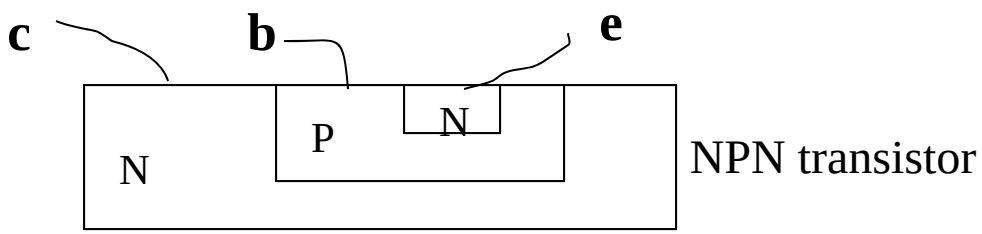
Fundamentals of Digital IC Design

Haibo Wang
ECE Department
Southern Illinois University
Carbondale, IL 62901

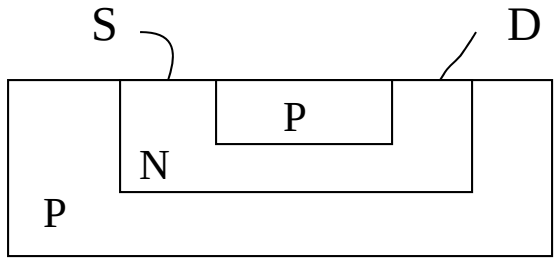
Semiconductor Devices

- Bipolar Transistors:



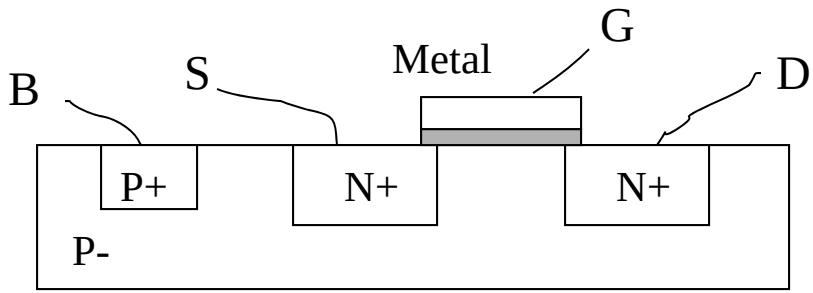
NPN transistor

- Junction Field Effective Transistor (JFET):



N-channel FET

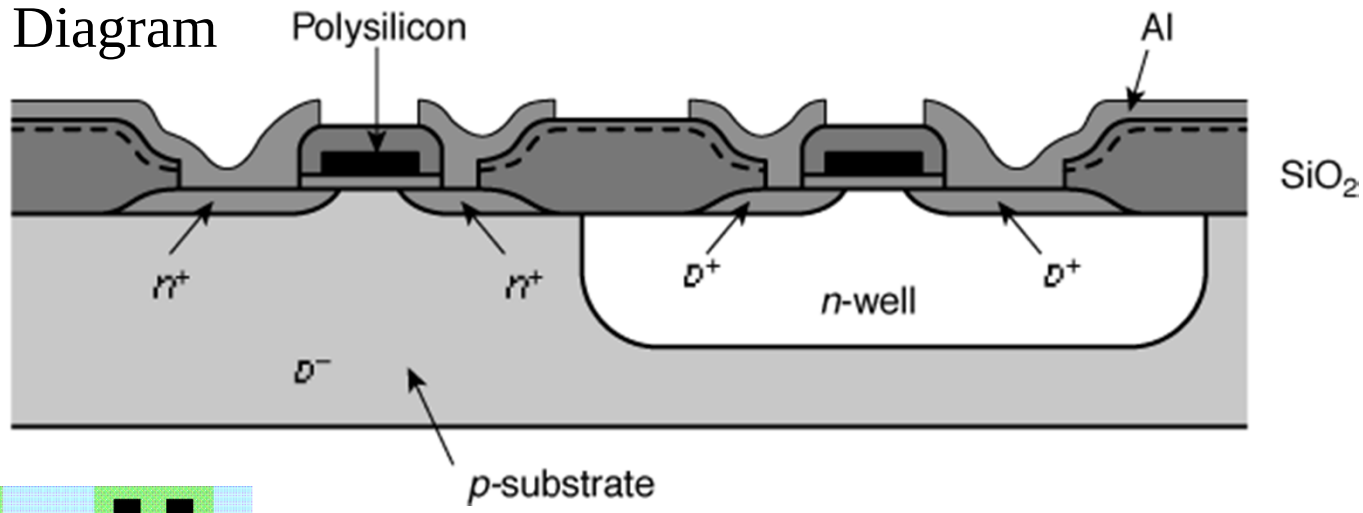
- Metal-Oxide-Semiconductor (MOS) Devices:



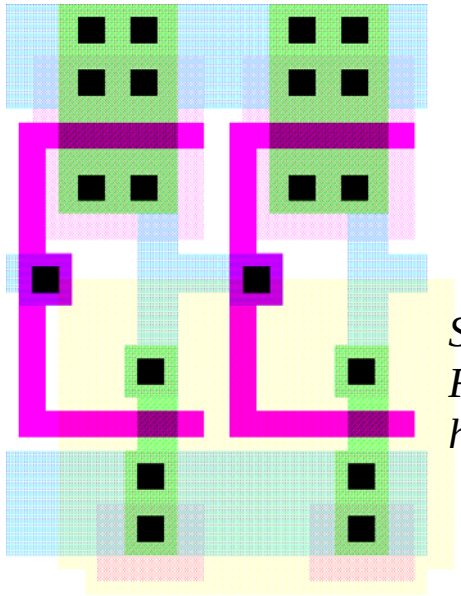
NMOS Device

CMOS Process

- Cross-Section Diagram



- Layout



Source from http://infopad.eecs.berkeley.edu/~icdesign
For a great tour of IC fabrication process, see
http://www.fullman.com/semiconductors/semiconductors.html

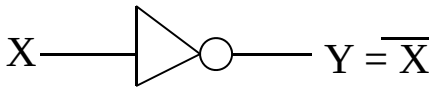
Digital Signals and Basic Logic Gates

- Digital signal values

True	1	High Voltage (e.g. 5V)[12]
False	0	Low Voltage (e.g. 0V)

- Basic logic gates

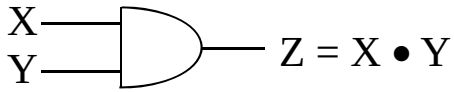
1. Inverter



X	Y
1	0
0	1

Truth Table of an inverter

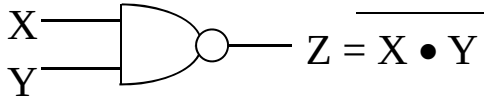
2. AND



X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

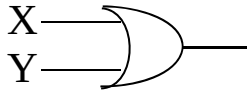
Truth Table of an AND gate

3. NAND



Basic Logic Gates

4. OR

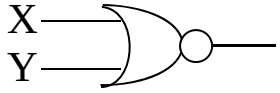


$Z = X + Y$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

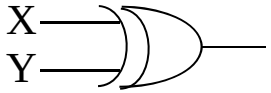
Truth Table of an OR gate

5. NOR



$Z = \overline{X + Y}$

4. XOR



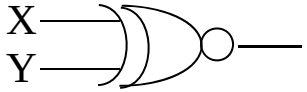
$Z = X\bullet\overline{Y} + \overline{X}\bullet Y$

