University of Science and Technology
Zewail City

Nanotechnology and Nanoelectronics Engineering

Digital Implementation of the RISC-V Based Open PULP Core

A Graduation Project
Submitted in Partial Fulfillment of
B.Sc. Degree Requirements in
Nanotechnology and Nanoelectronics Engineering

Prepared By

Ahmed Hassan	201600694
Mohamed Sayed	201600643
Omar Monzer	201600529
Seif Eldeen Emad	201600398

Supervised By

Associate prof. Dr. Hassan Mostafa
Si-Vision

2021

Abstract

The objective of this thesis is to carry out an ASIC Physical Design of the RISC-V based CV32E40P (RI5CY) PULP “Parallel Ultra-Low Power” Core. The aim of the RI5CY core is to satisfy the computational demands of IoT applications achieving higher code density, performance, and energy efficiency

The project aims to deliver the GDSII file by going through the RTL-to-GDSII flow. “NANGATE 45 nm” PDK was used through the flow. The core was implemented with three different synthesis flows (Topographical, Flat and Hierarchical) using Synopsys Design Compiler, followed by all main steps of Place & Route flow using Synopsys IC Compiler. Formal equivalence checking is performed after PnR process to ensure that both Pre-layout netlist and Post-layout netlist exhibit the same behavior using Synopsys Formality tool, and then Post-layout STA is done with Synopsys PrimeTime to ensure that design meets all timing requirements at a wide range of PDK operating corners. Finally, results were compared between the three implementation flows based on timing, area, power and synthesis runtime.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
CPU	Central Processing Unit
CTS	Clock Tree Synthesis
DC	Design Compiler
DEF	Design Exchange Format
DFT	Design For Testability
DRC	Design Rule Check
ECO	Engineering Change Order
EDA	Electronic design automation
FF	Flip Flops
GDSII	Geometrical Database Standard for Information Interchange
GRC	Global Route Congestion
ICC	IC Compiler
IoT	Internet of Things
ISA	Instruction Set Architecture
IP	Intellectual Property
LEF	Library Exchange Format
LVS	Layout Versus Schematic
MMMC	Multi Mode Multi Corner
NDR	Non Default Rules
PDK	Process Design Kit
PnR	Place and Route

PPA	Power, Area, Performance
PULP	Parallel Ultra-Low Power
RISC-V	Reduced Instruction Set Computer Five
RTL	Register Transfer Level
SDC	Synopsys Design Constraints
SoC	System on Chip
STA	Static Timing Analysis
TLU	Table Look Up
UPF	Unified Power Format
VHDL	VHSIC Hardware Description Language

Acknowledgements

First and foremost, we would like to praise and thank Allah, the almighty, for being by our side and blessing us with strength, guidance and knowledge to complete this work.

We would also like to thank our families and friends. Their tender love and unfailing support have always been the cementing force for building what we achieved.

We would like to express our sincere gratitude and appreciation to our supervisor Dr. Hassan Mostafa for his excellent guidance and invaluable support. In addition, we would like to thank our advisor at Si-Vision Eng. Hoda Thabet for her great help, caring and immense knowledge.

Chapter 1

Introduction

This chapter introduces the basic knowledge and brief background of topics related to this thesis. It also describes the objective behind this thesis and a map for this documentation in order to make it easier to navigate through different topics covered by it.

1.1 Motivation

Recently, there has been a shift towards open source hardware. The reason behind the increase in demand for open source hardware is that open specifications maximize the ability of programmers to take advantage of hardware features when optimizing software-defined solutions. Open hardware can also be more extensible and maximizes the ability of third-party programmers and partners to work with a given device [1].

Open hardware will also help to drive IoT adoption by creating a foundation for building low-cost, low-power, portable IoT solutions. The idea behind IoT is to have a network of computing devices, vehicles, embedded systems with sensors etc which generate, exchange and process data intelligently and continuously. About 50

billion devices are estimated to be connected with IoT by 2020, up from about 200 million in the year 2000 and 10 billion in 2013 [2].

RISC-V began in 2010 at the University of California, Berkeley along with many volunteer contributors. It is now growing rapidly as it is projected that 62.4 billion RISC-V CPU cores will be sold in 2025 which is about 6% of the overall CPU core market. RISC-V has broken down barriers in the semiconductor industry, bringing together different companies, industries, and geographies for open collaboration [3].

The Parallel Ultra Low Power (PULP) Platform started as a joint effort between the Integrated Systems Laboratory (IIS) of ETH Zürich and Energy-efficient Embedded Systems (EEES) group of the University of Bologna in 2013 to explore new and efficient architectures for ultra-low-power processing. PULP initiative aims to develop an open, scalable hardware and software research and development platform with the goal to break the energy efficiency barrier within a power envelope of a few milliwatts, as well as satisfy the computational demands of IoT applications requiring flexible processing of data streams generated by multiple sensors, such as accelerometers, low-resolution cameras, microphone arrays and vital signs monitors. They take an open-source approach based on the open-source RISC-V instruction set architecture [4].

1.2 RISC-V Architecture

RISC-V (pronounced "risk-five") is an open standard instruction set architecture (ISA) based on established reduced instruction set computer (RISC) principles. Being open source unlike most ISA; the RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture. A number of companies are offering or have announced RISC-V hardware, open source operating systems with RISC-V support are available and the instruction set is supported in several popular software toolchains.

In the case of RISC-V, only the ISA is standardized, leaving the microarchitecture and implementation to the processor developer. RISC-V is designed with a small, fixed-base ISA. It includes modular fixed-standard extensions that can be used with the majority of code. This architecture enables the development of application-specific extensions without needing to modify the standard ISA core. One of the advantages of RISC-V is that developers can optimize code with minimal memory needs and low power consumption while maintaining scalability and compatibility for future designs. There is also the opportunity to develop extensions to the basic ISA. A variety of security solutions have been developed to ensure secure processing with RISC-V [5].

RISC-V has a modular design, consisting of alternative base parts, with added optional extensions. The base specifies instructions (and their encoding), control flow, registers (and their sizes), memory and addressing, logic (i.e., integer) manipulation, and ancillaries. The base alone can implement a simplified general-purpose computer,

with full software support, including a general-purpose compiler. RISC-V has 32 (or 16 in the embedded variant) integer registers, and, when the floating-point extension is implemented, separate 32 floating-point registers. Except for memory access instructions, instructions address only registers. Like many RISC designs, RISC-V is a load–store architecture: instructions address only registers, with load and store instructions conveying to and from memory. RISC-V segregates math into a minimal set of integer instructions with add, subtract, shift, bitwise logic and comparing-branches. The integer multiplication instructions include signed and unsigned multiply and divide [5].

1.3 RI5CY Core

CV32E40P is a 4-stage in-order 32-bit RISC-V processor core. The ISA of CV32E40P has been extended to support multiple additional instructions including hardware loops, post-increment load and store instructions and additional ALU instructions that are not part of the standard RISC-V ISA. Figure 1.1 shows a block diagram of the core [4].

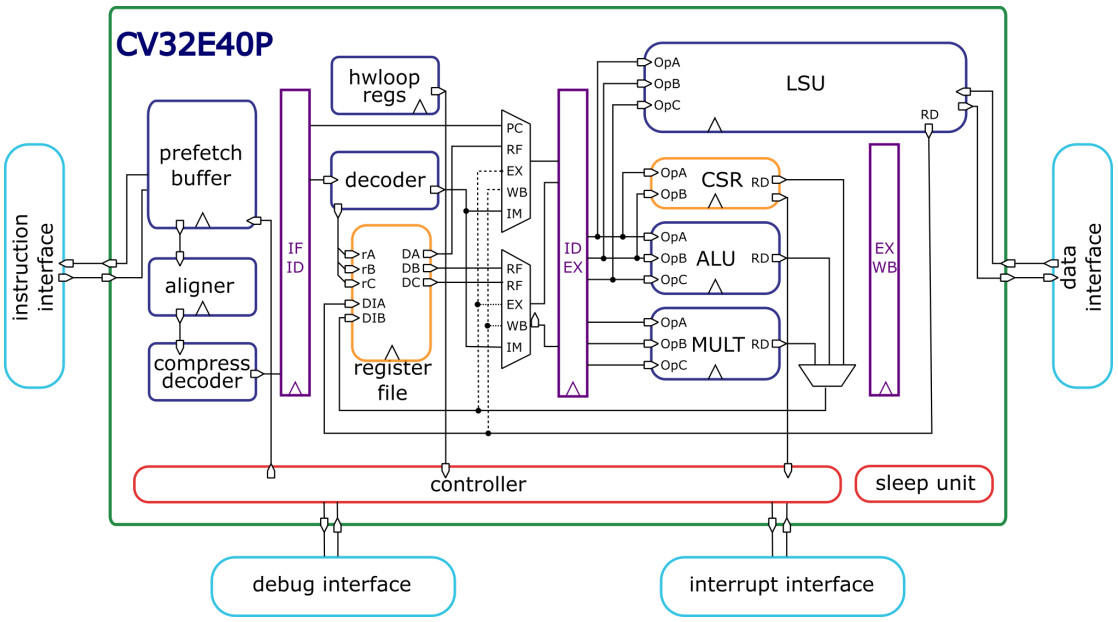


Figure 1.1: Block Diagram of CV32E40P RISC-V Core [4]

CV32E40P is a standards-compliant 32-bit RISC-V processor. Many features in the RISC-V specification are optional, and CV32E40P can be parametrized to enable or

disable some of them. CV32E40P supports the base instruction set: RV32I Base Integer Instruction Set, version 2.1. In addition, the following standard instruction set extensions are available.

Table 1.1: CV32E40P Standard Instruction Set Extensions [4]

Standard Extension	Version	Configurability
C: Standard Extension for Compressed Instructions	2.0	always enabled
M: Standard Extension for Integer Multiplication and Division	2.0	always enabled
Zicount: Performance Counters	2.0	always enabled
Zicsr: Control and Status Register Instructions	2.0	always enabled
Zifencei: Instruction-Fetch Fence	2.0	always enabled
F: Single-Precision Floating-Point	2.2	optionally enabled based on FPU parameter

Also, the following custom instruction set extensions are available.

Table 1.2: CV32E40P Custom Instruction Set Extensions [4]

Custom Extension	Version	Configurability
Xcorev: CORE-V ISA Extensions (excluding cv.elw)	1.0	optionally enabled based on PULP_XPULP parameter
Xpulpcluster: PULP Cluster Extension	1.0	optionally enabled based on PULP_CLUSTER parameter
Xpulpzfinx: PULP Share Integer (X) Registers with Floating Point (F) Register Extension	1.0	optionally enabled based on PULP_ZFINX parameter

CV32E40P has a 4-stage in-order completion pipeline, the 4 stages are [4]:

- Instruction Fetch (IF)

  Fetches instructions from memory via an aligning prefetch buffer, capable of fetching 1 instruction per cycle if the instruction side memory system allows. The IF stage also pre-decodes RISC-V compressed (RVC) instructions into RV32I base instructions.

- Instruction Decode (ID)

  Decodes fetched instruction and performs required registerfile reads. Jumps are taken from the ID stage.

- Execute (EX)

  Executes the instructions. The EX stage contains the ALU, Multiplier and Divider. Branches (with their condition met) are taken from the EX stage. Multi-cycle instructions will stall this stage until they are complete. The ALU, Multiplier and Divider instructions write back their result to the register file from the EX stage. The address generation part of the load-store-unit (LSU) is contained in EX as well.

- Writeback (WB)

  Writes the result of Load instructions back to the register file.

The following figure shows the CV32E40P Pipeline.

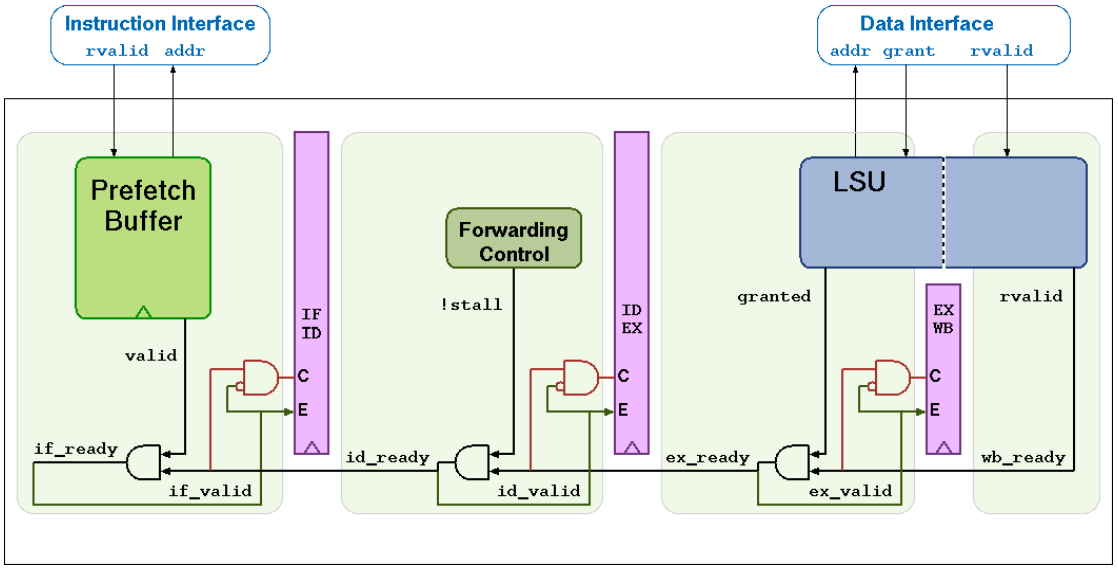


Figure 1.2: CV32E40P Pipeline [4]

Some instructions have a variable time which means that the instruction takes a minimum of 1 cycle and a maximum of 32 cycles. The cycle counts assume zero stall on the instruction-side interface and zero stall on the data-side memory interface.

1.4 Thesis Objective

The global semiconductor market has gone through tremendous growth due to the increasing consumption of consumer electronics devices across the globe. However, due to the challenges in overcoming the chip shortage and manufacturing limits in addition to high competition, it has been demanding to design faster, smaller and more complex devices.

The objective of our project is to carry out the digital implementation of RISC-V core on open PULP by going through the complete ASIC physical design from RTL to GDSII Implementation Flow. We went through all steps of design flow including synthesis, floorplanning, placement and routing (PnR), static timing analysis (STA) and equivalence checking. Also, we explored multiple design flows (Flat, Hierarchical and Topographical). Finally, we analyzed the results of each flow with respect to PPA (Power, Performance, Area).

We used the open source RTL of RISC-V based CV32E40P (RI5CY) core as a starting point. “NANGATE 45 nm” PDK was used throughout the flow. Synopsys tools were used to implement our work. Design Compiler (DC) was used for synthesis, followed by Synopsys IC Compiler (ICC) for PnR. Formal equivalence checking is performed after the PnR process using Synopsys Formality tool. Finally, post-layout STA is done with Synopsys PrimeTime.

1.5 Thesis Organization

This thesis documentation is divided into five chapters. Chapter1 is an introductory chapter that covers some background knowledge that is needed throughout this documentation which includes a short description of RISC-V ISA, in addition to an introduction to PULP RI5CY core. Also, it covers the thesis objective and this thesis organization.

In chapter two, a survey of the state of the art concerning the RISC-V cores is introduced. It presents a market and literature review of the design. It also includes an overview of the tools and techniques needed to build the state of the art system in addition to the technical approach. It also explains the project design including constraints, technical specifications and the block diagram of the design.

Chapter three describes the ASIC flow including synthesis, floorplan, placement, CTS, outing, STA, design validation, and chip finishing.

Chapter four describes the attempt to implement RI5CY core using different synthesis flows based on the methodologies mentioned in chapter three. It also shows the core differences between the three flows. In addition, it shows a step by step implementation of each flow and a guide for the tools used.

Chapter five gathers all the achieved results at each step of the three flows. These results include the achieved timing, the power consumption, the chip area, the congestion, etc.

Chapter six concludes the work done throughout the thesis comparing between the three design flows. Also, future work and possible modifications are presented.

Chapter 2

Literature Review

2.1 Market

The first commercial RISC-V processor, the Freedom E310 was released by SiFive back in 2016 [3]. Freedom E310 is a 32-bit 320 MHz microcontroller designed for small and low power applications. Since then, the market for RISC-V cores has expanded leading to more powerful multi cores out of order processors capable of performing high demand tasks such as deep learning or running servers and data centers. The RISC-V entered the SoC FPGA market in 2020 when microchip announced PolarFire SoC, the first-ever hard-core RISC-V processor inside an FPGA [3]. The scene for RISC-V cores has become more and more competitive as big companies as well as new startups are jumping on the open ISA train. Many companies have established themselves as the go-to for RISC-V cores that span a wide range of applications and we will discuss some of these companies.

Codasip

Codasip is a RISC-V processor IP provider. They provide a wide range of RISC-V standard processors for a variety of applications and requirements. Their lineup consists of Small and Low power processors, for light embedded applications, more powerful high-performance cores suitable for more demanding embedded applications, and single or multi core application processors capable of running Linux systems[6]. Their products are divided in 4 series with each one spanning a performance region as shown in the figure 1.

Figure 2.1: Codasip Processors Series [6]

Codasip provides 11 processor configuration to suit the customers applications from the small and low power L10 to A70XP-MP that has 4 cores and supports floating Point operations and Atomic instructions[6]. Aside from the wide variety of processors specs, Codasip provides another layer of customization through their unique EDA toolset, Codasip Studio[6]. Codasip Studio is a fast and easy highly

automated tool to modify RISC-V cores. Codasip processors can be deployed as is or can be further customized to create a unique processor that perfectly fits the customer application. Codasip Studio enables the user to change the microarchitecture features of the design to create custom instructions or add standard RISC-V extensions. Codasip Studio also allows the user to create their customized HDK and SDK as well as to verify the edited core.

Figure 2.2: Codasip Processors [6]

SiFive

SiFive is one of the pioneers of RISC-V instruction set architecture processors. Its products include a wide variety of cores, SoCs, Ips and development boards. SiFive IP cores span a wide range of applications from low-power small area embedded microcontrollers to high-performance multicore heterogeneous application processors [7]. SiFive lineup consists of 5 series and 3 types of cores E, S and U. E cores are low power 32- bit cores suitable for IoT and low power embedded applications. S cores are 64-bit cores that are used for more demanding applications such as machine learning[7]. U cores are 64-bit application cores used in datacenters and are capable of running Linux. SiFive also provides custom cores through their SiFive Core Designer. SiFive Core Designer enables the customer to use one of the available processors configured for common use cases as a starting point and modify and edit the core to suit his specific application[7].

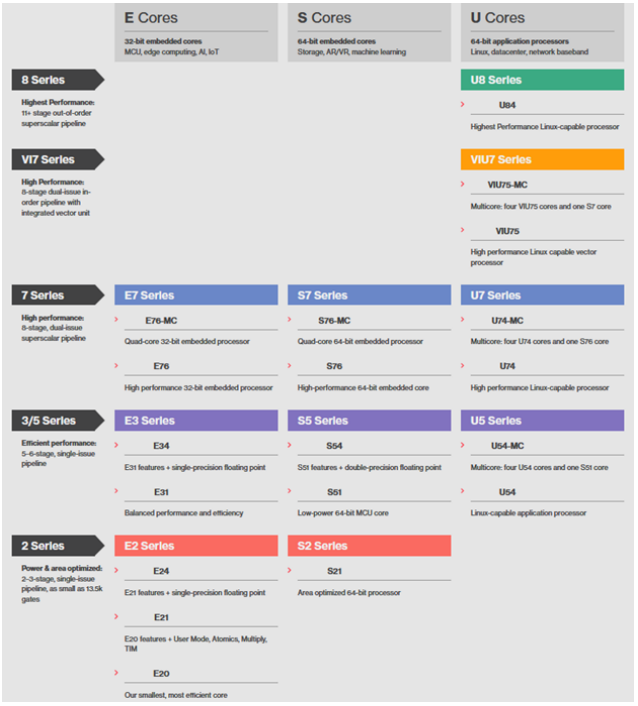


Figure 2.3: SiFive processors [7]

Andes Technology

Andes Technology is a Taiwanese embedded CPU cores supplier and one of the founding members of RISC-V International Association. Andes cores uses V5 architecture which is a new family of AsdeStar that is fully compliant with RISC-V[8]. It adds new features to RISC-V such as Andes Performance Extension which can speed up common program sequences such as memory access and branches. Another advantage of V5, is code size compaction through CoDense, stack overflow and underflow detection through StackSafe and frequency and power scaling through PowerBrake[8]. Andes also provides a software development environment that offers RISC-V compilers and a comprehensive GUI-based development environment. Moreover, Andes provides hardware development environment solutions consisting of an FPGA development board, Arduino-compatible Corvette-F1 board and an ICE debugger[8]. Andes RISC-V processors are divided into three series, A-series, A/D-series and V-series.

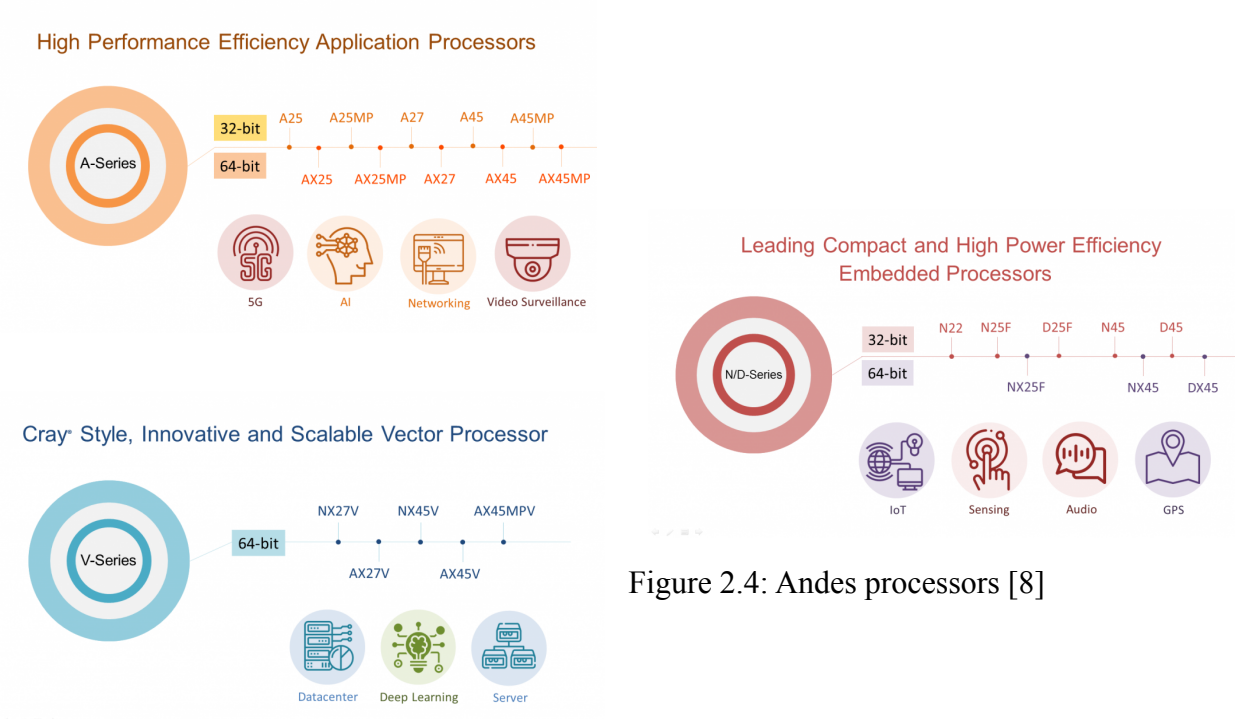


Figure 2.4: Andes processors [8]

2.2 Literature Review

In literature, the implementation of ASIC RISC-V processors has been a hot topic over the past decade since its design at the University of California, Berkeley. Researchers have been trying to push the processor's limits and are continuously trying to optimize it for various applications and enhance its performance. S. Kumar in [9] main goal was power optimization. He tried utilizing different techniques to reduce power consumption. He worked on three different designs, a design without a low power approach, a design with clock gating and clock tree optimization, and a design with Multi-Vth, Multi-Supply voltage, and power shut-off along with the previous techniques. His third design produced the best results with a total power consumption of 2.415 W and a total area of 2374572.1 mm2. He used Encounter RTL Compiler for the synthesis stage to generate the gate-level netlist, Cadence Encounter Digital Implementation System for the back-end flow. In [10] Spanish and Mexican academic institutions designed and fabricated the preDRAK processor, a RISC-V general-purpose processor capable of booting Linux systems. They used the CMOS 65nm technology in their ASIC implementation, they used Cadence Genus for the synthesis. As for the physical design and back-end, they used Cadence's Innovus along with Mentor Calibre for DRC. The estimated power consumption after place and route is 344 mW and a total chip area is 3.57 mm2. Multiple literature works tried to compare the ASIC and FPGA implementation of the processor. In [11], the main goal was to design RISC-V accelerators for post-quantum cryptography. The design was implemented on both FPGA and ASIC. The ASIC implementation only got through till the synthesis stage. Compared to the PULPino origins design, the cell area and combinatorial logic were increased by 64, 522 mm2. and 9, 969 mm2

respectively. However, the increase didn't have much impact on the total area as the memory area, which was the highest contribution, remained the same. As for the power consumption, they used The Switching Activity Interchange Format (SAIF) for accurate dynamic power calculations. They used Cadence's Incisive Enterprise Simulator and Cadence power analyzer Joules. The design had higher total power consumption due to the increased area, but the lower energy consumption overall. In [12] they fabricated a 64-bit dual-core RISC-V processor with custom vector accelerators using 45 nm technology. The dual-core RISC-V processor achieves a maximum clock frequency of 1.3 GHz at 1.2 V and peak energy efficiency of 16.7 double-precision GFLOPS/W at 0.65 V with an area of 3 mm2. They used Synopsys design compiler and IC compiler for the physical design flow. They used Multi-Vth libraries and SRAM-based memory cells. They used a bottom-up approach reducing the tool's runtime. They also used PrimeTime and Formality for the final signoff steps of static timing analysis and formal verification. As for the power consumption, it ranges from 300 mW to 430 mW at 1 V depending on the processor activity, and scales from 40 mW at 0.65 V up to 960 mW at 1.2 V.

Chapter 3

ASIC Physical Design Flow

3.1 Introduction

The ASIC physical design flow starts with the synthesis of the HDL into the standard target technology, followed by verification of the synthesized design checking that the mapped gates are working as they were intended to. The next stage is to perform floorplanning and determine the design and chip area and an initial estimate for the cells' locations. After floorplanning, the power network is synthesized and power rails are added to deliver the required power to the design. Then, cells and macros are placed all over the chip's area trying to have a good congestion control and meeting the timing requirements in the process. After the placement stage, the design is routed and cells are connected to each other the way it's described in the netlist. The design then goes through static timing analysis checking that timing is met. As a final step, the design is checked for clean LVS & DRC errors before the GDSII file is outputted [13].

3.2 Synthesis

Synthesis is the process of transforming the design's RTL into a synthesized gate-level netlist. The synthesis is performed in three steps. First, the translation where the tool transforms the RTL into boolean equations. Second, logic optimization which minimizes the boolean expression. The third and final step is mapping, where the optimized logic is mapped into the target library generating a gate-level netlist. The synthesis also inserts clock gating cells and DFT logic into the design if specified [14].

