ECE6703J

Computer-Aided Design of Integrated Circuits

Introduction

Instructor

- Weikang Qian
- Email: qianwk@sjtu.edu.cn
- Phone: 34206765-4301
- Office: Room 430, Long Bin Building
- Office hour
  - Tuesday & Thursday 7:00 – 8:00 pm
  - Or *by appointment*

Logistics

- Time: Tuesday & Thursday 4:00-5:40 pm
- Location: DZY 3-302
  - Only on-site!
- Textbook for Reference (Not Required):
  - "Electronic Design Automation: Synthesis, Verification, and Test (Systems on Silicon)," by Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting Cheng, Morgan Kaufmann Publishing, 2009.
  - "Logic Synthesis and Verification Algorithms," by Gary Hachtel and Fabio Somenzi, Springer Publishing, 1996.
  - "VLSI Physical Design Automation," by Sadiq Sait and Habib Youssef, World Scientific Publishing, 1999.

Teaching Assistant

- Cai, Yifei
  - Email: caiyifei1218@sjtu.edu.cn

Grading

- Composition
  - Quizzes: 4%
  - (About) 6 written assignments: 24%
  - (About) 2 programming assignments: 24%
  - Midterm exam: 24%
  - Final exam: 24%
- We will curve the final grades, if necessary.
- Questions about the grading?
  - Must be mentioned to the instructor within one week after receiving the item.

Assignment Deadline

- Each written assignment must be turned in **electronically** by 11:59 pm on the due date.
- Each programming assignment (PA) must be turned in by 11:59 pm on the due date to be accepted for full credit.
  - However, we still allow you to submit your PA within 3 days after the due date, but there is a late penalty.

Hours Late	Scaling Factor
(0, 24]	80 %
(24, 48]	60 %
(48, 72]	40 %

  - No PA will be accepted if it is more than 3 days late!

Honor Code

- You are allowed to discuss homework in oral with your classmates.

- BUT you cannot **read** another student's solution or **show** another student your solution (either written assignment or programming assignment).

"**Another student**" includes a student in the current semester or in the previous semester.

Honor Code: Teaching and Learning Materials

- Teaching and learning materials, such as lecture slides, assignments, **your solutions**, etc. may not be passed on to others without the expressed permission of the course instructor.
  - In particular, it is not permissible to post lecture slides, assignment questions, project descriptions, solutions, etc. on public sites
  - If you use Github to back up your code, make your repository **private**

Consequence of Honor Code Violation

- Any suspect of honor code violation will be reported to **the Honor Council at JI**.

- **Penalty** of honor code violation
  - Reduction of the grade for this assignment to **ZERO**, **plus**
  - Reduction of the final grade for the course by one grade point, e.g., B+ → C+, for **both students** involved

A Few Suggestions

- Taking notes in class is a good idea.
- Start your programming assignment early!
  - Don't wait until the last minute. Numerous lessons before
- Back up your code frequently in case your computer crashes.
  - … this indicates "computer crash" is NOT a reason for late submission!

Canvas

- Log into Canvas: https://jicanvas.com
- Check the class webpage on Canvas regularly for
  - Announcements
  - Slides
  - Assignments

What Background Do You Need?

- Computer science
  - Basic programming skills (C/C++)
  - Data structures and algorithms
    - Esp. graph and graph algorithms
  - No worries: See some relevant slides uploaded into Canvas folder Files/Data-Structures-and-Algorithms
- Computer engineering
  - Basic digital design (gates, flip flops, Boolean algebra)
  - Combinational and sequential logic design
- Mathematics
  - Discrete mathematics: Boolean logic, combinatorics
  - Continuous: basic calculus, linear algebra, matrix
- (Optional) Basic VLSI knowledge
  - Some chip layout experience is nice, but not essential

