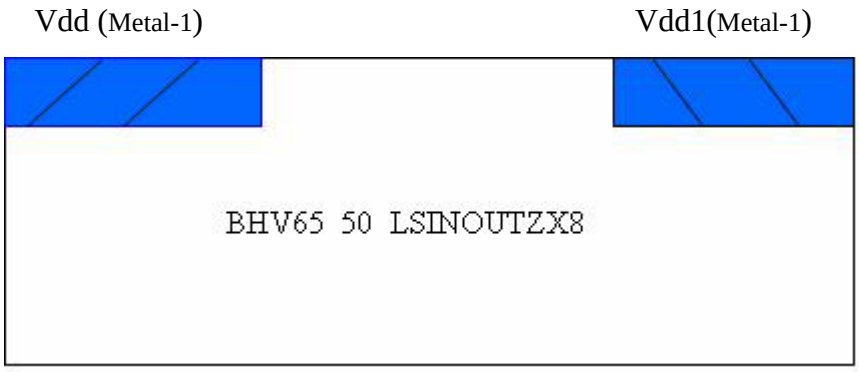


Fig 4.3.1.2 BHV65_50_LSINOUTZX8

So when vdd rail is passed from this cell it short vdd1 pin with vdd rail. To avoid shorting cells of SHIFT65_HV_50 are placed at boundary. And vdd rail is cut from that place which is shown in fig. below.

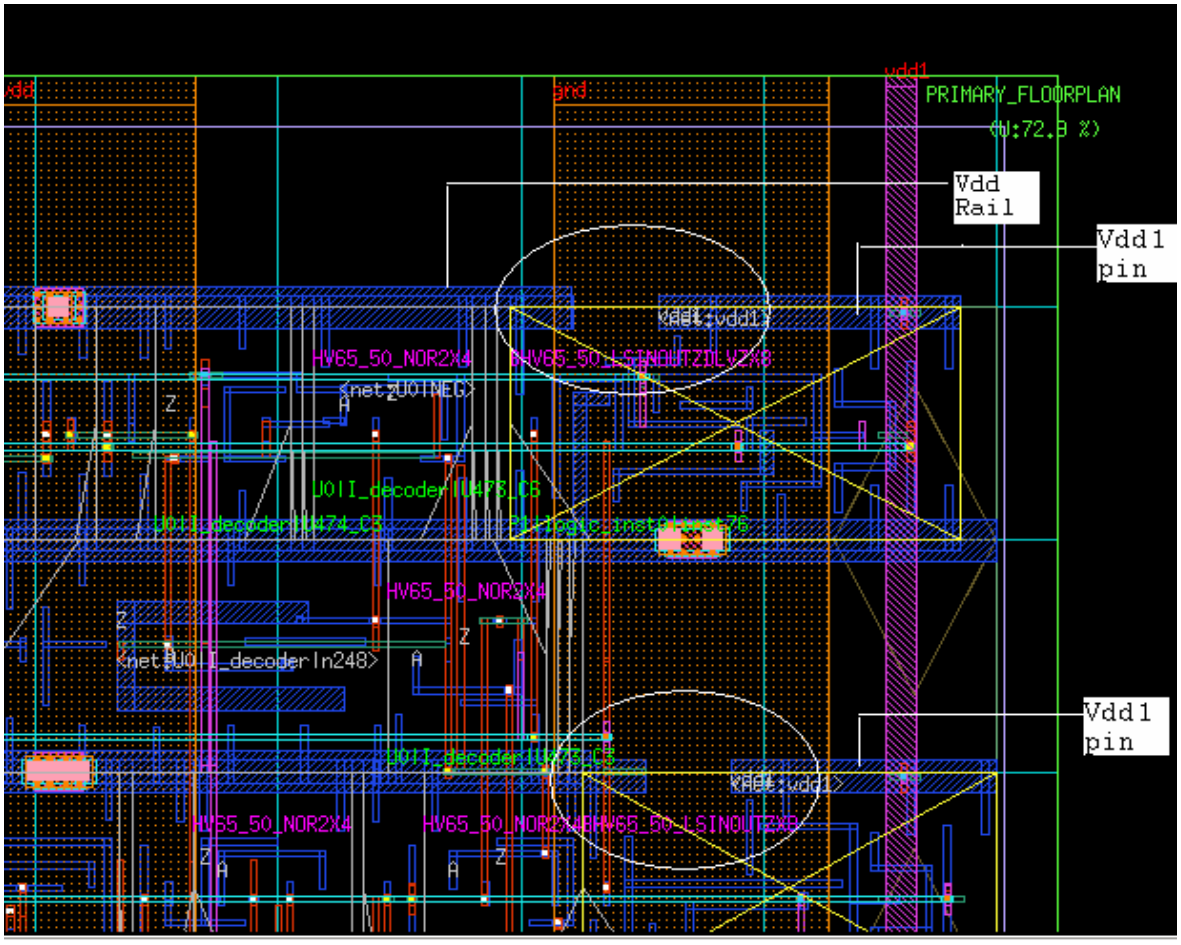


Fig. 4.3.1.2 After PG Routing (Magma view)

When vdd rail is cut, then p-substrate of these two cells remain unconnected and gives LVS error. So extra filler tie is added to reconnect p-substrate. And label vdd1 is added at that place.