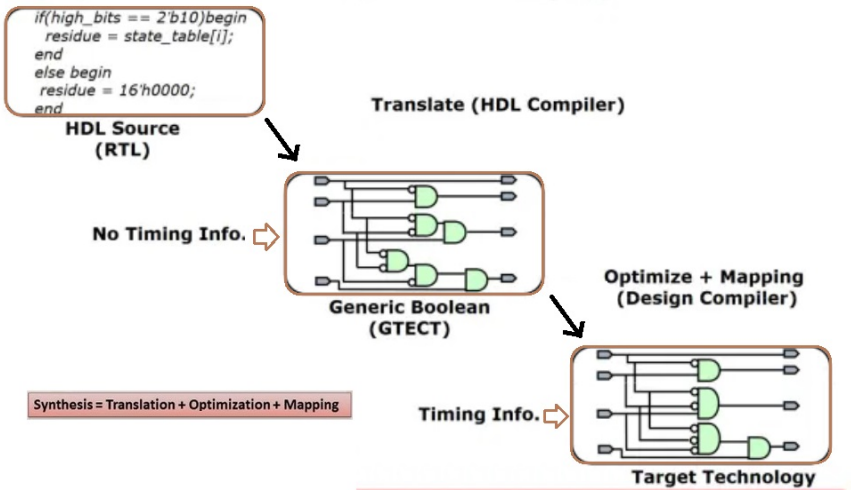


Figure 3.1: Synthesis Flow [14]

Inputs

RTL Files [14]

They contain the design abstraction modeling the synchronous flow of signals described in HDL such as VHDL, Verilog, SystemVerilog.

Libraries & Files [14]

- Physical Library
    - It contains standard cells, macros, and information regarding their shapes, sizes, orientation, and layout geometries.

- Timing Library
    - It contains the standard cells and macros timing information such as gate delays and delay models.

- Technology Files
    - They contain the technology design rule constraints, physical, and electrical parameters, resistance and capacitance values of metal layers and vias.

- TLU+ Files

- They contain the RC coefficients for accurate RC extraction models and net delay calculations. They take the effect of width, spacing, density, and temperature into consideration.

- Milkyway Library
    - It's a library storing all the design files starting from synthesis till the design's signoff. It holds the physical information such as cell placements and routing directions along with the width and height dimensions.

Design constraints [14]

Constraints are rules and instructions the tool has to follow during synthesis translation, optimization and mapping to ensure that the design will meet its specific specifications and fabrication requirements. Constraints can be classified into design rule constraints and optimization constraints.

- Design Rule Constraints
    - They are the constraints dictated by the fabrication foundry. Those constraints entail process, voltage, and temperature operating conditions. Wire-load model for resistance and capacitance estimation. Maximum capacitance, transition time, and fanout along with clock uncertainty. System interface constraints which include constraints imposed by the outside logic driving and receiving data from the design such as input and output delays. By default, the design rule

constraints are prioritized over the optimization constraints by the synthesis tool.

- Optimization Constraints
    - They are the constraints specified by the designer for his design to meet its goals. The designer can either choose those constraints for example to tighten further the capacitive and fanout constraints or to optimize the design by describing the design area, timing, and power goals. The designer can set clock definitions, timing exceptions, multi-voltage constraints, and specific types of standard cells to perform the logic mapping. Since design rule constraints have precedence, the tool can violate area and delay constraints to meet the ones with higher priority.

Outputs [14]

Netlist

It's a description of the design elements and their connectivity. The netlist contains combinational and sequential instances of the standard cells and macros with their ports and interconnection details. The netlist acts as the input for the PnR tool performing floorplanning, placement, and routing.

Constrain File

It contains the design constraints acting as an input for the PnR tool.

Reports

After synthesis is complete, different reports can be generated to report the design's area, power, timing, constraints, and quality of results.

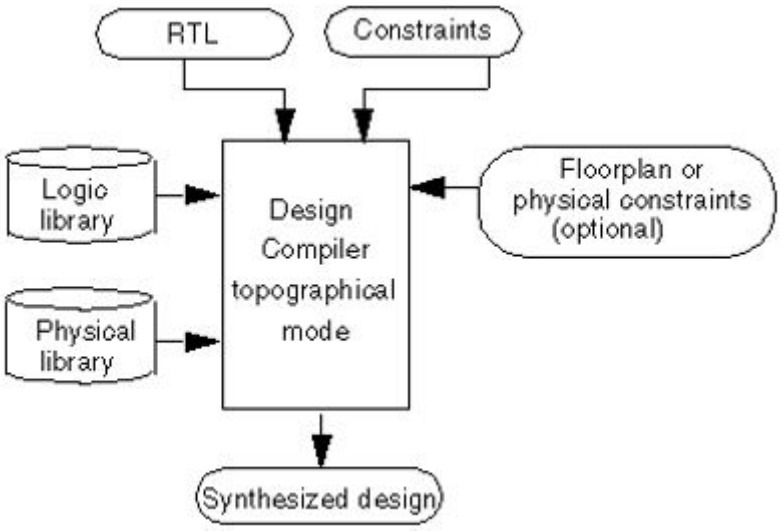


Figure 3.2: Design Compiler Inputs & Outputs [14]

3.3 Floorplanning

Introduction

Floorplanning is the first step of place and route (PnR) flow after the netlist is generated where the designer specifies roughly where each part of his design will be located on the chip die. The PnR stage aims at having a verified layout that can be fabricated. Chip planning deals with large modules such as caches, embedded memories, and intellectual property (IP) cores that have known areas, fixed or changeable shapes, and possibly fixed locations [13]. When modules are not clearly specified, chip planning relies on netlist partitioning which is prior to floorplanning. There are two styles of implementation in floorplanning. For small to medium ASIC designs, the flat approach has better area usage as there is no need to reserve an area around each block for power connections. However, for very large designs it is better to follow a hierarchical approach in order to optimize each sub-block independently [14].

Before the floorplanning stage, the design is split into individual circuit modules. A module becomes a rectangular block after it is assigned dimensions or a shape. These blocks can be either hard or soft. The dimensions and areas of hard blocks are fixed. For a soft block, the area is fixed but the aspect ratio can be changed [15]. The entire arrangement of blocks, including their positions, is called a floorplan. The floorplanning stage ensures that every chip module is assigned a shape and a location, so as to facilitate gate placement it also makes sure that every pin that has an external connection is assigned a location so that internal and external nets can be routed. The

last step of floorplanning is the power network which ensures that every cell in the design is connected to power and ground. The following flowchart (Figure 3.3) shows the steps of floorplanning.

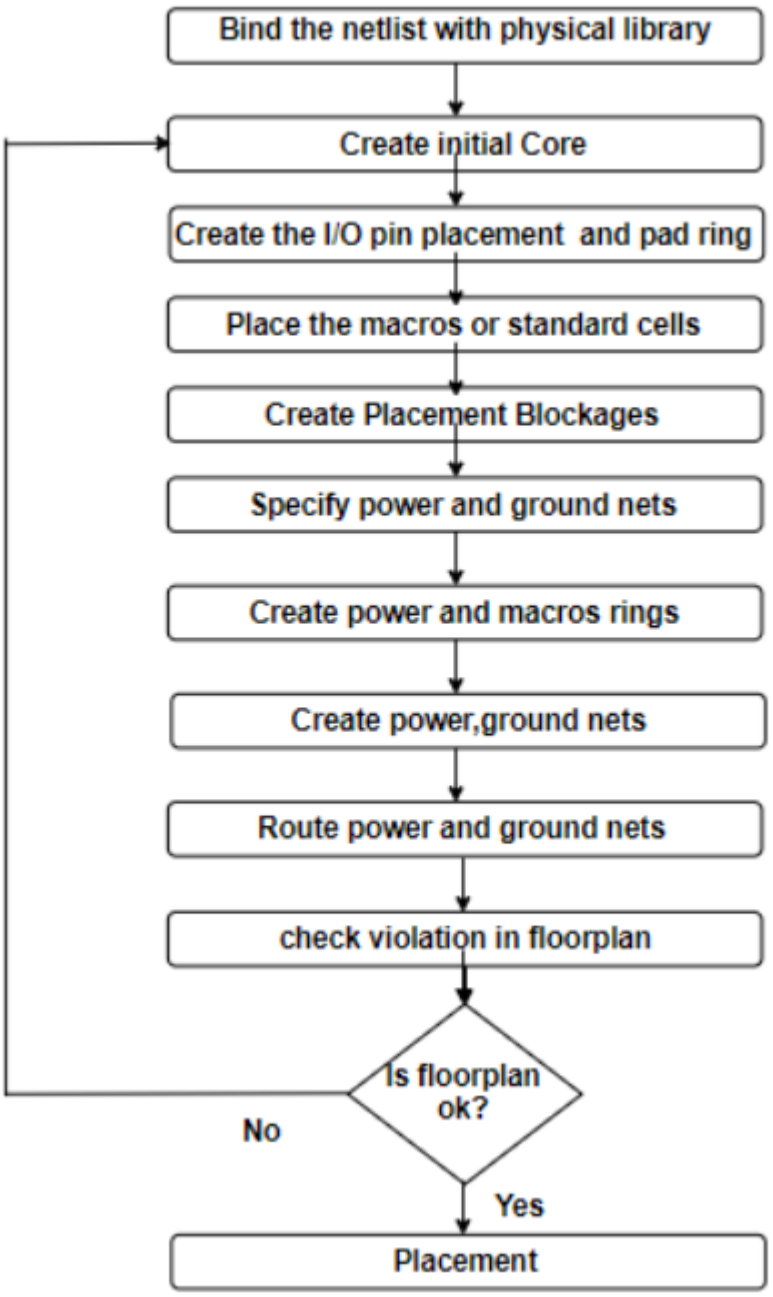


Figure 3.3: Floorplan Flowchart [14]

Floorplan

The objective of the floorplan stage is to optimize both the locations and the aspect ratios of the individual blocks. It also tries to minimize the area and shape of the global bounding box and keep its aspect ratio close to the given target value. It is also important to optimize the total wire length and the signal delays. It is also important to make sure that routing is possible and the IR drop will be within an acceptable range [13].

The inputs to the floorplanning stage include [17]:

1. The netlist defines all design elements and connections that are generated from the synthesis stage. It is a Verilog or VHDL format.
2. The library files (.lib or .lef) and the TLU+ file are technology-specific files that determine the timing information of each of the standard cells in the library at different corners in addition to the metal layers and the DRC of them. Also, the TLU+ file is used to specify the RC model of the corresponding metal layers and vias.
3. The constraints file that outputs from the synthesis in SDC format contains the timing constraints and other constraints.
4. It is optional to use a DEF file format that defines the floorplan in case it is made by third-party software or in case it is made by the synthesis tool in case of topographical flow.

The floorplan has to be constrained in order to output the optimum results that meet the design requirements. These constraints include the chip and core aspect ratio, chip

utilization which is the percentage of the area of the core that is used by standard cells and macros, row configuration and cell orientation, core to IO pad spacing, and blockage management. Standard cell rows can have various configurations by having a gap between each cell row or flipping every two rows together, etc [15]. The spacing between IO pads or core is used for inputs and outputs placement or power rings. The following figure (Figure 3.4) shows the defined floorplan.

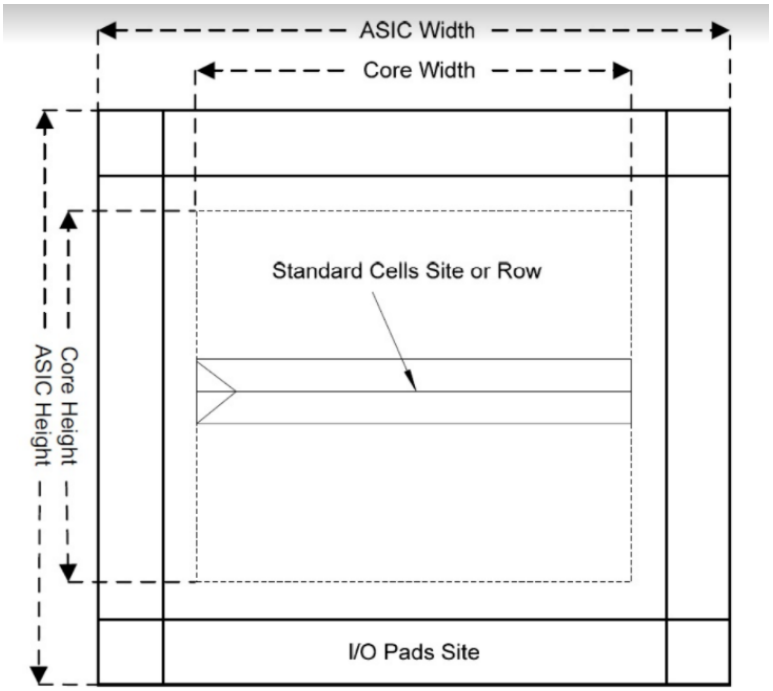


Figure 3.4: Floorplan Definition [13]

IO Pads Placement

Pads are circuits that translate the signal levels used in the ASIC core to the signal levels used outside the ASIC. Additionally, the pads circuits clamp signals to the power and ground rails to limit the voltage at the external connection to the ASIC pads. There are several types of pads in the design such as signal IO pads, power pads, corner pads, and filler cells. It is important to place the pads in a way such that the connection length between the pad and the target component of the pad is minimized [14]. It is a good approach to place macrocells around chip periphery as shown on Figure 3.5 so that chip area will be clustered. Also, the connection of fixed IO ports should be taken into consideration.

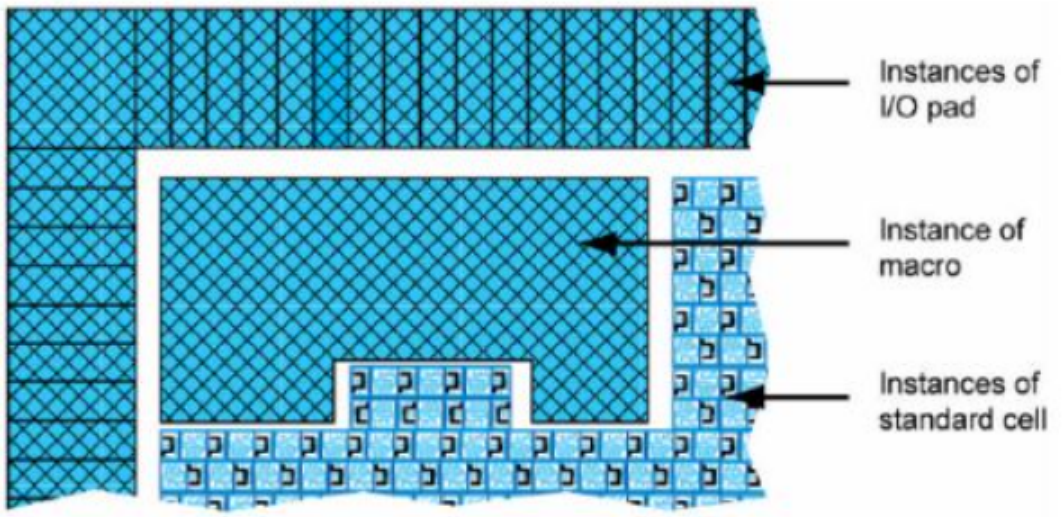


Figure 3.5: Macrocell Placement [14]

3.4 Power Network

The next step is to build the power grid. The objective of the power network is to supply power to all parts of the design. There are several levels of power distribution. Trunks are used to connect between power pads and power rings. Power rings are VDD and VSS rings that are formed around the core and macrocells. They are used to supply power and are usually made of thick metals in order to reduce the power dissipation, guarantee good connectivity and keep the thin lower metal layers for signal routing. The top two metal layers are usually used because each metal layer has one preferred direction that most of its routes are routed in that direction [13].

After inserting the power rings, the power stripes are inserted in order to distribute power from the design to the rest of the core area. Power stripes are vertical metal wires that are placed over the core area and are connected to the power rings. After creating the stripes, power rails are inserted which are used to supply power to the standard cells. The connection between the power rails and the standard cells is done through abutment, where standard cells have their power pins at the top and bottom of the cell, and power rails are placed at the top and bottom of each cell row, so when those cells are placed on the rows they automatically get connected to the power grid [15]. Power rails are connected to both the ring and the stripes to ensure even power distribution along the rail length. Metal vias are inserted between the metal layers, rings, grid, and rails. Figure 3.6 shows the components of the power network.

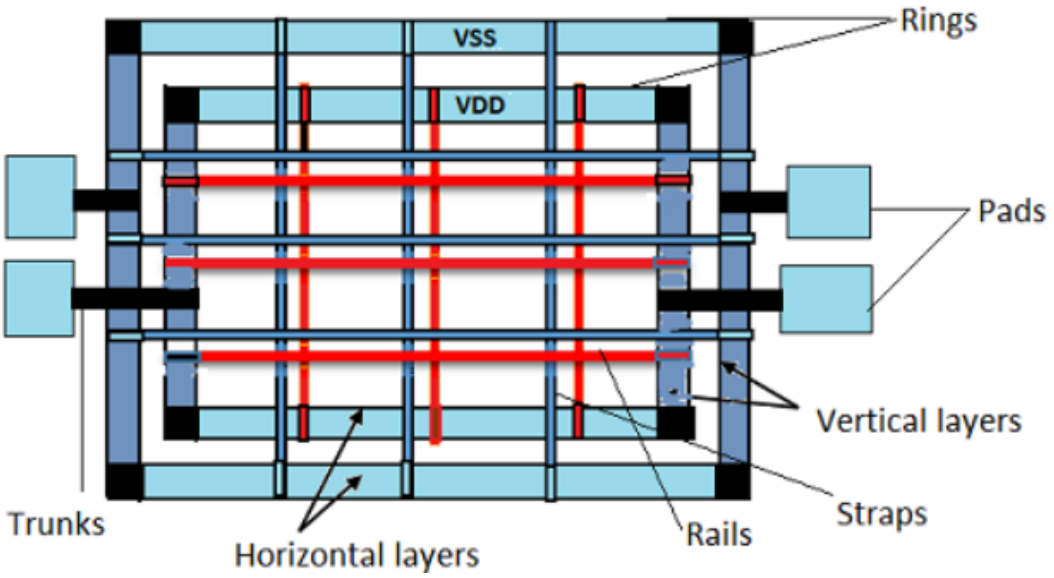


Figure 3.6: Power Network [14]

The ideal power distribution aims at maintaining a stable voltage to each cell of the design with little noise and IR drop. Other important factors are to avoid wear out due to self-heat or electromigration. It is also important to consume little chip area and routing and make the layout out easy to fabricate. A UPF file may be inserted before the power network step in order to define the power intent of the design [15].

There are several methods to improve the noise immunity and keep the voltage level stable within the core area. These methods include using decoupling capacitors to separate the routing of power signals. Other methods may use level shifters, retention registers, or power switches [14].

3.5 Placement

Introduction

Placement is one of the most critical steps in ASIC flow. Placement seeks to determine the locations of standard cells or logic elements within each block inside the rows that were created during placement while addressing optimization objectives. The goal of placement is to minimize the total wirelength and make the design routable while giving a good performance and minimizing the total power and heat dissipation. It is also important to minimize the congestion of cells and pins and minimize the cell density. Placement has three main steps: global placement, detailed placement and legalization. The inputs of placement are the same as the floorplan in addition to adding the DEF file which includes the floorplan and power network information [17].

Global Placement

When the floorplan is first created, standard cells are in a floating state. This means that they are placed arbitrarily in the ASIC core and have not been assigned to a fixed location within the standard cell rows. At this time one can partition the standard cell area and assign a group of cells to these partitions, or simply group a set of standard cells. Global placement often neglects specific shapes and sizes of placeable objects and does not attempt to align their locations with valid grid rows and columns. Some

overlaps are allowed between placed objects, as the emphasis is on judicious global positioning and overall density distribution. The global placer engine uses the information of the design hierarchy (if available) to place related standard cells close to each other to reduce both timing and congestion as much as possible. Usually the global placement process is done in iterations, the quality of the placement improves proportionally with the increase of this number of iterations [13].

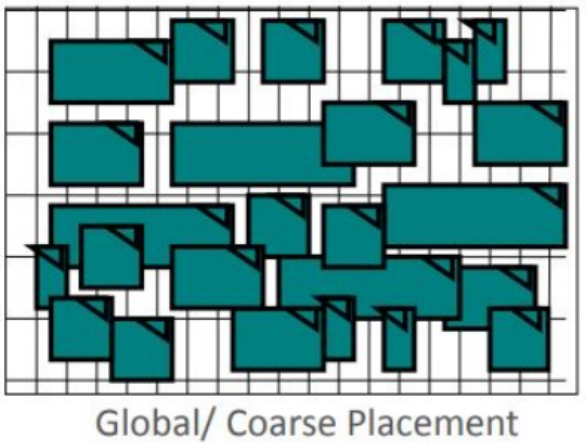


Figure 3.7: Global / Coarse Placement [14]

Detailed Placement & Legalization

Legalization is performed before or during detailed placement. It seeks to align placeable objects with rows and columns, make sure it does not violate any DRC rule, and remove overlap while trying to minimize displacements from global placement locations as well as impacts on interconnect length and circuit delay. Detailed

placement incrementally improves the location of each standard cell by local operations (e.g., swapping two objects) or shifting several objects in a row to create room for another object. Detail placement algorithms are executed to refine placement based on congestion, timing, and/or power requirements [13].

Figure 3.8: Detailed / Legalized Placement [14]

3.6 CTS

Clock Tree Synthesis is a process with which we aim that the clock gets distributed evenly to all sequential elements in the design[11].

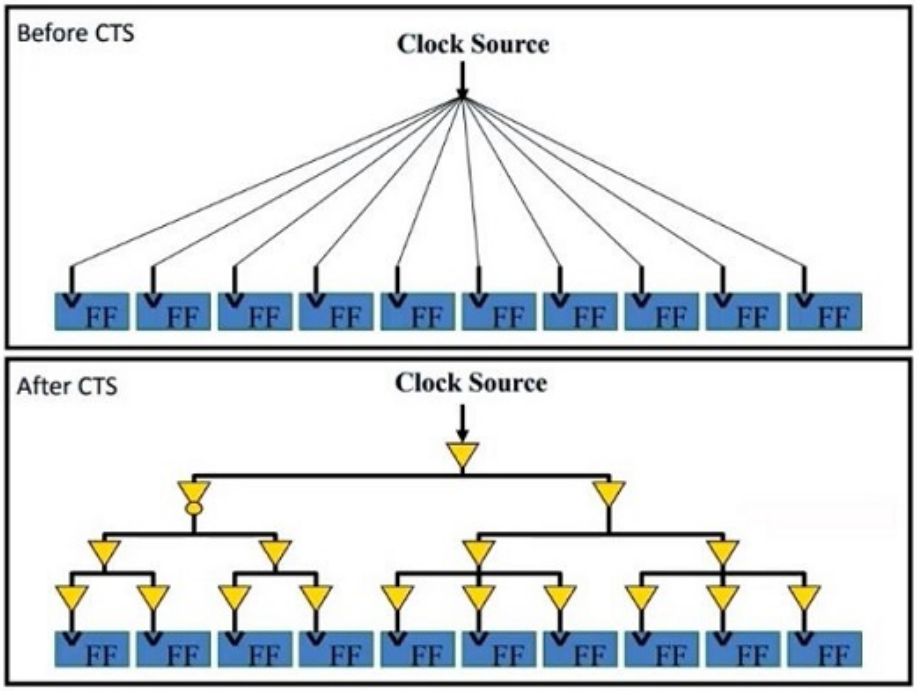


Figure 3.9: Clock Tree[13]

The main target of CTS is to minimize the skew and latency and also to meet the design constraints such as Maximum transition, Maximum load capacitance and Maximum Fanout. Clock tree can be built by clock tree inverters so as to maintain the exact transition (duty cycle) and clock tree balancing is done by clock tree buffers.

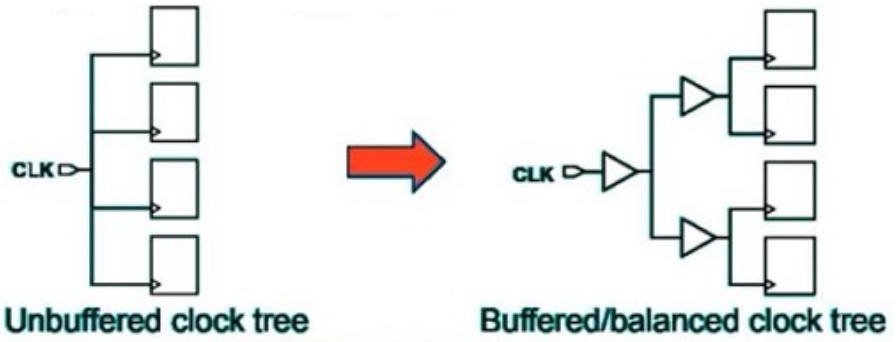


Figure 3.10: Difference Between Buffered and Unbuffered[13]

Checklist before CTS:

- Placement is completed and optimized.
- Power & Ground (PG) nets are pre-routed.
- Estimated congestion – Acceptable.
- Estimated Max trans/Cap - No violations.
- High Fan-out Nets are synthesized with buffers (clocks are not buffered
- still).

Checklist after CTS:

- Skew report.
- Clock tree report.
- Timing reports for setup and hold.
- Power and area report

The main problems with CTS is that the clock is power hungry, also the clock can have electromigration on one of its nets. The crosstalk is another problem too. We should be aware of the noise as the clock is considered a strong aggressor and it might need shielding.

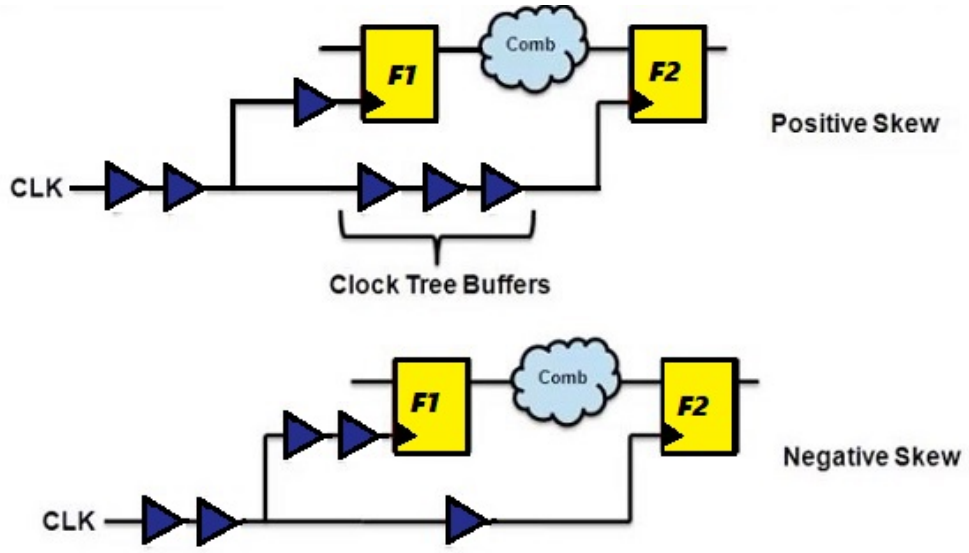


Figure 3.11: Clock Skew[13]

The skew of the clock is the main factor we aim to solve by CTS. We can have positive or negative skew whether the clock is late or early. It is important to remember that we have Local skew which is the difference between the clock arrival of two consecutive pins and Global Skew which is the max insertion delay and min insertion delay of any flip flop in the design.