$Z = X \oplus Y$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table of an XOR gate

5. XNOR

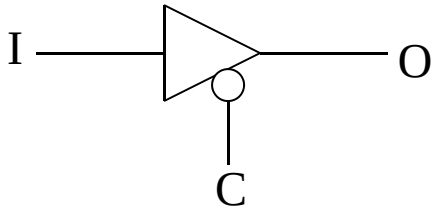


$Z = \overline{X\bullet\overline{Y} + \overline{X}\bullet Y}$

$Z = \overline{X \oplus Y}$

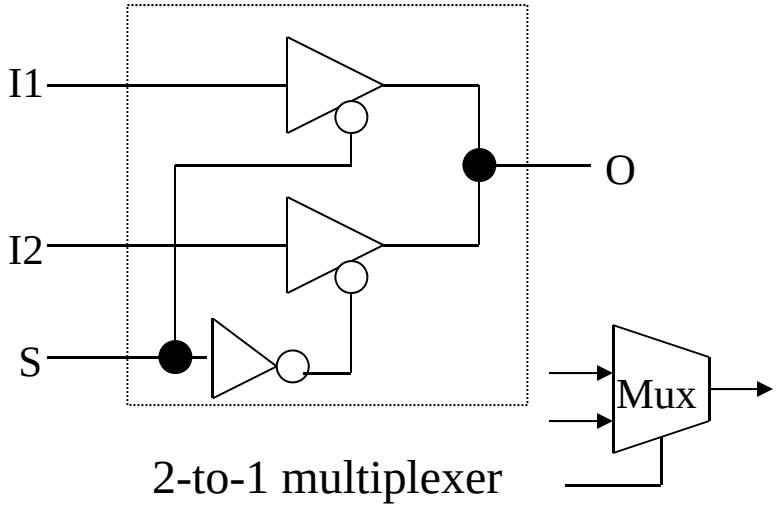
Basic Logic Gates

- Tri-state Buffer

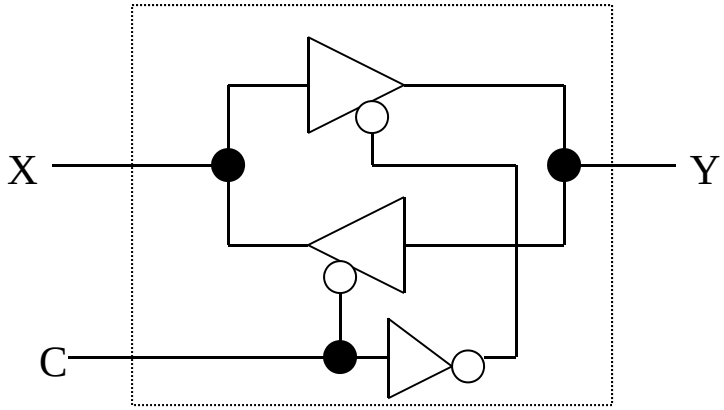


C	I	O
1	X	Z
0	0	0
0	1	1

Truth table of a tri-state buffer

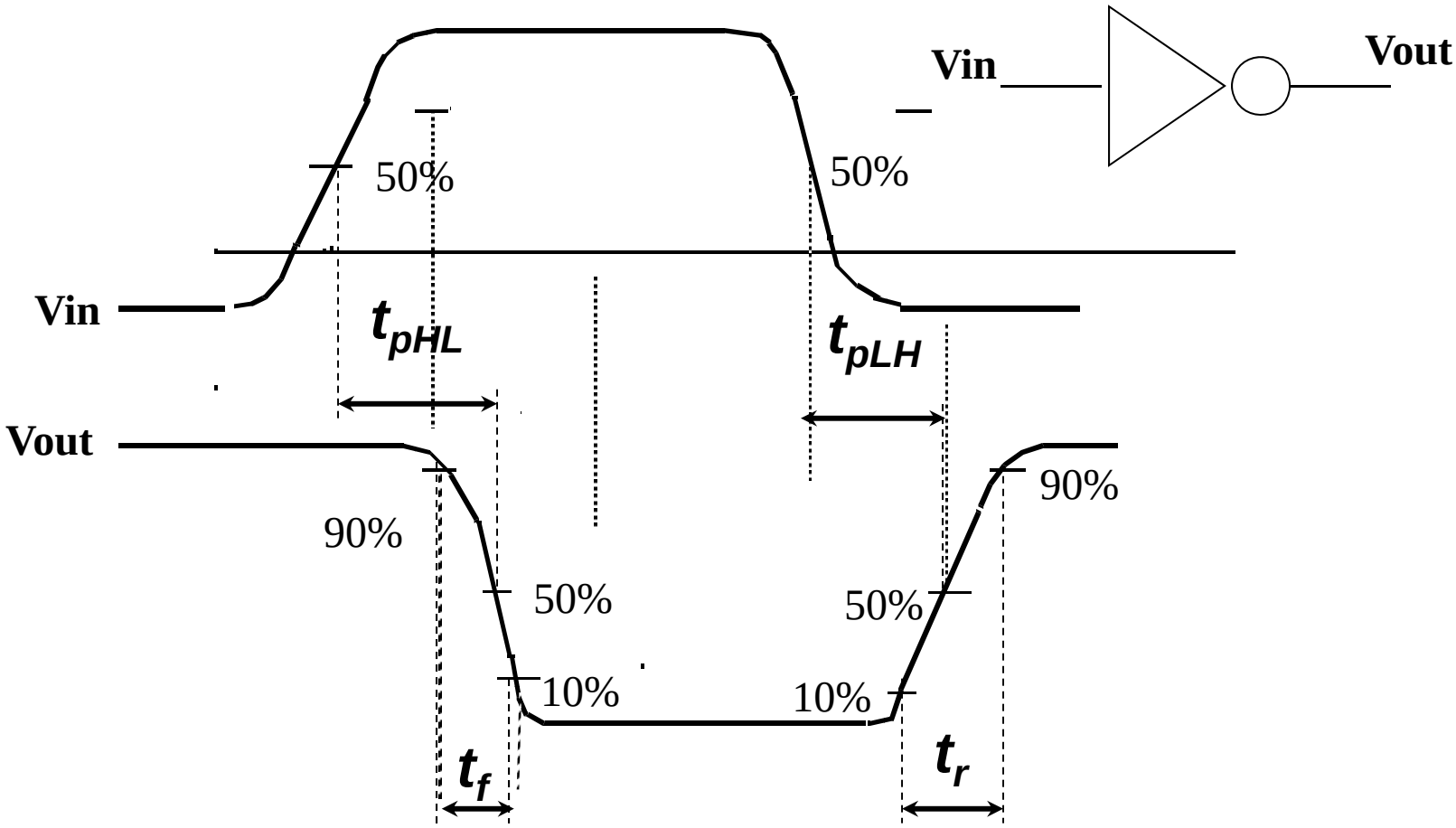


2-to-1 multiplexer



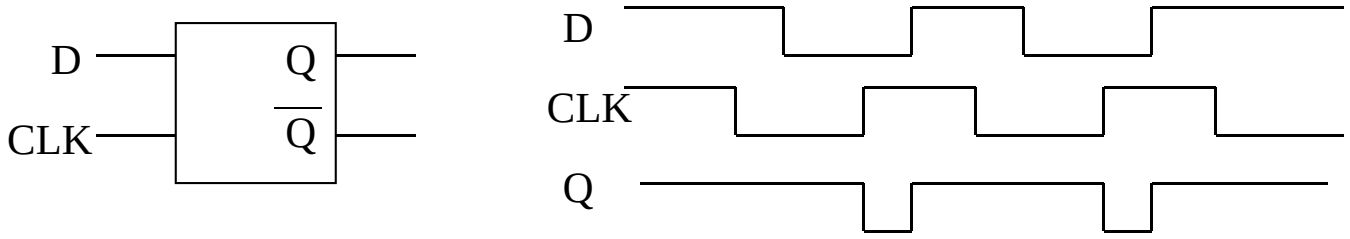
Bi-direction circuit

Gate Delay Definition



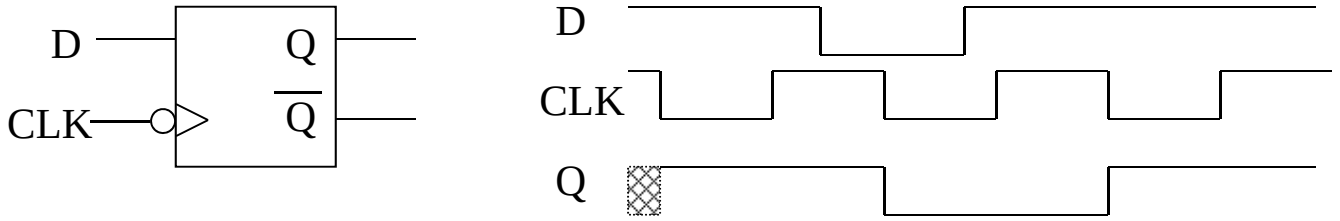
Sequential Logic Circuits

❑ D latch



- When CLK=1, Q always reflects the signal value at input D