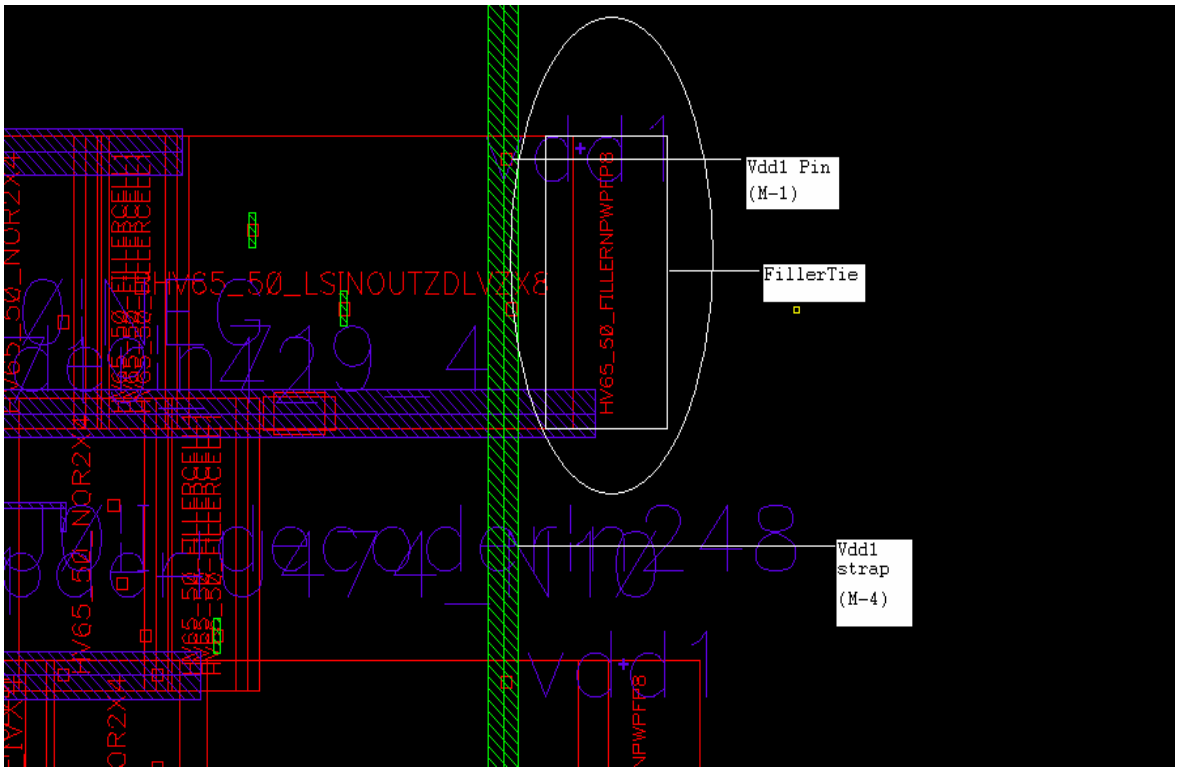


Fig. 4.3.1.3 OPUS view after adding filler tie

Magma PAR flow generates following output files.

