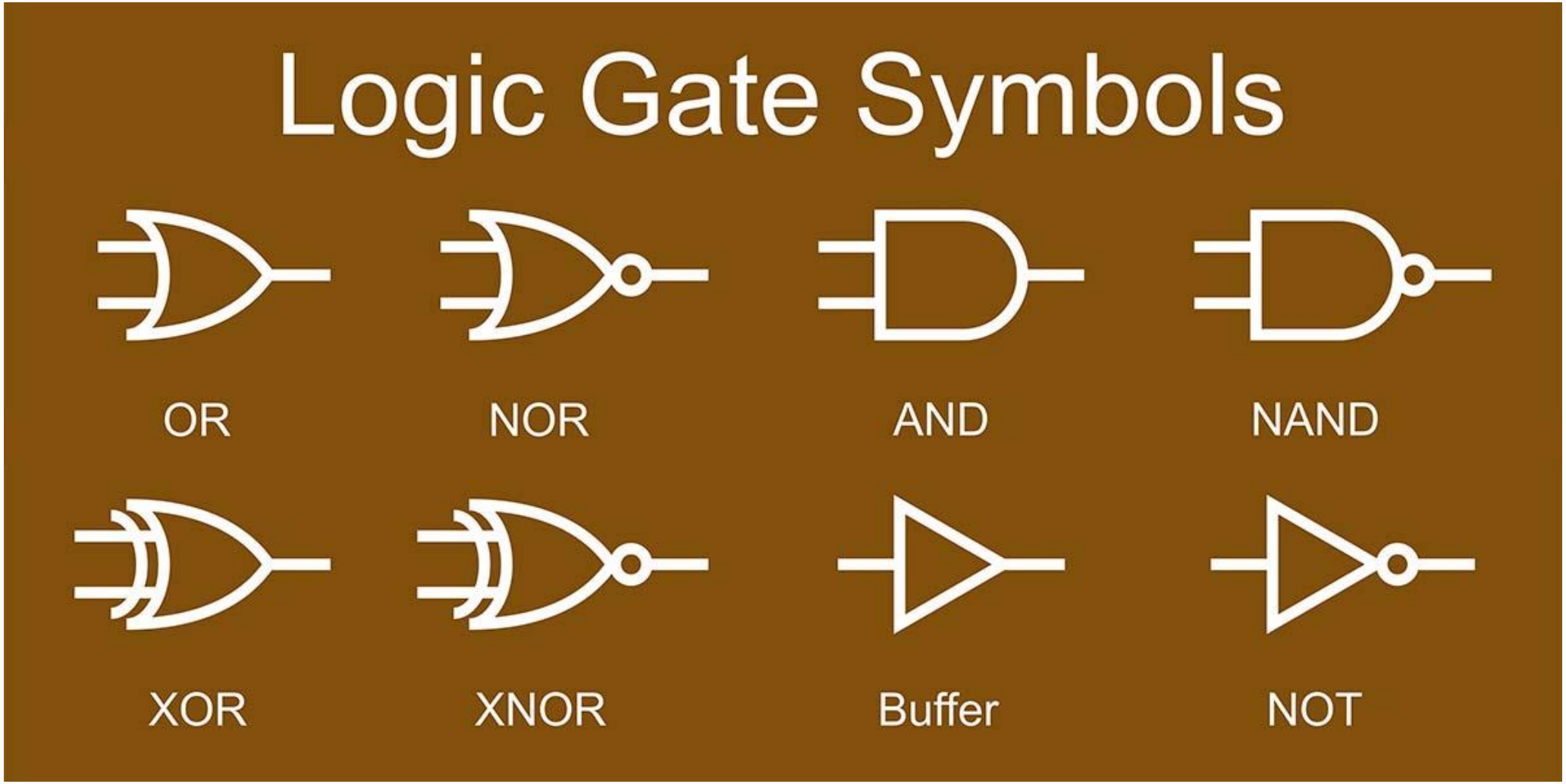
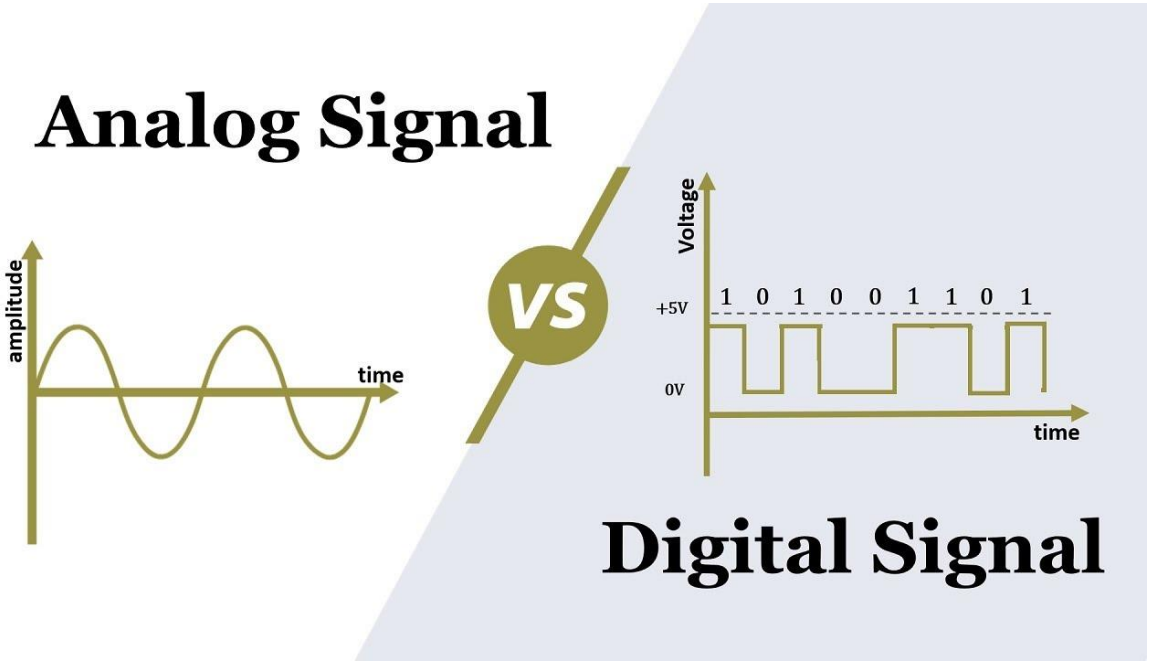
# TOPIC #1 Logic Design



