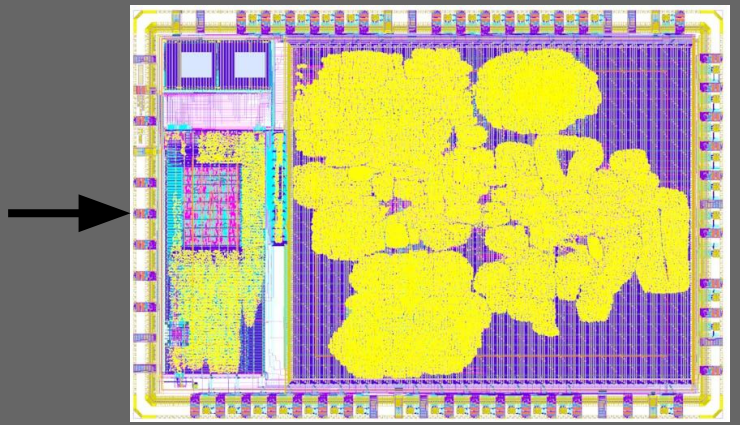
CPE 470 - Openlane & STA



EDA Design Flows

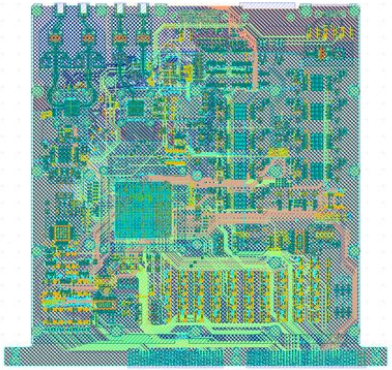
- Goal: Turn our design into a chip
  - RTL to **GDSII**
  - Requires 50+ steps
  - Too complex to do manually
- How to Manage? -> **Design Flows**
  - Use an EDA tool design flow to manage and abstract steps
- Failures can occur at any step of the process
  - Our job is to find and fix failures until we hit **Design Closure**

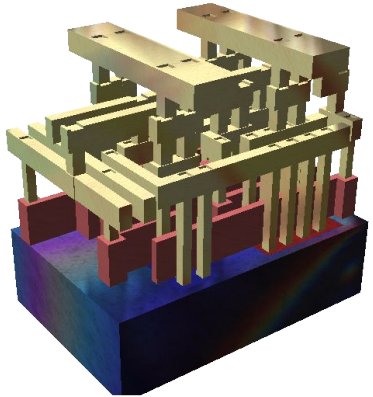
Glossary

GDSII: “Graphic Design System” 2, is a database that contains the final manufacturable design of a chip

Design Flow: combination of tools in series of steps to produce

Design Closure: meeting all performance, timing, power, functionality, and validation requirements