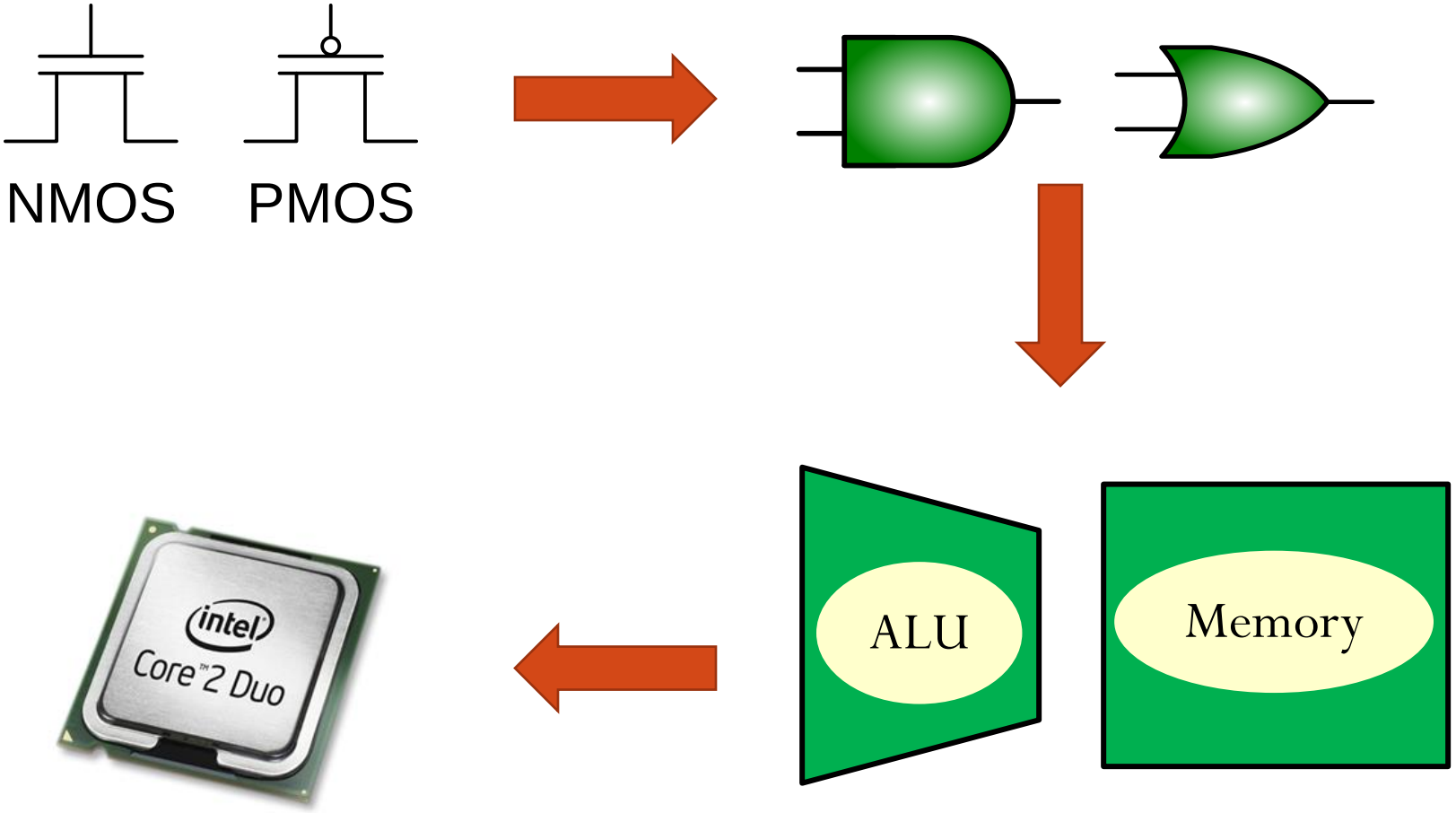
References and Copyright

- Slides used (Modified when necessary)
  - Rob Rutenbar, University of Illinois at Urbana-Champaign
  - Kia Bazargan, University of Minnesota
  - Hai Zhou, Northwestern University

Integrated Circuits

- Integrate circuits (IC) are everywhere in our daily lives.

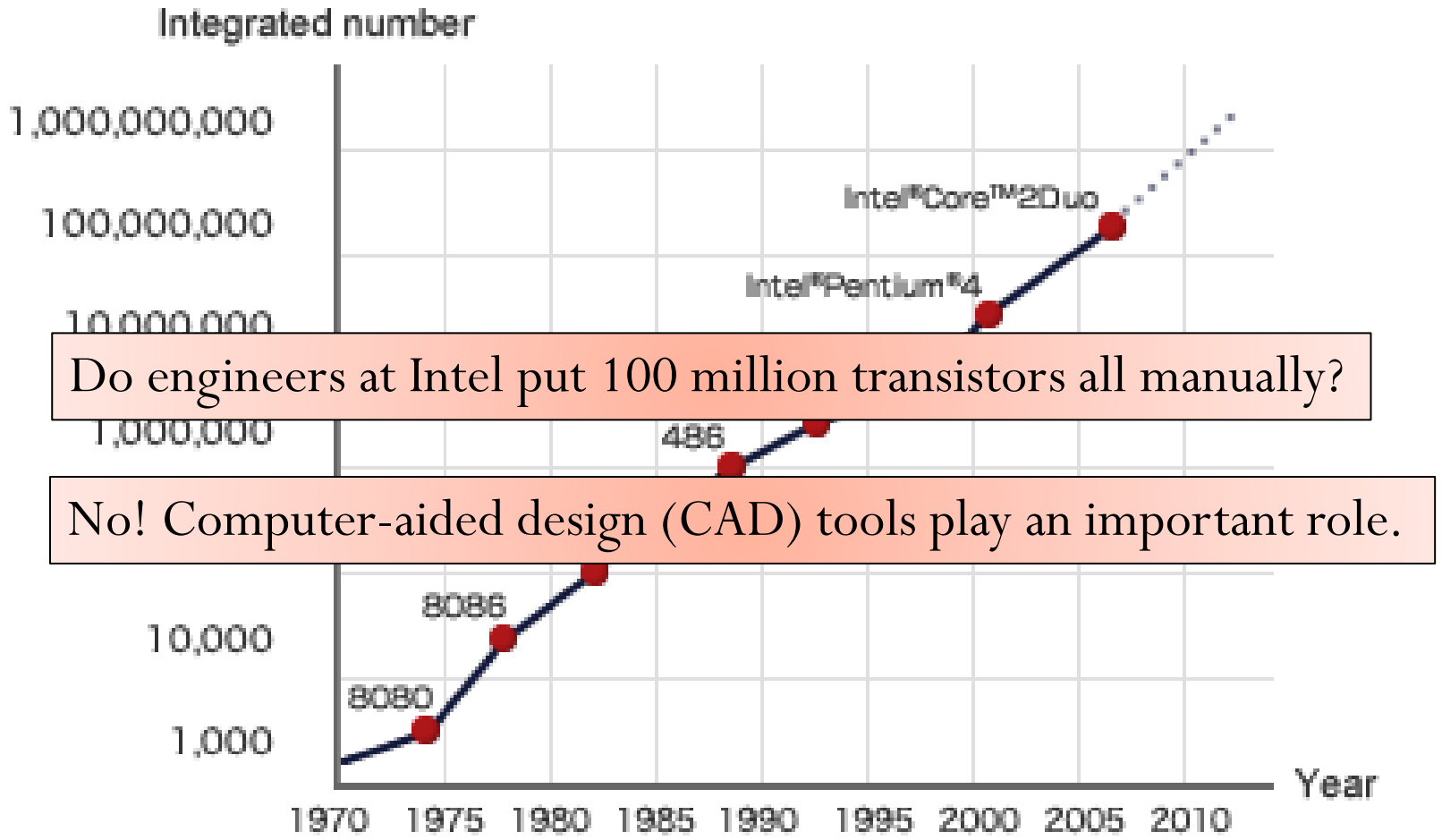
From Transistors to CPU



How Many Transistors on a CPU?

