Physical design of USB1.1

BODDU SIVA KOTI

ROLL NO: 214EC62414

Department of Electronics and communication Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769008,
India May 2016

Physical design of USB1.1

A thesis submitted in partial fulfillment of the requirement for the degree of

Master of Technology
in
Electronics and Communication Engineering
(Specialization: VLSI Design & Embedded Systems)

by
BODDU SIVA KOTI
Roll No. 214EC2414

Under the supervision of
Prof. Debiprasad Priyabrata Acharya

May 2016
Department of Electronics and communication Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769008,
India

DEPARTMENT OF ELECRTONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY,
ROURKELA, ODISHA -769008.

DECLARATION

I certify that,

- The work presented in this paper is an original content of the research done by myself under the general supervision of my supervisor.
- The work has not been submitted to any other institute for any degree or diploma.
- The data used in this work is taken from free source and its credit has been cited in reference.
- The materials (data, theoretical analysis and text) used for this work has been given credit by citing them in the text of thesis and their details in the references.
- I have followed the thesis guidelines provided by the institution

BODDU SIVA KOTI

DEDICATED TO
MY BELOVED FAMILY
AND
MY DEAR FRIENDS

DEPARTMENT OF ELECRTONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY,
ROURKELA, ODISHA -769008.

CERTIFICATE

This is to certify that the work done in the report entitled **—Physical design of USB1.1"** by **—BODDU SIVA KOTI**" is a record of research work carried out by him in National Institute of Technology, Rourkela under my supervision and guidance during 2015-16 in partial fulfillment of the requirement for the award of degree in Master of Technology in Electronics and Communication Engineering (VLSI & Embedded Systems), National Institute of Technology, Rourkela. To the best of my knowledge, this thesis has not been submitted for any degree or diploma.

Prof. D. P. Acharya

Place: Dept. Of Electronics and Communication Engg.

Date: National Institute of Technology

Rourkela-769008.

ACKNOWLEDGEMENT

This research work is one of the significant achievements in my life and is made possible because of the unending encouragement and motivation given by so many in every part of my life. It is immense pleasure to have this opportunity to express my gratitude and regards to them.

Firstly, I would like to express my gratitude and sincere thanks to Prof. D. P. Acharya, my supervisor, Department of Electronics and Communication Engineering for his esteemed supervision and guidance during the tenure of my project work. His valuable advices have motivated me a lot when I feel saturated in my work. His impartial feedback in every walk of the research has made me to approach a right way in excelling the work. It would also like to thank him for providing best facilities in the department.

Also, I would like to express my gratitude and respect to Prof. K. K. Mahapatra, Prof. S. K. Patra, Prof. S. Meher, Prof. A. K. Swain, Prof. M. N. Islam and Prof. P. K. Tiwari for their support, feedback and guidance throughout my M. Tech course duration. I would also like to thank all the faculty and staff of the ECE department, NIT Rourkela for their support and help during the two years of my student life in the department.

Lastly, I am very thankful to all my classmates and seniors of VLSI lab, especially Mr. Tom, Mr. Soubhagya, Mr. Debasish Nayak, Mr. Umakanta Nanda and all my friends who always encouraged me in the Successful completion of my thesis work. I would like to express my love and heartfelt respect to my parents, and friends for their consistent support, encouragement in every walk of my life without whom I would be nothing.

BODDU SIVA KOTI

Abstract

In earlier days, interfacing peripheral devices to host computer has a big problematic. There existed so many different kinds’ ports like serial port, parallel port, PS/2 etc. And their use restricts many situations, Such as no hot-pluggability and involuntary configuration. There are very less number of methods to connect the peripheral devices to host computer. The main reason that Universal Serial Bus was implemented to provide an additional benefits compared to earlier interfacing ports.

USB is designed to allow many peripherals to be connected using a single standardized interface. It provides an expandable, fast, bi-directional, cost effective, hot-pluggable Plug and Play serial hardware interface that makes the life of the computer users easier allowing them to plug different peripheral devices into a USB port and have them automatically configured and ready to use.

In this thesis demonstrated the USB v1.1 architecture part in briefly and generated gate level netlist form RTL code by applying the different constraints like timing, area and power. By applying the various types design constraints so that the performance was improved by 30%. And then it implemented in physically by using SoC encounter EDI system, estimation of chip size, power analysis and routing the clock signal to all flip-flops presented in the design. To reduce the clock switching power implemented register clustering algorithm (DBSCAN). In this design implementation TSMC 180nm technology library is used.

List of Contents

List of figures

List of figures

List of tables

Chapter 1

1. Introduction

1.1 Background

VLSI technology is the process of making integrated circuits by merging the billions of transistor in a single chip. In modern VLSI is more challenging thing because the day by day device dimensions are shrinking, so that there are so many other effects has to be consider like leakage current, power reduction, noise effects due to coupling and manufacturing effects. Actually the ASIC design is a twostep process, the first one is frontend design and other second is backend design or physical design. In frontend design the designer will write HDL coding and then will verify the functionality of design. Then next step is synthesis, in which generation of technology dependent gates and interconnection among them by considering the design constraints like area, timing, and power. The generated netlist and constraints will give input to the PnR tools that is backend design tool. In physical design the designer has to fix the chip dimensions and shape of the chip, placing hard macros in proper region in the core area, supply the power for all standard cells i.e. power grind creation, routing the clock signal for all flip-flops in design, routing the signal nets and the extraction of parasitics for whole design so that noise analysis can perform easily. And the final phase of physical design is sign off analysis with extracted parasites. In above all stages has to be taken care of setup and hold vioations.Then the final stage is generation GDS II file.

1.2 Motivation

In olden days, connecting different peripherals to computer was a great challenge. There were different types of connecting ports available (serial port, parallel port, PS/2 etc.) and but their utilization has limited with no hot pluggability and autonomus configuration. There are ery less options to have these peripheral devices connected in the original IBM PC implementation due to the limitation with respect to nonshareable IRQ lines and I/O address space. The important reason for USB implementation is to provide a replacement for those legacy ports in a computer to provide the addition of the devices with quick and easy user interface.

USB is designed in such a manner which provides different peripherals to be connected using a standardized interface It also gives an expandable, fast, bi-directional, low-cost, hot-pluggable Plug and Play serial hardware interface. This makes the life of the computer users comfortable and allow them to plug many peripheral devices with a USB port and have them automatically configured and ready to use. Using a single connector type, USB allows the user to connect to a wide variety of external devices, which includes keyboards, mice, printers, scanners, mass storage devices, telephones, modems, digital still-image cameras, video cameras, audio devices to a computer. USB devices does not consume system resources directly. They are neither mapped into I/O address space, nor do they use IRQ lines or DMA channels. The very important system resource required by a USB system are the memory buffers which are built in with the USB system software. Due to its successful and widespread acceptance, USB became the most widely used industry standard for connecting any device to PCs or laptops.

1.3 proposed idea

In physical design flow the clock routing to all sequential flip-flops is challenging task because the switching power is depends on the clock newt work. In this thesis proposed an idea of register clustering algorithm that is DBSCAN. This algorithm is based on density of the flip-flops and also clock pin capacitance. There are so many register clustering algorithms are present like multibit flip-flop but this algorithm will reduce the clock network capacitance largely compared to others, so that the clock switching power will reduce.

1.4 Thesis organization

This thesis will explain the architecture, synthesis, physical design flow and register clustering algorithm. The background, motivation and proposed idea explained in chapter 1. The brief architecture part of the USB explained in chapter 2. The chapter 3 explained synthesis process by using RTL compiler. In chapter 4 explained some of basic static timing analysis concepts. In chapter 5 physical design flow and timing closure by considering the onchip variation. In chapter 6, implemented register clustering algorithm to reduce the clock network capacitance. Finally the conclusion and future scope are explained in chapter 7.

Chapter 2

2. Architecture of USB

2.1 Architecture of USB

USB supports data exchange between ahost computer system and a wide range of parallel accessible peripherals or devices. The interfaced module share USB bandwidth through a host scheduled token based protocol. A USB system is described by three definitional areas namely, USB interconnect, USB devices and USB host. USB does not provide a mechanism for attached devices to arbitrate for use of the bus, as a result host is the master on the bus and totally controls all the bus activity.

The OHCI specification defines the software and hardware interface to the host controller in order to provide a common industry standard interface to the USB bus. The OHCI divides the USB host controller implementation into a software component called Host Controller Driver (HCD) and a hardware component termed as HC. The USB host is required to generate the required traffic on the USB cable as per the demands put forward by the client software. The data transfer demands put by different applications (client software) from time to time are analysed by HCD and data transfer scheduling lists are generated. The HC executes the schedule list generated by HCD by way of sending different tokens on the USB bus as per the protocol. The OHCI defines two levels of arbitration to select among the endpoints. In the first level of arbitration each endpoint requiring a different type of service is put in a list and host controller selects which list to service based on predefined protocol and parameters. In the second level all endpoints among the list are given equal priority and equal service opportunities.

2.2 System Overview

The Fig 1: V9090OCP USB Host Controller in a USB System below shows a megacell and its context. The responsibility of HC is to generate the traffic on USB bus by traversing through the EDs and TDs provided by HCD. The megacell has OCP interface on application side which can be easily migrated to any type of bus by having a wrapper around it. The other side of the megacell is USB PHY Transceiver interface. The megacell interacts with HCD through the PCI Bus with the use of PCI to OCP bridge in a typical PC environment. The Host controller can also be used in an embedded application with OCP to system Bus Bridge

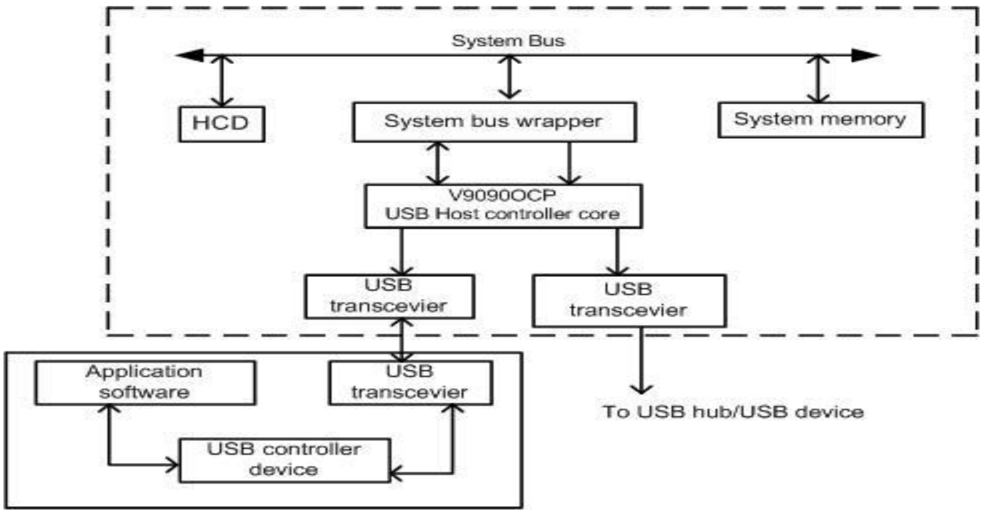


Figure 1: V9090OCP USB Host Controller in a USB System

2.3 Hardware Overview

- OCP compatible slave configuration Bus interface for read or write transactions to internal configuration and operational registers.
- OCP compatible Master bus with Simple Extension interface. This interface is used for performing write and reads from the system memory.
- Frame Management logic for generating SOF token on USB.
- FIFO is implemented to store the data to bridge between application interface and USB.
- Root Hub with configurable number of downstream ports.
- USB 1.1 complaint SIE

2.4 Features of USB

The HC has the following features

- Open HCI Rev 1.0a compatible
- USB Rev 1.1 compatible
- Number of downstream ports for Root Hub is user configurable
- Standard OCP interface on application side
- Supports both Full speed and Low speed devices
- Legacy support for Keyboard and Mouse.

2.5 USB States

This following Fig 2: USB States shows all the possible states in USB HC

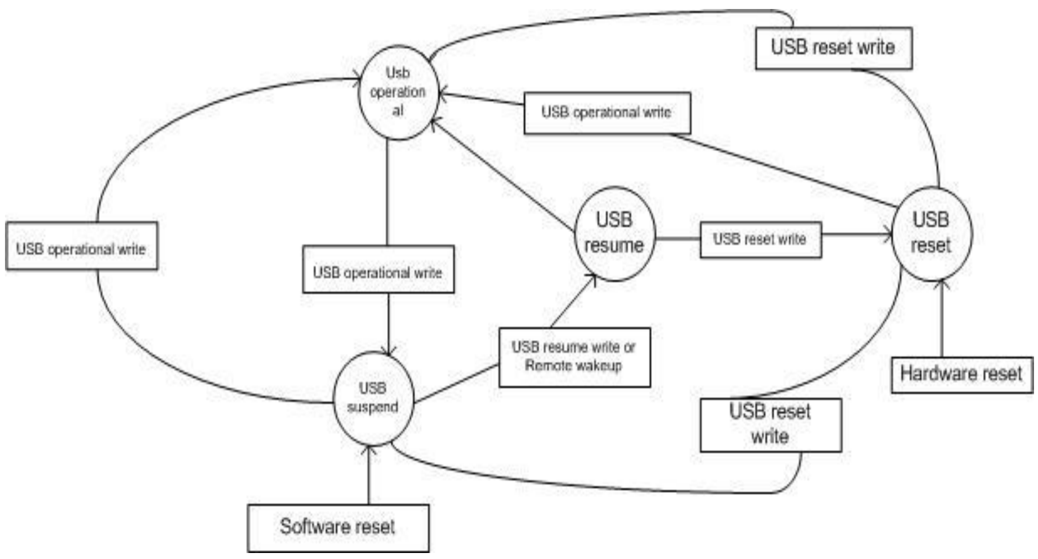


Figure 2: USB States

2.5.1 UsbOperational

In a USBOPERATIONAL state, the Host Controller starts to generate SOF Tokens. The USBOPERATIONAL state can have entry to USBRESUME or SBRESET states. This also lead to operate the USBRESET or USBSUSPEND states. While it transits from USBRESET or USBRESUME to USBOPERATIONAL, the Host Controller is taken control for terminating the USB reset or resume states. A transformation to the USBOPERATIONAL state has it affects on the frame management registers by the Host Controller. Also the Host Controllers state is to OPERATIONAL, the FrameRemaining field of HcFmRemaining is built with the corresponding value of the Frame Interval field in HcFmInterval.

2.5.2 UsbReset

In this state, the Host Controller forces the reset signalling to the bus. The Host Controllers and SOF Token used to generate which disabled while in USBRESET. This leads to , the FrameNumber field of HcFmNumber, does not consistently increment while the Host Controller is in the USBRESET state. The Host Controller defaults to the USBRESET state following a hardware reset. The HCD has an important role for USB Reset signalling.

2.5.3 UsbSuspend

This USBSUSPEND state explains the USB Suspend state. Here, Host Controllers lists processing and SOF Token generation has been disabled. However, the Host Controllers remote walk-up logic has to monitor the USB walk-up action. The FrameNumber field of HcFmNumber has no implication in increment of the Host Controller in the USBSUSPEND state. This transition leads to the conflicts between the Host Controller Driver which initiates a transition for the USBRESET state.

2.5.4 UsbResume

USBRESUME state holds the Host Controller to force resume signalling on the bus. While in USBRESUME, the Root Hub takes the responsiblity for propagating the USB Resume signal to downstream ports as given by the USB Specification. The Host Controller's list is ude to process and SOF Token are disabled while the USBRESUME. Also, the FrameNumber field of HcFmNumber does not increment with the Host Controller. The transition to USBRESUME is started initially with the Host Controller Driver or by a USB remote walk-up signalled with theRootHub.Legal state transitions from USBRESUME are to USBRESET and to USBOPERATIONAL.