# Introduction

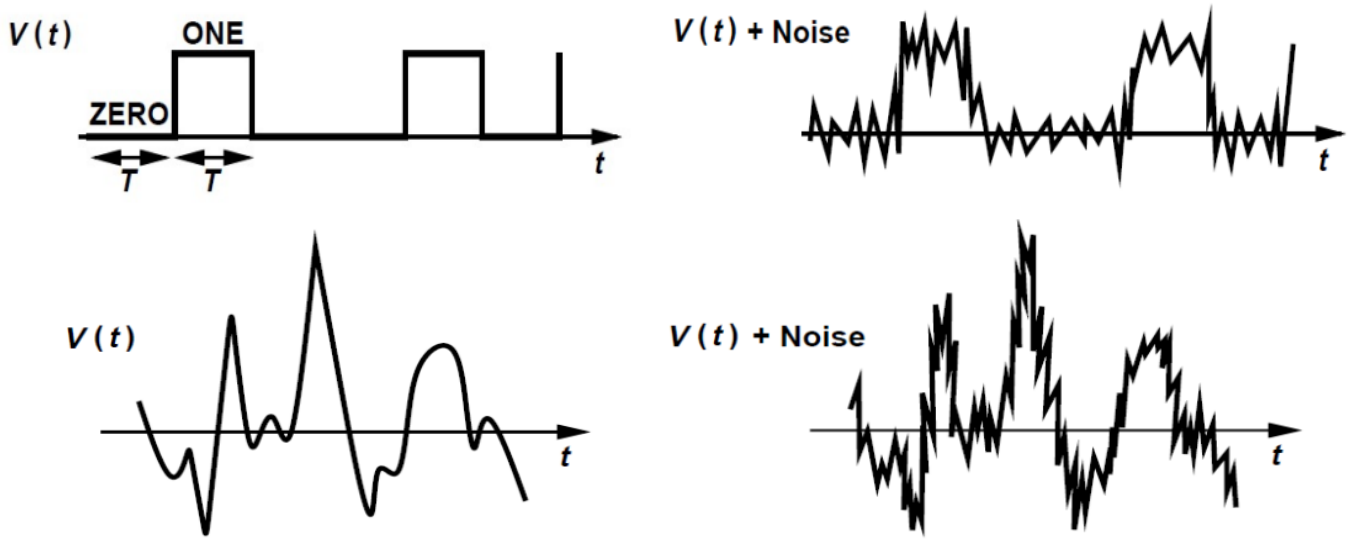
- Analog vs Digital
  Continuous and discrete
- The analog world or the digital world?



