

ECE 459/559

Secure & Trustworthy

Computer Hardware Design

VLSI Design Basics

Garrett S. Rose
Spring 2017

Recap

- Brief overview of VHDL
 - Behavioral VHDL
 - Structural VHDL
- Simple examples with VHDL
- Some VHDL for SIMON encryption

Summary

- Circuit-level view
 - Transistors as switches
 - Static CMOS circuits
- Top-down design flows
 - VHDL to silicon

Some General Terms

- VLSI – Very Large Scale Integration
- CMOS – Complimentary Metal-Oxide Semiconductor
- ASIC – Application Specific Integrated Circuit
- FPGA – Field Programmable Gate Array
- SoC – System on Chip
- NoC – Network on Chip

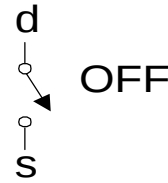
- HDL – Hardware Description Language (VHDL or Verilog)
- RTL – Register Transfer Language

Transistors as Switches

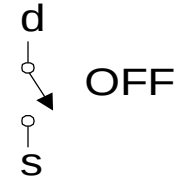
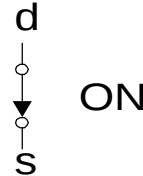
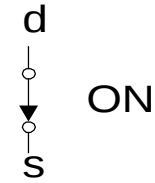
- We can view MOS transistors as electrically controlled switches
- Voltage at gate controls path from source to drain



$g = 0$

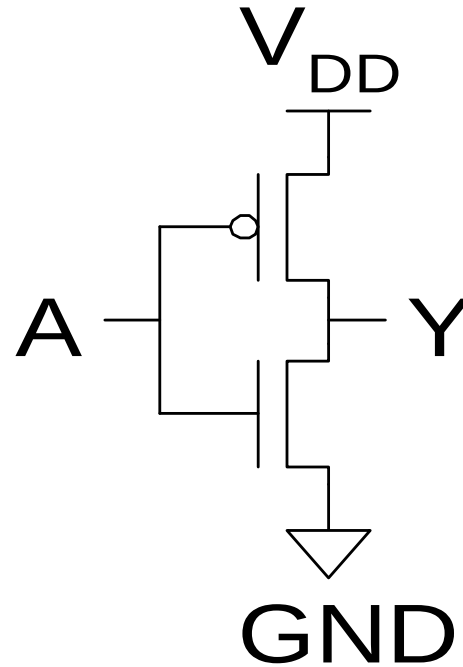
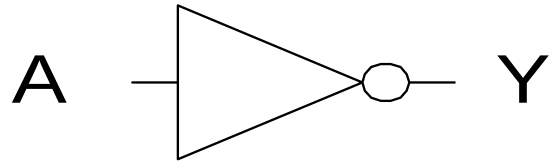


$g = 1$



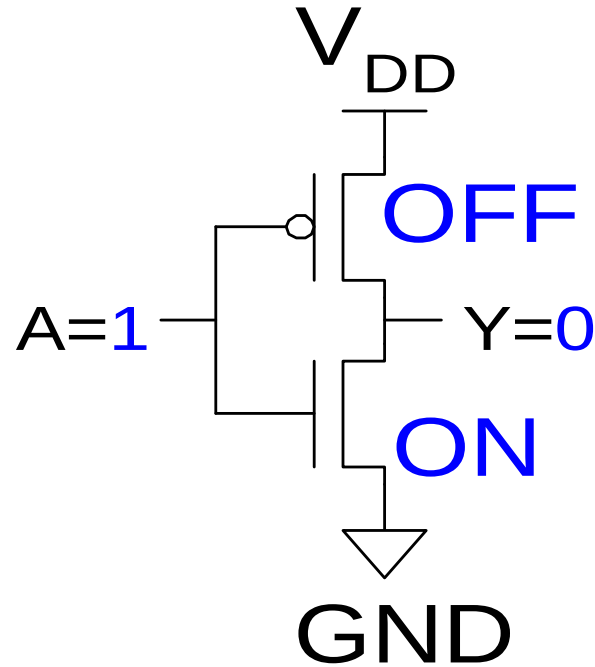
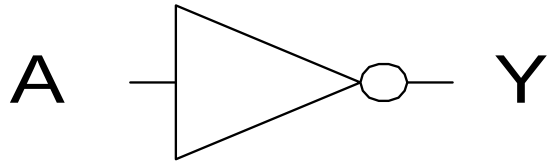
CMOS Inverter

A	Y
0	
1	



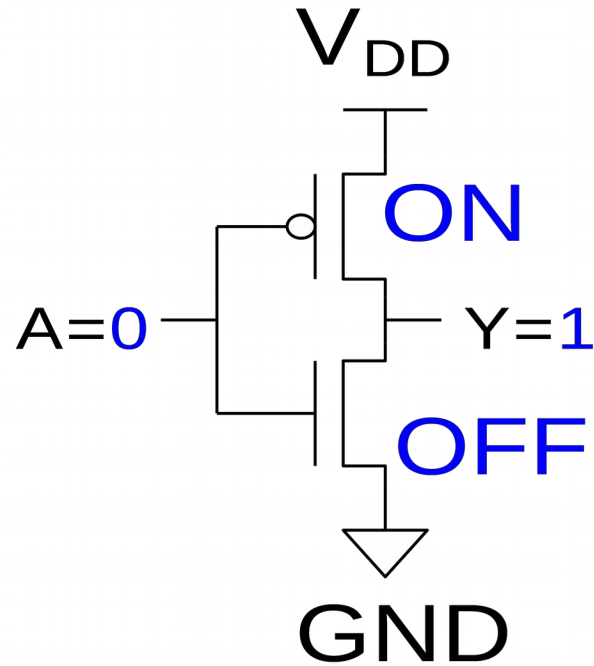
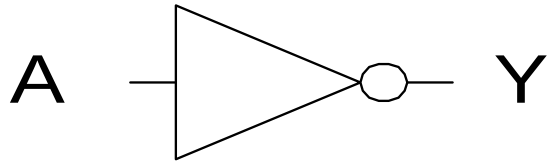
CMOS Inverter

A	Y
0	
1	0



CMOS Inverter

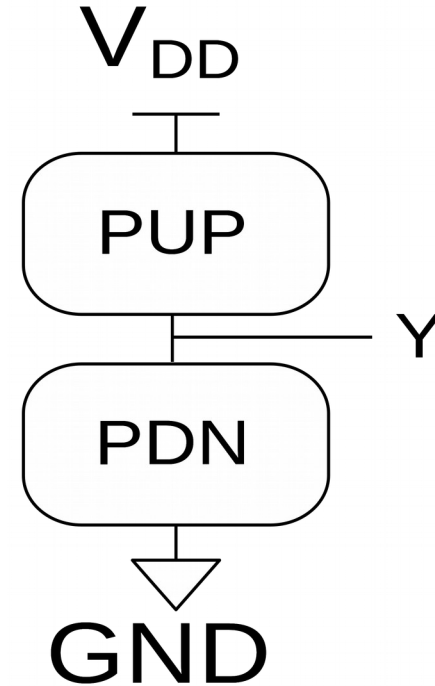
A	Y
0	1
1	0



More Complex Gates:

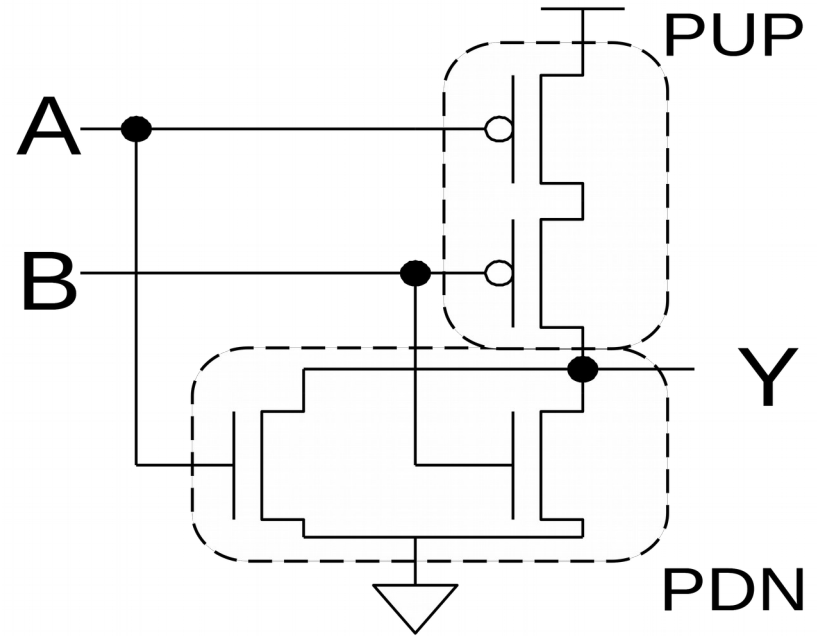
Pull-Up and Pull-Down Nets

- Pull-up net (PUP) off when pull-down (PDN) on
- PUP implemented as complement of PDN (*Complementary MOS*)
- If two FETs in parallel in PDN, counterparts in series in PUP
- Output (Y) connected to V_{DD} or GND, never both



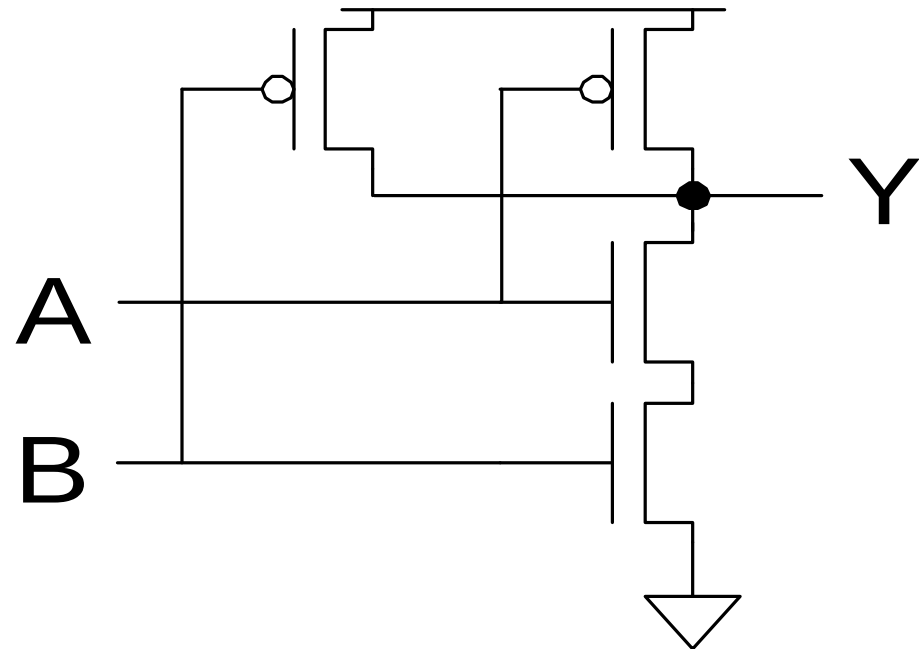
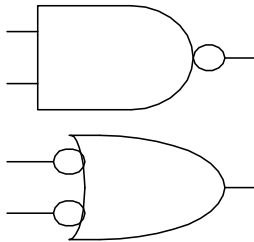
More Complex Gates: Pull-Up and Pull-Down Nets

- Pull-up net (PUP) off when pull-down (PDN) on
- PUP implemented as complement of PDN (*Complementary MOS*)
- *If two FETs in parallel in PDN, counterparts in series in PUP*
- Output (Y) connected to V_{DD} or GND, never both



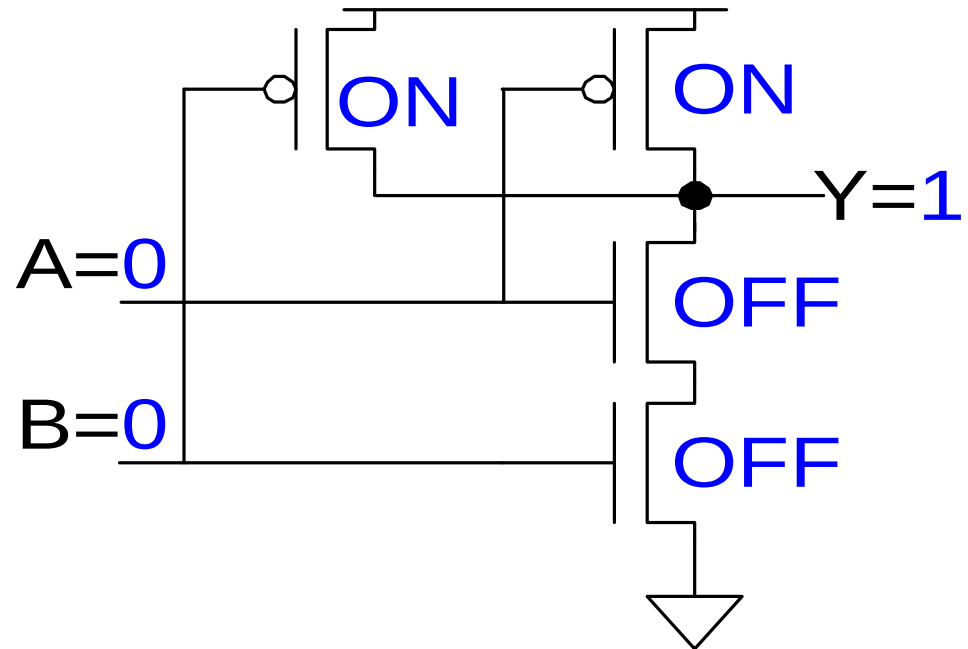
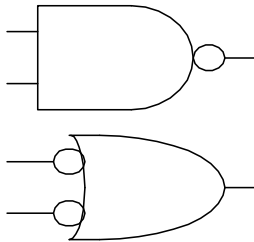
CMOS NAND Gate