2.4 Data Structures:

The basic building blocks for communication across the interface are the Endpoint Descriptor (ED) and Transfer Descriptor (TD).The Host Controller Driver assigns an Endpoint Descriptor to each endpoint in the system The Endpoint Descriptor contains the information necessary for the Host Controller to communicate with the endpoint. The fields include the maximum packet size, the endpoint address, the speed of the endpoint, and the direction of data flow. Endpoint Descriptors are linked in a list.A series of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint. The Transfer Descriptor contains the information necessary to describe the data packets to be ransferred. The fields include data toggle information, shared memory buffer location, and completion status codes. The descriptor explains about the information contained in it. The head pointers to the bulk and control Endpoint Descriptor lists are maintained within the operational registers in the HC. The Host Controller Driver initializes these pointers prior to the Host Controller gaining access to them Should these pointers need to be updated.

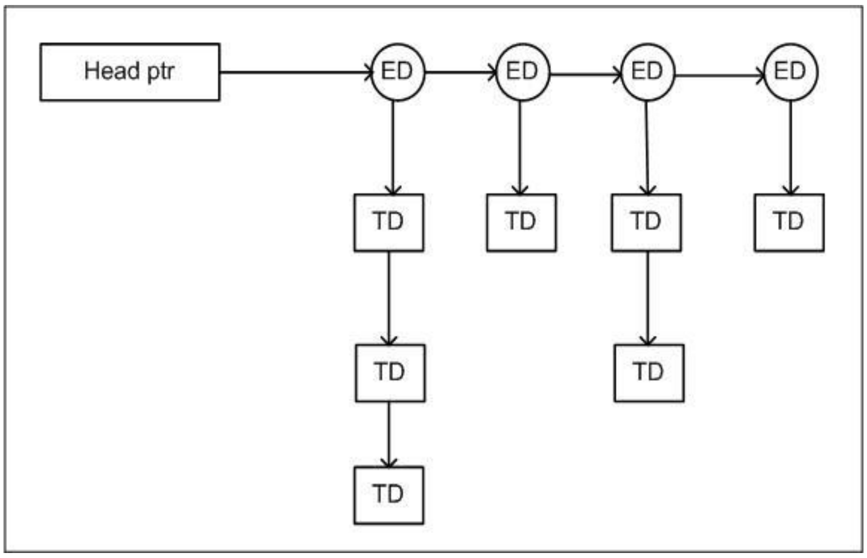


Figure 3: Typical List Structure

. The head pointers to the interrupt Endpoint Descriptor lists are maintained within the HCCA. There is no separate head pointer for isochronous transfers. The first isochronous Endpoint Descriptor simply links to the last interrupt Endpoint Descriptor. There are 32 interrupt head pointers. The head pointer used for a particular frame is determined by using the last 5 bits of the Frame Counter as an offset into the interrupt array within the HCCA.

2.7 Root hub and SIE

Basically, the Root Hub performs the following functions:

- Enabling the Downstream ports and detection of full-speed and low-speed devices.
- Controlling the Downstream ports.
- Packet framing and transmission on USB.
- Packet reception and content extraction.
- Suspend and resume signalling.

The Root Hub USB reset and resume signalling are controlled by the **Host Controller Functional State** bits. The HCD is responsible for all timing associated with these operations. The port reset and resume signal timing is controlled by the hardware.

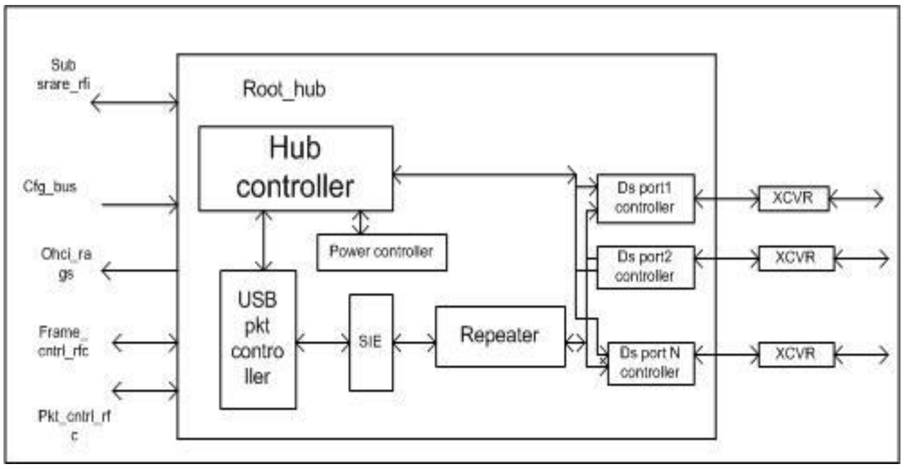


Figure 4: Root Hub & SIE Block Diagram

2.7.1 Hub controller

The Root Hub operational registers are implemented in the hub controller. These registers can be accessed through Root Hub register interface. These registers are implemented such that they are writable regardless of the HC USB state. These are must be writable only during the UsbOperational state. The hub controller interfaces with all the downstream ports. It controls each downstream port by setting power, enabling the ports, suspending and resuming the port.

2.7.2 USB packet controller

This block interacts with the FIFO, USB state block, frame controller, and list processor. Basically it initiates the packet transfers on USB on getting the requests from frame controller and list processor. If USB state is in operational state, the SOF packet will be transmitted for every 1ms. Similarly, while processing the nonperiodic and periodic lists by list processor, various types of command, data, and handshake packets will be encrypted and transmitted to the USB.

2.7.3 SIE

The serial interface engine performs the following functions:

- NRZ encoding and decoding
- Bit stuffing and destuffing
- 5bit and 11bit CRC calculation and check
- USB protocolisation and deprotocolisation
- Parallel to serial and serial to parallel conversion

The SIE sits between USB packet controller and repeater. During packet transmission, it takes the inputs from USB packet controller in terms of bytes, converts into serial form, add the protocol overheads, perform bit stuffing and NRZ encoding, and send to the repeater. During reception it just performs the reverse function of transmission and gives the byte aligned data to the USB packet controller.

2.7.4 Repeater

The repeater handles the connectivity between the USB packet controller and the downstream ports. It performs the function of multiplexing and demultiplexing for the downstream ports. During packet transmission it broadcasts the USB format serial data to all the enabled downstream ports. During packet reception, it detects the SOP from a particular port and forwards the received packet to the USB packet controller. It also takes care of the conflicts when more than one downstream port device tries to drive the USB.

Chapter 3

3. *Synthesis of USB*

3.1 Introduction

VLSI technology is wide use in modern digital systems and it is allows hundreds of thousands of transistors in a single ASIC and level of integration is increasing at drastically. This trend stressed the ability of designer to keep pace with the advances in technology. VLSI design proceeds through a number of different phases. The design begins with understanding the purpose of circuit behaviour i.e., the inputs and outputs of design and how they are related. The design representation at this level is communicated using the hardware description languages, timing diagrams and block diagrams. The foremost design phase is RTL (register transfer language) design. A RTL representation for design describes the registers and the operations which are performed on the values stored in the registers and control conditions which sequences these operations. The next phase is logic design which means converting registers controllers and computational blocks from register transfer language description into a logic level representation using the technology dependent blocks (standard cells) which are available in technology library. Then the last design phase is backend design, in this interconnection of technology dependent blocks are translated into a integrated circuit. The integrated circuit design, a wide range of CAD (computer aided design) analysis tools are used to measure the quality of correctness of system before fabrication.

Synthesiss is the process of mapping the high level description of a design into optimized gate level Netlist by considering the constraints like area, power and timing. Synthesis uses the standard cell library which has simple cells like AND, OR, INVERTER, NAND, NOR, and macro cells like MUX, ADDER, Flip-Flop and Memory elements etc. and technology is TSMC 180nm . Here for synthesis RC (RTL Compiler) is used which is efficient and easy to use. The circuit (USB) description written in HDL (hardware description language) Verilog language. Synthesis is a repetitive process and starts with stating the timing related constraints for each and every block in the design. These timing related constraints defines relationship of signals with respect to clock input.

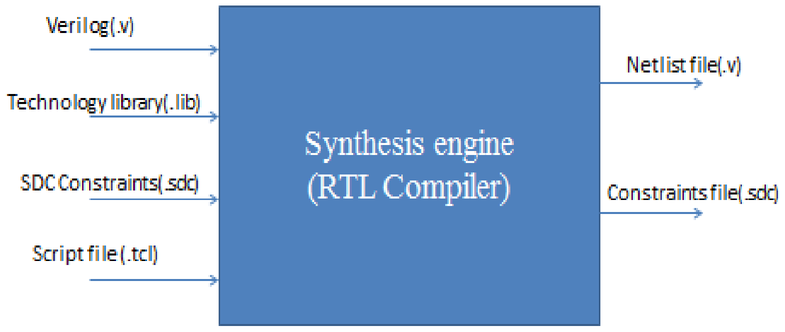


Figure 5: RTL compiler inputs and outputs

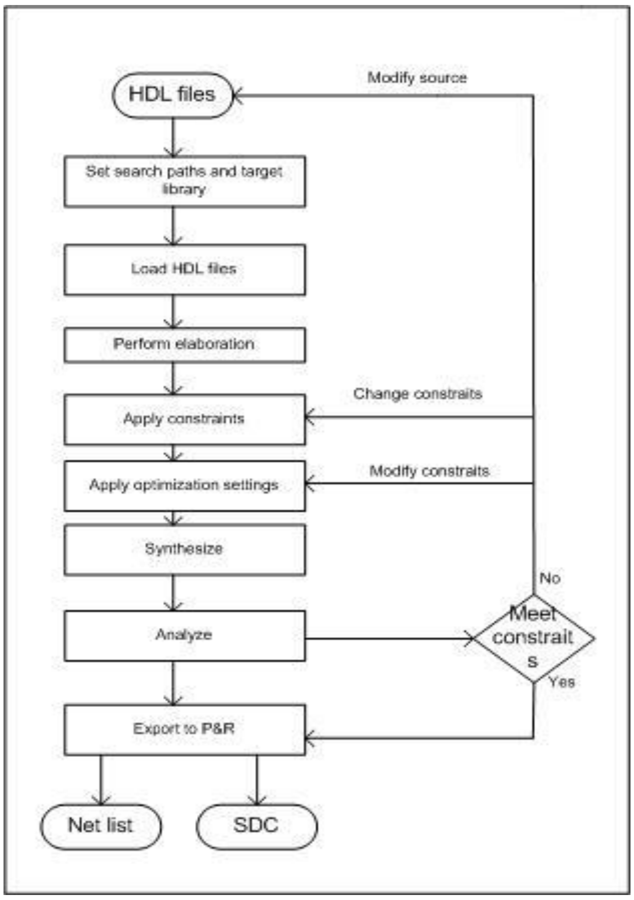


Figure 6: generic synthesis flow

Here search paths are the main directory path names that RC either explicitly or implicitly searches.

3.2 Specifying Search Paths

We can specify search paths for the standard cell libraries, Verilog files, and tcl scripts.

Lib_ search _path

This attribute is used to search the technology library path in which directory .lib file contained.

Script_search_path

This attribute is used to search the .tcl (tool command language) file in which directory it contains.

hdl_search_path

This attribute is used to search the HDL files (.v or .vhd) in which directory it contains.

To set the above search paths, type the below set attribute commands.

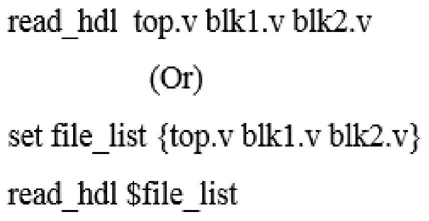
```
set_attribute  lib_search_path    …. /path
set_attribute  script_search_path    … /path
set_ attribute    hdl_ search_ path    …. /path
```

3.3 Loading HDL Files

HDL (Verilog or VHDL) files contains the actual functionality of the design, such as structural code or register transfer language implementations . Use read_hdl attribute to read the Verilog files into RC. When we issue a read_hdl attribute, RC reads the respected files and performs syntax checks.

If the design is described by multiple verilog files, we can read them in using the below

List the filenames of all the Verilog files and use the read_hdl attribute once to read these set of files simultaneously .



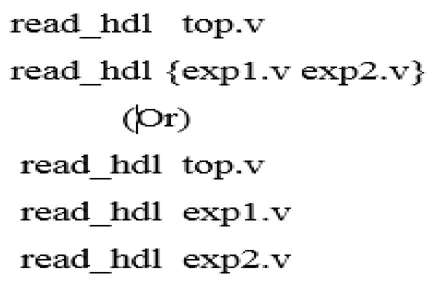
```
read_hdl  top.v blk1.v blk2.v
                (Or)
set file_list {top.v blk1.v blk2.v}
read_hdl $file_list
```

The host directory searched where the HDL files are specified using the hdl_ search _path root attribute.

The following attribute reads two Verilog files into a library.

```
read _hdl  –v  -library  slow .lib   {exp1.  v exp2.  v }
```

Use the read_hdl attribute multiple times to read the Verilog files sequentially.



```
read_hdl  top.v
read_hdl {exp1.v exp2.v}
        (Or)
 read_hdl  top.v
 read_hdl  exp1.v
 read_hdl  exp2.v
```

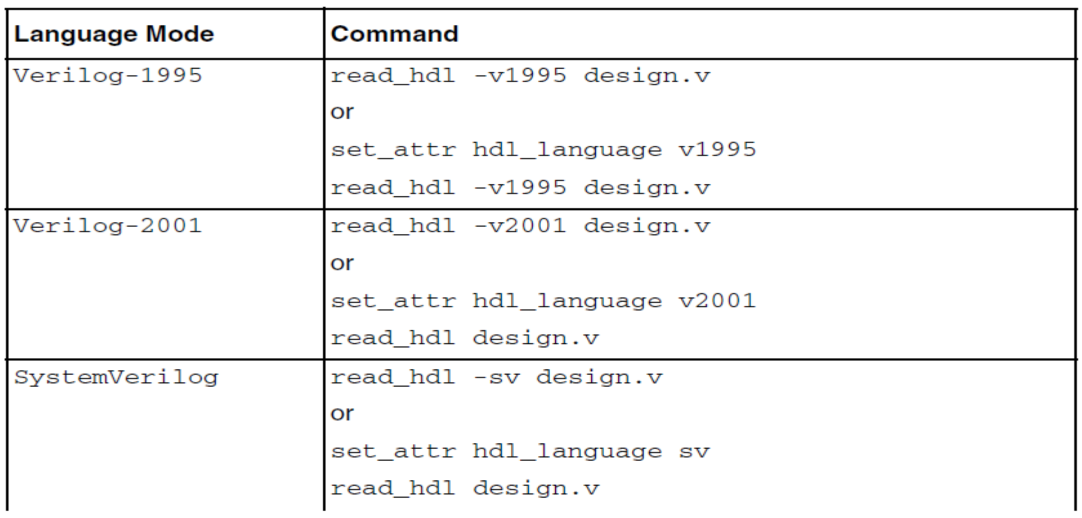
3.4 Specifying the. HDL Language. Mode

To specify the default HDL language version and to read these design files use the below attribute

set_attribute hdl_language { v1995 | v2001 | sv | vhdl}

Default : v1995

Table: Specifying the Language Mode



Language Mode	Command

Verilog-1995	`read_hdl -v1995 design.v`
	or
	`set_attr hdl_language v1995`
	`read_hdl -v1995 design.v`

Verilog-2001	`read_hdl -v2001 design.v`
	or
	`set_attr hdl_language v2001`
	`read_hdl design.v`

SystemVerilog	`read_hdl -sv design.v`
	or
	`set_attr hdl_language sv`
	`read_hdl design.v`

3.5 Elaboration

The elaboration involves various design checking and optimizations and is a necessary step to proceed with synthesis. The elaborate command automatically elaborates the top-level design and all of its submodules. During elaboration, RTL Compiler performs the following tasks:

- Builds data structures.
- Infers registers in the design.
- Performs higher- level HDL optimization, such as dead code removal.
- Checks semantics.

After elaboration, (RC) RTL Compiler will create the generic level netlist for the total design and then we can apply design constraints and perform other operations.

