Top-Down Design

Jason Biel
Electrical and Computer Engineering Department
Michigan State University

Design Optimization

- Optimization of
  - Area
  - Speed
  - Power dissipation
  - Reliability
  - Testability
  - Design time

Top-down and bottom-up design

- Bottom-up design creates abstractions from high detailed low-level designs
- Top-down design adds detail as the design progresses
- Can use a combination of both top-down and bottom-up

Bottom-Up Design

- Start with smallest detail and build up to highest abstraction
  - Design individual transistors
  - Combine transistors into gates
  - Defining our own cell library
    - Schematic
    - Symbol
    - Layout
  - Build larger circuits with our cell libraries

Top-Down Design

- Start design from overall description and end design with smallest detail
  - specification
  - architecture
  - logic design
  - circuit design
  - physical layout
- Verify at each level of abstraction

Top-Down Design

- Specification (Words): function, interface, cost, performance, etc.
- Architecture (Drawing,Simulation): large blocks, system level view
- Logic (Schematic,Simulation): gates + registers
- Circuits (Schematic,Simulation): transistor sized for speed, power, area
- Layout: Custom or existing library
- Extracted Layout (Simulation)

Specifications

- Setting Specifications
  - Agreeing with other designers what the interface is
  - Customer interviews
  - Comparison with competitors
- Good Requirements
  - Correct
  - Unambiguous
  - Complete
  - Verifiable
  - Consistent: do not contradict
  - Modifiable: can update easily

Architecture

- Divide and conquer
- Verify by simulating
- Hardware Description Languages (HDL)
  - Coded functional descriptions that can be mapped into hardware
- Two popular HDLs
  - VHDL: higher-level language, describes function at the "behavioral" level
  - Verilog: can describe circuit at behavioral level down to the transistor level. syntax similar to a C program
- Use
  - good for designing/simulating complex circuits before committing to physical design (layout)
  - only good for digital/logic circuits, not analog