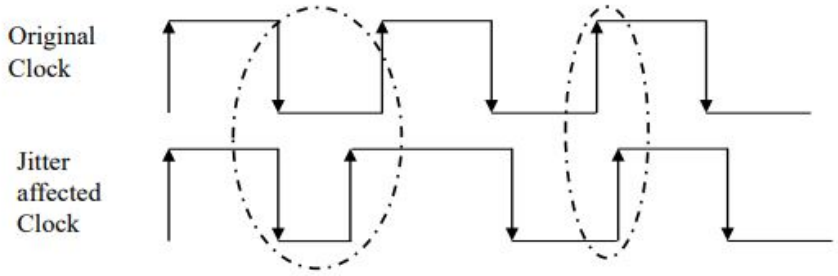


Figure 3.12: Clock Jitter[11]

Clock Jitter can also be a problem in the design. Mainly, It is the difference between clock periods and it is either random jitter or deterministic jitter and It is affected by temperature, crosstalk and voltage variations[10].

3.7 Routing

Routing is the most important and longest stage in which we convert the logical routes to physical ones. The main target of routing is to assign exact paths between the interconnections of the standard cells. Applying the constraints such as transition, skew and capacitance makes it very difficult for the tool to achieve the target outcome and that's why it takes a lot of time.

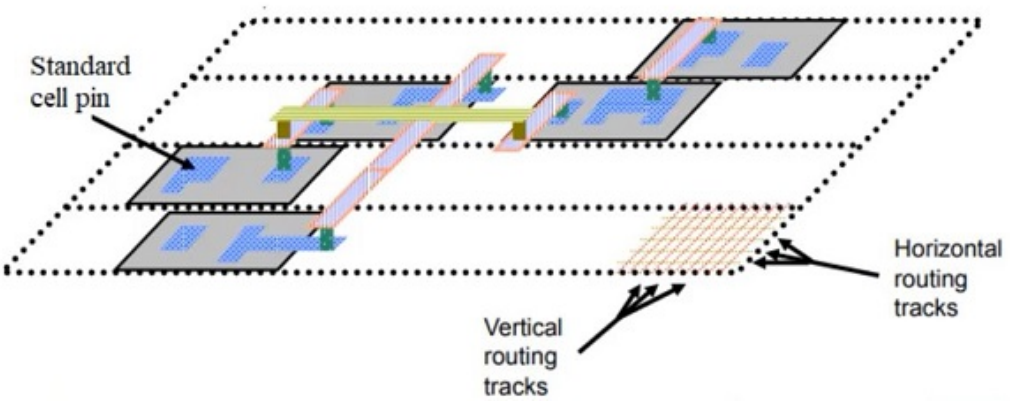


Figure 3.13: Routing[13]

Routing starts with global routing in which the design is logically routed. In this stage, The tool aims to find the shortest path without caring about the DRC errors. It also avoids the congested areas.

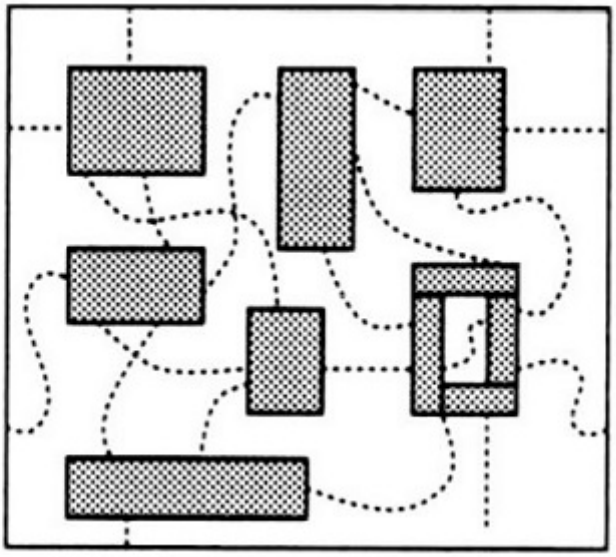


Figure 3.14: Global routing[13]

The second stage is track assignment. After the design is logically routed, These routes are assigned for each track which has a specific layer and geometry. It does not care for DRC rules too however It aims to minimize vias and also be timing aware in the routing.

Tha last stage is detailed routing in which the nets are physically routed in. It performs using the DRC rules.

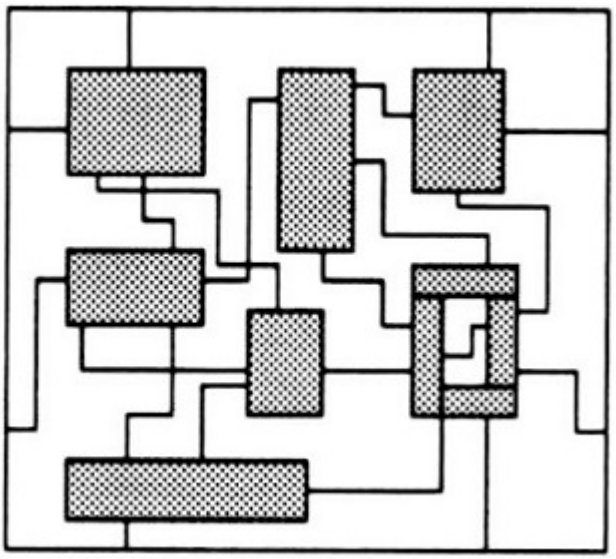


Figure 3.15: Detailed routing[13]

Now the design is finally routed, It is important to check if the physical design works correctly or not and this happens through the checks below.

Checklist before routing:

- All cells should be placed legally
- Clock Tree structure should be defined
- NDRs

Checklist after routing:

- LVS
- DRC
- Other design constraints such as Congestion and Timing

After that, We will be ready to output the GDS file and then sign off to the foundry.

3.8 Static Timing Analysis

3.8.1 Introduction

Static timing analysis or STA is a method that is used to verify that the design meets the timing requirements. Unlike other circuit simulations such as spice, STA is not concerned with simulating the functionality of the design under different test cases; its sole focus is the timing which makes it faster[13]. STA also gives better timing predictions as it checks the worst-case timing for all possible logic passes in the design. The static part of STA arises from the fact that the analysis of the design is done statically without applying any inputs or any change of data[13]. This is opposed to dynamic timing analysis, which is used to verify the functionality of the design by changing its inputs and observing its outputs.

Design constraints such as timing, area and power are the base on which the digital design is formed. While area and power can be of most importance in many designs, they are not a threat to the chip operation. Chips intended for smaller areas or less power consumption can still run just fine and perform the desired functionality. On the other hand, if the chip timing is not met it cannot function correctly at the desired clock. Here comes the role of STA which is to make sure that the data is present at the inputs of every synchronous device at the clock edge, under all possible conditions.

3.8.2 Importance of STA

The importance of STA is present in each stage of the ASIC backend flow. In synthesis, the synthesis tools perform static timing analysis to determine which cells are chosen to implement the RTL[14]. In the mapping stage of the synthesis, logic gates are translated to actual physical gates from the provided technology library[14]. Since the technology library has a wide selection of cells with different delays and sizes the decision to choose one over the other to implement the logic might come down to the timing constraint. In Pnr, STA is of most importance as it will determine where the cells are placed, how the cells are routed and how critical nets, such as the clock, are routed[14]. In the placement stage, if the STA shows that the timing of a certain path is not met the placement tool will try to place the cells in that path closer to each other thus reducing the length and the delay of the wire connecting them. If the timing is violated in the routing stage the tool will give higher priority to routing the path that has timing problems in order to guarantee that it will be solved. The same happens in the clock tree synthesis where the tool constantly checks to make sure that the skew of the added clock tree will not cause timing problems. Despite the fact that these tools constantly do STA, using a specialized sign-off timing analysis tool is essential to ensure that the design will work at the intended clock speed.

3.8.3 How it's done

To understand how STA works we should first introduce the timing-sensitive digital components. Flip-Flops (FF) and latches are the storage data elements of all digital designs. Flip-flops are edge-triggered devices, meaning that their operation is dependent on the existence of a synchronized signal we call the clock. When the clock changes from 0 to 1 (positive edge) the FF stores the value at its input D and makes that value available at its output Q.

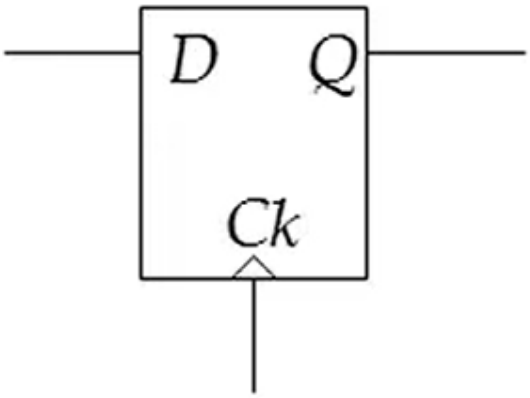


Figure 3.16: D Flip-Flop[13]

To ensure the correct functionality of the FF some timing constraints have to be considered. First is the setup-time, the data cannot change any later than this time before the clock edge. Hold-time, data cannot change during this time after the clock edge. Time to Q, which is the time required for the data captured at D to be present at Q. One other factor that affects the timing of FF is related to the clock which is the

skew. Skew is the phenomenon in which the same clock signal arrives at different times for different circuit components. While the skew is pounded, it is not possible to know if it is positive or negative at any particular point.

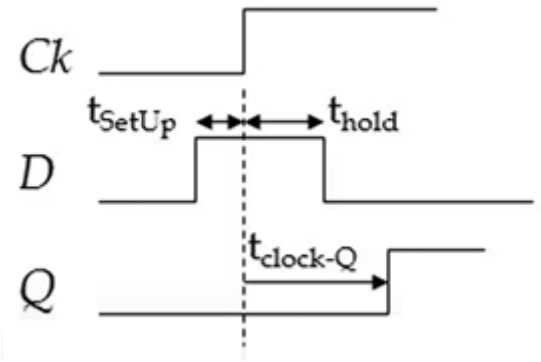


Figure 3.17: D Flip-Flop Timing[14]

For circuit shown in figure 3.18, in order to ensure its correct functionality, change of data at the output of FF1, as a result of the clock edge, has to propagate to the input of FF2 before the next clock edge.

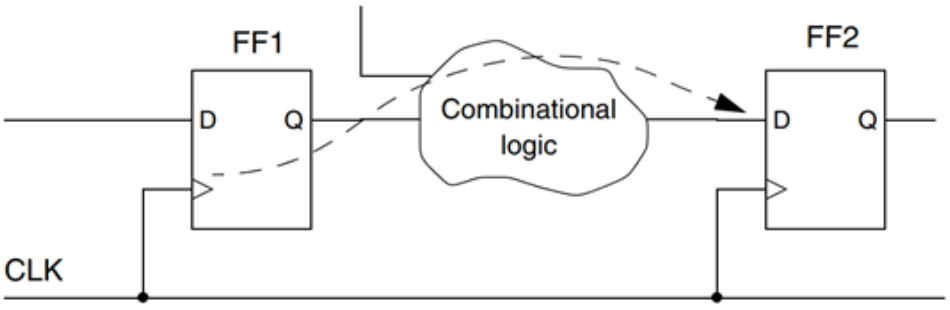


Figure 3.18: Digital Circuit[14]

3.8.4 Setup Check

The time diagram shown in figure 3.19 represents the timing requirement of the circuit. The data goes through the combinational logic with some delay t-logic and the output of the combinational logic is at the input of the second flip-flop, FF2. The change in value at FF2. D must occur before the arrival of the clock edge arriving at FF2, by at least an amount equal to the setup time requirement t-setup for the flip-flop. The difference between the arrival of the clock edge at FF1 and its arrival at FF2 is donated by t-skew. The amount of time by which the timing constraint is met is called the slack of the timing.

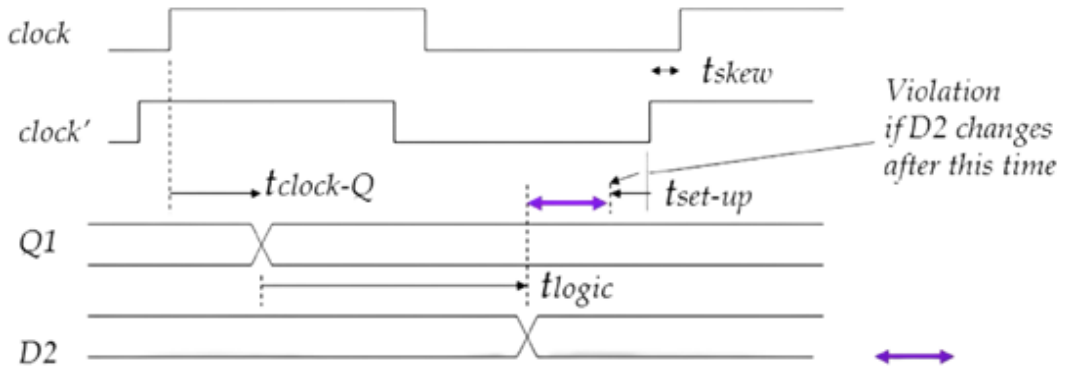


Figure 3.19: Setup Timing Check[14]

The following equation represents the clock required for the design not to have setup time violations. If the clock is larger than the right-hand side, we have a positive slack, and this means that the design can run at a higher speed. If the slack is negative the timing cannot be met and modification to the design or clock speed has to be done for the design to function correctly.

$$t_{clock} \geq t_{clock-Q-\max} + t_{\log ic-\max} + t_{set-up} + t_{skew}$$

Figure 3.20: Setup Time Equation[14]

3.8.5 Hold Check

The timing diagram in figure #, shows the hold violation check. This hold check makes sure that the data will remain the same during and after the arrival of the clock edge by at least the hold time of the FF in order to be captured correctly. The newly captured data at FF1 might lead to a change in the logic output and If the logic delay of the circuit is too short this change might happen during the capturing of data at the second FF. This will cause the FF to have an unpredictable state.

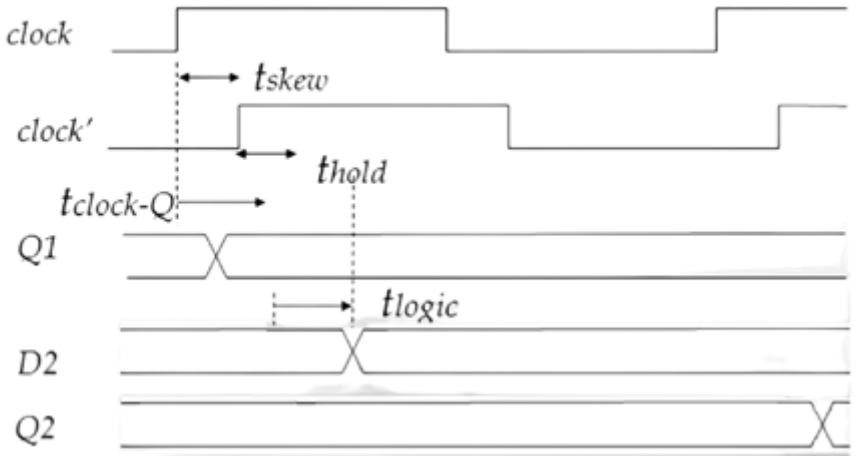


Figure 3.21: Hold Timing Equation[14]

The following equation shows the condition that ensures that the digital path doesn't have a hold violation.

$$t_{hold} + t_{skew} \leq t_{clock-Q-\min} + t_{\log ic-\min}$$

Figure 3.22: Hold Time Equation[14]

It should be noted that the clock speed is not a parameter in this equation. This is due to the fact that the hold violation is caused by the delay of the logic cells between the FFs being too small to guarantee the data stability during the FF data capture time. The cell delay and the hold time of the FF are determined by the technology library and are not a function of the clock speed.

The fact that the hold violation is independent of the clock speed makes it much more dangerous than the setup violation. If the design has been fabricated with a setup violation it will not work at the desired speed but reducing the clock will make the design function correctly. On the other hand, if a fabricated design has a hold violation, it will not work correctly under any given clock.

3.8.6 Timing report

To check whether the design meets the timing we use the report_timing command with the -delay_type min or max for setup and hold checks. The resulting timing report is similar to the following.

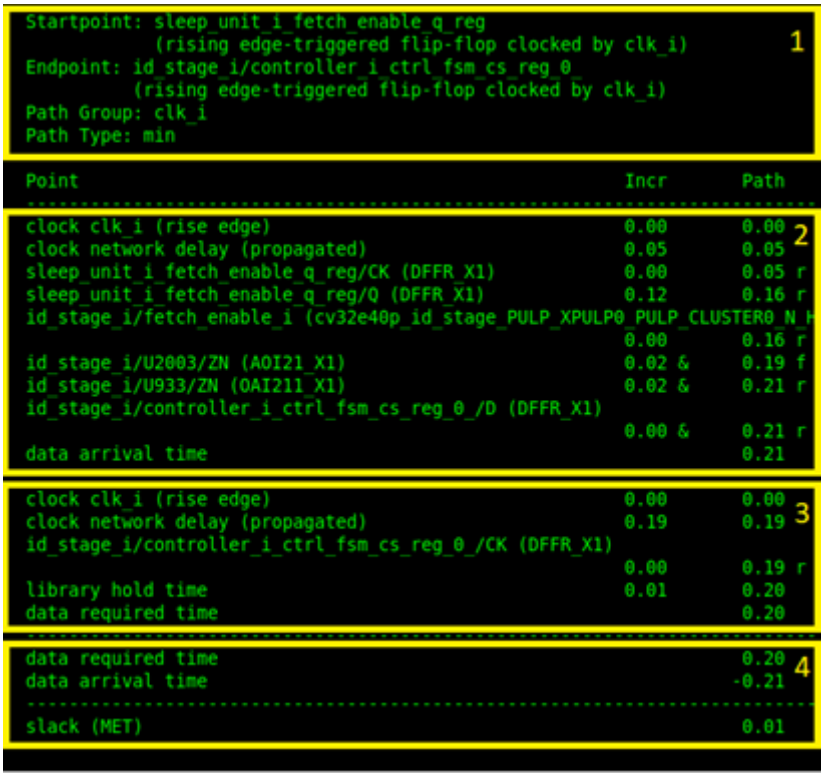


Figure 3.23: Timing Report

1- Header

The header contains the start point and the endpoint of the timing path. These points can be either a register, an input pin or an output pin. Subsequently, there exist four different types of timing paths as shown in the figure.

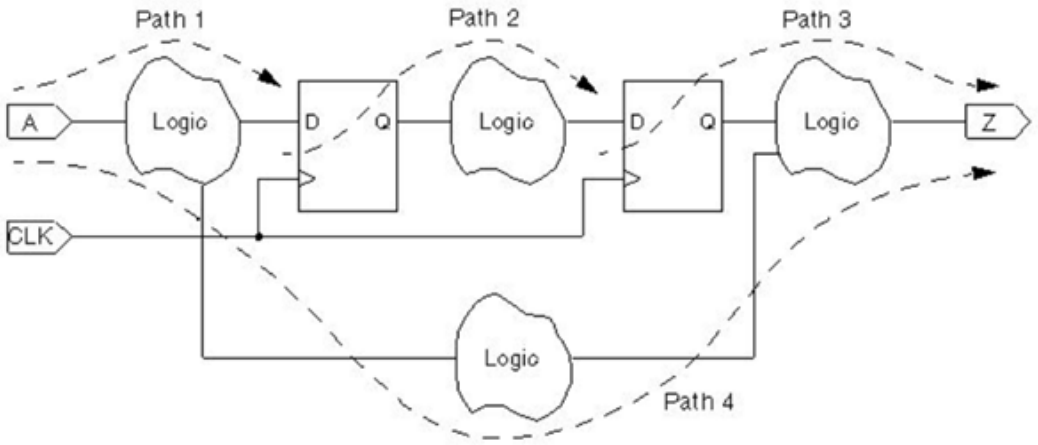


Figure 3.24: Types of Timing Paths[14]

The header also contains information about path groups and the type of the path, max for setup paths and min for hold paths.

2- Data arrival section

This section contains the delays that affect the data in the path which add up to the time at which the data arrive.

3- Data required section

In this section the details of the required time for the path in order to avoid timing violation.

4- Slack

The slack is given by the difference between the arrival time and the required time. If the slack is positive the timing is met and not timing violations are possible.

Chapter 4

Project Implementation

4.1 Flat, Hirerachiel and Topographical flows.

There are three different approaches to the ASIC physical implementation, flat, hierarchical and topographical. The flat design flow is based on dissolving the boundaries between the different modules of the design. This allows for more logic optimization that wasn't previously possible at the boundaries. This flow results in the best-optimized logic for area, power and speed. This approach, however, suffers from long runtime as the tool's optimization algorithms run longer as space or optimization expands. This approach is best suited for smaller ASIC designs as it can give optimal results.

The hierarchical flow tries to do the opposite of the flat by keeping the hierarchy of the design intact. This allows the tools to optimize the logic within the modules but not on the boundaries. This flow is suitable for large designs where the sub-systems are designed and optimized individually. The following figure shows the difference between these two approaches.

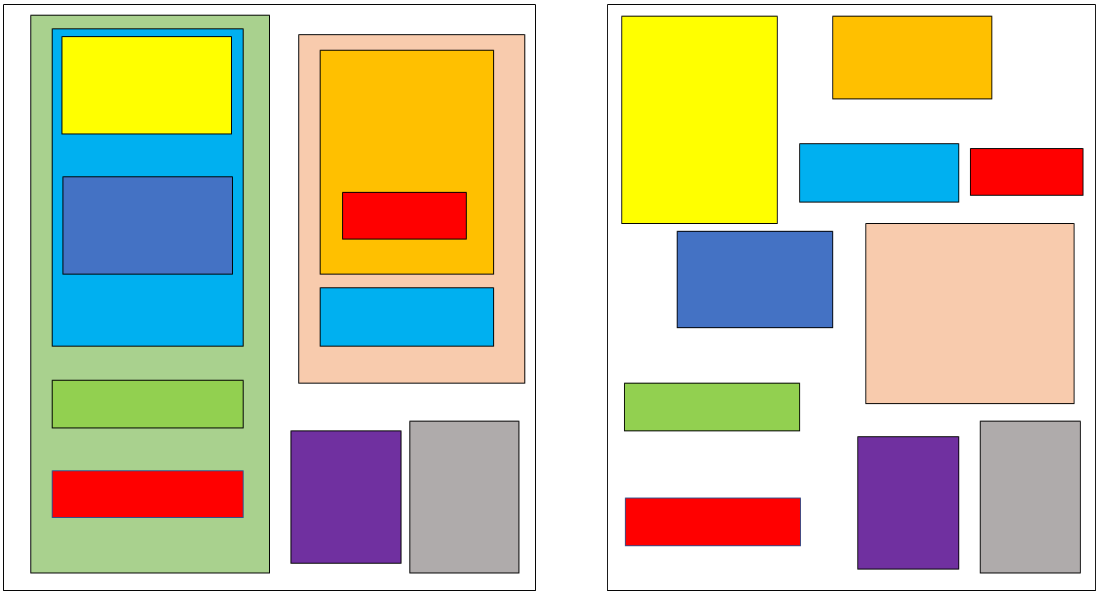


Figure 4.1 : Hierarchical VS Flat

In order to understand the significance of the topographical, we need to discuss how timing analysis is done. For a timing analysis to be accurate the cell delay, as well as the wire delay, should be known. The cell delay is available in the standard cells library files but what about the wire delay? Pre-layout the tools can only guess the length of the wire since the cells haven't been placed yet. The tool uses what is known as the Wire load model to estimate the wire delay. A library usually has different wire load models for different design sizes. As the size of the design, cells are placed further apart leading to longer wires. Given data extracted from similar-sized designs with the same technology and nets fanout, the EDA tools are able to predict the wire delays that will result after the layout. The tools then optimize the design based on these predictions. The wire load model is based on a statistical average and is not specific for any design. This leads to inaccurate timing results and in ultra-deep submicron designs as the effects of the parasitics become more relevant the wire load model fails to deliver enough accuracy.

To overcome the problems of the wire load model the topographical technology is used. This technology aims to deliver a much better estimation of the wire delays. This is done by doing the synthesis over two runs. The first run is done to produce a starting netlist that is used to perform an initial floorplan placement for the cells. In the second synthesis run the physical information from the Milkyway database is then used to accurately predict the wire delays which produces much more accurate results. This flow is summarized in the following figure.

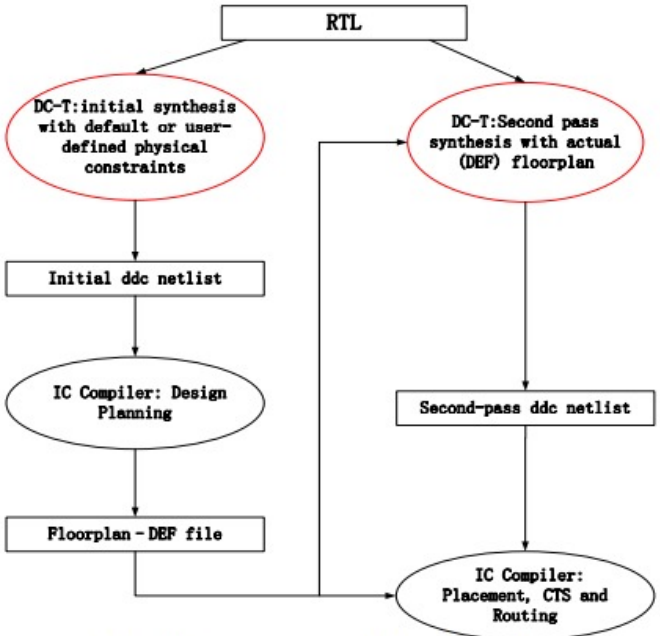


Figure 4.2: Topographical Flow[14]

4.2 Synthesis

Design Setup

The cv32e40p project utilizes new SystemVerilog constructs, syntax, and semantics which the old standards don't accommodate [4]. Therefore, the SystemVerilog 2009 standard is set as the first step in the design compiler synthesis script using *set hdlin_sverilog_std 2009*. The SystemVerilog 2009 standard merges the Verilog 2005 and SystemVerilog 2005 standards and adds additional extensions beyond them [16].

We then need to define the library files with (.db) file extension. It is possible to do that by setting the application variables *link_library* and *target_library* to the library files path. This can be done using the command: *set_app_variable*. Both those commands specify the design and technology libraries, respectively, to be used whilst compiling the design. The design library is then mapped into a UNIX directory to store intermediate design representations using the *define_design_lib* command [16].

In order to run the design in topographical mode, the milkyway library should be created and the TLUPlus files need to be inserted. we create the milkyway library that holds all the design files using the *create_mw_lib* command. This command has an argument (*-technology*) where the technology file with file extension (.tf) is defined. For inserting the parasitics files TLUPlus files (for parasitics extraction), we set the arguments: (*-max_tluplus*, *-min_tluplus* & *tech2itf_map*) using *set_tlu_plus_files* command [17].

The design HDL source files are then specified using the *analyze* command which translates and stores the design intermediate format in the design library specified before. The cv32e40p design uses clock gating cells, but only a simulation module for those cells was provided with the HDL, and they advised us to use the gating cells provided by the technology library we are working with. As a result, the standard clock gating cells were analyzed as well and their instances were added in the HDL design files. The cv32e40p also gave us the freedom of choosing between implementing the design using flip flop registers or latches. Because synthesis tools, in general, don't always handle latches properly, and whilst being asynchronous, they create difficulties in timing analysis and routing delays. This might lead the design to fail meeting the timing requirements and fail under PVT variations while increasing the design size, timing glitches, and oscillations. As a result, we chose the flip flop registers and analyzed their files as latches.