cert_65hv.gdscad

cert_65hv.v

cert_65hv.routed.def

cert_65hv.routed.lef

Fig.4.3.1.4 cert_65hv.v

```
// Verilog file generated from Magma Bedrock database
// Sun Mar 18 20:37:49 2007
// Bedrock Root: Magma_root

// Entity:core_logic_bench_interface Model:core_logic_bench_interface Library:L0
module core_logic_bench_interface (CLK, CS, END_RUN, RESET, RUN, RW,
```

```
ADDRESS,
    DATA, OUT_STACK, PAT0, PAT1, PAT10, PAT11, PAT12, PAT13, PAT14,
PAT15,
    PAT2, PAT3, PAT4, PAT5, PAT6, PAT7, PAT8, PAT9, SEL_INST, CLK_14,
CLK_17,   CLK_4, CLK_16, CLK_6, CLK_18, CLK_13, CLK_15);
  input CLK, CS, END_RUN, RESET, RW, CLK_14, CLK_17, CLK_4, CLK_16,CLK_6,
    CLK_18, CLK_13, CLK_15;
  output RUN;
  input [4:0] ADDRESS;
  input [15:0] OUT_STACK;
  output [5:0] PAT0;
  output [5:0] PAT1;
  output [5:0] PAT10;
  output [5:0] PAT11;
  output [5:0] PAT12;
  output [5:0] PAT13;
  output [5:0] PAT14;
  output [5:0] PAT15;
……
wire n100, n101, n102, n103, n104, n105, n106, n107, n108, n109, n11, n110,
    n111, n112, n113, n114, n115, n116, n117, n118, n119, n12, n120, n121,
    n122, n123, n124, n125, n126, n127, n128, n129, n13, n130, n131, n132,
    n133, n134, n135, n136, n137, n138, n139, n140, n141, n142, n143, n144,
    n145, n146, n147, n148, n149, n15, n150, n151, n152, n153, n154, n155,
    n156, n157, n158, n159, n16, n160, n161, n162, n163, n164, n165, n166,
    n167, n168, n169, n170, n171, n172, n173, n174, n175, n176, n177, n178,
.....
// Entity:cert_65hv Model:cert_65hv Library:L0
module cert_65hv (CLK, CLK_LOGIC_1, RESET, SYNCH, PORT_OUT);
  input CLK, CLK_LOGIC_1, RESET;
  output SYNCH;
```

```
  output [7:0] PORT_OUT;
  wire [15:0] ADDRESS;
  wire [7:0] DATA;
  wire CS, CS0, CS1, CS2, CS3, CS4, RW, SYNOPSYS_UNCONNECTED_1,
SYNCH_pred,
    \PORT_OUT[7]_pred , \PORT_OUT[6]_pred , \PORT_OUT[5]_pred ,
    \PORT_OUT[4]_pred , \PORT_OUT[3]_pred , \PORT_OUT[2]_pred ,
    \PORT_OUT[1]_pred , \PORT_OUT[0]_pred , RESET_next, \AD0|n1 , \AD0|n14 ,
    \AD0|n15 , \AD0|n4 , \AD0|n7 , \I_spare_bloc|sparenet1BB ,
    \I_spare_bloc|sparenet2BB , \I_spare_bloc|sparenet3Z ,
    \I_spare_bloc|sparenet4Z , \I_spare_bloc|sparenet5Z , \AD0|U16_N8 ,
    \AD0|U1_N5 , \AD0|U8_N5 , ADDRESS_3_1, \AD0|n5_1 , ADDRESS_12_1,
    ADDRESS_9_1, RESET_next_1, RESET_next_2, RESET_next_3, \AD0|n9_2 ,
    \AD0|n7_1 , \AD0|n9_3 , ADDRESS_9_2, ADDRESS_7_1, \AD0|n5_2 , RW_1,
    ADDRESS_6_1, ADDRESS_8_1, CLK_1, CLK_2, CLK_3, CLK_4, CLK_5, CLK_6,
CLK_7,   CLK_8, CLK_9, CLK_10, CLK_11, CLK_12, CLK_13, CLK_14, CLK_15,
CLK_16,CLK_17, CLK_18, CLK_19, CLK_20;
……..
 HV65_50_IVX26 CLK_L2 (.A(CLK_16), .Z(CLK_14));
  HV65_50_IVX26 CLK_L3_2 (.A(CLK_14), .Z(CLK_7));
  HV65_50_IVX26 CLK_L0 (.A(CLK), .Z(CLK_18));
  HV65_50_IVX26 CLK_L1_1 (.A(CLK_18), .Z(CLK_16));
  HV65_50B_BFX8 \BUF_PORT_OUT[6]  (.A(\PORT_OUT[6]_pred ),
.Z(PORT_OUT[6]));
  HV65_50B_BFX8 \BUF_PORT_OUT[5]  (.A(\PORT_OUT[5]_pred ),
.Z(PORT_OUT[5]));
  HV65_50B_BFX8 \BUF_PORT_OUT[0]  (.A(\PORT_OUT[0]_pred ),
.Z(PORT_OUT[0]));
  HV65_50_BFX17 CLK_SYNC_SKEW_3 (.A(CLK_17), .Z(CLK_20));
  HV65_50_BFX26 CLK_SYNC_SKEW (.A(CLK_17), .Z(CLK_19));
  HV65_50_IVX9 CLK_L1 (.A(CLK_18), .Z(CLK_17));
```

```
  HV65_50B_BFX8 \BUF_PORT_OUT[3]  (.A(\PORT_OUT[3]_pred ),
.Z(PORT_OUT[3]));
  HV65_50_IVX26 CLK_L3_1 (.A(CLK_14), .Z(CLK_5));
  HV65_50_NOR3X4 \AD0|U3_C3  (.A(ADDRESS_3_1), .B(\AD0|n1 ), .C(\AD0|n7 ),
     .Z(CS3));
HV65_50_NOR3X4 \AD0|U7_C5  (.A(ADDRESS[3]), .B(ADDRESS[2]),.C(\AD0|n7_1 ),
     .Z(CS2));
  HV65_50_NOR2X4 \AD0|U8_C5  (.A(\AD0|U8_N5 ), .B(\AD0|n7_1 ), .Z(CS0));
  HV65_50B_BFX8 \BUF_PORT_OUT[4]  (.A(\PORT_OUT[4]_pred ),
.Z(PORT_OUT[4]));
  HV65_50B_BFX4 BL1_R_BUF_6 (.A(\AD0|n7 ), .Z(\AD0|n7_1 ));
  HV65_50_OR3X4 \AD0|U2_C2  (.A(ADDRESS[2]), .B(ADDRESS[1]),
.C(ADDRESS[0]),
     .Z(\AD0|n1 ));
Endmodule
```