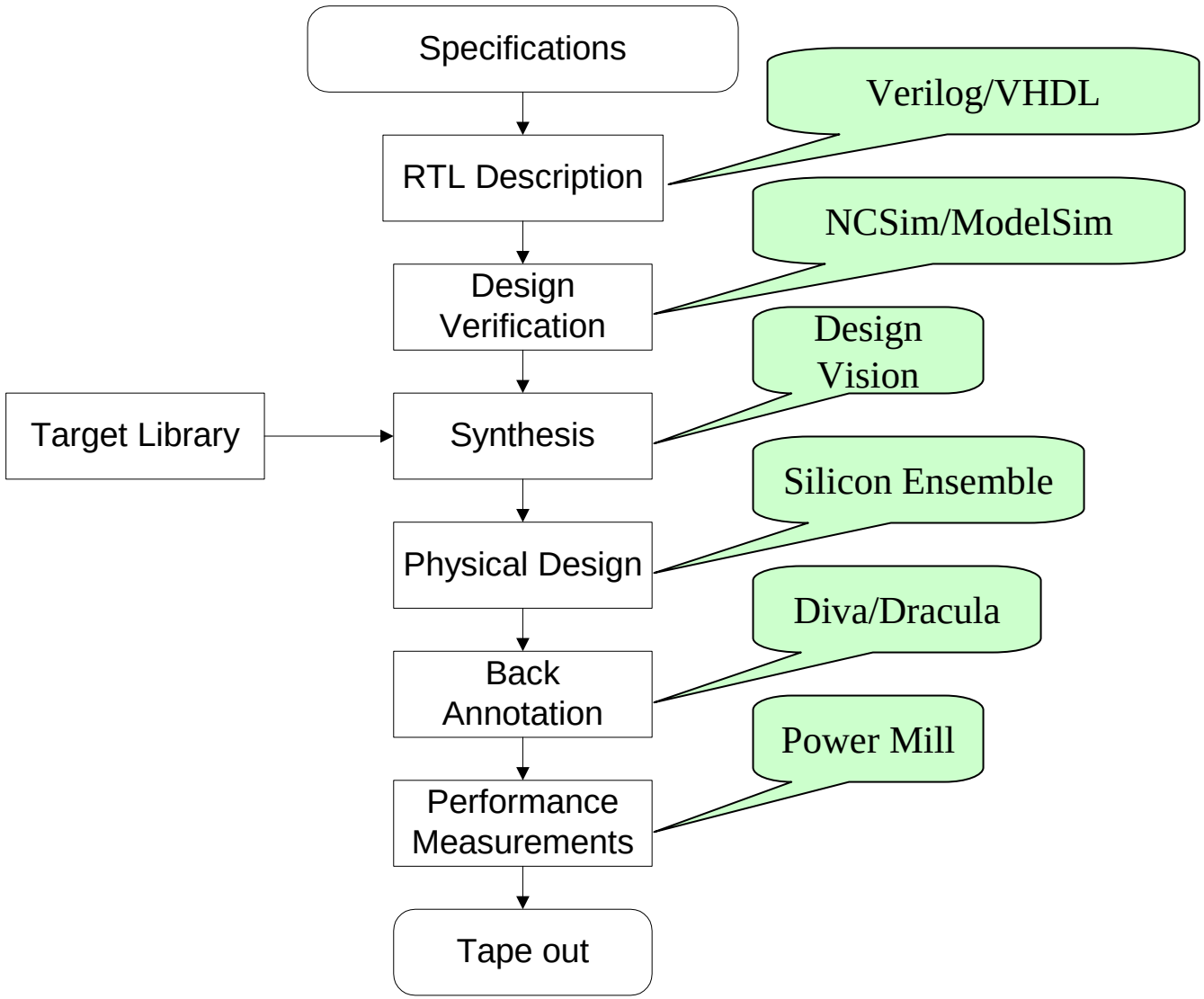
Architecture →Logic Design

- Logic Synthesis
  - Tool: Synopsys Design Vision
  - Input: 1) Circuit described in an HDL
    2) Logic cell library
  - Output: A Verilog file describing a function (circuit) with logic cells from the library