Build & Compilation [16]

Design-Build

After setting the Verilog standards, defining target libraries, and analyzing the design's HDL files we start by building the design from the intermediate formats saved in the design library using the *elaborate* command. The command *current_design* is then used to set the working design, then *check_design* is used to check the internal design representation for consistency and issues errors if the design has severe problems and the compiler can't accept it, or warnings acting as informative messages which don't necessarily indicate a problem.

The constraints (.tcl) file is then read, interpreted, and evaluated using *source* command. Then the *link* command is used to link and resolve the design references. In order for the design to be functional, it must be connected to all of the library components and designs it references to be located and linked to the current design. The *link* command uses the *link_library* and *search_path* variables along with the *local_link_library* design attribute to resolve those design references by looking in the files specified by those commands and linking them.

Design Compilation

Logic-level & gate-level synthesis and optimization are done by the *compile* command. Optimization trades off between timing and area constraints to provide the smallest circuit design meeting the timing requirements, operating at the required speed. In the optimization process, both design rule constraints & optimization

constraints are considered, but precedence and priority are given to the design rule constraints.

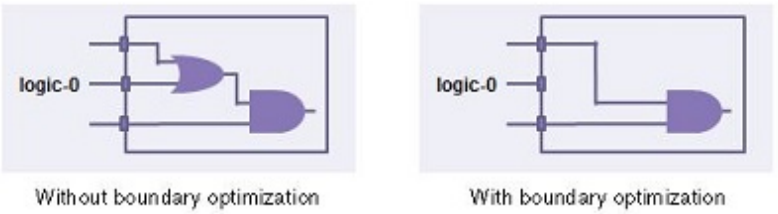
The *compile* command can take multiple optimization arguments which allows it to prioritize certain constraints over others.

- *-no_map*
    - Doesn't map the design to the specified target technology. It just represents the design in its generic boolean equations and flip flops.
- *-map_effort*
    - It specifies the relative CPU computational time the tool spends in the mapping phase of the synthesis.
- *-area_effort*
    - It specifies the relative CPU computational time the tool spends in the area recovery phase of the synthesis.
- *-power_effort*
    - It specifies the relative CPU computational time the tool spends in the power recovery phase of the synthesis.
- *-gate_clock*
    - It enables clock gating optimization by the automatic insertion and removal of the clock gates from the design. The optimization can be based on the switching activity and the dynamic power consumption of the used registers.
- *-incremental_mapping*
    - Allows incremental optimization to the design constraints based on previous runs.

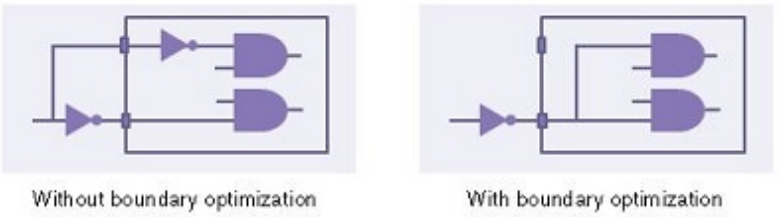
- -exact_map
    - It performs an exact mapping of the design's sequential elements. Under normal conditions, sequential units surrounded by logic elements may be used if their behaviour resembles the HDL. This argument specifies that an exact match must be used.
- *-no_design_rule*
    - It terminates the compile option before fixing the design rule violations. This allows us to see the results in the constraints report before fixing the violations.
- *-only_design_rule*
    - It performs only design rule fixing, thus, mapping optimizations aren't performed.
- *-no_hold_time*
    - It performs only hold time fixes, without fixing the design rule violations.
- *-scan*
    - It takes into account the DFT and scan insertion by replacing the sequential elements by scan flip flops.
- *-top*
    - It only fixes design rules violation & top-level timing violations without mapping the design.

- *-boundry_optimization*
    - It optimizes across all hierarchical boundaries of the design as shown below.

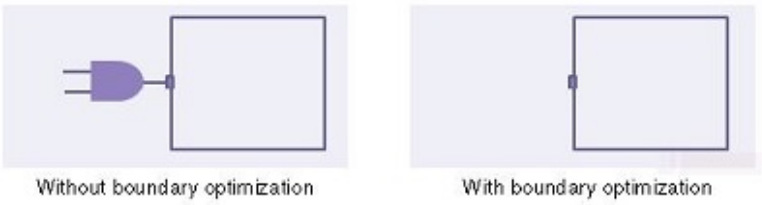
- Propagation of constants across the hierarchy:



- Propagation of equal and opposite information across the hierarchy:



- Propagation of unconnected port information across the hierarchy:



- Pushing of inverters across the hierarchy (phase inversion):

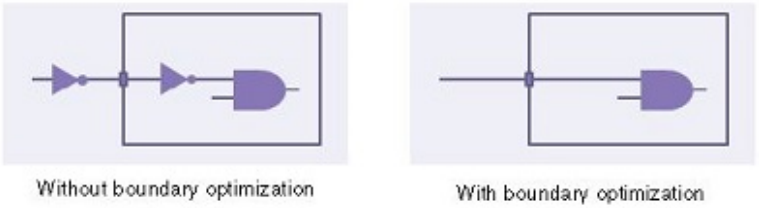


Figure 4.3: Boundary Optimizations [14]

Flat Flow

The *compile* command, by default, maps and optimizes the design with hierarchical levels and boundaries. The *compile* command can synthesize and optimize a flat design, using different ungrouping options as shown below.

- *-ungroup_all*
    - It collapses all hierarchical levels and their boundaries
- *-auto_ungroup area | delay*
    - If the *area* option is specified, it favours ungrouping the small hierarchical regions in the design for better area recovery.
    - If the *delay* option is specified, it performs automatic ungrouping for hierarchies which are more likely to improve the overall design timing.

Another command which can be used for logic & gate synthesis and optimization is the *compile_ultra* command where it performs a high-effort compile on the design for better Quality of Results (QoR). The *compile_ultra* command by default enables all the optimization ultra features, ungroups all hierarchies, and performs hierarchical boundary optimizations. It also uses *-timing_high_effort* & *-area_high_effort* options by default. The command follows strategies intended to prioritize the design's area on the compilation run-time. The *compile_ultra* command has similar features to the *compile* command with some options of its own.

- *-no_auto_ungroup*
    - It disables automatic ungrouping and preserves all hierarchies.
- *-no_seq_output_inversion*

    - By default, the *compile_ultra* command inverts the sequential elements during mapping and optimization. This argument disables sequential output inversion.
- *-no_boundry_optimization*
    - It disables any boundary optimizations that were to be automatically performed.
- *-retime*
    - It uses adaptive retime algorithms to further improve timing delays.
- *-self_gating*
    - It enables XOR self-gating insertion into the design.

Topographical Mode [17]

In order to synthesize the design in the topographical mode, the design has to go through two synthesis passes. At first, the design is synthesized with the design rule and optimization constraints to create an initial netlist. The netlist is then sent to ICC where real parasitic capacitance and resistance are calculated instead of using wire load models, then a floorplan with physical constraints is generated. The design is then sent back to go through the second pass synthesis where the floorplan constraints are used to create a more optimized netlist.

The *compile_ultra* command should be used in the second pass of topographical mode with *-spg* option. *-spg* enables physical guidance and congestion optimization. Congestion optimization reduces routing-related congestion. Physical guidance enables Design Compiler Graphical to save coarse placement information and pass this coarse placement information to IC Compiler. With this coarse placement, IC Compiler can begin the implementation flow with the *place_opt* command. It no longer needs to recreate the coarse placement.

Reports [16]

After the design compilation is complete, we start viewing different design reports to check area, timing and power along with other design metrics.

- *report_area*
    - It displays the area statistics for the current design. It reports combinantional, non-combiantiona, and total area along with the number of cells in the design.
- *report_resources*
    - It lists the resources (arithmetic and comparator units) & data path blocks (path containing more than one resource) in the current design.
- *report_timing*
    - *It displays the design timing information. It displays the worst setup path in the design by default along with its startpoint, endpoint, and various other info.*
- *report_power*
    - It calculates and reports the static and dynamic power of the design. The command uses the user-annotated switching activity to calculate the net switching power, cell internal power, and cell leakage power.
- *report_qor*
    - It displays the quality of results statistics and information for the current design. It reports timing-path group, cell count, design's area, static power, design-rule violations, and compile-time details.
- *report_constraint*
    - It displays all of the design rule and optimization violations in the design.

Outputs [16]

After the design is compiled and synthesized, the *write_sdc* command is used to output a Synopsys Design Constraints file which can be used as an input constraint for the ICC and PrimTime. The synthesis netlist is then written using the *write* command with *-format verilog* & *-output* arguments to be used by ICC in the PnR stage.

4.3 Floorplanning

Design Setup

Floorplanning is the first step of the PnR flow. In the beginning, the environment should be set up and the input files must be provided.

First, we need to define the library files with the (.db) file extension. It is possible to do that by setting the application variables *link_library* and *target_library* to the library files path. This can be done using the command: *set_app_variable* [17]. After that, we create the Milkyway library that holds all the design files. This is done using the command: *create_mw_lib.* This command has an argument (*-technology*) where the technology file with file extension (.tf) is defined. For topographical flow, there is no need to create a new Milkyway library as it was already created in the previous steps. To open the Milkyway library, the command *open_mw_lib* is used [17].

For inserting the parasitics files TLUPlus files (for parasitics extraction), we set the arguments: (*-max_tluplus, -min_tluplus & tech2itf_map)* using *set_tlu_plus_files* command. To read the netlist (.v or .vhd) that is generated by the synthesis stage, *the import_designs* command is used. The constraints file that is generated after synthesis (.tcl or .sdc) is read using source or read_sdc commands. By this step, the design setup is done and it is possible to move to floorplanning.

Floorplanning

The floorplan specifies a position for each macrocell or module within the design. It also assigns the pins of the design according to the positions of the cells within the design.

The first step is to define the constraints of the floorplan using *the create_floorplan* command. Here are the arguments of this command that define the floorplan [17]:

-control_type	Specifies how the core area of the floorplan is sized. The possible options are *aspect_ratio* (default), *width_and_height* or *boundary* (based on PnR boundary).
-core_aspect_ratio	Specifies the aspect ratio of the floorplan.
-core_utilization	Specifies the utilization for the core area (the total area of the core occupied by all standard cells and macro cells divided by the total core area).
-core_width/height	Specifies the dimensions of the core area
-left/right/bottom/top_io2core	Specifies the distance between the core area and the IO pads.
-no_double_back	Specifies that the command places pairs of cell rows without flipping one row in each pair.

It is a rule of thumb for ASIC designs to start the first row of cells from the bottom and to flip the first row.

In order to place hard macros or leaf cells, the command *create_fp_placement* is used. This command can be optimized for congestion or timing. Standard cells should not have the attribute *placed* after the floorplan.

4.4 Power Network

Power mesh generation is a very critical step in the flow as it connects each cell on the design to the power and ground pins. As mentioned in the previous chapter, the power network consists of trunks, power rings, power straps, power rails, and vias. Usually, the routing of the power network is done in the top two metal layers as they are the widest layer which is suitable for better IR drop and voltage level [13]. In order to set the layers for routing, *set_ignored_layers* with option *-max_routing_layer*. If the design has ten routing layers, then we set the maximum routing layer to 8 leaving the top two layers for power routing.

The logical power and ground connections are defined using the command *derive_pg_connection* with arguments: *-power_pin, -ground_pin, -power_net & -ground_net.*

To define the power network, the command *set_fp_rail_constraints* is used. Here are the main arguments of the command [17]:
For ring constraints:

-set_ring	To add power ring constraints.
-horizontal/vertical_ring_layer	Specifies the horizontal/vertical metal layers for the power ring.
-ring_spacing	Specifies the spacing between the core rings (in microns).

-ring_width	Specifies the width of the power rings (in microns).
-ring_offset	Specifies the offset from the I/O pads to the power rings (in microns).
-extend_strap	There are three ways to extend the core ring: *core_ring* (default) extends the power straps to the existing core ring, *boundary* extends the power straps to the top-level cell boundary and creates power pins, *pad_ring* extends the power straps to an existing pad ring.

For power straps constraints:

-add_layer	To add power strap layer constraints (mutually exclusive to *-set_ring*).
-layer	Specifies the layer on which to create the power grid.
-direction	Specify the direction for the power and ground wires (vertical or horizontal).
-max_strap	Controls the power strap density by setting the maximum number of straps (default is 128).
-min_strap	Controls the power strap density by setting the minimum number of straps (default is 16).
-max_pitch	Specifies the maximum pitch (in microns). This option is

	mutually exclusive to *-max_strap* and *-min_strap* options.
-min_pitch	Specifies the minimum pitch (in microns). This option is mutually exclusive to *-max_strap* and *-min_strap* options.
-max_width	Specifies the maximum width of the power wires (in microns).
-min_width	Specifies the minimum width of the power wires (in microns).
-spacing	Constrains the spacing between the power and ground wires. The options are minimum spacing with the *minimum* keyword (default), interleaving spacing with the *interleaving* keyword, or explicitly specifying the distance in microns.

The power virtual pads should be placed using the *create_fp_virtual_pad* command. It may also be specified using the GUI. It is a rule of thumb to place one pair of IO power pads every four to sex signal pads. The option *-net* is used to specify the name of the power or ground net [17].

In order to synthesize the power networks based on the user-specified constraints, the *synthesize_fp_rail* command is used. *-nets* option is used to specify the power and ground nets. The voltage supply is specified using the *-voltage_supply* option. It is also possible to define the target voltage drop and the power budget using the options *-target_voltage_drop* and *-power_budget*. After that, the command *commit_fp_rail* is used to commit the power network based on the power network synthesis results [17].

The next step is to connect standard cell power and ground pins to the power and ground rings and straps and to connect power and ground rails in the standard cells. This is done using the command *preroute_standard_cells*. There are many options that can be used with this command. For example, *-fill_empty_rows* is used to fill all rows, even those without cells, with power and ground rails. *remove_floating_pieces* is used to remove floating, unconnected rail segments [17].

After committing the power network, it is needed to analyze it for voltage (IR) drop and electromigration (EM) on the specified power and ground nets. This is done using the *analyze_fp_rail* command. *-nets* option is used to specify the power and ground nets. Also, *-power_budget* and *-voltage_supply* are used to specify the power budget and the voltage supply [17].

After the power network and before global placement, it is possible to add tab cells using the command *add_tap_cell_array*. This command adds tap cells to the design, forming a two-dimensional array structure. A TAP cell is a special non-logic cell with a well tie, substrate tie, or both. Tap cells are typically used when most or all standard cells in the library contain no substrate or well taps [14]. The design rules typically specify a distance limit from every transistor of a standard cell to a well or substrate tie. Tap cells are inserted before global placement so that all standard cells that are subsequently placed can satisfy the distance limit.

The flat and hierarchical flow have no differences after the synthesis stage. However, the topographical flow uses the information gathered until this stage such as deep

submicron effects and interconnect parasitics in order to have accurate estimates of resistance and capacitance that are necessary to calculate path delays in synthesis without the need for wire load model-based timing approximations. For this reason, *the write_def* or *write_floorplan* command is used to save the floorplan information in (.def) or (.fp) format in order to be used in the second pass of synthesis in topographical mode.

4.5 Placement

Placement is the next step after finishing the power network. However, for topographical flow, it is done directly after the second synthesis pass. Thus, it is needed to set up the design as mentioned previously in the floorplanning section. In addition to that, it is needed to read the floorplan file of coarse placement information generated from the synthesis tool [15]. This is done using the *read_floorplan* command.

Before starting the placement process, it is necessary to do some pre-placement checks. To check physical constraints, the *check_physical_constraints* command is used to provide information about possible errors in the input. It checks for cell areas in hardbound, correct layers in the library against those in the floorplan, resistance, and capacitance for different route layers, narrow placement areas in the floorplan, legal sites for library cells in the floorplan, etc. Also, the *check_physical_design -stage pre_place_opt* command is used to check that the floorplan and netlist data are ready and the design constraints are set [17].

In order to generate a legally placed netlist, the command *place_opt* is used. This command can be optimized for area, congestion, power, or DFT using the options: *-area_recovery, -optimize_dft, -congestion*, and *-power*. For topographical mode, *the -spg* option must be used in order to enable features of using Synopsys Physical Guide information from the Synopsys Design Compiler tool. The *compile_ultra* command in Design Compiler needs to run with the same option. With the *-spg* option, the

place_opt uses Design Compiler's Physical Guide information to guide optimization [16].

It is possible to use *the psynopt* command after the initial placement in order to perform incremental timing-driven logic optimization with placement legalization. This command optimizes for setup time by default, but it has options that make it possible to optimize on area, power, congestion, hold time, and design rules.

After placement, the command *check_legality* is used to check the legality of the placement and report any violations. If there are any violations, the *legalize_placement* command can be used to perform placement legalization on the current design after the coarse placement has already been performed.

Before moving to the CTS stage, tie cells are inserted in order to tie unused inputs to logic high or logic low using tie-high or tie-low cells. This is done by first defining the tie pins group and using the command *connect_tie_cells* in order to tie them.

4.6 CTS

Clock tree synthesis is done after placement in order to connect each flip flop of the design to the clock signal. Before going through CTS, some checks should be made. Mainly, the *check_physical_design -stage pre_clock_opt* command is used to check the same items as *pre_place_opt*. In addition, the design must be placed and the clock constraints must be set. The command *check_clock_tree* is used for common problems that might adversely impact clock tree synthesis in order to make sure that the design is ready to enter the CTS stage.

Most importantly, the design should be constrained as needed. So, the *set_driving_cell* command is used to set attributes on the clock input port of the current design to specify the library cell that drives the clock. The option *-lib_cell* is used to specify the driving buffer cell. Some other parameters should be inputted to the constraints through *set_clock_tree_options* such as Target skew, Max Fanout, Max Capacitance, Max Transition. The below table shows some options for this command. We also used the command *define_routing_rule* to define the Non-default-rules (NDR) and the metal layers that will be used for CTS.

-clock_trees	Specifies the clock trees on which to set the specified clock tree options.
-layer_list	Specifies the layers that can be used for routing the clock nets in the specified clock trees.

-target_early_dela y	Specifies the minimum insertion delay constraint in nanoseconds for the specified clock trees.
-target_skew	Specifies the desired value for maximum skew in nanoseconds for the specified clock trees.
-max_capacitance	Specifies the maximum capacitance design rule constraint in main library units for the specified clock trees.
-max_transition	Specifies the maximum transition time design rule constraint in the main library unit for the buffers and inverters used.
-max_fanout	Specifies the maximum fanout design rule constraint for the cells in the specified clock trees.
-buffer_relocation	Enables and disables buffer relocation on the specified clock trees during the clock tree optimization.
-buffer_sizing	Enables or disables buffer sizing on the specified clock trees during the clock tree optimization.
-gate_relocation	Enables or disables gate relocation on the specified clock trees during the clock tree optimization.
-gate_sizing	Enables or disables gate sizing on the specified clock trees during the clock tree optimization.

The first stage of CTS is the synthesis by the command *clock_opt -only_cts -no_clock_route* which will synthesize the clock tree but without routing so it has

faster runtime. The second stage is optimization using the command *clock_opt -only_psyn -no_clock_route.* The designer might consider focusing on fixing hold violations by the command *set_fix_hold [all_clocks].* The third stage is Clock Tree routing by using *route_group -all_clock_nets.* Also optimization can be done with *clock_opt -only_psyn -congestion.* Timing and congestion analyses are made between the stages to ensure the target is achieved.

4.7 Routing

Routing of signal nets is done after CTS. Before the routing stage, Spare cells should be added to the design using the command *insert_spare_cells -lib_cell {NOR2_X4 NAND2_X4}*. Spare cells are used with ECO to provide new functions to a design that exhibits post-production problems. If not used, they are tied to VDD or VSS to help avoid electrostatic discharge, ground, or power bounce. The command *check_routeability* is then used to check the design is ready to be routed.

Defining the routing constraints starts by setting the wire delay model with the command *set_delay_calculation_options.* It selects the delay model as Arnoldi or Elmore in addition to the effort for post-route designs. To set the routing options, the command *set_route_options* is used. Its options can specify the weight of timing, congestion, and power in the stages of routing: global routing, track assignment, and detailed routing. To define the signal integrity options that are used for analysis or optimization, the command *set_si_options* is used. It may state whether the cross-talk between signals or the minimum delta delay is considered or not.

Routing starts with *route_auto* command. It performs global routing, track assignment, detail routing, and search and repair in one step. Then, *route_opt* which does global routing and post route optimization on the design. The options of this command can specify the focus of the optimization such as timing, power, area, crosstalk, etc. Also, detailed routing can be done using the command *route_zrt_detail* which helped reduce LVS and DRC greatly after using it many times. It was also effective to use *focal_opt -drc_nets all, focal_opt -drc_pins all* to reduce DRC errors.

Mainly, it performs post-route optimization to fix setup, hold, or logical design rule constraint (DRC) violations on the design. The selected optimization is referred to as the focal metric. Also, the *psynopt* command can be used to perform incremental preroute or post-route synthesis on the current design.

Sometimes, it is a good idea to remove all short nets and reroute them in order to solve short violations. This can be done by removing the group of all short nets within the design and rerouting them using eco routing. The commands to do this are as follows:

```
foreach_in_collection net [get_attribute [get_drc_errors -filter {type == "Short"}] nets] {
        if {[sizeof_collection [get_nets $net] ]} {
         if {[sizeof_collection [get_nets -quiet -filter "net_type!=power && net_type!=ground" $net ] ]} {
        if {[sizeof_collection [get_vias -quiet -of_objects [get_nets $net] ] ] } { remove_via [get_vias -of_objects [get_nets $net] ] }
      if {[sizeof_collection [get_net_shapes -quiet -of_objects [get_nets $net] ] ] } { remove_net_shape [get_net_shapes -of_objects [get_nets $net] ] }
   }
  }
}
route_zrt_eco -open_net_driven true -utilize_dangling_wires true -reroute modified_nets_first
```

To verify the routing, *verify_zrt_route* and *verify_lvs* are used. There are many other routing commands that can be used but the one that made a more significant change was *route_zrt_eco -open_net_driven true -reuse_existing_global_route true -reroute* which helped decrease open nets drastically.

4.8 STA

PrimeTime is a sign-off timing analysis tool that is used to ensure that the design meets the timing requirements[18]. It is used in post-layout STA to verify that the intended clock speed for the design is met at a wide range of PDK process corners[18]. The operation of PrimeTime can be categorized into three main points; Setup, Violations Fixing and Applying modifications. These points and the details of PrimeTime operation are thoroughly discussed in this section.

4.8.1 Setup

In this stage, we provide PrimeTime with the needed inputs such as the Technology Library, the Verilog netlist, the constraints file and the extracted parasitics[18]. We also specify the design clock so that PrimeTime can use its delays to accurately provide a timing report.

The first step in using PrimeTime is to specify the Technology Library to be used. For every PDK there exists a wide range of corners that tries to estimate the expected variations in the manufacturing process as well as the operating environment temperature and voltage. These variations become more significant in smaller technologies. The manufacturing process variations are classified into three categories: slow-slow, typical-typical, and fast-fast process[14]. Indicated by the naming, the fast-fast corner results in the fastest logic with the lowest cell delay while the slow-slow corner gives the worst delay for the cells[14]. The variation in temperature has also a great effect on the design performance as in high temperatures

the cells have higher delays than in lower temperatures[14]. For the PDK Nangate that we are using in this project, there are five available temperatures which are -40, 0, and 125 °C. The last possible variation is the operating voltage. The voltage has a great effect on the cell speed as higher voltage leads to lower delay. Nangate comes with two different voltage variants 0.95V and 1.25V. After choosing the desired library that contains the delay information for the combination of these three possible variations, we import it using the *set link_path* command[18].

set link_path "/library_path/NangateOpenCellLibrary_ff1p25v0c.db"

In this example, we import the fast-fast, 1.25V and 0 °C libraries which we will use in this specific test.

The next step is to provide a gate-level netlist that contains the standard cell used in the design and their connections. This comes in the form of a Verilog netlist from ICC. The command *read_verilog* is used and the path for the netlist is specified as shown in the following example[18].

read_verilog "../outputs/${design}_icc.v"

We then link the gates in the netlist to the standard cells in the technology file through the *link* command.

Next, we specify the timing constraints for the design. These constraints are the same as the ones used in the Synthesis and PNR. This is done through sourcing the constraints file similar to what we have done previously.

We then provide the parasitics file extracted from the physical layout in ICC. Since these parasitics are only an estimation for the RC resulting from the wires and physical layout, we have a best-case and a worst-case scenario[18]. The best-case results lead to a faster design which is relevant for calculating possible hold time violations[18]. On the other hand, the worst parasitics lead to longer delays which in turn causes setup time violations. The min or max parasitics file is chosen based on the type of timing violation we want to check in the test. To import the SPEF file that contains the parasitics information we use the *read_parasitics* command[18].

read_parasitics ../../Pnr/output/${design}.spef.max

In this example, we load the SPEF file that contains the max parasitic information indicating that the desired timing violations that we want to check are the setup time.
Lastly, we specify the propagated clock that PrimeTime will use for STA calculation. Since the clock tree is synthesized in PNR we can accurately measure its skew and delays of the clock. This step is important because without it, PrimeTime will use an ideal clock with no skew or it will use clock uncertainty if specified in the constraints file. Doing so will result in inaccurate timing reports and subsequently missing critical time violations or reporting non-existing ones. To use the synthesised clock, we use the command *set_propagated_clock* and specify the clock net like so[18].

set_propagated_clock [get_clocks clk_i]

4.8.2 Violations Fixing

After finishing the setup stage, we will check if the design correctly meets the timing requirements. If it doesn't, PrimeTime provides a synthesis engine that can suggest modifications to the design netlist in order to solve the setup or the hold violations.

To check the timing, we first use the command *update_timing* which updates timing for the current design[18]. Then we use the *report_timing* command and provide it with the type of violation we want to check[18]. We use max to check for setup violations and min to check for hold violations. The type of violation desired to be checked must match the SPEF parasitic file as previously discussed. If the slack is zero or positive then there is no need for adjustments and the timing is met. If the timing is not met, modifications to the design netlist are required.

In order to fix the timing violation, we used the command *fix_eco_timing*. PrimeTime resolves the Setup Fixing by upsizing the cells which in turn decrease their delay and improve setup slack[18]. The drawback of this solution is that it increases the area of the cells which might cause problems in the layout. Restrictions on the resizing of the cells can be imposed to limit the increase in the area using the eco_alternative_area_ratio_threshold setting. To solve the hold violations PrimeTime uses cell downsizing or puffer insertion or both this is done in order to add delay and improve hold slack. The *fix_eco_timing* command has multiple options and settings that help the user to solve the timing violations without causing further problems[18]. Some of these settings are:

-type fixing_type	Specifies the desired fixing type setup or hold.

-methods method_list	Specifies a fixing method type which include size_cell, insert_buffer_at_load_pins and insert_buffer. Either one or multiple of them can be used.
-slack_lesser_than *-slack_greater_than*	Specifies that only those paths with a slack less or greater than the slack_limit option are to be fixed.
-setup_margin *-hold_margin*	Specifies an additional fixing margin to be applied to setup or the hold slack. By default, the setup margin is zero.
-buffer_list	Specifies the list of allowable buffers from the technology library to be used for hold fixing.
-ignore_drc	If the time violating path contains DRC error PrimeTime will not try to solve the timing problem in that bath. This argument is used to override this behavior so that the DRC error is ignored and prime time would prioritize the timing fix.