Microprocessor	Year	Transistors
4004	1971	2,300
8080	1974	4,500
8086	1978	29,000
Intel286	1982	134,000
Intel386	1985	275,000
Intel486	1989	1,200,000
Intel Pentium	1993	3,100,000
Intel Pentium II	1997	7,500,000
Intel Pentium III	1999	9,500,000
Intel Pentium 4	2000	42,000,000
Intel Itanium 2	2003	220,000,000

Moore's Law



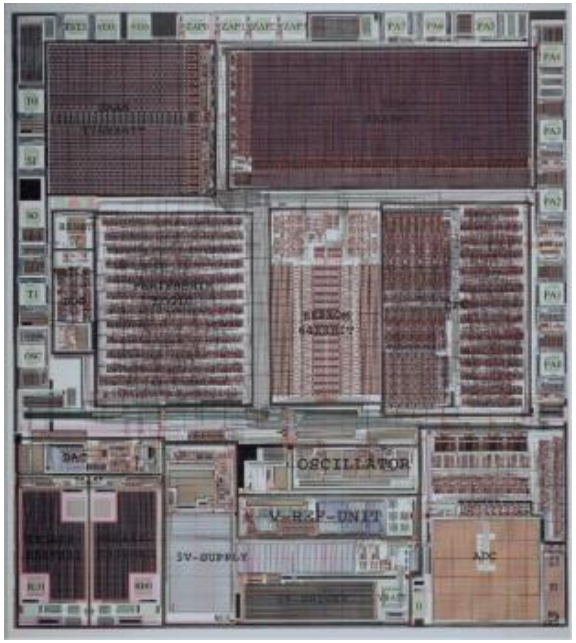
Do engineers at Intel put 100 million transistors all manually?

No! Computer-aided design (CAD) tools play an important role.

What's this Course About?

- Computer-aided design (CAD) tools for very large scale integrated (VLSI) circuits.
  - Indeed, we focus on the algorithms.
  - Many tools: logic synthesis, placement
- CAD is also known as electronic design automation (EDA).

Designer Specification → **VLSI CAD tools** →

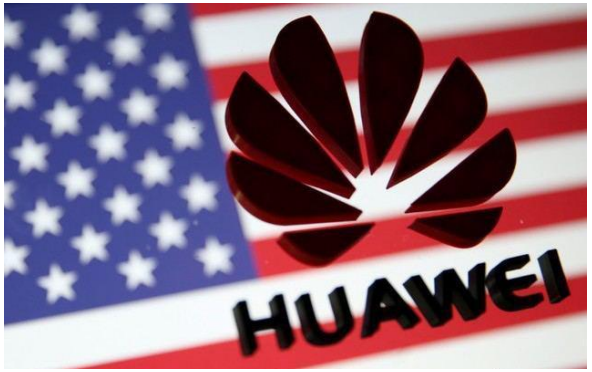


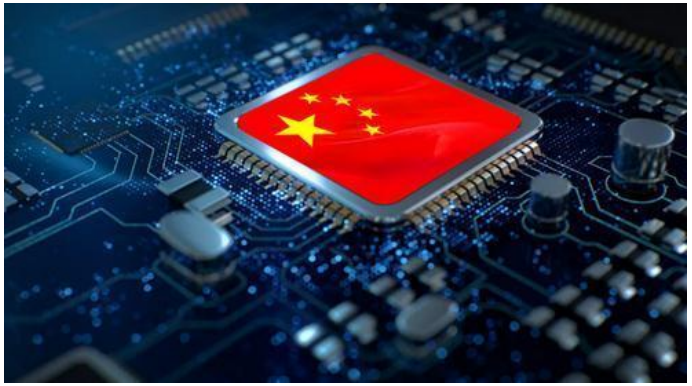
EDA is Much Needed in China!

- Big Three

cadence™ SYNOPSYS®

Mentor Graphics®





A Tale of Two Design Types

- ASIC: Application-specific integrated circuit.
  - An integrated circuit customized for a particular use, rather than intended for general-purpose use.
  - Example: DSP chips, microprocessors, memories.

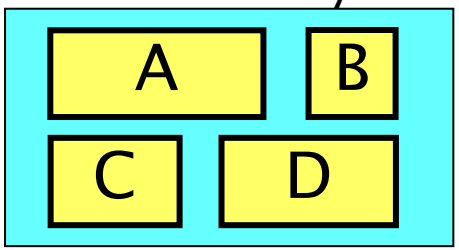
- FPGA: Field-programmable gate array.
  - **Programmable** logic blocks and **programmable** interconnects which allow the same FPGA to be used in many different applications.

ASIC Design Styles

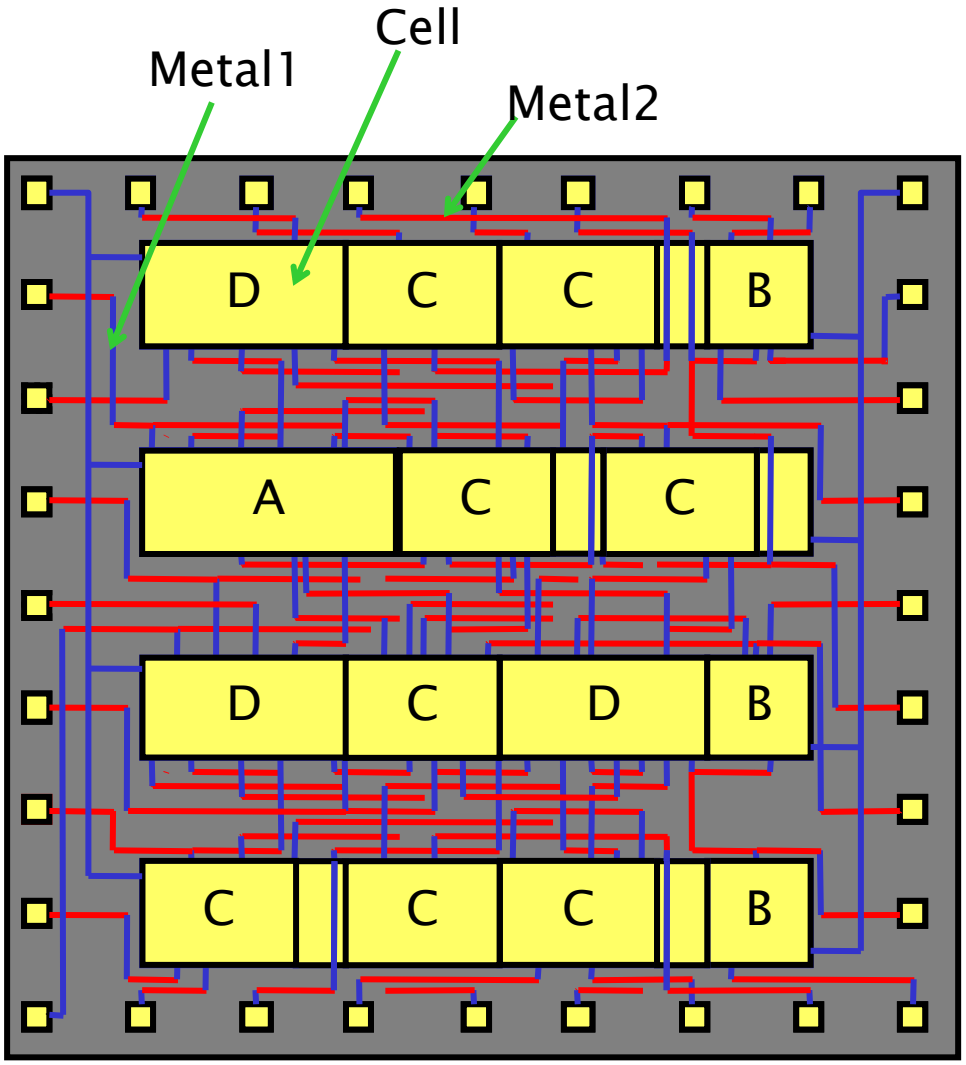
- Full-custom design
  - Transistors are hand-drawn.
  - Best performance (area, speed, etc.).
  - Costly. Only affordable when producing in large volume.
  - Today, only things like microprocessors are full-custom.

