EECE 2322: Fundamentals of Digital Design and Computer Organization

Lecture 1_2: Gates and Numbers

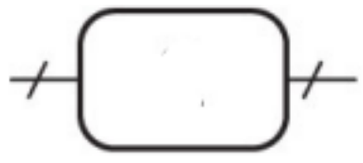
Xiaolin Xu
Department of ECE
Northeastern University

Types of Digital Systems

- No state present
  - Always formulating the instant output
  - Combinational Logic System, Output = Function (Input)
- State present
  - Next State = Function (State, Input)
  - Sequential Logic System
    - Synchronous Sequential System
      - State updates controlled by clock signal
    - Asynchronous Sequential System
      - State updated at any time
- Output = Function (State, Input) – Mealy machine
- Output = Function (State) –Moore machine

Types of Digital Systems

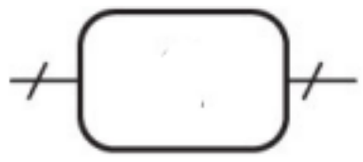
- No state present
  - Always formulating the instant output
  - Combinational Logic System, Output = Function (Input)



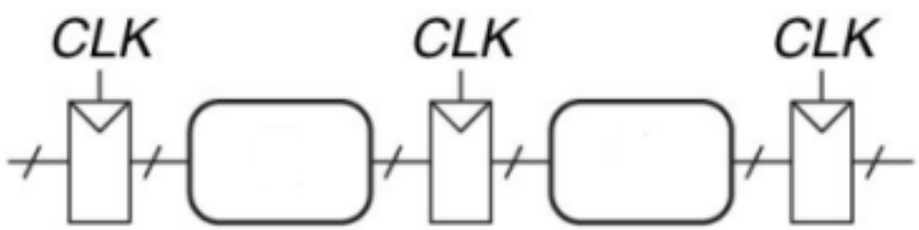
- State present
  - Next State = Function (State, Input)
  - Sequential Logic System
    - Synchronous Sequential System
      - State updates controlled by clock signal
    - Asynchronous Sequential System
      - State updated at any time
  - Output = Function (State, Input) – Mealy machine
  - Output = Function (State) –Moore machine

Types of Digital Systems

- No state present
  - Always formulating the instant output
  - Combinational Logic System, Output = Function (Input)



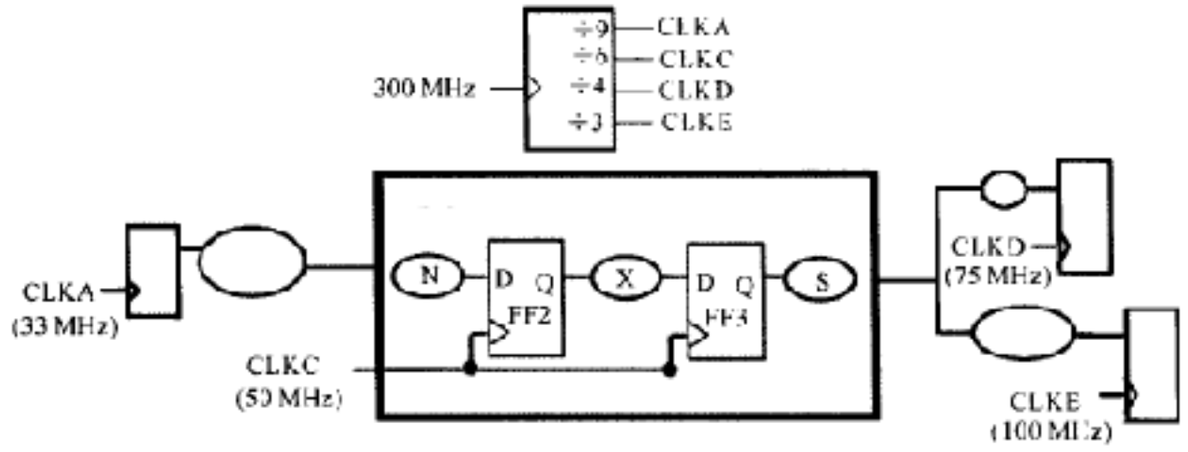
- State present
  - Next State = Function (State, Input)
  - Sequential Logic System
    - Synchronous Sequential System
      - State updates controlled by clock signal
    - Asynchronous Sequential System
      - State updated at any time
- Output = Function (State, Input) – Mealy machine
- Output = Function (State) –Moore machine

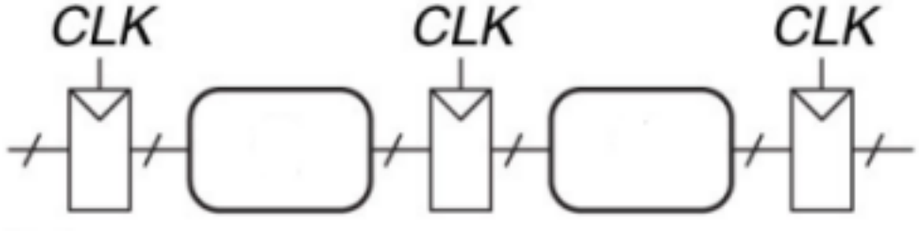


Types of Digital Systems

- No state present
  - Always formulating the instant output
  - Combinational Logic System, Output = Function (Input)

- State present
  - Next State = Function (State, Input)
  - Sequential Logic System
    - Synchronous Sequential System
      - State updates controlled by clock signal
    - Asynchronous Sequential System
      - State updated at any time
  - Output = Function (State, Input) – Mealy machine
  - Output = Function (State) –Moore machine



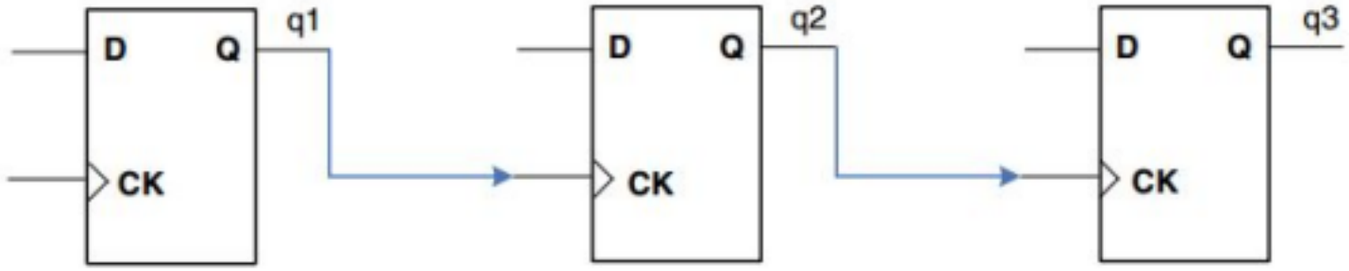


Types of Digital Systems

- No state present
  - Always formulating the instant output
  - Combinational Logic System, Output = Function (Input)
- State present
  - Next State = Function (State, Input)
  - Sequential Logic System
    - Synchronous Sequential System
      - State updates controlled by clock signal
    - Asynchronous Sequential System
      - State updated at any time
- Output = Function (State, Input) – Mealy machine
- Output = Function (State) –Moore machine

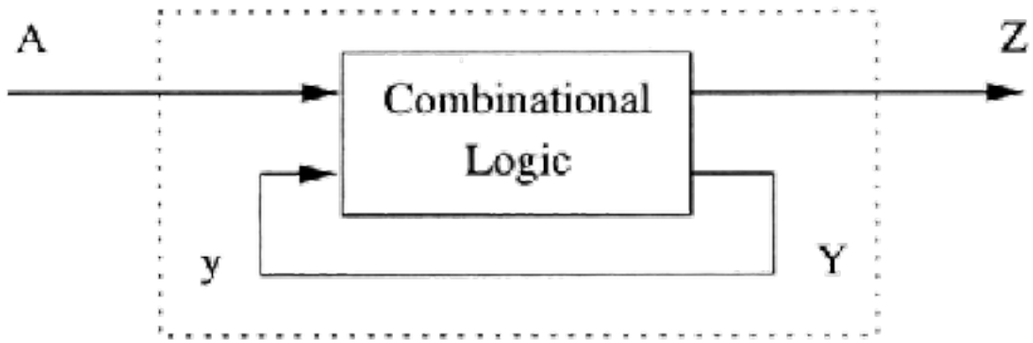
Types of Digital Systems

- No state present
  - Always formulating the instant output
  - Combinational Logic System, Output = Function (Input)
- State present
  - Next State = Function (State, Input)
  - Sequential Logic System
    - Synchronous Sequential System
      - State updates controlled by clock signal
    - Asynchronous Sequential System
      - State updated at any time
- Output = Function (State, Input) – Mealy machine
- Output = Function (State) –Moore machine