- If CLK=0, Q stores the last D value which appears at D just before CLK falls to 0

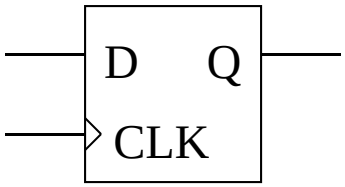
❑ D Flip-flop



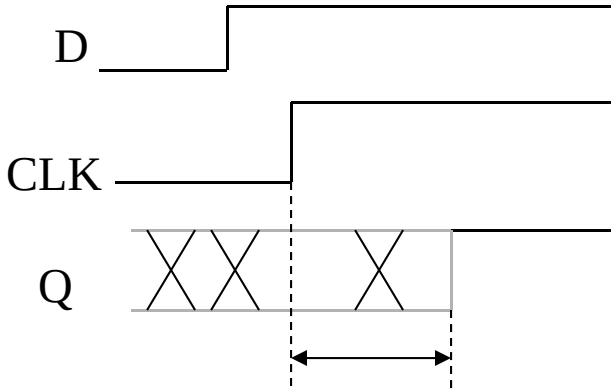
- D flip-flop will not change its output values unless there is a negative edge event at CLK input (CLK switches from logic 1 to 0).
- When a negative edge appears at CLK input , D Flip-flop updates Q to the current D value

Timing Parameters for D Flip-Flops

- Clock-Q Delay

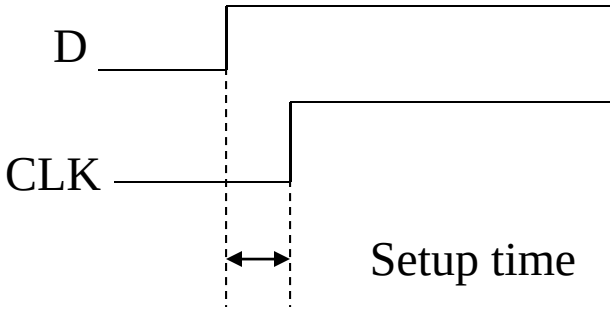


DFF

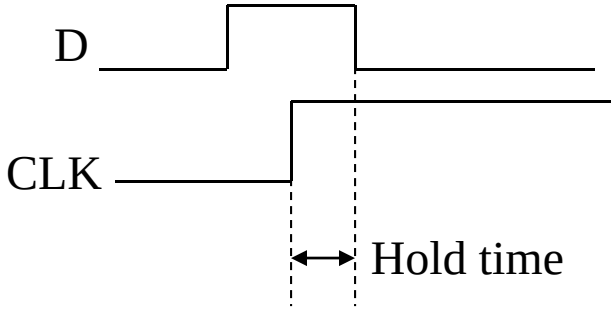


Clock-Q delay

- Setup Time

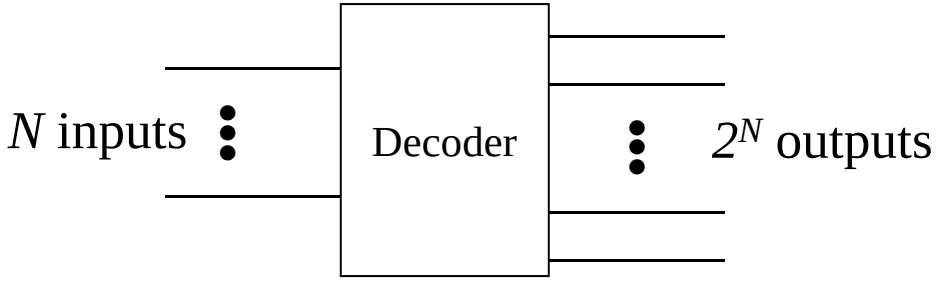


- Hold time



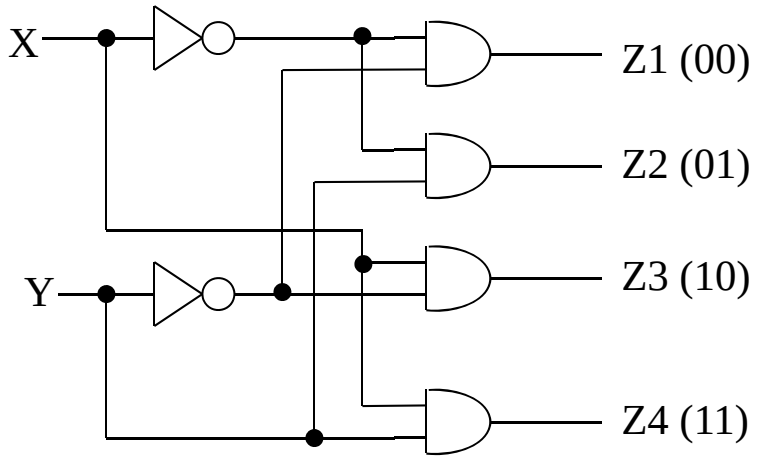
Decoder Circuits

- A decoder circuit uniquely selects one of its outputs according to its input signals