- Semi-custom design
  - Try to design by reusing some already designed parts.
  - Not quite as dense (transistors / area) or as fast (GHz) as full-custom design.
  - One type: **standard-cell design**.

Standard Cell Design



This is the design style we will focus on in this course.



Comparison of ASIC Design Styles

Design Methods	Cost / Development Time	Performance	# Companies Involved
Full-Custom	Large	Best	Few
Semi-Custom	Small	Good	Many

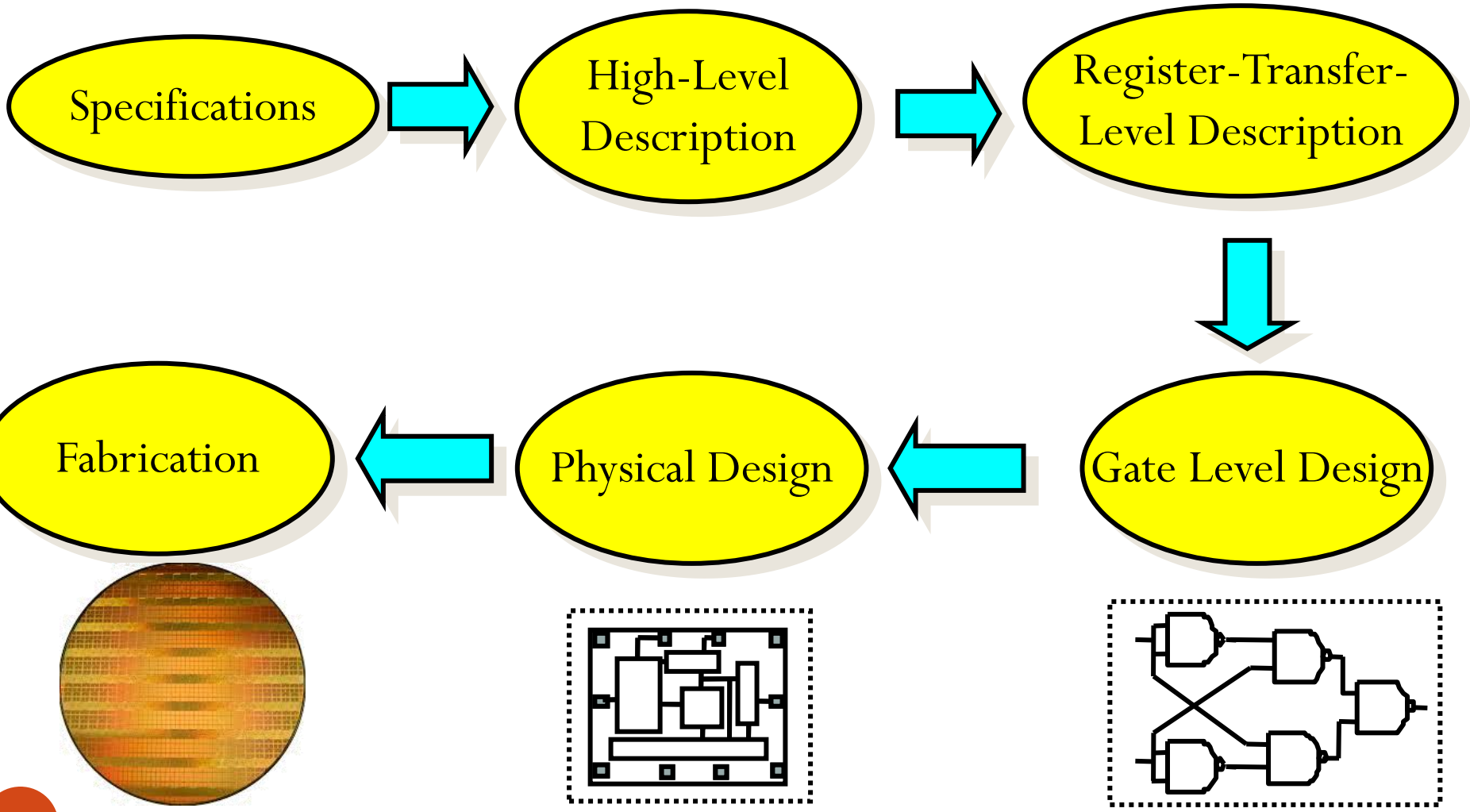
IC Design Steps

SystemC
SystemVerilog

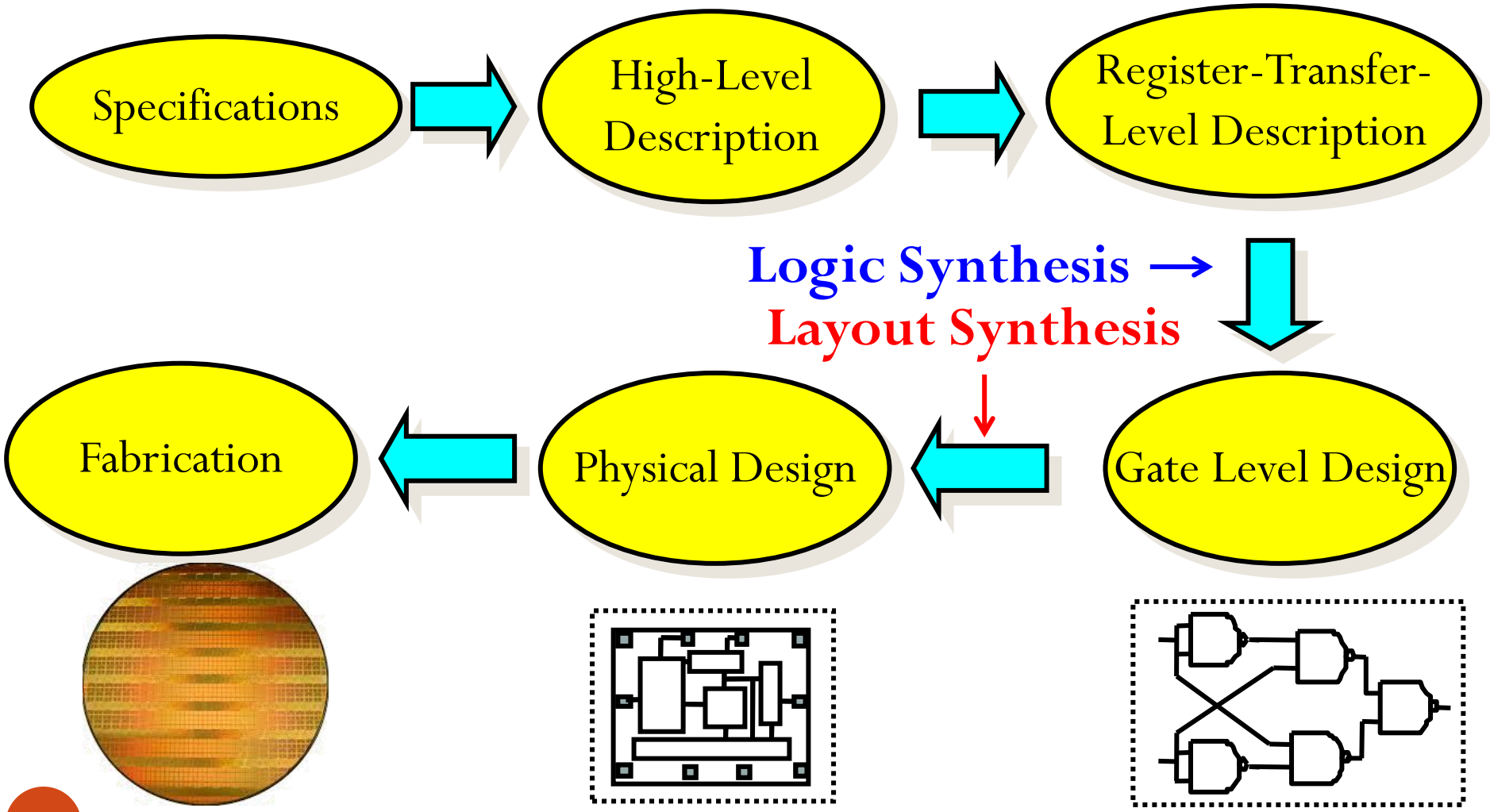
Verilog
VHDL

```
module mux( input a,
    input b, input sel,
    output z, output zbar);
    assign z = sel ? b : a;
    assign zbar = ~z;
endmodule
```