Fig. 4.3.1.5 cert_65hv.routed.def

```
VERSION 5.3 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "/" ;
BUSBITCHARS  "[]" ;
DESIGN cert_65hv ;
TECHNOLOGY cmos065_7m4x0y2z ;
UNITS DISTANCE MICRONS 1000 ;
DIEAREA ( 0 0 ) ( 312800 312800 )  ;
TRACKS X 0 DO 6256 STEP 50 LAYER M1 ;
TRACKS Y 0 DO 6256 STEP 50 LAYER M1 ;
TRACKS X 0 DO 6256 STEP 50 LAYER M2 ;
TRACKS Y 0 DO 6256 STEP 50 LAYER M2 ;
TRACKS X 0 DO 6256 STEP 50 LAYER M3 ;
TRACKS Y 0 DO 6256 STEP 50 LAYER M3 ;
```

```
TRACKS X 0 DO 6256 STEP 50 LAYER M4 ;
TRACKS Y 0 DO 6256 STEP 50 LAYER M4 ;
TRACKS X 0 DO 6256 STEP 50 LAYER M5 ;
……
   - P0/counter0|E_CLK_2
            ( P0/CLK_SYNC_SKEW_2 Z )
            ( P0/counter0|ff1|latch0 GN )
       + NONDEFAULTRULE double_all
       + SOURCE NETLIST
       + USE CLOCK
       + ROUTED M2 ( 14200 26400 )  ( * 30150 )
         NEW M2  ( 14200 30050 ) via1a_cert_65hv1
         NEW M2  ( 14200 26400 ) via1_cert_65hv_NBT ;
      - CLK_20
            ( CLK_SYNC_SKEW_3 Z )
            ( P0/counter1|I_cbuf CP )
       + NONDEFAULTRULE double_all
       + SOURCE NETLIST
       + USE CLOCK
       + ROUTED M1 ( 17000 37800 )  ( 18250 * )
         NEW M1 ( 18250 37900 )  ( 19450 * )
         NEW M1 ( 17000 37800 )  ( * 38600 )
         NEW M1 ( 19450 37900 )  ( * 38200 )
         NEW M1 ( 18250 37800 )  ( * 37900 ) ;
END NETS
END DESIGN
```

Fig. 4.3.1.6 cert_65hv.routed.lef

```
VERSION 5.3 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "|" ;
```

```
BUSBITCHARS "[]" ;
UNITS
DATABASE MICRONS 1000 ;
END UNITS
########## LAYERS ################
LAYER poly
    TYPE MASTERSLICE ;
END poly
LAYER contact
    TYPE MASTERSLICE ;
END contact
LAYER NWELL
    TYPE MASTERSLICE ;
END NWELL
LAYER ODThin
    TYPE MASTERSLICE ;
END ODThin
LAYER ODThick
    TYPE MASTERSLICE ;
END ODThick
……..
LAYER M1
    TYPE ROUTING ;
    WIDTH 0.100 ;
    SPACING 0.090 ;
    PITCH 0.200 ;
    DIRECTION HORIZONTAL ;
END M1
LAYER VIA1
    TYPE CUT ;
END VIA1
```

```
LAYER M2
      TYPE ROUTING ;
      WIDTH 0.100 ;
      SPACING 0.100 ;
      PITCH 0.200 ;
      DIRECTION VERTICAL ;
END M2
LAYER VIA2
      TYPE CUT ;
END VIA2
LAYER M3
      TYPE ROUTING ;
      WIDTH 0.100 ;
      SPACING 0.100 ;
      PITCH 0.200 ;
      DIRECTION HORIZONTAL ;
END M3
……
LAYER VIA5 ;
      RECT -0.180 -0.180 0.180 0.180 ;
      SPACING 0.700 BY 0.700 ;
END via5a
VIARULE via6a GENERATE
 LAYER M6 ;
      DIRECTION VERTICAL ;
      OVERHANG 0.020 ;
      METALOVERHANG 0 ;
 LAYER M7 ;
      DIRECTION HORIZONTAL ;
      OVERHANG 0.080 ;
      METALOVERHANG 0 ;
```

```
 LAYER VIA6 ;
      RECT -0.180 -0.180 0.180 0.180 ;
      SPACING 0.700 BY 0.700 ;
END via6a
END LIBRARY
```

4.3.2 Signoff flow:

Signoff flow is same as it is mentioned in above macro flow. All results are clean. So, only Delay and Timing Analysis are discussed.

