

# **Chương 3**

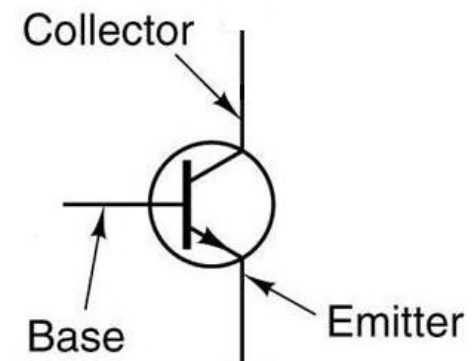
## **Mạch logic số**

# Nội dung

- Transistor và các cổng logic
- Đại số Boole
- Mạch tổ hợp
- Mạch tính toán
- Mạch tuần tự
- Mạch bộ nhớ

# Transistor và các cổng logic

- Transistor
  - Phần tử cơ bản nhất cấu tạo máy tính số ngày nay là transistor do John Bardeen và Walter Brattain phát minh năm 1947.
  - Transistor thường được sử dụng như một thiết bị khuếch đại hoặc một khóa điện tử
- Mỗi transistor đều có ba cực:
  - Cực gốc (*base*)
  - Cực góp (*collector*)
  - Cực phát (*emitter*)

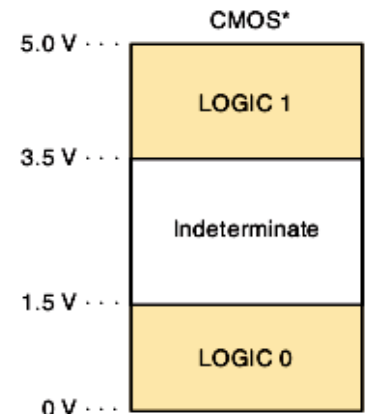
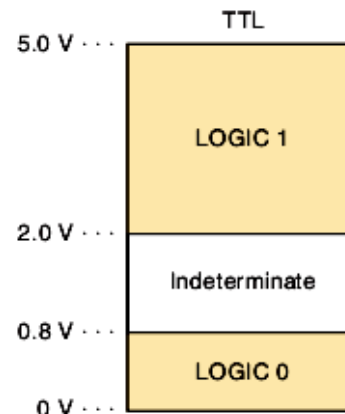


# Transistor và các cổng logic

- Cổng logic (gate)
  - Các transistor được ghép nối lại để tạo thành các cổng logic có thể thực hiện các phép toán logic cơ bản: NOT, AND, OR, NAND (NOT AND) và NOR (NOT OR)

- Giá trị logic

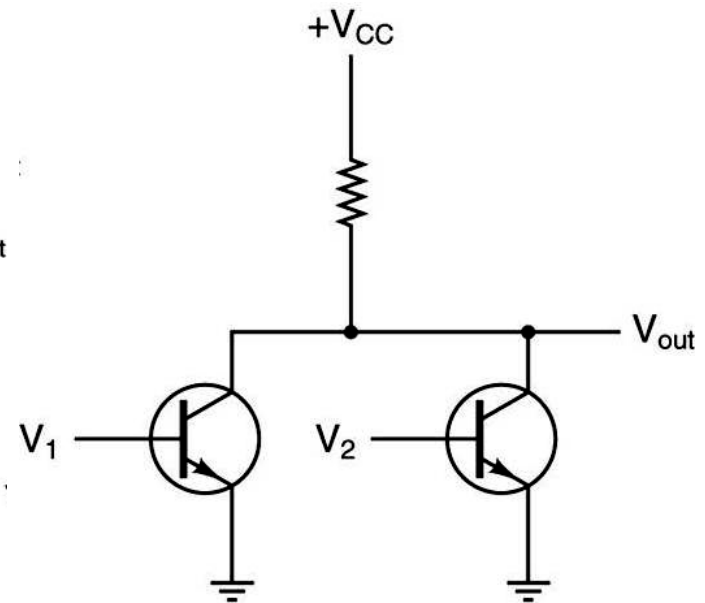
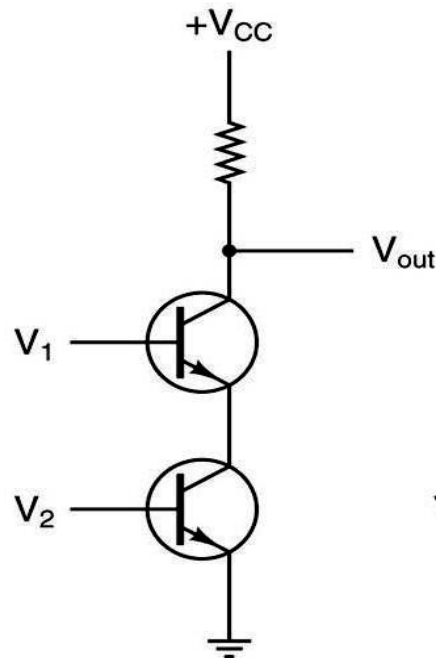
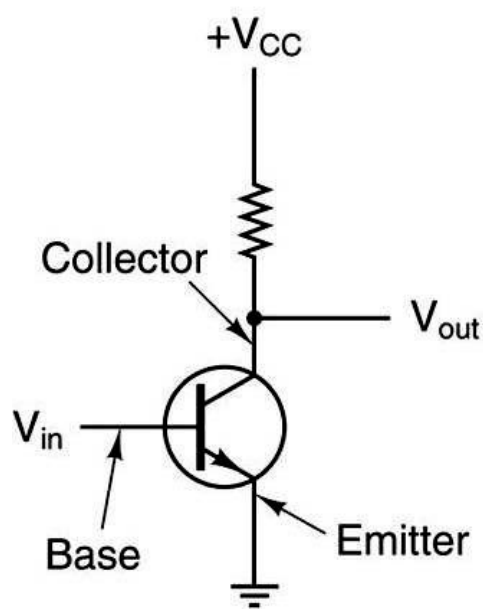
- 0 : mức điện áp 0...1,5 volt
- 1 : mức điện áp 2...5 volt



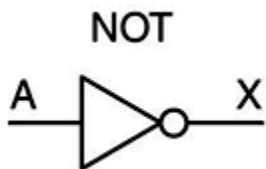
- Các cổng cơ bản này lại được lắp ghép thành các phần tử chức năng lớn hơn như mạch cộng 1 bit, nhớ 1 bit, v.v... từ đó tạo thành 1 máy tính hoàn chỉnh

# Transistor và các cổng logic

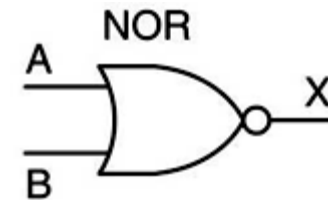
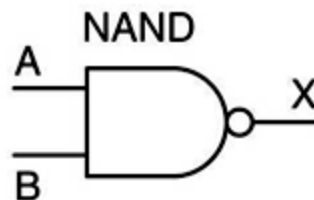
- Cấu tạo các cổng cơ bản NOT, NAND và NOR