IC Design Steps

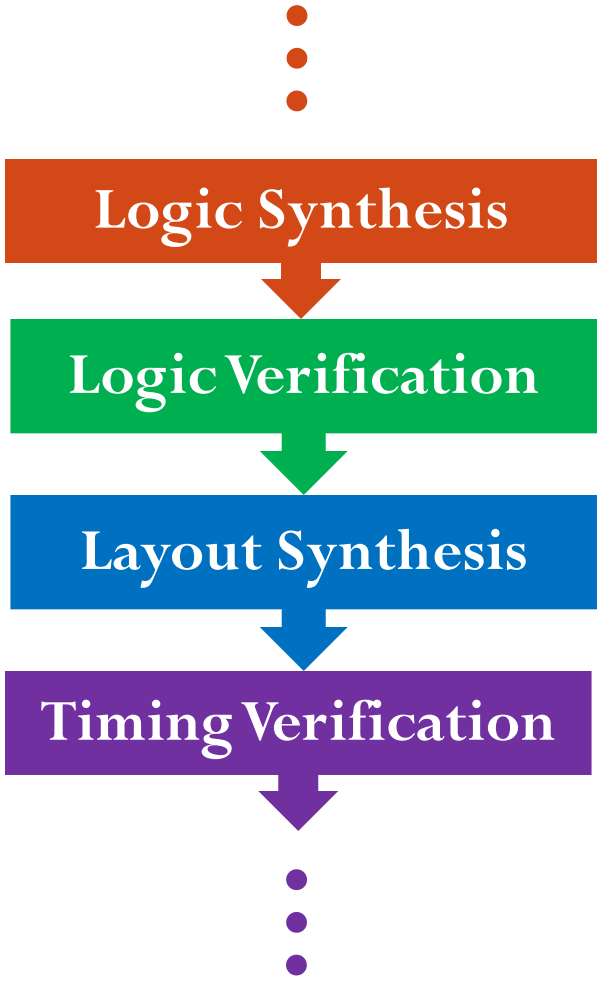


Two Major Components of this Course

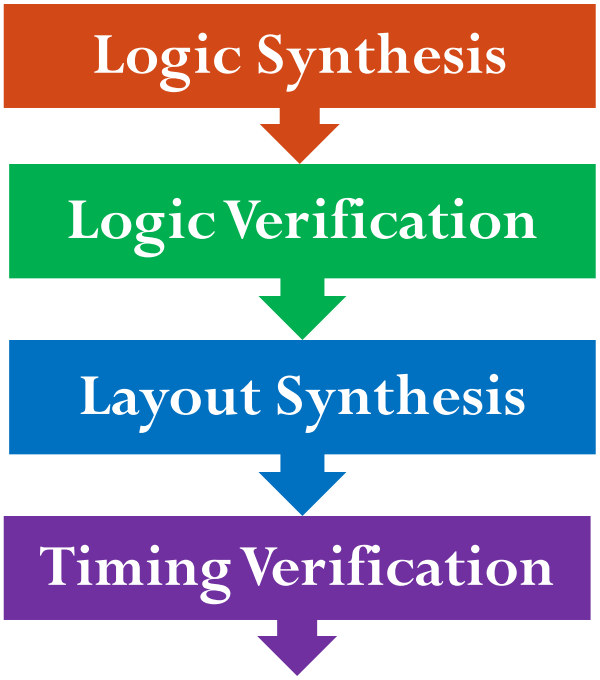


Synthesis and Verification

- Synthesis
  - Go forward in design: Make new stuff.
- Verification
  - Look backward: Check that it worked.
  - Important! Make sure that the design goals such as function and speed are achieved.
  - Lesson: The Pentium FDIV bug
    - (4195835 * 3145727) / 3145727 = 4195579 **Wrong!**
    - Loss to Intel: $475 million!



The Focus of Our Course



- Start with some Boolean / logic design description …
- …end with gates+wires, located at (x,y) coordinates on chip

- Big goals
  - Explain the critical algorithms, data structures, and modeling assumptions used in each of these big steps

Course Topics

- Logic Synthesis and Verification
