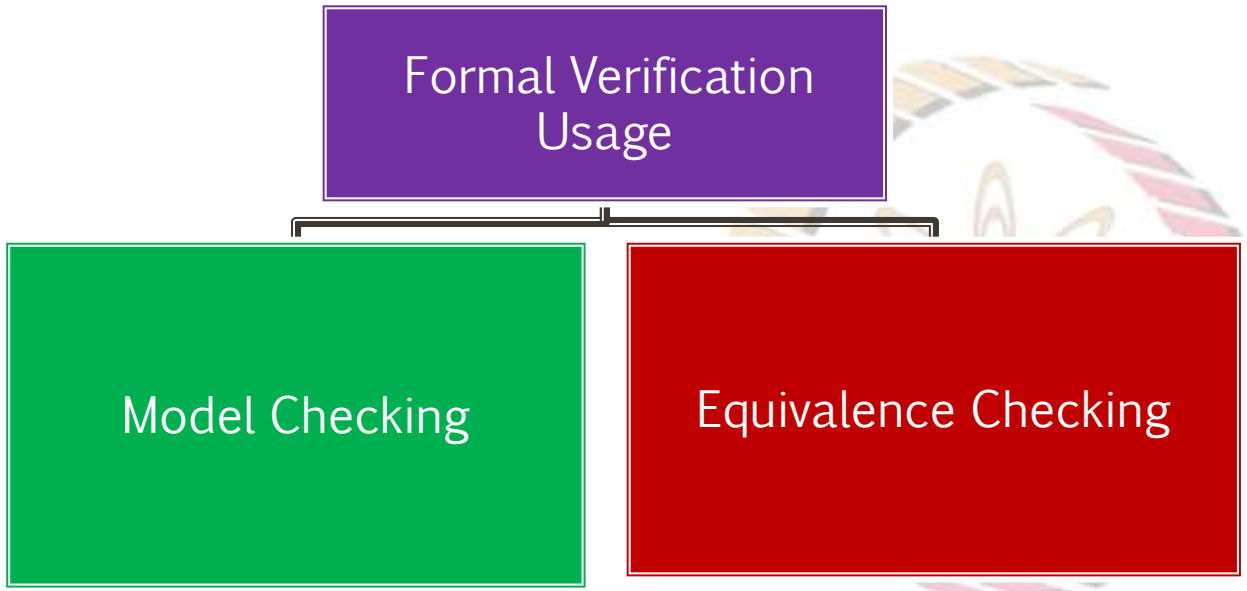
VLSI DESIGN FLOW: RTL TO GDS

Lecture 20
Formal Verification- IV

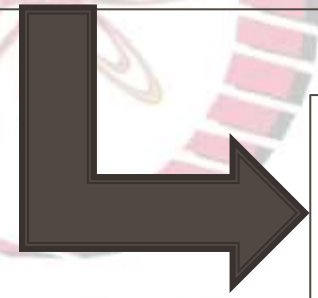
Sneh Saurabh
Electronics and Communications Engineering
IIIT Delhi