Syntax: elaborate top_module.v

3.6 Applying timing Constraints

In RTL Compiler, a clock signal waveform is a periodic signal with one rising edge and one falling edge per period. Clock waveforms may be applied to design objects such as input ports, clock pins of sequential cells, external clocks (also known as virtual clocks), mapped cells, or hierarchical boundary pins.

To define clocks use the define clock command

We can group more than one clock that are synchronous to each other, allowing timing analysis to be performed between these clocks. This group is called a clock domain. If a clock domain is not specified, RTL Compiler will assume all the clocks are in the same domain. By default, RTL Compiler assigns clocks to domain_1, but we can create our own domain name with the -domain argument in define_clock command.

The below two different clocks are created and then these two clocks assigned to separate clock domains:

```
define_clock -domain Siva -name sys_clk -period 445  [find / -port sys_clk]
define_clock -domain Sudha -name osc_clk_i -period 900  [find / -port osc_clk_i]
```

To remove clocks, use the rm command. If we have defined a clock and saved the object variable, for example as sync, we can remove the clock object as shown in the following.

```
rm $sync.
```

3.7 Applying Design Rule Constraints

When optimizing the design,RTL Compiler tries to satisfy all design rule constraints (DRCs) . Some of DRCs include maximum transition, maximum fan-out, and capacitance limits; operating conditions; and wire -load models. These constraints are specified using attributes on a module or port, or from the technology library. However, even without user-specified constraints, rules may still be inferred from the technology library.

- To specify a maximum transition limit for all nets in a design or on a port, use the Max_transition attribute on a top-level block or port:

  ```
  Set_attribute max_transition value  [ design | port ]
  ```

- To specify a maximum fan-out limit for all nets in a design or on a port, use the max_fanout attribute on a top-level block or port:

  ```
  Set_attribute max_fan-out value  [ design | port ]
  ```

- To specify a maximum capacitance limit for all nets in a design or on a port, use the max_capacitance attribute on a top-level block or port :

  ```
  Set_attribute max_capacitance value  [ design | port ]
  ```

  To specify a specific wire- load model to be used during synthesis , use the force_wireload attribute. The below one specifies the 1x1 wire-load model on a design named top.

  ```
  set_attribute force_wireload 1x1 top
  ```

3.8 Defining Optimization Settings

By default, RTL Compiler will perform optimizations that can result in logic changes to any object in the design. We can prevent any logic changes in a design or block while still allowing mapping optimizations in the surrounding logic, by using the preserve attribute.

- To preserve hierarchical instances, use the below command:

  Set_attribute preserve true *object*

  Where *object* is a hierarchical instance name.
- To preserve primitive instances, use the below command:

  Set_attribute preserve true *object*

  Where *object* is a primitive instance name.
- To preserve modules or submodules, use the below command:

  Set_attribute preserve true *object*

  Where *object* is a module or submodule name

3.9 Setting Boundary Optimization

RTL Compiler performs boundary optimization for all hierarchical instances in the design during synthesis. Examples of boundary optimizations include:

- Constant propagation across hierarchies

  This includes constant propagation through both input ports and output ports.
- Removing undriven or unloaded logic connected.
- Collapsing equal and opposite pins

Two hierarchical boundary pins are considered equal (opposite), if RTL Compiler determines that these pins always have the same (opposite or inverse) logic value.

- Hierarchical pin inversion

RTL Compiler (RC) might invert the polarity Of a hierarchical boundary pin to improve QoR. However it is not guaranteed, That QoR improved globally by this local optimization.

- Rewiring of equivalent signals across hierarchy

Hierarchical boundary pins are feedthrough pins, if output pins always have the same (or inverted) logic value as an input pin. Such feedthrough pins can be routed around the Subdesign and no connections or logic is needed inside the sub-design for these pins.

- RTL Compiler can disconnect if more than one input is identical and then one of them and use the other output to drive the fan-out logic for both.

3.10 Performing Synthesis

Synthesis is the process by which it transforms the HDL (Verilog or VHDL) design into a real gates and interconnection among them, given that all the stipulated constraints and optimization settings.

During the synthesis stage RC will do below four processes :

- RTL Optimization
- Global Focus Mapping
- Global Incremental Optimization
- Incremental Optimization (IOPT).

3.10.1 RTL Optimization

During RTL optimization, RC effectively performs optimizations like data path synthesis, speculation, multiplexer optimization, and carry save arithmetic (CSA) optimizations, resource sharing. After this phase, RC performs logic optimizations like structuring and redundancy removal.

3.10.2 Global Focus Mapping

In this phase RC does global mapping at the end of the RTL technology independent optimizations(during the synthesize –to_mapped command).Inthis step mapping, restructuring and the design concurrently, and also optimizations like splitting, pin swapping, buffering, pattern matching, and isolation .

3.10.3 Global Incremental Optimization

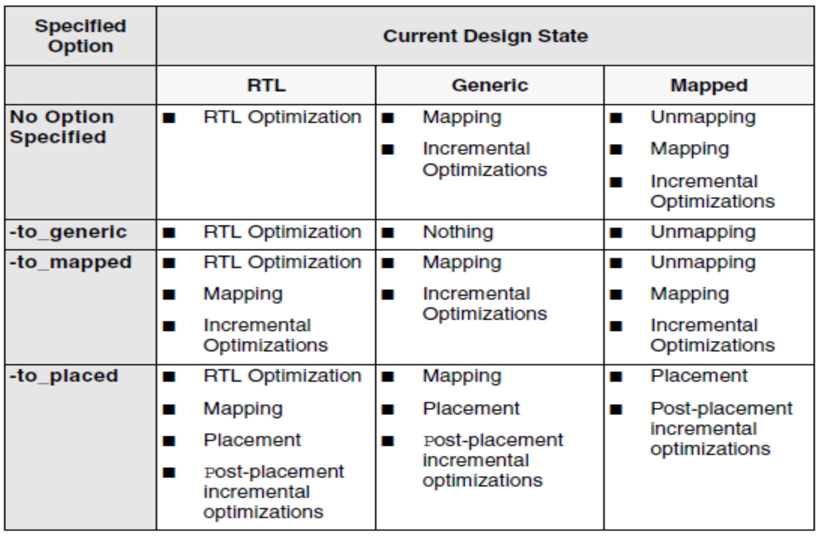
After global mapping, RC does the synthesis global incremental Optimization. In this phase mainly targeted at area optimization and power optimization. Optimizations performed at this stage include global sizing of cells and Optimization of buffer trees.

3.9.4 Incremental Optimization (IOPT)

This is the final stage of optimization RC does incremental optimization. Optimizations performed during IOPT improve timing and area and fix DRC violations.Optimizations performed during this phase include multibit cell mapping, incremental clock Gating, incremental retiming, tie cell insertion, and assign removal.

Synthesize command is executed the below two steps .

- Synthesizing the design into generic logic gates (RTL optimizations are performed in this step).
- Mapping to these generic gates to technology library and performing incremental optimization.
- Table 2: Actions Performed by the synthesize Command



Specified Option	Current Design State		
	RTL	Generic	Mapped
No Option Specified	■ RTL Optimization	■ Mapping ■ Incremental Optimizations	■ Unmapping ■ Mapping ■ Incremental Optimizations
-to_generic	■ RTL Optimization	■ Nothing	■ Unmapping
-to_mapped	■ RTL Optimization ■ Mapping ■ Incremental Optimizations	■ Mapping ■ Incremental Optimizations	■ Unmapping ■ Mapping ■ Incremental Optimizations
-to_placed	■ RTL Optimization ■ Mapping ■ Placement ■ Post-placement incremental optimizations	■ Mapping ■ Placement ■ Post-placement incremental optimizations	■ Placement ■ Post-placement incremental optimizations

3.10.4 Setting Effort Levels

We can specify the three effort levels with the -effort {low | medium | high } preference in the synthesize command. The probable values for this effort option are as follows.

Low : The design is mapped to gates, but RC will very little RTL optimization, incremental clean up, DRC fixing, or redundancy identification and removal. The low effort is generally not endorsed.

Medium (default setting): RC performs enhanced timing driven structuring, incremental synthesis, and redundancy identification and removal on the design

High : RC make sures the timingdriven structuring on bigger segments of logic and spends considerable time to makes the incremental clean up. This high effort level encompasses forceful redundancy identification and removal.

3.10.5 Generating Reports

Timing Reports

By using report timing command we can produce reports on the timing of the present working design . The evasion timing report generates the in depth view of the maximum dangerous path in the present design.

The timing report gives the ensuing data:

- kind of standard cell (flop-flop, nand, or, inverter etc.)
- The standard cells fan-out and timing features (output load, input slew, and the propagation delay).
- Arrival time for each point on the most dangerous timing path.

Area Reports

The area report provides the summary of the area of each standard cell in the present design. The report gives the number of standard cells and the area of each cell based on the definite technology library.

3.11 ILM model

This ILM timing model is very useful in hierarchical design, it is a gate level model or partial netlist of a actual physical design. In this having the connections between inputs to first stage of the flip-flops and output side from the output of flip-flop to final output pin. That means it containing the timing information of only interface logic, there is no information of middle flip-flips. The main advantage of this model is reduces the time for timing closure. And the other advantage of this model is

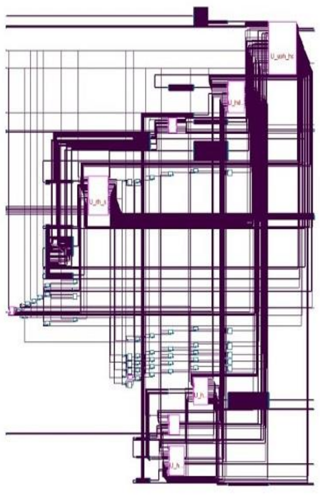


Figure 7: Original netlist

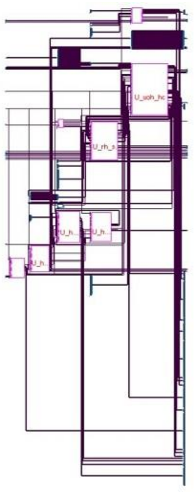
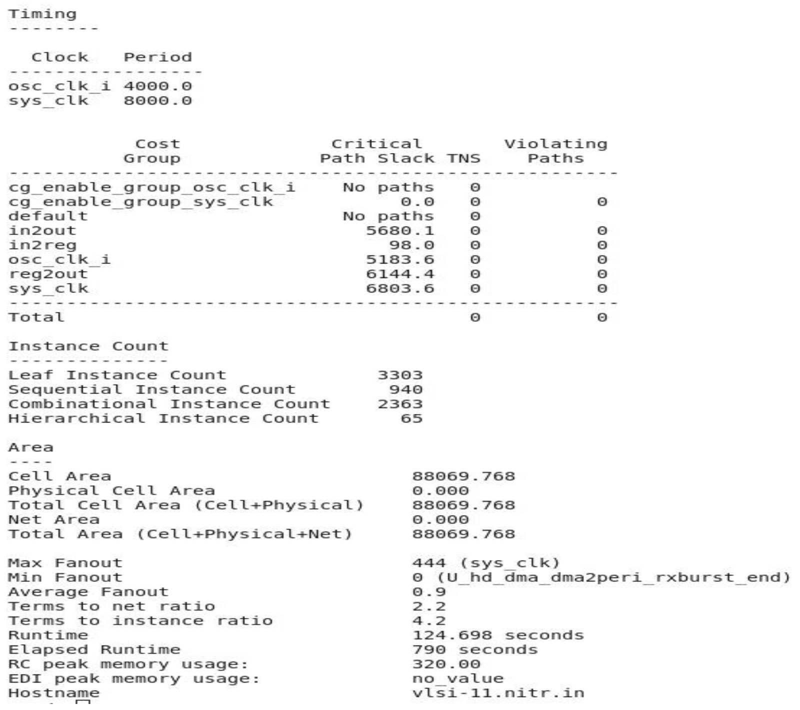


Figure 8: ILM model for USB



```
Timing
--------

  Clock   Period
-----------------
osc_clk_i 4000.0
sys_clk   8000.0

           Cost              Critical        Violating
           Group            Path Slack TNS    Paths
-----------------------------------------------------
cg_enable_group_osc_clk_i     No paths   0
cg_enable_group_sys_clk          0.0   0           0
default                       No paths   0
in2out                        5680.1   0           0
in2reg                          98.0   0           0
osc_clk_i                     5183.6   0           0
reg2out                       6144.4   0           0
sys_clk                       6803.6   0           0
-----------------------------------------------------
Total                                  0           0

Instance Count
--------------
Leaf Instance Count             3303
Sequential Instance Count        940
Combinational Instance Count    2363
Hierarchical Instance Count       65

Area
----
Cell Area                           88069.768
Physical Cell Area                  0.000
Total Cell Area (Cell+Physical)     88069.768
Net Area                            0.000
Total Area (Cell+Physical+Net)      88069.768

Max Fanout                          444 (sys_clk)
Min Fanout                          0 (U_hd_dma_dma2peri_rxburst_end)
Average Fanout                      0.9
Terms to net ratio                  2.2
Terms to instance ratio             4.2
Runtime                             124.698 seconds
Elapsed Runtime                     790 seconds
RC peak memory usage:               320.00
EDI peak memory usage:              no_value
Hostname                            vlsi-11.nitr.in
```

Figure 9: qor report for USB

Chapter 4

4. *Brief Review on STA concepts*

4.1 Introduction

STA is a method of validating the timing of the present design under vilest case condition. In recent digital Integrated circuit design flow, static timing analysis is vital to validate the timing of critical paths for consequent optimization , to estimate the possible clock frequencies, to avoid over-design, and to reach timing closure for the design in stringent timing constraints. Static timing analysis is *static* since the analysis of the design is carried out statically and does not depend upon the data values being applied at the input. This is in contrast to simulation based timing analysis where a stimulus is applied on input signals, resulting behavior is observed at the output and verified. Rapid growing design complexities and increasing on-chip variations, however, complicate this analysis for nanometer design. These on-chip variations, including manufacturing process, voltage and temperature variations, affect wire delays and gate delays in different portions of a chip. Although statistical timing analysis and multi-corner timing analysis have been proposed to handle these variations, not all sources of variability are accurately modeled.

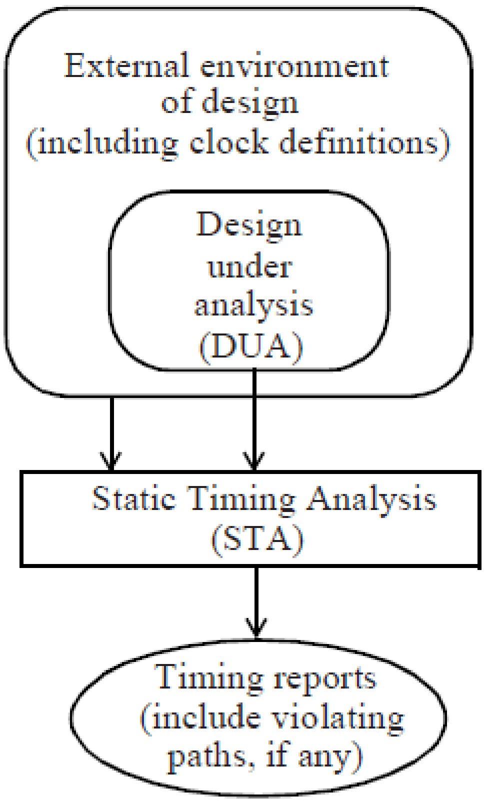

External environment of design (including clock definitions)
Design under analysis (DUA)
Static Timing Analysis (STA)
Timing reports (include violating paths, if any)

Figure 10: Static timing analysis

A STA of a design typically provides a profile of the design's performance by measuring the timing propagation from inputs to outputs . Timing analysis computes the amount of time signals propagate in a circuit from its primary inputs to its primary outputs through various circuit elements and interconnect.Signals arriving at an input of an element will be available at its output at some time later. Each element introduces a delay during signal propagation a signal transition is characterized by it inputslew and output slew, which is defined as the amount of time required for a signalto transition from high to low or low to-high .To account for timing modeling limitations in considering design and electrical complexities ,as well as multiple sources of variability , such as manufacturing variations, temperature fluctuation and voltage drops , timing analysis is typically done using an early-late split ,where each circuit node has an early(lower) bound and a late (Upper) bound on its time. By convention, if the mode is not explicitly specified, both late and early modesshould be considered. Both delay and slew are computed separately on each modes . Suppose, in early mode , an output slew is computed using the input slew takenfrom the early mode, and similarly, in late mode , the output slew is computed using input slew.