- Ký hiệu

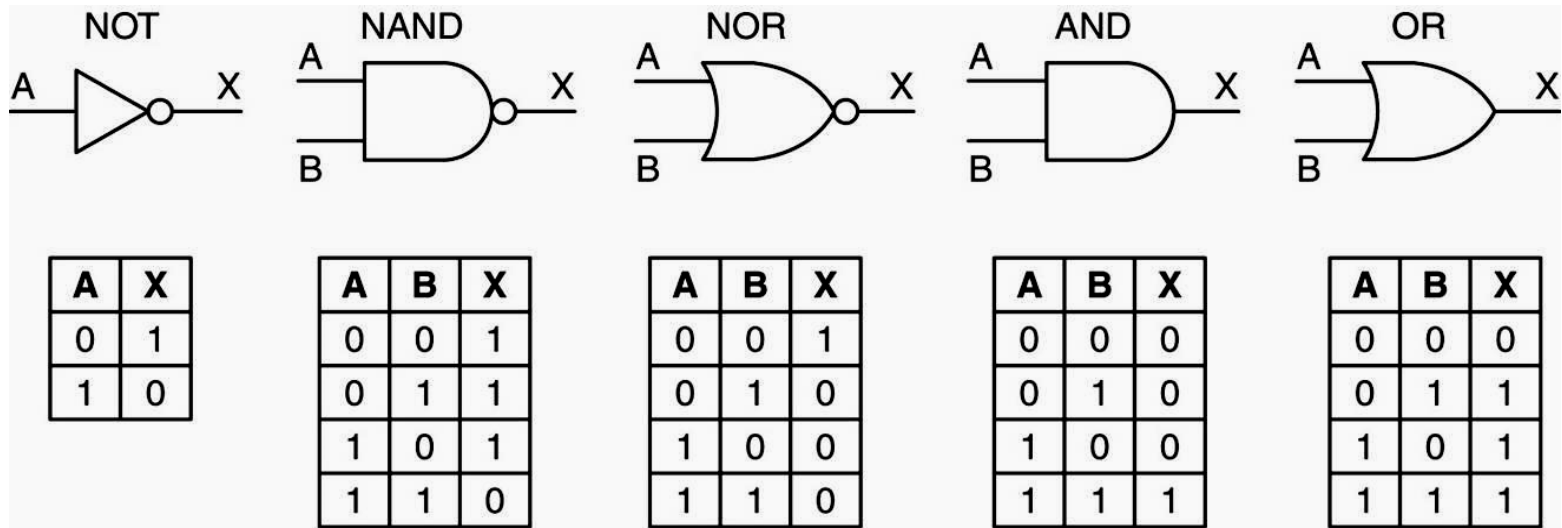


IT-FIT 2021



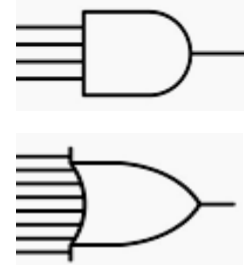
# Transistor và các cổng logic

- Bảng chân trị và ký hiệu các cổng logic cơ bản



- Đối với các cổng nhiều ngõ vào, ngõ ra X=1 khi:

- AND : mọi ngõ vào bằng 1
- OR: ít nhất 1 ngõ vào bằng 1
- NAND : ít nhất 1 ngõ vào bằng 0
- NOR : mọi ngõ vào bằng 0



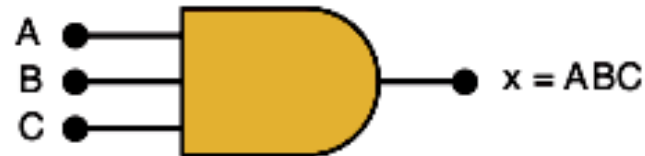
# Transistor và các cổng logic

- Bảng chân trị các cổng OR và AND 3 ngõ vào



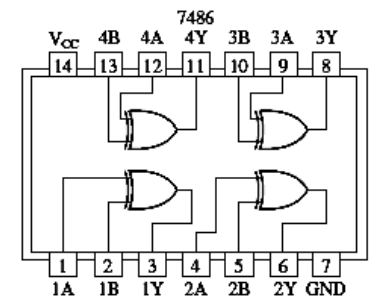
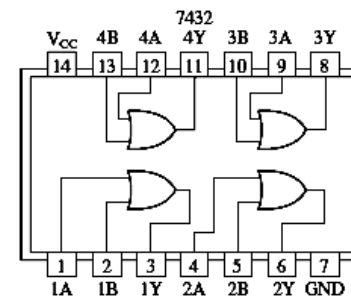
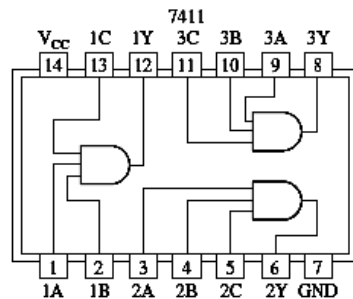
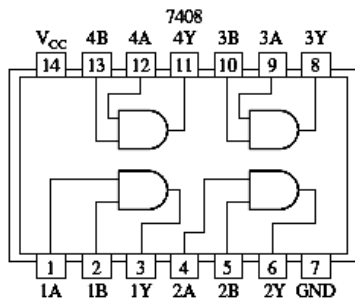
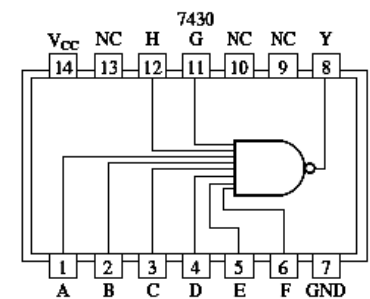
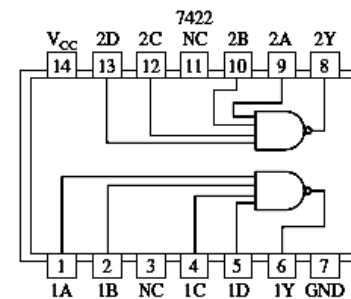
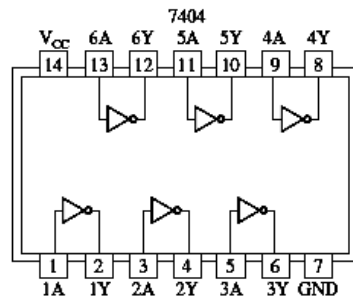
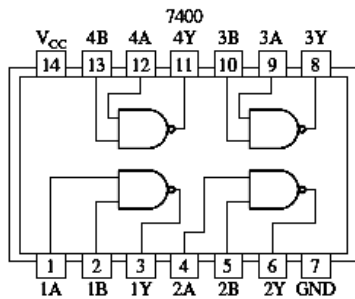
A	B	C	$x = A + B + C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A	B	C	$x = ABC$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



# Transistor và các cổng logic

- Một số vi mạch họ 7400





# Đại số Boole

- Giới thiệu
  - Đại số Boole (Boolean algebra) do nhà toán học George Boole phát triển từ năm 1854 làm cơ sở cho phép toán logic.
  - Năm 1938 Claude Shannon chứng minh có thể dùng đại số Boole để thiết kế mạch số trong máy tính
  - Đại số Boole dựa trên các biến logic và các phép toán logic
    - Biến logic có thể nhận giá trị 1 (TRUE) hoặc 0 (FALSE)
    - Phép toán logic cơ bản là AND, OR và NOT
    - Hàm logic gồm tập các phép toán và biến logic

# Đại số Boole

- Các phép toán logic cơ bản
  - Phép toán logic cơ bản AND, OR và NOT với ký hiệu như sau:
    - A AND B :  $A \cdot B$
    - A OR B :  $A + B$
    - NOT A :  $\overline{A}$
  - Các phép toán khác: NAND, NOR, XOR:
    - A NAND B :  $\overline{A \cdot B}$
    - A NOR B :  $\overline{A + B}$
    - A XOR B :  $A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$
  - Thứ tự ưu tiên: NOT, AND và NAND, OR và NOR