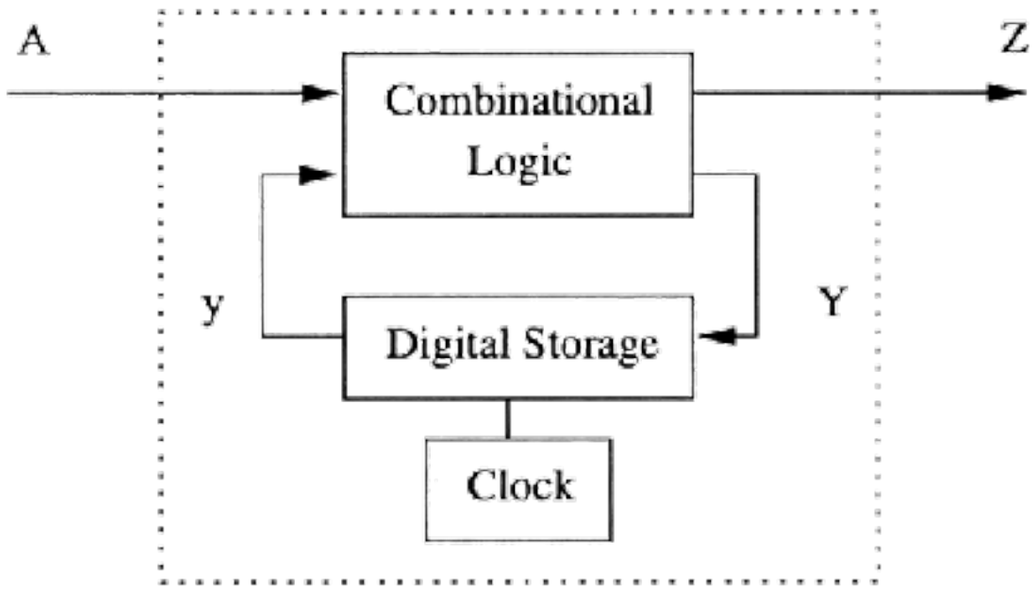


Comparison

Asynchronous

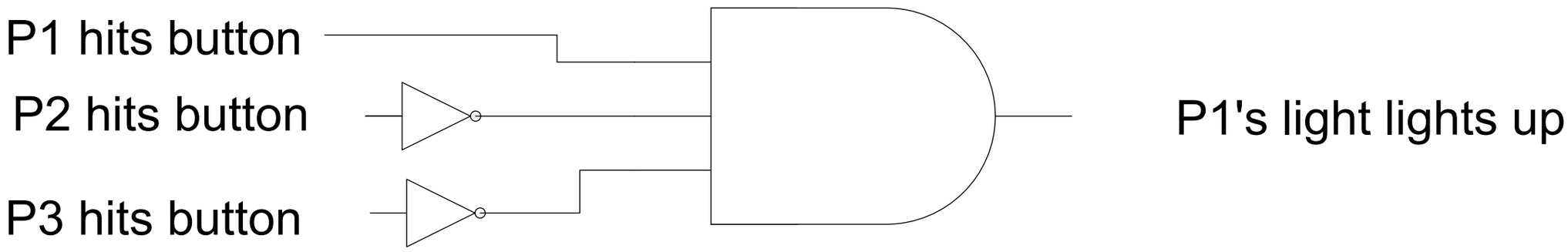


Synchronous



Comparison

- Three people hit their buttons



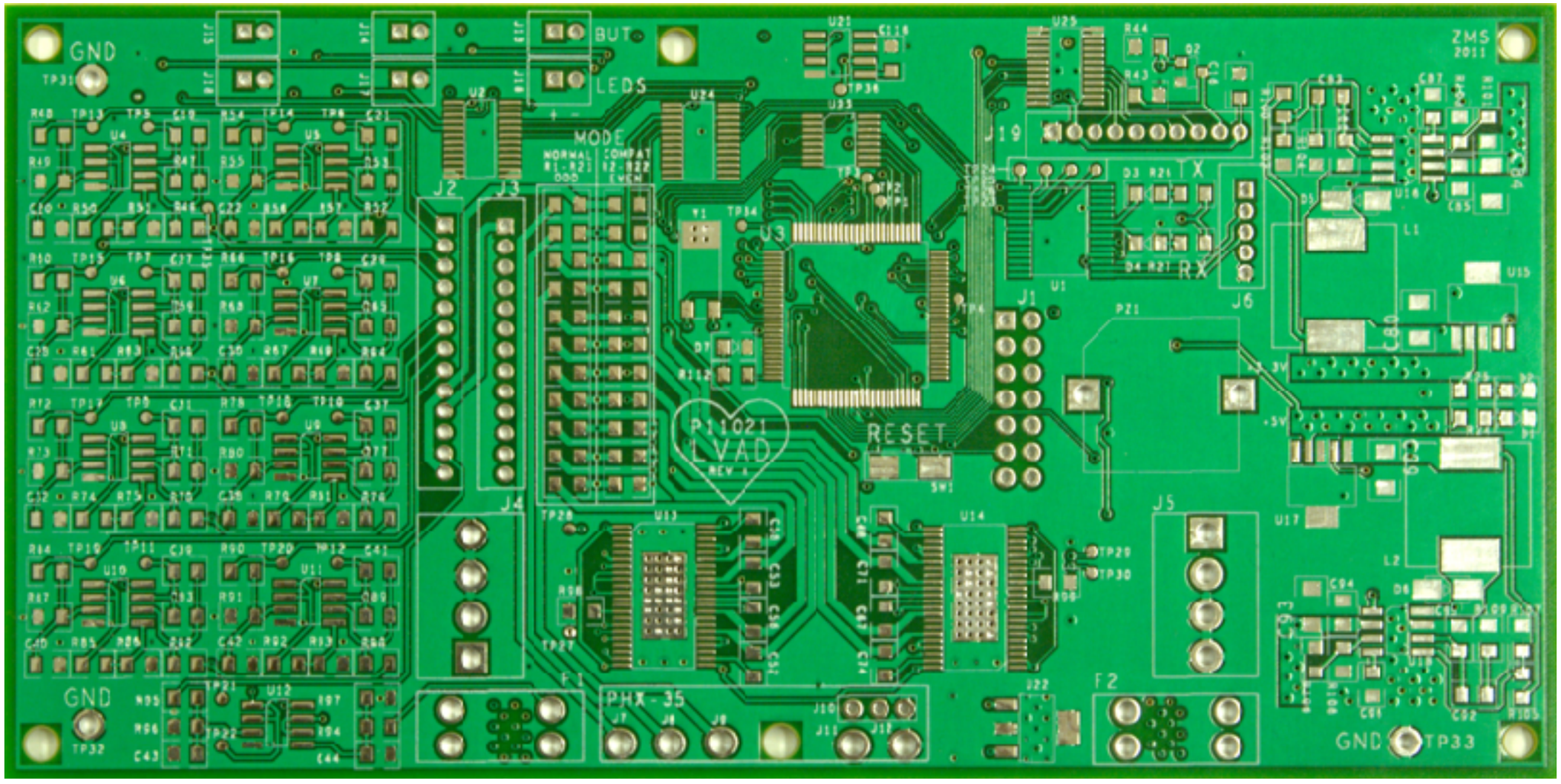
- Why P1 is highlighted as the results?

Module of Digital System

- Low level digital modules
  - Gates - AND, OR, NOR, etc.
  - Blocks - Adder, subtractor, shifter, etc.
- High level digital modules
  - PLDs (Programmable Logic Device)
  - ASICs (Application Specific Integrated Circuits)
  - Microprocessors/Microcontrollers