4.3.2.1 Delay Analysis:

Table 4.3.2.1.1 Delay Analysis

Net Type	Total	RC network	Not Annotated
Internal nets	6188	6188	0
Boundary/port nets	12	12	0
Total	6200	6200	0

Review the log file for the errors, if any.

4.3.2.2 Timing Analysis:

Table 4.3.2.2.1 Annotation Coverage

Timing check type	Total	Annotated	Not Annotated
cell setup arcs	453	453	0
cell hold arcs	453	453	0
cell min pulse width arcs	868	868	0

create_clock -name CLK -period 4 -waveform { 0 2 } [get_ports {CLK}]
set_clock_uncertainty 0.6 [get_clocks {CLK}]

Table 4.3.2.2.2 Analysis Coverage for Std_cell (Period – 4 ns)

Type of Check	Total	Met	Violated	Untested
Setup	445	199	246	0
Hold	445	64	381	0
min_pulse_width	868	868	0	0

create_clock -name CLK -period 6 -waveform { 0 3 } [get_ports {CLK}]
set_clock_uncertainty 0.6 [get_clocks {CLK}]

Table 4.3.2.2.3 Analysis Coverage for Std_cell (Period – 6 ns)

Type of Check	Total	Met	Violated	Untested
Setup	445	369	76	0
Hold	445	64	381	0
min_pulse_width	868	868	0	0

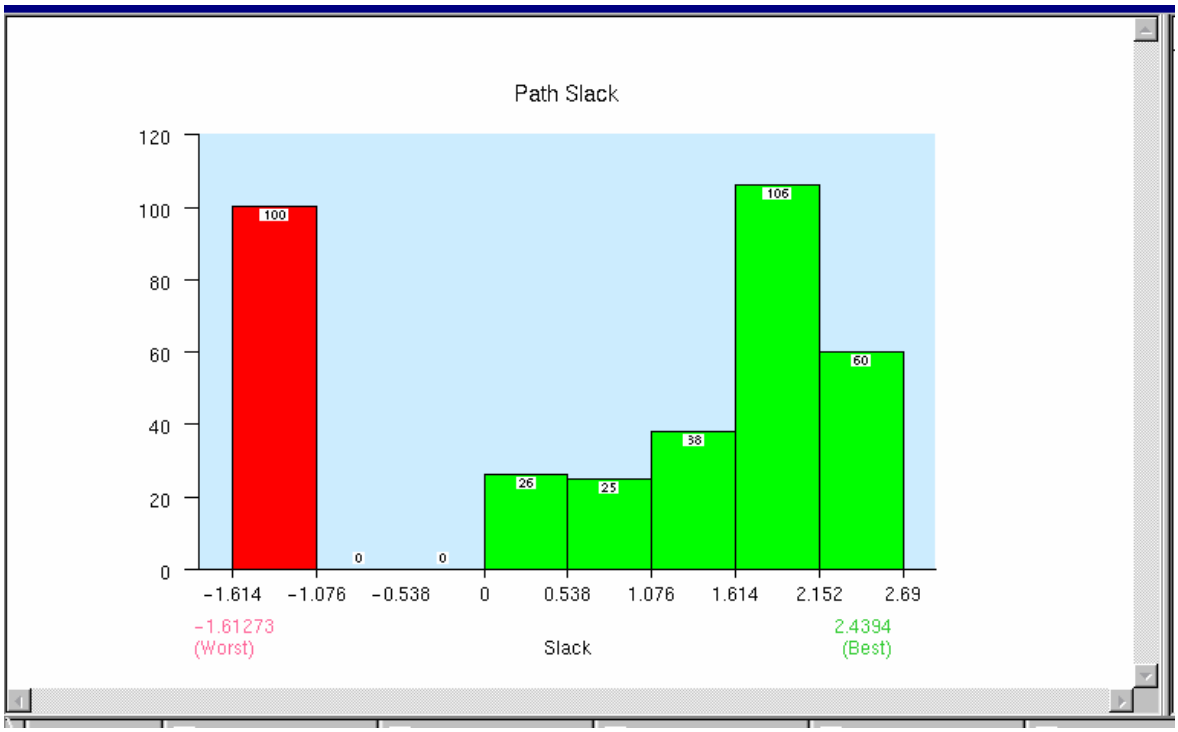


Fig. 4.3.2.2.1 Path Slack for above table

create_clock -name CLK -period 8 -waveform { 0 4 } [get_ports {CLK}]
set_clock_uncertainty 0.6 [get_clocks {CLK}]

Table 4.3.2.2.4 Analysis Coverage for Std_cell (Period – 8 ns)