# Why we need digital system

- Digital signal is more **robust** than the analog one

- Digital signal is easy to be used (store and operate)



# Computer Arithmetic

- Skip

# Boolean Algebra (1/5)

- Logic algebra

  A kind of algebra about 0 and 1

- But!  What is algebra?

  AB =? BA ； x+y=z and x+w=z =>? y=w

- Any operation don’t follow the postulate(公設) / axiom (公理) must be proved
- Any theorem is derived from another theorem or postulate/axiom.

# Boolean Algebra (2/5)

1. Closure with respect to “+” and “•”
2. An identity element with respect to “+” and “•”.

$$x + 0 = 0 + x = x \text{ and } x \bullet 1 = 1 \bullet x = x$$

3. Commutative with respect to “+” and “•”

$$x + y = y + x \text{ and } x \bullet y = y \bullet x$$

4. Distributive over “+” and “•”.

$$x \bullet (y + z) = (x \bullet y) + (x \bullet z) \text{ and } x + (y \bullet z) = (x + y) \bullet (x + z)$$

5. For $x \in B$, there exists $x' \in B$ (complement of x) such that

$$x + x' = 1 \text{ and } x \bullet x' = 0.$$

6. There exist at least two elements $x, y \in B$, such that $x \neq y$.

# Boolean Algebra (3/5)

| | | | | | |
|---|---|---|---|---|---|
| Pos. | 2 | (a) | $x + 0 = x$ | (b) | $x \cdot 1 = x$ |
| Pos. | 5 | (a) | $x + x' = 1$ | (b) | $x \cdot x' = 0$ |
| Thm. | 1 | (a) | $x + x = x$ | (b) | $x \cdot x = x$ |
| Thm. | 2 | (a) | $x + 1 = 1$ | (b) | $x \cdot 0 = 0$ |
| Thm. | 3, involution | (a) | $(x')' = x$ | (b) | |
| Pos. | 3, commutative | (a) | $x + y = y + x$ | (b) | $xy = yx$ |
| Thm. | 4, associative | (a) | $x + (y + z) = (x + y) + z$ | (b) | $x(yz) = (xy)z$ |
| Pos. | 4, distributive | (a) | $x(y + z) = xy + xz$ | (b) | $x + yz = (x + y)(x + z)$ |
| Thm. | 5, DeMorgan | (a) | $(x + y)' = x' \cdot y'$ | (b) | $(xy)' = x' + y'$ |
| Thm. | 6, absorption | (a) | $x + xy = x$ | (b) | $x(x + y) = x$ |

# Boolean Algebra (4/5)

- Thm. 1(a): $x + x = x$

- Thm. 1(b): $x \bullet x = x$

# Boolean Algebra (5/5)

- Thm. 2: $x + 1 = 1$

- Thm. 6: $x + xy = x$

# DeMorgan’s Law(1/3)

- (x + y)' = x' • y’ (General)

  To prove the two set are equivalent, we use **Elementwise method**.

# DeMorgan’s Law(2/3)

- $(x + y)' = x' \bullet y'$

# DeMorgan’s Law(3/3)

- (x • y)' = x’ + y’
  - Homework

# Simplification of Boolean Algebra

1. Mathematical way
   - Not intuitive, complicate
2. Truth table
   - Easy, but may not be the simplest circuit.
   - Used with K-Map
3. Karnaugh map
   - Easy, often used in Logic design course but the number of the parameters may not exceed 6.
4. Other ways

# Truth Table(1/3)

| $x$ | $y$ | $x + y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**AND Logic**

| $x$ | $y$ | $x + y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**OR Logic**

| $x$ | $x'$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

**NOT Logic**

# Truth Table(2/3)

- Example
  - $F_1 = x + y'z$
  - $F_2 = x'y'z + x'yz + xy'z' + xy'z$
    $= x'y'z + x'yz + xy'$

| $x$ | $y$ | $z$ | $F_1$ | $F_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

