## Course Goals

- Understand key components in VLSI designs
- Become familiar with design tools (Cadence)
- Understand design flows
- Understand behavioral, structural, and physical specifications
- Be able to apply VLSI design practices
- Be able to contribute to an industry digital VLSI design project



## Lab

- No lab this week
- Make sure that you have an engineering UNIX account (for more details please contact the TA)
- Contact ECS if you don't



## Topics

- MOS (Metal Oxide Semiconductor) transistor theory
- Transistor characterization and performance estimation
- CMOS logic design
- VLSI design methodologies
- VLSI subsystem design
- Verification
- Testing



## What is VLSI design?

- The process of creating an integrated circuit from specifications to fabrication

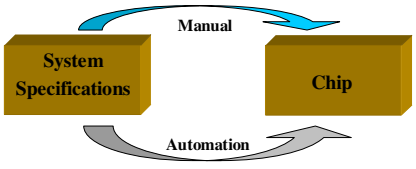
## What is an integrated circuit?

- A single integrated component that contains all the primary elements of an electrical circuit: transistors, wiring, resistors, capacitors, etc.



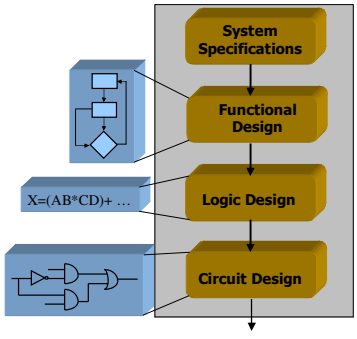
## VLSI Design Automation

- Large number of components
- Optimize requirements for higher performance
  - Performance relates to speed, power and size.
- Time to market competition
- Cost
  - Using computer makes it cheaper by reducing time-to-market.

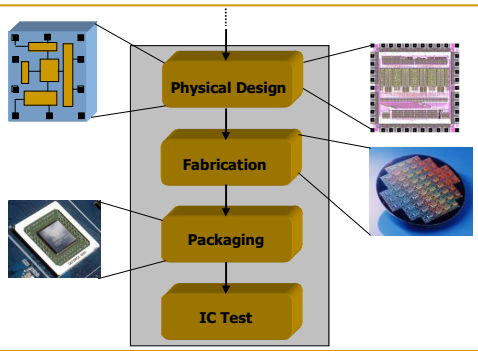





## VLSI Design Cycle

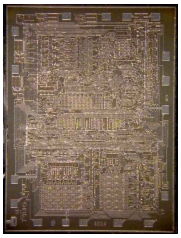


## VLSI Design Cycle





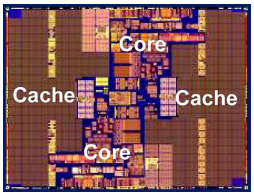

## Intel 4004



- First microprocessor
- Designed in 1971
- 2300 transistors
- 10-um process
- ~100 KHz



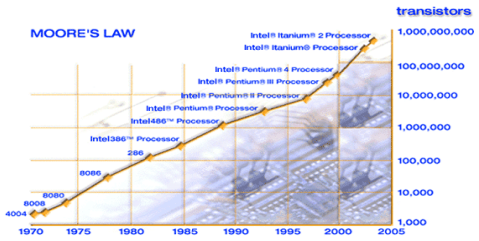
## Intel Itanium Processor




- Released in 2005
- 1.72 Billion transistors
- 90-nm process
- 2 GHz



## Moore's law




- The number of transistors per die doubles every 18-24 months – same for performance



## Semiconductor Technology Scaling

- A new process generation is released every two to three years
  - –2X smaller transistors
  - –1.5X faster transistor performance
  - –1.3X lower operating voltage
  - –3X lower switching power
  - –2X higher chip density



## Billion-Transistor Integration

**How do we do it?**

- Circuit Partitioning
- Computer-aided design tools
  - Automation
- Design methodology
- Verification
- Testing

## Partitioning

- Break the problem into smaller more manageable parts
- Reuse logic modules
- Simplifies layout
- But it is not always easy deciding how to partition the problem
  - It's a very complex problem
  - It's NP hard
- Discussed in details in "CAD Algorithms" course



## Computer-aided design

- Schematic capture
- Synthesis
- Simulation
- Verification
- Automated placement and routing



## CAD Tools

- Cadence
- Synopsys
- Mentor graphics
- Magma
- Start-ups developed tools
- In-house tools



## Design methodology

- Functional specification
  - What does the chip do?
- Behavioral specification
  - How does it do it? (abstractly)
- Logic design
  - How does it do it? (logically)
- Layout
  - How does it do it? (physically)



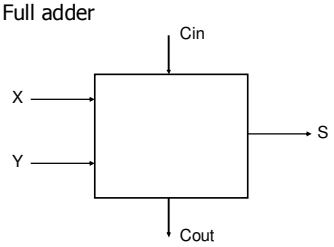
## Design constraints

- Budget
  - Total cost
- Silicon area
- Power requirements
  - Dynamic
  - Static
- Speed
  - Performance
- Schedule
  - Time to market



