10 days - Workshop (VLSI → Silicon Valid chip ⇒ designing a processer. RISC-V) (100MHz - 180MHz)

⇒ we are designing a processer.

⇒ GCC → C compiler in the linux (Application is written in the C language)

01 step-1 we will take a code (Application) and learn in the GCC

step-2 some code in the spec (C model)

if 00 = 01 → After meeting

step-2 → we will write a soft copy of the hole code using RTL (Verilog)

02 have to check 01 == 02

SoC design

Processor → GLN (Synth P1) → should be synthisible to gates

Peripherals (IPs ASSC

Macros (memory) (Synth RTL) → pre defined use the instance

Synthesis

Analog IPs (Timer etc) ADC, ~~Phase lock loop~~ (PLL) clock multiplier.

need not be synthisible. we will explore by mosfets

03 → step-4 → SoC integration → check 01 == 02

Next step ⇒ APR (RTL2GDS) (only one can act on the whole chip)

GDSII → fab (chip out)

04 Peripherals (like the (USB, pots)

We will use in Code in C for any application & general output

O1 == O2 == O3 == O4

O → output

[100MHz - 130MHz]

↳

Application = Arduino boards
TV panel

1. Vcc
1113 outputs
28 pin
2 special pin
4 pin total

Package QFN-48
Quad flat no-lead

(not board) Arduino → chip design

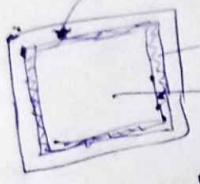
9x9

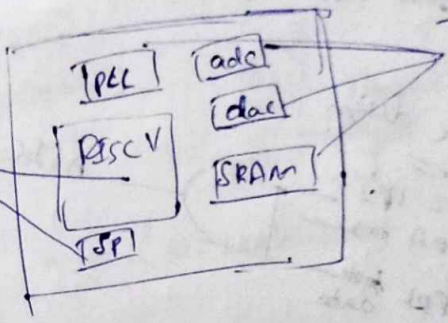
1mm

9mm → Wire bond (for Connection)

→ chip

Pads → Sends signals from inside to outside & vice versa
chip to outside (package & world)
outside to chip, outside the chip





foundry IP's → fab (where chip gets manf)

Intellectual property (It needed some intelligence to build this blocks)

RISC-V (Instruction set architecture (ISA)

Architecture to RTL → Layout (implementation) flow

C language (spec)

Compiler → Converts the code into instructions (based on the hardware) (.exe file)

OS

Assembler → (converts the commands to instructions) (binary language)

↓ given to hardware

instruction set is called abstract interface

instruction set architecture (ISA) of computer

acts as the mediator.

Instruction set

↓

Hardware language

↓

Synthesis to Gate level.

↓

APR

Pdt ① → RISC V

Pdt ② → RTL & Synthesis of RISC-V based CPU Core - picorv32

Pdt ③ → APR

Digital ASIC design

RTL IP's
EDA tools
PDK data

→ ASIC

RTL designs

- libreCores.org
- OpenCores.org
- github.com

→ Open source

EDA tools

- Qflow
- OpenLANE
- OpenROAD

PDK (Process design kit)

+ Google

→ Skywater

for 130nm Predictive PDK

↳ the interface b/w the FAB & the designer

Contents

- Collection of files used to model a fab process for the EDA tools used to design
- process design rule : DRC, LVS. ..
- device model
- Digital standard cell library
- IO library. -

is 130nm fast
- not complicated to new
- very cheap
- G.I. Shale value

Yes (may be)
- Eg: 130nm

intel : PENTIUM @ 3.46GHz (Q4QM)
single cycle RV32i design (300-400 MHz)
(155,000 - 200,000) ← Area (micron)

APR
flow explained (openflow verify)
from synthesis to signoff

openLANE → good for flow configuration
6 layers
sky130nm (area M5 lithium metal M0 aluminium aluminium)

(LEC)
Yosys (RTL to GLN) (Synthesis)
openSTA (STA) (OpenROAD)
fault (DFT)
openRoad (APR)
magic (Verification)
netgen (RC extraction)
abc → strategies
→ targets

openlane (like a container)

New step (*)
fake ant. diodes insertion script
- add a fake antenna diode next to every cell after placement
- & run antenna checker (magic)
- → if we are getting violation then add the real antenna diode.

↓ Before routing
↓ after the detail routing
↓ fake ant diodes swapping script.