# Truth Table(3/3)

- Simplify F= x’y’z + x’yz + xy’ = x’z(y’ + y) + xy’ = x’z + xy’

# Karnaugh map

- There are some details to discuss
- Skip but refer to 清大開放式課程 數位邏輯設計

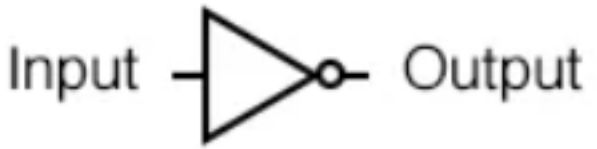
# Logic Gate (1/2)

2 - input AND gate


| A | B | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

2 - input OR gate


| A | B | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




| Input | Output |
|---|---|
| 0 | 1 |
| 1 | 0 |

# Logic Gate (2/2)

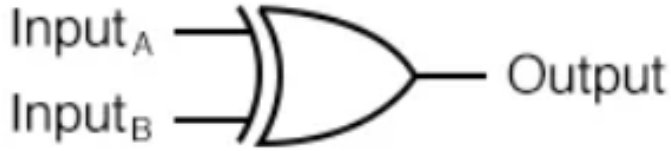
2 - input NAND gate


| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

2 - input NOR gate

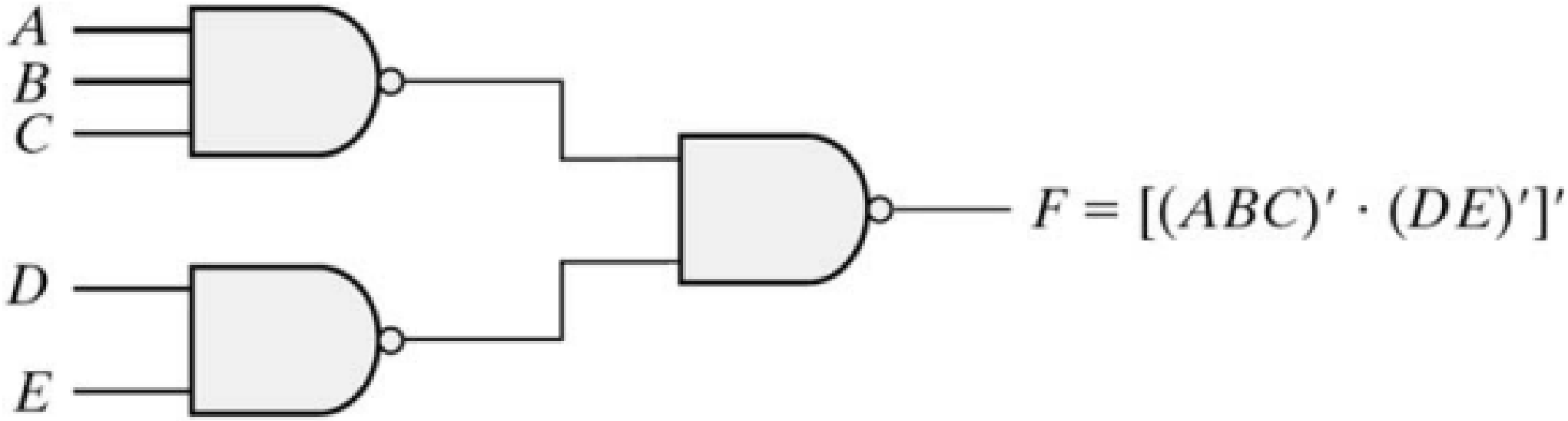

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Exclusive-OR gate




| A | B | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Exercise 1



# Exercise 2-1(algebra)

- F = ((AB)’+A’)B+A