A	B	Y
0	0	
0	1	
1	0	
1	1	



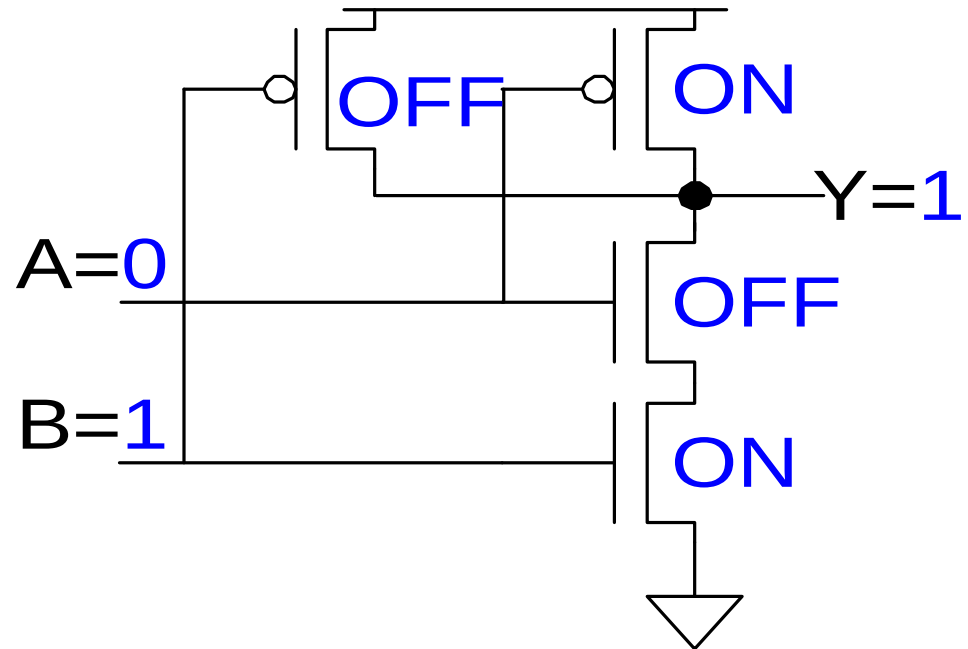
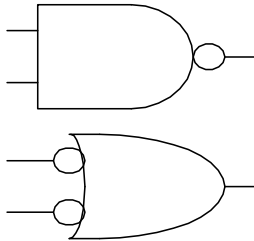
CMOS NAND Gate

A	B	Y
0	0	1
0	1	
1	0	
1	1	



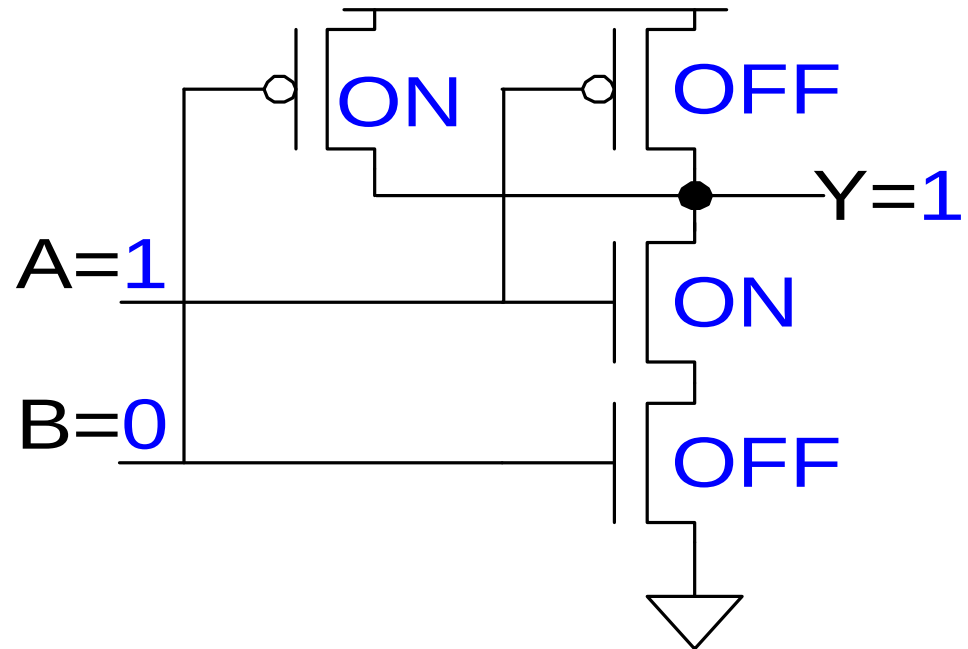
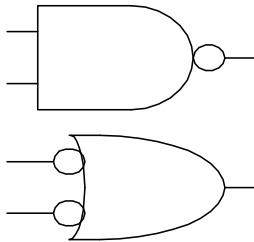
CMOS NAND Gate

A	B	Y
0	0	1
0	1	1
1	0	
1	1	



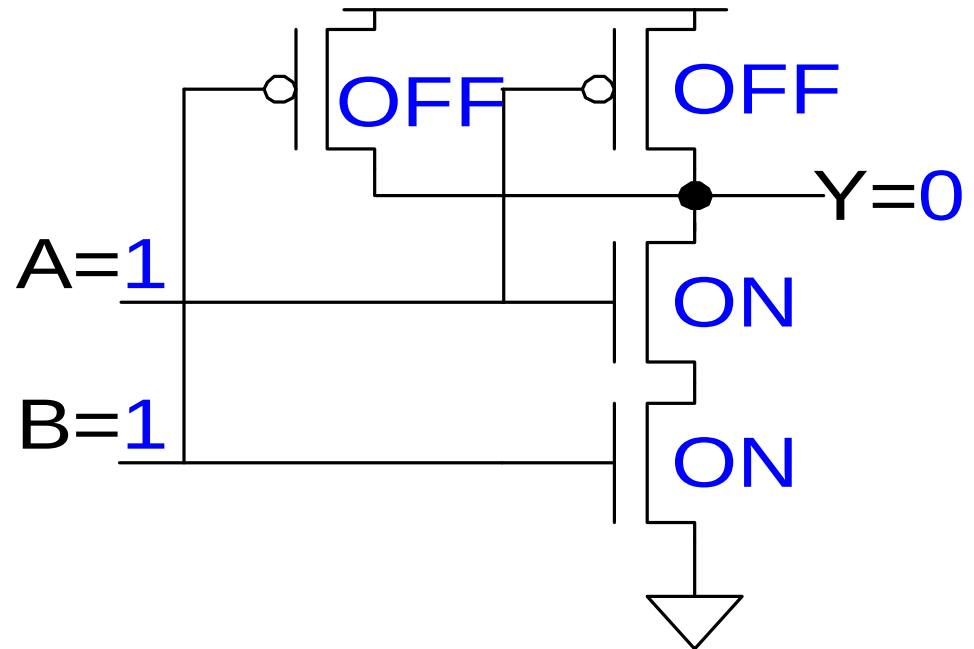
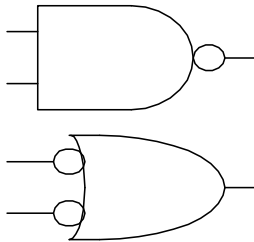
CMOS NAND Gate