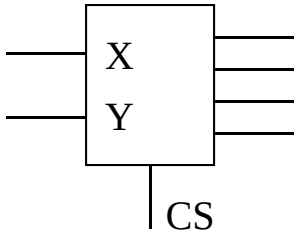
- 2-to-4 decoder implementation

X	Y	Z1	Z2	Z3	Z4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

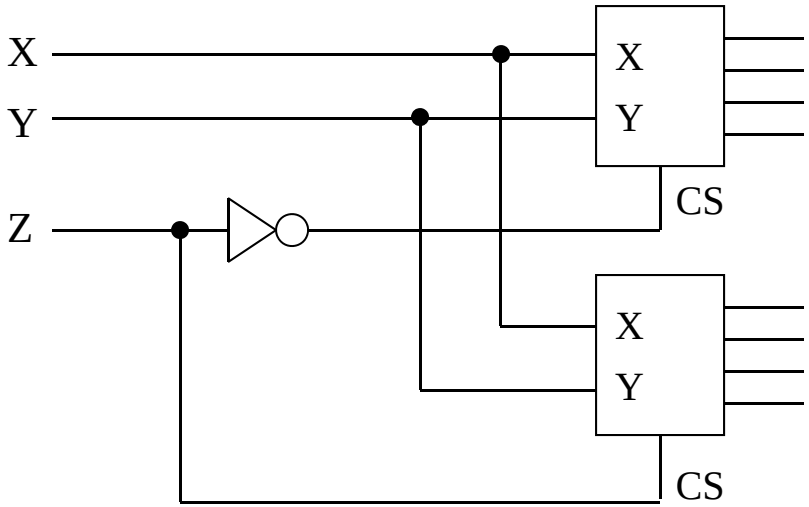


Decoder Circuit

- 3-to-8 decoder implementation
  - Assume that we have 2-to-4 decoders available as standard components
  - When CS is low (0), all the outputs of the decoder are low (0)

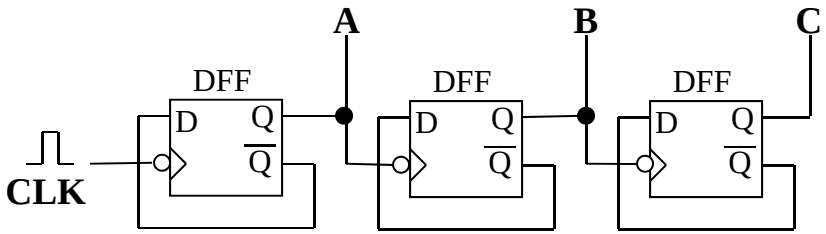


2-to-4 decoder



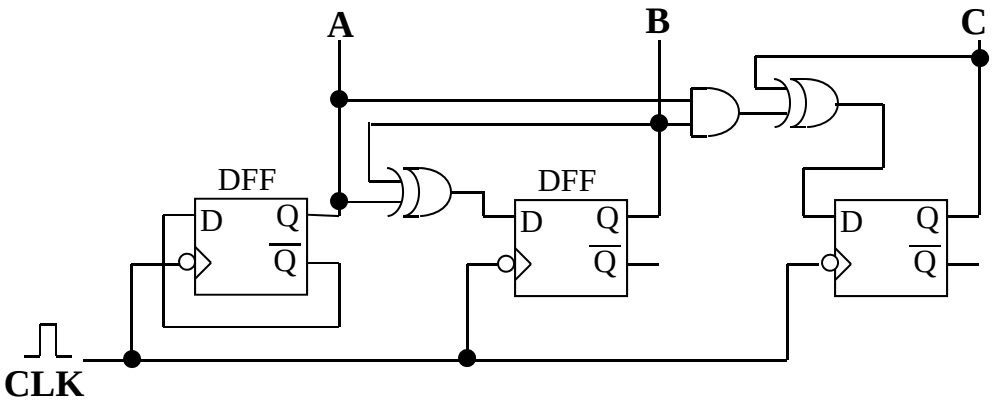
Sequential Logic Circuits

- Counter
  - Basic binary counter



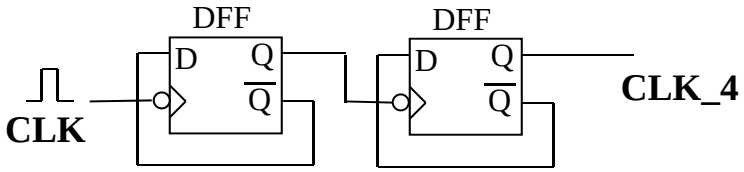
Clk	C	B	A
Init.	0	0	0
↓	0	0	1
↓	0	1	0
↓	0	1	1
↓	1	0	0
↓	1	0	1
↓	1	1	0
↓	1	1	1

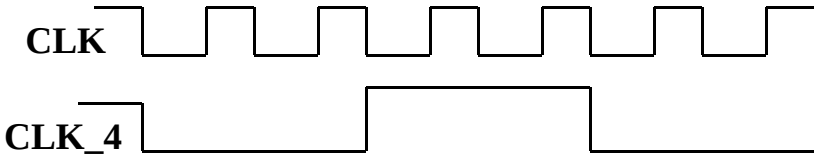
  - Synchronous counter



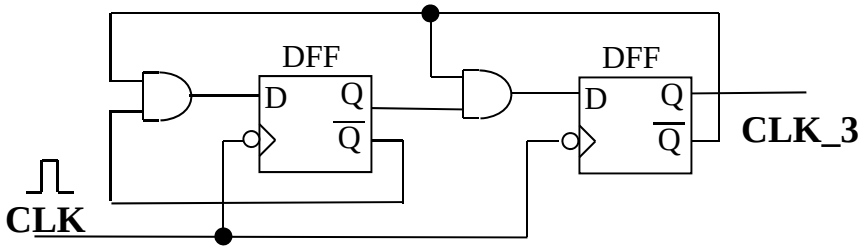
Sequential Logic Circuits

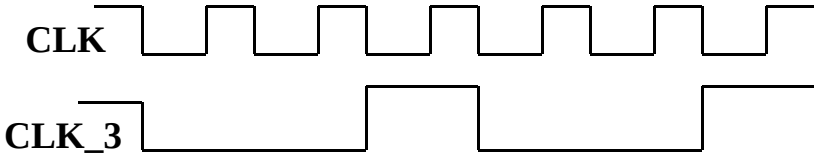
- frequency divider
  - Divide clock frequency by 4