4.2 CMOS digital design flow

In the design flow the STA can be performed at different stages of implementations. Static timing analysis is rarely done at RTL level, at this stage more important to verify the functionality of the design as opposed to timing because at this all blocks are implemented in behavioral level. After verifying the functionality of the design, RTL has been to synthesize to gate level, then at this stage STA is performed with the timing information. We can perform STA before the logic optimization to find out the worst case or critical timing paths. And after logic optimization also we can run the STA, still there are failing paths are optimized. Then next phase is physical design, at this clock trees are considered as ideal, i.e the wire is considered as zero. Once clock tree built then STA can perform with actual wire delays.

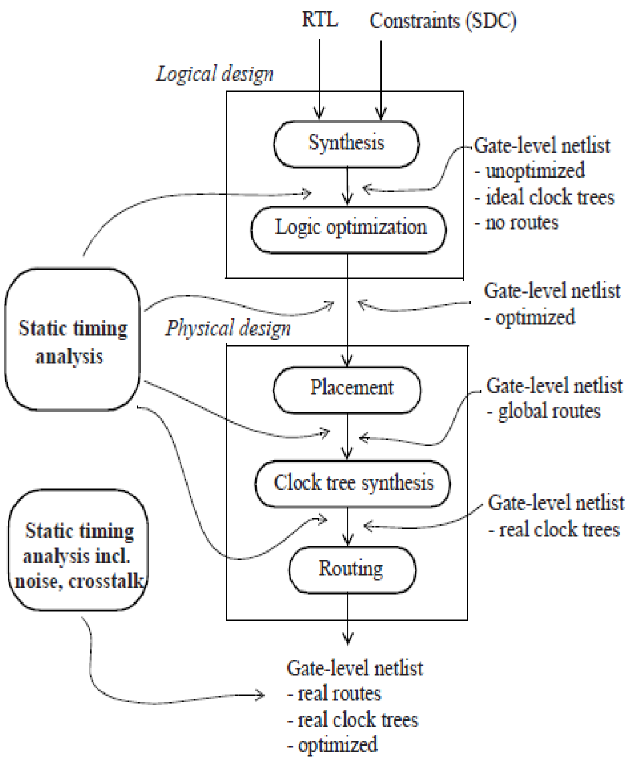


Figure 11: CMOS digital design flow

In the physical design (backend design) have to perform STA at each and every step to verify any failure paths (setup or hold violations) are present or not. In physical implantation the standard cells are connected by metal wires. The RC parasites (Resistance and capacitance) of a metal wire impact the path delay. In a deep submicron technology these RC parasites are the major effected to delay of the path and power dissipation of design. The performance (speed and power) of the design depends on the impact of interconnect parasites.At the logic design stage, the interconnect is assumed as idle because there is no placement information of standard cells. And wire load models are used to estimate the wire delay of particular interconnect, this wireload models provides RC parasites values based on the fan-out of standard cells which are driving.

Before the routing of metal traces finalized, the CAD tools estimate the routing distance to obtain the parasitics for the route. Since routing is not fixed, the phase is called global route to distinguish it from detail route. In the global route stage estimated routes are used to determine the RC values that are necessary to find out wire delays . During in this stage tools can not consider the effect of coupling noise. Then the final stage I s detailed route, in this stage have to consider the effect of coupling noise, because after detail route all the standard cells are fixed in a particular position. After routing an extraction tools are used to extract the parasitics (RC values). Such an extractor may took less runtime to obtain parasitics with less accuracy. By iterative optimization accurate resistance and capacitance values extracted with a longer runtime.

To summarize, the STA can be performed on a gate-level netlist depending on:

- How interconnect is modeled - ideal interconnect , wire load model , global routes with approximate RCs, or real routes with accurate RCs.
- How clocks are modeled whether clocks are ideal (zero delay) or propagated (real
- Whether the coupling between signals is included whether any crosstalk noise is analyzed .

4.3 Limitations of STA

- Reset sequence*:*

  Reset sequence means after synchronous reset or asynchronous reset all flip-flops are should be reset. This cannot be checked during STA, because the initial values are not synthesized.
- X handling:

  The static timing analysis deals with only logic-0 or logic-1. This is in determine value, so this type checks are not verified in STA. X means logic value in between logic-1 and logic-0,that means this is noise or glitch. In STA glitch analysis can be done but this is different from unknown value X.
- PLL settings:

  Mostly PLL is outside, configurations may not be loaded or set properly during STA.
- Asynchronous clock domain crossings:

  STA will not check asynchronous clock domain crossing, for this type of check other tools are required.
- IO interface timing:

  During STA the IO interfacing timing constraints checks not possible to check.

4.4 STA concepts

4.3.1 Switching Waveform

The excitation to RC network below shown in figure, when the sw0 activates the output goes to a high and when sw1 switch activates the output goes to low. The output response equation is given like

$$v = vdd * [1 - exp\{-\frac{t}{Rdh * Cload}\}]$$

Where product of Rdh and Cload is called as time constant , which is related to transition on the output signal.

When output signal goes from logic-high to logic-low, caused by activating *SW1*, the output signal transition shown in Figure (c). The output capacitance discharges through the *SW1* switch which is *on*. The output voltage transition given by the equation:

V = Vdd * e-t/(Rdl * Cload)

In a CMOS inverter, the output charging and discharging waveforms do not appear like the RC charging and discharging waveforms of given figure since the PMOS pull-up and the NMOS pull-down transistors are both *on* simultaneously for a small amount of time.

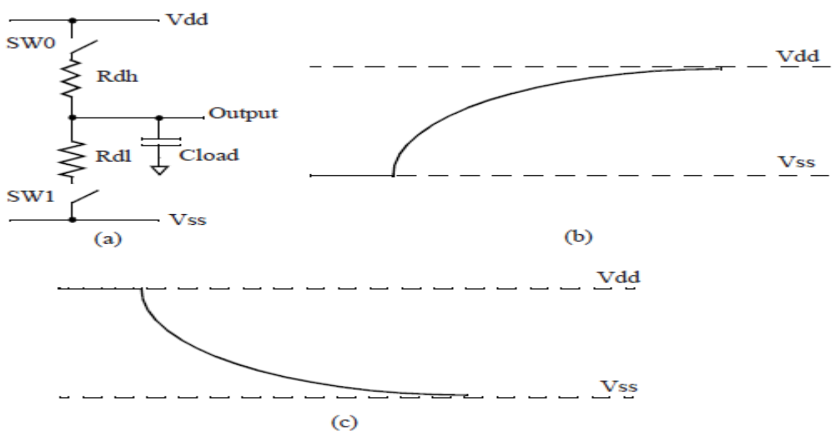


Figure 12: RC charging and discharging waveforms.

4.4.2 Propagation Delay

The propagation delay of a CMOS inverter is defined with respect to some threshold points on switching waveform. It is the delay between 50% of input to 50% of output. There two propagation delays are present which are output fall delay(Tf) and output rise delay (Tr).

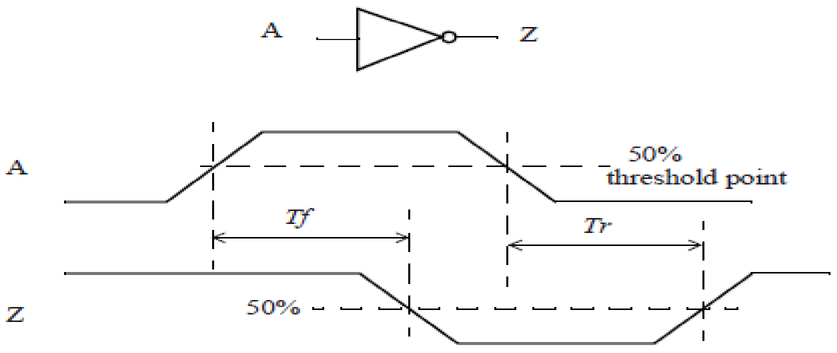


Figure 13: Propagation delays of CMOS inverter.

4.4.3 Slew of a Waveform

In STA the slew of waveform is measured with respect to rise or fall transition. And these transition levels are defined with respect some threshold values. And these transition values are specified in the percent of Vdd. In the below figure the rise transition is in between 30% to 70% and fall transition is in between 70% to 30% of Vdd.

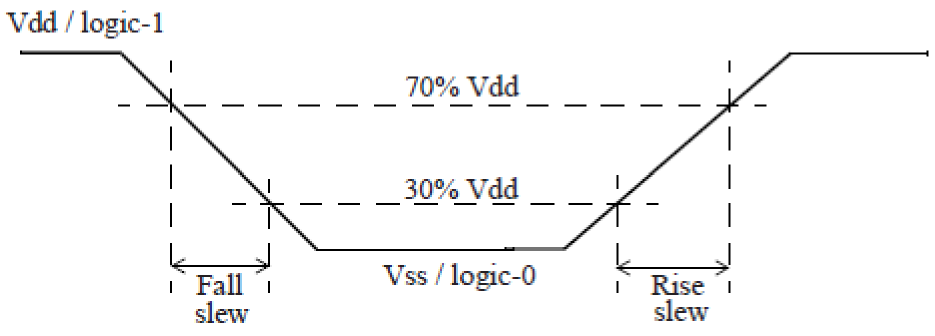


Figure 14: Rise and fall transition times.

4.4 .4 Skew between Signals

Skew is defined as the time difference between arrival and required time of a signal at the sink points i.e sequential flip-flops. In the below figure shown clock latency and clock skew. Normally the starting point of clock tree is a node which is called as clock root point. Clock latency is defined as the total time taken from clock root to all respective sequential sink points in the design.

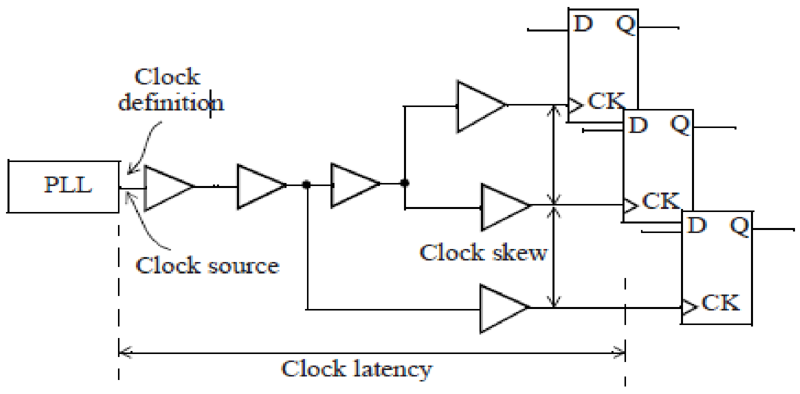


Figure 15: Clock tree clock latency and clock skew.

4.4.5 Timing Arcs and Unateness

These timing arc are useful to calculate the path delay from one point to another point. And these timing arcs of a cell from each input to each output. The timing arc has specific sense i.e how the output transition changes for change input transition. In the following figure shown each cell and timing arc of a cell. Unateness is defined with respect to timing arc only that is if rise transition on input of the cell causes rise transition on the output of the cell then we can call it as positive unate. Similarly negative unate is the opposite transitions on input and output of a cell. In the below figure(c) the output of OR gate is cannot expect the output signal transition if having the one signal transition this is called Non-unate arc.

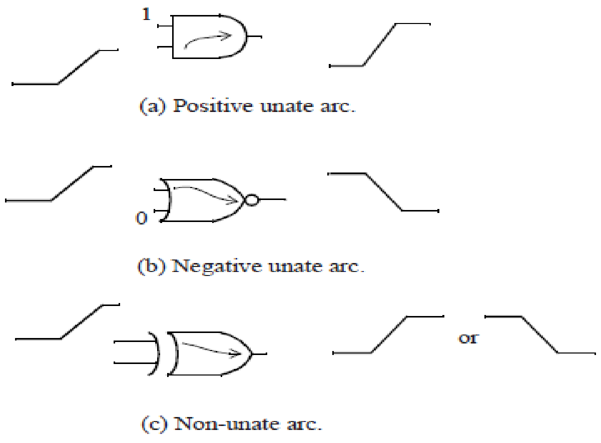


Figure 16: Timing sense of arcs

4.4.6 Minimum and Maximum Timing Paths

The entire delay taken for the signal to reach from start point to endpoint is called path delay. This path delay is summation of standard cell delay, wire delay along the path. Normally, many number of paths from one point to another point in logic path. The paths related to the maximum timing and minimum timing are referred to as the max path and min path respectively. The max path also called as longest path and min path as shortest path.

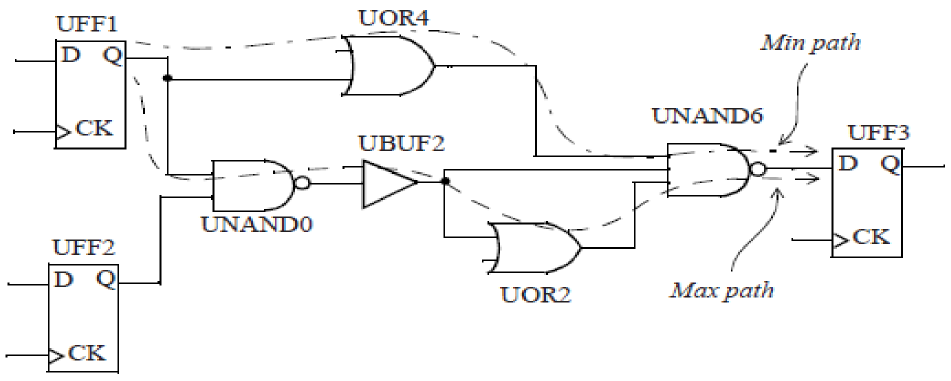


Figure 17: Max and min timing paths.

4.4.7 Clock Domains

Now away days most of the designs are synchronous designs that means clock signal should arrive to all flip- flops so that all flip-flop outputs changes with respect to clock signal. Normally clock feeds many number of flip-flops. The bunch of register driven by one clock is known as clock domain. In the below figure shows two clock domains one is USBCLK clock and another is MEMCLK clock signal.

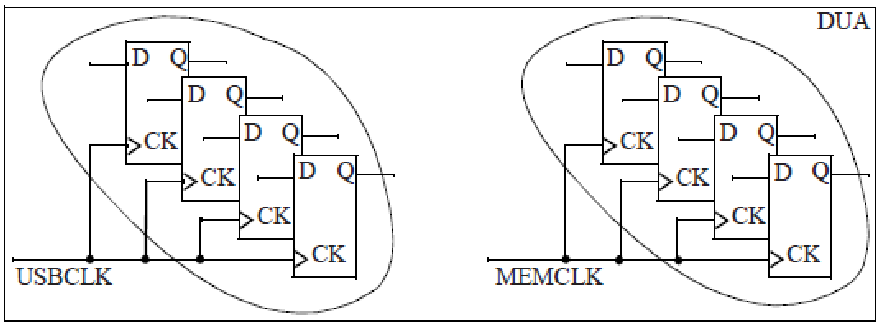


Figure 18: Two clock domains

4.4.8 Operating Conditions

STA is usually executed at a particular operating condition. An operating condition is the mixture of PVT. Standard cell delays and metal wire delays are calculated based on the particular PVT condition. Normally three varieties of process models available in TSMC 180nm technology library. They are *slow*, *typical and fast.* The *slow* and *fast* signify the extreme corners of the manufacturing process of a foundry . For the robust design, the design is validated at the extreme corners of the manufacturing process as well as environment extremes conditions for temperature and power supply . The below Figure (a) shows a cell delay variation with respect to process corners. Figure (b) shows standard cell delays variation with power supply , and Figure (c) cell delays variation with temperature .