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	



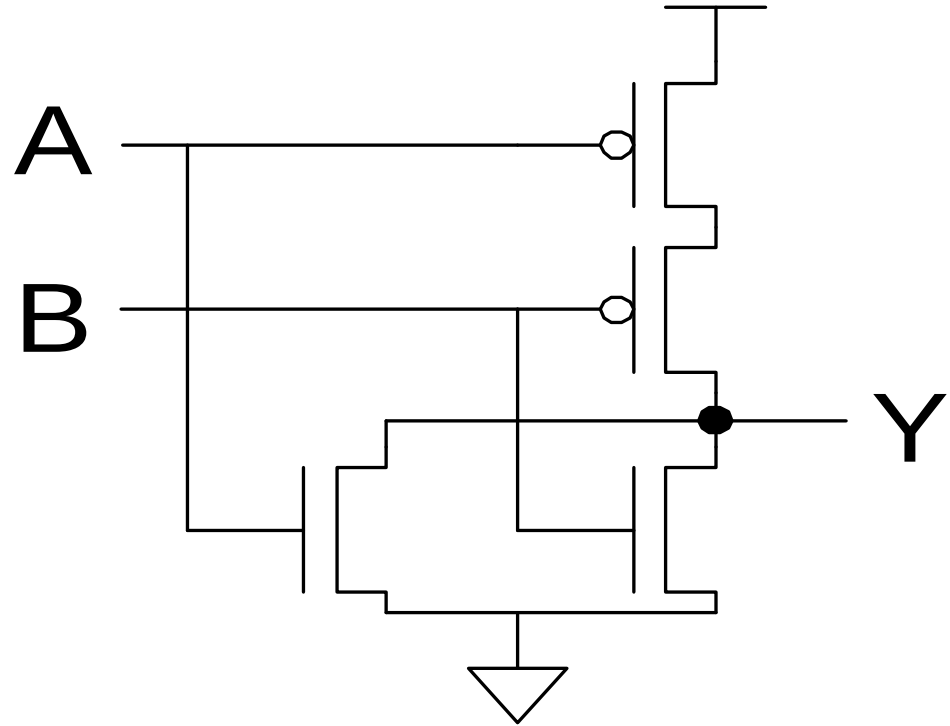
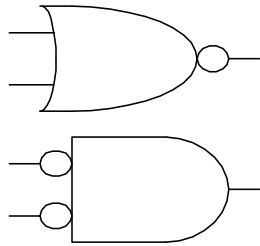
CMOS NAND Gate

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



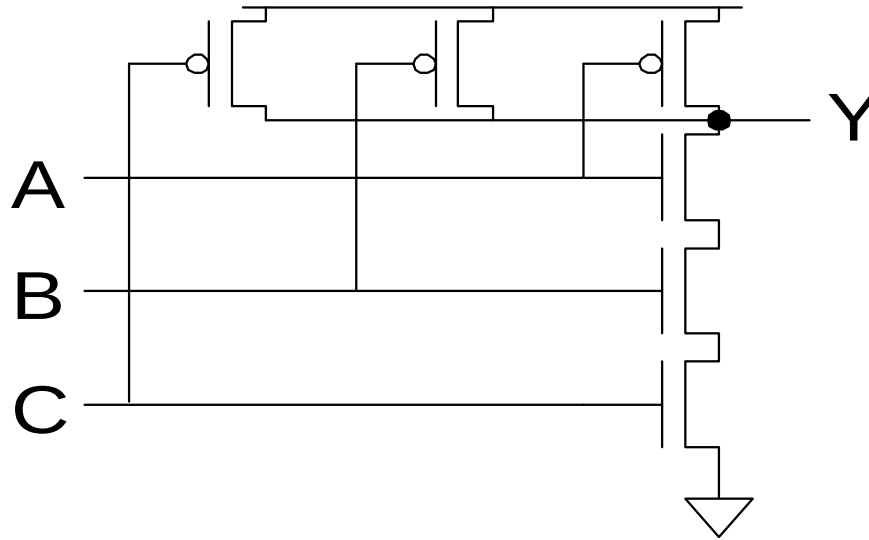
CMOS NOR Gate

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



3-Input NAND Gate

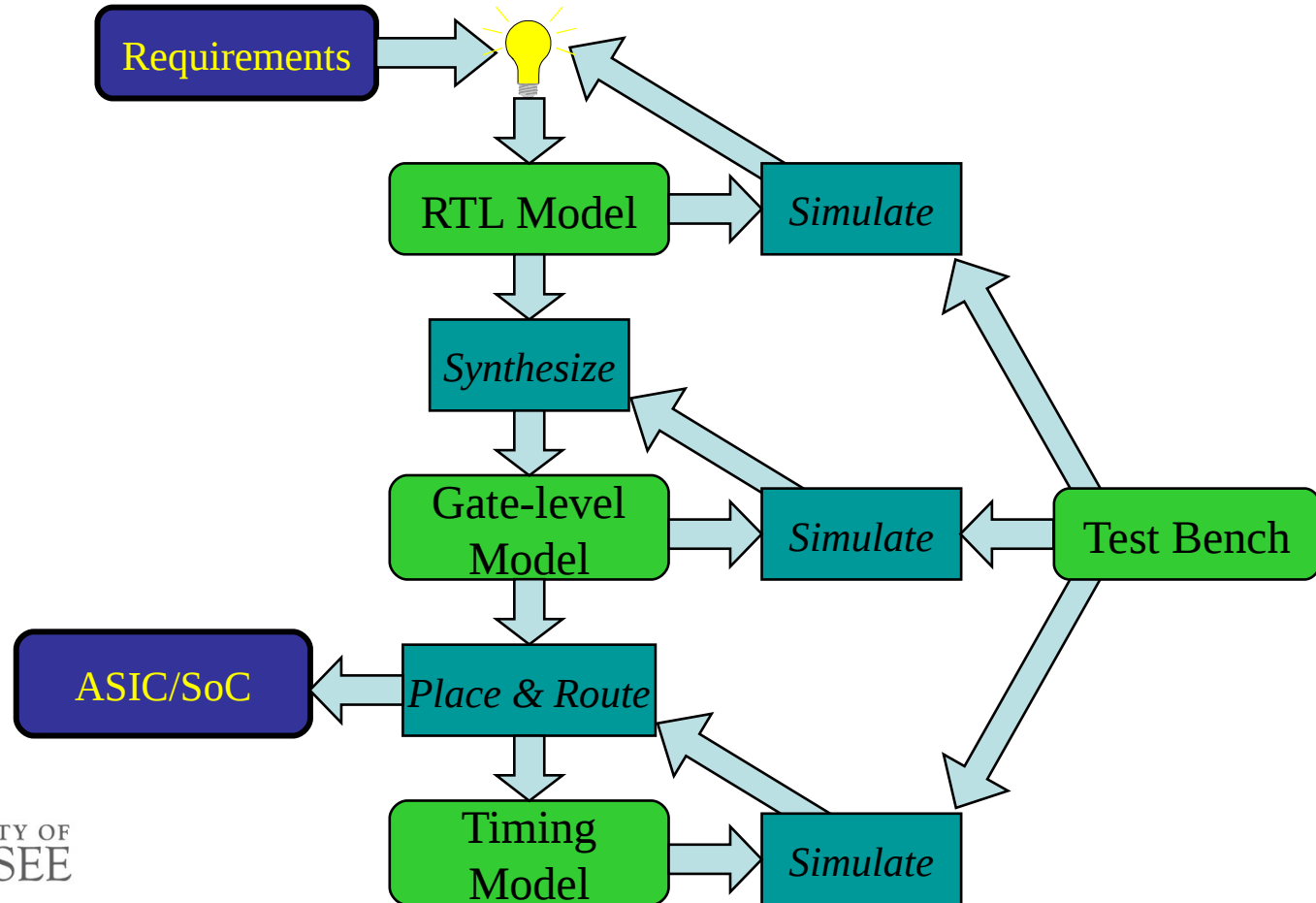
- Y pulls low if ALL inputs are 1
- Y pulls high if ANY input is 0