EXCLUSIVE OR gate



# Đại số Boole

- Bảng chân trị (Truth table)

P	Q	NOT P ( $\bar{P}$ )	P AND Q ( $P \cdot Q$ )	P OR Q ( $P + Q$ )	P NAND Q ( $\overline{P \cdot Q}$ )	P NOR Q ( $\overline{P + Q}$ )	P XOR Q ( $P \oplus Q$ )
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

- Ứng dụng đại số Boole
  - Phân tích chức năng mạch logic số
  - Thiết kế mạch logic số dựa trên hàm cho trước

# Đại số Boole

- Ví dụ 1: Cài đặt 1 hàm logic  $M=F(A, B, C)$  theo bảng chân trị cho trước
- Qui tắc:  $M=0$  nếu mọi đầu vào là 0,  $M=1$  nếu mọi đầu vào là 1 (tổng các tích).
- Bước 1: Xác định các dòng trong bảng chân trị có kết quả bằng 1
- Bước 2: Các biến đầu vào được AND với nhau nếu giá trị trong bảng bằng 1. Nếu giá trị biến bằng 0 cần NOT nó trước khi AND
- Bước 3: OR tất cả các kết quả từ bước 2.

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$M=\bar{A}BC+A\bar{B}C+AB\bar{C}+ABC$$

# Đại số Boole

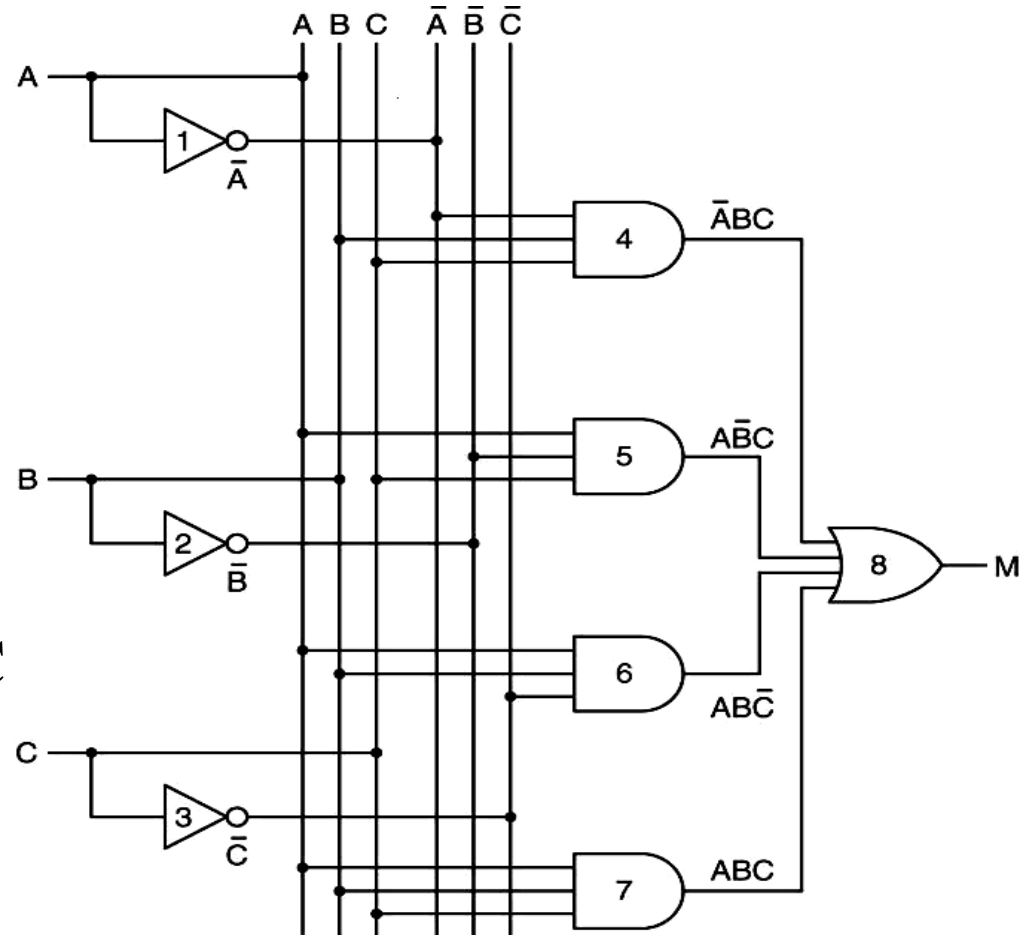
- Ví dụ 1 (tiếp)

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$M = \bar{A}BC + A\bar{B}C + ABC\bar{C} + ABC$$

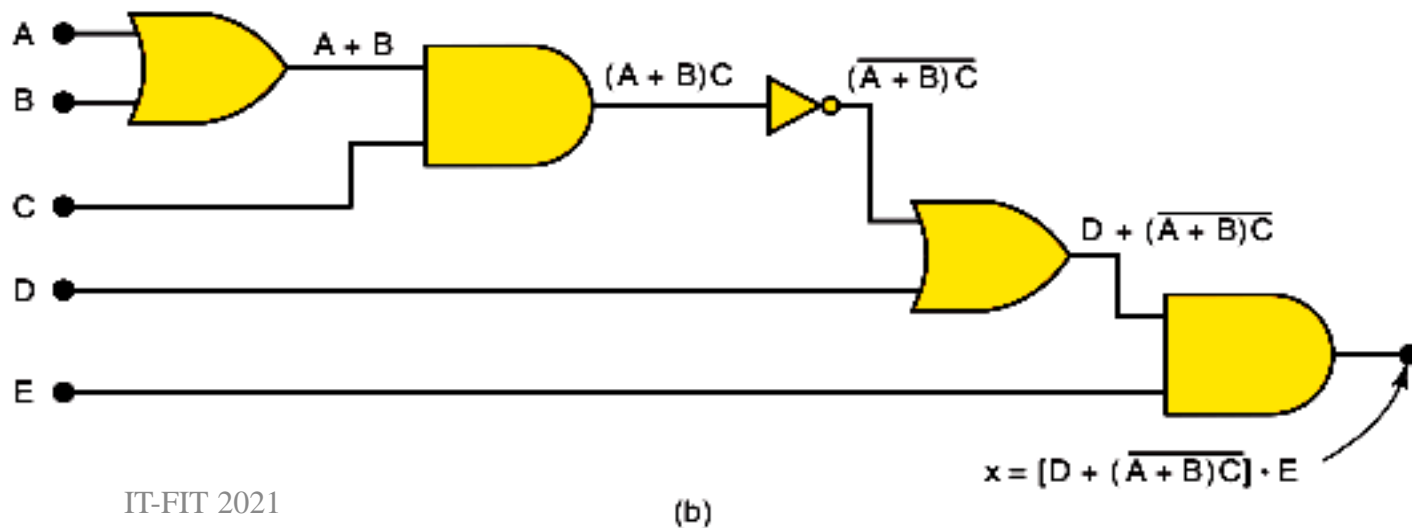
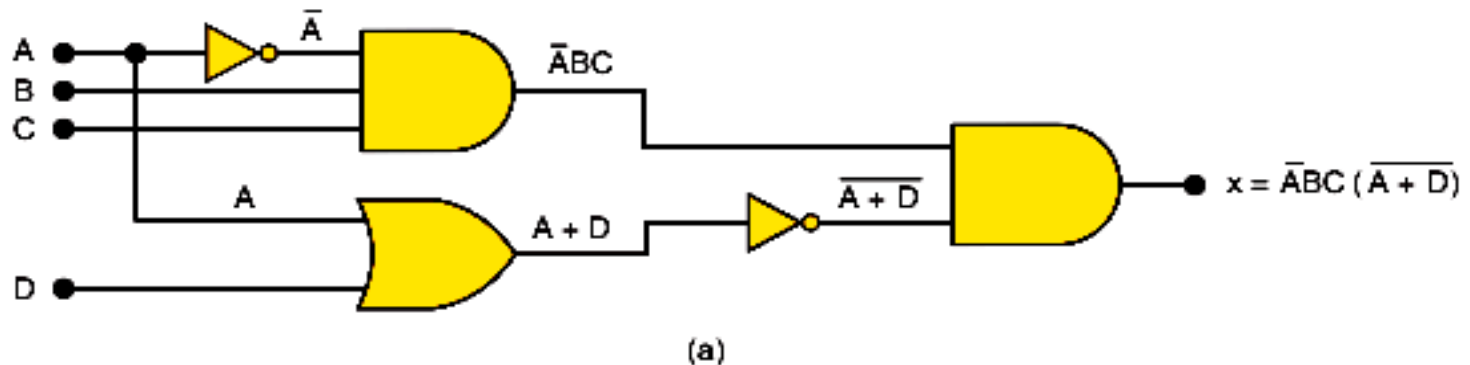
**Chú ý:**