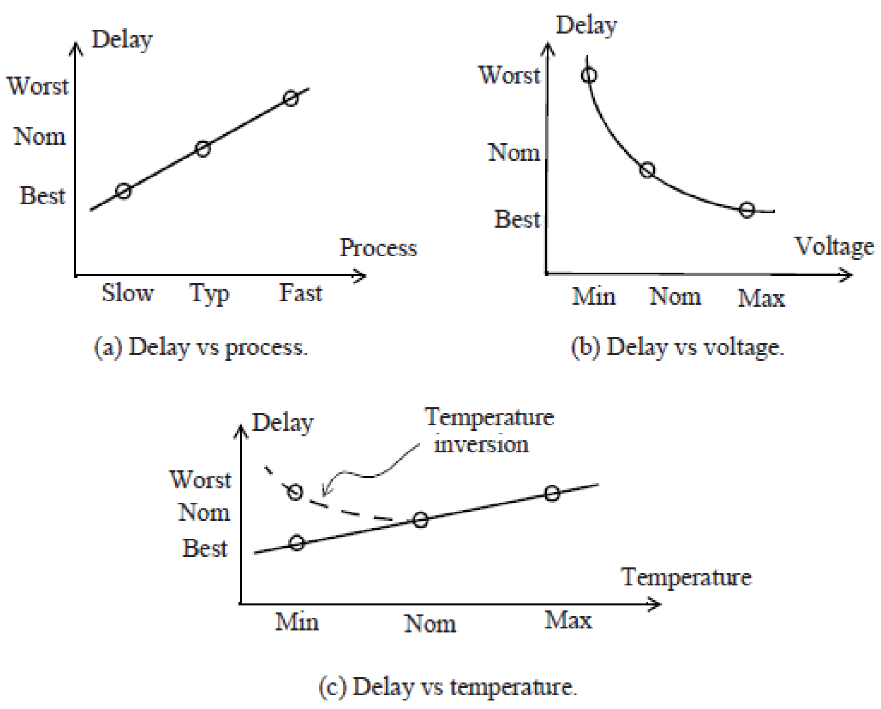


Figure 19: Delay variations with PVT.

Three PVT conditions described below:

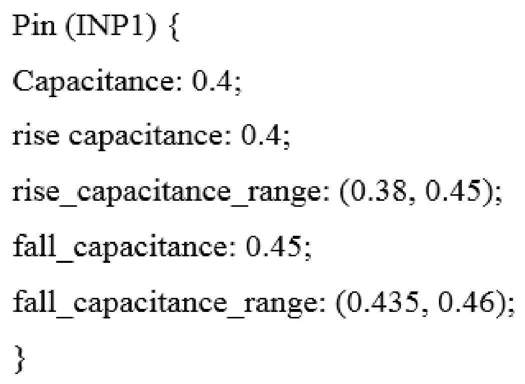
- *WCS (Worst Case Slow)*: Process is *slow* , temperature is highest (say 125C) and voltage is lowest (say 1.62v, that is1.8V minus 10% for 180nm technology).
- *TYP (Typical)*: Process is *typical* , temperature is nominal (say 25C i.e. ambient temperature) and voltage is nominal (say 1.8V).
- *BCF (Best Case Fast)*: Process is *fast*, temperature is lowest (say 0C) and voltage is highest (say 1.98v, that is 1.8V plus 10% for 180nm TSMC technology).

4.5 Standard Cell Library

Standard cell library provides to the designer which contains the information of a cell such as timing, area and functionality of a cell. And the information in the format of liberty syntax.

4.5.1 Pin Capacitance

In the technology library each cell can specify the input and output capacitance but most of library characterized only input pin capacitance and output capacitance is zero.



```
Pin (INP1) {
Capacitance: 0.4;
rise capacitance: 0.4;
rise_capacitance_range: (0.38, 0.45);
fall_capacitance: 0.45;
fall_capacitance_range: (0.435, 0.46);
}
```

The above rise_capacitance means when signal transition is rising then the tool will take this capacitance while calculating the path delay. And similarly fall capacitance is taken when the signal transition from high to low.

4.5.2 Timing Modeling

The standard cell timing models given by the foundry which are provides to the designer exact timing information of standard cells present in the design. Normally the timing information's are extracted from SPICE circuit simulations.

The delay for the standard cell is reliant on below factors:

- The input slew of standard cell.
- The output load capacitance

If the input slew is more, that means transition time is less, so that the output put capacitance charging and discharging phenomenon is very fast, there by the delay of standard cell decreases. And the other scenario is, if the output load is large more time took to charge, there by the delay will increases.

4.5.3 NLDM Models

Normally most of standard cell libraries characterized in the form of NLDM table because which are accurate timing models. And some advance timing models which current source based (CCS and ECSM). This NLDM table models capture standard cell delay through different combinations of input slew and output load. The NLDM model is represented in the form of two dimensional matrix as shown below.

In the below lookup table specifies that the first variable is the input slew and the next variable is the output load. In the below table index_1 is input slew and index_2 is output load, it is in the form of 3x3 matrix. There are 9 combinations for the 3 input slew values and 3 output load values. And in the matrix the values are the standard cell delays. Here the input max input slew is 0.7, if the slew exceeds this value then cell will not provides accurate delay values. Normally if input slew exceeds this max limit, then the tool will put buffer so that slew value decreases.

```
pin (OUT) {
 max_transition : 1.0;
 timing() {
  related_pin : "INP1";
  timing_sense : negative_unate;
  cell_rise(delay_template_3x3) {
   index_1 ("0.1, 0.3, 0.7"); /* Input transition */
   index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
   values (  /*  0.16      0.35     1.43 */ \
    /* 0.1 */   "0.0513,  0.1537,  0.5280", \
    /* 0.3 */   "0.1018,  0.2327,  0.6476", \
    /* 0.7 */   "0.1334,  0.2973,  0.7252");
   }
  cell_fall(delay_template_3x3) {
   index_1 ("0.1, 0.3, 0.7"); /* Input transition */
   index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
   values (  /*  0.16      0.35      1.43 */ \
    /* 0.1 */   "0.0617,  0.1537,   0.5280", \
    /* 0.3 */   "0.0918,  0.2027,   0.5676", \
    /* 0.7 */   "0.1034,  0.2273,   0.6452");
  }
```

Among the synchronous clock and asynchronous clr or set, the asynchronous signals are dominated on the standard cell operation. When asynchronous signals are inactive then the synchronous clock signal comes in picture so that the data will latch in circuit by clock edges. There are two asynchronous constraints checks are available which are called removal and recovery checks. The recovery check is the minimum amount of time that an asynchronous signal is stable after being de-asserted before the next active clock edge. Similarly, the removal check is the minimum amount time after a clock edge that the asynchronous signal must remain active before it can be de-asserted. And the other than this asynchronous checks there are minimum pulse width checks, which is due to unequal rise and fall delay of standard cell. This minimum pulse width checks are relevant to both synchronous and asynchronous signals.

```
pin(CDN) {
  direction : input;
  capacitance : 0.002236;
  . . .
  timing() {
    related_pin : "CDN";
    timing_type : min_pulse_width;
    fall_constraint(width_template_3x1) { /*low pulse check*/
      index_1 ("0.032, 0.504, 0.788"); /* Input transition */
      values ( /*    0.032     0.504    0.788 */ \
                   "0.034,     0.060,    0.377");
    }
  }
  timing() {
    related_pin : "CK";
    timing_type : recovery_rising;
    rise_constraint(recovery_template_3x3) { /* CDN rising */
      index_1 ("0.032, 0.504, 0.788"); /* Data transition */
      index_2 ("0.032, 0.504, 0.788"); /* Clock transition */
      values(  /*      0.032      0.504     0.788 */ \
        /* 0.032 */  "-0.198,    -0.122,  0.187", \
        /* 0.504 */  "-0.268,    -0.157,  0.124", \
        /* 0.788 */  "-0.490,    -0.219,  -0.069");
    }
  }
```

4.6 Timing Verification

In STA there are two most important checks (setup check and hold check) should be verified for the synchronous flip-flop. Once the clock defined at the flip-flop clock input, these two checks are automatically inferred to flip-flop. Generally these timing checks are performed at the multiple scenarios like worst case slow and best case fast conditions. Typically setup check of a flip-flop is checked worst case slow condition and hold check is at best case fast condition.

4.6.1 Setup Check

This check verifies timing correlation between the clock and data pin of the flip-flop such that setup requirement is met. In other words setup check is, data should be stable at the D pin of flip-flop before clock reaches at the clock pin of flip-flop .

In general there are two flip-flops, one is launch flip-flop and another is capture flip-flop . This launch flip-flop launches the data at one clock edge and the capture flip-flop capture the data at next clock edge such that the capture flip-flop should satisfy the setup requirements at this point. The setup equation is given below.

$$T_{launch} + T_{ck2q} + T_{dp} < T_{capture} + T_{cycle} - T_{setup}$$

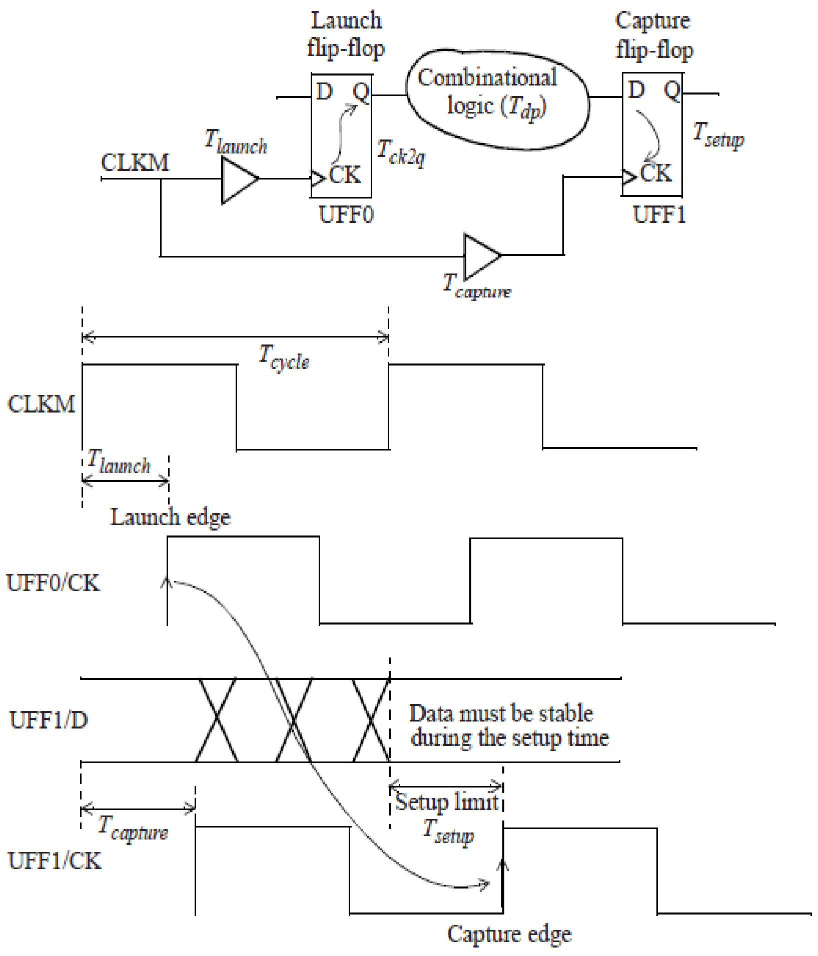


Figure 20: For setup check lock and data signals.

4.6.2 Hold Check

This hold check ensures that the flip-flop output should not change certain time after clock edge so that it will not overwrite the previous data which is latched to this flip-flop. This hold check verifies with best case fast condition. The hold check very critical compared to setup check, if any hold violations in we can’t do anything in our hand, if setup violation is present in the design if you increase the time period then setup violation will resolve. Finally setup check is dependent on time period but not hold time. The hold time equation will be given as follows.

$$T_{launch} + T_{ck2q} + T_{dp} > T_{capture} + T_{hold}$$

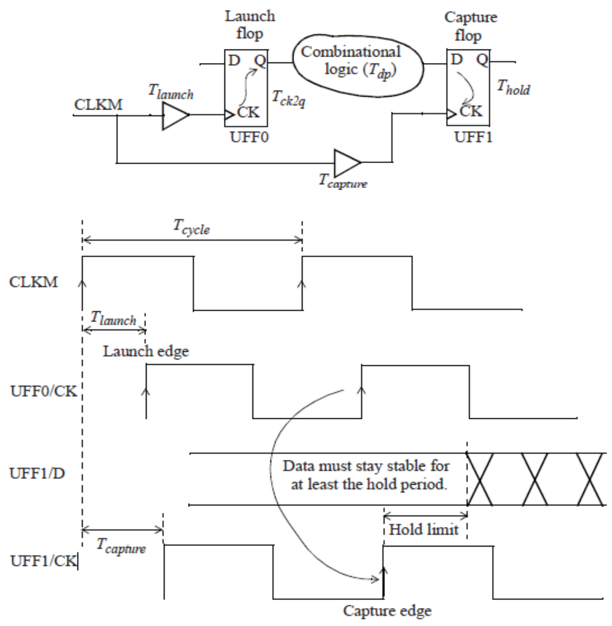


Figure 21: For hold check the clock and data signal

4.6.3 Multicycle Paths

In STA this multicycle paths are timing exceptions, in cases the combinational data path will take more time cycles to pass the logic from input to output. In this cases combinational path declared as multicycle path. We can specify that after certain number of clock cycles the data should be captured. The below figure shows that the combinational logic has taken three cycles to propagate the logic from Q pin of UFF0 to D pin of UFF1 without considering the wire delays

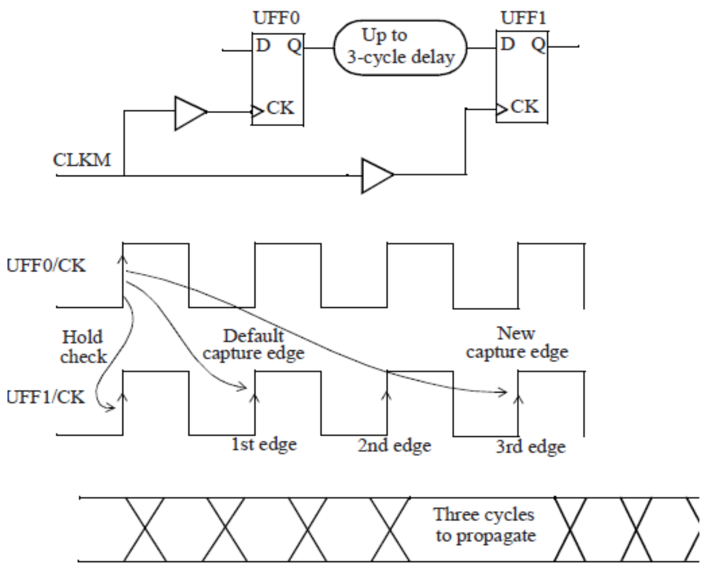


Figure 22: A three-cycle multicycle path.

4.6.4 False Paths

In STA certain paths are ignored that is those paths are not real in the functional operation of the design, such paths are treated as false paths so that time required to analyze the path will be decreased. The best example for false path is, if two asynchronous clock domains are crossing then we can set those are the false path, so that tool will not take account in timing verification.

Chapter 5

5. *Physical design of USB*

5.1 Floor-Planning

Floor-planning is the process of placing the hard macros on the die or within other block, thereby defining routing areas between them. Floor-plan is mapping the logical netlist into physical model in the layout. Because floor-planning significantly affects circuit timing and performance, especially for complicated hierarchical designs, the quality of your floor-plan directly affects the quality of final design

- Calculation of Core, Die size and Aspect Ratio.
- 70% of the core utilization is reasonable so that, design can be timing closure
- Aspect ratio is 1 for square shape
- Initializing the Core
- Rows are flipped, double backed and made channel less.