PDKs → Skywater 130nm
Skywater

(.lib, .lef, .tech.lef, .ef)

* Lib.lef → timing
Lib.tech → tool related information

Picorv32a → design

src → RTL.(V)
↳ SDC

*** Config.tcl
↳ Clock period ≤ (clk)

Command
↳ Carrying all the Commands.

day① → done Synthesis, Ground
find the dff ratio(.1.) to

dffratio ⇒ = ff / total Cells × 100.

task①

=

No of Cells = 14876 = 1613 / 14876 × 100
dff = 1613 = 10.8%.

Day - 2

defining the die & Core

→ ~~Big~~ rec defining width & height of Core

⇒ Core & die.

Utilization factor = $\frac{\text{Area occupied by netlist}}{\text{Total Area of Core.}}$

100% = 1 = utilization factor

ideal utilization = 100%.

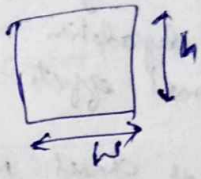
practical utilization = 80 - 85%.

good to start at = 60 - 70%.

Aspect ratio = h/w

h = height
w = width

1 ⇒ Square
<1> Rectangle.



② define location of preplaced cells. → Macros, I/P's

Partitioning → This is where several

⇒ macro guidelines (explained)

→ decoupling capacitance

⇒ we have to surround the decoupling capacitance around the macros.

→ floorplan (Core Configuration)

Readme file

↳ we will have some details in this

→ Powerplan

↳ we will place the ~~metal~~ wire for the connection of the power & ground pins of all the cells.

→ Need for characterization of libraries

Cell design flow

inputs ⇒ process design kits (PDK):

from foundry: DRC & LVS rules, spice models
library & user-defined specs
↳ height-cell, supply voltage, no-layers
pin locations etc.

Spice models contains
↳ I-V characteristics
↳ Threshold voltage (V_{th})
↳ mobility degradation
↳ short channel effects
↳ C
↳ short short channel modulation.

typical spice model looks like

```
.model nmos nmos level = 54
+ vth0 = 0.45  u0 = 0.03  tox = 1.4e-9
+ rdsw = 150   eta0 = 0.01  k1 = 0.5  k2 = 0.02
```

design steps

↳ Circuit design, layout design → we will implement the things in the layout.
↳ characterization
↳ design the circuit with the respects.

W of nmos
W of pmos

outputs
↳ CDL (Circuit description language) (Circuit design)
↳ GDS II, lef, RC extraction (layout design)
↳ timing, noise, power libs, function.

⇒ Mos diagram → Euler path & draw stick diagram
⇒ & after we will design the layout in the tool using all the DRC & LVS rules.

→ After characterization. (Guna software)

↳ It is the process of generating the timing & function of the design in the standard format called .lib.

Circuit & layout (or) Netlist

↓

RC → Delay Calculator → .lib (SDF)

↓

[redacted]

⇒ for running the floorplan, we use

run_floorplan

→ we will set all the required switches before running the floorplan.

⇒

⇒ shift+n (to go to last line of the script)

f+v → align layout to center

z → Zoom in

shift+z ⇒ Zoom out

→ legalization

Day-3

→ IO placement division

→ Spice deck Creation for CMOS inverter

→ extract

↳ for extracting the design

→ ~~extract~~

ext2spice cthresh 0 rthresh 0

↓

extracting to spice.

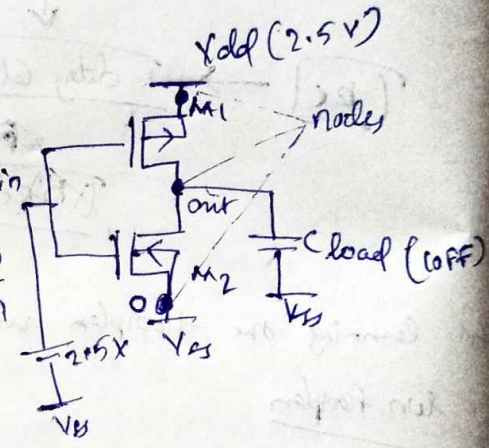
Spice deck

↳ Component Connectivity

↳ Component Values => 0.375u / 0.25 = 1.5 / 1 (both PMOS & NMOS)

↳ identify "nodes"

↳ name "nodes"