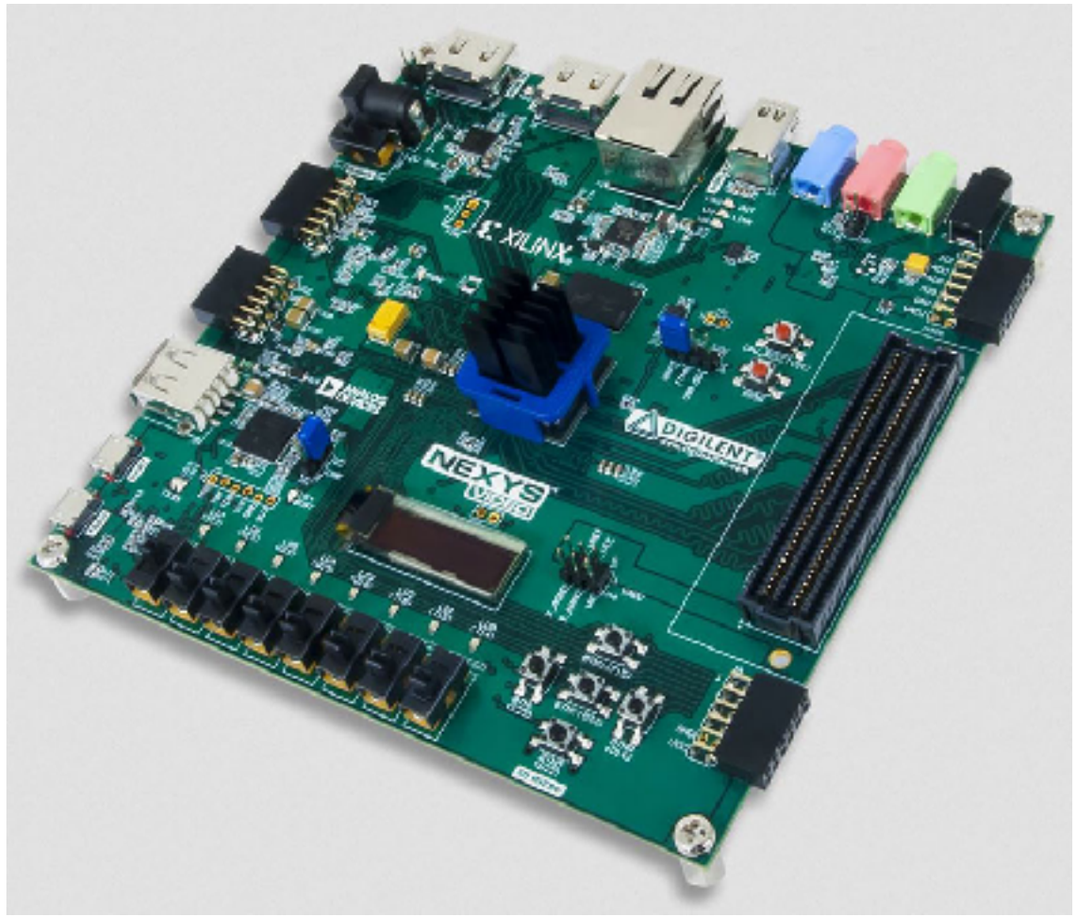
Printed Circuit Board

- PCB



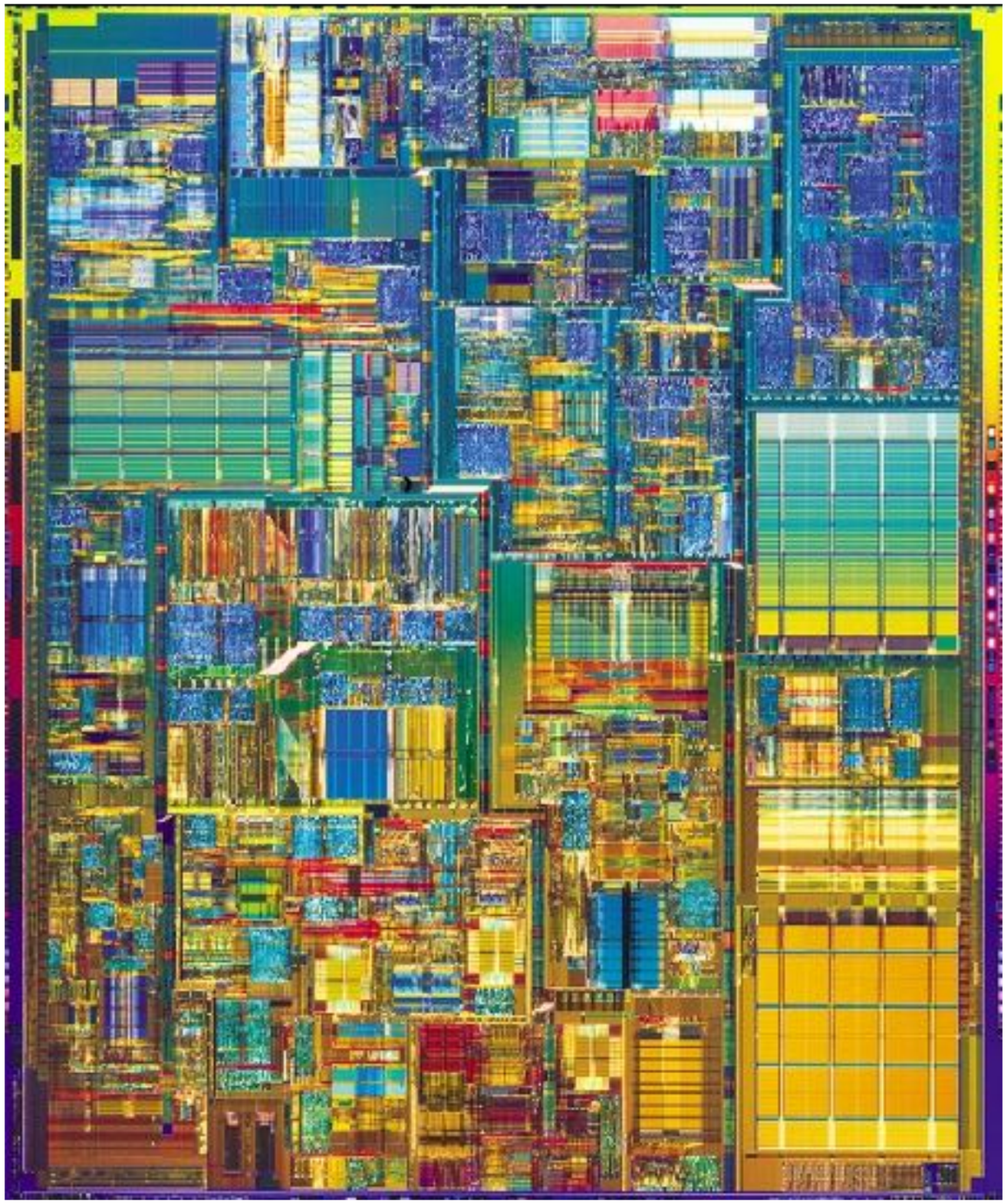
Field Programmable Gate Array

- FPGA



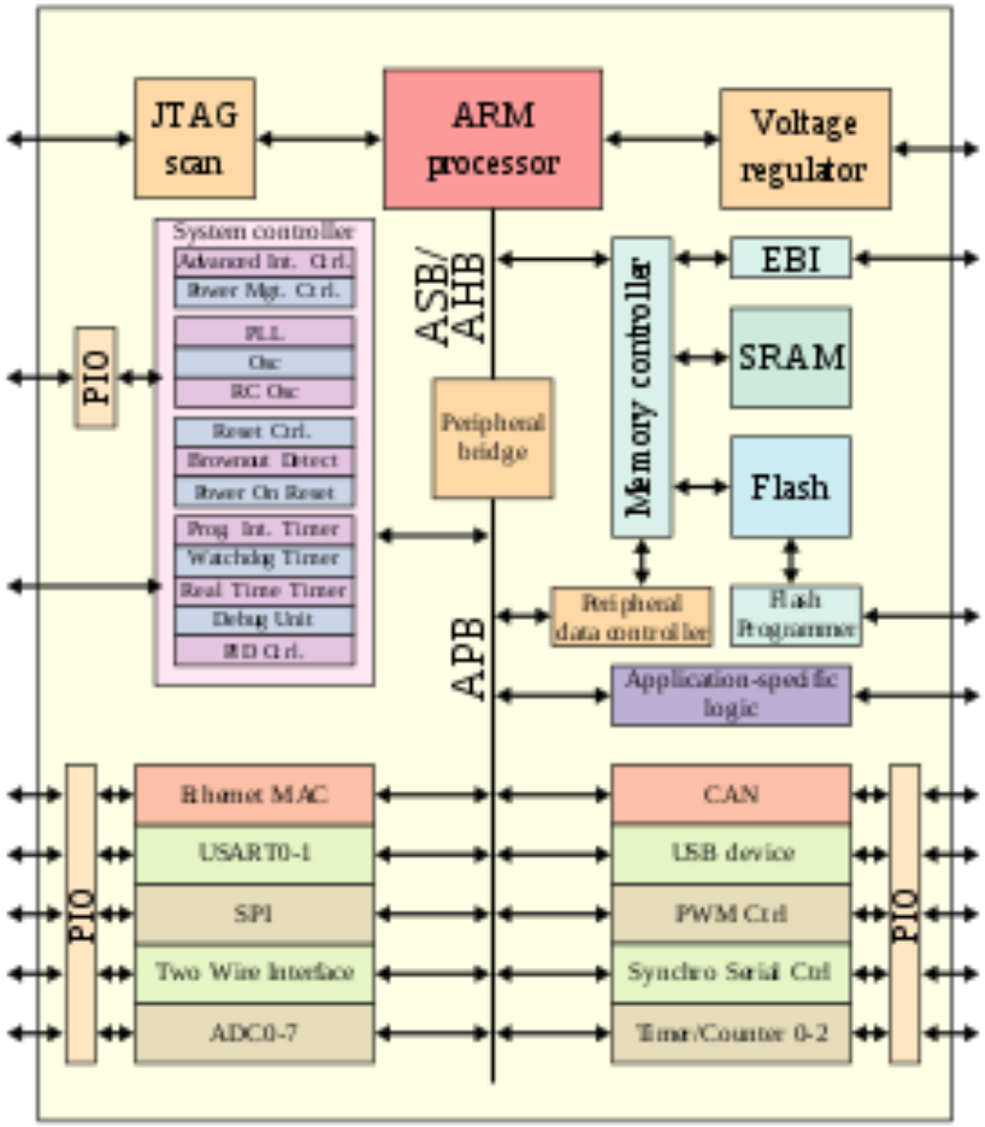
Very Large Scale Integrated Circuit

- VLSI
  - Intel Pentium IV



System on Chip

- SoC



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**

Digital Systems Design Flow

- 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)

Function Specification

- Full adder
  - Input: A, B, Carry_in
  - Output: Carry_out, Sum

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

Behavior Specification

- VHDL
  - 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;