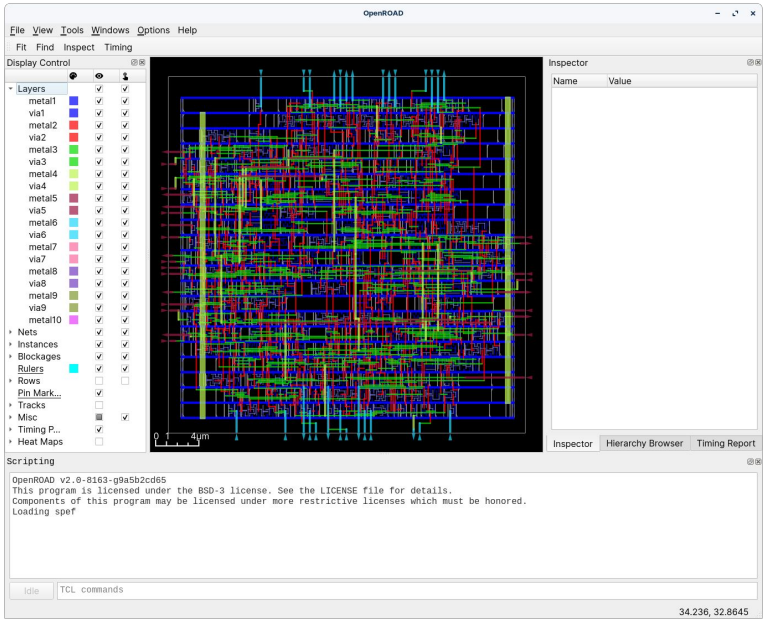


OpenROAD

OpenROAD

Realization of Open, Accessible Design

- Goal: automated RTL to GDSII
  - 24 Hours
  - No Human In Loop (Automated)
- Headed by:
  - UCSD
  - Arm
  - Qualcomm
- Funded by DARPA
  - Defense Advanced Research Projects Agency
  - DoD Research



- Both a project and a tool
  - Tool -> GUI application for viewing and even editing designs
    - Integrated design flow
  - Project -> overarching organizing with their own IP, designs, tools

Openlane

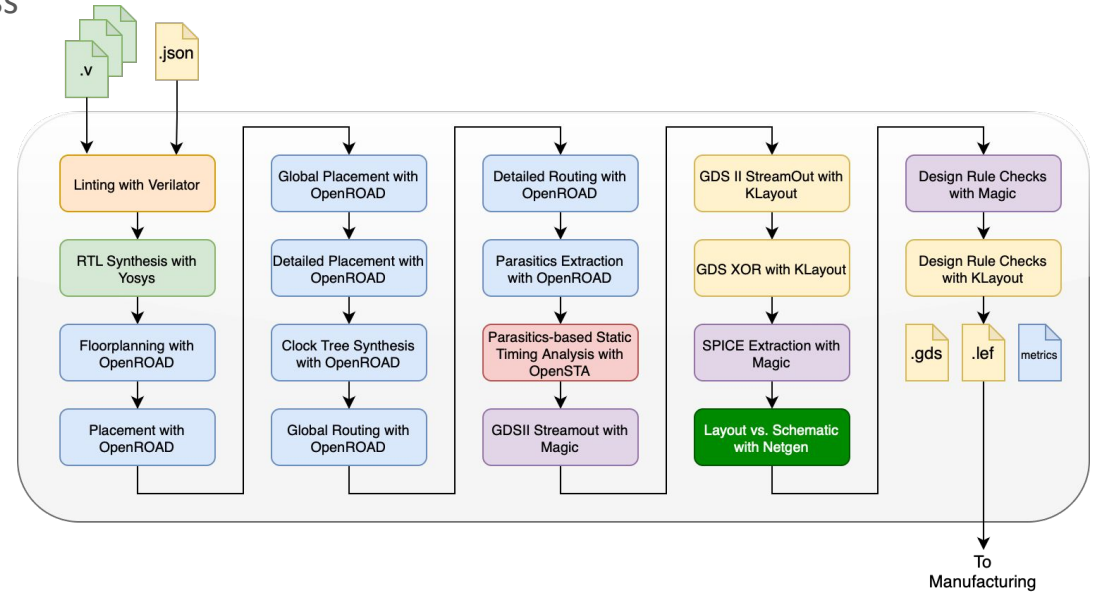
- The most mature open source EDA design flow
  - Tested on silicon across many different chips and PDKs
- Wrapper around OpenROAD and several other tools
  - Verilator, Yosys
  - Magic, KLayout, OpenSTA
- TCL-Based
  - Set up ASIC parameters using tcl script or JSON file
- Primarily developed by eFabless