Lecture Plan



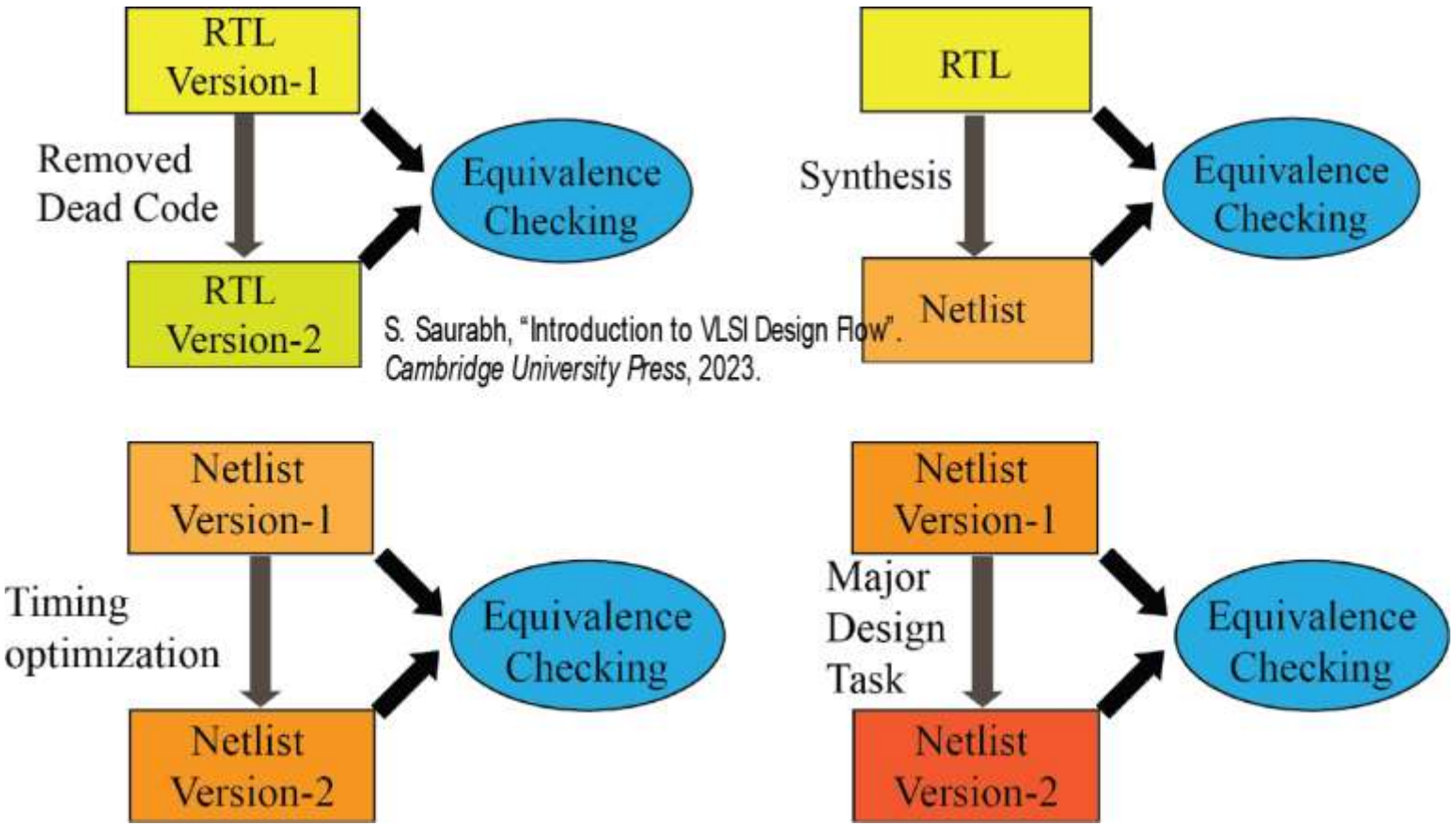
- Equivalence Checking

Formal Verification



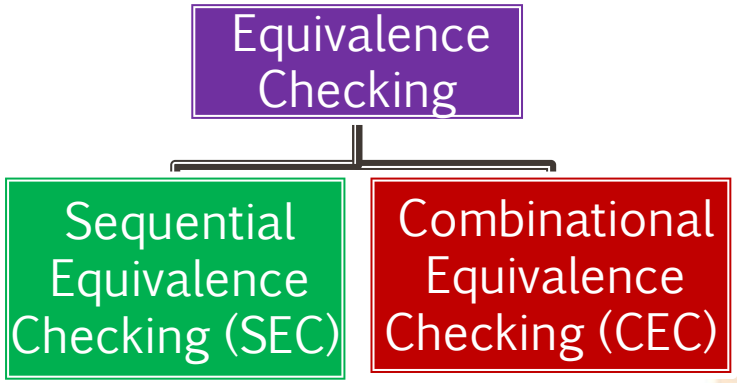
Equivalence Checking

Equivalence Checking: Usage



S. Saurabh, "Introduction to VLSI Design Flow", *Cambridge University Press*, 2023.

Formal Verification : Equivalence Checking (2)



Sequential Equivalence Checking

- Generalized approach to comparing two models
- First convert models to FSM
- Given their initial states, it is checked whether the two FSMs produce matching output sequences for all input sequences