Type of Check	Total	Met	Violated	Untested
Setup	445	393	45	0
Hold	445	64	381	0
min_pulse_width	868	868	0	0

create_clock -name CLK -period 12 -waveform { 0 6 } [get_ports {CLK}]
set_clock_uncertainty 0.6 [get_clocks {CLK}]

Table 4.3.2.2.5 Analysis Coverage for Std_cell (Period – 12 ns)

Type of Check	Total	Met	Violated	Untested
Setup	445	445	0	0
Hold	445	64	381	0
min_pulse_width	868	868	0	0

Here by increasing time period setup violations become zero. Now by decreasing clock_uncertainty hold violations reduced to zero.

create_clock -name CLK -period 12 -waveform { 0 6 } [get_ports {CLK}]
set_clock_uncertainty 0.2 [get_clocks {CLK}]

Table 4.3.2.2.6 Analysis Coverage for Std_cell (Uncertainty 0.2ns)

Type of Check	Total	Met	Violated	Untested
Setup	445	445	0	0
Hold	445	440	5	0
min_pulse_width	868	868	0	0

4.4 Method-1 Certification:

This section explains the Method-1 certification of Libraries. Method-1 certification is used when there are updates in .lib file or when there are back-end updates. When front end update is there then .def and .lef files are not available, wireload

models are used to generate .dspf file. This is then used in delay calculation in order to generate sdf file which will be then used for back-annotation using ncsim/ncverilog.
For method 1, following need to be done:

- Generate the gate level netlist using IPSCREEN. Follow the IPSCREEN FLOW.
- For Front end change, the delay calculation is done. This basically corresponds to the flow being used during METHOD 1 certification. The following commands are used:

Example: Certification of library **CORI65LPHVT**

cd 65LP

65LP: mkdir ANNOTATION

65LP: cd ANNOTATION

ANNOTATION: cp ../IPSCREEN/FUSION/db/CERT_CORE.ref.v CERT_CORE .v

The above command copies the .v file generated from IPSCREEN to ANNOTATION for delay calculation. Next copy .synopsys_pt.setup in ANNOTATION directory.

ANNOTATION:

cp ~certif18/Rachna/CERTIFICATION/CORE/.synopsys_pt.setup .

Next link the library to this directory using the following command:

ANNOTATION: ln -s

../LIBRARIES/CORI65LPHVT/libs/CORI65LPHVT_wc_1.05V_125C .db .

CORI65LPHVT not contain standard cells like OR, AND, INVERTER so CORELPHVT library is included.

ANNOTATION: ln -s

../LIBRARIES/CORE65LPHVT/libs/ CORE65LPHVT_wc_1.05V_125C .db .

Next edit .synopsys_pt.setup by using the following command:

ANNOTATION: vi .synopsys_pt.setup

This file contains the search paths and library paths for the libraries .db files. The syntax to be edited is as follows.(#: comment)

Fig. 4.4.1: .synopsys_pt.setup file

```
#/*******************************************************/
#/* setup search and library paths                      */
#/*******************************************************/
```

```
set search_path [list . ./]
set link_path "* CORI65LPHVT_wc_1.05V_125C.db
CORE65LPHVT_wc_1.05V_125C.db "
read_file CORI65LPHVT_wc_1.05V_125C.db
read_file CORE65LPHVT_wc_1.05V_125C.db
#read_file CORE9GPHS.db
# This file from HandOffKit 1.8.2.2
#source synopsys_unicad_pt.setup
```

So here the link path for .db files of the libraries are given and the commands to read these .db files.

Next source PRIMETIME using the following command:

ANNOTATION: s /sw/cshrc/solaris/primetime_2002.09-sp1-2

Next the command file is edited by first copying its contents.

ANNOTATION:

cp ~certif18/CERTIFICATION/Rachna/CORE/command .

ANNOTATION: vi command

The contents of this file are as follows:

Fig. 4.4.2 command file

```
read_verilog CERT_CORE.v
##read_verilog Cert_SPHS.v
current_design CERT_CORE
link
current_design CERT_CORE
set_wire_load_model -library [get_libs {CORI65LPHVT}] -name  area_1Kto2K
[get_designs {CERT_CORE}]
set_input_transition 4.0 [all_inputs]
set_load 4.0 [all_outputs]
##write_sdf -signi 5 1.sdf
write_sdf -map pt.map -version 3.0 -signi 5 -context none pt.sdf
write_sdf -map verilog.map -version 3.0 -signi 5 -context verilog verilog.sdf
```

```
read_sdf pt.sdf
exit
```

The wire load model is taken from the log file of the ipscreen. The sdf files for pt.map (primetime) which is used for back-annotation are written.

The map file contains the primetime mapping view for the cells in the library. Its contents can be seen using the command:

ANNOTATION: vi ../LIBRARIES/CORI65LPHVT/behaviour/verilog/<>.verilog .map .