4.8.3 Applying modifications

After PrimeTime makes modifications to the netlist, we would need to write them into ICC so the layout would be adjusted. To do so, we use the *write_changes* which outputs the netlist changes that have been made to the design since it was linked[17].

write_changes -format icctcl -output ./ecol.tcl

In the previous command, the output is formatted as icctcl which is a TCL script for ICC. We then run the TCL script on ICC which applies the changes by resizing existing cells or adding buffers. The added cell placement will need to be legalized and modifications to the routing are done through the *route_eco* command[17]. We then extract the netlist and the parasitics from the modified physical design and run it through PrimeTime again. Ideally, the violations would be fixed however multiple iterations might be needed. Finally, after fixing all the setup and holding time for a certain corner, further tests are done to make sure the design meets the timing requirements under all conditions.

4.9 Finishing, Checks & Outputs

After routing the signal nets, some last finishing and checks are made before the generation of the GDSII file. Finishing the design includes adding filler cells and tieng spare cells to VDD or VSS. The timing checks are done using STA tools as explained in the previous section. The physical validation includes design rule check (DRC), layout versus schematic (LVS), electrical rule check (ERC) and lithography process.

Chapter 5

Results

5.1 Flat Flow

5.1.1 Synthesis

As previously discussed, the Design Compiler can perform more optimizations on the flat design flow compared to the hierarchical. This is due to the possibility of boundary optimizations between modules. To explore this advantage, multiple incremental runs were performed while we were tightening the timing constraints to push the tool to give the best results. After multiple iterations a clock period of 2.2 ns was reached which corresponds to 454.54 MHz. The following figure 5.1 represents the timing report for the most critical path.

```
Point                                                          Incr       Path
---------------------------------------------------------------------------
clock clk_i (rise edge)                                        0.00       0.00
clock network delay (ideal)                                    0.00       0.00
ex_stage_i/alu_i/R_4429/CK (DFFS_X1)                           0.00 #     0.00 r
ex_stage_i/alu_i/R_4429/Q (DFFS_X1)                            0.08       0.08 r
ex_stage_i/alu_i/U64/ZN (AND2_X1)                              0.04       0.11 r
ex_stage_i/alu_i/U1722/ZN (NAND4_X1)                           0.03       0.15 f
ex_stage_i/alu_i/U1721/ZN (NOR2_X1)                            0.04       0.19 r
ex_stage_i/alu_i/U497/ZN (AND3_X1)                             0.06       0.24 r
ex_stage_i/alu_i/U1737/ZN (NAND3_X1)                           0.04       0.28 f
ex_stage_i/alu_i/U1732/ZN (OR2_X1)                             0.05       0.33 f
ex_stage_i/alu_i/U634/ZN (NAND4_X1)                            0.03       0.36 r
ex_stage_i/alu_i/U1371/ZN (NAND3_X1)                           0.04       0.40 f
ex_stage_i/alu_i/comparison_result_o (cv32e40p_alu)            0.00       0.40 f
id_stage_i/branch_decision_i (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_
                                                               0.00       0.40 f
id_stage_i/U839/ZN (AND2_X1)                                   0.05       0.45 f
id_stage_i/U1588/ZN (NOR2_X1)                                  0.03       0.48 r
id_stage_i/U596/ZN (AND2_X1)                                   0.04       0.52 r
id_stage_i/U603/ZN (NAND3_X2)                                  0.05       0.58 f
id_stage_i/pc_set_o (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_N_HWLP2_I
                                                               0.00       0.58 f
U1668/ZN (NAND2_X1)                                            0.05       0.63 r
U1667/ZN (NOR2_X1)                                             0.04       0.67 f
U1373/Z (BUF_X2)                                               0.05       0.72 f
U2567/ZN (AOI22_X1)                                            0.05       0.77 r
U2566/ZN (NAND3_X1)                                            0.04       0.81 f
U2530/ZN (NOR3_X1)                                             0.04       0.85 r
U2531/ZN (AOI21_X1)                                            0.03       0.88 f
instr_addr_o[18] (out)                                         0.00       0.88 f
data arrival time                                                         0.88

clock clk_i (rise edge)                                        2.20       2.20
clock network delay (ideal)                                    0.00       2.20
output external delay                                         -1.32       0.88
data required time                                                       0.88
---------------------------------------------------------------------------
data required time                                                       0.88
data arrival time                                                       -0.88
---------------------------------------------------------------------------
slack (MET)                                                              0.00
```

Figure 5.1 : Flat Timing Report

The expected total power consumption of the design is 1.3414 mW. Only 23.2% of that power is consumed by the clock network while 31.9 % and 44.9% are consumed by the combinational logic and the registers respectively. The power consumption is summarized in the following report (figure 5.2).

```
                 Internal         Switching           Leakage             Total
Power Group      Power            Power               Power               Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network    129.5200         180.6785          971.4440          311.1700  (  23.20%)
register         538.7291          24.6227        3.8997e+04          602.3519  (  44.90%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational    106.3523         218.7421        1.0278e+05          427.8772  (  31.90%)
--------------------------------------------------------------------------------------------------
Total            774.6014 uW      424.0432 uW     1.4275e+05 nW     1.3414e+03 uW
```

Figure 5.2: Flat Power Report

The total of the design is about 0.0335 mm2. The majority of this area or about 58.5% is dedicated to combinational logic. The following figure 5.3 of the area report summarizes this data.

```
Number of ports:                          536
Number of nets:                          4163
Number of cells:                         2247
Number of combinational cells:           1962
Number of sequential cells:               280
Number of macros:                           0
Number of buf/inv:                        446
Number of references:                      58

Combinational area:       19679.744133
Buf/Inv area:             1926.904002
Noncombinational area:    13908.874425
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          33588.618558
```

Figure 5.3 : Flat Area Report

5.1.2 Floorplan

We used a floorplan with 25% core utilization. An Incremental floorplan placement was needed to decrease the congestion. Following are figures for the floorplan (figure 5.4), the timing reports (figure 5.5 and 5.6) and the congestion map (figure 5.7) after this stage.

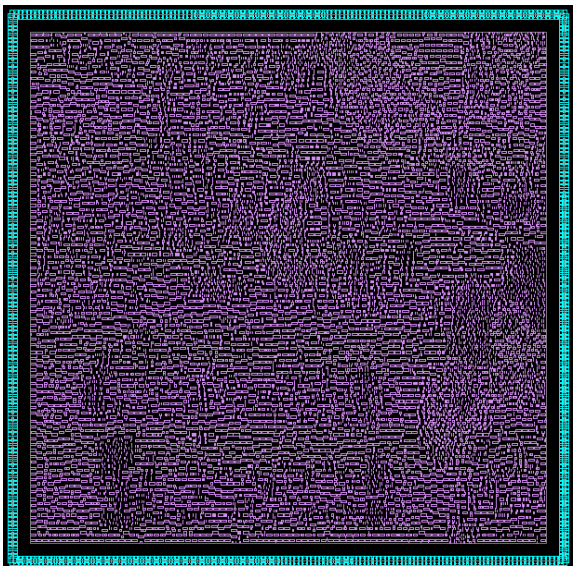


Figure 5.4: Flat Floorplan

```
Point                                                   Incr       Path
-----------------------------------------------------------------------
clock clk_i (rise edge)                                 0.00       0.00
clock network delay (ideal)                             0.00       0.00
ex_stage_i_alu_i/R_4205/CK (SDFFS_X1)                   0.00 #     0.00 r
ex_stage_i_alu_i/R_4205/Q (SDFFS_X1)                    0.08       0.08 r
ex_stage_i_alu_i/U64/ZN (AND2_X1)                       0.04 *     0.12 r
ex_stage_i_alu_i/U1722/ZN (NAND4_X1)                    0.03 *     0.14 f
ex_stage_i_alu_i/U1721/ZN (NOR2_X1)                     0.03 *     0.18 r
ex_stage_i_alu_i/U497/ZN (AND3_X1)                      0.05 *     0.23 r
ex_stage_i_alu_i/U1737/ZN (NAND3_X1)                    0.02 *     0.25 f
ex_stage_i_alu_i/U1732/ZN (OR2_X1)                      0.04 *     0.30 f
ex_stage_i_alu_i/U634/ZN (NAND4_X1)                     0.01 *     0.31 r
ex_stage_i_alu_i/U1371/ZN (NAND3_X1)                    0.07 *     0.38 f
ex_stage_i_alu_i/comparison_result_o (cv32e40p_alu)     0.00       0.38 f
id_stage_i/branch_decision_i (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_
                                                        0.00       0.38 f
id_stage_i/U878/ZN (NAND2_X1)                           0.05 *     0.42 r
id_stage_i/U73/ZN (NAND3_X1)                            0.03 *     0.46 f
id_stage_i/U598/Z (CLKBUF_X1)                           0.04 *     0.50 f
id_stage_i/U148/ZN (NAND2_X1)                           0.06 *     0.55 r
id_stage_i/U709/ZN (INV_X1)                             0.05 *     0.60 f
id_stage_i/U326/ZN (NOR2_X1)                            0.12 *     0.72 r
id_stage_i/exc_cause_o[2] (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_N_H
                                                        0.00       0.72 r
U2577/ZN (AND2_X1)                                      0.08 *     0.80 r
U2576/ZN (NOR2_X1)                                      0.02 *     0.82 f
U1685/ZN (AND3_X1)                                      0.03 *     0.85 f
U1686/ZN (AOI21_X1)                                     0.04 *     0.89 r
instr_addr_o[4] (out)                                   0.00 *     0.89 r
data arrival time                                                  0.89

clock clk_i (rise edge)                                 2.20       2.20
clock network delay (ideal)                             0.00       2.20
output external delay                                  -1.32       0.88
data required time                                                 0.88
-----------------------------------------------------------------------
data required time                                                 0.88
data arrival time                                                 -0.89
-----------------------------------------------------------------------
slack (VIOLATED)                                                  -0.01
```

Figure 5.5: Floorplan Timing Report Max

```
Point                                                   Incr       Path
-----------------------------------------------------------------------
clock clk_i (rise edge)                                 0.00       0.00
clock network delay (ideal)                             0.00       0.00
cs_registers_i/mie_q_reg_25_/CK (DFFR_X1)               0.00 #     0.00 r
cs_registers_i/mie_q_reg_25_/QN (DFFR_X1)               0.06       0.06 r
cs_registers_i/U619/ZN (OAI22_X1)                       0.01 *     0.08 f
cs_registers_i/mie_q_reg_25_/D (DFFR_X1)                0.00 *     0.08 f
data arrival time                                                  0.08

clock clk_i (rise edge)                                 0.00       0.00
clock network delay (ideal)                             0.00       0.00
cs_registers_i/mie_q_reg_25_/CK (DFFR_X1)               0.00       0.00 r
library hold time                                       0.01       0.01
data required time                                                 0.01
-----------------------------------------------------------------------
data required time                                                 0.01
data arrival time                                                 -0.08
-----------------------------------------------------------------------
slack (MET)                                                        0.07
```

Figure 5.6: Floorplan Timing Report Min

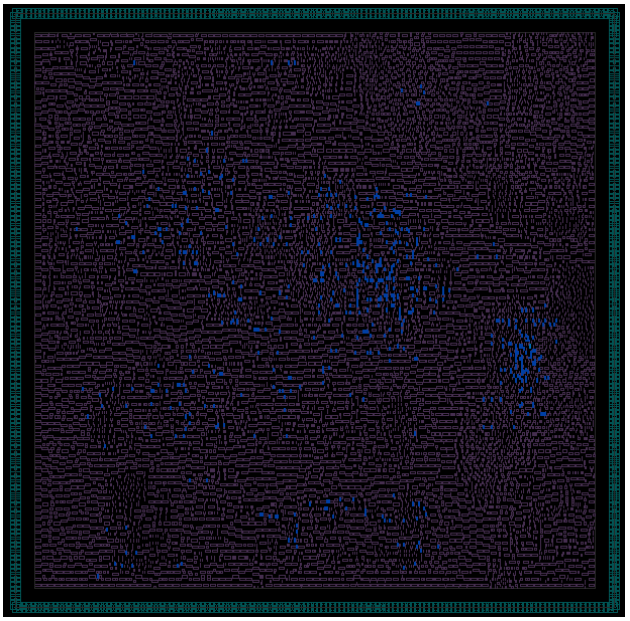


Figure 5.7: Flat Floorplan Congestion Map

5.1.3 Power Network

For creating the power network, we used the top five metal layers to create the power straps then we connected the cells to the power rails. Using metals from 6 to 10 for power straps helped to decrease the overall IR drop. We also used an interleaving pattern for metal 6, 7, and 8 power straps to allow for more routing tracks on these metal layers. Doing so helped to decrease the congestion while maintaining a good power distribution. Follows are the figures showing the power network IR drop distribution (figures 5.8 and 5.9) as well as timing (figures 5.10 and 5.11) and congestion (figure 5.12) after creating the power network.

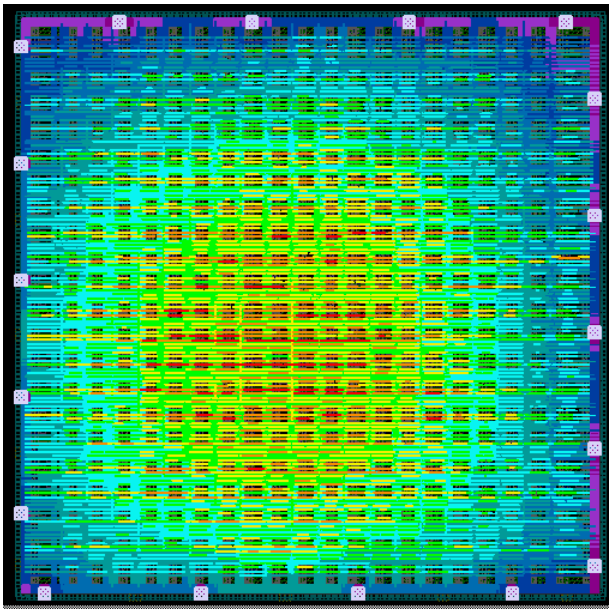


Figure 5.8: Power Network

```
Net Name: VDD
Number of Pad Cells: 0
Number of Pad Cells Supplying Current: 0
Number of Virtual Pad Nodes: 18
Number of Virtual Pad Nodes Supplying Current: 18
Number of Net Wires: 4023
Number of Net Vias: 30838
Number of Extracted Resistors: 106380
Number of Extracted Nodes: 86038
Current from Pad Cells:    0.00 mA (0.00%)
Current from Virtual Pad Nodes:  454.55 mA (100.00%)
Maximum IR drop:   8.060 mV at (156.8000 155.1150) (161.5500 155.2850) layer metal1
Maximum Wire EM: 5.588713e+01 A/cm at (225.3620 0.1640) (230.3620 3.3000) layer metal10
Maximum Via EM: 9.315198e+05 A/cm_2 at (304.9975 208.3300) (305.8425 208.4700) layer via2
```

Figure 5.9: Power Network Summary

Point	Incr	Path
clock clk_i (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
ex_stage_i_alu_i/R_4205/CK (SDFFS_X1)	0.00 #	0.00 r
ex_stage_i_alu_i/R_4205/Q (SDFFS_X1)	0.08	0.08 r
ex_stage_i_alu_i/U64/ZN (AND2_X1)	0.04 *	0.12 r
ex_stage_i_alu_i/U1722/ZN (NAND4_X1)	0.03 *	0.14 f
ex_stage_i_alu_i/U1721/ZN (NOR2_X1)	0.03 *	0.18 r
ex_stage_i_alu_i/U497/ZN (AND3_X1)	0.05 *	0.23 r
ex_stage_i_alu_i/U1737/ZN (NAND3_X1)	0.02 *	0.25 f
ex_stage_i_alu_i/U1732/ZN (OR2_X1)	0.04 *	0.30 f
ex_stage_i_alu_i/U634/ZN (NAND4_X1)	0.01 *	0.31 r
ex_stage_i_alu_i/U1371/ZN (NAND3_X1)	0.07 *	0.38 f
ex_stage_i_alu_i/comparison_result_o (cv32e40p_alu)	0.00	0.38 f
id_stage_i/branch_decision_i (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_N		
	0.00	0.38 f
id_stage_i/U878/ZN (NAND2_X1)	0.05 *	0.42 r
id_stage_i/U73/ZN (NAND3_X1)	0.03 *	0.46 f
id_stage_i/U598/Z (CLKBUF_X1)	0.04 *	0.50 f
id_stage_i/U148/ZN (NAND2_X1)	0.06 *	0.55 r
id_stage_i/U709/ZN (INV_X1)	0.05 *	0.60 f
id_stage_i/U326/ZN (NOR2_X1)	0.12 *	0.72 r
id_stage_i/exc_cause_o[2] (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_N_HW		
	0.00	0.72 r
U2577/ZN (AND2_X1)	0.08 *	0.80 r
U2576/ZN (NOR2_X1)	0.02 *	0.82 f
U1685/ZN (AND3_X1)	0.03 *	0.85 f
U1686/ZN (AOI21_X1)	0.04 *	0.89 r
instr_addr_o[4] (out)	0.00 *	0.89 r
data arrival time		0.89
clock clk_i (rise edge)	2.20	2.20
clock network delay (ideal)	0.00	2.20
output external delay	-1.32	0.88
data required time		0.88
data required time		0.88
data arrival time		-0.89
slack (VIOLATED)		-0.01

Figure 5.10: Power Timing Report Max

Point	Incr	Path
clock clk_i (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
cs_registers_i/mie_q_reg_25_/CK (DFFR_X1)	0.00 #	0.00 r
cs_registers_i/mie_q_reg_25_/QN (DFFR_X1)	0.06	0.06 r
cs_registers_i/U619/ZN (OAI22_X1)	0.01 *	0.08 f
cs_registers_i/mie_q_reg_25_/D (DFFR_X1)	0.00 *	0.08 f
data arrival time		0.08
clock clk_i (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
cs_registers_i/mie_q_reg_25_/CK (DFFR_X1)	0.00	0.00 r
library hold time	0.01	0.01
data required time		0.01
data required time		0.01
data arrival time		-0.08
slack (MET)		0.07

Figure 5.11: Power Timing Report Min

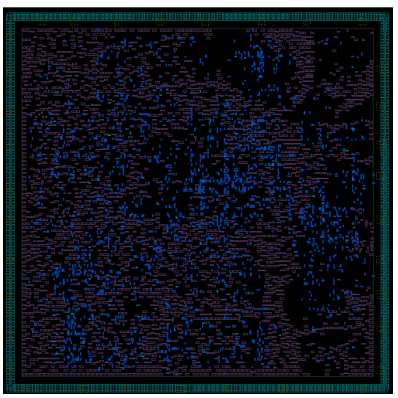


Figure 5.12: Power Network Congestion

5.1.4 Placement

The main goal of the placement step was to reach a legalized detailed placement for the cells that achieve the minimum congestion while maintaining good timing. To achieve this goal several incremental placements focused on fixing the congestion was needed as initial placement had some congestion problems. The following figures show the final congestion map (figure 5.13) and the timing reports (figures 5.14 and 5.15) after the placement.

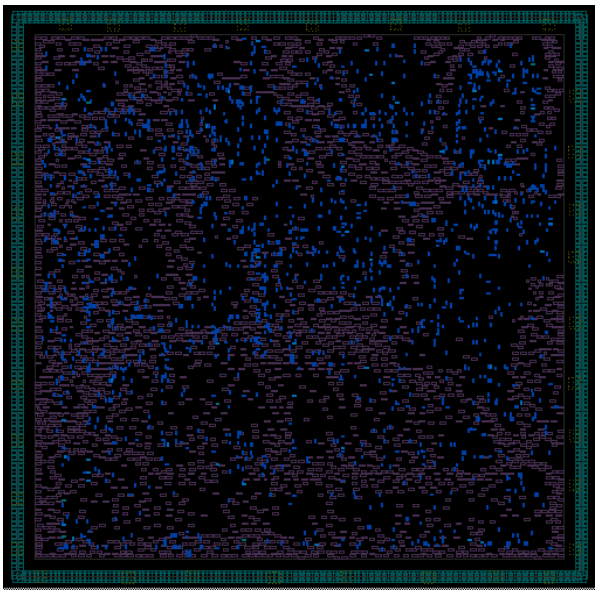
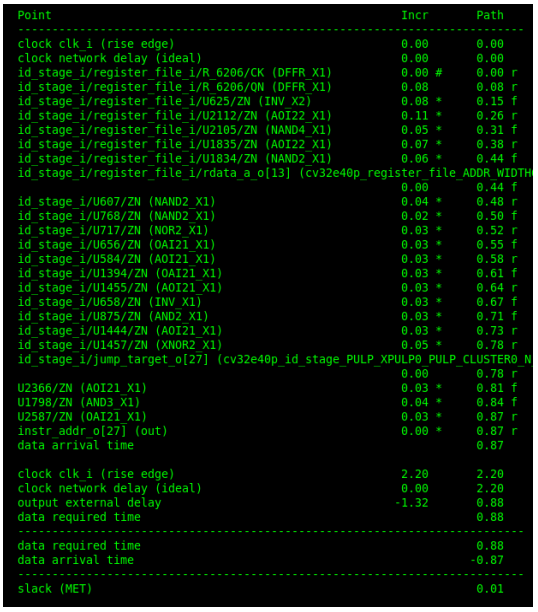


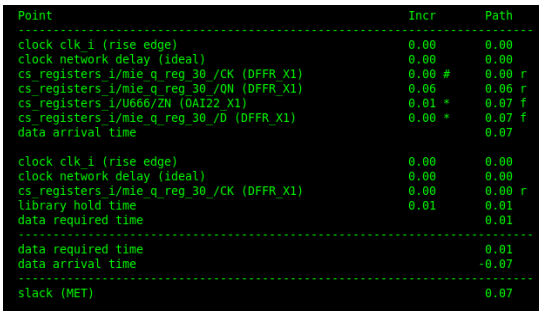
Figure 5.13: Placement Congestion



```
Point                                                    Incr       Path
-----------------------------------------------------------------------
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (ideal)                              0.00       0.00
id_stage_i/register_file_i/R_6206/CK (DFFR_X1)           0.00 #     0.00 r
id_stage_i/register_file_i/R_6206/QN (DFFR_X1)           0.08       0.08 r
id_stage_i/register_file_i/U625/ZN (INV_X2)              0.08 *     0.15 f
id_stage_i/register_file_i/U2112/ZN (AOI22_X1)           0.11 *     0.26 r
id_stage_i/register_file_i/U2105/ZN (NAND4_X1)           0.05 *     0.31 f
id_stage_i/register_file_i/U1835/ZN (AOI22_X1)           0.07 *     0.38 r
id_stage_i/register_file_i/U1834/ZN (NAND2_X1)           0.06 *     0.44 f
id_stage_i/register_file_i/rdata_a_o[13] (cv32e40p_register_file_ADDR_WIDTH
                                                         0.00       0.44 f
id_stage_i/U607/ZN (NAND2_X1)                            0.04 *     0.48 r
id_stage_i/U768/ZN (NAND2_X1)                            0.02 *     0.50 f
id_stage_i/U717/ZN (NOR2_X1)                             0.03 *     0.52 r
id_stage_i/U656/ZN (OAI21_X1)                            0.03 *     0.55 f
id_stage_i/U584/ZN (AOI21_X1)                            0.03 *     0.58 r
id_stage_i/U1394/ZN (OAI21_X1)                           0.03 *     0.61 f
id_stage_i/U1455/ZN (AOI21_X1)                           0.03 *     0.64 r
id_stage_i/U658/ZN (INV_X1)                              0.03 *     0.67 f
id_stage_i/U875/ZN (AND2_X1)                             0.03 *     0.71 f
id_stage_i/U1444/ZN (AOI21_X1)                           0.03 *     0.73 r
id_stage_i/U1457/ZN (XNOR2_X1)                           0.05 *     0.78 r
id_stage_i/jump_target_o[27] (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_N
                                                         0.00       0.78 r
U2366/ZN (AOI21_X1)                                      0.03 *     0.81 f
U1798/ZN (AND3_X1)                                       0.04 *     0.84 f
U2587/ZN (OAI21_X1)                                      0.03 *     0.87 r
instr_addr_o[27] (out)                                   0.00 *     0.87 r
data arrival time                                                   0.87

clock clk_i (rise edge)                                  2.20       2.20
clock network delay (ideal)                              0.00       2.20
output external delay                                   -1.32       0.88
data required time                                                  0.88
-----------------------------------------------------------------------
data required time                                                  0.88
data arrival time                                                  -0.87
-----------------------------------------------------------------------
slack (MET)                                                         0.01
```

Figure 5.14: Placement Timing Report Max



```
Point                                                    Incr       Path
-----------------------------------------------------------------------
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (ideal)                              0.00       0.00
cs_registers_i/mie_q_reg_30_/CK (DFFR_X1)                0.00 #     0.00 r
cs_registers_i/mie_q_reg_30_/QN (DFFR_X1)                0.06       0.06 r
cs_registers_i/U666/ZN (OAI22_X1)                        0.01 *     0.07 f
cs_registers_i/mie_q_reg_30_/D (DFFR_X1)                 0.00 *     0.07 f
data arrival time                                                   0.07

clock clk_i (rise edge)                                  0.00       0.00
clock network delay (ideal)                              0.00       0.00
cs_registers_i/mie_q_reg_30_/CK (DFFR_X1)                0.00       0.00 r
library hold time                                        0.01       0.01
data required time                                                  0.01
-----------------------------------------------------------------------
data required time                                                  0.01
data arrival time                                                  -0.07
-----------------------------------------------------------------------
slack (MET)                                                         0.07
```

Figure 5.15: Placement Timing Report Min

5.1.5 Clock Tree Synthesis

The clock tree was created with a target skew of less than 0.5ns and max fanout of 10. We also used non default rules or NDR for the clock tree routing. We used double the spacing and the metal width for metals 3 to 6. This is done to ensure the quality of the clock tree as a golden signal. Following are the CTS routing (figure 5.16), the timing reports (figures 5.17 and 5.18) and congestion (figure 5.19) reports.