The ASIC

- The term “ASIC” has been applied to many design styles
- Technically, refers only to application specific circuits (i.e., any microchip you design yourself)
- Often, ASIC is used to refer to automated designs developed using some hardware description language
- Usually want an ASIC fast – clear design flows applied
- ASICs are low volume integrated circuits
- In recent years, ASICs are less common since an FPGA can be used to implement desired function
 - Some might say... “FPGA is the new ASIC”

Typical ASIC Design Flow

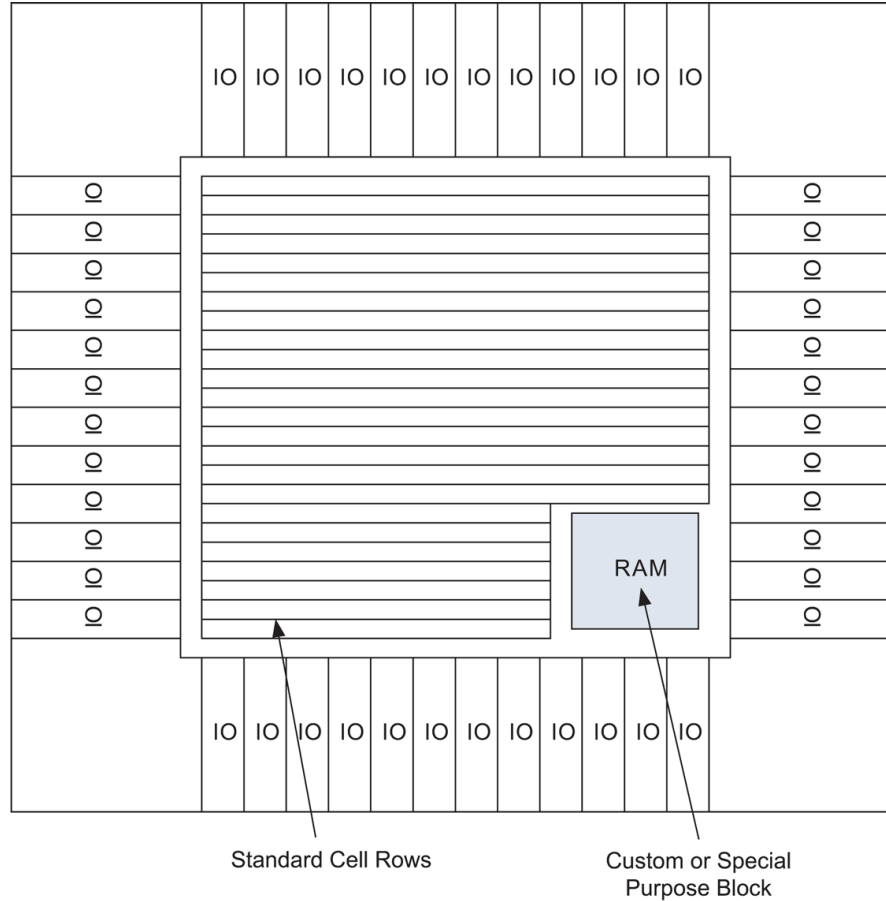


ASIC Design Steps

- RTL design and verification – must write the HDL code
 - Can use VHDL or Verilog
- Synthesis – compiles HDL design description into a *gate level netlist*
- Floorplanning – before place & route, must decide where functional modules will be placed on the die or FPGA
- Place & Route
 - Placement – determines where standard cells are placed
 - Routing – adds wires (configures switch blocks) connecting gates to implement final design
- ***Every step must include simulation & verification***

Placement of Cells

ASIC Standard Cell View



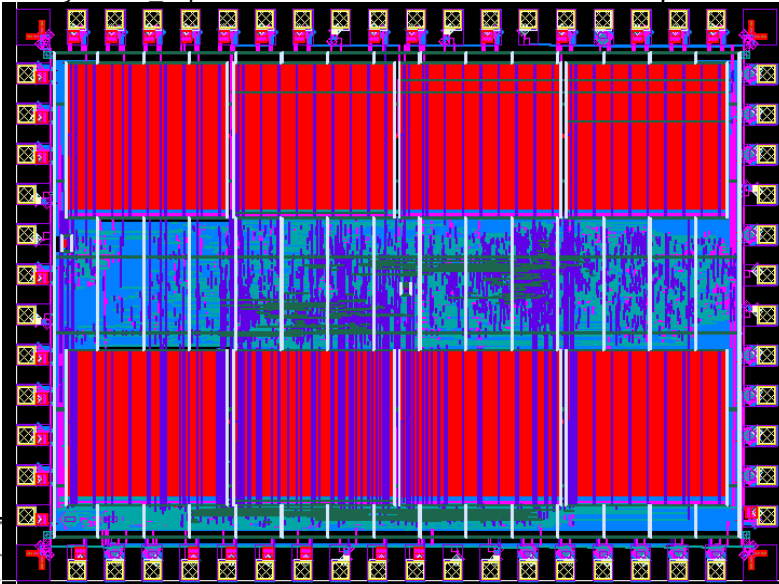
System on Chip (SoC)

- SoC implies a system of fairly high level blocks (e.g., memory, processors, DSP, etc.) integrated into one design
- SoC often refers to heterogeneous systems encompassing a great deal of functionality, often mixed signal
- Complex blocks are designed individually and not modified at the highest level – each block essentially a “black box”
- *Designers often use intellectual property (IP) cores for the building blocks of higher level designs*
- Repository of useful, yet free IP: www.OpenCores.org