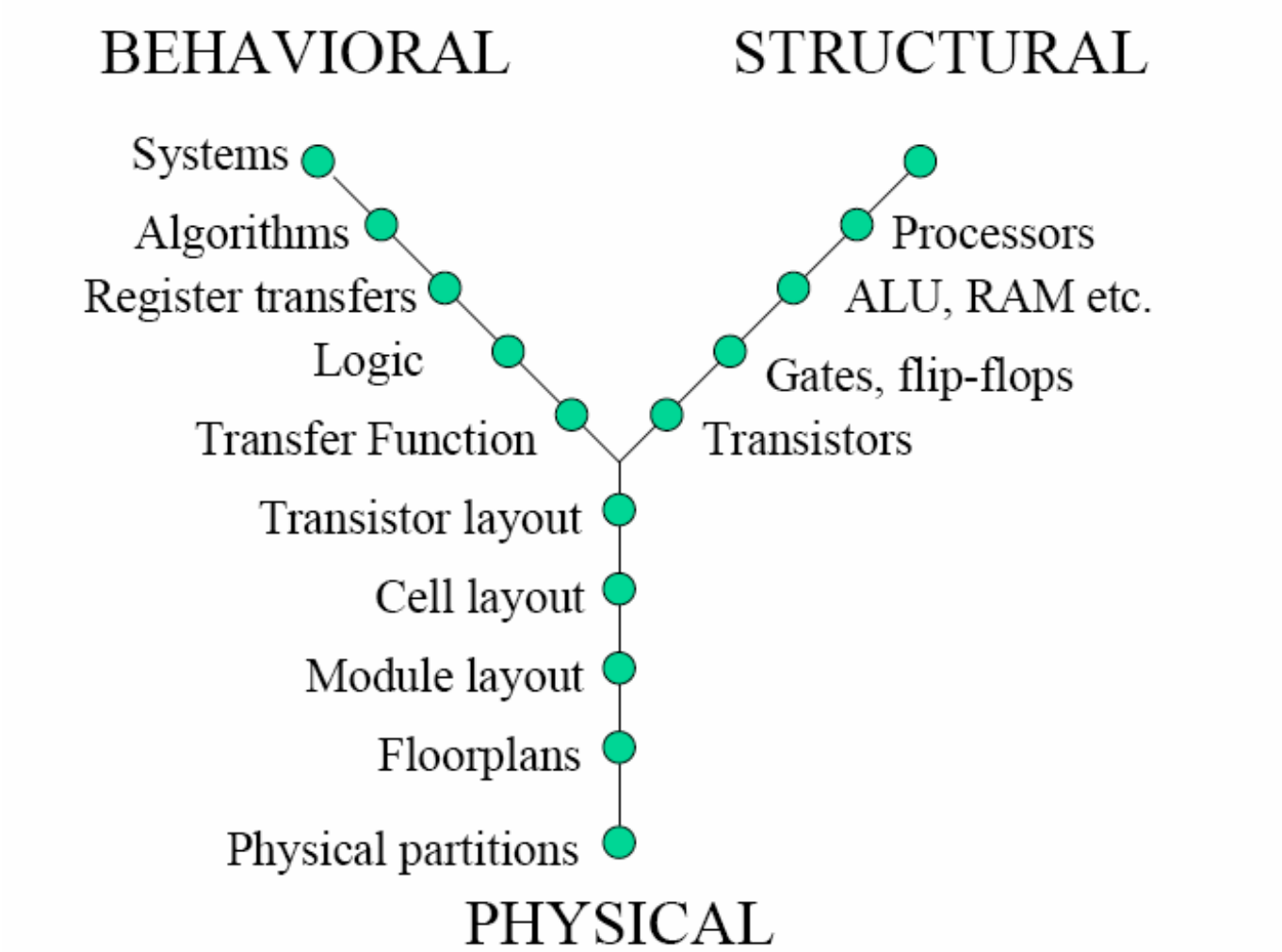
Logic → Physical Design

- Place and Route
  - Tool: Silicon Ensemble by Cadence
  - Input: 1) Synthesized circuit
    2) Layout cell library
  - Output: Physical design of circuit
- Extraction to a Netlist for Simulation
  - Tool: Virtuoso by Cadence
  - Input: Layout (e.g., GDSII format file)
  - Output: Netlist (like SPICE file) of the circuit that can be simulated