Behavior Specification

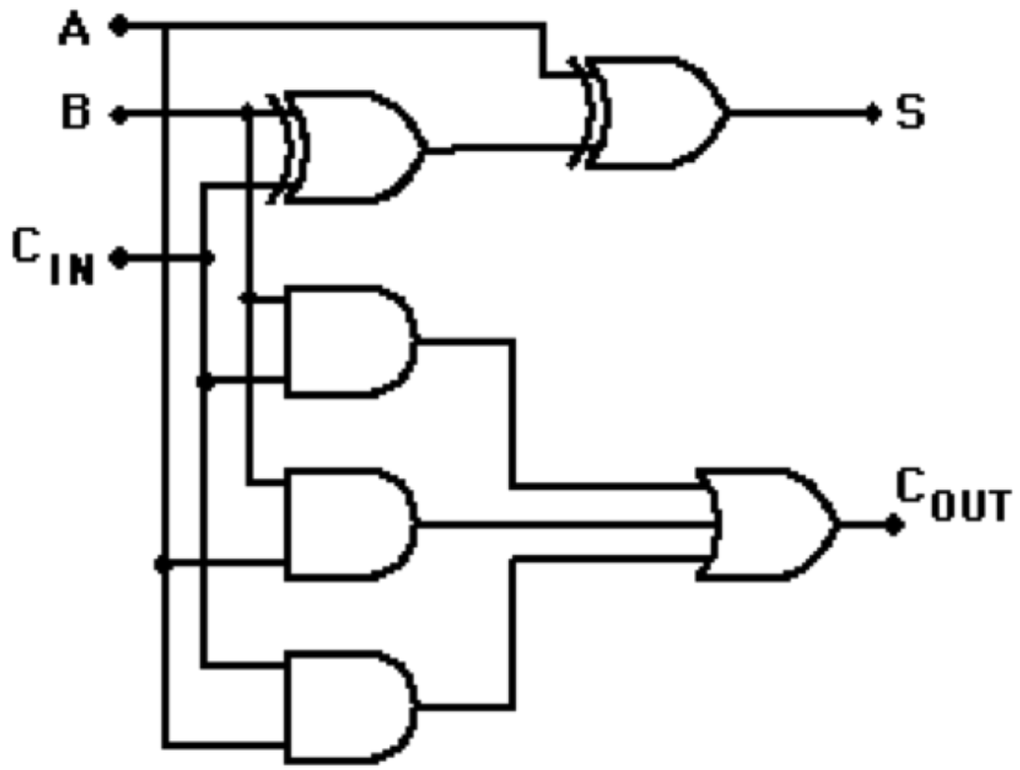
- Verilog
  - module fulladder (a, b, cin, sum, cout);
  - input a, b, cin;
  - output sum, cout;
  - reg sum, cout;
  - always @ (a or b or cin)
  - begin
  - sum <= a ^ b ^ cin;
  - cout <= (a & b) | (a & cin) | (b & cin);
  - end
  - endmodule

Logic Design

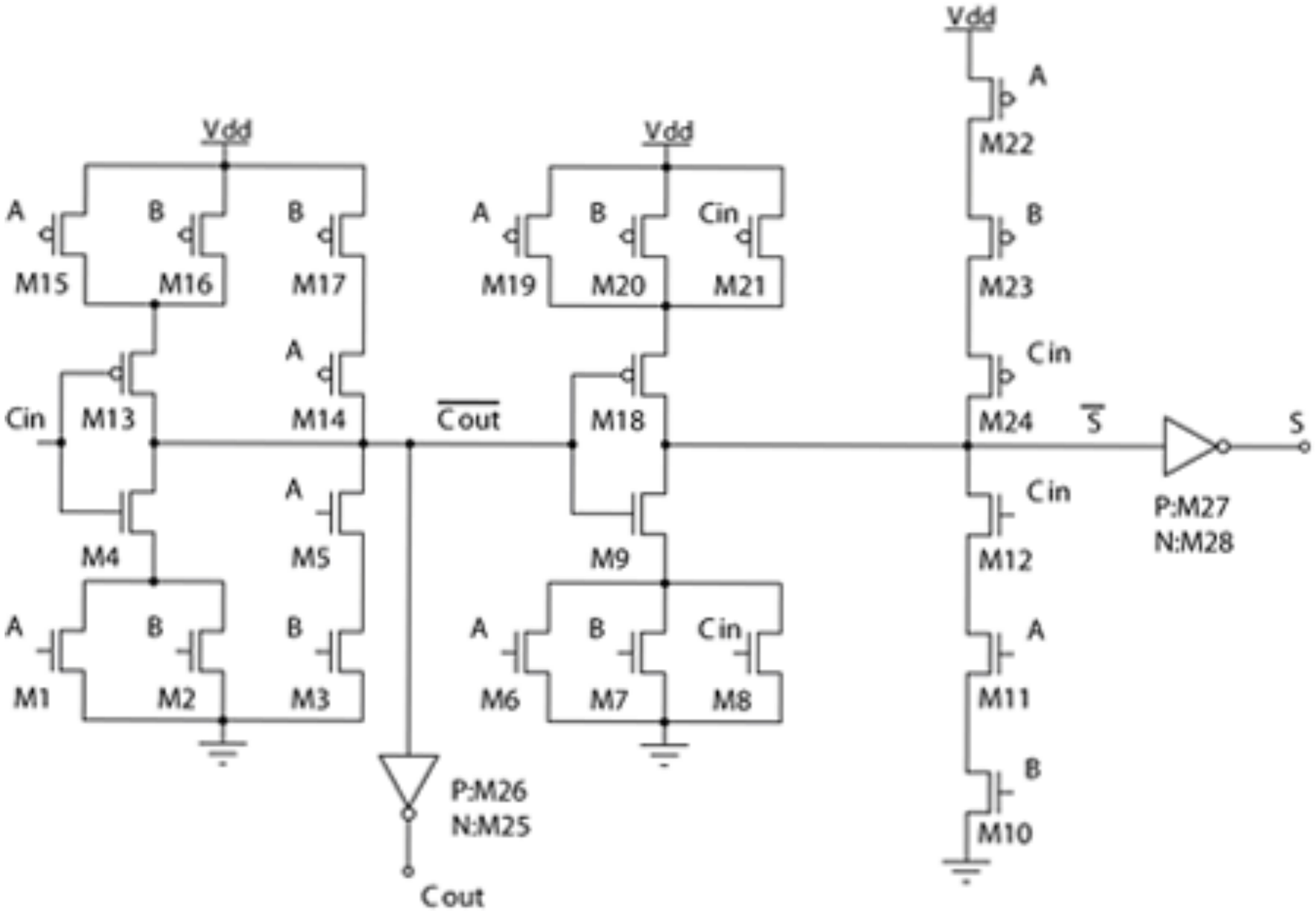
- More bits are chained with carry in

Full Adder Truth Table

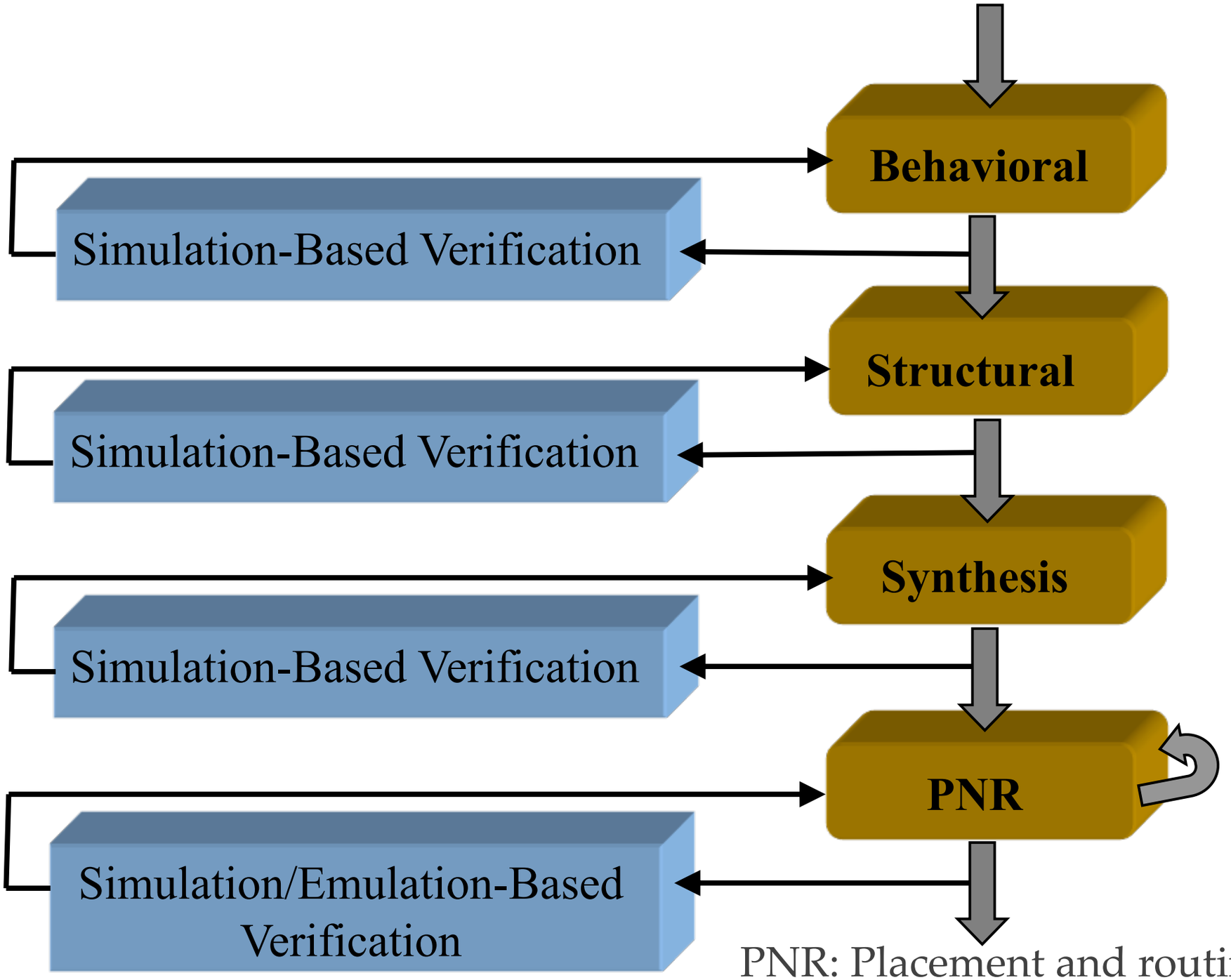
CARRY IN	*input B*	*input A*	*CARRY OUT*	*SUM digit*
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Transistor Schematic