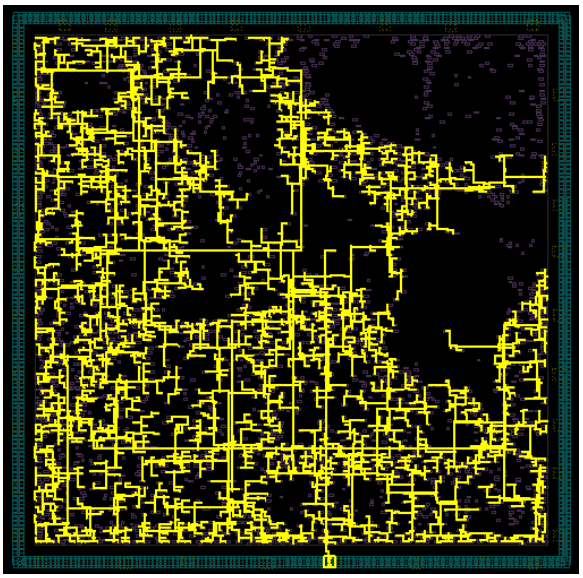


Figure 5.16: CTS Routing

```
Point                                                    Incr       Path
-----------------------------------------------------------------------
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (propagated)                         0.20       0.20
id_stage_i/register_file_i/R_5509/CK (DFFS_X2)           0.00       0.20 r
id_stage_i/register_file_i/R_5509/Q (DFFS_X2)            0.13       0.33 r
id_stage_i/register_file_i/U2260/ZN (OAI222_X1)          0.06 *     0.39 f
id_stage_i/register_file_i/U2620/ZN (AOI22_X1)           0.04 *     0.42 r
id_stage_i/register_file_i/U2655/ZN (AND4_X2)            0.06 *     0.48 r
id_stage_i/register_file_i/U2582/ZN (AND3_X4)            0.04 *     0.52 r
id_stage_i/register_file_i/U1891/ZN (NAND4_X4)           0.02 *     0.54 f
id_stage_i/register_file_i/rdata_a_o[16] (cv32e40p_register_file_ADDR_WIDT
                                                         0.00       0.54 f
id_stage_i/U126/ZN (INV_X4)                              0.03 *     0.57 r
id_stage_i/U1538/ZN (OAI22_X1)                           0.04 *     0.61 f
id_stage_i/DP_OP_483_142_4633_U126/ZN (NOR2_X2)          0.03 *     0.64 r
id_stage_i/DP_OP_483_142_4633_U114/ZN (NOR2_X2)          0.02 *     0.65 f
id_stage_i/DP_OP_483_142_4633_U98/ZN (NAND2_X1)          0.02 *     0.67 r
id_stage_i/U1525/ZN (INV_X1)                             0.01 *     0.69 f
id_stage_i/U1705/ZN (AND2_X1)                            0.03 *     0.72 f
id_stage_i/U463/ZN (NAND2_X1)                            0.01 *     0.74 r
id_stage_i/U2462/ZN (OAI21_X1)                           0.02 *     0.75 f
id_stage_i/U467/ZN (XNOR2_X2)                            0.04 *     0.79 f
id_stage_i/jump_target_o[23] (cv32e40p_id_stage_PULP_XPULP0 PULP_CLUSTER0_
                                                         0.00       0.79 f
U2583/ZN (NAND2_X2)                                      0.02 *     0.81 r
U1379/ZN (NAND3_X2)                                      0.03 *     0.84 f
U2582/ZN (OAI21_X2)                                      0.02 *     0.86 r
U2645/ZN (OAI211_X1)                                     0.03 *     0.88 f
instr_addr_o[23] (out)                                   0.00 *     0.88 f
data arrival time                                                   0.88

clock clk_i (rise edge)                                  2.20       2.20
clock network delay (propagated)                         0.00       2.20
output external delay                                   -1.32       0.88
data required time                                                  0.88
-----------------------------------------------------------------------
data required time                                                  0.88
data arrival time                                                  -0.88
-----------------------------------------------------------------------
slack (VIOLATED: increase significant digits)                       0.00
```

Figure 5.17: CTS Timing Report Max

```
Point                                                    Incr       Path
-----------------------------------------------------------------------
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (propagated)                         0.07       0.07
sleep_unit_i/fetch_enable_q_reg/CK (DFFR_X1)             0.00       0.07 r
sleep_unit_i/fetch_enable_q_reg/Q (DFFR_X1)              0.12       0.19 r
id_stage_i/fetch_enable_i (cv32e40p_id_stage_PULP_XPULP0 PULP_CLUSTER0 N
                                                         0.00       0.19 r
id_stage_i/U2166/ZN (AOI21_X1)                           0.03 *     0.21 f
id_stage_i/U1007/ZN (OAI211_X1)                          0.02 *     0.24 r
id_stage_i/controller_i/ctrl_fsm_cs_reg_0_/D (DFFR_X1)
                                                         0.00 *     0.24 r
data arrival time                                                   0.24

clock clk_i (rise edge)                                  0.00       0.00
clock network delay (propagated)                         0.20       0.20
id_stage_i/controller_i/ctrl_fsm_cs_reg_0_/CK (DFFR_X1)
                                                         0.00       0.20 r
library hold time                                        0.02       0.21
data required time                                                  0.21
-----------------------------------------------------------------------
data required time                                                  0.21
data arrival time                                                  -0.24
-----------------------------------------------------------------------
slack (MET)                                                         0.02
```

Figure 5.18: CTS Timing Report Min

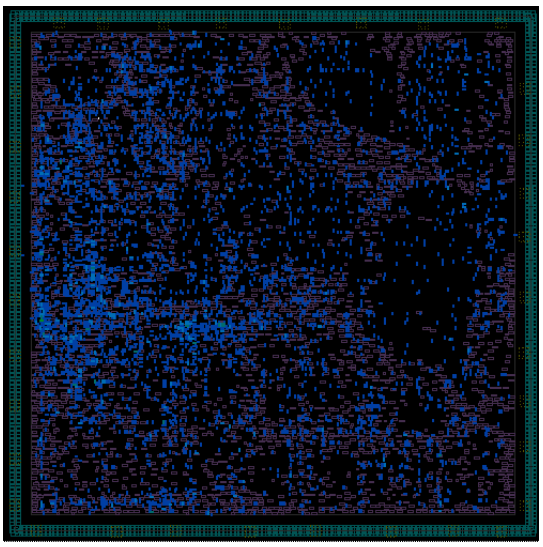


Figure 5.19: CTS Congestion Map

5.1.6 Routing

The routing was done using the commands previously discussed in chapter 5 with the goal of achieving the best timing while maintaining a relatively low DRC and LVS violation count. The following figures show the final routing results (figure 5.20) and the DRC (figure 5.21) and LVS (figure 5.22) reports.

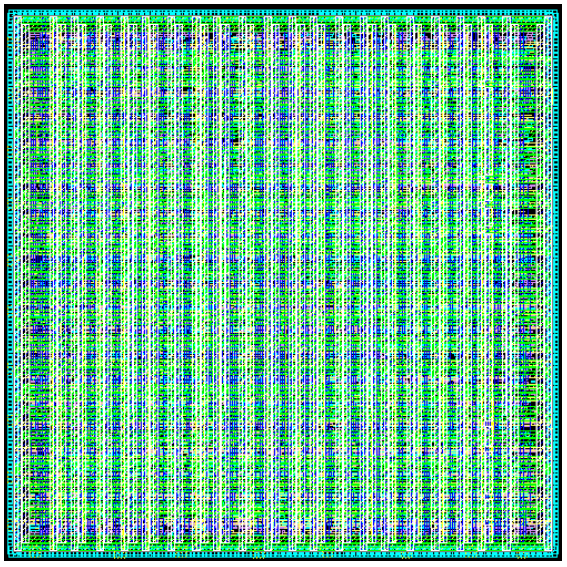


Figure 5.20: Routing

```
Verify Summary:

Total number of nets = 21800, of which 0 are not extracted
Total number of open nets = 0, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 7
Total number of antenna violations = no antenna rules defined
Total number of voltage-area violations = no voltage-areas defined
Total number of tie to rail violations = not checked
Total number of tie to rail directly violations = not checked
```

Figure 5.21: DRC Report

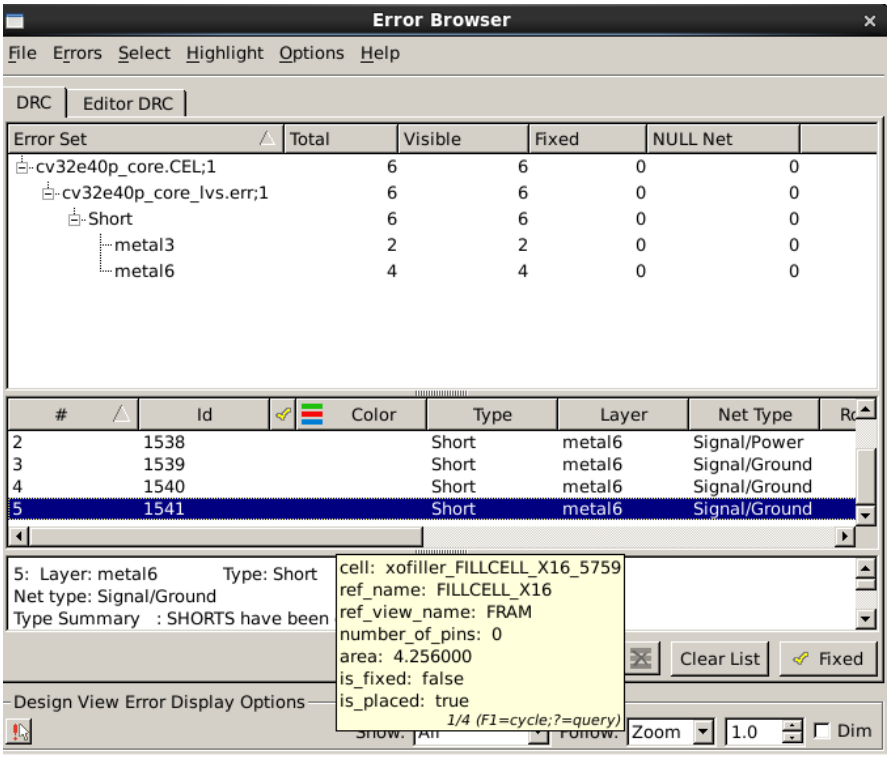


Figure 5.22: LVS Report

5.1.7 Prime Time

We ran the PrimeTime with the fast-fast corner and the min parasitic to check for the design hold violations. After going through the process discussed in detail in chapter 5 section 8, our design was able to pass the hold violation test with the worst path having a positive slack of 0.04 ns as shown in the following timing report in figure 5.23.

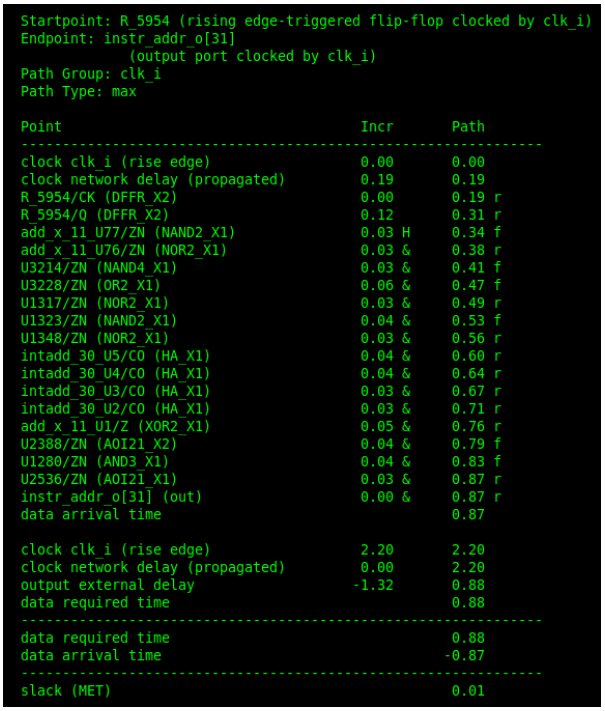
```
Startpoint: if_stage_i_prefetch_buffer_i_fifo_i_mem_q_reg_1__24_
            (rising edge-triggered flip-flop clocked by clk_i)
Endpoint: if_stage_i_prefetch_buffer_i_fifo_i_mem_q_reg_1__24_
            (rising edge-triggered flip-flop clocked by clk_i)
Path Group: clk_i
Path Type: min

Point                                                   Incr       Path
-----------------------------------------------------------------------------
clock clk_i (rise edge)                                 0.00       0.00
clock network delay (propagated)                        0.12       0.12
if_stage_i_prefetch_buffer_i_fifo_i_mem_q_reg_1__24_/CK (DFFR_X1)
                                                        0.00       0.12 r
if_stage_i_prefetch_buffer_i_fifo_i_mem_q_reg_1__24_/QN (DFFR_X1)
                                                        0.03 &     0.15 f
U3120/ZN (AOI22_X1)                                     0.01 &     0.16 r
if_stage_i_prefetch_buffer_i_fifo_i_mem_q_reg_1__24_/D (DFFR_X1)
                                                        0.00 &     0.16 r
data arrival time                                                  0.16

clock clk_i (rise edge)                                 0.00       0.00
clock network delay (propagated)                        0.12       0.12
if_stage_i_prefetch_buffer_i_fifo_i_mem_q_reg_1__24_/CK (DFFR_X1)
                                                                   0.12 r
library hold time                                       0.00       0.12
data required time                                                 0.12
-----------------------------------------------------------------------------
data required time                                                 0.12
data arrival time                                                 -0.16
-----------------------------------------------------------------------------
slack (MET)                                                        0.04
```

Figure 5.23: PrimeTime Timing Report Min

We were also able to obtain a clean setup report when using the slow-slow corner with the max parasitic. The following timing report in figure 5.24 shows that the timing was met with 0.01ns positive slack.



```
Startpoint: R_5954 (rising edge-triggered flip-flop clocked by clk_i)
Endpoint: instr_addr_o[31]
          (output port clocked by clk_i)
Path Group: clk_i
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock clk_i (rise edge)                  0.00       0.00
clock network delay (propagated)         0.19       0.19
R_5954/CK (DFFR_X2)                      0.00       0.19 r
R_5954/Q (DFFR_X2)                       0.12       0.31 r
add_x_11_U77/ZN (NAND2_X1)               0.03 H     0.34 f
add_x_11_U76/ZN (NOR2_X1)                0.03 &     0.38 r
U3214/ZN (NAND4_X1)                      0.03 &     0.41 f
U3228/ZN (OR2_X1)                        0.06 &     0.47 f
U1317/ZN (NOR2_X1)                       0.03 &     0.49 r
U1323/ZN (NAND2_X1)                      0.04 &     0.53 f
U1348/ZN (NOR2_X1)                       0.03 &     0.56 r
intadd_30_U5/CO (HA_X1)                  0.04 &     0.60 r
intadd_30_U4/CO (HA_X1)                  0.04 &     0.64 r
intadd_30_U3/CO (HA_X1)                  0.03 &     0.67 r
intadd_30_U2/CO (HA_X1)                  0.03 &     0.71 r
add_x_11_U1/Z (XOR2_X1)                  0.05 &     0.76 r
U2388/ZN (AOI21_X2)                      0.04 &     0.79 f
U1280/ZN (AND3_X1)                       0.04 &     0.83 f
U2536/ZN (AOI21_X1)                      0.03 &     0.87 r
instr_addr_o[31] (out)                   0.00 &     0.87 r
data arrival time                                   0.87

clock clk_i (rise edge)                  2.20       2.20
clock network delay (propagated)         0.00       2.20
output external delay                   -1.32       0.88
data required time                                  0.88
-----------------------------------------------------------
data required time                                  0.88
data arrival time                                  -0.87
-----------------------------------------------------------
slack (MET)                                         0.01
```

Figure 5.24: PrimeTime Timing Report Max

5.2 Hierarchical Flow

5.2.1 Synthesis

After the flow we have gone through in the previous chapter, Power, area, and Timing Results have been outputted.

For Power, The synthesis achieved total power of 139uW

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	95.3726	193.2482	883.1340	289.5040	(20.76%)	
register	635.9032	26.1949	4.1490e+04	703.5873	(50.46%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	130.8300	173.8844	9.6553e+04	401.2668	(28.78%)	
Total	862.1058 uW	393.3275 uW	1.3893e+05 nW	1.3944e+03 uW		

```
Total Dynamic Power    =   1.2554 mW  (100%)

Cell Leakage Power     = 138.9265 uW
```

Figure 5.25: Hierarchical Synthesis Power Report

While area of 29374 has been achieved

```
Combinational area:       17226.958147
Buf/Inv area:             1539.608001
Noncombinational area:    12147.954377
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          29374.912523
Total area:                 undefined
```

Figure 5.26: Hierarchical Synthesis Area Report

This was under the constraints mentioned above and most importantly with clock 2.61 which means the processor runs with 383 MHz.

```
clock clk_i (rise edge)                                  2.61       2.61
clock network delay (ideal)                              0.00       2.61
output external delay                                   -1.57       1.04
data required time                                                  1.04
-------------------------------------------------------------------------
data required time                                                  1.04
data arrival time                                                  -1.04
-------------------------------------------------------------------------
slack (MET)                                                         0.00
```

Figure 5.27: Hierarchical Synthesis Timing Report

5.2.2 Floorplan

In the floorplan stage, Where the cells are placed initially, The difference between Hierarchical and Flat appears when viewing the chip. It makes it obvious that the Hierarchical flow preserves the hierarchy of the design where each module is given a certain space unlike the flat flow.

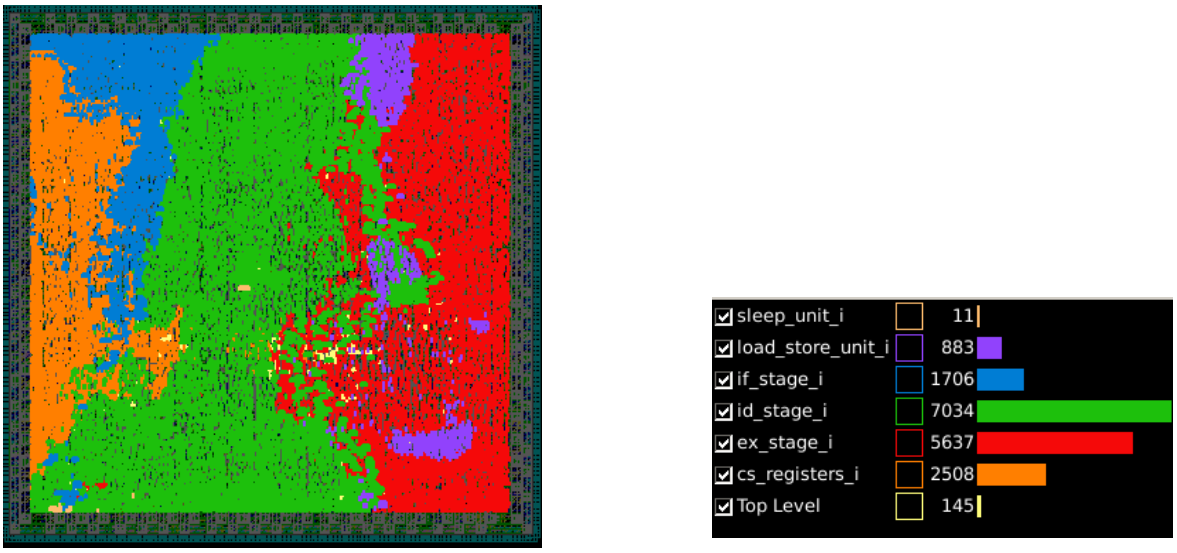


Figure 5.28: Hierarchical Hierarchy Graph

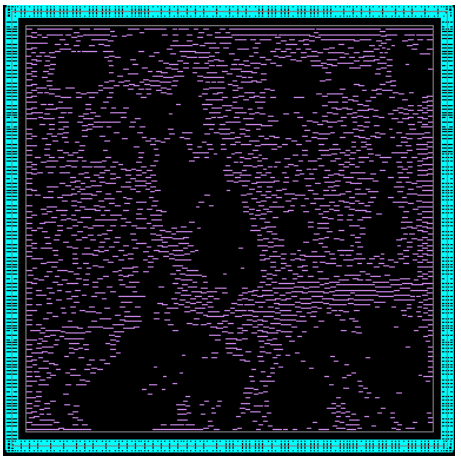


Figure 5.29: Hierarchical Floorplan

```
clock clk_i (rise edge)                                  2.61       2.61
clock network delay (ideal)                              0.00       2.61
output external delay                                   -1.57       1.04
data required time                                                  1.04
-------------------------------------------------------------------------
data required time                                                  1.04
data arrival time                                                  -1.04
-------------------------------------------------------------------------
slack (MET)                                                         0.00
```

Figure 5.30: Hierarchical Floorplan Setup Timing Report

```
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (ideal)                              0.00       0.00
id_stage_i/regfile_alu_waddr_ex_o_reg_3_/CK (DFFR_X1)
                                                         0.00       0.00 r
library hold time                                        0.01       0.01
data required time                                                  0.01
-------------------------------------------------------------------------
data required time                                                  0.01
data arrival time                                                  -0.09
-------------------------------------------------------------------------
slack (MET)                                                         0.08
```

Figure 5.31: Hierarchical Floorplan Hold Timing Report

5.2.3 Power Network

In the Power stage, It is important to analyze two important factors to modify the power constraints and layers which are IR Drop and Electromigration. For our design, We have reached Max IR drop 9.926 mV and Maximum wire EM: 8.167e+01 A/cm, layer metal9 and Maximum Via EM: 6.965e+05 A/cm_square, layer via5.

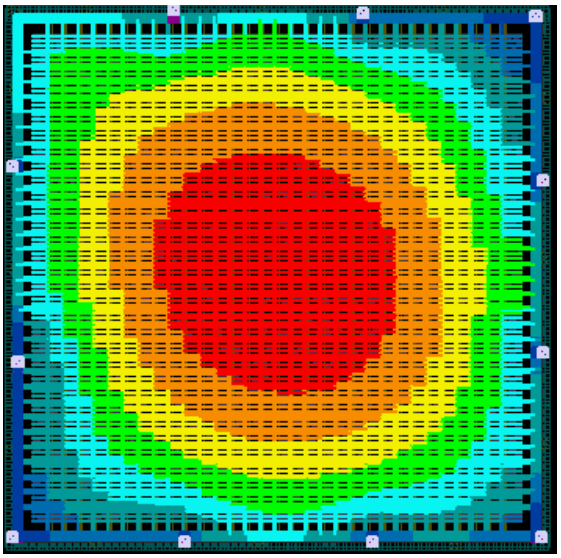


Figure 5.32: Hierarchical Power Drop

5.2.4 Placement

In the Placement stage, The goal is to reach a good congestion map with minimizing the hot spots. It is very important so we can minimize the DRC and LVS errors and also the routing runtime.

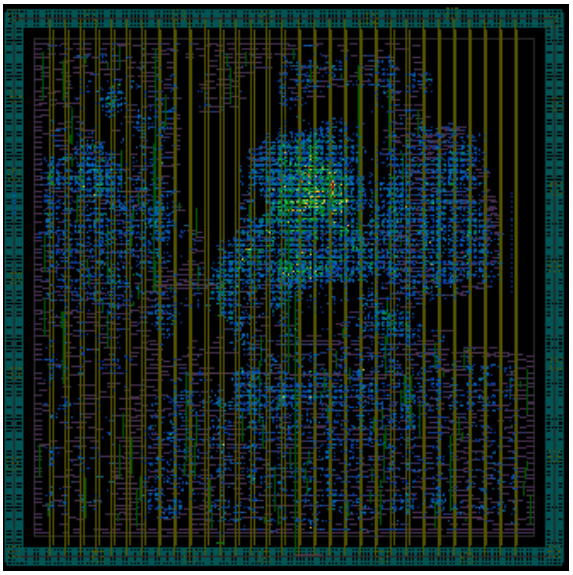


Figure 5.33: Hierarchical GRC

```
if_stage_i/prefetch_buffer_i/instruction_obi_i/trans_addr_i[4] (cv32e40p_obi_interface_TRANS_STABLE0)
                                                  0.00       1.26 r
if_stage_i/prefetch_buffer_i/instruction_obi_i/U97/ZN (INV_X1)
                                                  0.01 *     1.27 f
if_stage_i/prefetch_buffer_i/instruction_obi_i/U74/ZN (AOI21_X1)
                                                  0.05 *     1.32 r
if_stage_i/prefetch_buffer_i/instruction_obi_i/obi_addr_o[4] (cv32e40p_obi_interface_TRANS_STABLE0)
                                                  0.00       1.32 r
if_stage_i/prefetch_buffer_i/instr_addr_o[4] (cv32e40p_prefetch_buffer_PULP_OBI0_PULP_XPULP0)
                                                  0.00       1.32 r
if_stage_i/instr_addr_o[4] (cv32e40p_if_stage_PULP_XPULP0_PULP_OBI0_PULP_SECURE0_FPU0)
                                                  0.00       1.32 r
instr_addr_o[4] (out)                             0.00 *     1.33 r
data arrival time                                            1.33

clock clk_i (rise edge)                           2.61       2.61
clock network delay (ideal)                       0.00       2.61
output external delay                            -1.57       1.04
data required time                                           1.04
---------------------------------------------------------------------------
data required time                                           1.04
data arrival time                                           -1.33
---------------------------------------------------------------------------
slack (VIOLATED)                                            -0.28
```

Figure 5.34: Hierarchical Placement Setup Timing Report

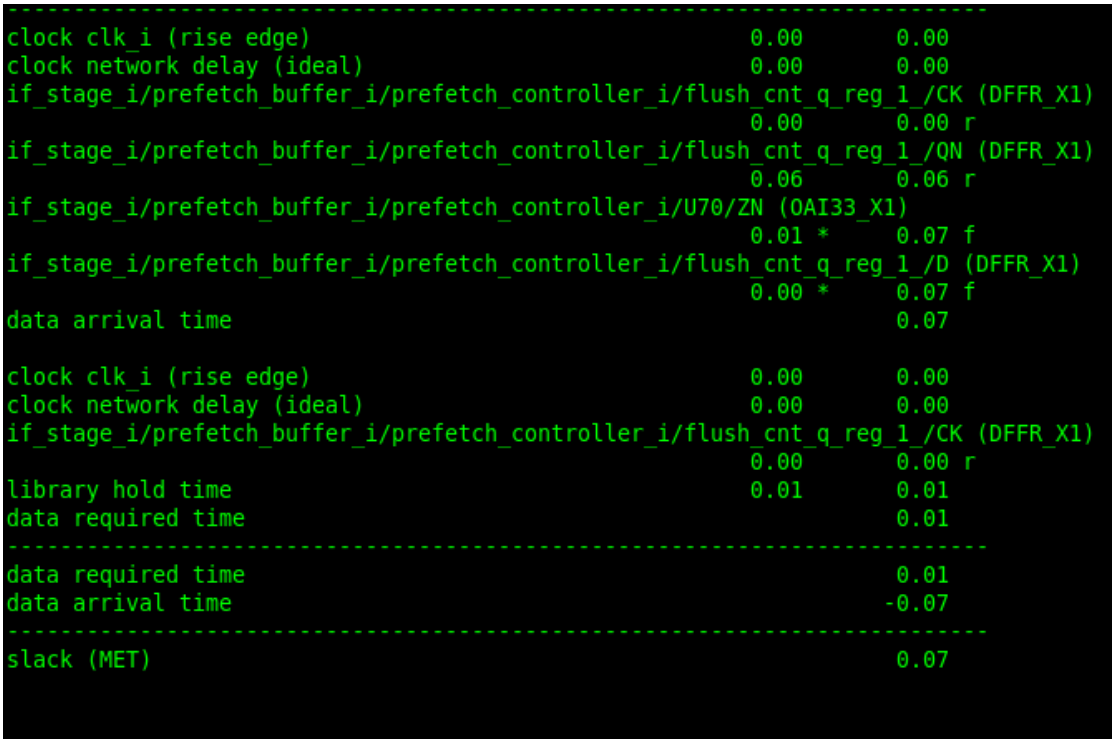


Figure 5.35: Hierarchical Placement Hold Timing Report

5.2.5 Clock Tree Synthesis

The clock tree was created with a target skew of less than 0.5ns and max fanout of 10. We also used non default rules or NDR for the clock tree routing. We used double the spacing and the metal width for metals 3 to 6. This is done to ensure the quality of the clock tree as a golden signal. Following are the CTS routing and the timing reports and congestion reports. We also analyzed the setup and hold to consider fixing them while creating the tree.