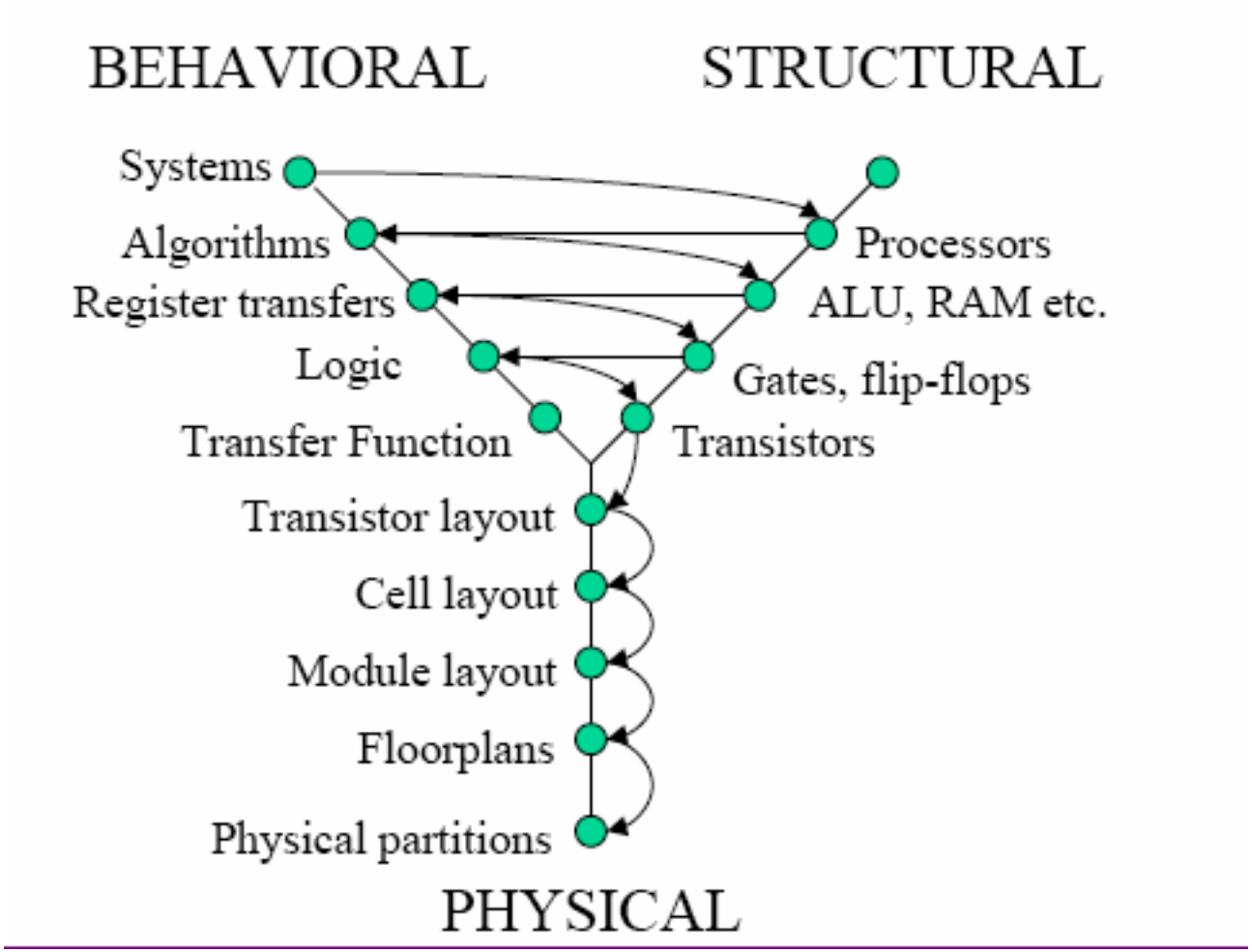
Top-Down Design Flow



Y-Chart


BEHAVIORAL
STRUCTURAL
Systems
Algorithms
Register transfers
Logic
Transfer Function
Processors
ALU, RAM etc.
Gates, flip-flops
Transistors
Transistor layout
Cell layout
Module layout
Floorplans
Physical partitions
PHYSICAL

Top-Down/Bottom UP



Lab 4

- Goal
  - Expose students to top-down design, methodologies to synthesize and place-and-route circuits described by HDL files.
- Procedure
  - Design a multiplexer and 8-bit adder using given VHDL and Verilog files
  - Logic synthesis
  - Place and Route (layout)
  - Functional simulation

Strongly advised: work in the lab while a TA is available to explain top-down design and answer questions

Decoder VHDL Behavioral Description

Example VHDL Code
2-to-4 Decoder

```
library ieee;
use ieee.std_logic_1164.all;

-----------------------------------------------

entity DECODER is
port(   I:  in std_logic_vector(1 downto 0);
        O:  out std_logic_vector(3 downto 0)
    );
end DECODER;

-----------------------------------------------

architecture behv of DECODER is
begin
    -- process statement
    process (I)
    begin
        -- use case statement
        case I is
        when "00" => O <= "0001";
        when "01" => O <= "0010";
        when "10" => O <= "0100";
        when "11" => O <= "1000";
        when others => O <= "XXXX";
    end case;

    end process;

end behv;
```