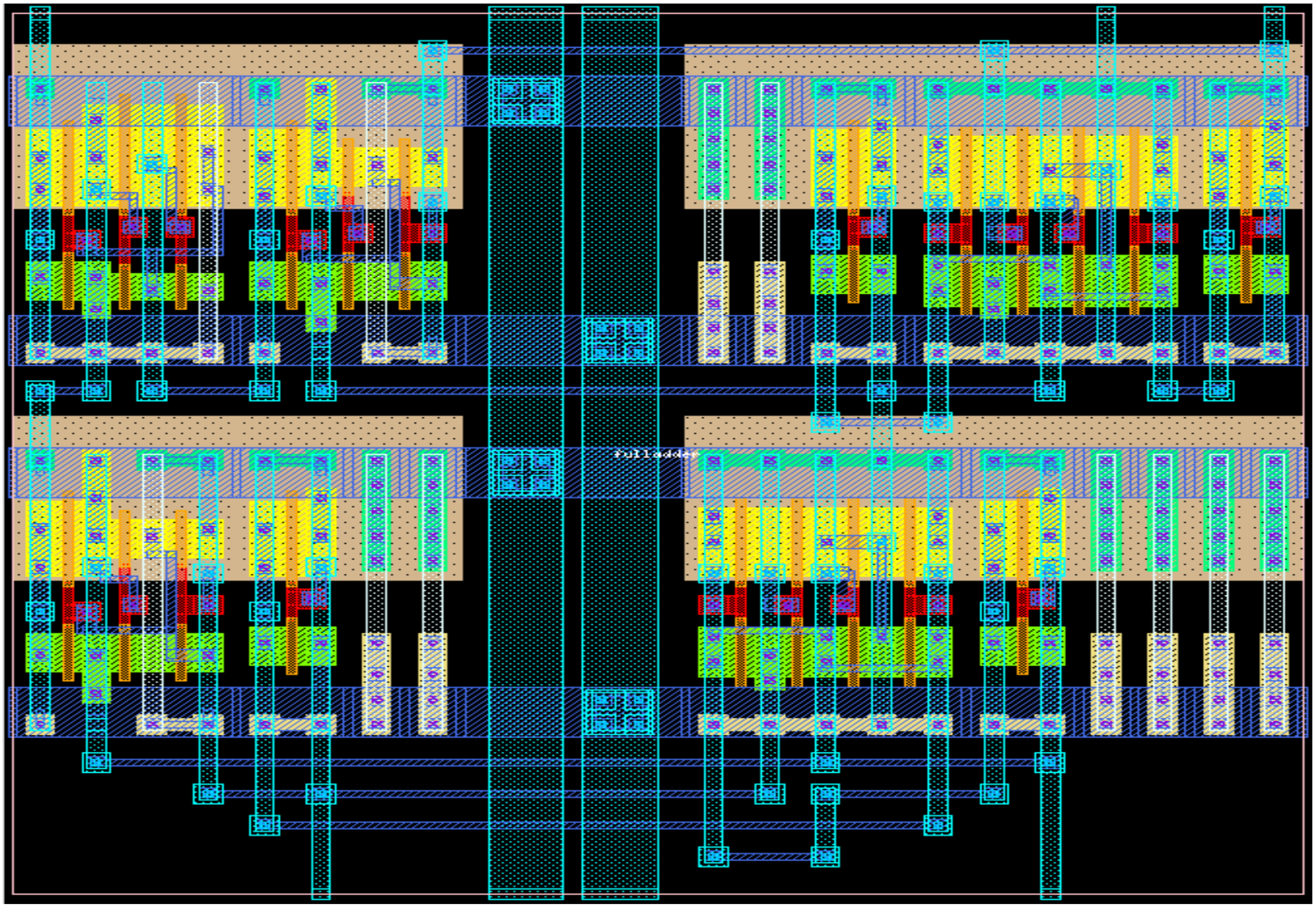


Practical Design Process in Iterative



PNR: Placement and routing

Layout



More about Hardware Description Languages

- Two primary choices
  - VHDL (VHSIC (Very high-speed IC) hardware description language)
  - Verilog HDL
- Can be used for behavioral specification, architectural definition, implementation, and verification

More about Hardware Description Languages

- High-level partitioning of problem into functional blocks
  - Can be expressed in a variety of forms - text, graphically, formal languages
- Must choose between different possible architectures and weigh the costs and benefits of each choice
  - Architecture design is often a matter of balancing tradeoffs

More about Hardware Description Languages

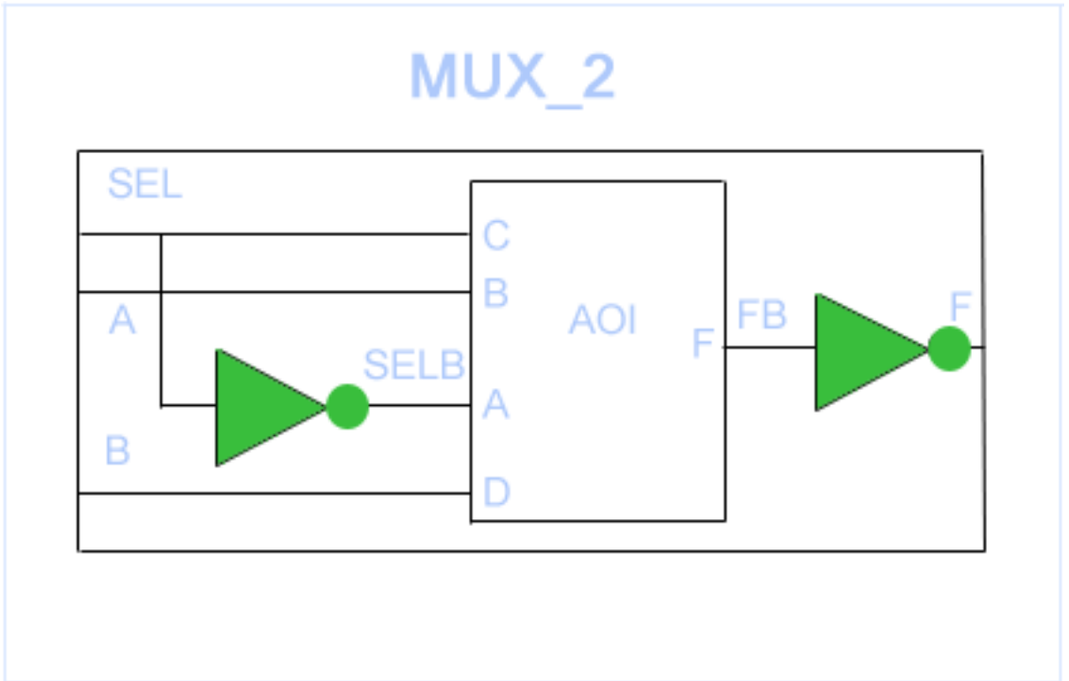
- Elaboration: Before simulation begins, the design hierarchy is first elaborated. This means all the pieces of the model code (entities, architecture and configurations) are put together.
- Initialization: The nets in the model are initialized just before simulation starts.
- Simulation cycle:
  - Simulation cycle is then continuously repeated during which processes are executed and signals are updated.
- Advantage: **Top-Down design methodology, Technology independent**

More about Hardware Description Languages

- // Verilog code for 2-input multiplexer
- module INV (input A, output F); // An inverter
- assign F = ~A;
- endmodule

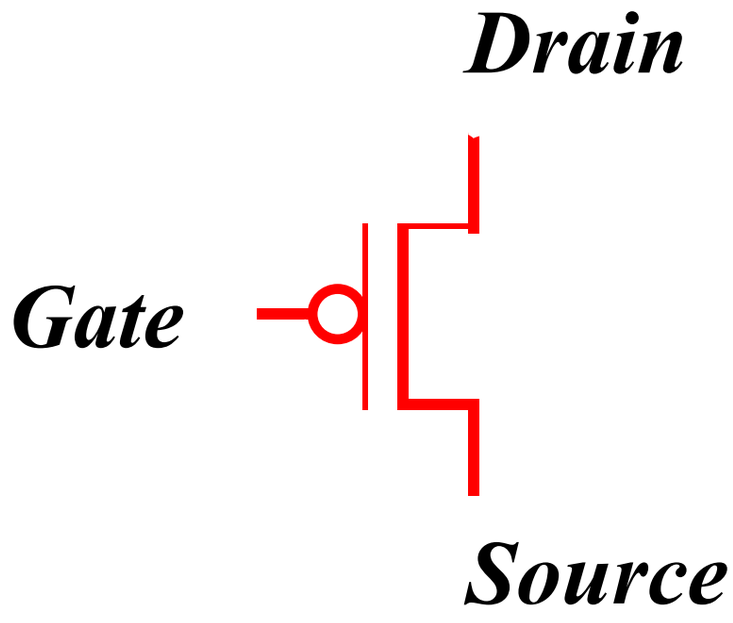
- module AOI (input A, B, C, D, output F);
- assign F = ~((A & B) | (C & D));
- endmodule