- Mạch thiết kế theo cách này chưa tối ưu.
- Có 3 cách biểu diễn 1 hàm logic



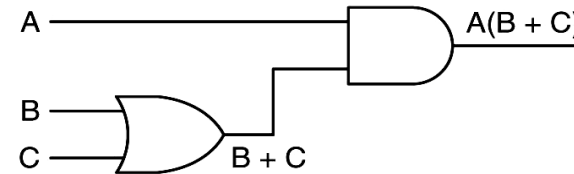
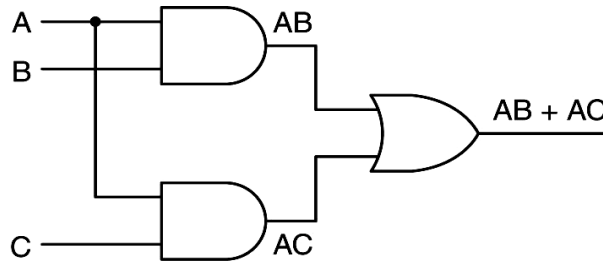
# Đại số Boole

- Ví dụ 2: Xác định hàm logic từ mạch cho trước



# Đại số Boole

- Các mạch tương đương
  - Ví dụ:  $AB+AC$  và  $A(B+C)$



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

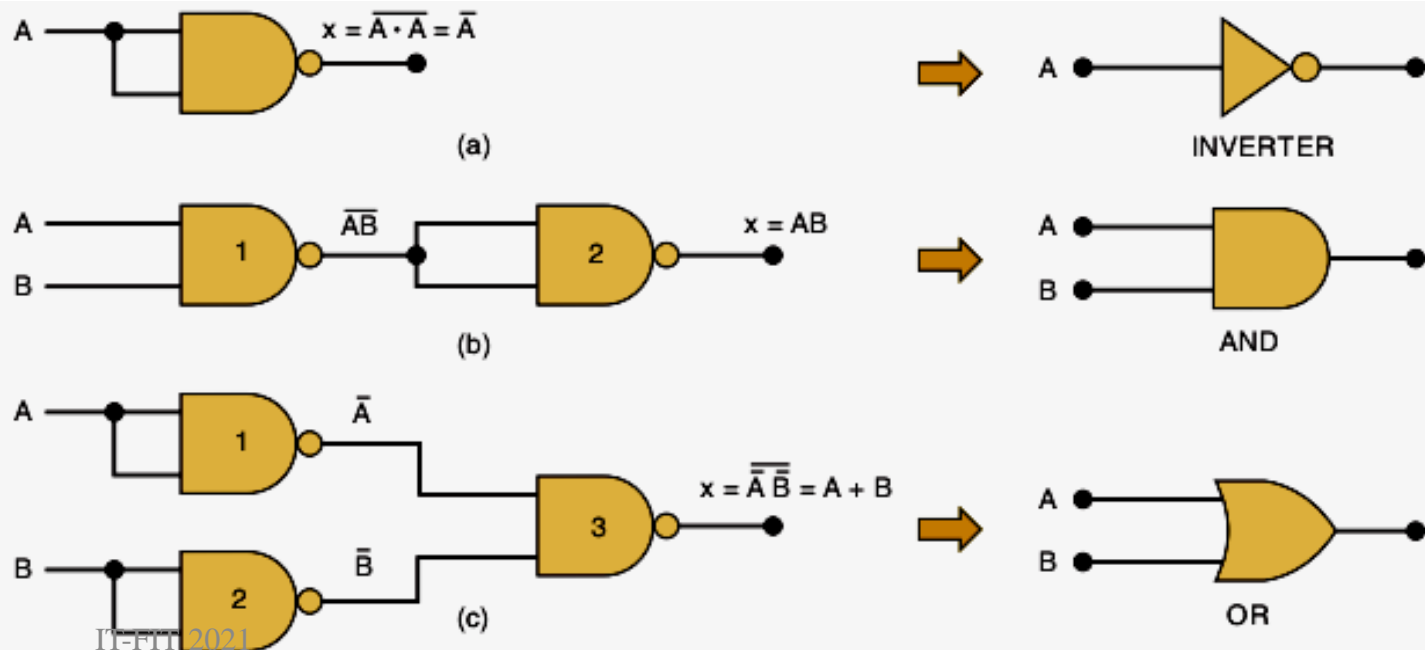
(a)

A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

# Đại số Boole

- Các mạch tương đương (tiếp)
  - Nhận xét: Nên sử dụng mạch tiết kiệm các cổng logic nhất
  - Trong thực tế người ta dùng cổng NAND (hoặc NOR) để tạo ra mọi cổng khác





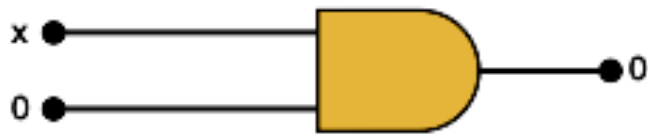
# Đại số Boole

- Các định luật của đại số Boole
- Đồng nhất
- Rỗng
- Không đổi
- Ngịch đảo
- Giao hoán
- Kết hợp
- Phân phối
- Rút gọn

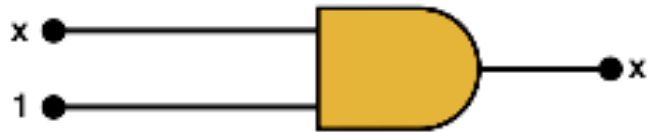
Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

# Đại số Boole

- Các định luật của đại số Boole (tt)