Figure 5.36: Hierarchical Clock Tree

```
clock clk_i (rise edge)                                  2.61       2.61
clock network delay (propagated)                         0.00       2.61
output external delay                                   -1.57       1.04
data required time                                                  1.04
-------------------------------------------------------------------------
data required time                                                  1.04
data arrival time                                                  -1.04
-------------------------------------------------------------------------
slack (MET)                                                         0.00
```

Figure 5.37: Hierarchical CTS Setup Timing Report

```
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (propagated)                         0.24       0.24
if_stage_i/prefetch_buffer_i/fifo_i/mem_q_reg_1__23_/CK (DFFR_X1)
                                                         0.00       0.24 r
library hold time                                        0.01       0.25
data required time                                                  0.25
-------------------------------------------------------------------------
data required time                                                  0.25
data arrival time                                                  -0.26
-------------------------------------------------------------------------
slack (MET)                                                         0.01
```

Figure 5.38: Hierarchical CTS Hold Timing Report

5.2.6 Routing

In the final stage, The design is routed and the congestion got a little better with our repeating commands.

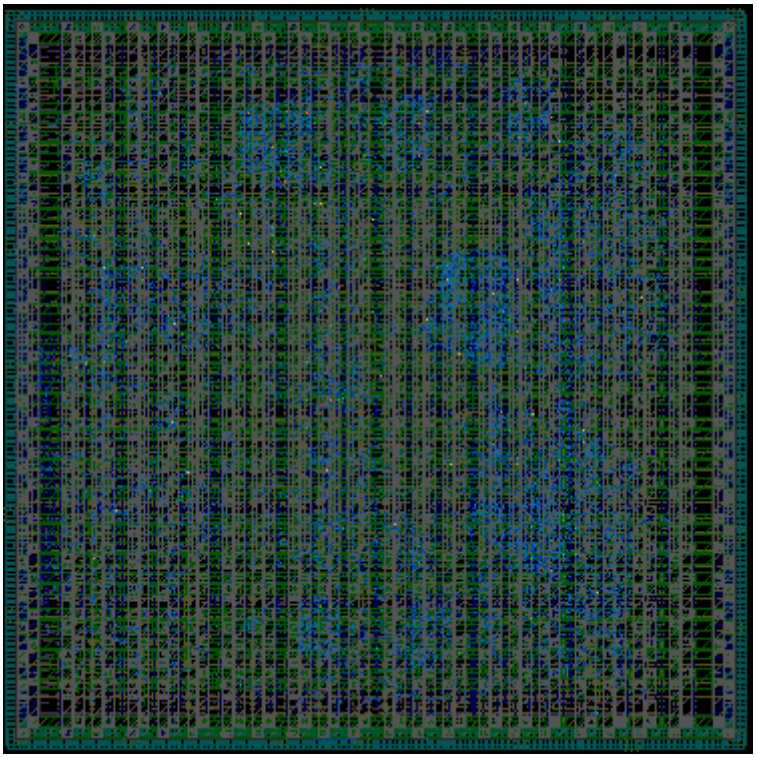


Figure 5.39: Hierarchical Routing Congestion

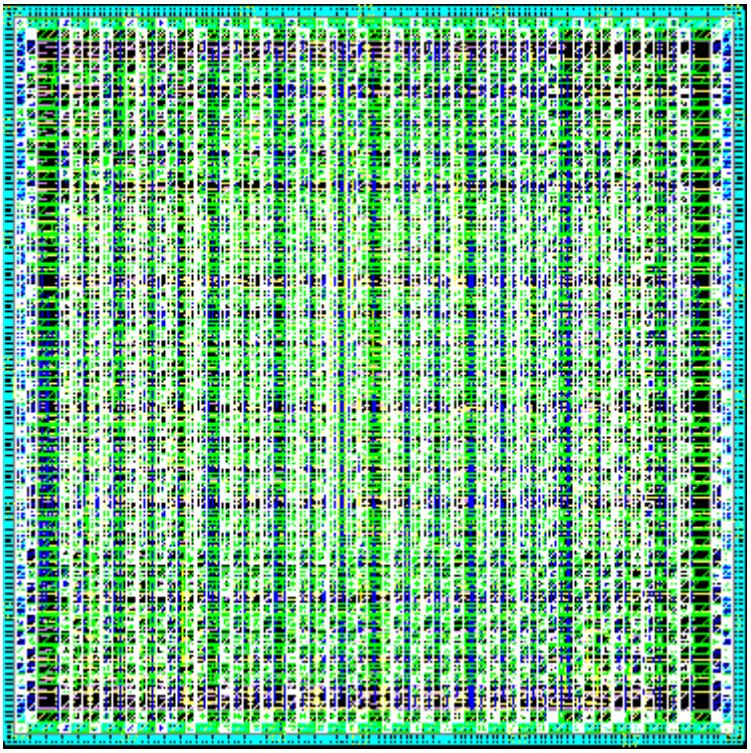


Figure 5.40: Hierarchical Routing

The main important results of routing are DRC and LVS reports. For DRC, We have reached 180 errors. For LVS, We have reached 132 errors

cv32e40p_core_1_imported_lvs.err;1	132
Min Area	67
Open	2
Open Locator	2
Short	61

Figure 5.41: Hierarchical LVS Report

5.2.7 Primetime

PrimeTime is the industry standard for static timing analysis. We used it to calculate exactly the figures of setup and hold and check if there are any violations. We used the fast-fast corner and the min parasitic to check for the design hold violations. But for the setup time analysis, We used slow slow corner and the max parasitic. After going through the process discussed in detail in chapter 5 section 8, our design was able to pass the hold violation test with the worst path having a positive slack of 0.01 ns for hold and setup as shown in the following timing reports

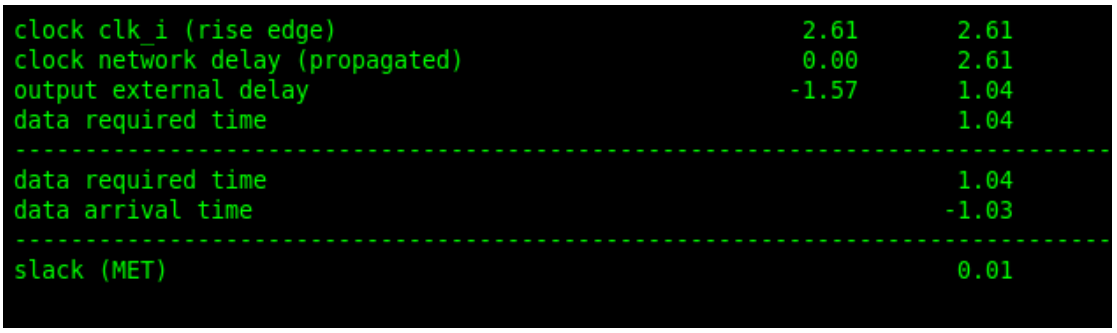


Figure 5.42: Hierarchical PrimeTime Setup Timing Report

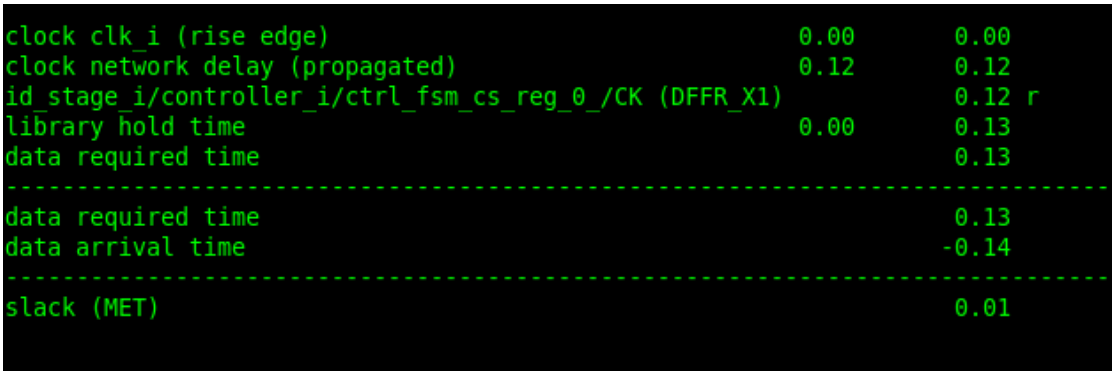


Figure 5.43: Hierarchical PrimeTime Hold Timing Report

5.3 Topographical Flow.

5.3.1 Synthesis (First Pass)

The first pass of synthesis is done to generate an initial netlist of the design in order to be used by the ICC to estimate the parasitics and timing. Thus, the results of this pass are an initial representation and may not show the actual characteristics of the design. The slow-slow (SS) corner is used in this step. The topographical flow is synthesized with hierarchical flavor.

Clocking gating is used in order to optimize the power of the design. The DC tool automatically inserts clock gating cells by using *-gate_clock* option with the *compile_ultra* command. *-timing_high_effort* option is used also in order to optimize the timing of the design.

The total area used by the cells in the design is about 0.028 mm2. It is noted that the area of the interconnects is not yet defined as the synthesis tool uses a wire load model to estimate the parasitics of the interconnects which does not have an estimation of the total interconnect area. The following figure (Figure 5.) shows a summary of the area used and the distribution of it among components.

```
Number of ports:                         2491
Number of nets:                         17604
Number of cells:                        16030
Number of combinational cells:          13742
Number of sequential cells:              2206
Number of macros/black boxes:               0
Number of buf/inv:                       2242
Number of references:                      91

Combinational area:              16525.250147
Buf/Inv area:                     1274.406008
Noncombinational area:           11641.224367
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 28166.474514
```

Figure 5.44: Area Summary

Power consumption is mainly due to dynamic power (represented in cells switching power and nets internal power) and leakage power. The total dynamic power consumed by the design is 1.41 mW. The leakage power of the cells is 0.13 mW. The below figure (Figure 5.) shows the power consumption distribution of the design.

Cell	Cell Internal Power (uW)	Driven Net Switching Power (uW)	Tot Dynamic Power (uW) (% Cell/Tot)	Cell Leakage Power (nW)
Netlist Power	962.2635	447.5752	1.410e+03 (68%)	1.332e+05
Estimated Clock Tree Power	N/A	N/A	(N/A)	N/A

Figure 5.45: Power Consumption Summary

The below figure (Figure 5.) shows how power is consumed among the various power groups of the design.

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	145.5035	150.7781	955.0328	297.2365	(19.26%)	
register	715.8830	25.9415	3.6688e+04	778.5107	(50.45%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	100.8785	270.8540	9.5597e+04	467.3309	(30.29%)	
Total	962.2650 uW	447.5735 uW	1.3324e+05 nW	1.5431e+03 uW		

Figure 5.46: Power Consumption Groups

The synthesis tool constraints are tightened in order to achieve the best possible frequency. The design could reach a clock period of 2 ns while not having any violations. Setup time is more important to check at this stage as hold time gets better in the PnR stage. The below figure (Figure 5.) shows one of the critical paths of the design that meets the setup requirements.

```
Startpoint: if_stage_i/instr_rdata_id_o_reg[19]
            (rising edge-triggered flip-flop clocked by clk_i)
Endpoint: instr_addr_o[11]
          (output port clocked by clk_i)
Path Group: clk_i
Path Type: max

Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (ideal)                              0.00       0.00
if_stage_i/instr_rdata_id_o_reg[19]/CK (DFFR_X1)         0.00       0.00 r
if_stage_i/instr_rdata_id_o_reg[19]/Q (DFFR_X1)          0.13       0.13 r
id_stage_i/register_file_i/raddr_a_i[4] (cv32e40p_register_file_ADDR_WIDTH6_DATA_WIDTH32_FPU0_PULP_ZFINX0)
                                                         0.00       0.13 r
id_stage_i/register_file_i/U3/ZN (NOR2_X1)               0.03 *     0.16 f
id_stage_i/register_file_i/U35/ZN (NAND2_X1)             0.05 *     0.21 r
id_stage_i/register_file_i/U80/ZN (NOR2_X4)              0.04 *     0.25 f
id_stage_i/register_file_i/U132/ZN (AOI22_X1)            0.04 *     0.29 r
id_stage_i/register_file_i/U133/ZN (NAND4_X1)            0.04 *     0.32 f
id_stage_i/register_file_i/U153/ZN (NOR3_X1)             0.03 *     0.35 r
id_stage_i/register_file_i/U179/ZN (NAND4_X1)            0.04 *     0.39 f
id_stage_i/register_file_i/rdata_a_o[3] (cv32e40p_register_file_ADDR_WIDTH6_DATA_WIDTH32_FPU0_PULP_ZFINX0)
                                                           0.00       0.39 f
U4995/ZN (INV_X1)                                          0.02 *     0.41 r
U4997/ZN (OAI22_X1)                                        0.02 *     0.43 f
U5001/ZN (NOR2_X1)                                         0.03 *     0.47 r
U5005/ZN (OAI21_X1)                                        0.03 *     0.49 f
U5006/ZN (AOI21_X2)                                        0.04 *     0.53 r
U5038/ZN (OAI21_X4)                                        0.02 *     0.56 f
U6168/ZN (AOI21_X1)                                        0.03 *     0.59 r
U6315/ZN (OAI21_X1)                                        0.02 *     0.61 f
U6318/ZN (XNOR2_X1)                                        0.04 *     0.65 f
U6319/ZN (AOI22_X1)                                        0.04 *     0.69 r
U6324/ZN (NAND3_X1)                                        0.03 *     0.71 f
U6325/ZN (OAI21_X1)                                        0.03 *     0.75 r
U6333/ZN (NAND2_X1)                                        0.02 *     0.77 f
U6335/ZN (NAND2_X1)                                        0.02 *     0.78 r
U6337/ZN (NAND2_X1)                                        0.01 *     0.80 f
instr_addr_o[11] (out)                                     0.00 *     0.80 f
data arrival time                                                     0.80

clock clk_i (rise edge)                                    2.00       2.00
clock network delay (ideal)                                0.00       2.00
output external delay                                     -1.20       0.80
data required time                                                    0.80
-----------------------------------------------------------------------------
data required time                                                    0.80
data arrival time                                                    -0.80
-----------------------------------------------------------------------------
slack (MET)                                                           0.00
```

Figure 5.47: Setup Time Critical Path

5.3.2 Floorplan

The below table (Table 5.1) shows the parameters used in the floorplan. The first row of cells starts from the bottom and it is flipped which is a rule of thumb.

Table 5.1: Floorplan Parameters

Aspect Ratio	1
Core Utilization	0.25
IO to Core Clearance	12.4

The below graphs (Figure 5.48 and Figure 5.49) show the design hierarchy of the floorplan.

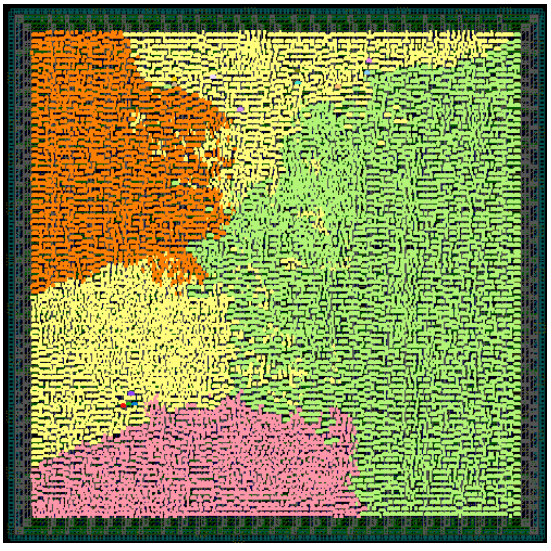


Figure 5.48 : Design Hierarchy

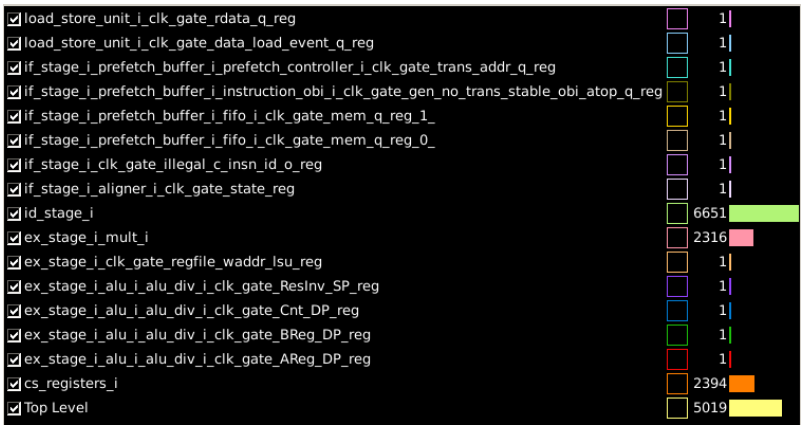


Figure 5.49: Design Modules

5.3.3 Power Network

The parameters used in the topographical flow are specified in the below table (Table 5.2).

Table 5.2: Power Network Parameters

Voltage supply	1.1
Max IR mV	22
Power budget	500
Min width	2.5
Ring spacing	0.8
Ring width	5
Min/max strap	20/128

The most important factors in the power network are the IR drop and electromigration. The maximum IR drop in the design is 7.916 mV. The maximum wire electromigration is 74.9 A/cm. The maximum via electromigration is 6.50186 A/cm2. The following graphs show the power network (Figure 5.), the congestion map after the power network (Figure 5.), and the power summary of VDD and VSS signals (Figure 5. and Figure 5.).

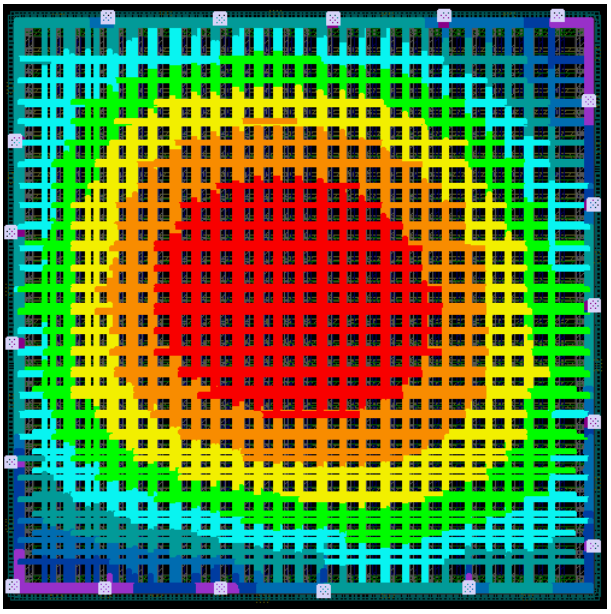


Figure 5.50: Power Network

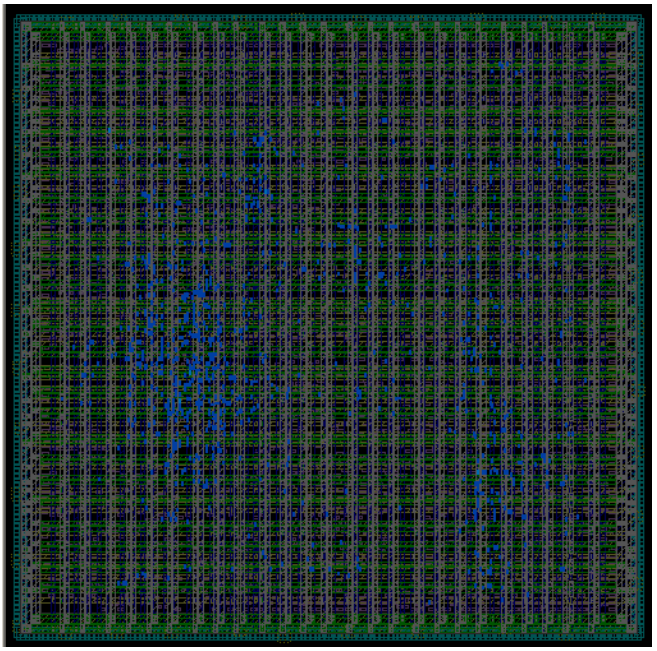


Figure 5.51: Congestion Map after Power Network

```
Net Name: VSS
Number of Pad Cells: 0
Number of Pad Cells Supplying Current: 0
Number of Virtual Pad Nodes: 20
Number of Virtual Pad Nodes Supplying Current: 20
Number of Net Wires: 554
Number of Net Vias: 2848
Number of Extracted Resistors: 20176
Number of Extracted Nodes: 15061
Current from Pad Cells:    0.00 mA (0.00%)
Current from Virtual Pad Nodes:  151.53 mA (100.00%)
Maximum IR drop:   2.438 mV at (172.7600 184.4300) (182.7700 186.9300) layer metal7
Maximum Wire EM: 1.997010e+01 A/cm at (232.5630 1.3360) (237.5630 9.1000) layer metal10
Maximum Via EM: 1.733515e+05 A/cm_2 at (232.5630 6.6000) (237.5630 11.6000) layer via9
The percentage of routing tracks used by the power net for layer metal10: 44.39%
The percentage of routing tracks used by the power net for layer metal9: 46.71%
The percentage of routing tracks used by the power net for layer metal8: 69.56%
The percentage of routing tracks used by the power net for layer metal7: 68.03%
The percentage of routing tracks used by the power net for layer metal6: 0.00%
The percentage of routing tracks used by the power net for layer metal5: 0.00%
The percentage of routing tracks used by the power net for layer metal4: 0.00%
The percentage of routing tracks used by the power net for layer metal3: 0.00%
The percentage of routing tracks used by the power net for layer metal2: 0.00%
The percentage of routing tracks used by the power net for layer metal1: 13.03%
The average percentage of routing tracks used by net VSS : 24.17%
```

Figure 5.52: VSS Signal Power Summary

```
Net Name: VDD
Number of Pad Cells: 0
Number of Pad Cells Supplying Current: 0
Number of Virtual Pad Nodes: 19
Number of Virtual Pad Nodes Supplying Current: 20
Number of Net Wires: 795
Number of Net Vias: 3282
Number of Extracted Resistors: 24776
Number of Extracted Nodes: 17603
Current from Pad Cells:    0.00 mA (0.00%)
Current from Virtual Pad Nodes:  155.70 mA (100.00%)
Maximum IR drop:   2.224 mV at (165.1600 190.5100) (167.3100 193.0100) layer metal7
Maximum Wire EM: 2.874032e+01 A/cm at (284.6200 3.3000) (287.1200 3.6985) layer metal10
Maximum Via EM: 1.871115e+05 A/cm_2 at (284.6300 1.2200) (287.1100 5.3800) layer via9
The percentage of routing tracks used by the power net for layer metal10: 44.81%
The percentage of routing tracks used by the power net for layer metal9: 47.57%
The percentage of routing tracks used by the power net for layer metal8: 72.47%
The percentage of routing tracks used by the power net for layer metal7: 69.44%
The percentage of routing tracks used by the power net for layer metal6: 0.00%
The percentage of routing tracks used by the power net for layer metal5: 0.00%
The percentage of routing tracks used by the power net for layer metal4: 0.00%
The percentage of routing tracks used by the power net for layer metal3: 0.00%
The percentage of routing tracks used by the power net for layer metal2: 0.00%
The percentage of routing tracks used by the power net for layer metal1: 13.13%
The average percentage of routing tracks used by net VDD : 24.74%
```

Figure 5.53: VDD Signal Power Summary

5.3.4 Synthesis (Second Pass)

The second pass of synthesis is done after the floorplan and power mesh generation. It uses the (.def) file that is generated from the floorplan in order to optimize the netlist and has a more accurate estimation of the parasitics and delays. The output results of this pass are more representative of the actual merits of the design. The slow-slow (SS) corner is used in this step.

The total area used by the cells in the design is about 0.03 mm2. The area increased a little compared to the first pass. It is noted that the area of the interconnects is still not yet defined. The below figure (Figure 5.54) shows a summary of the area used and the distribution of it among components.

```
Number of ports:                         2180
Number of nets:                         19236
Number of cells:                        17911
Number of combinational cells:          15775
Number of sequential cells:              2131
Number of macros/black boxes:               0
Number of buf/inv:                       3587
Number of references:                      68

Combinational area:              19262.656155
Buf/Inv area:                     2048.466018
Noncombinational area:           11349.422367
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 30612.078522
```

Figure 5.54: Area Summary

Power consumption is mainly due to dynamic power (represented in cells switching power and nets internal power) and leakage power. The total dynamic power

consumed by the design is 4.9 mW. The leakage power of the cells is 0.15 mW. This is a notable increase compared to the first pass. The following figure shows the power consumption distribution of the design.

Cell	Cell Internal Power (uW)	Driven Net Switching Power (uW)	Tot Dynamic Power (uW) (% Cell/Tot)	Cell Leakage Power (nW)
Netlist Power	3.954e+03	960.6052	4.915e+03 (80%)	1.511e+05
Estimated Clock Tree Power	N/A	N/A	(N/A)	N/A

Figure 5.55: Power Consumption Summary

The following figure (Figure 5.56) shows how power is consumed among the various power groups of the design.

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	1.9832	578.6100	12.2814	580.6055	(11.46%)	
register	3.8220e+03	32.0838	3.7626e+04	3.8917e+03	(76.82%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	130.0761	349.9137	1.1349e+05	593.4804	(11.72%)	
Total	3.9541e+03 uW	960.6074 uW	1.5113e+05 nW	5.0658e+03 uW		

Figure 5.56: Power Consumption Groups

The timing is checked in order to know if the timing constraints are still met. The below figure (Figure 5.57) shows one of the critical paths of the design that meets the setup requirements.

```
Startpoint: if_stage_i/instr_rdata_id_o_reg_17_
            (rising edge-triggered flip-flop clocked by clk_i)
Endpoint: instr_addr_o[17]
          (output port clocked by clk_i)
Path Group: clk_i
Path Type: max

Point                                                                  Incr       Path
-----------------------------------------------------------------------------------------
clock clk_i (rise edge)                                                0.00       0.00
clock network delay (ideal)                                            0.00       0.00
if_stage_i/instr_rdata_id_o_reg_17_/CK (DFFR_X1)                       0.00 #     0.00 r
if_stage_i/instr_rdata_id_o_reg_17_/Q (DFFR_X1)                        0.10       0.10 r
id_stage_i/register_file_i/raddr_a_i[2] (cv32e40p_register_file_ADDR_WIDTH6_DATA_WIDTH32_FPU0_PULP_ZFINX0)
                                                                       0.00       0.10 r
id_stage_i/register_file_i/U10/ZN (INV_X1)                             0.02 *     0.12 f
id_stage_i/register_file_i/U21/ZN (NOR2_X2)                            0.02 *     0.14 r
id_stage_i/register_file_i/U22/ZN (NAND2_X1)                           0.03 *     0.17 f
id_stage_i/register_file_i/U143/ZN (INV_X1)                            0.02 *     0.19 r
id_stage_i/register_file_i/U191/ZN (INV_X1)                            0.02 *     0.21 f
id_stage_i/register_file_i/U49/ZN (NOR2_X4)                            0.08 *     0.28 r
id_stage_i/register_file_i/U2982/ZN (AOI22_X1)                         0.05 *     0.33 f
id_stage_i/register_file_i/U2986/ZN (NAND4_X1)                         0.03 *     0.35 r
id_stage_i/register_file_i/U2991/ZN (NOR4_X1)                          0.02 *     0.37 f
id_stage_i/register_file_i/U3002/ZN (NAND3_X2)                         0.02 *     0.39 r
id_stage_i/register_file_i/rdata_a_o[8] (cv32e40p_register_file_ADDR_WIDTH6_DATA_WIDTH32_FPU0_PULP_ZFINX0)
                                                                       0.00       0.39 r
U5193/ZN (AOI22_X1)                                                    0.02 *     0.41 f
U5194/ZN (INV_X1)                                                      0.02 *     0.43 r
U5202/ZN (NAND2_X1)                                                    0.02 *     0.45 f
U5204/ZN (OAI21_X1)                                                    0.04 *     0.49 r
U5208/ZN (AOI21_X2)                                                    0.02 *     0.52 f
U11416/ZN (OAI21_X2)                                                   0.03 *     0.55 r
U6987/ZN (AOI21_X1)                                                    0.02 *     0.57 f
U11531/ZN (INV_X1)                                                     0.02 *     0.59 r
U12711/ZN (AOI21_X1)                                                   0.02 *     0.61 f
U12714/Z (XOR2_X1)                                                     0.05 *     0.66 f
U13129/ZN (AOI21_X2)                                                   0.02 *     0.68 r
U13132/ZN (NAND3_X1)                                                   0.03 *     0.70 f
U13142/ZN (NOR3_X2)                                                    0.05 *     0.75 r
U13202/ZN (NOR2_X2)                                                    0.02 *     0.77 f
U13163/ZN (NAND2_X1)                                                   0.01 *     0.78 r
U13116/ZN (NAND2_X1)                                                   0.02 *     0.80 f
instr_addr_o[17] (out)                                                 0.00 *     0.80 f
data arrival time                                                                 0.80

clock clk_i (rise edge)                                                2.00       2.00
clock network delay (ideal)                                            0.00       2.00
output external delay                                                 -1.20       0.80
data required time                                                                0.80
-----------------------------------------------------------------------------------------
data required time                                                                0.80
data arrival time                                                                -0.80
-----------------------------------------------------------------------------------------
slack (MET)                                                                       0.00
```

Figure 5.57: Setup Time Critical Path

5.3.5 Placement

Placement is done after the second pass of synthesis. It uses the optimized floorplan and netlist generated by the synthesis tool. It is important that all the cells within the design are legally placed with good congestion. The design also should be routable and meet the timing requirements. The following figures show the congestion map (Figure 5.58) and timing reports of the design after placement (Figure 5.59 and Figure 5.60).