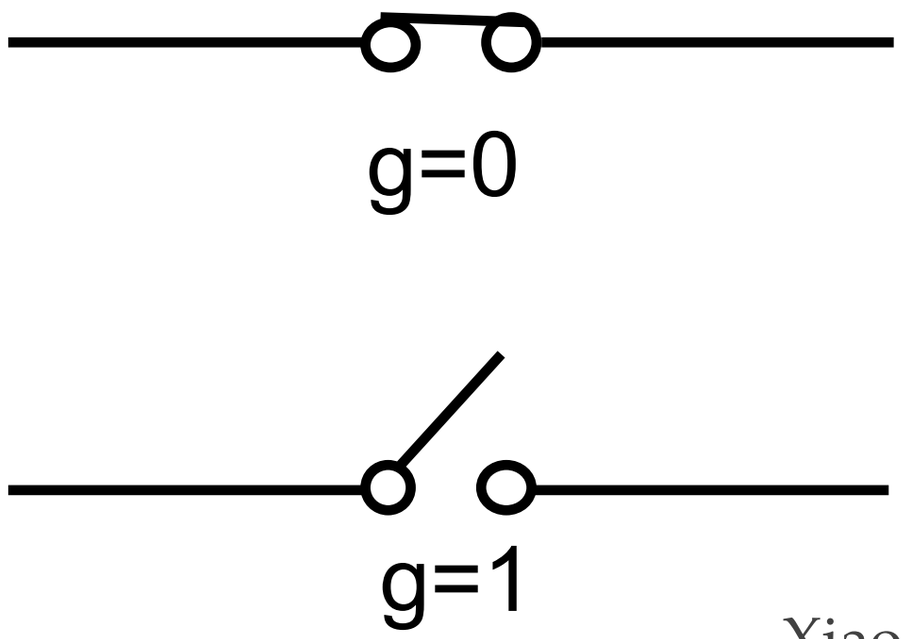
- module MUX2 (input SEL, A, B, output F); // 2:1 multiplexer
- // wires SELB and FB are implicit
- // Module instances...
- INV G1 (SEL, SELB);
- AOI G2 (SELB, A, SEL, B, FB);
- INV G3 (.A(FB), .F(F)); // Named mapping
- endmodule
- // end of Verilog code



PMOS Transistor

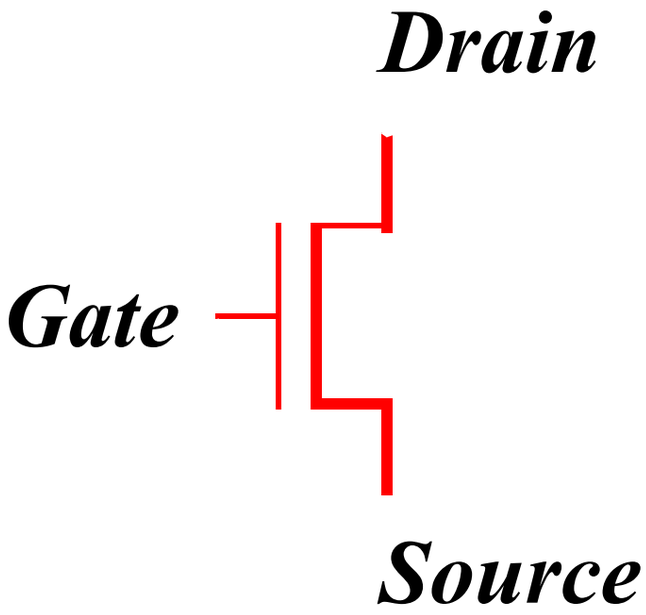
- Basic rules of PMOS
  - If the gate is “low”, the switch is on
  - If the gate is “high”, the switch is off

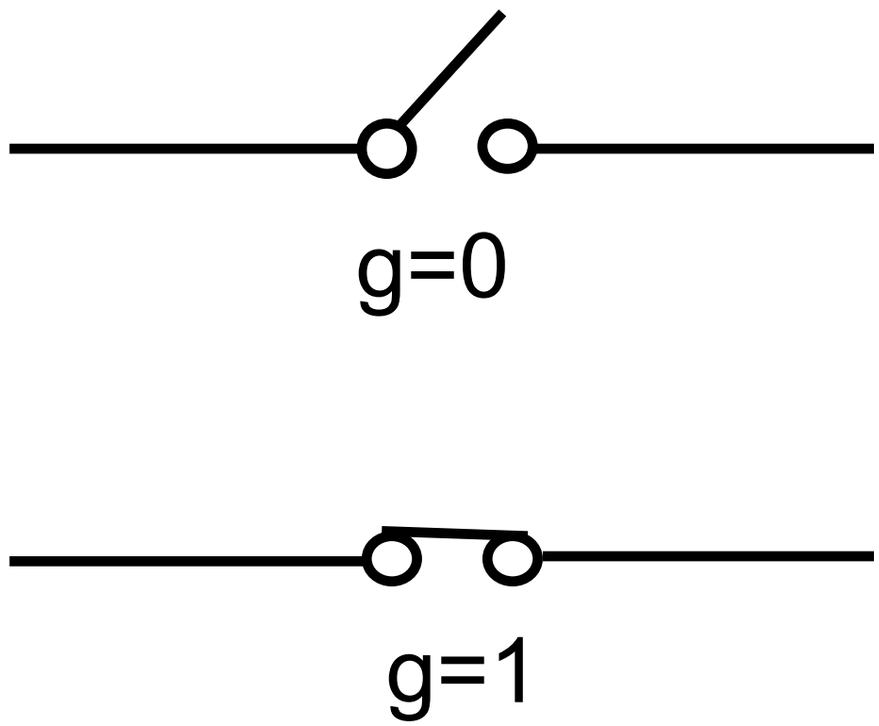




NMOS Transistor

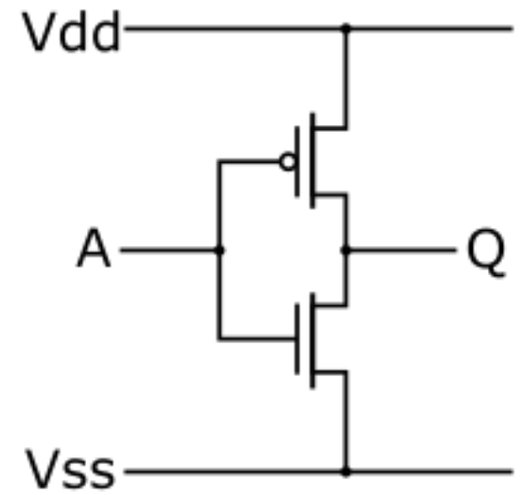
- Basic rules of NMOS
  - If the gate is “high”, the switch is on
  - If the gate is “low”, the switch is off