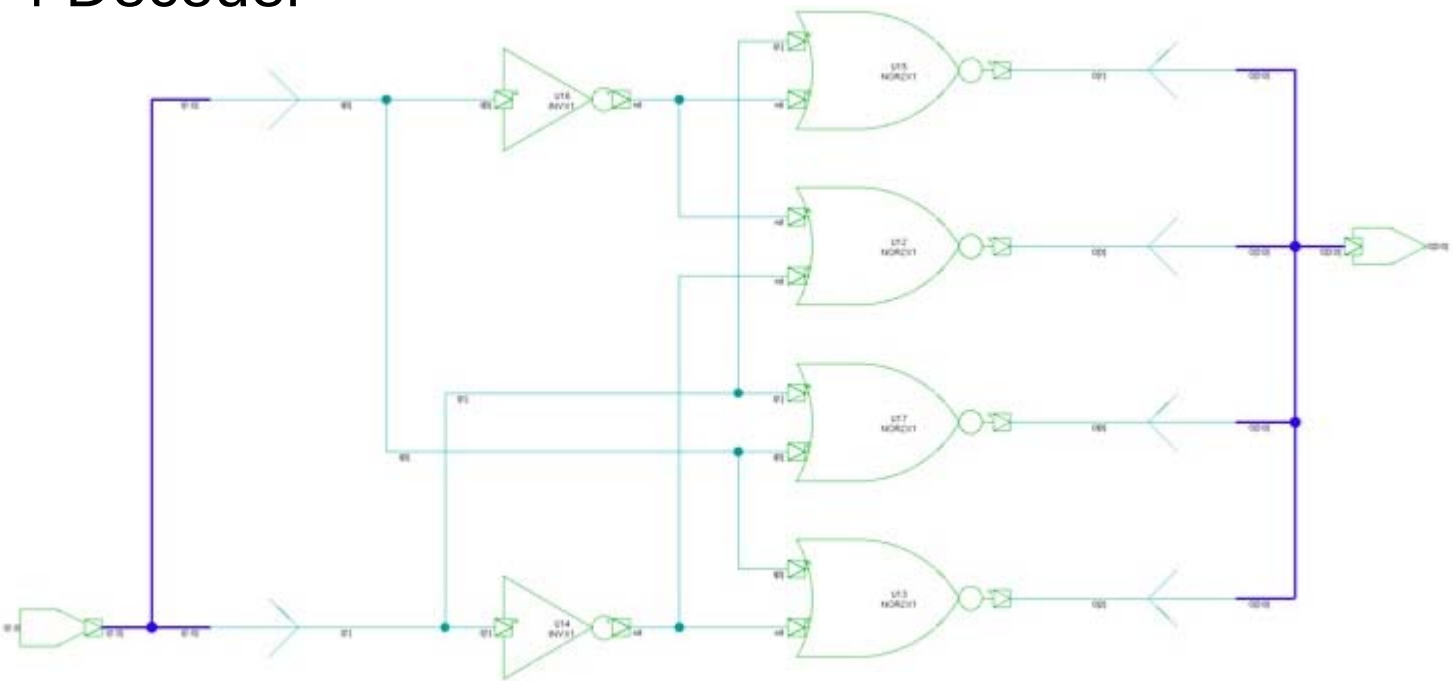
Decoder Synthesized Verilog Structural Code and Schematic

Example Verilog Code
2-to-4 Decoder

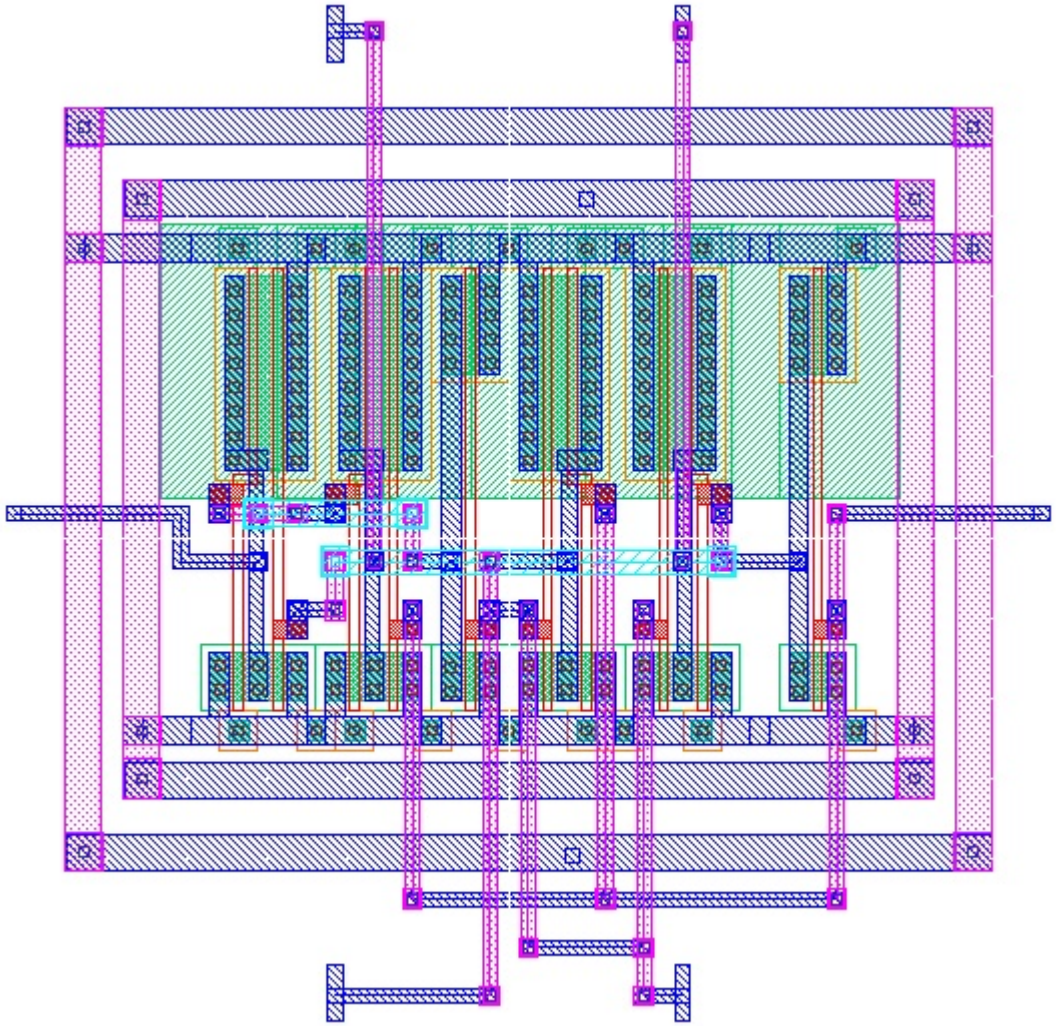
```
module DECODER ( I, O );
  input [1:0] I;
  output [3:0] O;
  wire   n8, n9;

  NOR2X1 U12 ( .A(n8), .B(n9), .Y(O[3]) );
  NOR2X1 U13 ( .A(I[0]), .B(n9), .Y(O[2]) );
  INVX1 U14 ( .A(I[1]), .Y(n9) );
  NOR2X1 U15 ( .A(I[1]), .B(n8), .Y(O[1]) );
  INVX1 U16 ( .A(I[0]), .Y(n8) );
  NOR2X1 U17 ( .A(I[1]), .B(I[0]), .Y(O[0]) );
endmodule
```