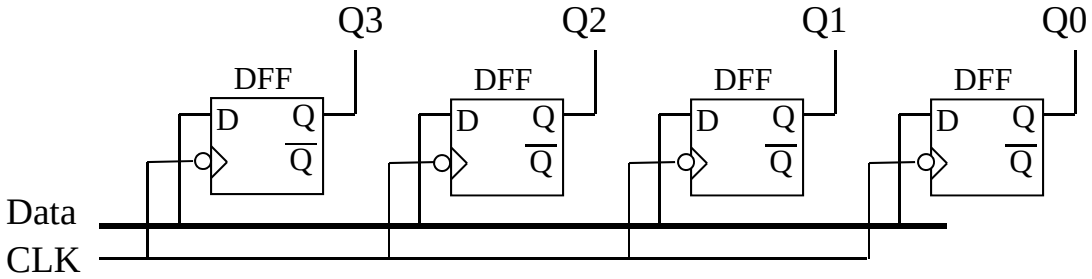
  - Divide clock frequency by 3





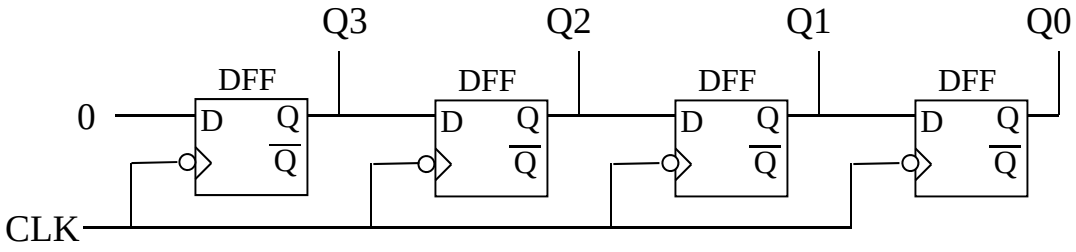
Sequential Logic Circuits

- Register
  - A row of storage elements (e.g. D flip-flops)



Q[3:0]

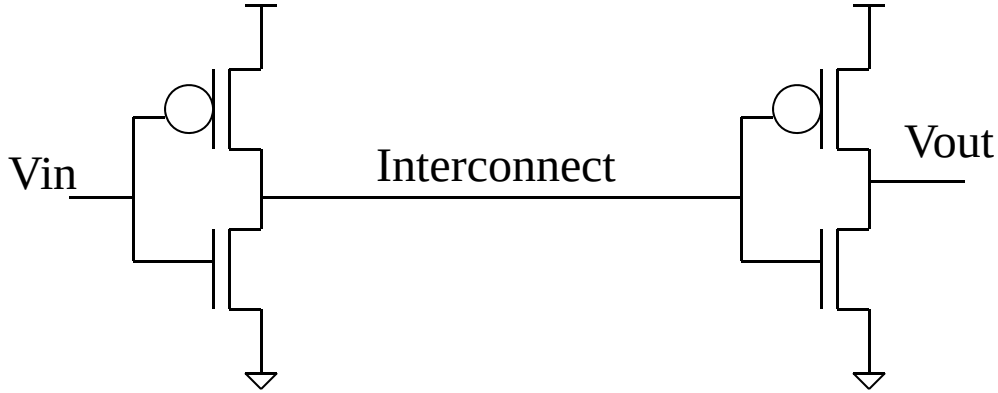
- Shift register

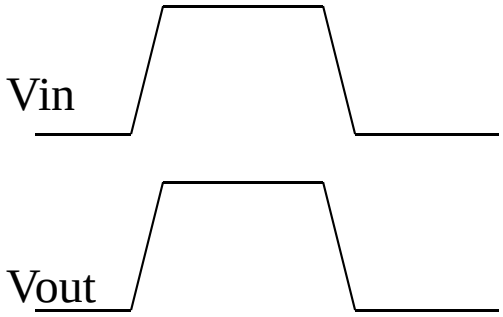


CLK	Q3	Q2	Q1	Q0
Init.	1	0	1	0
↓	0	1	0	1
↓	0	0	1	0
↓	0	0	0	1
↓	0	0	0	0

Major Parasitic Effects in Digital ICs

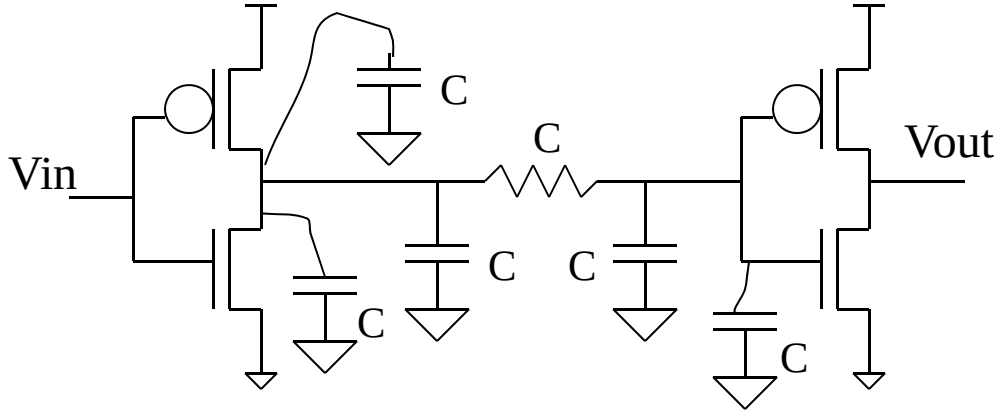
- Ideal Circuit

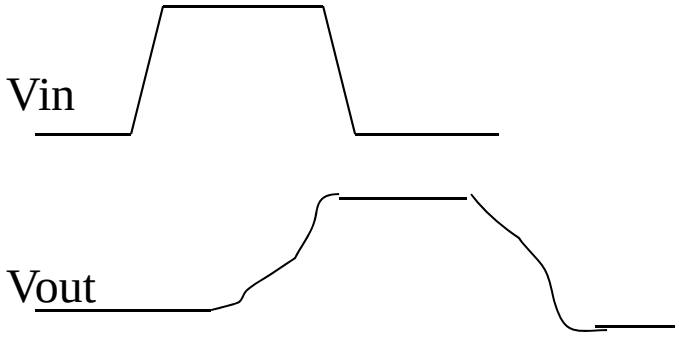




No delay!

- Real Circuit

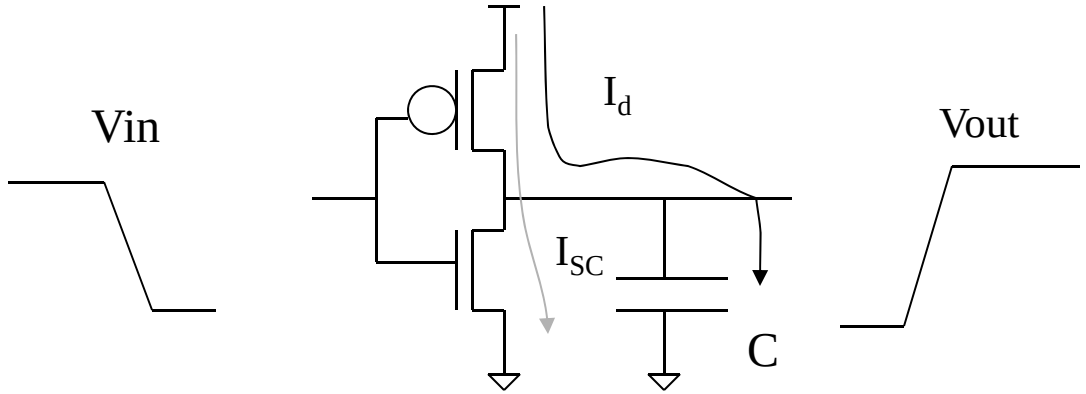




Significant delay!