** model description ***

*** Netlist description ***

M1 out in Vdd Vdd PMOS W=0.375u L=0.25u

M2 out in 0 0 NMOS W=0.375u L=0.25u

D G S B

Cload out 0 10f

Vdd vdd 0 2.5

Vin in 0 2.5

*** Simulation Commands ***

.op

.dc Vin 0 2.5 0.05

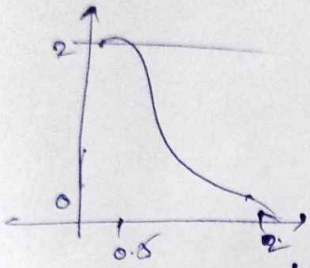
*** .include 45nm_025um_model.mod ***

.LIB "45nm_025um_model.mod" CMOS_MODELS

.end

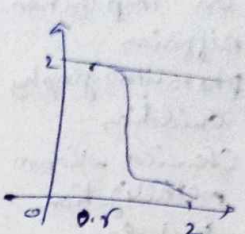
Spice waveform : $W_n = W_p = 0.375u$, $L_{n,p} = 0.25u$ device

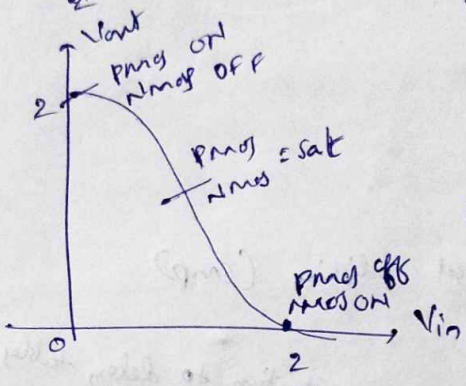
($W_n/L_n = W_p/L_p = 1.5$)



Static

$W_n = 0.375$, $W_p = 0.9375u$





$V_{in} = V_{out}$

→ 16-mask Cmos process

linden length

split

---

list → for creating the list

split → for splits a string into a list

join → joins list elements into a string

Concat → Concatenates lists

→ length

→ linden (returns a element at a given index)

→ lenge

→ linsert

→ leeplace

→ lset (sets/modifies a specific list elements)

→ lsearch

→ isort (sorts a list) (increasing order, A-Z)

→ iappend (Appends an elements to a list variable)

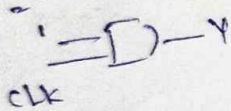
→ foreach → iterates over list items.

fabrication steps

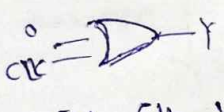
- silicon wafer preparation
- ion implantation
- diffusion
- photolithography
- oxidation
- Chemical - Vapour deposition
- metallization
- package.

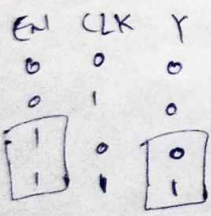
→ lithography
→ etching
→ deposition
→ chemical mechanical polishing (CMP)
→ ion

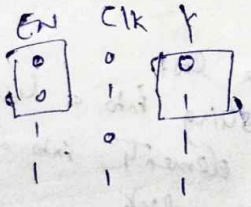
Day-4



Introduction to delay tables





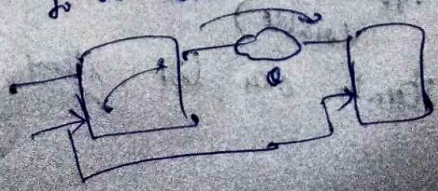


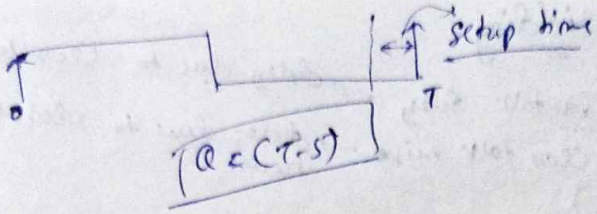
⇒ delay tables is the representation of delay of all the gates.

Delay depends on
- input slew
- output load

slew output of one gate is equal to the slew input of next gate.

Setup time
↳ The time interval before the active clock edge where data need to be stable.