Glossary

TCL: Tool Command Language, a scripting language

- In between Bash and Python

efabless.com

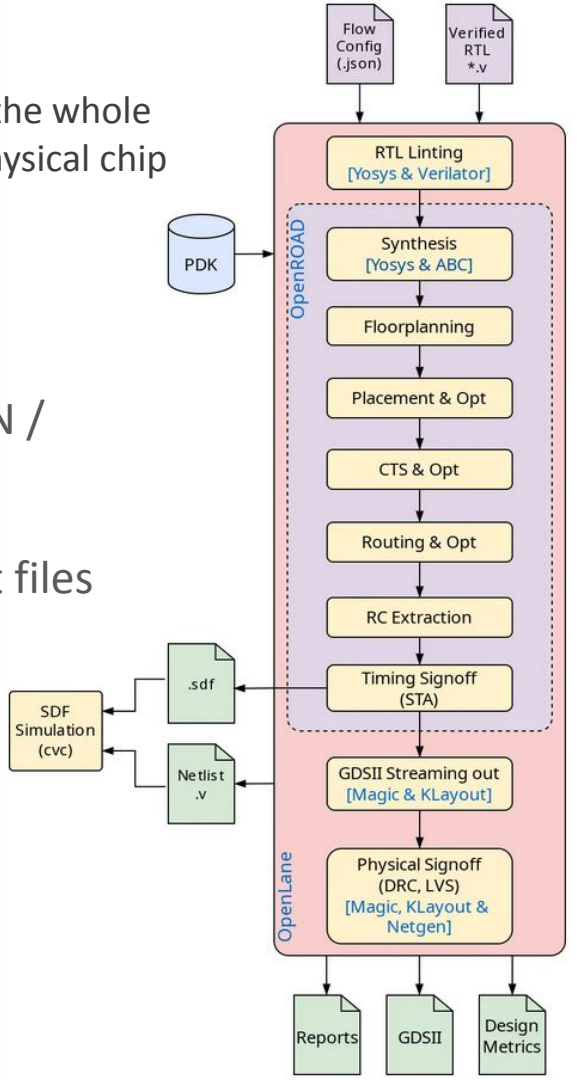


OpenLane2

Glossary

VLSI: very large scale integration, the whole process of going from design to physical chip

- Our main **VLSI** tool flow
- Rewritten version of the original Openlane
  - Moved from TCL to Python
  - Configuration moved from JSON / TCL to JSON / YAML
- More centrally configurable
  - Less spread out configuration across different files
- Operates one central **Flow**
  - Divided into a series of **Steps**
    - Each step gets its own directory
    - Stored results per step



efabless.com

Review: Verilator & Yosys

Steps 1-4

- Verilator is used only as linter
- Make sure HDL is free of linting errors before starting flow
  - Run linter independently while designing and testing

Steps 10-11: To Be Reviewed Later…

Steps 5-9

- Design must be synthesizable
  - Watch out for latches
- Generally the flow will be your first time synthesizing
  - Often synthesis is only done when starting the path to chip
  - Synthesis is more tightly coupled to the rest of the design process

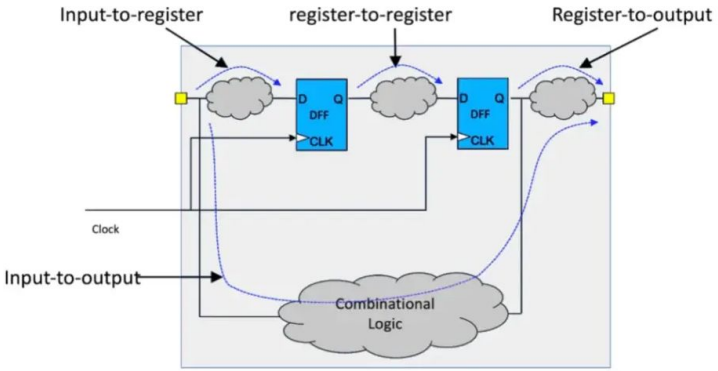
Static Timing Analysis: OpenSTA

Step 12

- Use **STA** to make sure that design meets desired timing constraints
- Often Concerned with Register to Register Paths
  - Calculate logic delay between each register
  - Find the critical path(s) of the design
- See if longest path can be achieved within one clock period
  - Depends on target frequency of the design