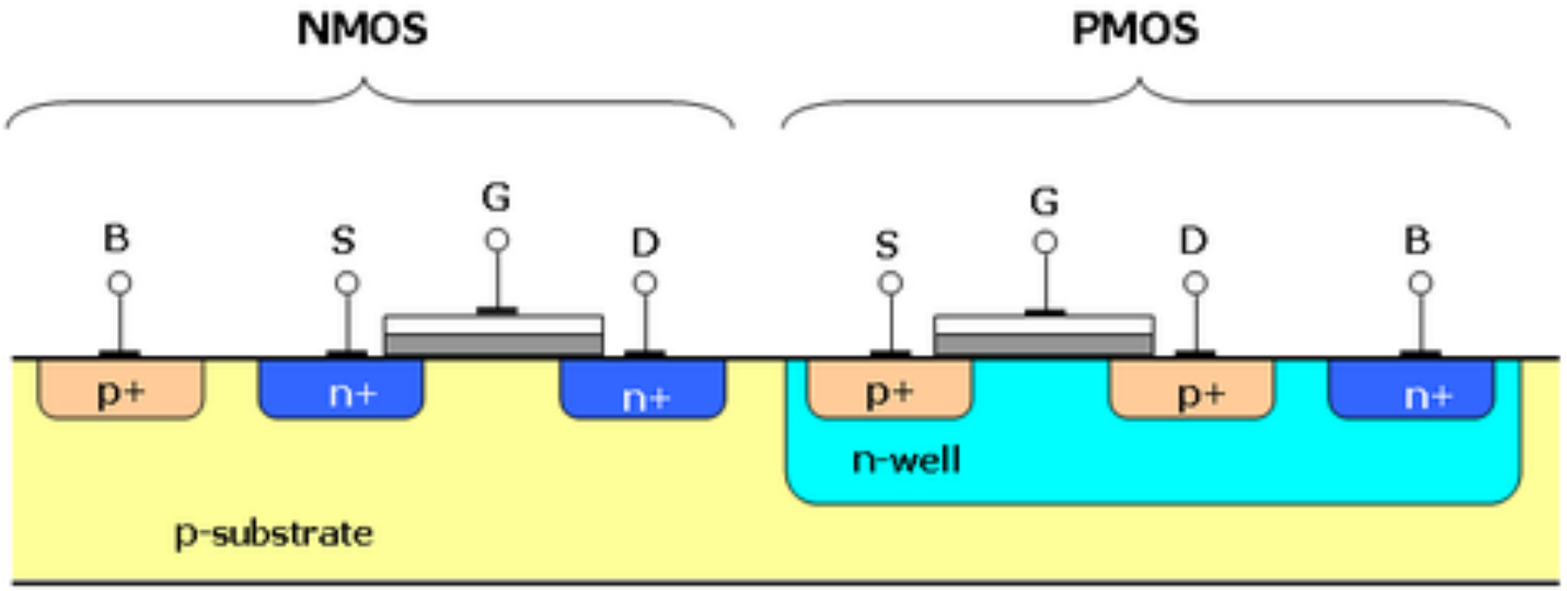


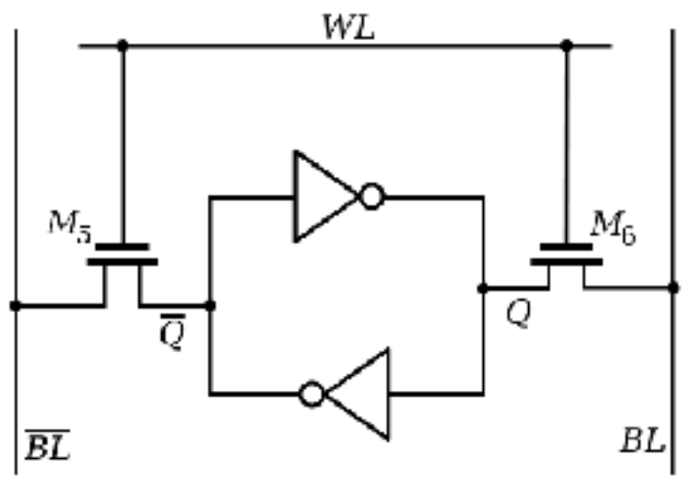


CMOS Circuit

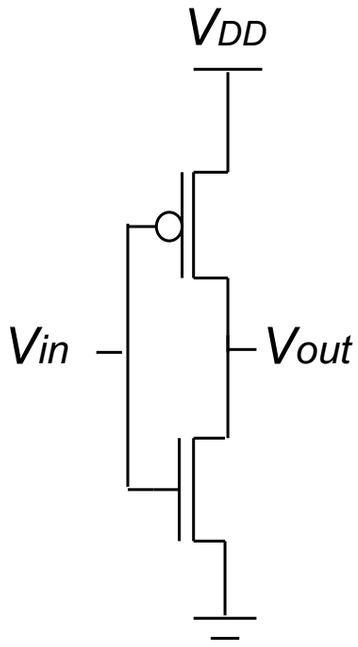
- Extremely low static current flow
- Less current means less power

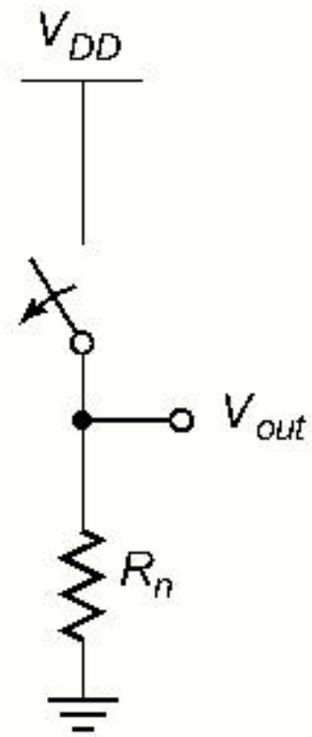


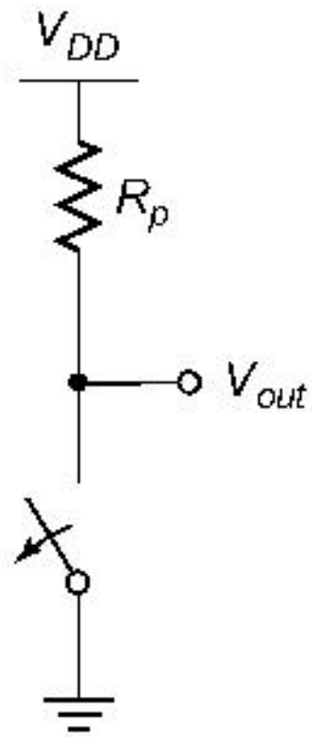




CMOS Inverter: First-order DC Analysis



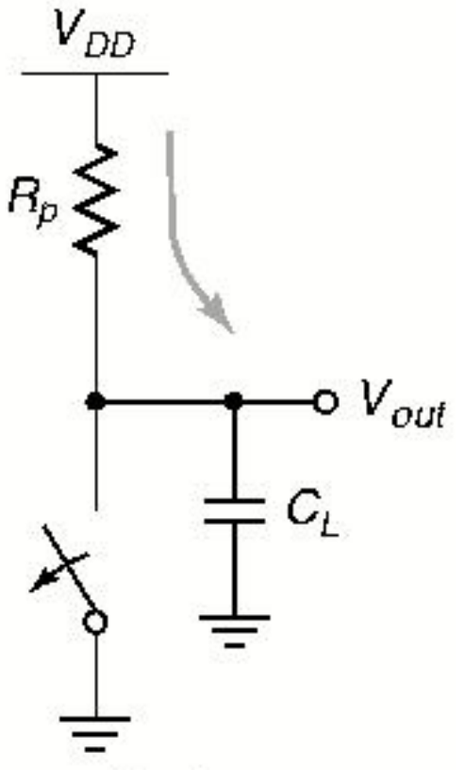




$V_{OL} = 0$
$V_{OH} = V_{DD}$

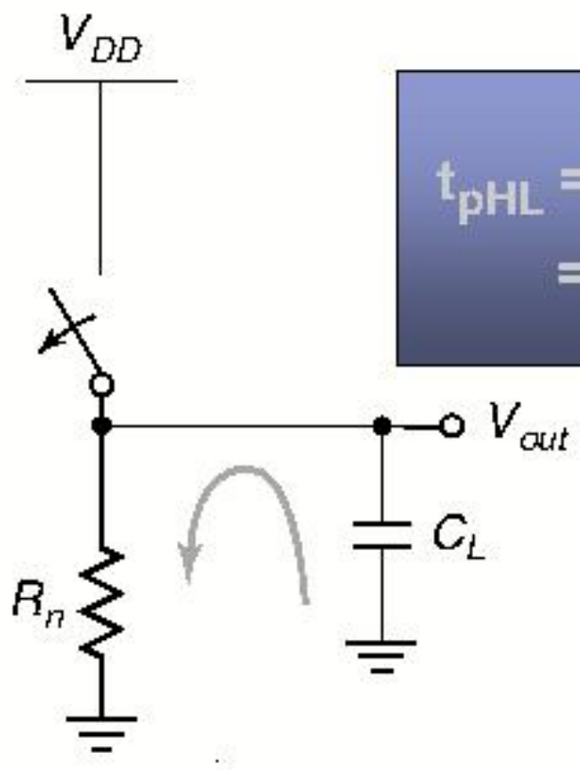
- High noise margin
- Ratioless
- low output impedance
- Extremely high input impedance
- No static power

CMOS Inverter: Transient Response



V_{in} = V_{DD} →0

Output: Low-to-High



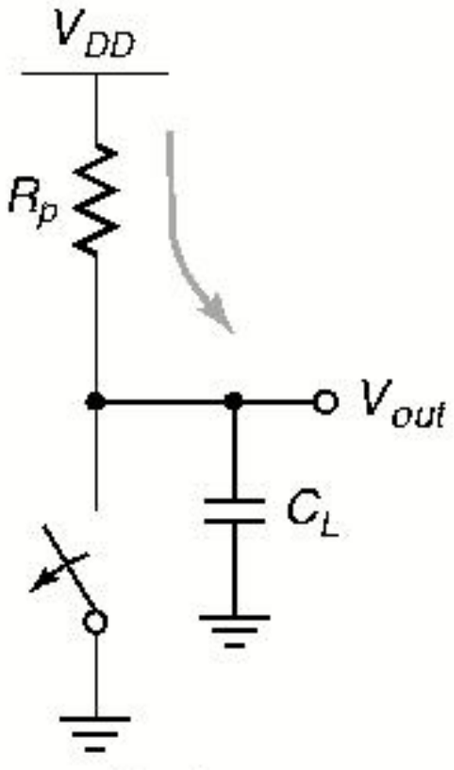
V_{in} = 0 → V_{DD}

High-to-Low

$t_{pHL} = f(R_{on}.C_L)$
$= 0.69\ R_{on}C_L$

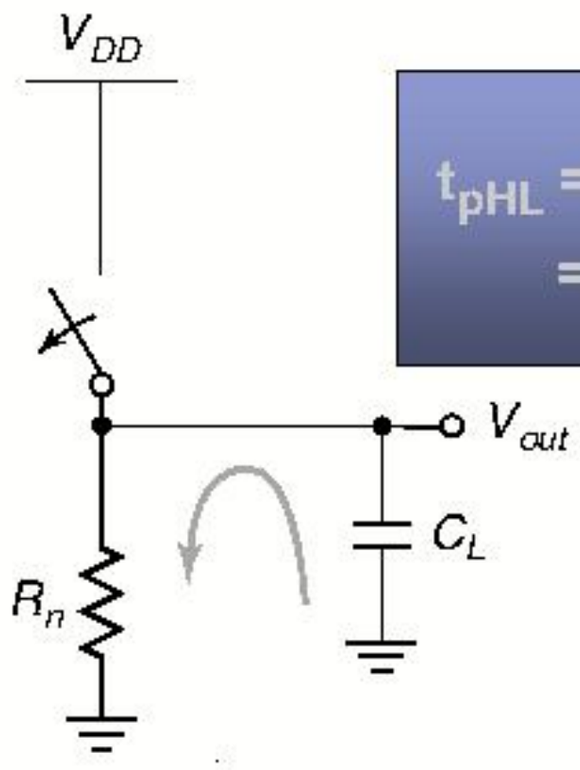
- CL is composed of the drain diffusion capacitances of the NMOS and PMOS transistors, the capacitance of connecting wires, and the input capacitance of the fan-out gates