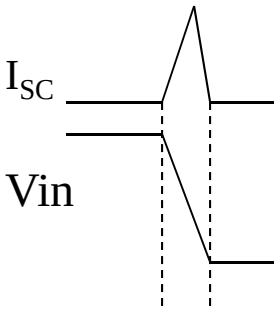
Power Consumption of CMOS Gates

- Dynamic Power Consumption



$$P_{Dynamic} = 0.5 \bullet C \bullet V_{DD}^2 \bullet f$$

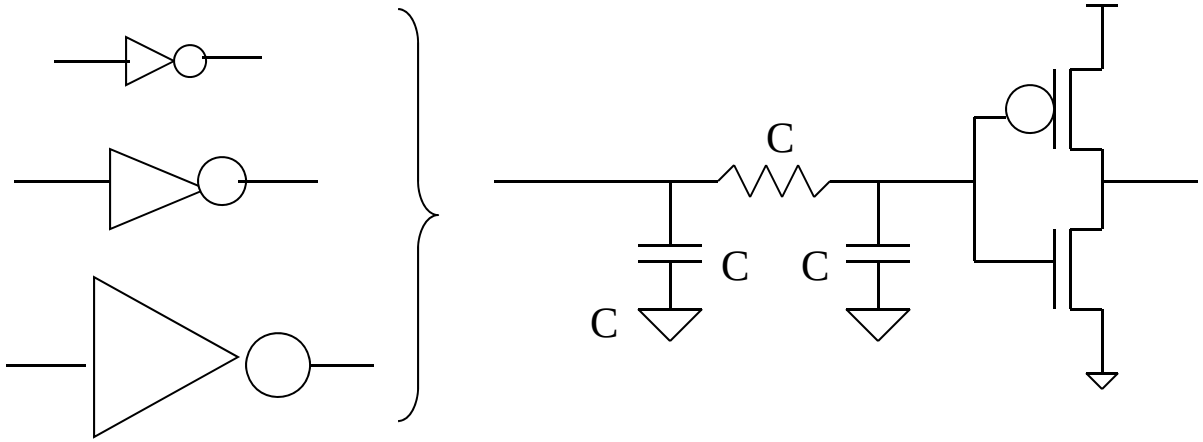
- Short-Circuit Power Consumption

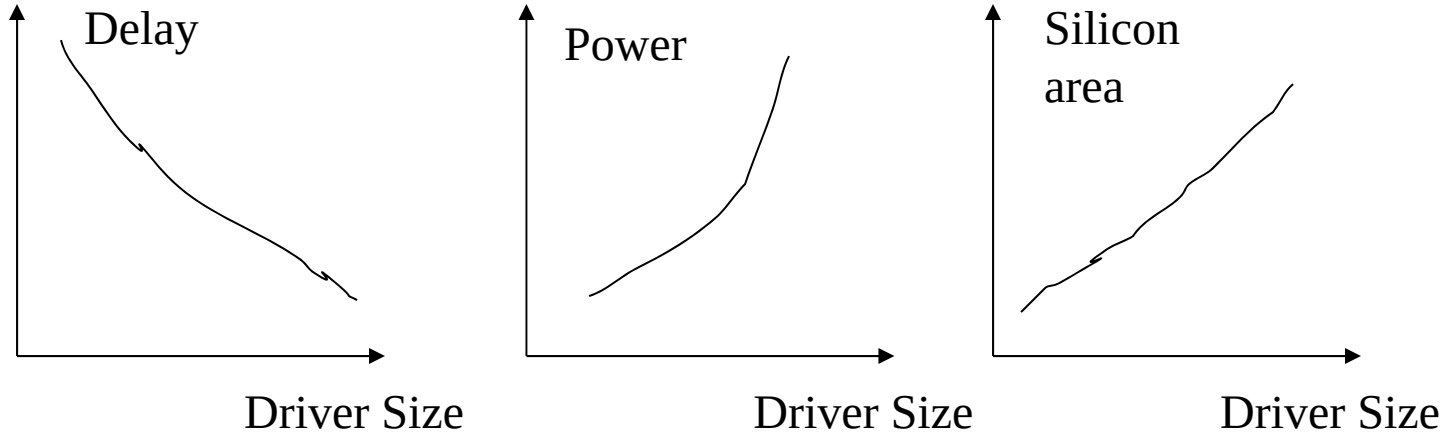


$$P_{SC} = \int I_{SC}(t) V dt$$

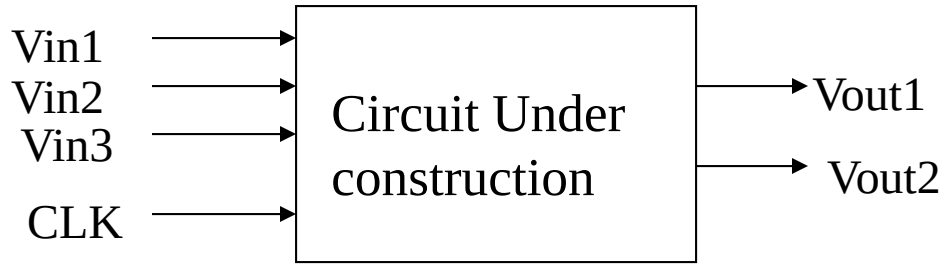
- Leakage Power Consumption

Transistor Sizing





Design Constraints for Digital ICs



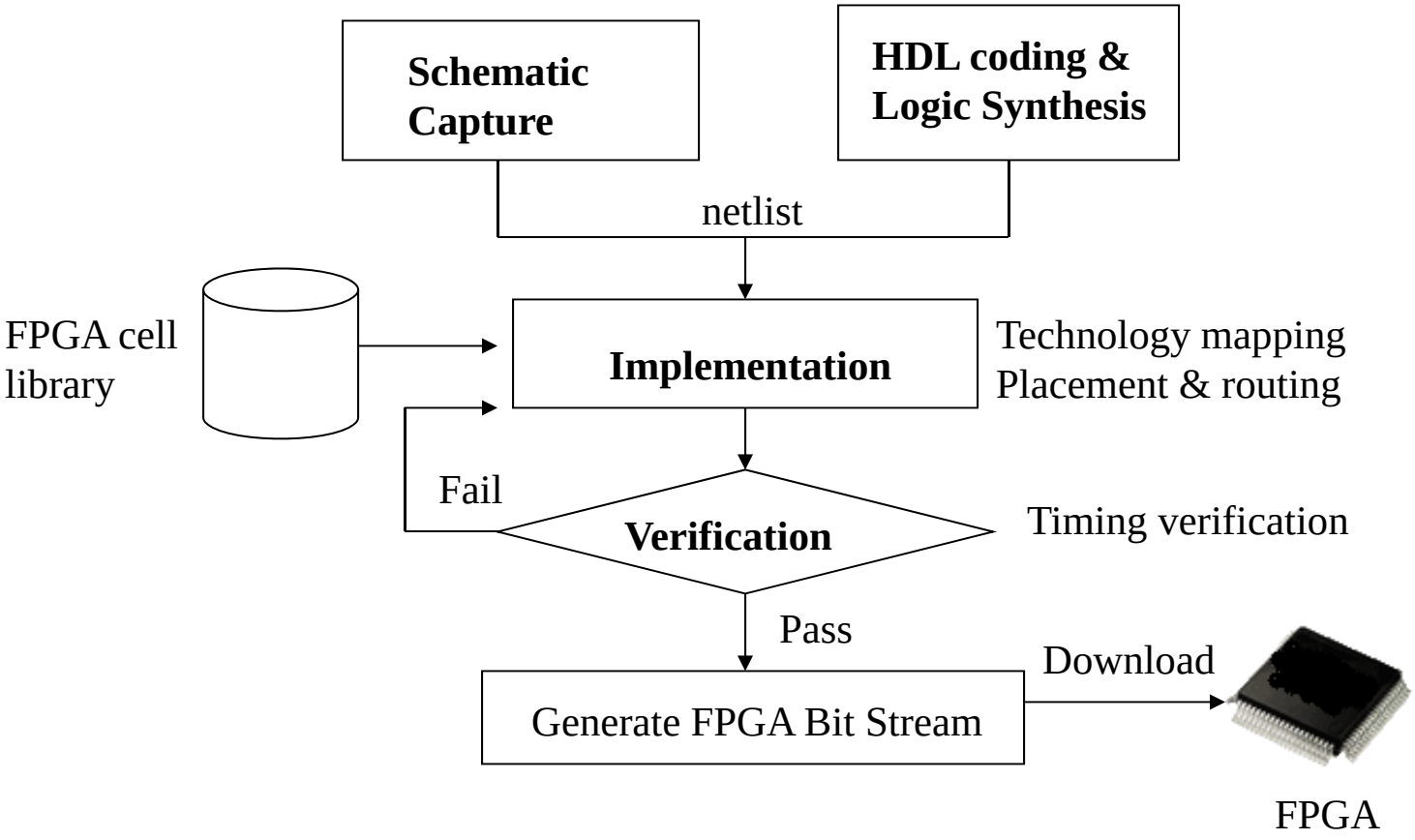
- Delay (speed) constraints
  - — For example, the delay from Vin1 to Vout1 should be smaller than 10 ns; the clock frequency should be greater 100MHz
- Power constraints
- Area constraints

How to Find Circuits Complying with Design Constraints

Full-custom approach	Standard-cell based approach Gate-array approach FPGA approach
1. It heavily depends on designers' expertise. 2. Finding a proper circuit implementation can be very time consuming. 3. Very creative circuit implementations can be achieved.	1. These approaches heavily use design automation techniques to search circuits complying with constraints. 2. They are time saving approaches. 3. These approaches require pre-developed technology libraries. 4. The quality of the final implementation depends on algorithms and libraries used in the search.