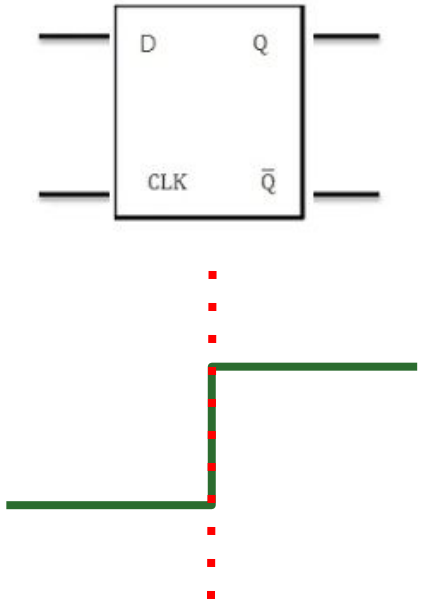
Glossary

STA: static timing analysis



Ideal Register / Flip Flop

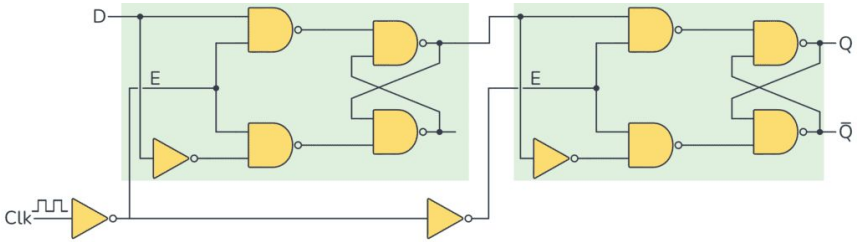
- Samples Data perfectly on clock edge
  - Instantaneous Capture on Perfect Edge
- Immune to noise before and after edge
- Always holds a known binary value
  - 0 or 1
- Data is instantly made available to output



Real Register Part 1

- Takes some time to sample data

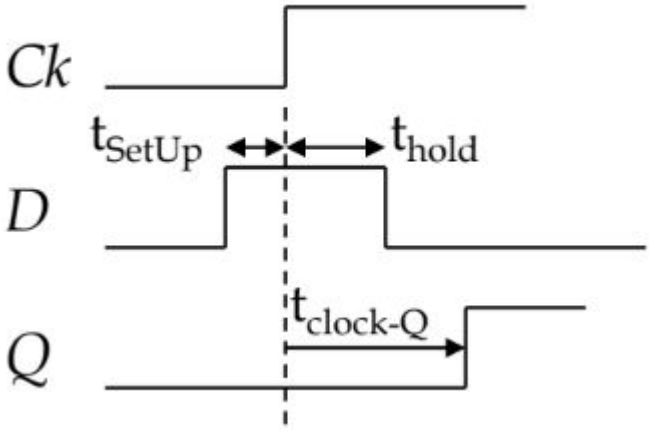
- Requires data to be stable during that time
  - **Setup Time**: min time before the edge that data must remain stable
  - **Hold Time**: min time after the edge that data must remain stable



Glossary

Setup Time t_S: data must be stable t_S before edge

Hold Time t_H: data must be stable t_H after edge



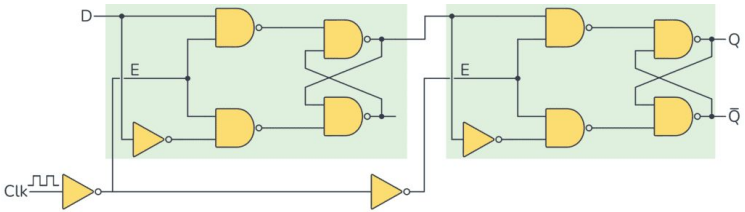
Real Register Part 2

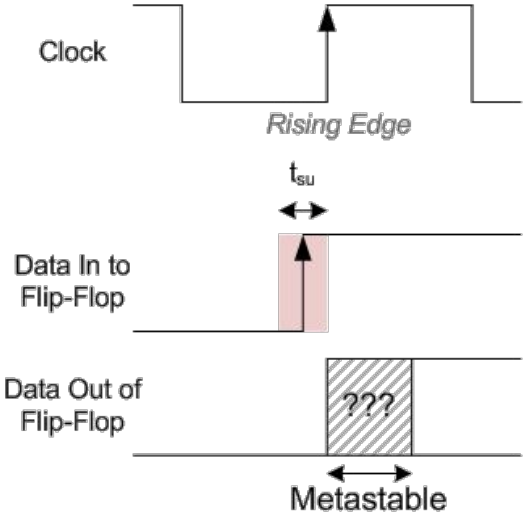
Glossary

Metastability: register gets stuck between 0 and 1 states for unknown amount of time