  - Computational Boolean Algebra
  - Binary Decision Diagram
  - Boolean Satisfiability
  - Two-level Logic Synthesis
  - Multi-level Logic Synthesis
  - Technology Mapping
- Layout Synthesis
  - Placement
  - Routing
- Timing Analysis

Questions?

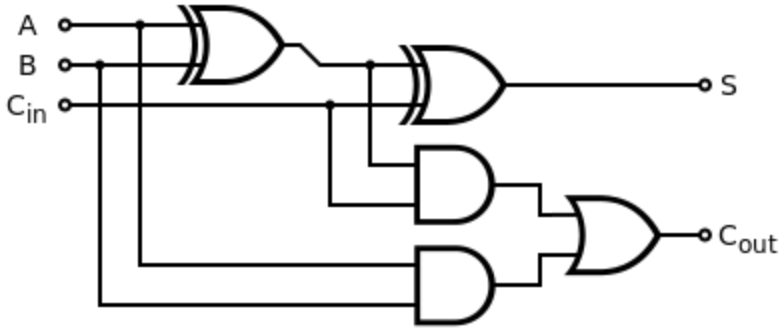
Representation of Digital Circuits

- We will deal with digital circuits.
  - How can we represent digital circuits in **computer memory**?

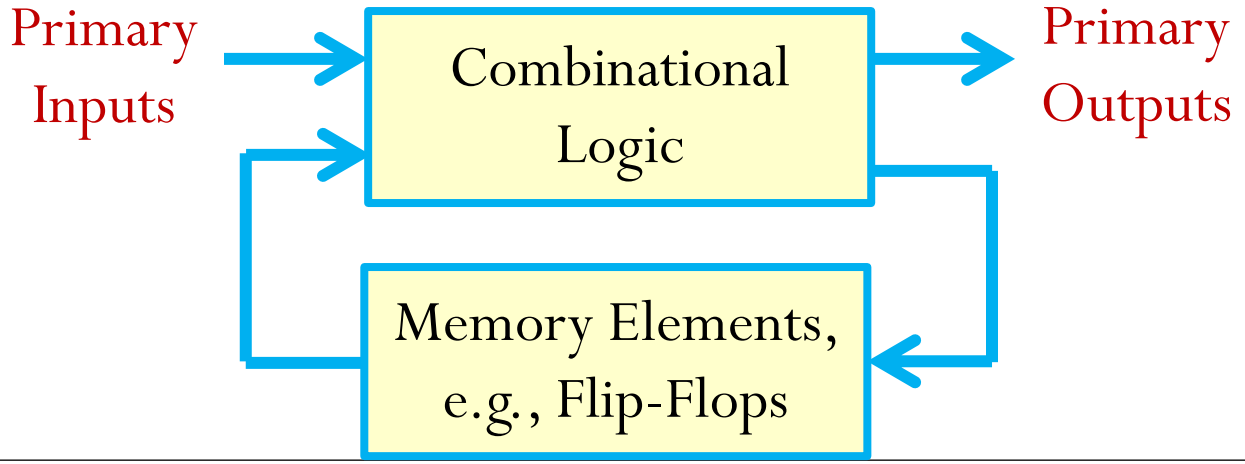
- As we know, there are two types of digital circuits
  - Combinational circuits
  - Sequential circuits

Two Types of Digital Circuits

- Combinational circuits
  - Current outputs only depend on **current** inputs, not on the **previous** inputs.

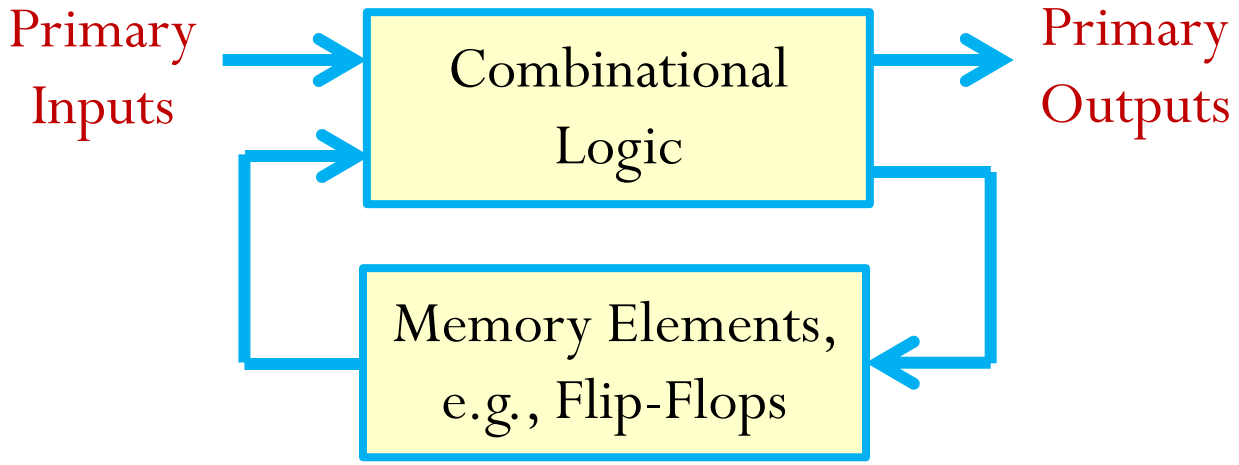


- Sequential circuits
  - Current outputs depend on the previous inputs.