Uncertainty

UC = jitter + skew + margin

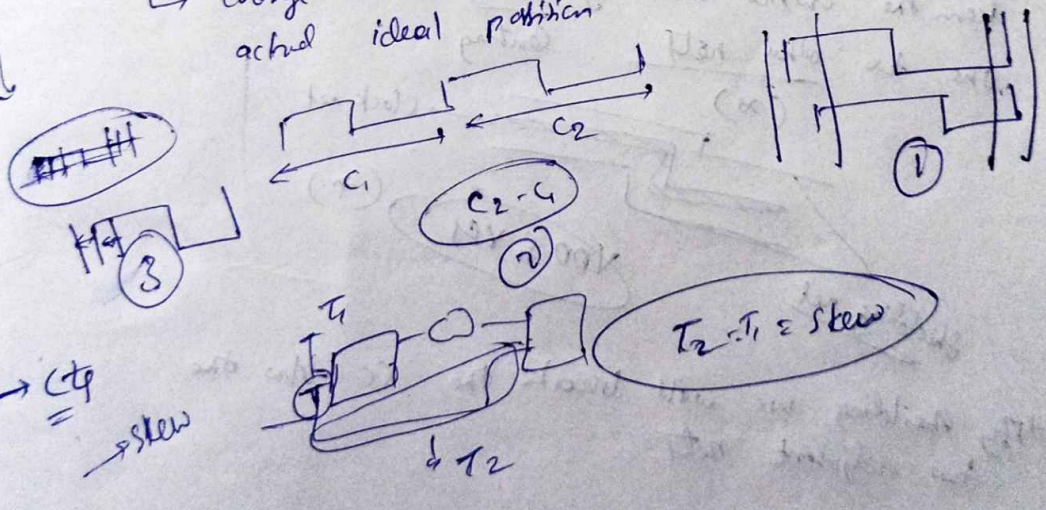
jitter → Undesired deviation occured in the clocks signal

Period jitter | Cycle-Cycle jitter | Absolute jitter

(1) Absolute jitter
↳ change in the clock position from its ideal position

(2) Cycle-to-Cycle jitter
↳ difference of two clock cycles

(3) Period jitter (used in STA)
↳ change the transition of the clock from its actual ideal position



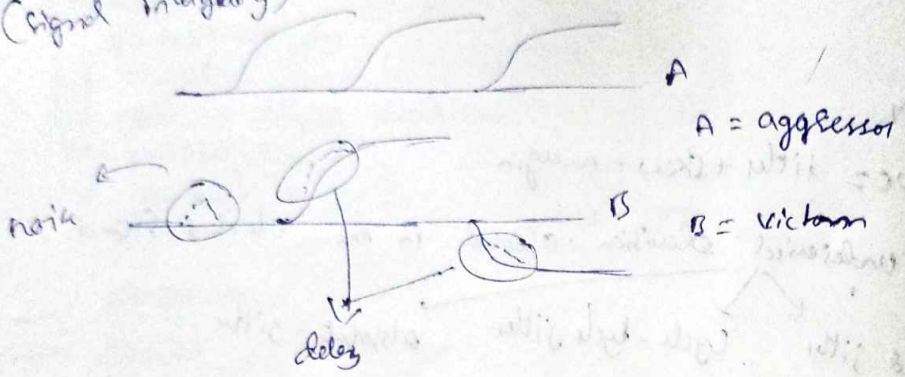
→ Clk = → Skew

→ H-Tree
↳

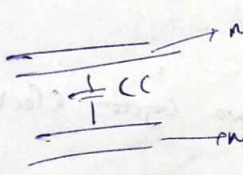
Clock net shielding.

Cross talk → Cross talk delay → delay due to crosstalk
Cross talk noise → delay due to crosstalk noise

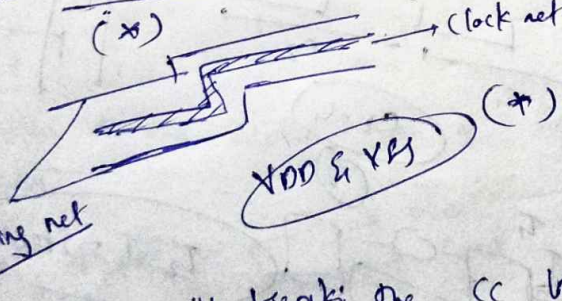
(Signal integrity)



Cross talk occured due to the Coupling Capacitance b/w the two adjacent nets.



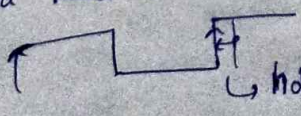
→ Clock net shielding means protecting the clock net from the outside sides by placing the clock net in b/w the two other nets Routing



→ By shielding we will break the CC b/w the two adjacent nets

Hold time

↳ The time interval after the active clock edge wele data need to be stable.



Day-5 27

25-11-14-0.3

mvm gate

cns -3854.15
wns -3062

27-11-09-0.2

new strategy haven't set

tns => -266.43
wns => -2.95

hell set

Routing

- Global route (imigenry)
- track assignment
- detail route
- search & repair.

is the process of creating metal wire to all the signal & clock pin.