## Functional specification

- Full adder



## Behavioral specification

- VHDL
- Verilog

```
entity adder is
    -- i0, i1 and the carry-in ci are inputs of the adder.
    -- s is the sum output, co is the carry-out.
    port (i0, i1 : in bit; ci : in bit; s : out bit; co : out bit);
end adder;
architecture rtl of adder is
    begin -- This full-adder architecture contains two concurrent assignment.
     -- Compute the sum. s <= i0 xor i1 xor ci;
     -- Compute the carry. co <= (i0 and i1) or (i0 and ci) or (i1 and ci);

end rtl;
```



## Behavioral specification

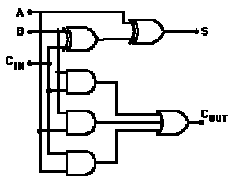
```
module fulladder (a,b,cin,sum,cout);
  input a,b,cin;
  output sum,cout;

  reg sum,cout;
  always @ (a or b or cin)
  begin
    sum <= a ^ b ^ cin;
    cout <= (a & b) | (a & cin) | (b & cin);
  end
endmodule
```



## Logic design

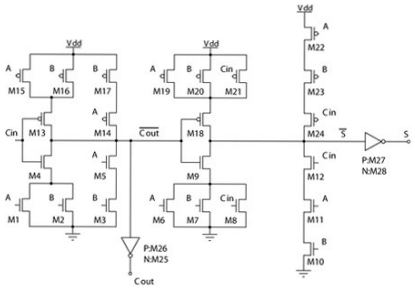
**Full Adder Truth Table**

| CARRY IN | input B | input A | CARRY OUT | SUM digit |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



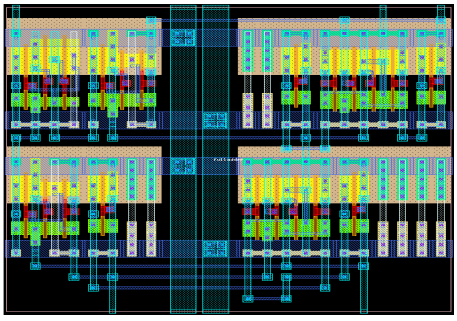



## Transistor schematic






## Layout

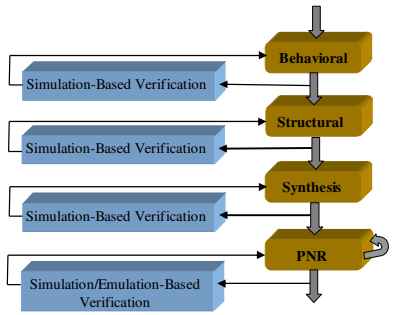




## Testing and verification

- Needed at every phase of the design
- Important to ensuring design robustness
- Critical to keeping to budget
- Time-consuming and high-cost

## Design Process is Iterative




PNR: Placement and routing



## VLSI design methodologies

- Full custom
  - Design for performance-critical cells
  - Very expensive
- Standard cell
  - Faster
  - Performance is not as good as full custom
- Gate array
- Field Programmable Gate Array



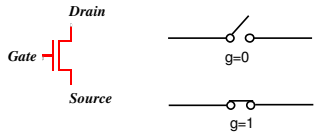
## Comparison of Design Styles

| | Full Custom | Standard Cell | Gate Array | FPGA |
|---|---|---|---|---|
| Area | Compact | Moderate | Moderate | Large |
| Performance | High | Moderate | Moderate | Low |
| **Production Volume:** | **Mass Production Volume** | **Medium Production Volume** | **Medium Production Volume** | **Low Production Volume** |
| **Complexity:** | **High** | | | **Low** |



## nMOS transistor

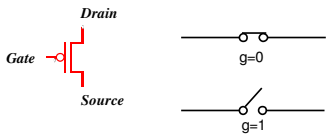
- If the gate is "high", the switch is on
- If the gate is "low", the switch is off





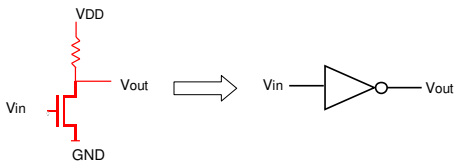

## pMOS transistor

- If the gate is "low", the switch is on
- If the gate is "high", the switch is off






## Logic gates in nMOS

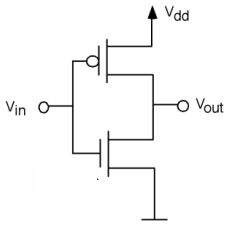


## Problems with nMOS

- Static current flow
  - When the transistor is on, there is a path from VDD to ground
  - Large power dissipation
- Logic high out does not go all the way to VDD



## Solution: CMOS




- No static current flow
- Less current means less power