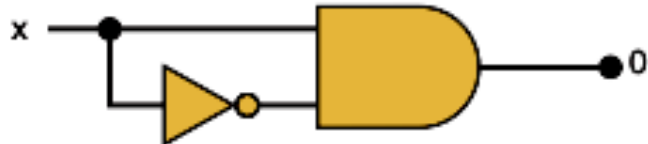
(1)  $x \cdot 0 = 0$



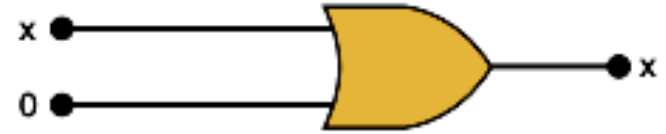
(2)  $x \cdot 1 = x$



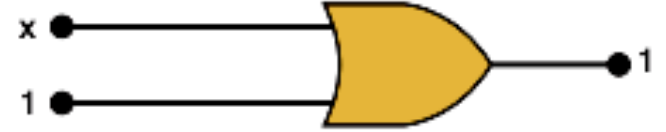
(3)  $x \cdot x = x$



(4)  $x \cdot \bar{x} = 0$



(5)  $x + 0 = x$



(6)  $x + 1 = 1$



(7)  $x + x = x$



(8)  $x + \bar{x} = 1$

# Đại số Boole

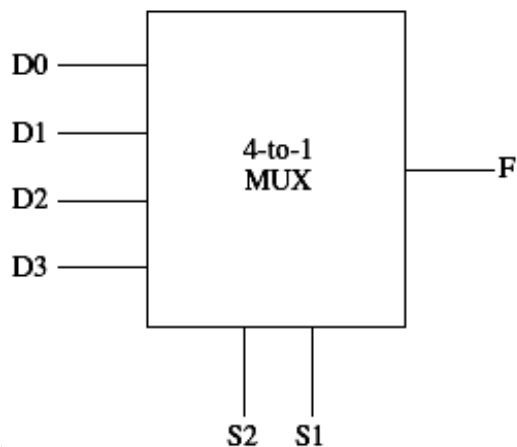
- Ứng dụng các định luật
  - Đơn giản biểu thức logic  $\rightarrow$  Tiết kiệm cổng logic
  - Ví dụ : Chứng minh  $AB + AC^{\overline{}} + BC = AB + AC^{\overline{}}$ 
$$\begin{aligned} & AB + AC^{\overline{}} + BC \\ &= AB + AC^{\overline{}} + 1 \cdot BC \\ &= AB + AC^{\overline{}} + (A + A) \cdot \overline{BC} \\ &= AB + AC^{\overline{}} + ABC + AB\overline{C} \\ &= AB + ABC + AC^{\overline{}} + AB\overline{C} \\ &= AB \cdot 1 + ABC + AC^{\overline{}} \cdot 1 + AC^{\overline{}} \cdot B \\ &= AB(1 + C) + AC^{\overline{}}(1 + B) \\ &= AB \cdot 1 + AC^{\overline{}} \cdot 1 = AB + AC^{\overline{}} \end{aligned}$$

# Mạch tổ hợp

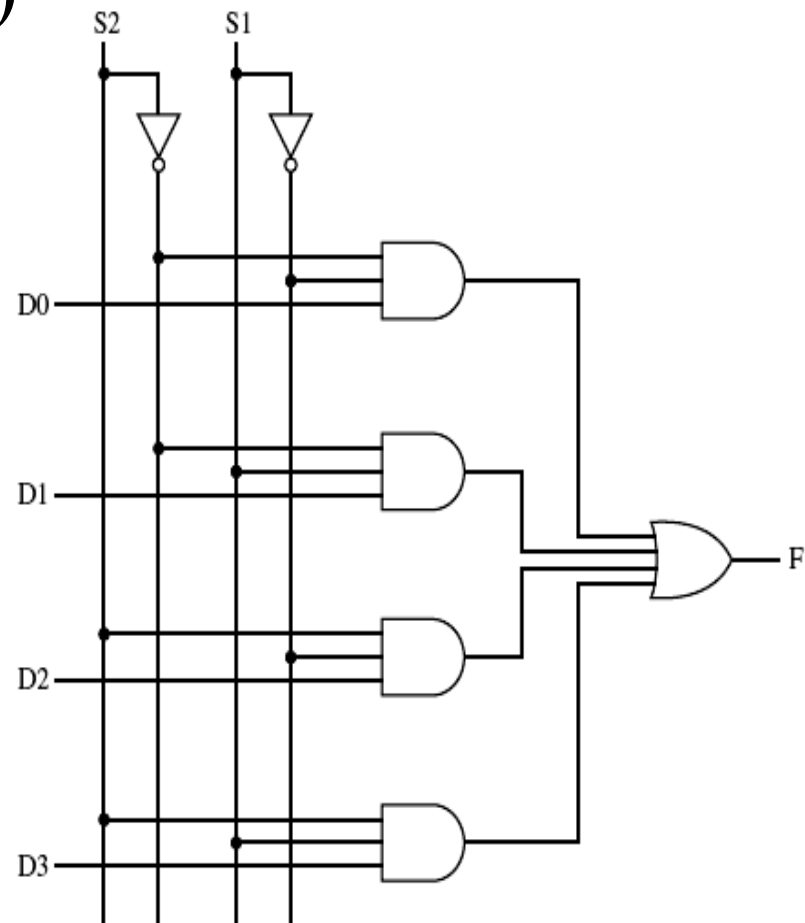
- Khái niệm
  - Mạch tổ hợp (combinational circuit) là mạch logic trong đó tín hiệu ra chỉ phụ thuộc tín hiệu vào ở thời điểm hiện tại.
  - Là mạch không nhớ (memoryless) và được thực hiện bằng các cổng logic cơ bản
  - Mạch tổ hợp được cài đặt từ 1 hàm hoặc bảng chân trị cho trước
  - Được ứng dụng nhiều trong thiết kế mạch máy tính

# Mạch tổ hợp

- Bộ dồn kênh (Multiplexer)
  - $2^n$  đầu vào dữ liệu D
  - n đầu vào lựa chọn S
  - 1 đầu ra F
  - (S) xác định đầu vào (D) nào sẽ được nối với đầu ra (F)