# Exercise 2-2(K-Map)

- F = ((AB)’+A’)B+A

# Exercise 3-1 (algebra)

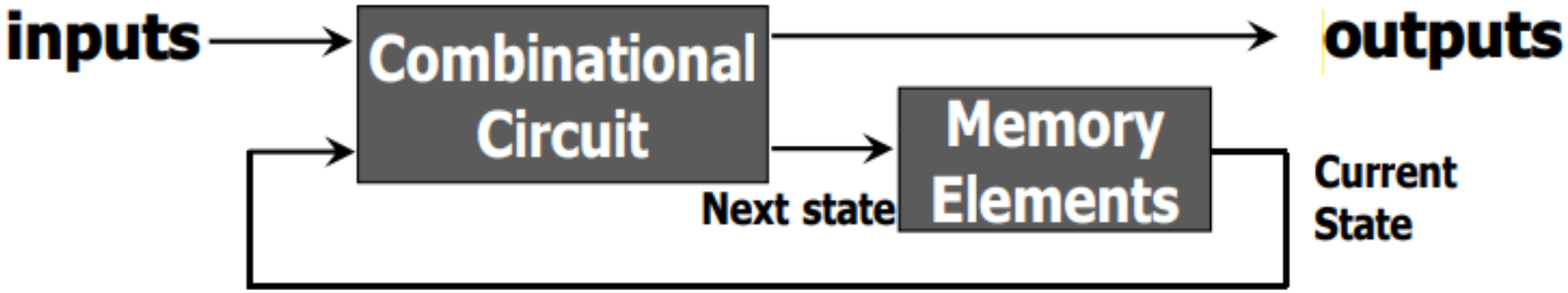
- F = (A+BC)(A+B)’

# Exercise 3-2 (K-Map)

- F = (A+BC)(A+B)’

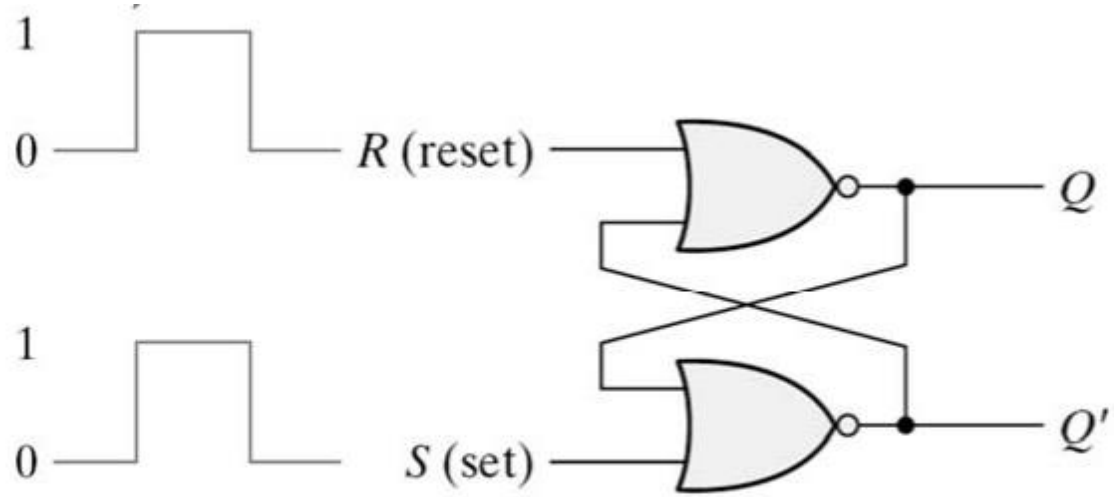
# Combination/ Sequential Logic

- Combinational circuits contain no memory elements, and the outputs depend on the current inputs.
- Designing the sequential logic, you need to consider “clock” well. The main element in sequential logic is memory devices. And we will only introduce Flip-Flop.



# Latch(1/3)

- Latch is a logic device that it can help us be able to control signal well.
- Latch is an asynchronous sequential circuit. (state changes whenever inputs change).




(a) Logic diagram

| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | (after $S = 1, R = 0$) |
| 0 | 1 | 0 | 1 | |
| 0 | 0 | 0 | 1 | (after $S = 0, R = 1$) |
| 1 | 1 | 0 | 0 | (forbidden) |