System on Chip Design

```
-- Revision:  
-- Revision 0.01 - File Created  
-- Additional Comments:  
--  
-----  
--  
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;  
use IEEE.STD_LOGIC_ARITH.ALL;  
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

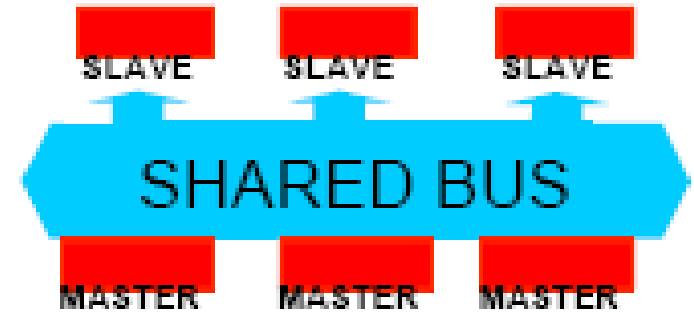
entity SRAD Top is



- Start with high level HDL description
- Some blocks synthesized from HDL, some custom
- Research opportunities in power/temp. management, interconnection issues, etc.
- Example: an ultrasound image processing system

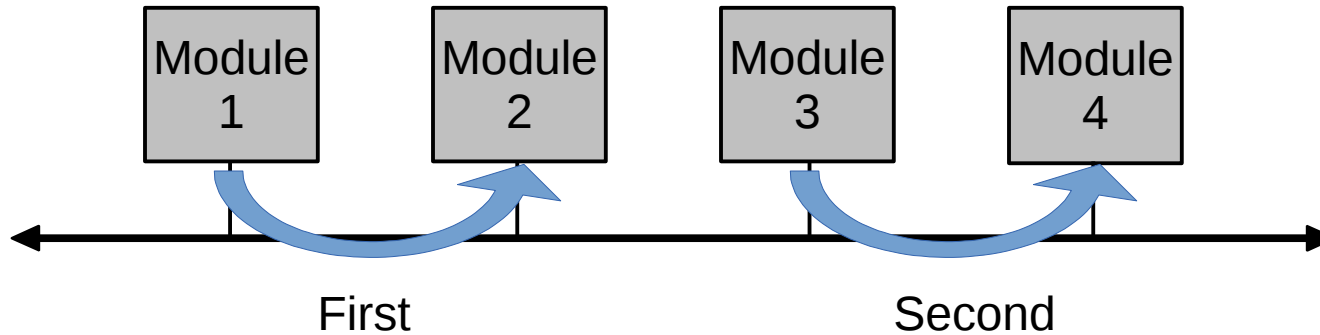
SoC Communication

- On-chip communication major design consideration for IP blocks
- Shared Bus (broadcast)
 - Low area
 - Poor scalability
 - High energy consumption
- Network on Chip (point-to-point)
 - Scalability
 - Low energy consumption
 - High area

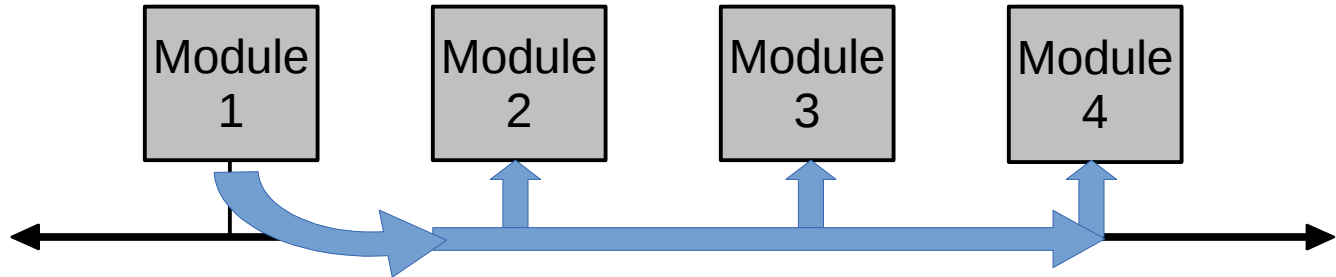


Bus Basics

- Bus communications follows strict order – serial nature

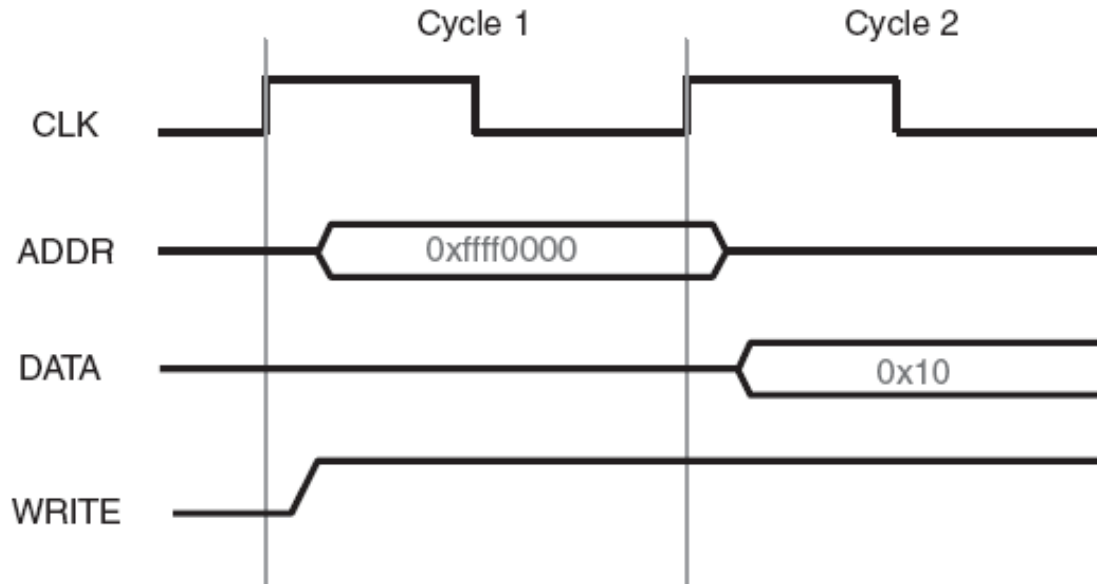


- Can broadcast – multiple destinations at the same time



Bus Basics

- Bus communication operates in units of cycles, messages and transactions

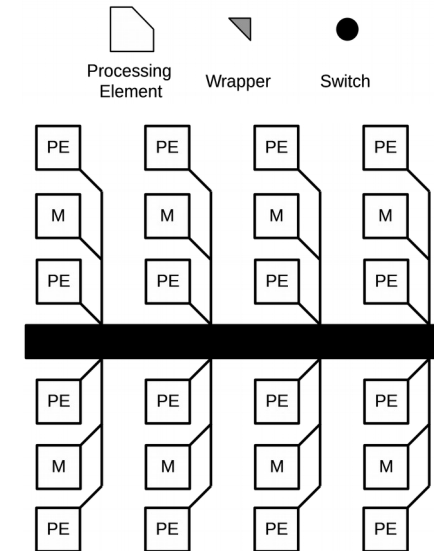
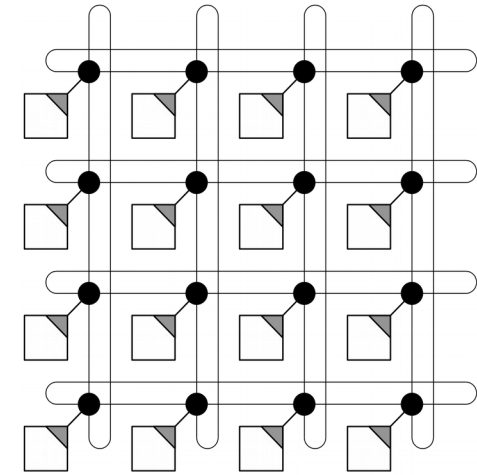


Network on Chip (NoC)

- As more and more complex systems are integrated, interconnection becomes a critical issue