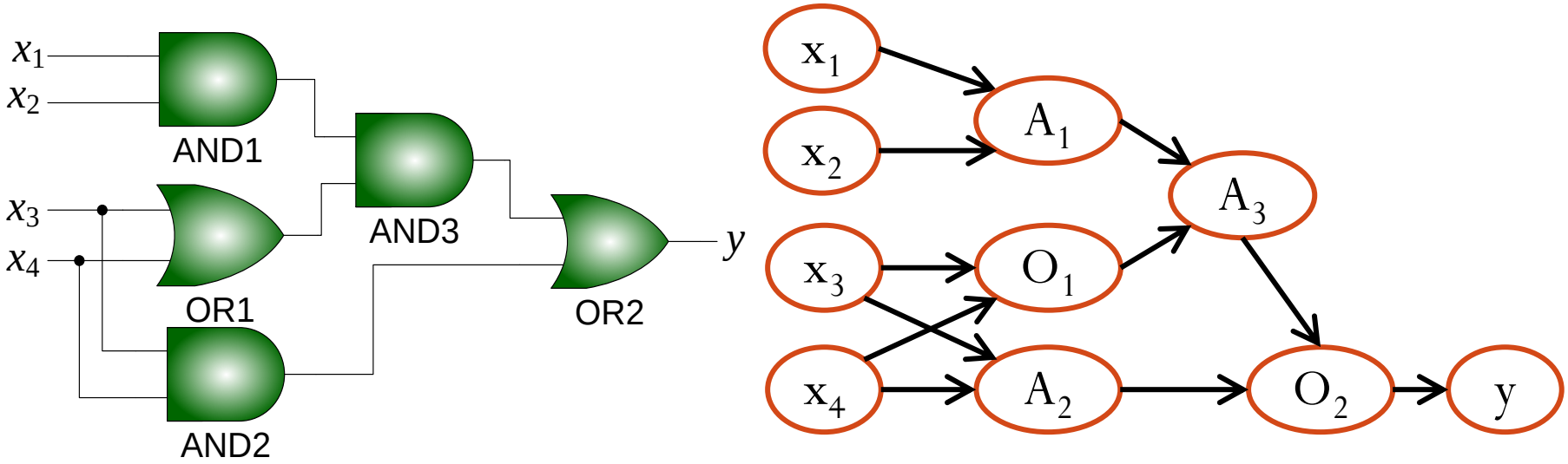
Sequential Circuits

- The **combinational part** of a sequential circuit is critical.
- For the **combinational part** of a sequential circuit,
  - its inputs include **primary inputs** and the **outputs** of the memory part.
  - its outputs include **primary outputs** and the **inputs** to the memory part.



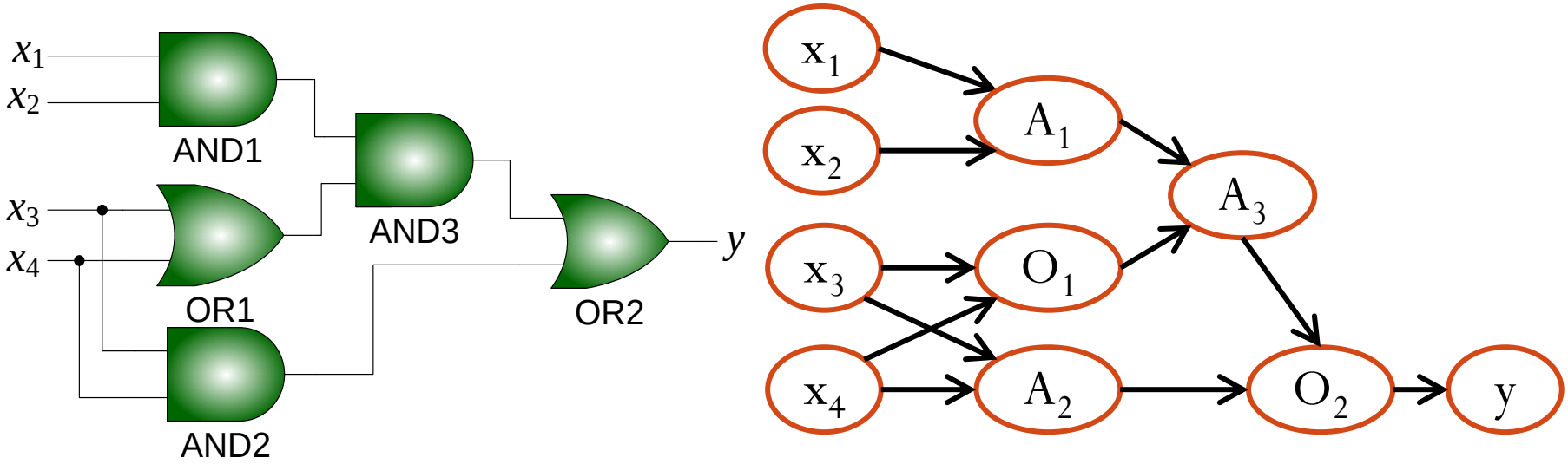
Representation of Combinational Circuits

- Represented as a directed graph.
  - Inputs, outputs, and gates ➔ nodes
  - Wires ➔ directed edges.
    - Why directed edges? Signal flow has direction.