- If signal is not stable, risk **metastability**
  - Register will risk incorrect data, or even worse get stuck

- Once sampled, takes some time to propagate to the output
  - t_{CQ}: clock-to-Q propagation delay

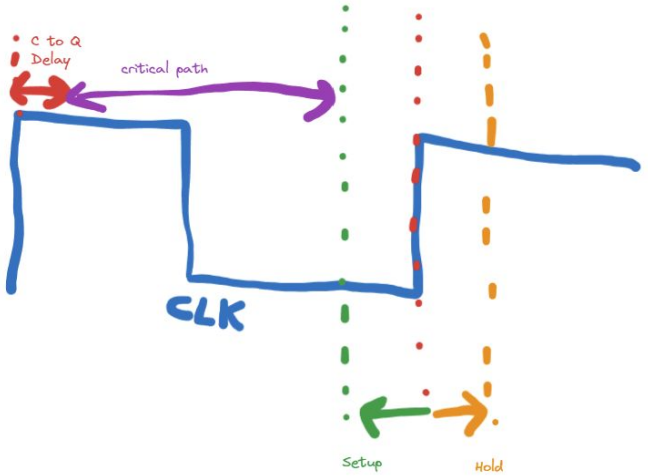


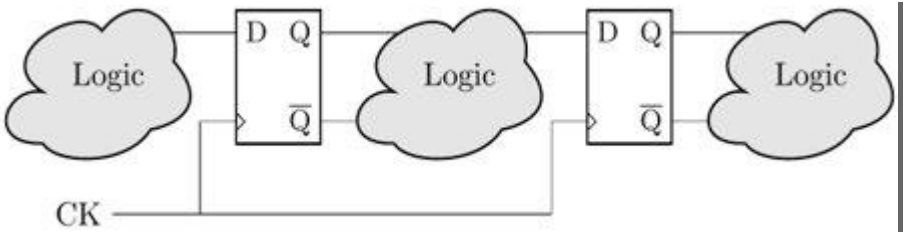


Max Path: Setup Time Violations

- Max Path asks the question:
  - Is your critical path logic too slow to keep up with your clock?
- **Setup Time Violation**: logic is too slow such that data changes past setup time period
  - Fixable by lowering clock frequency
  - Or use faster logic
- Usable Portion of Clock Period:

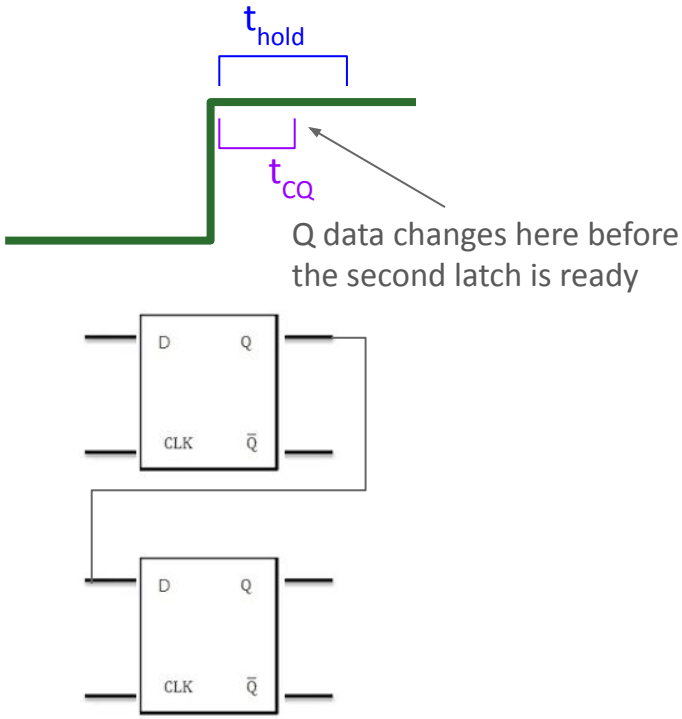
$$t_{logic} \leq t_{clock} - t_{setup} - t_{cq}$$