Schematic representation
2-to-4 Decoder



Decoder Layout

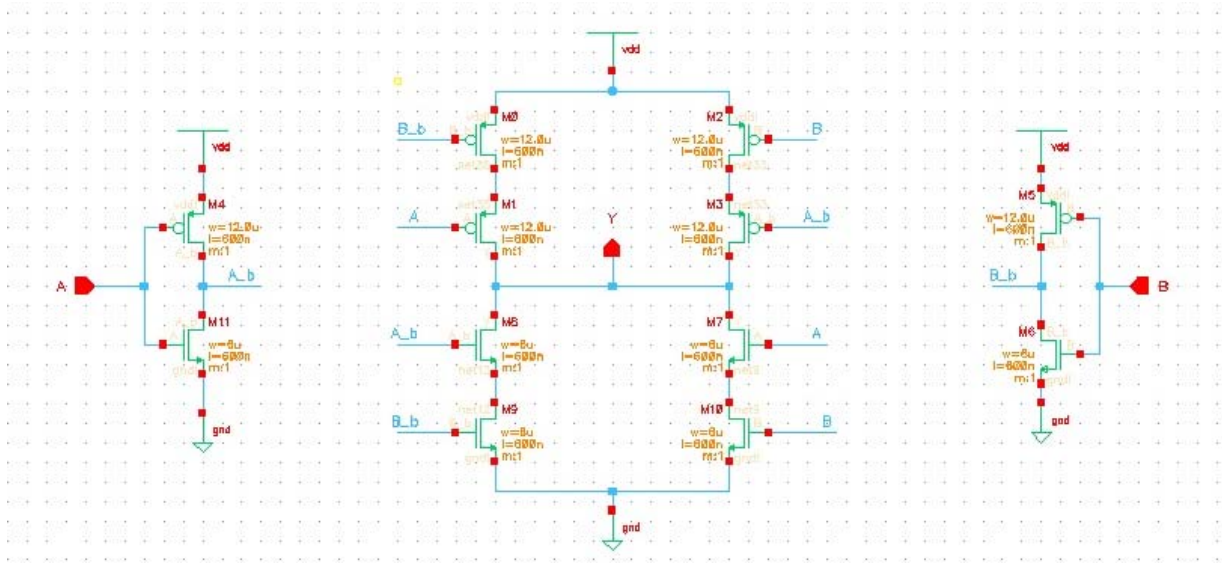


Library Example

From Standard Cell Library

- XOR symbol

- XOR schematic



Library Example

XOR Layout from Standard Cell Library