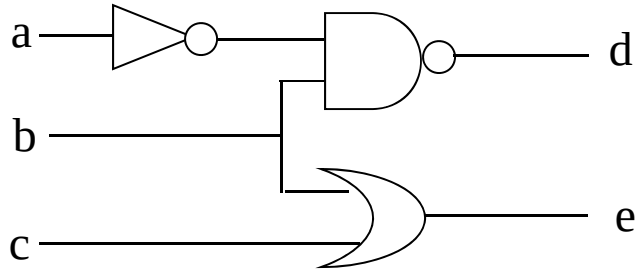
Example: How to Search a Proper Circuit Implementation in FPGA Design Flow

- FPGA Design Flow



Example: How to Search a Proper Circuit Implementation in FPGA Design Flow

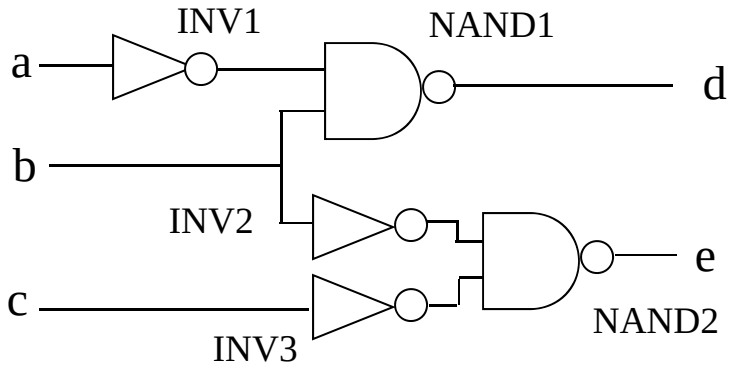
- Given Logic Function
  - This circuit is either from schematic capture or from logic synthesis



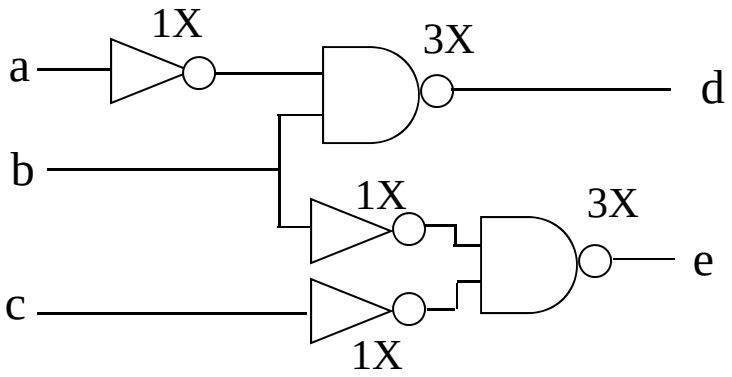
- FPGA Library Components
  - The library contains five components
  - Three inverters at size 1, 3, 5
  - Two two-input NAND gates at size 1 and 3
- Design Constraints
  - The maximum delay between an input and an output should not exceed 5 ns

Example: How to Search a Proper Circuit Implementation in FPGA Design Flow

- Step 1: Use the available logic gates to implement the given function. (Technology Mapping)



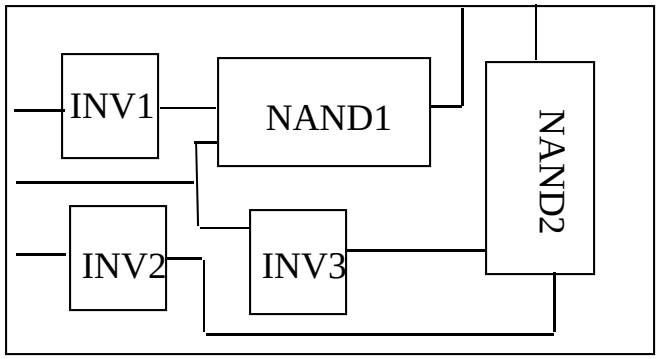
- Step 2: Select proper size for each gate used in the above circuit. (Gate Sizing)
  - — After this step, the circuit can be simulated to verify that it complies with timing constraints.
  - — In the simulation, the parasitic capacitance and resistance on interconnects are estimated (Estimated wire load)



Example: How to Search a Proper Circuit Implementation in FPGA Design Flow

- Step 3: Determine where to place these gates and how to connect them (Placement & Routing).