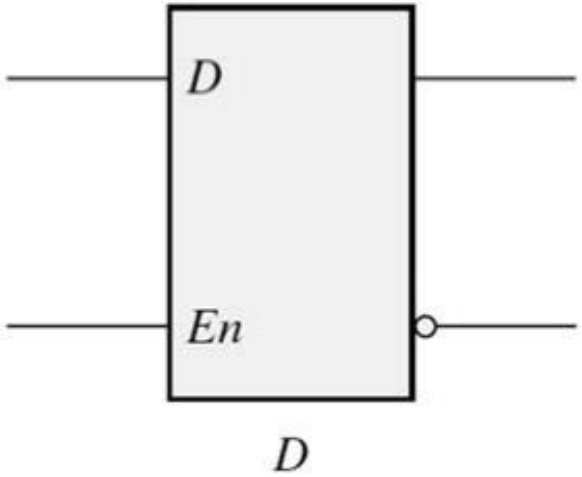
(b) Function table

# Latch(2/3)

- Exercise: Draw the output of an SR latch for the input waveforms shown below.

# Latch(3/3)

- D Latch
  - D latch can eliminate the undesirable condition of the indeterminate state in the SR latch
  - D >> Q when En = 1; no change when En = 0
  - A transparent latch when En = 1, then D >> Q

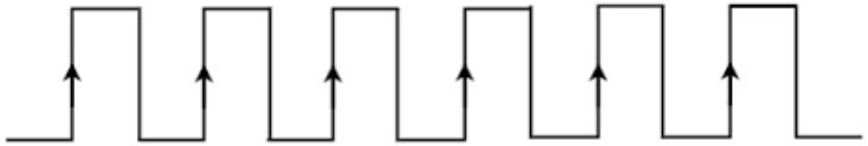



**En = 0, No change**
**En = 1, Q = D**

# Trigger

- A trigger
  - The change of the output state of a latch or flip-flop is enabled by a change of the control input (Enable). This momentary change is called trigger.
- Level triggered

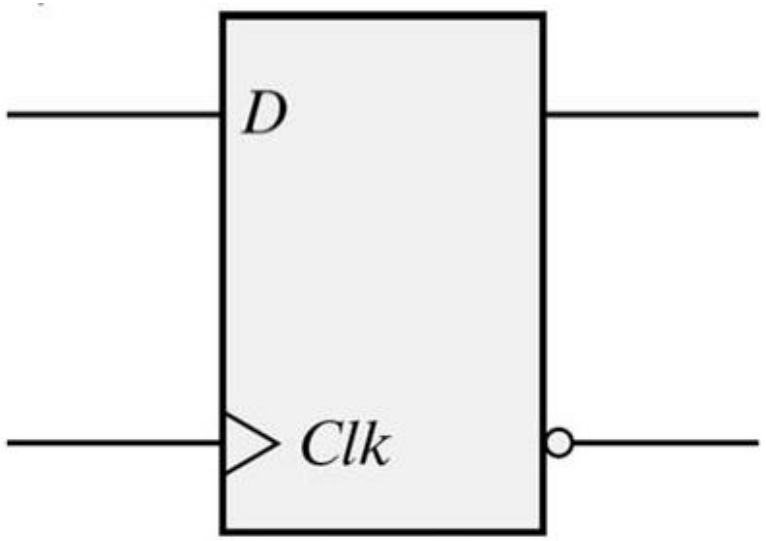
  - The state transition starts as soon as clock(Enable) is during logic 1 or logic 0 level.
  - The change of input makes the combination logic keep changing with the input latch at logic 1 or logic 0.
- Edge triggered



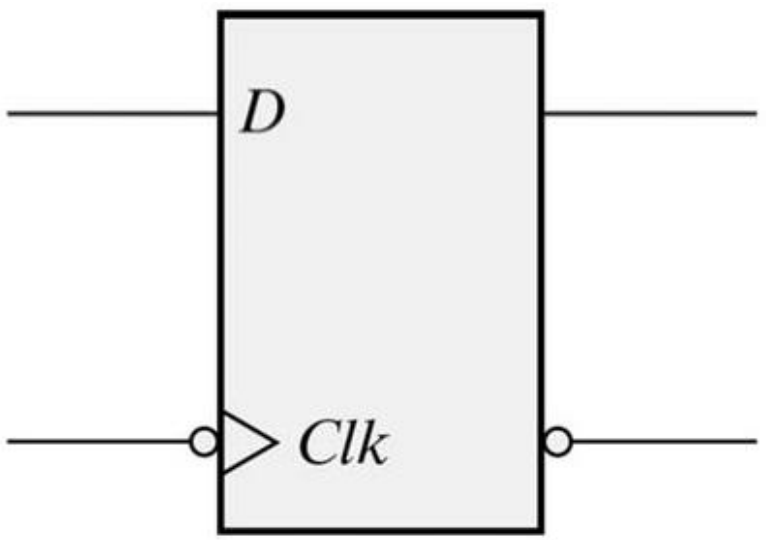
  - The state transition starts only at positive or negative edge of the clock signal.
  - The edge triggered flip-flops will isolate the input changes (current state) and output driving logic (previous state).