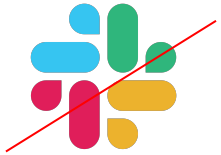


Min Path: Hold Time Violations

- Min Path asks the question:
  - Is your minimum logic path so fast that one register's output will change before the next one has completed hold time?
- **Hold Time Violation:** input to register will change before register is done with its hold time
- Not affected by Clock Speed, only by design
  - **NOT FIXABLE AFTER TAPEOUT**
  - Have to tell the tools (Openlane) to try harder to fix



Slack



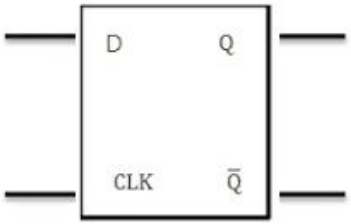
- **Slack** is the amount of leeway between your design's timing and a design constraint
- **Max Path**:
  - **Positive Slack** means you are passing your timing requirement.
    - With a positive slack of 3, you could speed up your clock frequency by up to 3
  - **Zero Slack** means you are passing but with no wiggle room
  - **Negative Slack** means you are violating setup time.
    - To fix, slow down your clock frequency by the amount of negative slack
- **Min Path** is not affected by clock frequency
  - **Negative Slack** in your min path is a hold time violation, which the tools will need to fix by inserting dummy logic
- Computing: Slack = Required Time - Arrival Time
  - S = R - A

Arrival Time

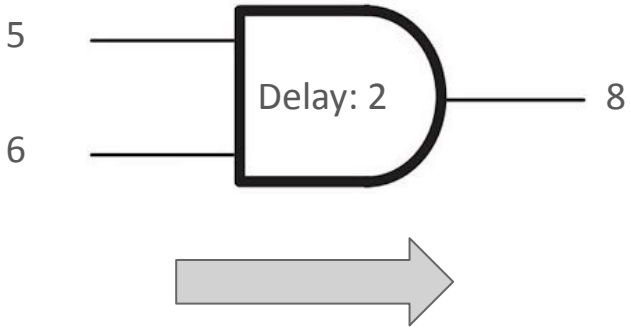
S = R - **A**

- Starting at the front, when does each signal arrive?
  - The “front” is the output of a register
  - Output of a register arrives at $t=t_{cq}$, often ~0

- Each gate takes its worst case (largest) input and adds its delay
  - Out Arrival = delay + max(Inputs Arrival)



t_{cq} (often 0)

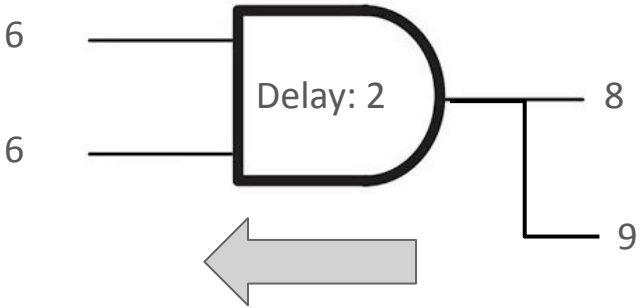


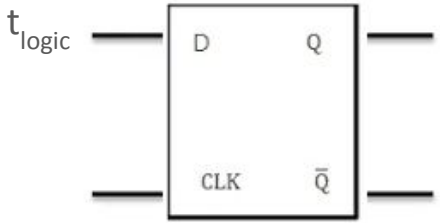
Required Time

S = **R** - A

- Starting at the back, when does each signal have to arrive?
  - The “back” is the input to a register
  - Input to a register required at the max logic propagation time
    - $t_{logic} \leq t_{clock} - t_{setup} - t_{cq}$