Similarly, the .lib file is also there in libs directory. The verilog.map file takes its values for the cells from this .lib file.

If there are more than one library, concatenate the two .map files to one and name them pt.map using the command:

ANNOTATION: cat ../LIBRARIES/../libs/<>.pt .map ../LIBRARIES/../libs/<>.pt .map > pt .map

ANNOTATION: cat ../LIBRARIES/../behaviour/verilog /<>.verilog .map ../LIBRARIES/../behaviour/libs/<>.verilog .map > verilog .map

Next run the command file with the following commands and write the 3 files:

ANNOTATION: pt_shell -f command | & tee log_sdf

If there are errors running this command, check the data provided in .synopsys_pt.setup file using the command:

ANNOTATION: vi .synopsys_pt .setup

and edit it.

Next carefully analyze the log_sdf file for errors using the command:

ANNOTATION: vi log_sdf

If there are errors with the net arcs from input, no need to worry because this is not included in the standard cell library. However, if there are errors with the cell arcs it should be reported.

This completes delay calculation. Next back-annotation is performed as follows.

4.4.1 Back-Annotation for Verilog Models:

Next in order to go for Verilog back-annotation the CERT_CORE.v file is edited and is changed to a test bench file such that the design can be back-annotated.

ANNOTATION: vi CERT_CORE .v

Following changes are done in the .v file

- Add the timescale using the following verilog syntax

  `timescale 1ns/10ps

- And at the last (after the last endmodule) add the following statements:

Figure 4.4.1.1: Verilog module for Back-annotation

```
initial
        $ sdf_annotate("verilog.sdf",CERT_CORE);
initial
#10 $finish;
Endmodule
```

The above verilog syntax basically says that annotate CERT_CORE verilog.sdf

Next go to IPSCREEN and copy .synopsys_dc.setup file which is already generated during synthesis.

ANNOTATION:

cp ../IPSCREEN/DCFX01_CORE/synthesis/.synopsysy_dc .setup .

ANNOTATION:vi .synopsys_dc.setup

Fig. 4.4.1.2: .synopsys_dc.setup file

```
#/*********************************************************/
#/* setup search and library paths                        */
#/*********************************************************/
set search_path [list . ./]
set link_path "* CORI65LPHVT_wc_1.05V_125C.db
```

```
CORE65LPHVT_wc_1.05V_125C.db "
define_design_lib WORK -path ./work
if {[file isdirectory ./work] == 0} {
file mkdir ./work}
source
$env(SYNOPSYSHANDOFFKITROOT)/unicad_setup/synopsys_unicad_dc.tcl.setup
set search_path  { .
/home/rci/CERTIF/PRODUCTS/CMOS065LP/LIBRARIES/CORI65LPHVT@1
.0.a@20070112.0/libs
/home/rci/CERTIF/PRODUCTS/CMOS065LP/LIBRARIES/CORE65LPHVT@4
.1@20060620.0/libs }
set target_library { CORI65LPHVT_wc_1.25V_125C.db
CORE65LPHVT_wc_1.25V_125C.db}
set link_library  { * CORI65LPHVT_wc_1.25V_125C.db
CORE65LPHVT_wc_1.25V_125C.db }
```

Next source Design Analyzer using the following command:

ANNOTATION: s /sw/cshrc/solaris/synps_u-2003.06-sp1-3

Launch Design Analyzer.

ANNOTATION: design_analyzer &

This opens the Synopsys Design Analyzer window. However, if there are errors, check .synopsys_dc.setup.

In DesignAnalyzer window:

GO TO SETUP --> DEFAULTS

Search Path : This will have the path of the library set in .synopsys_dc .setup

Link Library :

Target Library :

read

set current_design

link

uniquify

save as .vhd

This will generate the VHDL file.

The CERT_CORE.vhd contains the cells whose parameters are to be read from the library so in order to include the library there is a script which is run as follows:

ANNOTATION: ~certif1/bin/add_lib2vhd .pl CERT_CORE .vhd CERT_CORE_new .vhd

Thus CERT_CORE_new.vhd is created which has the libraries included. Edit the names of the libraries accordingly.

For VHDL Back-annotation, the following 3 files are required:

cds.lib hdl.var sdf_command

ANNOTATION: cp ~certif18/CERTIFICATION/Rachna/CORE/cds.lib .

ANNOTATION: cp ~certif18/CERTIFICATION /Rachna/CORE/hdl.var .

ANNOTATION:

cp ~certif18/CERTIFICATION/Rachna/CORE/sdf_command .

ANNOTATION: vi cds .lib

The contents of cds.lib file are as shown in figure 4.6.2.1.