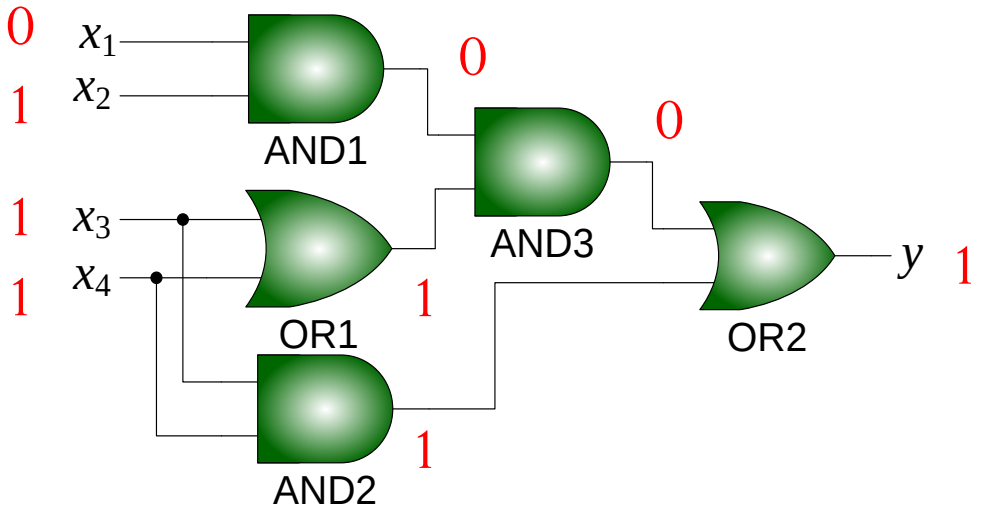
Representation of Combinational Circuits

- Since a combinational circuit has no loops, the corresponding graph is a **directed acyclic graphs (DAG)**.
  - DAG: A directed graph with no cycles.



Traversal of Combinational Circuits

- Many operations on combinational circuit need to traverse it.
- Example: obtain the output given an input pattern.

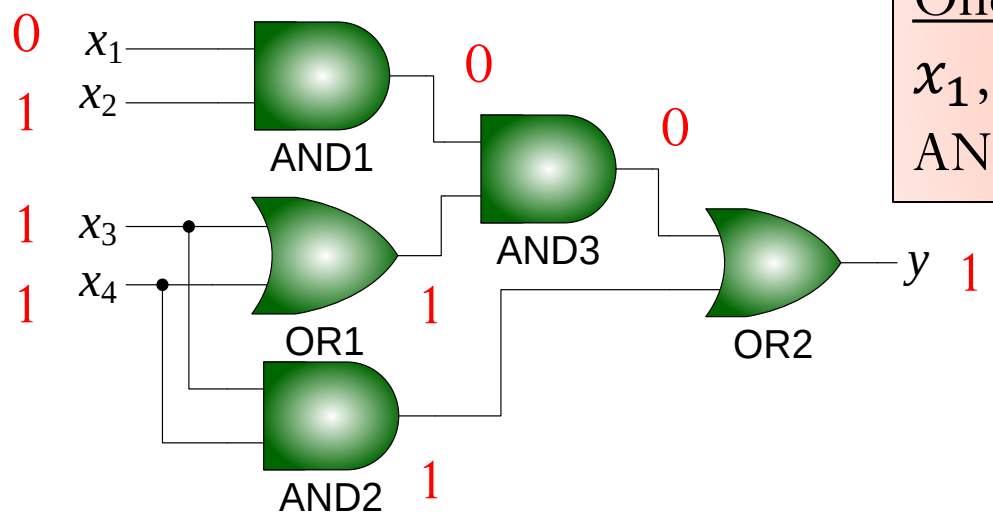


How to approach this kind of traversal as a computation?

Answer: Topological Sorting.

Topological Sorting

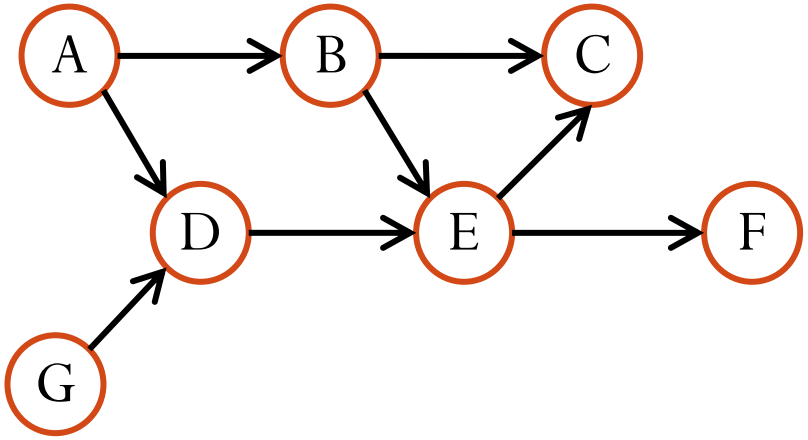
- Observation:
  - The output of each gate can be computed only when **all** of its input are known. ➔ We have to obtain its inputs before obtaining output.
  - In other words, if there is a wire (edge) from gate (node) u to v, then gate (node) u should be visited before gate (node) v.
    - This is exactly **topological sorting**.



One possible visiting order:
x_1, x_2, x_3, x_4, AND1, OR1, AND2, AND3, OR2, y.

Topological Sorting

- **Topological sorting**: an ordering on nodes of a **directed acyclic graph** so that **for each** edge (v_i, v_j) (means: an edge **from** v_i **to** v_j) in the graph, v_i is before v_j in the ordering.
  - Also known as **topological ordering**.



A topological sorting is: A, G, D, B, E, C, F

Topological Sorting: Algorithm

- Based on a **queue**.
- Algorithm:
  1. Compute the in-degrees of all nodes. (**in-degree**: number of **incoming** edges of a node.)
  2. **Enqueue** all in-degree 0 nodes into a queue.
  3. While queue is not empty
     1. **Dequeue** a node v from the queue and visit it.
     2. Decrement in-degrees of node v's neighbors.
     3. If any neighbor's in-degree becomes 0, **enqueue** it into the queue.

Circuit Netlist Format

- We not only need to store a circuit in computer memory, but also need to store it as a text file, very frequently.
  - … to be processed by different programs, e.g., layout synthesis tool, SPICE, schematic viewer, etc.
  - Such files essentially store a netlist of gates.

- Many formats exist:
  - Berkeley Logic Interchange Format (BLIF)
  - Structured Verilog Format
  - Bench Format

Example: Bench Format

```
INPUT(x1)
INPUT(x2)
INPUT(x3)
INPUT(x4)
OUTPUT(y)
g1=AND(x1,x2)
g2=OR(x3,x4)
g3=AND(x3,x4)
g4=AND(g1,g2)
y=OR(g3,g4)
```