# Flip-Flop(1/4)

- D Flip-Flop
  - Edge-Triggered



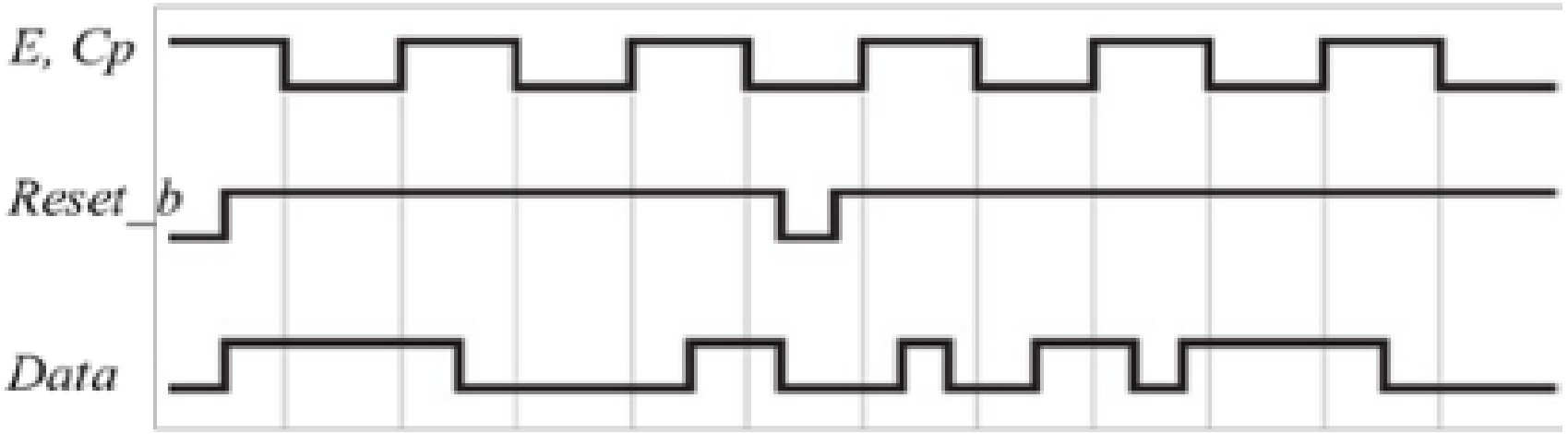

(a) Positive-edge



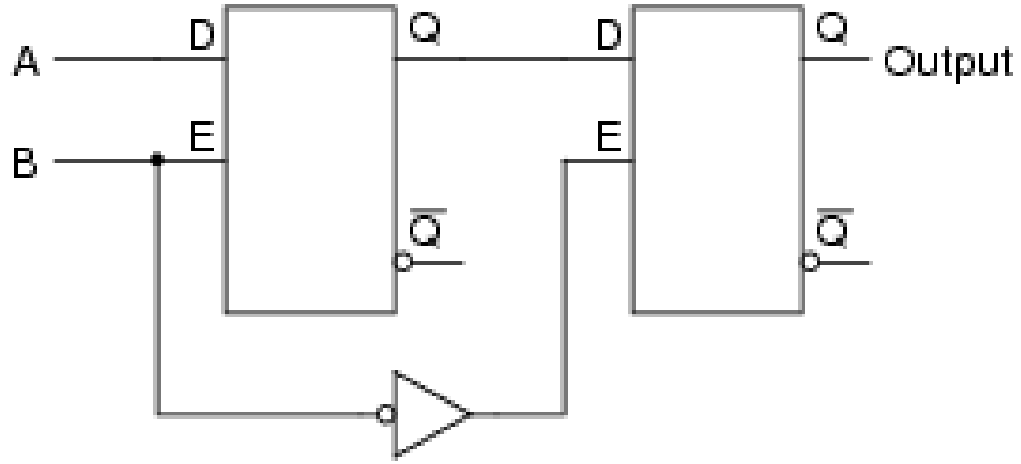

(a) Negative-edge

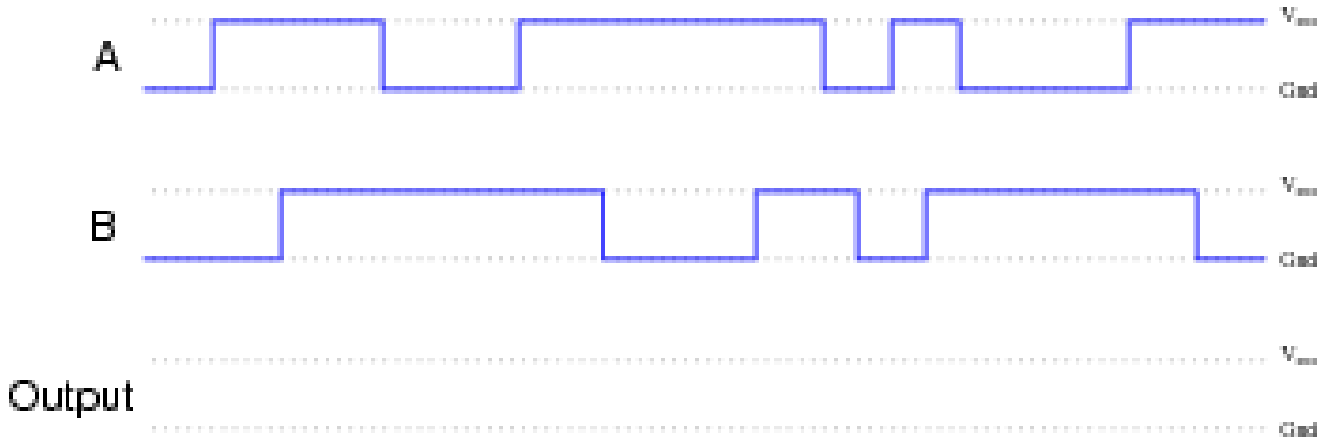