- Each gate takes its worst case (smallest) output and subtracts its delay
  - In Required = min(Output Required) - delay



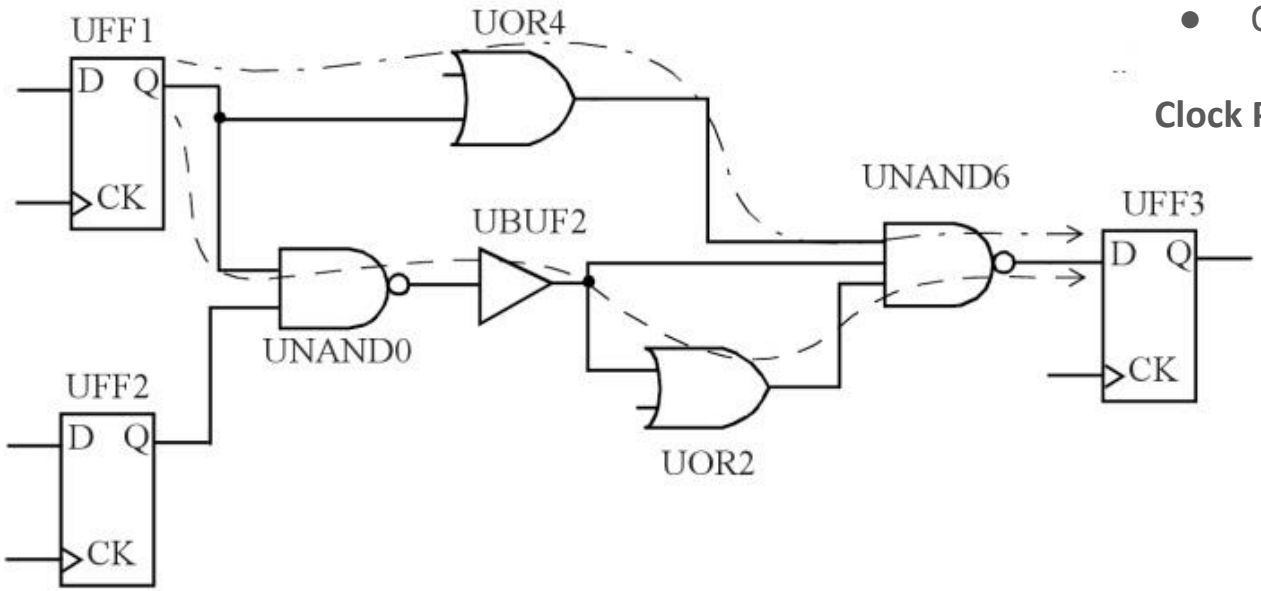


STA Example

Delays:

- Buffer: 1
- NAND: 1
- 3-Input NAND: 2
- OR: 2

Clock Period: 5



https://www.vlsi-expert.com/2011/03/static-timing-analysis-sta-basic-timing.html

IPVT Timing Corners

Glossary

IPVT: interconnect, process, voltage, temperature

- Gates do not have universally consistent delay
- Perform differently in different conditions:
  - **Interconnect**: Variations in wire resistance & capacitance
  - **Process:** Variations in silicon across a wafer
  - **Voltage**: Faster when higher voltage
  - **Temperature**: Faster when cold

Interconnect Corners:
Max, Min, Nominal

PVT Corners:

tt_025C_1v80
nmospmos_temperature_voltage

- Faster Corners
  - Less likely to fail setup time / max path
  - More likely to fail hold time / min path
- Slower Corners
  - Less likely to fail hold time / min path
  - More likely to fail setup time / max path
- Must run STA and pass all or most corners!

Process Variation	Temperature	Voltage
ss (slow)	100 C	1.6 V
tt (typical)	40 C	1.8 V
ff (fast)	25 C	1.95 V

References

- https://openlane2.readthedocs.io
- https://openlane.readthedocs.io/en/latest/
- https://openroad.readthedocs.io/en/latest/