- An NoC is literally a network (usually passing packets) on the chip not unlike the networks of macroscopic systems such as supercomputers, LANs, or the internet
- NoC has become more attractive since bus architectures only allow two devices to communicate at a time
- The on-chip network can be implemented in a variety of ways such as a simple crossbar, Clos, mesh, and so on

Network on Chip (NoC)

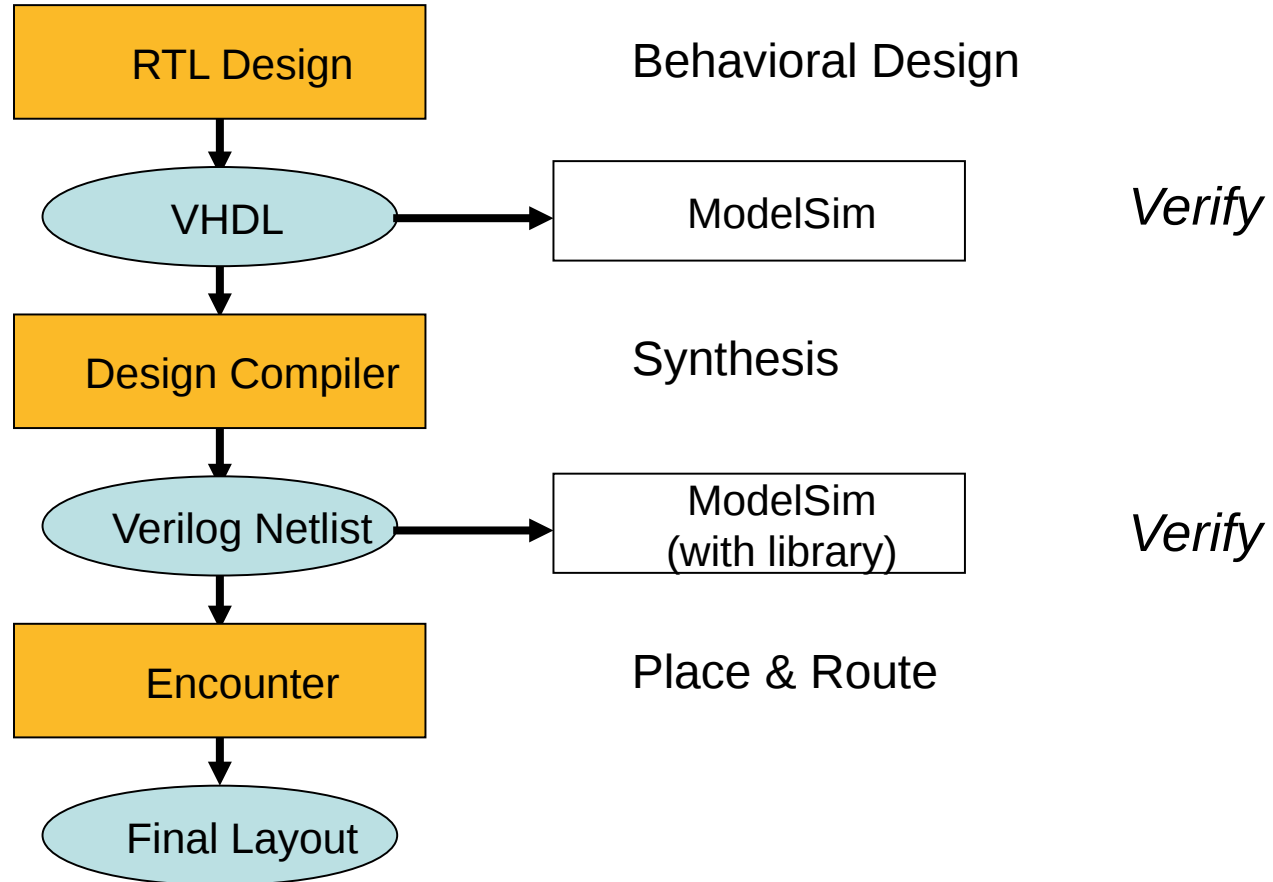
- Networks can be implemented on chip to circumvent issues:
 - Synchronization – NoC may be globally asynchronous
 - Multiple paths to avoid faults and allow many connections
 - Cool, low-power operation



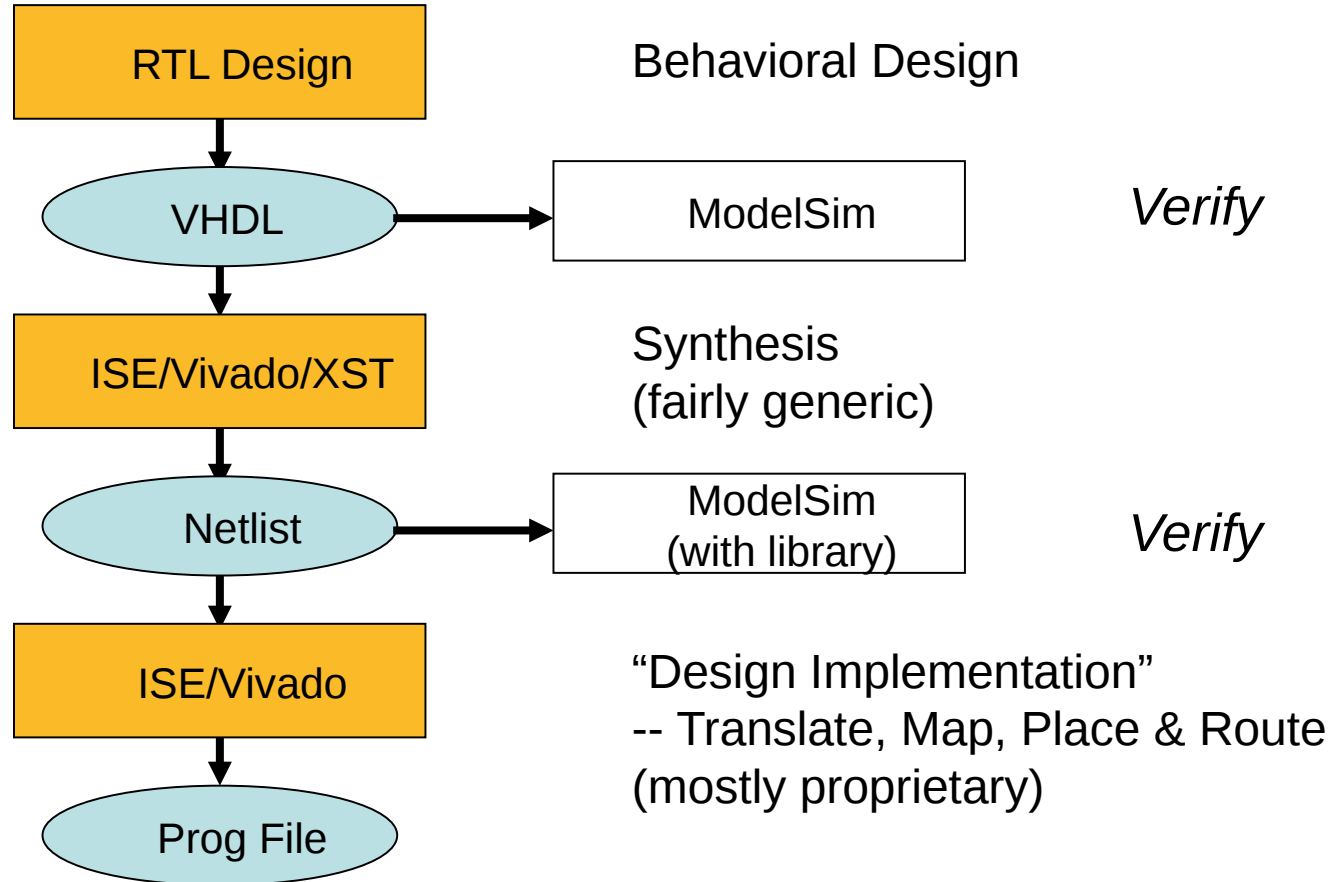
Design Flow Revisited

- Some tools useful in flow:
 - RTL Verification – ModelSim
 - Synthesis – Design Compiler
 - Place & Route – Cadence Encounter
- RTL (register transfer level) code written in VHDL or Verilog using any text editor (e.g., *gedit* on Linux) can be verified using *NCLaunch*, *ISim* or *ModelSim*
- *Design Compiler* takes high level HDL code and synthesizes to a gate-level netlist (this is a Verilog netlist)
- *Encounter* takes the Verilog netlist from *Design Compiler* as input to place and route the final design

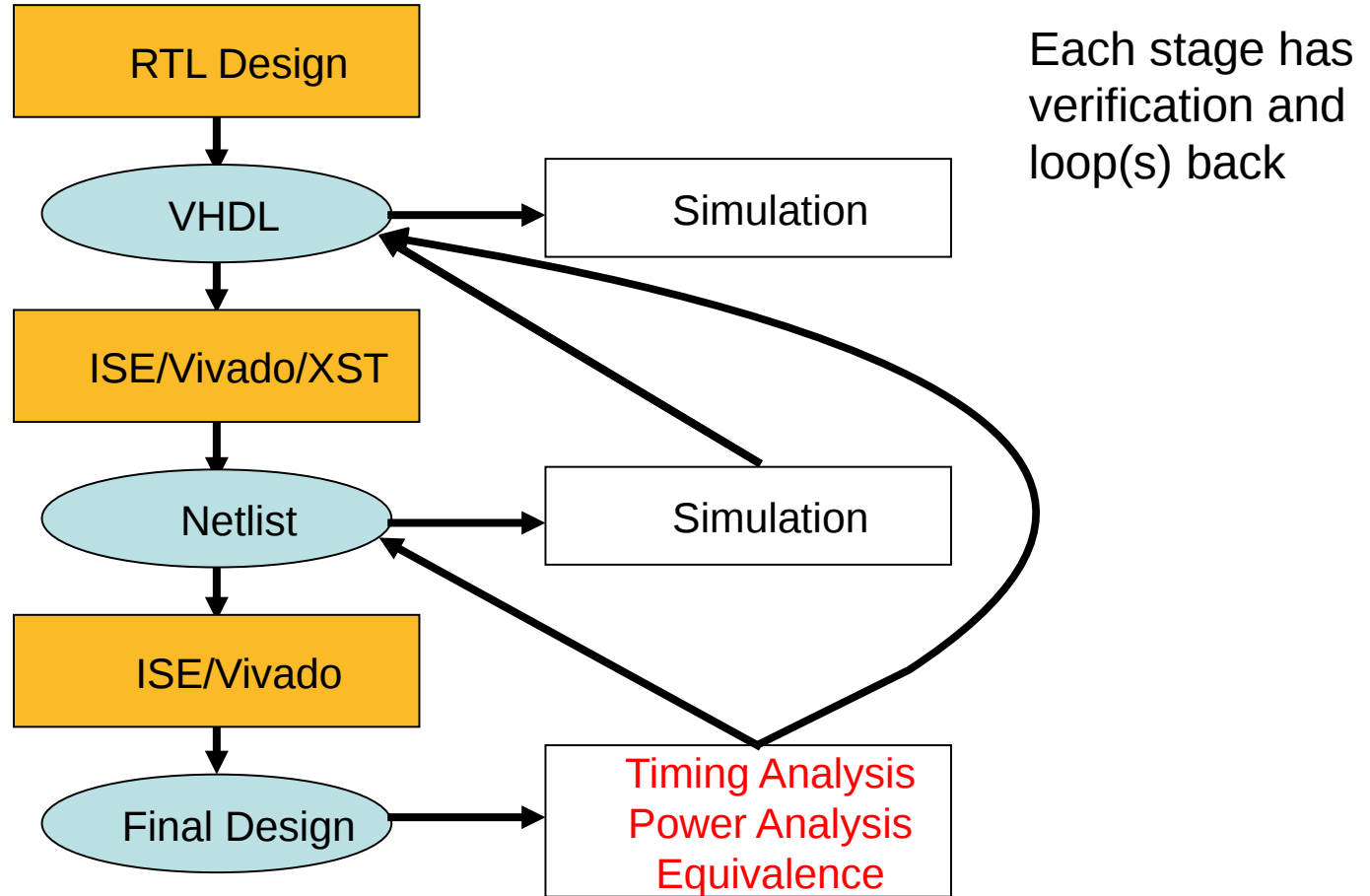
Design Flow Revisited - Silicon



Design Flow Revisited - FPGA



Still More...



What the Designer Controls

- HDL description – likely starts high-level then becomes more structured with time
 - *It all starts here...*
- Constraints – extra files are included with HDL indicating performance targets to synthesis and other tools
 - Timing constraints needed to meet performance targets
 - Pin placement also falls under constraints
 - Can constrain tool to place blocks at certain locations
- CAD tool options – tools can be “tweaked” to use different algorithms, seed parameters, etc.

Coding for Circuits

- The design always begins with the initial behavioral description
 - the RTL code
- The RTL description is a very high level form written in some HDL, either VHDL or Verilog
- RTL describes the design in terms of microarchitectural components such as registers & ALUs
- A lower level HDL form would be a gate level netlist and even lower than that is transistor level
 - netlists can be written in an HDL such as Verilog

Relationship to Security?

- **Fault-injection attacks** – force circuit-level faults to disrupt desired behavior
- **Hardware Trojans** – circuit- and/or logic-level circuits added to a design for malicious purposes
- **Reverse engineering** – need to hide design details at all levels (circuit, logic, RTL, etc.)
- How would you encrypt a circuit (design and implementation)?
- Several mitigation techniques exist for HW security that can be applied at various levels of abstraction