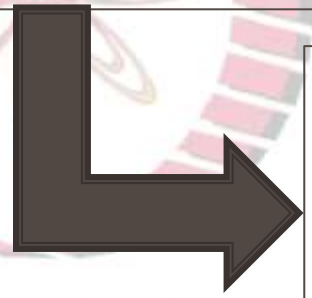
Challenges of SEC:

- Need to explore all reachable states
  - ➢ Encounters state explosion problem
  - ➢ Computationally difficult

Combinational Equivalence Checking

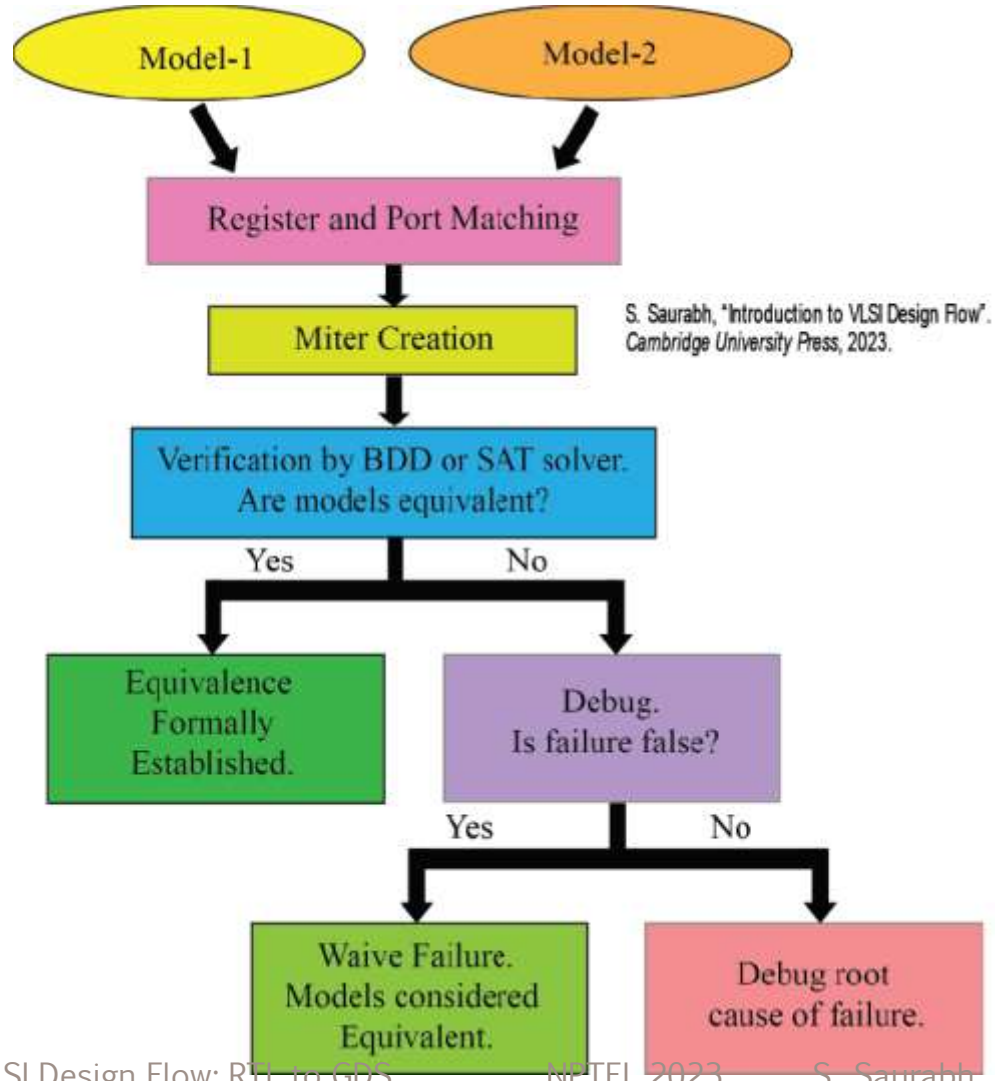
- Assume that there is a one-to-one correspondence among memory elements or flip-flops and the ports of the two models.
- Problem reduced to establishing the equivalence of pairs of combinational circuits
- Assumption holds in most cases
  - ➢ Integral part of design flows