Aspect Ratio defined as the ratio between the horizontal resources to vertical resources.

Core Utilization= Standard Cell Area / (Row Area + Channel Area).

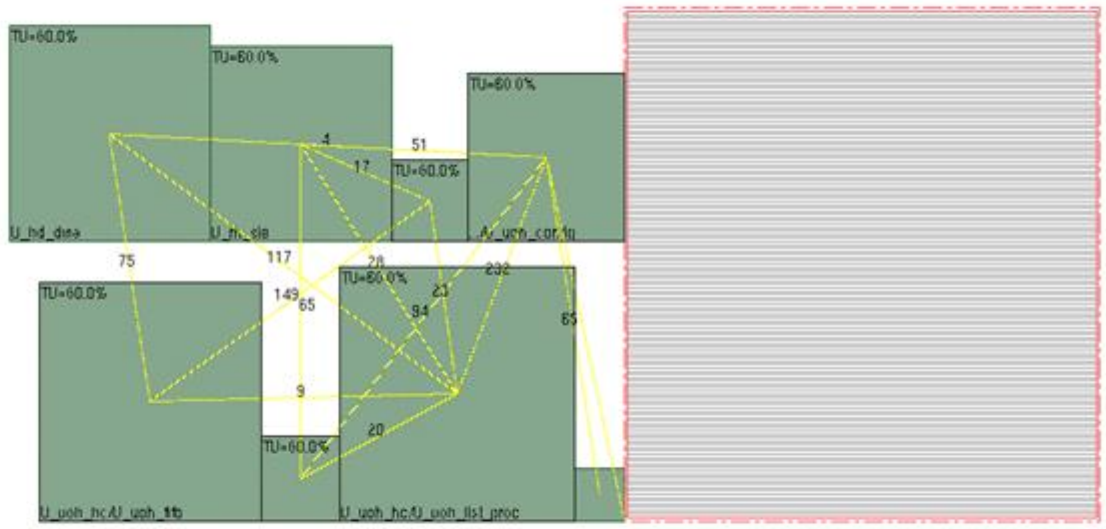


Figure 23: floor-plan view and module interconnection

5.2 partitioning the design

This is the process of dividing the whole core area into small and manageable blocks. This step is very advantage because by partitioning the design we can estimate the different functional block and also the standard cells can be placed easier with less routing distance. Partitioning can be done at RTL stage and then design each module or block separately. This type of partitioning called as logical partitioning.

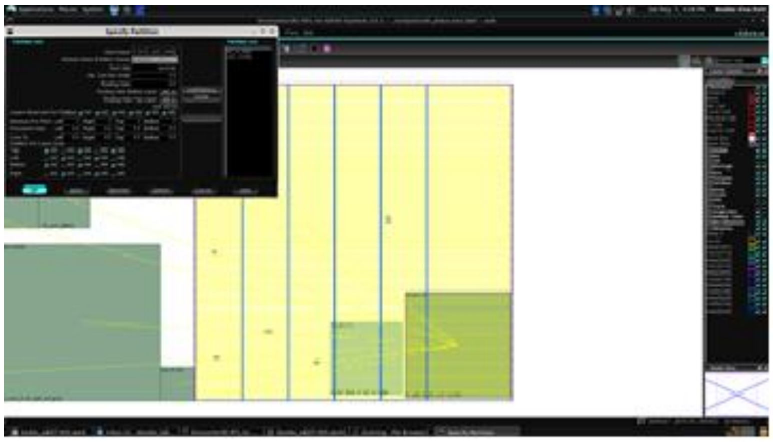


Figure 24: Partitioning the of USB design in SoC encounter GUI

5.3 Power Planning

In this stage we have to create power grid to supply the current to standard cells and macros in the design. Normally for power supply nets have to use higher metal layers so that they allow more current. Here the power supply nets are VDD and VSS. For supply power to core cells have to create power ring around core area. The cells in middle of core may not sufficient power because the voltage drop occurs while current passes through the metal layers. To avoid this problem power stripes will create in vertical direction with higher metals. Here to avoid IR drop and electro migration we need to calculate the power supply net widths such they can supply sufficient power. The calculation of power ring width as follows
"CALCULATION OF CORE RING WIDTH"

a) The width and height of the core area is obtained from estimation sheet

b) Current at the top/bottom and left/right is determined by the following equations

$$I_{top} = I_{bottom} = \{Ict * [Wc / (Wc + Hc)] / 2\}$$

$$I_{left} = I_{right} = \{Ict * [Hc / (Wc + Hc)] / 2\}$$

I_{ct} = Total core current

W_c = Width of core

H_c= height of core

Depending on the EM limit the total width of the stripe of metal required is calculated.

Total width of top and bottom = I_{top}/EM limit

Total width of left and right = I_{left}/EM limit

Width determined by IR drop

It is required that the IR drop should be less than 5% VDD

$$IR\ drop = I*R$$

$$R = R0*(L/2)/W$$

$$W = [R0*(L/2)]/R = [R0*(L/2)]*(I/0.05VDD)$$

L/2 is chosen because the drop is maximum here

$$Hence\ W = I\ \{*R0*(L/2)\}/0.05*VDD$$

Whichever width is limiting that is taken as the core ring width.

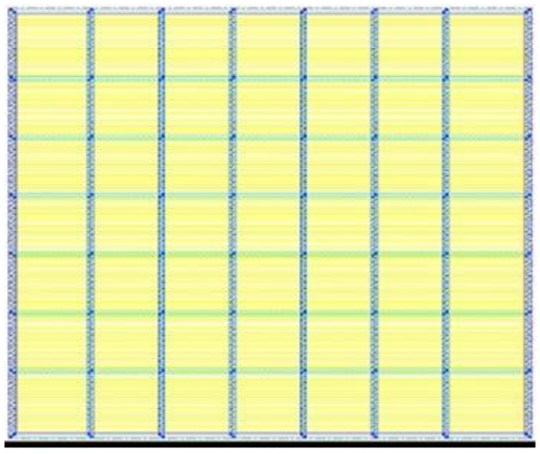
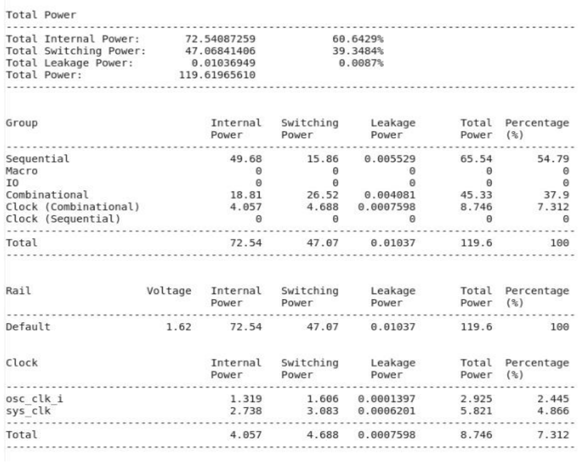


Figure 25: power ring and power stripes for standard cells



```
Total Power
----------------------------------------------------------------------------------------
Total Internal Power:       72.54087259          60.6429%
Total Switching Power:      47.06841406          39.3484%
Total Leakage Power:         0.01036949           0.0087%
Total Power:               119.61965610
----------------------------------------------------------------------------------------


Group                           Internal   Switching      Leakage       Total  Percentage
                                Power      Power          Power         Power  (%)
----------------------------------------------------------------------------------------
Sequential                         49.68       15.86     0.005529       65.54       54.79
Macro                                  0           0            0           0           0
IO                                     0           0            0           0           0
Combinational                      18.81       26.52     0.004081       45.33        37.9
Clock (Combinational)              4.057       4.688    0.0007598       8.746       7.312
Clock (Sequential)                     0           0            0           0           0
----------------------------------------------------------------------------------------
Total                              72.54       47.07      0.01037       119.6         100
----------------------------------------------------------------------------------------


Rail                  Voltage   Internal   Switching      Leakage       Total  Percentage
                                Power      Power          Power         Power  (%)
----------------------------------------------------------------------------------------
Default                  1.62      72.54       47.07      0.01037       119.6         100


Clock                           Internal   Switching      Leakage       Total  Percentage
                                Power      Power          Power         Power  (%)
----------------------------------------------------------------------------------------
osc_clk_i                          1.319       1.606    0.0001397       2.925       2.445
sys_clk                            2.738       3.083    0.0006201       5.821       4.866
----------------------------------------------------------------------------------------
Total                              4.057       4.688    0.0007598       8.746       7.312
----------------------------------------------------------------------------------------
```

Figure 26: power report

5.4 Placement of standard cells

This placement stage is after the floor-planning and power grid creation for standard cells. In this stage the placement of cells is done based on connectivity of netlist. After placing the standard cells we need to check the global route congestion map, it will provide the density regions based pin density and cell density. This congestion map is very useful, by using this we can estimate the design is routable or not in further stage like CTS and routing. Placement stage the tool will divide the total core area into grid cells (Gcells) and each grid cells again divided into bins for routing. The tool will assigns the routing tracks to grid cells, basically the grid cell has a capacity the number of routing tracks it can allow without any shorts and DRC violations. If the number routing tracks is more than the required then it will called as congestion. While placing the standard cells the tool will remove the wire load models (WLM) and uses (virtual route) VR to calculate the RC values for the timing information. Here VR is shortest distance between two points, this is called as Manhattan distance. Here the RC values are more accurate than the wire load model RC parasitcs.

Placement of standard cells is done in 4 optimization stages.

(a). Pre-placement Optimization

In this phase the tool will optimizes the netlist before placement, in design some signals like reset, scan enable are high fan-out nets. If these nets are in data path there by the delay will increase and causes setup violations, so avoid that problem HFNs are collapsed. It can also downsize the cells.

(b).In-placement optimization

In this phase it re-optimizes of the netlist based on VR. Re-optimization like cell sizing, cell moving, cell bypassing, net splitting, gate duplication, buffer insertion, area, cell downsizing recovery.Optimization performs several times to fix the setup , incremental timing and congestion driven placement.

(c).Post placement optimization

After the standard cell CTS will do netlist optimization with ideal clocks before the real clock tree built. It can fix setup, hold, max transition and maximum capacitance violations.in this optimization phase the placement engine will do placement optimization based on global routing. And also it will do HFN synthesis again for better timing. High fan-out synthesis is also called as bufferTreesynthesis.

(d).Post placement optimization after CTS

Here it will optimizes timing with propagated clock means the real clock which having parasitic information. And also it maintains the some clock skew.

5.5 Congestion how to avoid congestion

There some techniques are present to avoid the congestion.

Macro padding:

Normally in a design along with standard cells some hard macros also present, it will create placement halo around the macro so that no standard cell will place around that.by doing this pin connection between standard cells and macro pins is easy.

Placement blockage

This is useful in hierarchical design, if in particular area we avoid the placement of standard cells by creating this placement blockage.

Maximum utilization constraint:

By changing the core utilization factor can maintain the congestion should be small.

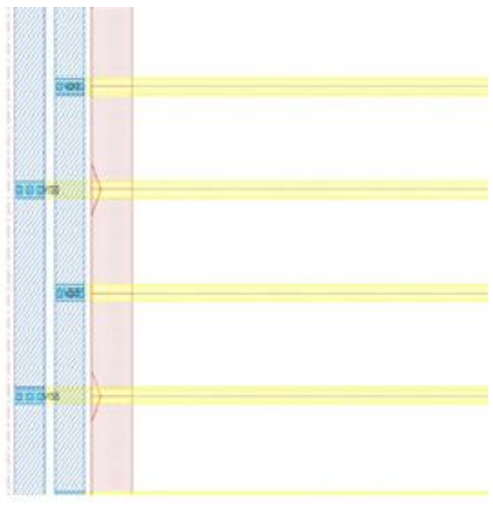


Figure 27: endcap cells

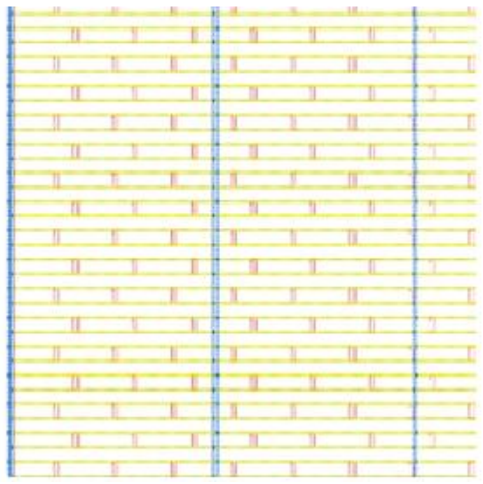


Figure 28: well tap cells in checkerboard fashion

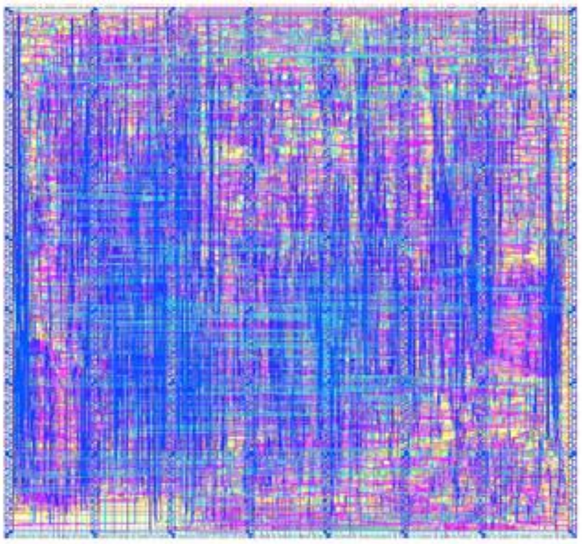
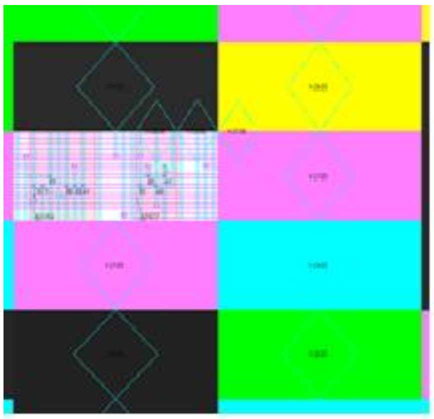


Figure 29: placement of a standard cells



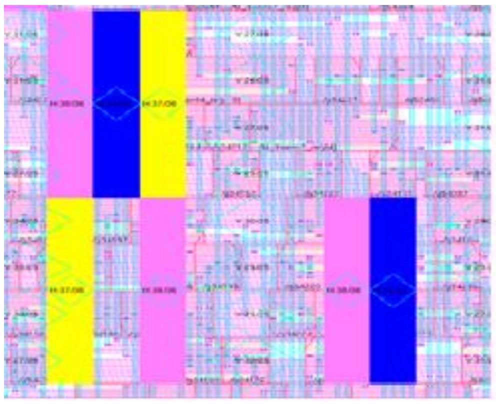
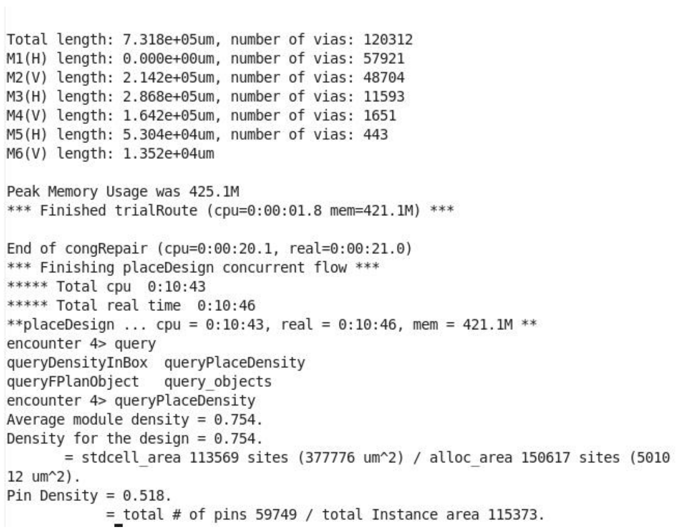