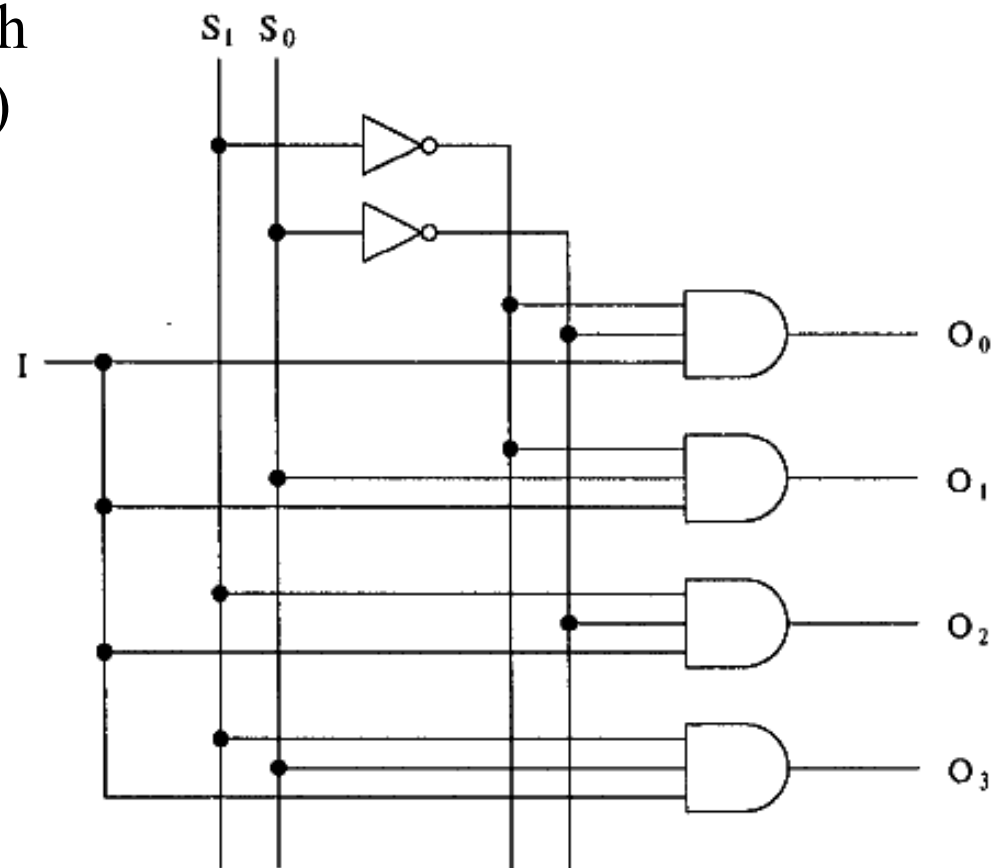
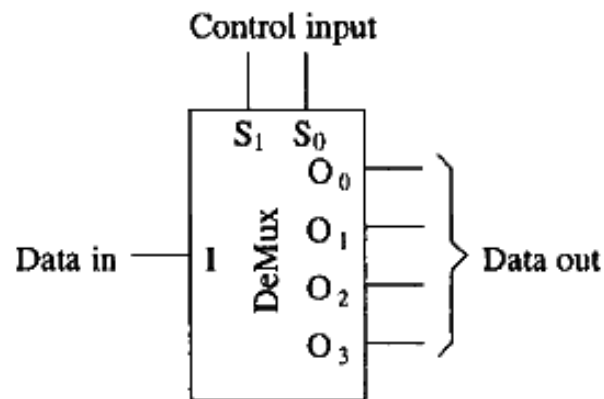
S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3



# Mạch tổ hợp

- Bộ phân kênh (Demultiplexer)

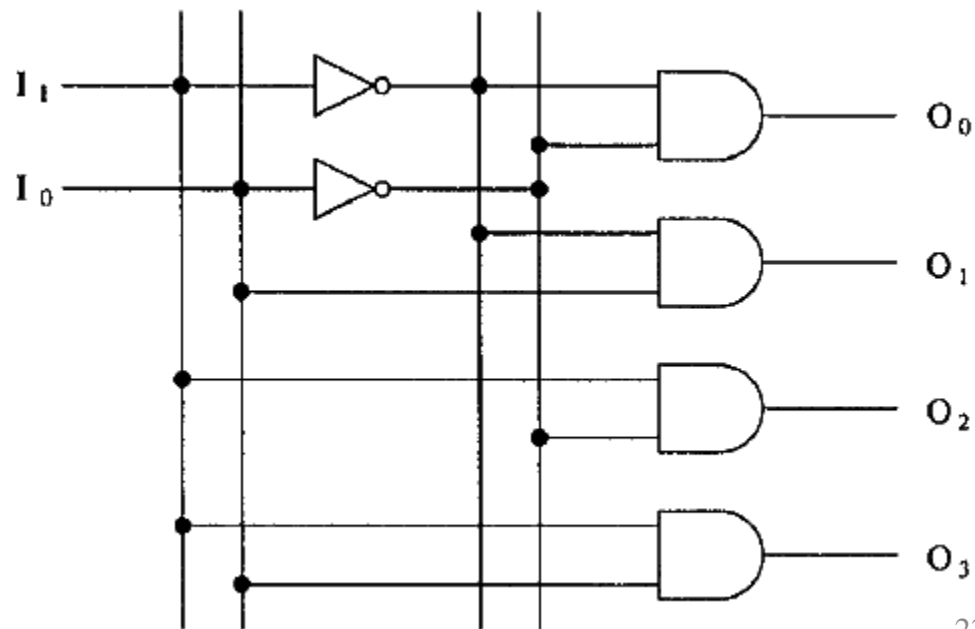
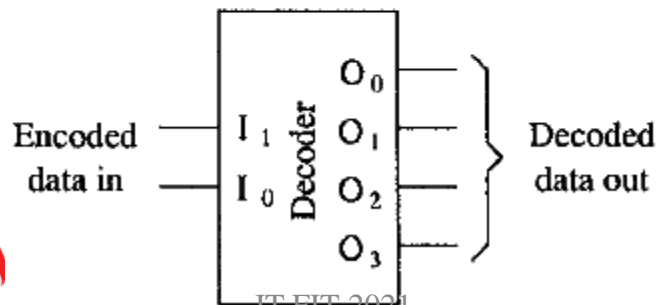
- Ngược với bộ dồn kênh
- Tín hiệu điều khiển (S) sẽ chọn đầu ra nào kết nối với đầu vào (I)
- Ví dụ: Demux 1-to-4



# Mạch tổ hợp

- Bộ giải mã (Decoder)
  - Bộ giải mã chọn một trong  $2^n$  đầu ra (O) tương ứng với một tổ hợp của n đầu vào (I)
  - Ví dụ : Mạch giải mã 2 ra 4

$I_1$	$I_0$	$O_3$	$O_2$	$O_1$	$O_0$
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

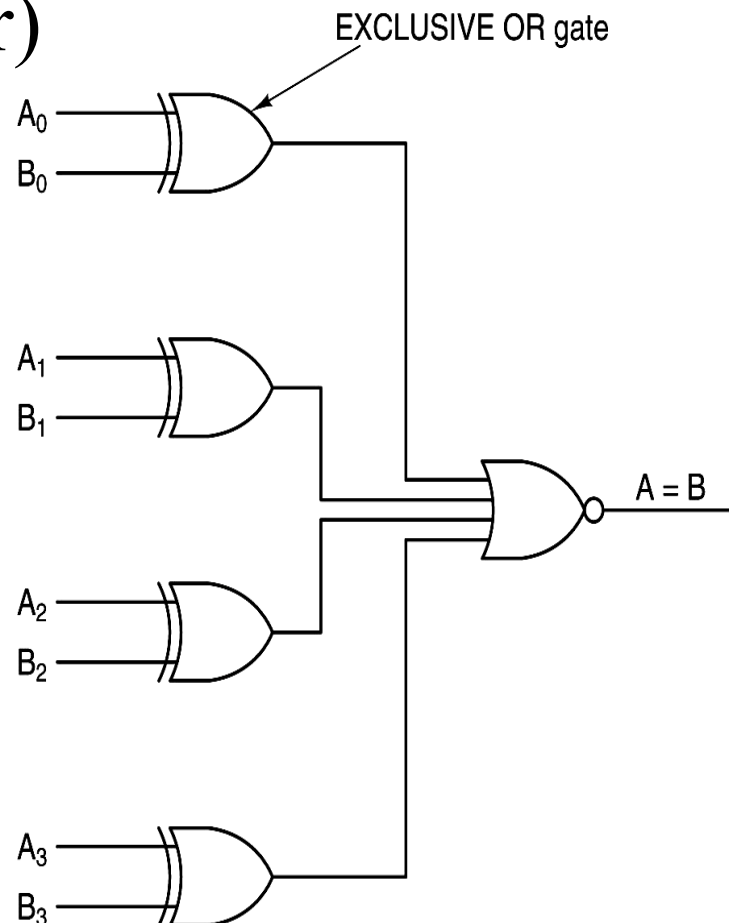


# Mạch tổ hợp

- Mạch so sánh (Comparator)