Formal Verification



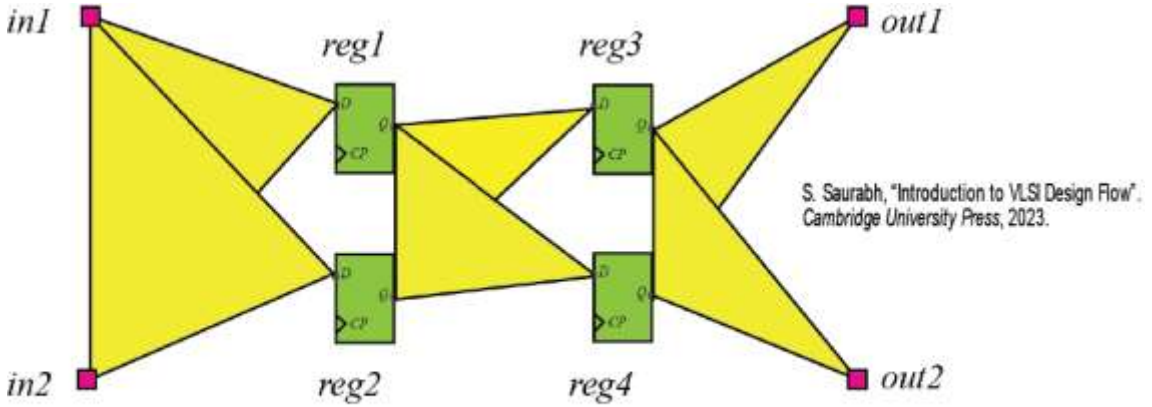
Combinational Equivalence Checking (CEC)

Combinational Equivalence Checking (CEC): Steps



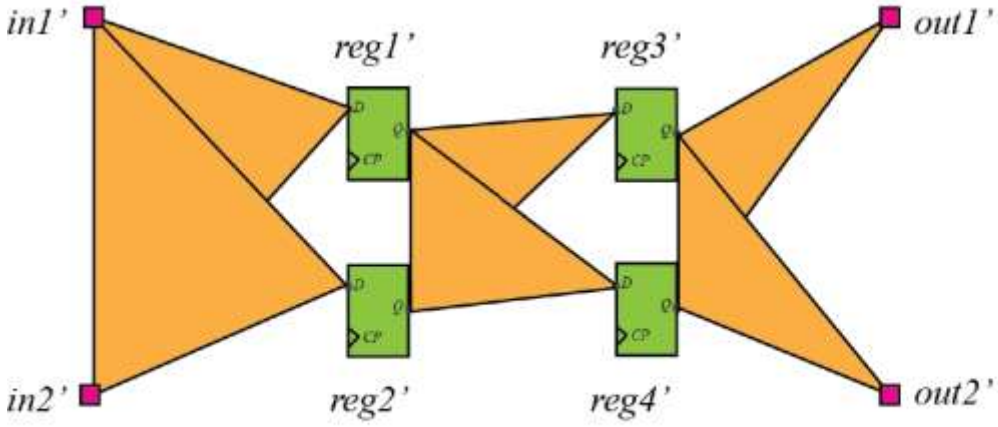
S. Saurabh, "Introduction to VLSI Design Flow". *Cambridge University Press*, 2023.

CEC: Register and Port Matching (Illustration)



S. Saurabh, "Introduction to VLSI Design Flow", Cambridge University Press, 2023.

Model-1



Model-2

CEC: Register and Port Matching (Techniques)

No exact algorithm : heuristics employed

Name-based matching:

- Works very well for I/O Ports
- Works good where a model has not undergone sequential optimization and register correspondence is kept intact
- RTL synthesis keeps names of registers based on some rules/conventions (such as add suffix to the name of signal in the RTL code)

- Sophisticated structural or functional analysis can also be used
- User can guide the tools to match the registers