Figure 30: Vertical congestion

Figure 31: Horizontal congestion



```
Total length: 7.318e+05um, number of vias: 120312
M1(H) length: 0.000e+00um, number of vias: 57921
M2(V) length: 2.142e+05um, number of vias: 48704
M3(H) length: 2.868e+05um, number of vias: 11593
M4(V) length: 1.642e+05um, number of vias: 1651
M5(H) length: 5.304e+04um, number of vias: 443
M6(V) length: 1.352e+04um

Peak Memory Usage was 425.1M
*** Finished trialRoute (cpu=0:00:01.8 mem=421.1M) ***

End of congRepair (cpu=0:00:20.1, real=0:00:21.0)
*** Finishing placeDesign concurrent flow ***
***** Total cpu  0:10:43
***** Total real time  0:10:46
**placeDesign ... cpu = 0:10:43, real = 0:10:46, mem = 421.1M **
encounter 4> query
queryDensityInBox  queryPlaceDensity
queryFPlanObject   query_objects
encounter 4> queryPlaceDensity
Average module density = 0.754.
Density for the design = 0.754.
       = stdcell_area 113569 sites (377776 um^2) / alloc_area 150617 sites (5010
12 um^2).
Pin Density = 0.518.
            = total # of pins 59749 / total Instance area 115373.
```

Figure 32: placement density

5.6 Clock Tree Synthesis

This CTS phase will do after placement of the standard cells in the core area of the chip.CTS is a process of balancing clock skew and minimizing insertion delay in order to meet timing, power requirements and other constraints. After placement we will have the positions of standard cells and hard macros, but the clock network will be ideal. In this CTS the actual propagated clock comes in picture i.e this is real routing of clock network. In this the CTS engine will insert clock buffers and clock inverters such the setup and hold violations will be fixed. If the data path delay is more compared to clock time period there will be setup violation will occurs, to fix this we have to minimize the data path delay. For minimizing the data path delay the CTS engine will insert clock buffers and also put higher drive strength gate so that it passes more current and there by the delay will decrees. And Clock tree synthesis provides the following features to achieve timing closure:

- Global skew clock tree synthesis
- skew clock tree synthesis
- Real clock useful skew clock tree synthesis
- Interclock delay balance
- Splitting a clock net to replicate the clock gating cells
- Clock tree optimization
- High-fan out net synthesis.
- Concurrent multiple corners (worst-case and best-case) clock tree synthesis.
- We can do skew grouping to minimize the skew between the flip-flops which are having same insertion delay.

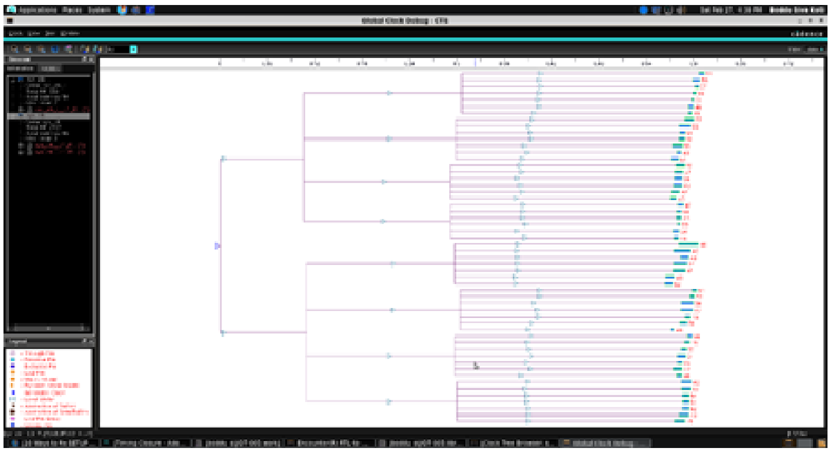


Figure 33: clock tree and sinks of sys_clk

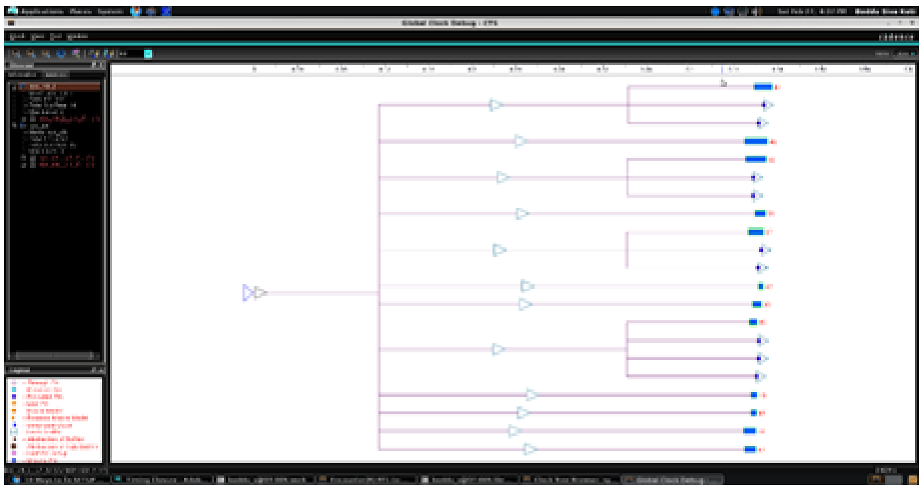


Figure 34: clock tree and sinks of osc_clk_i

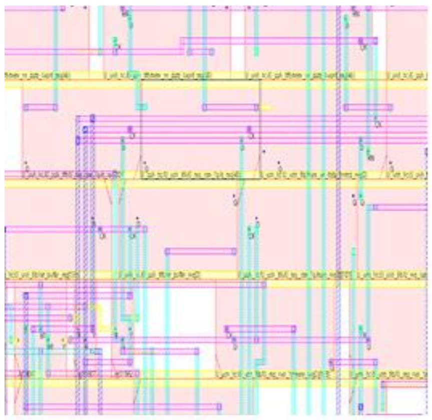
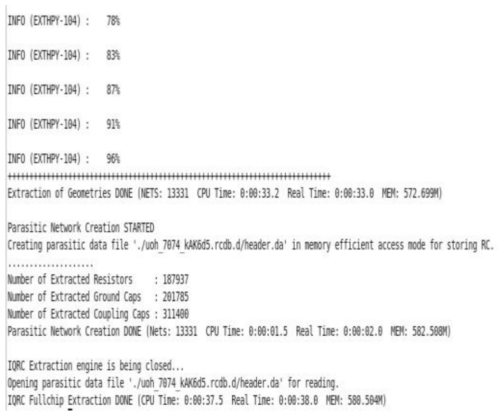


Figure 35 :   Clock shielding



```
INFO (EXTHPY-104) :    78%

INFO (EXTHPY-104) :    83%

INFO (EXTHPY-104) :    87%

INFO (EXTHPY-104) :    91%

INFO (EXTHPY-104) :    96%
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Extraction of Geometries DONE (NETS: 13331  CPU Time: 0:00:33.2  Real Time: 0:00:33.0  MEM: 572.699M)

Parasitic Network Creation STARTED
Creating parasitic data file './uoh_7074_kAK6d5.rcdb.d/header.da' in memory efficient access mode for storing RC.
....................
Number of Extracted Resistors     : 187937
Number of Extracted Ground Caps   : 201785
Number of Extracted Coupling Caps : 311400
Parasitic Network Creation DONE (Nets: 13331  CPU Time: 0:00:01.5  Real Time: 0:00:02.0  MEM: 582.508M)

IQRC Extraction engine is being closed...
Opening parasitic data file './uoh_7074_kAK6d5.rcdb.d/header.da' for reading.
IQRC Fullchip Extraction DONE (CPU Time: 0:00:37.5  Real Time: 0:00:38.0  MEM: 580.504M)
```

Figure 36: extracted parasitics

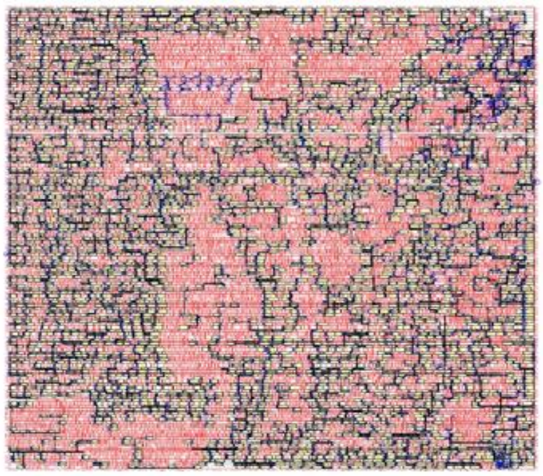


Figure 37: clock routing to all sinks

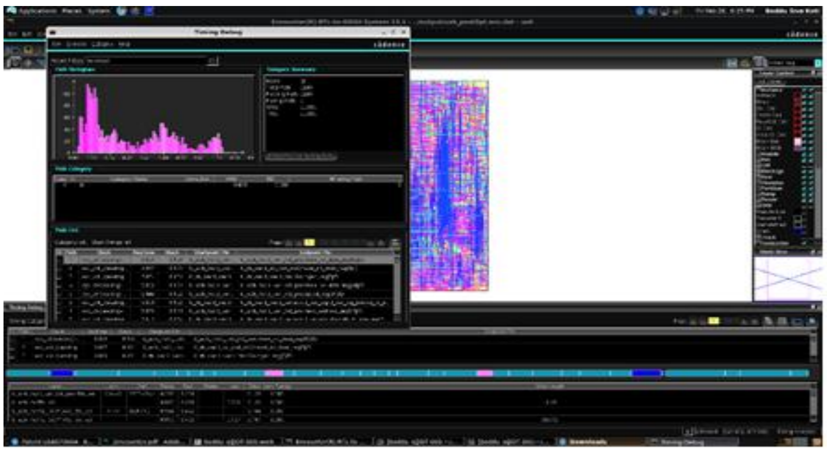


Figure 38: timing slack histogram

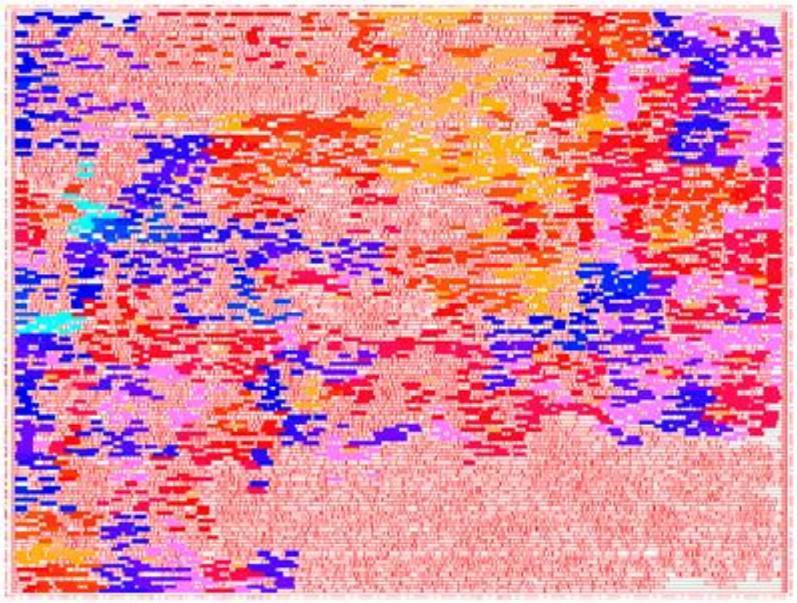


Figure 39: Max and min clock routing distance

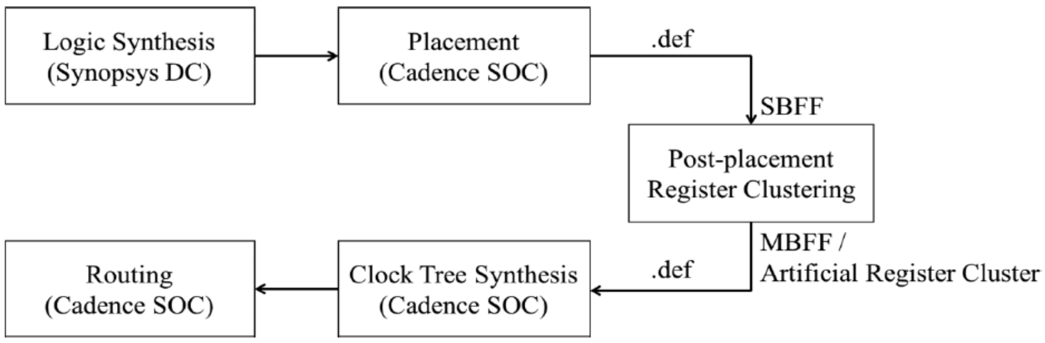
Chapter 6

6. Register clustering algorithm

6.1 Introduction.

The idea behind group the registers which having similar characteristics, so that that clock network capacitance greatly reduced, thus switching power of the clock network should be minimized. In previous so many techniques are present to do register clustering at the postplacement phase. There are some techniques based on Euclidian distance grouping of same leaf cluster. This type of algorithm iteratively calculates the Euclidian distance and forms the clusters. By grouping the registers we can minimize the local clock tree capacitance.

propose a novel latch placement methodology to minimize local clock-tree capacitance as technology development improves the driving strength of inverters inside registers, it is now possible to share common inverters in several flip-flops (FFs), resulting in the multibit flip-flop (MBFF). The MBFF clustering problem is to effectively and efficiently merge several single-bit flip-flops (SBFFs) into an MBFF. Compared with traditional register-clustering techniques, the emerging MBFF technique leads to better power reduction. MBFF reduces the clock load by having a single clock input pin, reducing the corresponding wire load. In addition, shared clock inverters within an MBFF further reduce power consumption compared with SBFF.



Logic Synthesis (Synopsys DC)
Placement (Cadence SOC)
.def
SBFF
Post-placement Register Clustering
MBFF / Artificial Register Cluster
.def
Clock Tree Synthesis (Cadence SOC)
Routing (Cadence SOC)

Figure 40: Post placement register clustering flow

There are two ways we can do register clustering at the post placement stage of physical design

1. MBFF (Multibit flip flop) register clustering
2. Artificial Register clustering (DBSCAN algorithm).

6.2 MBFF register clustering

The device variations can be reduced effectively by replacing the minor flip-flop into bigger multibit flip-flop. By reconciliation the 1-bit flip-flop into multibit flip-flop it avoids the inverters. In a specified time interval the rising and falling, the least sized inverter is enough the to measure the driving capability of clock buffer in Multibit flip-flop configuration. In below Figure clearly explained the highest number of least-sized inverters that can be driven by a clock buffer in different processes . For this reason that of, several flip-flops and shares a common clock buffer to avoid unnecessary power waste. The right figure shows the outlook of 1 and 2-bit flip-flops. If we replace the two 1-bit flip-flops as shown in left side figure by the 2-bit flip-flop as shown in right side figure, the total power consumption can be reduced because the two 1-bit flip-flops can share the same clock buffer.

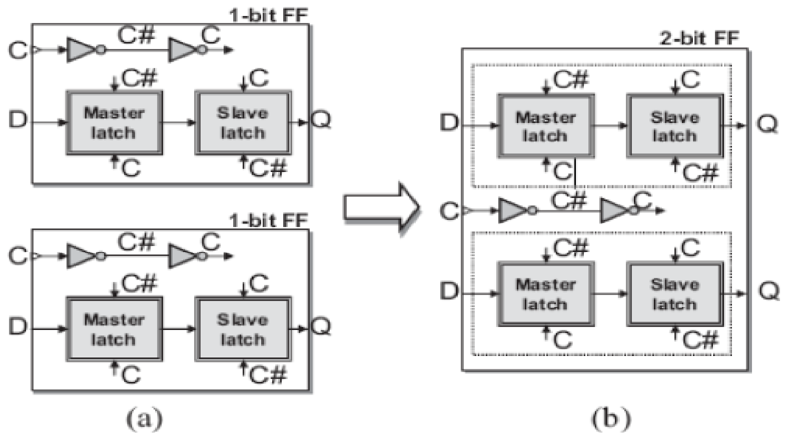


Figure 41: Illustration of mergingstwos1-bitsflip-flopssintosones2-bitsflip-flop.