CMOS Inverter: Transient Response



V_{in} = V_{DD} →0

Output: Low-to-High



V_{in} = 0 → V_{DD}

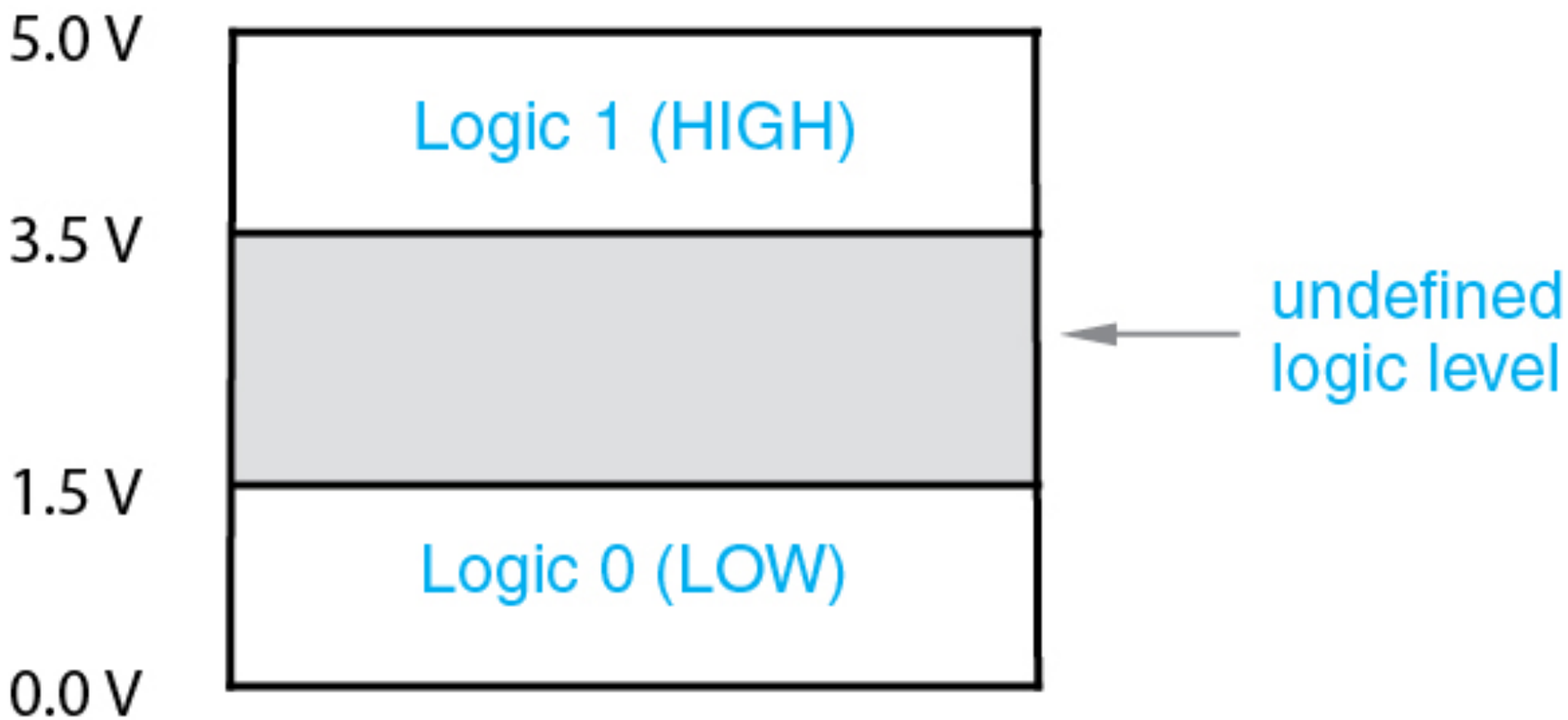
High-to-Low

$t_{pHL} = f(R_{on}.C_L)$
$= 0.69\ R_{on}C_L$

To reduce delay:
Reduce CL
Reduce Rp,n
Increase W/L ratio

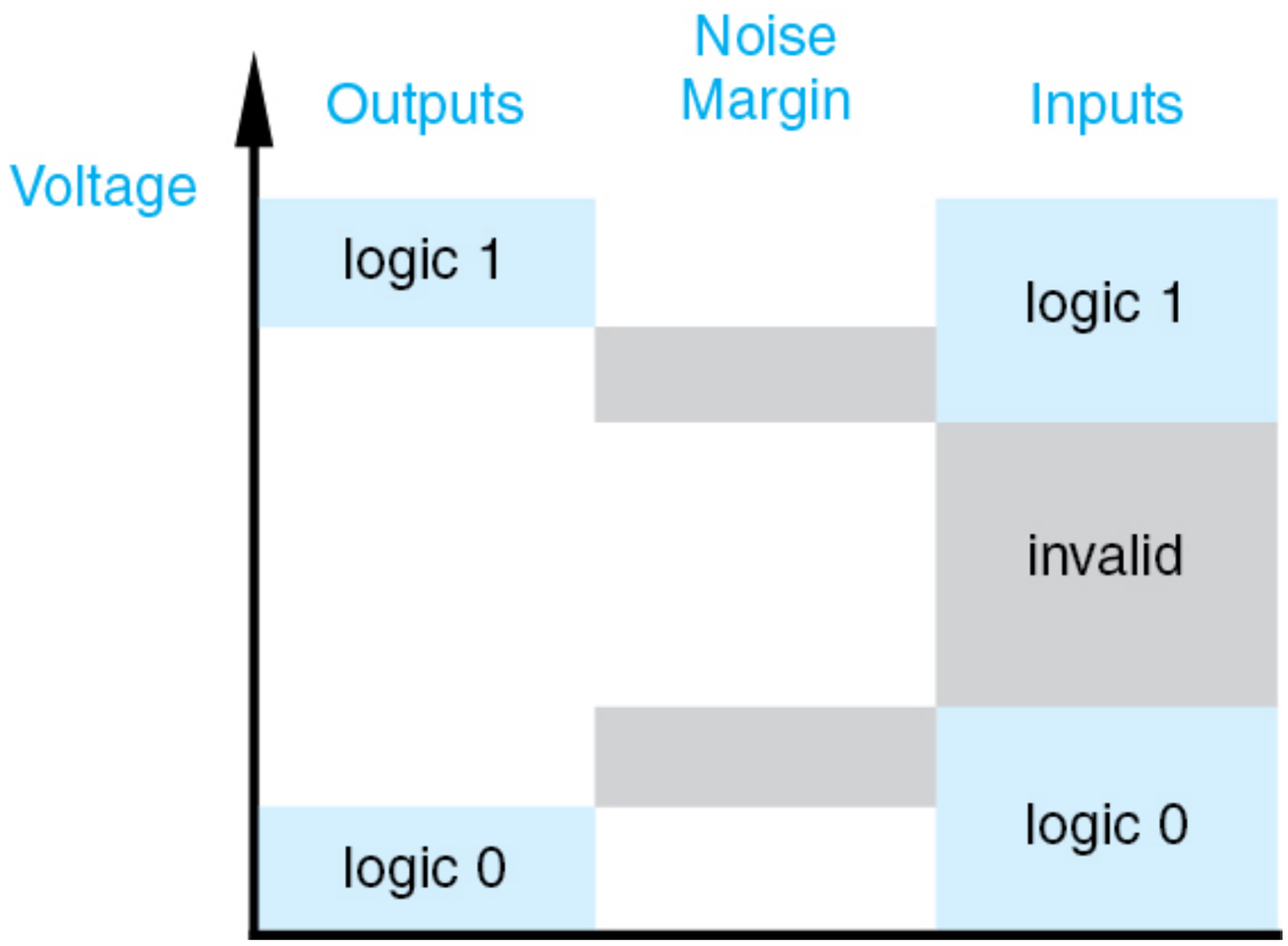
- CL is composed of the drain diffusion capacitances of the NMOS and PMOS transistors, the capacitance of connecting wires, and the input capacitance of the fan-out gates

Digital and Analog



Digital and Analog

- Difference between inputs and outputs, why? Benefits?



Basic Operator

- Boolean Algebra
  - Operators

AND	•	^	
OR	+	v	
NOT	$\bar{A}$	¬A	A’

Values

1 (true) 0 (false)

Boolean Algebra Rules

Rules hold for any Boolean value

a,b, c ... stand for these values

Identity element

$a + 0 = a$

$a \bullet \_\_ = a$

+ and • are commutative:

$a + b = b + a$

$a \bullet b = b \_\_\_$

Associative and Distributive

Associative rules:

$a + (b + c) = (a + b) + c$

$a \bullet (b \bullet c) =$ _____

Distributive rules:

$a + (b \bullet c) = (a + b) \bullet (a + c)$

$a \bullet (b + c) =$ _____

Note: these rules look like algebra !