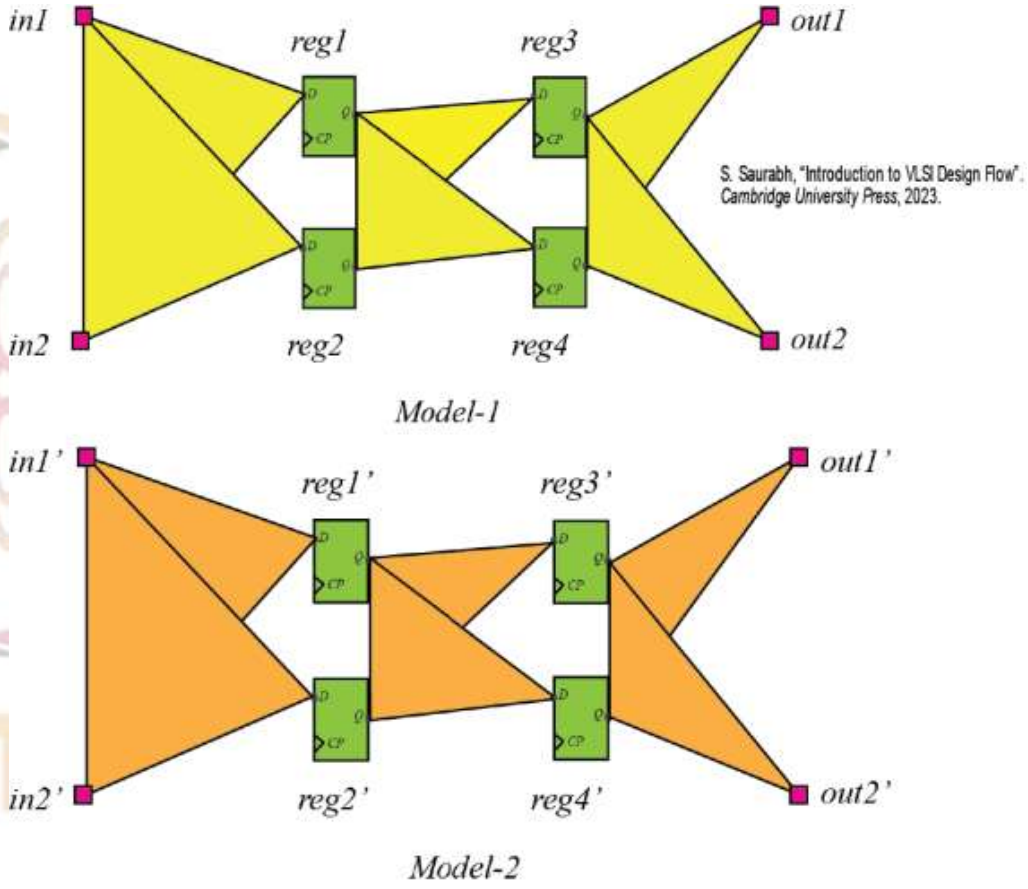
```
module flip_flop_d(d, clk, q);
      input d, clk;
      output q;
      reg   q;

      always @(posedge clk) q <= d;
endmodule
```