Fig. 4.4.1.3: cds.lib file

```
INCLUDE $CDS_INST_DIR/tools/inca/files/cds.lib
DEFINE CORE65LPHVT ./lib_CORE65LPHVT_nc
DEFINE CORI65LPHVT ./lib_CORI65LPHVT_nc
DEFINE work ./work
```

Fig. 4.4.1.4: sdf_command file

```
Compiled_sdf_file=”sdf.compiled”,
Log_file=”log”;
```

So the above directories are made as follows:

ANNOTATION: mkdir lib_CORE65LPHVT_nc lib_CORE65LPHVT_nc work

Next compile the VHDL models as follows:

ANNOTATION: ncvhdl -messages -logfile core.log –work ../LIBRARIES/CORI65LPHVT/behaviour/VHDL/CORI65LPHVT.components

ANNOTATION: ncvhdl -messages -logfile core.log –work ../LIBRARIES/CORI65LPHVT/behaviour/VHDL/CORI65LPHVT.VHDL-FUNCT

Repeat the procedure for all the available libraries.And finally do for design itself.

ANNOTATION: ncvhdl -messages -logfile core.log -work work CERT_CORE_new.vhd

Next do elaboration of the current design.

ANNOTATION: ncelab -messages -sdf_cmd_file sdf_command -work work :cfg_SYN

Next simulate the design using ncsim.

ANNOTATION: ncsim -messages -logfile <name of logfile> <name of snapshot>

The name of the snapshot is same which is mentioned in the end of `ncelab` command. If there are errors at any stage check the cause of error and again perform that step.

The Log files to check

- For ncverilog, check the file ncverilog.log.
- For ncelab, check the log file mentioned in sdf.cmd file.
- For ncsim, check the log file mentioned in the command.

This ends the back-annotation for VHDL models.

4.4.2 SPIVsCDL Check:

For SpiVsCDL check first make spivscdl directory using following command.

mkdir spivscdl

cd spivscdl

Then execute following commands.

- netlist2spice -cfg netlist2spi.cfg –attx $UCDPRJDIR/IPSCREEN/ FUSION/ Synthesis/ CERT_CORE.power.attx -verilog $UCDPRJDIR/ IPSCREEN/ FUSION/ Synthesis/ CERT_CORE.ref.v +cdl top.spi
- cp calibre.config.spi CERT_CORE.lay.net
- bsub -I -q crnd-batch calibrerun -lvs -batch -lvsLayoutPaths "cert_90i.lay.net" -lvsLayoutPrimary cert_90i -lvsSourcePath "c alibre.config.cdl" -lvsSourcePrimary cert_90i -lvsRunDir ./ -options " - lvsAutoMatch 0 -lvsRunWhat NVN -lvsUseHCells 1 -lvsH CellsFile top.hcell”

This completes the Method 1 certification.

Chapter – 5

Conclusion And Future Scope

CHAPTER-5
CONCLUSION AND FUTURE SCOPE

5.1 Conclusion:

Verification means checking against an accepted entity-generally, a higher level of specifications. In IC design, verification is divided into two major tasks:
1. Specification Verifications and 2. Implementation Verification.

The objective of implementation verification is to find out, within practical limits, whether the system will work after implementation. In the domain of nanometer ASIC technologies (180 nm and below), designs cannot meet their realistic timing objectives; creating the well known “timing closure problem”. Timing closure is now considered the biggest area of difficulty for ASIC performance-oriented designs. The certification flow provides the solutions to all these issues.

5.2 Future Scope:

The field of certification is very vast and as the device dimensions are shrinking according to the Moore’s law, it creates new design constraints and issues like skew tolerant circuit design, low power issues etc. The future work lies in understanding these issues in detail.

References

[1] Smith M.J.S., “Application Specific Integrated Circuits”, Pearson Education publishers, 2004.

[2] Rochit Rajsuman, “System-on-a-Chip”, Artech publishers, 2000

[3] Negus Christopher, Red Hat Fedora Linux 3-Bible, Wiley Dreamtech India Pvt. Ltd, 2005.

[4] Kang S.M. and Yusuf Leblibici, “CMOS Digital Integrated Circuits: Analysis and Design”, Tata McGraw Hill.

[5] Myung-Soo Jang, Hoon-Sang Jin, Byoung-Hyun Lee, “A Hierarchical Design System for Deep Submicron Technology” , IEEE ,2000.

[6] Swinnen Marc, Arnout Guido, Timing issues related to the automated placement and routing of high performance ASICS, IEEE, 1991.

[7] William H. Kao, Chi-Yuan Lo, Mark Basel, Parasitic Extraction: Current State of the Art and Future Trends, Proceedings Of The IEEE, Vol. 89, No. 5, May 2001.

[8] Philip Nowe , Timing Analysis is Everything , Circuit Cellar, the Magazine for Computer Applications. Issue 160 November 2003

[9] Documents on Certification and Macro library.

[10] Documents on Avanti flow, Magma flow and Signing-off flow.

[11] User guides of tools: Avanti Astro, PrimeTime, Formality, Star-RCXT, Calibre.