- So sánh các bit của 2 ngõ vào và xuất kết quả 1 nếu bằng nhau.
- Ví dụ : Mạch so sánh 4 bit dùng các cổng XOR

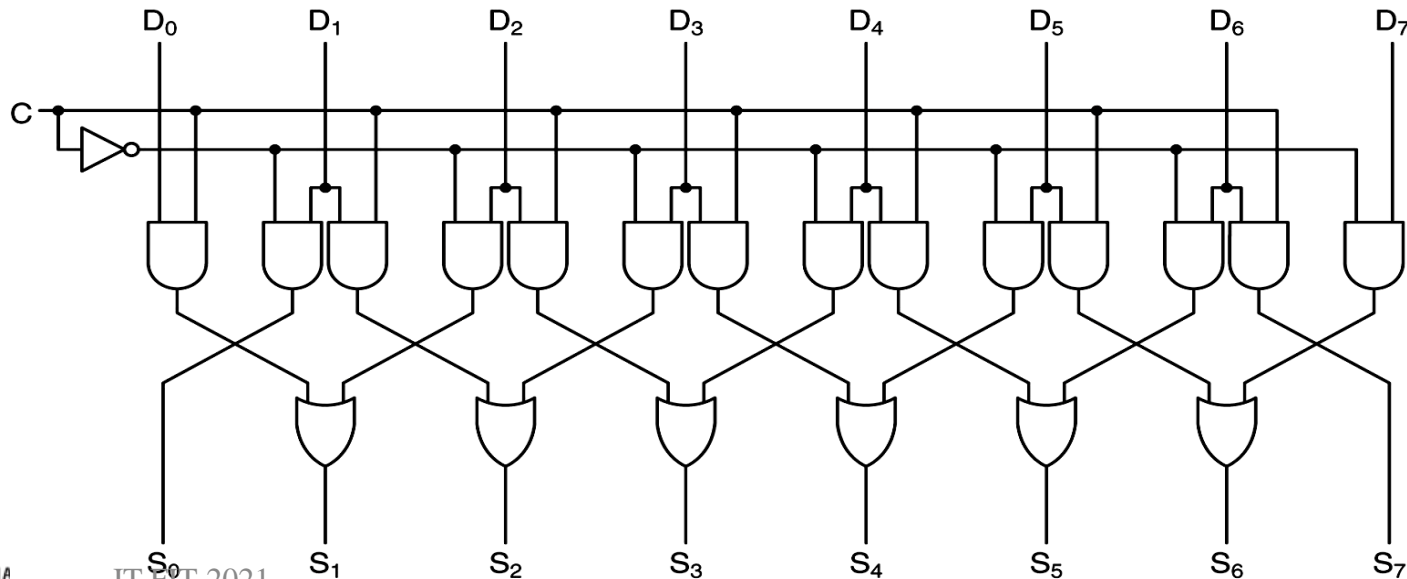
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0





# Mạch tính toán

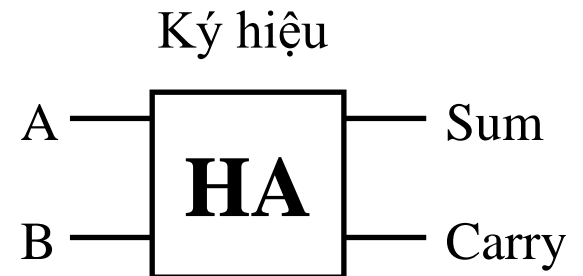
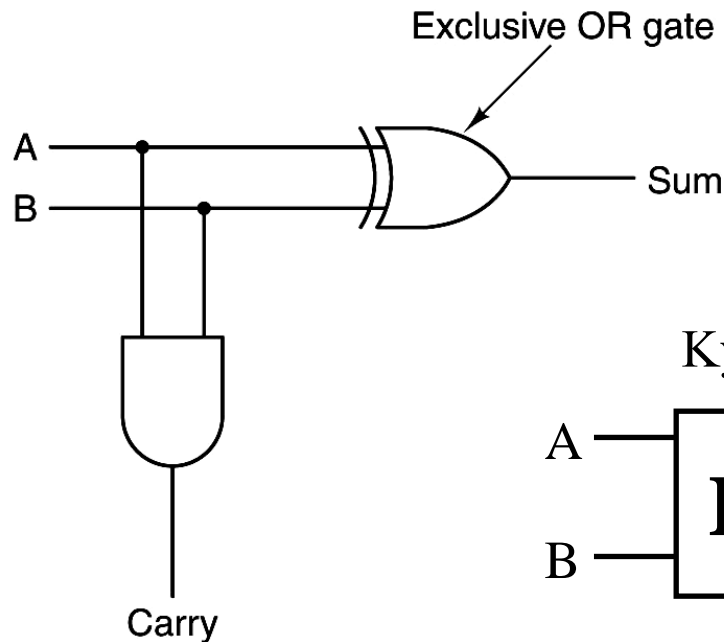
- Mạch dịch (Shifter)
  - Dịch các tín hiệu sang trái hoặc phải 1 vị trí. Ứng dụng cho phép nhân/ chia cho 2.
  - Ví dụ : mạch dịch 8 bit với tín hiệu điều khiển chiều dịch trái ( $C=0$ ) hay phải ( $C=1$ )



# Mạch tính toán

- Mạch cộng bán phần (Half adder)
  - Cộng 2 bit đầu vào thành 1 bit đầu ra và 1 bit nhớ

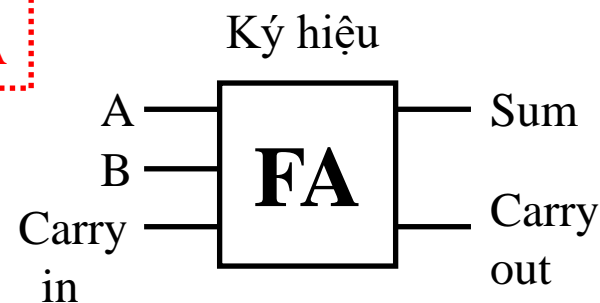
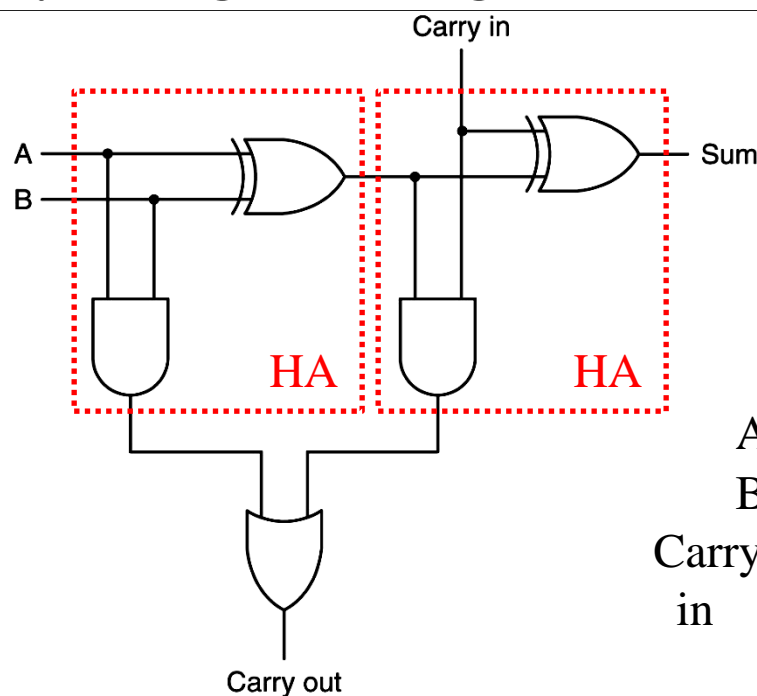
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