  — Some algorithms first place the components and then route the interconnects.
  — Some algorithms perform the placement and routing simultaneously



➢ Steps 1, 2, and 3 are included in the implementation phase of the FPGA design Flow

- Step 4: Perform post-layout simulation to verify that the generated circuit complies with timing constraints

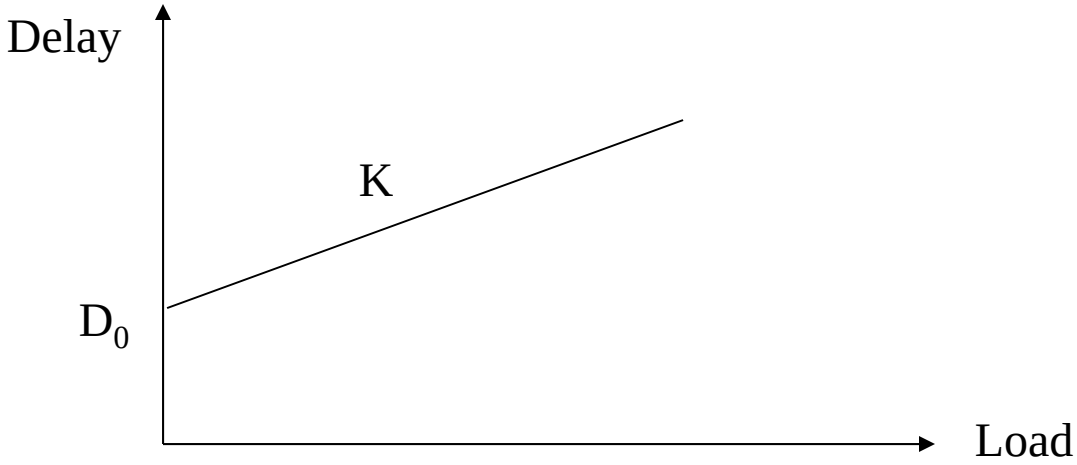
  — Since detail information of each interconnect is available, the circuit can be simulated with accurate wire load.
  — The process that writes the accurate wire load into the circuit netlist is called back annotation

What Information Should Be Stored in FPGA Library

- Logic function
- Timing (delay) information
- Power consumption information
- Silicon area
- Layout information
- Placement & routing constraints
- ••••
- ••••

Example: How to Store Delay Information

- Normally, the delay of gate is proportional to the load on its output



$$\text{Delay} = D_0 + K*\text{Load}$$

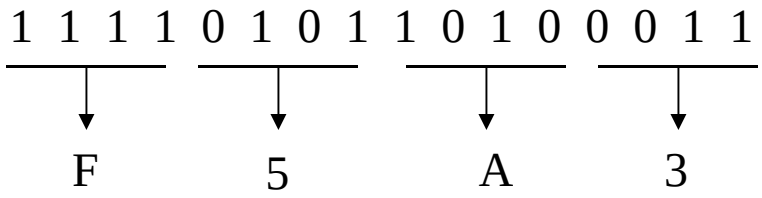
So, we can store two parameters (D_0 and K) for each gate to model its delay property.

Binary Number System

- Converting binary numbers to decimal numbers

Binary		Decimal
1011	→ 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0	= 11
0111	→ 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0	= 7

- Converting binary numbers to hexadecimal numbers

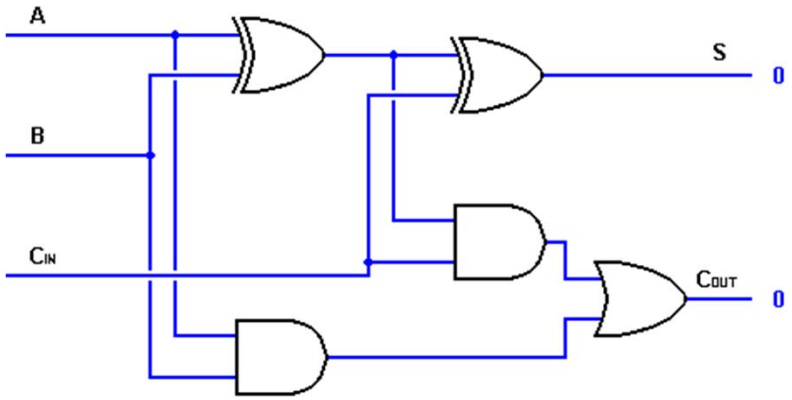


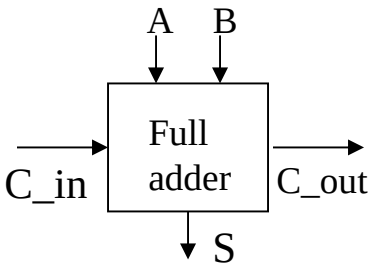
Binary Addition

- Example

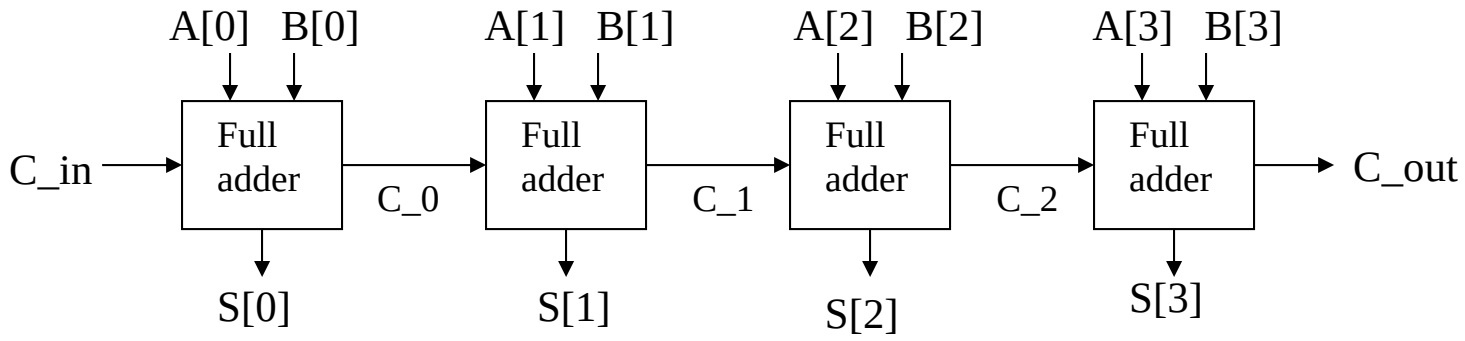
```
    0 1 0 1
+   1 0 0 1
-------------
    1 1 1 0
```

- A single-bit full adder





- 4-bit adder



Handling Negative numbers

- Signed magnitude
  - The left-most bit is a sign bit
    0 indicates positive a number and
    1 indicates negative a number

Signed magnitude			Decimal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	0
1	0	1	-1
1	1	0	-2
1	1	1	-3

- One's complement
  - For positive number A, it is represented as usual binary number
  - For negative number -A, its representation is obtained by flipping the bits of the binary representation of A

1's complement			Decimal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	1	1	0
1	1	0	-1
1	0	1	-2
1	0	0	-3

Handling Negative numbers

- Two's complement
  - For positive number A, it is the same as the one's complement
  - For negative number A, add one to the one's complement representation

2's complement			Decimal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	1	1	-1
1	1	0	-2
1	0	1	-3
1	0	0	-4

- By using two's complement number representation, minus operations can be performed by adders