(a)The flip-flops before merging (b) the flip-flops after merging.

Lexicographical order : <1,100,100>, <2,172,192>, <4,312,285>

Bit NO	Power	Area	Normalized power per bit	Normalized area per bit
1	100	100	1.00	1.00
2	172	192	0.86	0.96
4	312	285	0.78	0.71

Table 3: Illustration advantage of MBFF

6.3 DBSCAN algorithm

This is data clustering algorithm proposed by Martin Ester, HansPeter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a density based clustering algorithm given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbours), marking as outliers points that lie alone in low density regions (whose nearest neighbours are too far away). DBSCAN is one of the most common clustering Algorithms.

This algorithm needs two parameters ε (eps) and radius of the cluster (r). Here eps means that the least number of points to form a cluster (dense area) that is grouping of registers in core area. And r value is the required to form a big cluster or small cluster that means if the register or flip-flop inside the specified radius then that flip-flop is a part of the cluster. In this algorithm the software will select the randomly one flip-flop and calculate Euclidian distance from that point to nearest neibouring registers, if the nearest register is within radius then the random selected point consider as core point. Like this iteratively will calculate the Euclidian distance and forms a clusters which are similar characters. Here I implemented this algorithm in python software and the input are given in the form of excel sheet, the inputs are the placement location a flip-flop and the pin capacitance of specified registers. So that the load capacitance will balance there by skew minimization, clock network capacitance will reduce. The switching power is mainly depends on the clock capacitance. From this algorithm we can minimize the power, area and skew. In recent technology advancement the skew affects the dominantly, if skew will be more there the circuit performance will degrades. This algorithm implemented at the postplacement stage. After placing the standard cells we will extract the placement coordinates of the registers from the .def file and the pin capacitances from the library (.lib), here TSMC 180nm technology library is used. And the tool is used for physical design is SoC encounter from cadence.

Algorithm:

- Input data S={x1,x2……………………xn}€ R
- Choose values for r>0,Eps>0.
- Ai={x € S:d(xi,x) <Eps};i=1,2,3………..n
- If |Ai| <r, we shall not involve this Ai in our calculations.
- Take union of Ai and Aj if Ai∩Aj=ɸ

- Repeat step 4 till no union takes place.

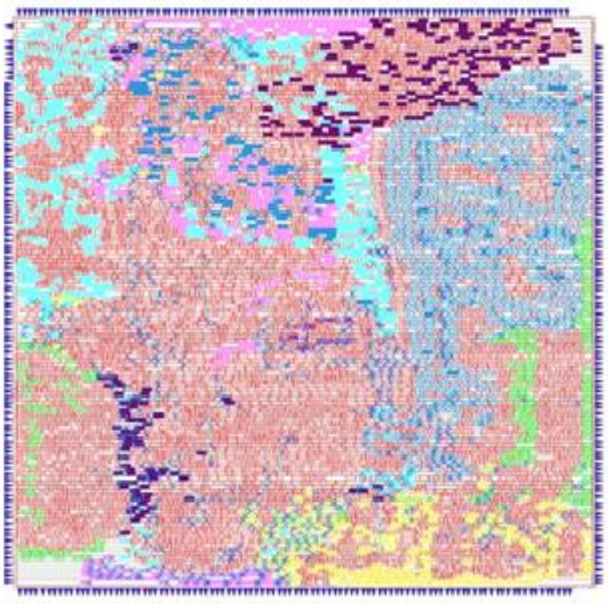
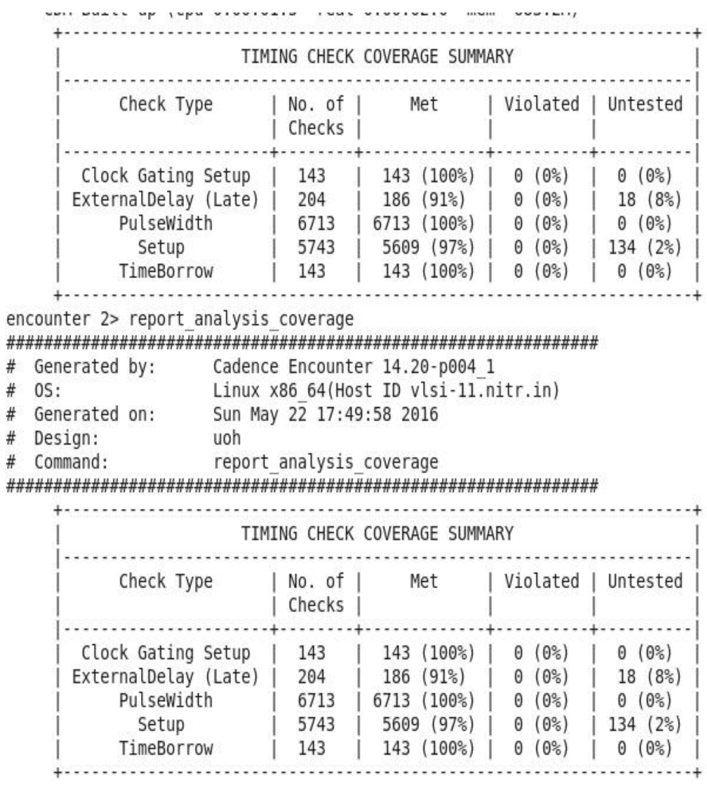
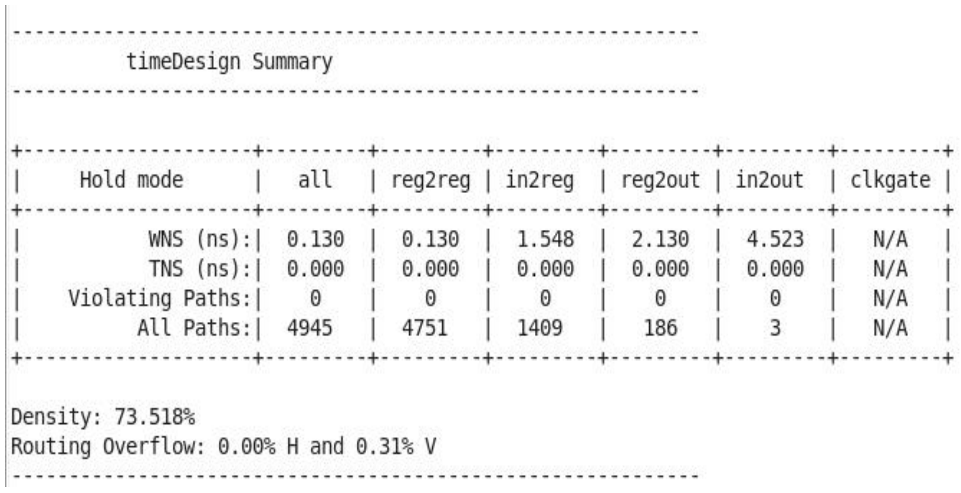


Figure 42: Register clustering



```
     +-----------------------------------------------------------------+
     |                   TIMING CHECK COVERAGE SUMMARY                 |
     |-----------------------------------------------------------------|
     |      Check Type      | No. of |     Met     | Violated | Untested |
     |                      | Checks |             |          |          |
     |----------------------+--------+-------------+----------+----------|
     |  Clock Gating Setup  |  143   |  143 (100%) |  0 (0%)  |  0 (0%)  |
     | ExternalDelay (Late) |  204   |  186 (91%)  |  0 (0%)  |  18 (8%) |
     |      PulseWidth      |  6713  | 6713 (100%) |  0 (0%)  |  0 (0%)  |
     |        Setup         |  5743  |  5609 (97%) |  0 (0%)  | 134 (2%) |
     |      TimeBorrow      |  143   |  143 (100%) |  0 (0%)  |  0 (0%)  |
     +-----------------------------------------------------------------+
encounter 2> report_analysis_coverage
###############################################################
#  Generated by:      Cadence Encounter 14.20-p004_1
#  OS:                Linux x86_64(Host ID vlsi-11.nitr.in)
#  Generated on:      Sun May 22 17:49:58 2016
#  Design:            uoh
#  Command:           report_analysis_coverage
###############################################################
     +-----------------------------------------------------------------+
     |                   TIMING CHECK COVERAGE SUMMARY                 |
     |-----------------------------------------------------------------|
     |      Check Type      | No. of |     Met     | Violated | Untested |
     |                      | Checks |             |          |          |
     |----------------------+--------+-------------+----------+----------|
     |  Clock Gating Setup  |  143   |  143 (100%) |  0 (0%)  |  0 (0%)  |
     | ExternalDelay (Late) |  204   |  186 (91%)  |  0 (0%)  |  18 (8%) |
     |      PulseWidth      |  6713  | 6713 (100%) |  0 (0%)  |  0 (0%)  |
     |        Setup         |  5743  |  5609 (97%) |  0 (0%)  | 134 (2%) |
     |      TimeBorrow      |  143   |  143 (100%) |  0 (0%)  |  0 (0%)  |
     +-----------------------------------------------------------------+
```

Figure 43: report coverage analysis



```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.130  |  0.130  |  1.548  |  2.130  |  4.523  |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |    0    |    0    |   N/A   |
|          All Paths:|  4945   |  4751   |  1409   |   186   |    3    |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

Density: 73.518%
Routing Overflow: 0.00% H and 0.31% V
------------------------------------------------------------
```

Figure 44: summary of timing report

Chapter 7

7. Conclusion and future scope

7.1 Conclusion

In this we applied different synopsis design constraints to the design at the synthesis stage itself there by generated gate level netlist and output constraints by considering the area, power and timing. For synthesis we used RTL compiler and the technology library was TSMC 180nm. And also optimized the gate level netlist by boundary optimization, grouping the instances.

In this stage estimated the chip area and done power analysis by considering the elecrtomigration and IR drop effects. And analysed the congestion effects and applied various techniques to reduce the congestion. Proposed register clustering algorithm to reduce the clock network capacitance there by switching power reduced. Timing closure is achieved by considering the on chip variations and PVT conditions up to routing.

7.2 Future scope

After detailed routing achieving the timing closure by considering the cross talk noise and fixing the any design rule violations. Exploring the Post layout simulation, DFT and scan insertion in the design. Finally generation of GDSII file for fabrication of chip.

Bibliography

[1].W. Hou, D. Liu, and P.-H. Ho, “Automatic register banking for low power clock trees,” in *Proc. Quality Electron. Design*, San Jose, CA, Mar. 2009, pp. 647–652.

[2].Chang Y.-T, Hsu C.-C, Lin P.-H,Tsai Y.-W, and Chen S.- F(Nov. 2010) “*Post-placement power optimization with multi-bit flip-flops*,” in Proc. IEEE/ACM Computer Aided Design Int. Conf., San Jose, CA,pp. 218–223.

[3]. Charles Alpert, Andrew Kahng, Gi-Joon Nam, Sherief Reda, and Paul Villarrubia. 2005. A semi-persistent clustering technique for VLSI circuit placement. In *Proceedings of the 2005 International Symposium on Physical Design (ISPD’05)*. ACM, New York, NY, 200–207.
[4]. Zhi-Wei Chen and Jin-Tai Yan. 2010. Routability-driven flip-flop merging process for clock power reduction. In *2010 IEEE International Conference on Computer Design (ICCD’10)*. 203–208.

[4]. M. P.-H. Lin, Chih-Cheng Hsu, and Yao-Tsung Chang. 2011. Post-placement power optimization with multibit flip-flops. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 12, 1870–1882.

[5]. Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang, “Power-aware placement,” in *Proc. Design Autom. Conf.*, Jun. 2005, pp. 795–800.

[6]. Cadence online support *http://support.cadence.com/*

[7]. Static timing analysis for nanometre designs a practical approach by J. Bhasker Rakesh Chadha. Springer edition ISBN 978-0-387-93819-6,

[8]. SoC encounter user guide for physical design flow

[9]. Primetime user *guide for* timing closure in the design flow.

[10]. For VLSI basic concepts http://www.vlsi-expert.com/p/vlsi-basic.html

[11].For basic physical design concepts Www.vlsi.pro/category/back-end/physical-design-pnr/

[12]. Jin-Tai Yan and Zhi-Wei Chen. 2010. Construction of constrained multi-bit flip-flops for clock power reduction. In *International Conference on Green Circuits and Systems (ICGCS’10)*. 675–678.

[13]. J. Z. Yan, C. Chu, and Wai-Kei Mak. 2011. Safe Choice: A novel approach to hypergraph clustering for wire length-driven placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 7, 1020–1033.

[14]. Campello, R. J. G. B.; Moulavi, D.; Sander, J. (2013). *Density Based Clustering Based on Hierarchical Density Estimates*. Proceedings of the 17th Pacific Asia Conference on Knowledge Discovery in Databases, PAKDD 2013. Lecture Notes in Computer Science 7819. p. 160. doi:10.1007/9783642374562_ 14. ISBN 9783642374555.

[15]. Sander, Jörg (1998). *Generalized Density Based Clustering for Spatial Data Mining*. München: Herbert Utz Verlag. ISBN 3896754696.

[16]. K. Usami and M. Horowitz, “Clustered voltage scaling technique for low-power design,” in *Proc. Int. Symp. Low Power Electron. Design*, 1995, pp. 3–8.

[17]. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In KDD, pages 226–231, 1996.

[18].USB functional operation http://www.usb.org/home

[19].RTL compiler user guide for synthesis from the cadence

[20]. Kawagachi H. and Sakurai T(Jun.1997)“*A reduced clock swing flip-flop (RCSFF) for 63% clock power reduction*,” in VLSI Circuits Dig. Tech. Papers Symp., pp. 97–98.

[21]. Duarte D, Narayanan V and Irwin M. J (Apr. 2002) “*Impact of technology scaling in the clock power,*” in Proc. IEEE VLSI Comput. Soc. Annu. Symp, Pittsburgh, PA, pp. 52–57.

[22]. The Hamada M C. K,, Fujita T, Hara H, Ikumi N, and Oowaki Y(Dec. 2006) “*Conditional data mapping flip-flops for low-power and high-performance systems*,” IEEE Trans.Very Large Scale Integration. (VLSI) Systems, Vol.14, pp.1379–1383.

[23]. D.-J. Lee and I. L. Markov. Obstacle-aware clock-tree shaping during placement. In *Proceedings of the International Symposium on Physical Design*, 2011.

[24]. U. Soma Naidu, K. Venkateswarlu, “A Novel Approach POWER OPTIMIZED MULTI-BIT FLIP-FLOPS USING GATED DRIVER TREE” in International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering *Vol. 2, Issue 6, June 2013*

[25]. Ya-Ting Shyu, Jai-Ming Lin, Chun-Po Huang, Cheng-Wu Lin, Ying-Zu Lin, and Soon-Jyh Chang, “Effective and Efficient Approach for Power Reduction by Using Multi-Bit Flip-Flops”, *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION* (VLSI) SYSTEMS, VOL. 21, NO. 4, APRIL 2013.

[26]. Shefali Sharma, Bipan Kaushal,“Shift Register Design Using Two Bit Flip-Flop” Proceedings of 2014 RAECS UIET Panjab University Chandigarh, 06 – 08 March, 2014

[27]. Jhen-Hong He1, Li-Wei Huang2, Jui-Hung Hung3, Yu-Cheng Lin4, Guosyuan Liou5, Tsai-Ming Hsieh “Clock Network Power Saving Using Multi-Bit Flip-Flops in Multiple Voltage Island Design” in Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013).