S. Saurabh, "Introduction to VLSI Design Flow". *Cambridge University Press*, 2023.

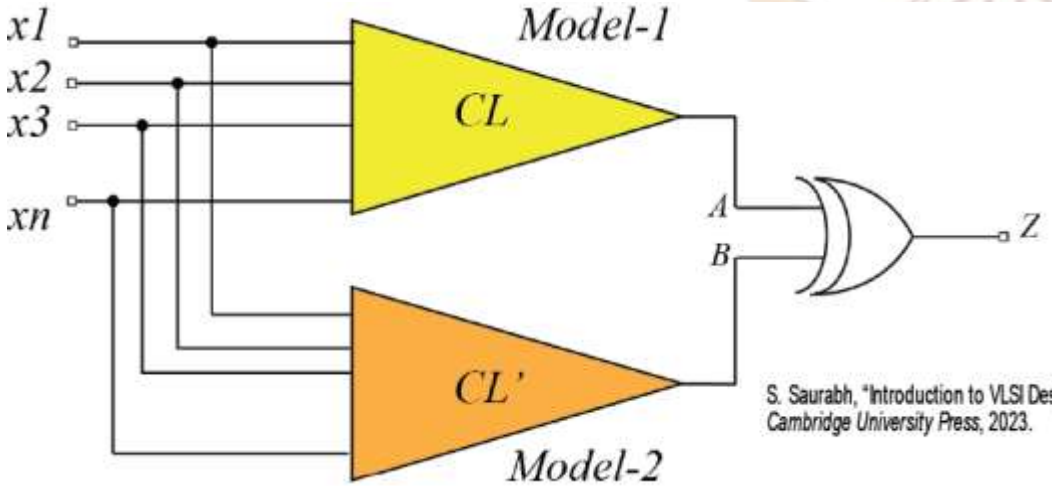
CEC: Miter Creation

- Need to ensure that the corresponding combinational circuits lying between matched registers/ports are equivalent
- Derive multiple small combinational circuits (known as miters) from the given two models
  - A separate miter is derived for each register and output port of the two models



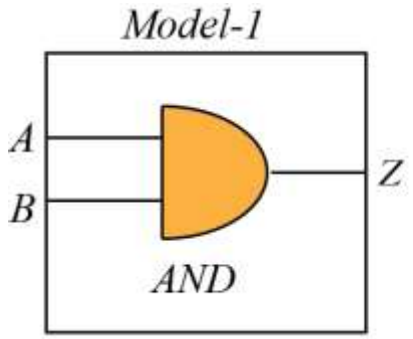
S. Saurabh, "Introduction to VLSI Design Flow", Cambridge University Press, 2023.

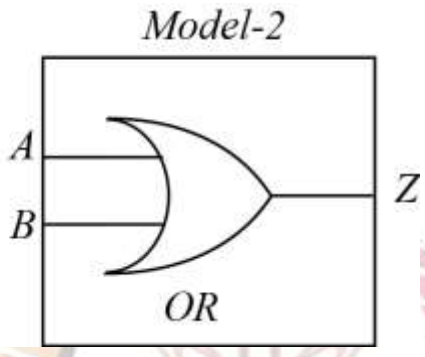
CEC: Miter and Equivalence

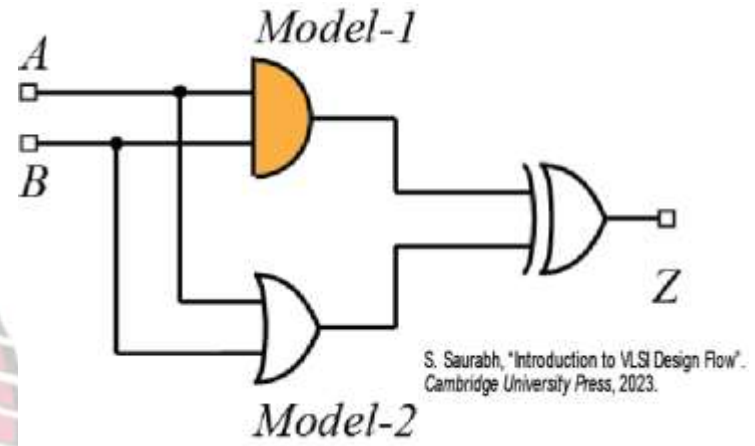


- Z=1 for any combination of $\{x1, x2, x3, \ldots, xn\}$ ⇒ Model-1 and Model-2 are not equivalent
- Z=0 for all possible combinations of $\{x1, x2, x3, \ldots, xn\}$ ⇒ Model-1 and Model-2 are equivalent

CEC: Illustration







S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.

Given two models with ports matched by names.

- Let us draw the miter circuit for checking equivalence of Model-1 and Model-2.

- Let us determine patterns of values that can be assigned to ports A and B such that the output of the miter is 1.

- A=0, B=1
- A=1, B=0

CEC: Establishing Equivalence

Using BDDs:

- Build a BDD for a miter,
  - Equivalent models: reduce to a terminal vertex with the value of 0
  - Nonequivalent models: yield some other vertex.

Using SAT Solver:

- Invoke a SAT solver on the Boolean expression representing a miter
  - Equivalent models: UNSAT
  - Nonequivalent models: SAT

CEC:

- Create miter circuits for all flip-flops and output ports in the models
- For equivalent models, all miter circuits should be equivalent

Safety and False Failures:

- CEC can sometimes give false failures
- Still it is always safe

References

- S. Saurabh, “Introduction to VLSI Design Flow”. Cambridge: *Cambridge University Press*, 2023.