# Flip-Flop(2/4)

- Exercise: Please plot the graph of D-FF and D-latch according to the following plot.

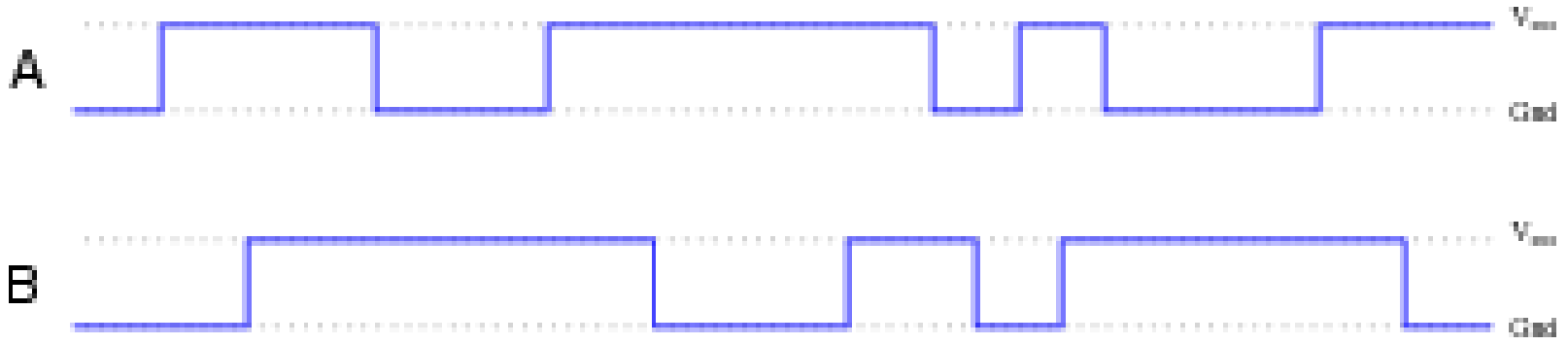


# Flip-Flop(3/4)



# Flip-Flop(4/4)



# Homework

- 1
- 2(a)(c)
- 3
- 4
- 5