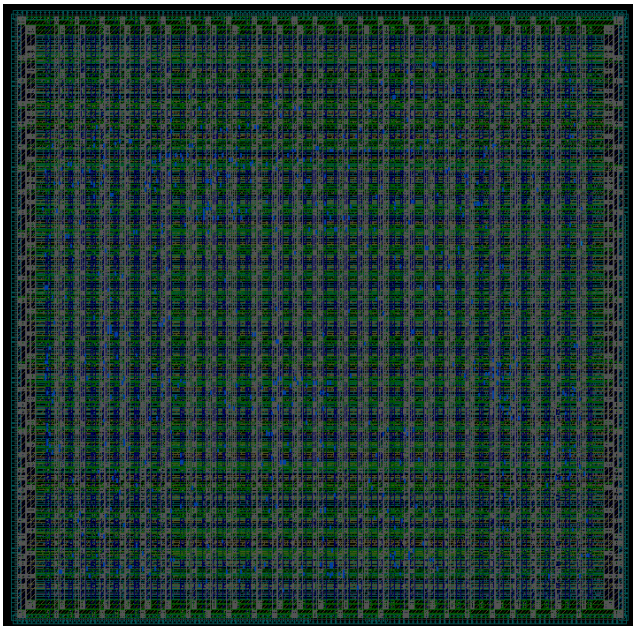


Figure 5.58: Congestion Map After Placement

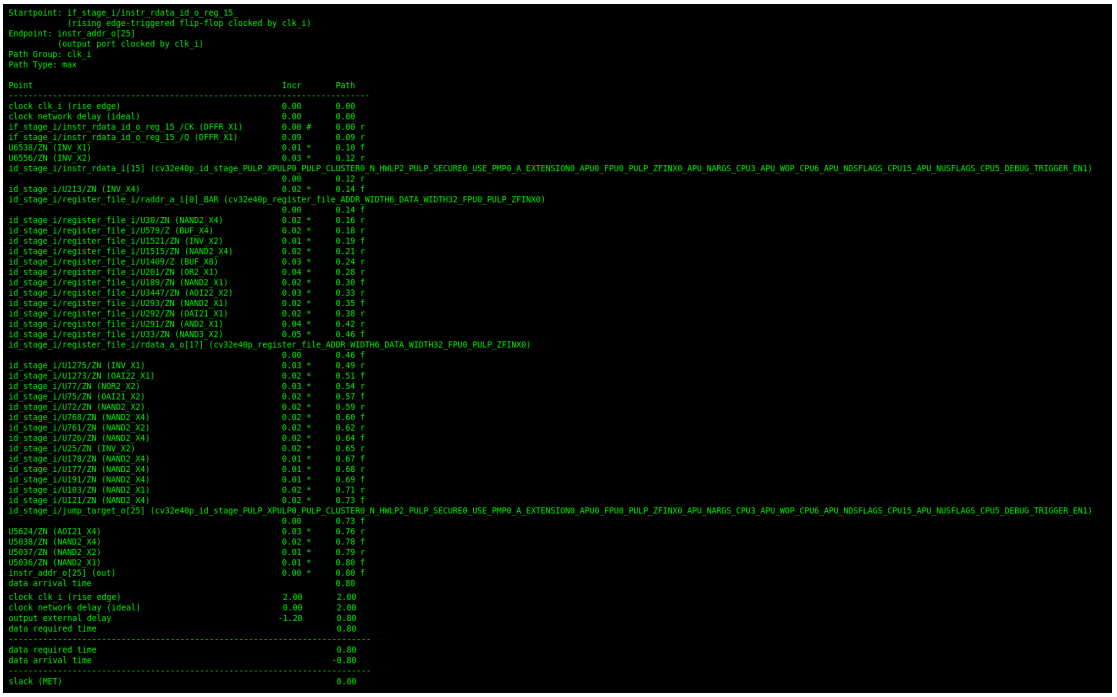


Figure 5.59: Setup Time Critical Path

```
Startpoint: ex_stage_i/alu_i/alu_div_i/ResReg_DP_reg_0_
            (rising edge-triggered flip-flop clocked by clk_i)
Endpoint: ex_stage_i/alu_i/alu_div_i/ResReg_DP_reg_0_
          (rising edge-triggered flip-flop clocked by clk_i)
Path Group: clk_i
Path Type: min

Point                                                   Incr       Path
-----------------------------------------------------------------------------
clock clk_i (rise edge)                                 0.00       0.00
clock network delay (ideal)                             0.00       0.00
ex_stage_i/alu_i/alu_div_i/ResReg_DP_reg_0_/CK (DFFR_X1)
                                                        0.00 #     0.00 r
ex_stage_i/alu_i/alu_div_i/ResReg_DP_reg_0_/QN (DFFR_X1)
                                                        0.06       0.06 r
U6084/ZN (OAI22_X1)                                     0.01 *     0.07 f
ex_stage_i/alu_i/alu_div_i/ResReg_DP_reg_0_/D (DFFR_X1)
                                                        0.00 *     0.07 f
data arrival time                                                  0.07

clock clk_i (rise edge)                                 0.00       0.00
clock network delay (ideal)                             0.00       0.00
ex_stage_i/alu_i/alu_div_i/ResReg_DP_reg_0_/CK (DFFR_X1)
                                                        0.00       0.00 r
library hold time                                       0.01       0.01
data required time                                                 0.01
-----------------------------------------------------------------------------
data required time                                                 0.01
data arrival time                                                 -0.07
-----------------------------------------------------------------------------
slack (MET)                                                        0.07
```

Figure 5.60: Hold Time Critical Path

5.3.6 Clock Tree Synthesis

The clock is a golden signal that should be connected to each flip flop in the design. The clock tree is generated in order to connect the signal from the clock pins to the cells while having minimum skew. It is also important to minimize the power as the clock is a major source of power consumption in ASIC designs. Signal integrity and clock routes congestion should be considered in order not to affect the design. The following figures show the congestion map (Figure 5.61), the clock tree (Figure 5.62), and the timing reports after CTS (Figure 5.63).

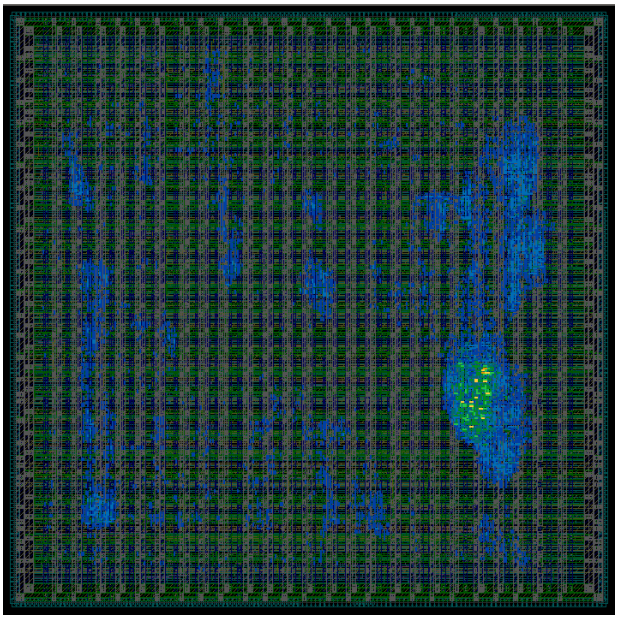


Figure 5.61: Congestion Map After CTS

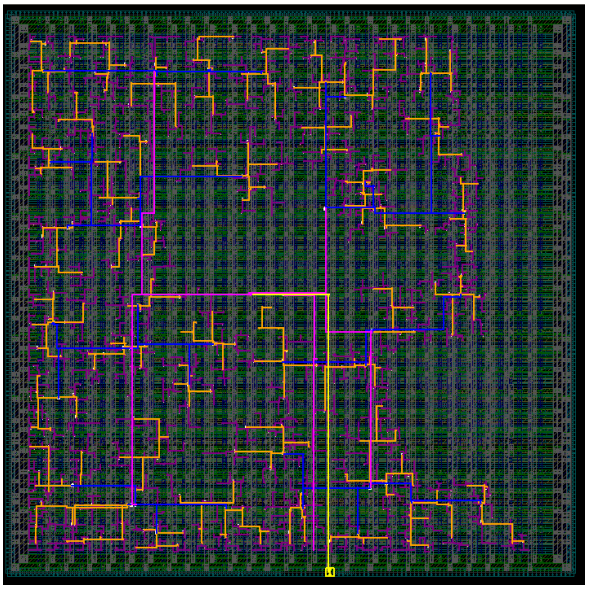


Figure 5.62: Clock Tree

```
Startpoint: id_stage_i/instr_rdata_id_o_reg_16_
            (rising edge-triggered flip-flop clocked by clk_i)
Endpoint: instr_addr_o[31]
          (output port clocked by clk_i)
Path Group: clk_i
Path Type: max

Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock clk_i (rise edge)                                  0.00       0.00
clock network delay (propagated)                         0.20       0.20
id_stage_i/instr_rdata_id_o_reg_16_/CK (DFFR_X1)         0.00       0.20 r
id_stage_i/instr_rdata_id_o_reg_16_/QN (DFFR_X1)         0.08       0.28 r
U9987/ZN (INV_X4)                                        0.02 *     0.30 f
id_stage_i/instr_rdata_i[16] (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_N_HWLP2_PULP_SECURE0_USE_PMP0_A_EXTENSION0_APU0_FPU0_PULP_ZFINX0_APU_NARGS_CPU3_APU_WOP_CPU6_APU_NDSFLAGS_CPU15_APU_NUSFLAGS_CPU5_DEBUG_TRIGGER_EN1)
                                                         0.00       0.30 f
id_stage_i/U214/ZN (INV_X8)                              0.02 *     0.32 r
id_stage_i/register_file_i/raddr_a_i[1]_BAR (cv32e40p_register_file_ADDR_WIDTH6_DATA_WIDTH32_FPU0_PULP_ZFINX0)
                                                         0.00       0.32 r
id_stage_i/register_file_i/U270/ZN (OR3_X2)               0.04 *     0.35 r
id_stage_i/register_file_i/U266/ZN (INV_X8)               0.02 *     0.37 f
id_stage_i/register_file_i/U465/Z (BUF_X16)               0.03 *     0.40 f
id_stage_i/register_file_i/U1502/ZN (AOI22_X1)            0.04 *     0.44 r
id_stage_i/register_file_i/U1498/ZN (NAND4_X1)            0.03 *     0.47 f
id_stage_i/register_file_i/U1494/ZN (AOI211_X1)           0.05 *     0.52 r
id_stage_i/register_file_i/U325/ZN (OAI22_X1)             0.03 *     0.55 f
id_stage_i/register_file_i/U2983/ZN (INV_X2)              0.02 *     0.57 r
id_stage_i/register_file_i/U1549/ZN (NAND2_X4)            0.02 *     0.59 f
id_stage_i/register_file_i/rdata_a_o[13] (cv32e40p_register_file_ADDR_WIDTH6_DATA_WIDTH32_FPU0_PULP_ZFINX0)
                                                         0.00       0.59 f
id_stage_i/U346/ZN (NAND2_X4)                            0.03 *     0.62 r
id_stage_i/U2635/ZN (NAND2_X4)                           0.01 *     0.63 f
id_stage_i/U342/ZN (NOR2_X2)                             0.02 *     0.65 r
id_stage_i/U297/ZN (OAI21_X2)                            0.02 *     0.67 f
id_stage_i/U2257/ZN (AOI21_X2)                           0.02 *     0.69 r
id_stage_i/U719/ZN (OAI21_X2)                            0.02 *     0.72 f
id_stage_i/U2649/ZN (INV_X4)                             0.02 *     0.73 r
id_stage_i/U309/ZN (NAND2_X4)                            0.01 *     0.75 f
id_stage_i/U775/ZN (NAND3_X2)                            0.02 *     0.76 r
id_stage_i/U224/ZN (NAND3_X4)                            0.02 *     0.79 f
id_stage_i/U1375/ZN (AOI21_X4)                           0.03 *     0.81 r
id_stage_i/U351/ZN (OAI21_X2)                            0.02 *     0.83 f
id_stage_i/U319/ZN (XNOR2_X2)                            0.04 *     0.87 f
id_stage_i/jump_target_o[31] (cv32e40p_id_stage_PULP_XPULP0_PULP_CLUSTER0_N_HWLP2_PULP_SECURE0_USE_PMP0_A_EXTENSION0_APU0_FPU0_PULP_ZFINX0_APU_NARGS_CPU3_APU_WOP_CPU6_APU_NDSFLAGS_CPU15_APU_NUSFLAGS_CPU5_DEBUG_TRIGGER_EN1)
                                                         0.00       0.87 f
U5206/ZN (NAND2_X2)                                      0.01 *     0.88 r
U5205/ZN (NAND2_X1)                                      0.01 *     0.89 f
instr_addr_o[31] (out)                                   0.00 *     0.89 f
data arrival time                                                   0.89

clock clk_i (rise edge)                                  2.00       2.00
clock network delay (propagated)                         0.00       2.00
output external delay                                   -1.20       0.80
data required time                                                  0.80
-----------------------------------------------------------------------------
data required time                                                  0.80
data arrival time                                                  -0.89
-----------------------------------------------------------------------------
slack (VIOLATED)                                                   -0.09
```

Figure 5.63: Setup Time Critical Path

```
Startpoint: instr_rdata_i[24]
            (input port clocked by clk_i)
Endpoint: if_stage_i/prefetch_buffer_i/fifo_i/mem_q_reg_0__24_
          (rising edge-triggered flip-flop clocked by clk_i)
Path Group: clk_i
Path Type: min

Point                                                   Incr       Path
-----------------------------------------------------------------------
clock clk_i (rise edge)                                 0.00       0.00
clock network delay (propagated)                        0.00       0.00
input external delay                                    0.20       0.20 r
instr_rdata_i[24] (in)                                  0.00       0.20 r
U9443/ZN (OAI22_X1)                                     0.01 *     0.21 f
U9444/ZN (INV_X1)                                       0.01 *     0.22 r
if_stage_i/prefetch_buffer_i/fifo_i/mem_q_reg_0__24_/D (DFFR_X1)
                                                        0.00 *     0.22 r
data arrival time                                                  0.22

clock clk_i (rise edge)                                 0.00       0.00
clock network delay (propagated)                        0.20       0.20
if_stage_i/prefetch_buffer_i/fifo_i/mem_q_reg_0__24_/CK (DFFR_X1)
                                                        0.00       0.20 r
library hold time                                       0.01       0.21
data required time                                                 0.21
-----------------------------------------------------------------------
data required time                                                 0.21
data arrival time                                                 -0.22
-----------------------------------------------------------------------
slack (MET)                                                        0.01
```

Figure 5.64: Hold Time Critical Path

5.3.7 Routing

The routing was done using the commands previously discussed in chapter 5 with the goal of achieving the best timing while maintaining a relatively low DRC and LVS violation count. The following figures show the final routing results (Figure 5.65), the congestion map (Figure 5.66), and the DRC (Figure 5.67), and LVS (Figure 5.68) reports.

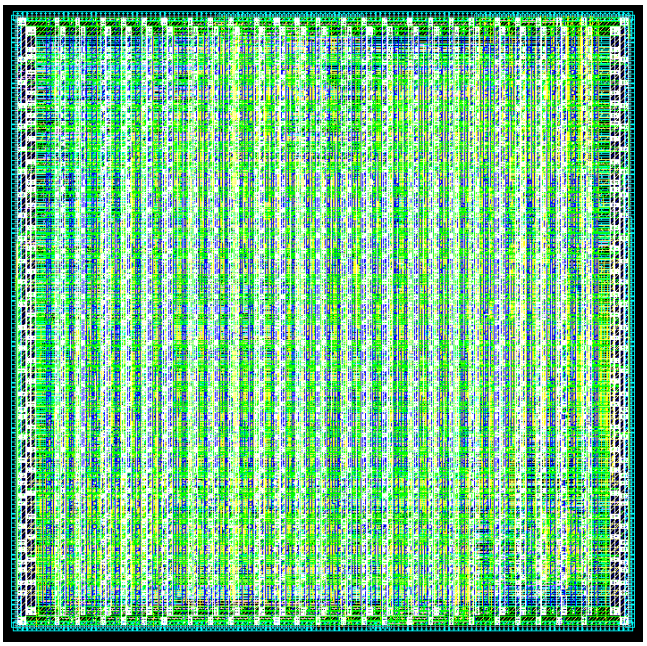


Figure 5.65: Routed Design

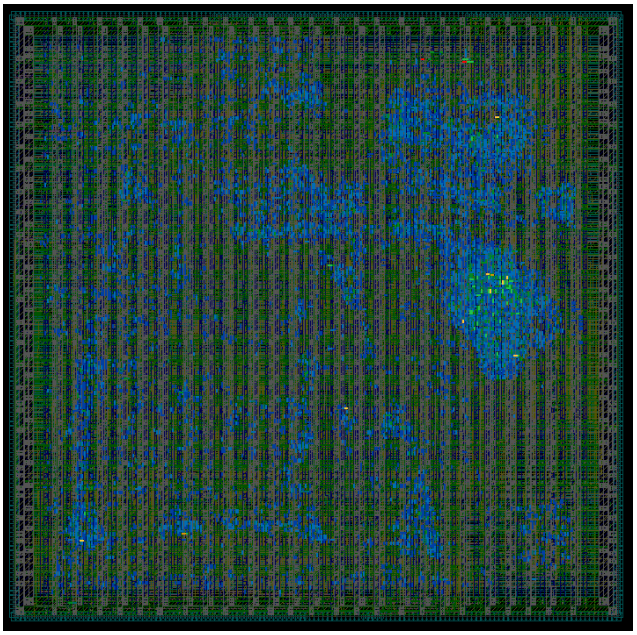


Figure 5.66: Congestion Map After Routing

```
Total number of nets = 22203
0 open nets, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = antenna checking not active
Total number of voltage-area violations = no voltage-areas defined
```

Figure 5.67: DRC Summary

Error Set	Total	Visible	Fixed	NULL Net
cv32e40p_core.CEL;2	225	225	0	223
cv32e40p_core_lvs.err;1	225	225	0	223
Open	2	2	0	0
Short	223	223	0	223

Figure 5.68: LVS Summary

5.3.8 PrimeTime

PrimeTime was run with the fast-fast corner and the min parasitic to check for the design hold violations. After going through the process discussed in detail in chapter 5 section 8, our design was able to pass the hold violation test with the worst path having a positive slack of 0.05 ns.

We were also able to obtain a clean setup report when using the slow-slow corner with the max parasitic. The timing report shows that the timing was met with 0.00 ns positive slack.

Chapter 6

Conclusion & Future Work

6.1 Flows Comparison & Conclusion

After finishing the three flows, they are compared in order to understand the advantages and disadvantages of each flow. Table 6.1 shows a comparison of the three flows after synthesis in terms of runtime, total cell area, power consumption, and clock period.

Table 6.1: Post-Synthesis Results

	Hierarchical	Flat	Topographical (pass1)	Topographical (pass2)
Runtime (mins)	2	5	2	3
Area	29374	33588.6	28166.47	29691.45
Dynamic Power(mW)	1.2554	1.201	1.41	1.51
Leakage Power(mW)	0.138	0.14	0.13	0.15

Total Power (mW)	1.39	1.341	1.54	1.66
Clock (ns)	2.61	2.2	2	2

As shown in the above table, the runtime is the worst in the flat flow. This is reasonable as it tolerates the design as one huge module which complicates the analysis. However, it should be noted that the synthesis runs twice in topographical mode. This makes the total runtime of the topographical flow almost the same as the flat flow.

The hierarchical flow has the best total cell area among the three flows. After the hierarchical, comes the topographical then the flat. For the power, the flat flow is the best. This is because the optimization between modules helps in having more optimum results. The topographical flow could achieve the best clock period. The iteration between ICC and DC returns the parasitics and delays information in the coarse placement which helps make the netlist more optimized.

The previous results are initial estimations of the merits of the three flows. The following table (Table 6.2) compares the three flows after layout which gives slightly different results from the post-synthesis results. The topographical flow has the best area in contrast to the post-synthesis result. The power has the same trend as the post-synthesis outputs. The dynamic power depends on the frequency of the design. Thus, power cannot show a replicable trend here as the frequencies of the three flows are different.

Table 6.2: Post-Layout Results

	Hierarchical	Flat	Topographical
Technology(nm)	45	45	45
Area mm2	0.134	0.155	0.1326
Dynamic Power(mW)	1.6	1.678	1.79
Leakage Power(mW)	0.155	0.163	0.18
Total Power (mW)	1.77	1.841	1.97
Clock (ns)	2.61	2.2	2

To conclude, it is recommended to use the topographical flow in hierarchical flavor in case the ASIC designer aims for the best performance and area. The flat flow can be used if the design is small enough so that the tool can optimize the whole design without much effort. The hierarchical flow is recommended to be used in case the runtime is important for the designer or if he wants to have a quick estimation of the design merits. Also, the hierarchical flow should be used if power is critical in the design.

6.2 Future Work and Recommendations

Multi-mode multi-corner analysis

Multi-mode multi-corner (MMMC) simulations are essential to guarantee that the design will work under different conditions and in different environments. The Nangate PDK that we used didn't provide or cover all the possible corners for the expected operating condition of the design. The available temperature corners were at temperatures -40, 0 and 125 °C which is far from the expected operating temperature of a processor ranging from 25 to 80 °C. The PDK didn't also provide the typical-typical variation and only two variants of supply voltage at 0.95 and 1.25 voltage. We would recommend the use of a PDK that supports a wider range of corners and modes to match the intended design. We would also recommend the use of the MMMC flow in synopsys ICC which helps to optimize the design placing and routing over different corners which result in a more optimized design.

Implementing DFT

DFT or Design for testability is a design technique for post manufacturing testing. It makes testing chips possible and cost-effective through adding additional circuitry to the chip. The addition of this circuitry is done to the design netlist after the synthesis phase. The added hardware adds overhead to the design which reduces its efficiency. This degradation in performance is mainly due to leakage power of the added logic as well as the wasted area. These tradeoffs become more relevant in submicron technologies.

Physical verification

We recommend the use of a specialized physical verification tool to solve DRC and LVS violations. Depending on tools such as ICC to solve such problems is a tedious job as the tool is not optimized for this kind of task. The best approach we found was to minimize the number of DRC and LVS violations through the different routing commands and using iterative improvements on each step in the PNR. The design should be then imported to a specialized physical verification tool such as Synopsys IC Validator to solve any remaining issues.

References

[1]C. Tozzi, "Open Source Hardware: What It Means and Why It Matters – Channel Futures", Channel Futures, 2021. [Online]. Available: https://www.channelfutures.com/open-source/open-source-hardware-what-it-means-and-why-it-matters. [Accessed: 11- Jul- 2021].

[2]*Cisco.com*, 2021. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. [Accessed: 11- Jul- 2021].

[3]"RISC-V", *RISC-V International*, 2021. [Online]. Available: https://riscv.org/. [Accessed: 11- Jul- 2021].

[4]P. platform, "PULP platform", *Pulp-platform.org*, 2021. [Online]. Available: https://pulp-platform.org/. [Accessed: 11- Jul- 2021].

[5]"Specifications - RISC-V International", *RISC-V International*, 2021. [Online]. Available: https://riscv.org/technical/specifications/. [Accessed: 11- Jul- 2021].

[6]"Codasip RISC-V Processors | Codasip", *Codasip*, 2021. [Online]. Available: https://codasip.com/products/codasip-risc-v-processors/. [Accessed: 10- Jul- 2021].

[7]"RISC-V Core IP - SiFive", *https://www.sifive.com/share.png*, 2021. [Online]. Available: https://www.sifive.com/risc-v-core-ip. [Accessed: 10- Jul- 2021].

[8]"AndesCore™ Processors - Andes Technology", *Andes Technology*, 2021. [Online]. Available: http://www.andestech.com/en/products-solutions/andescore-processors/. [Accessed: 10- Jul- 2021].

[9]*Iosrjournals.org*, 2021. [Online]. Available: http://www.iosrjournals.org/iosr-jvlsi/papers/vol6-issue3/Version-2/J0603025964.pdf. [Accessed: 11- Jul- 2021].

[10]"An Academic RISC-V Silicon Implementation Based on Open-Source Components", *Ieeexplore.ieee.org*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9268664. [Accessed: 11- Jul- 2021].

[11]T. Fritzmann, G. Sigl and J. Sepúlveda, "RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 239-280, 2020. Available: 10.46586/tches.v2020.i4.239-280.

[12]"A 45nm 1.3GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators", *Ieeexplore.ieee.org*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/6942056. [Accessed: 11- Jul- 2021].

[13]A. Kahng, *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Dordrecht: Springer Netherlands, 2011.

[14]v. adventure, "VLSI Backend Adventure", *Vlsi-backend-adventure.com*, 2021. [Online]. Available: http://vlsi-backend-adventure.com/. [Accessed: 11- Jul- 2021].

[15]M. Farag, *ASIC Design of the Opensparc T1 Processor Core*.

[16] Synopsys, ”Design Compiler Guide”, Synopsys, Inc, December 2011.

[17]Synopsys, ”IC Compiler Implementation User Guide”, Synopsys, Inc, March 2012.

[18]Synopsys, ”Synopsys Timing Constraints and Optimization User Guide”, Synopsys, Inc, March 2010.