# Mạch tính toán

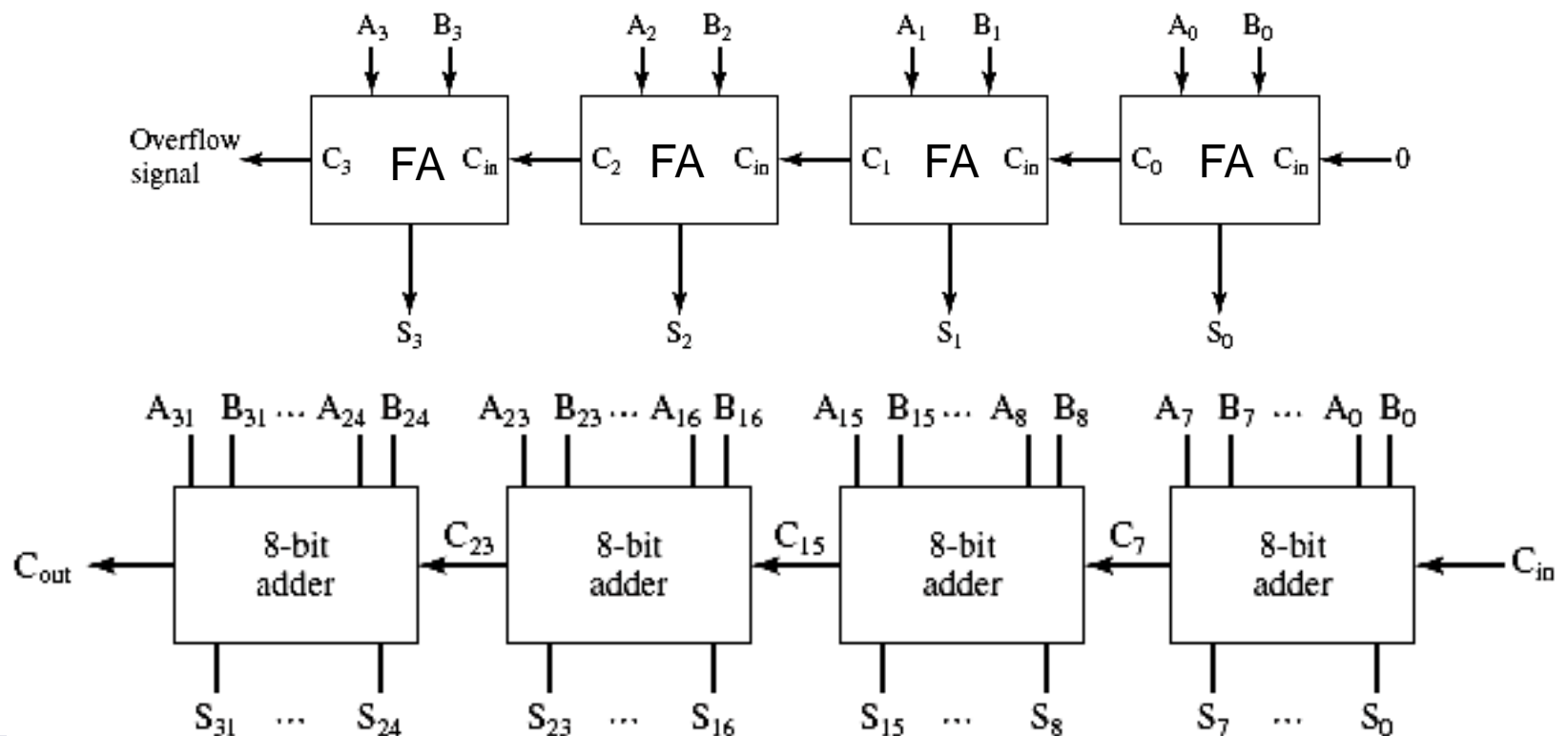
- Mạch cộng toàn phần (Full addder)
  - Cộng 3 bit đầu vào thành 1 bit đầu ra và 1 bit nhớ
  - Cho phép xây dựng bộ cộng nhiều bit

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



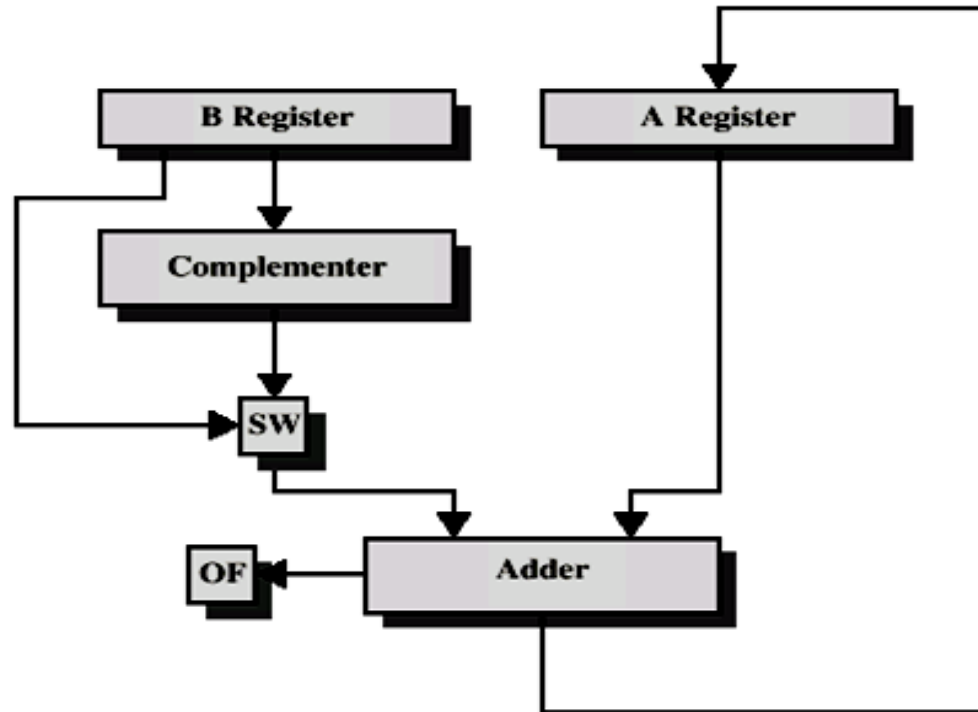
# Mạch tính toán

- Mạch cộng nhiều bit
  - Ghép từ nhiều bộ cộng toàn phần



# Mạch tính toán

- Mạch cộng và trừ
  - Mạch trừ: Đổi sang số bù 2 rồi cộng
  - Có thể dùng chung mạch cộng để thực hiện phép trừ



# Mạch tính toán

- Ví dụ ALU 1 bit

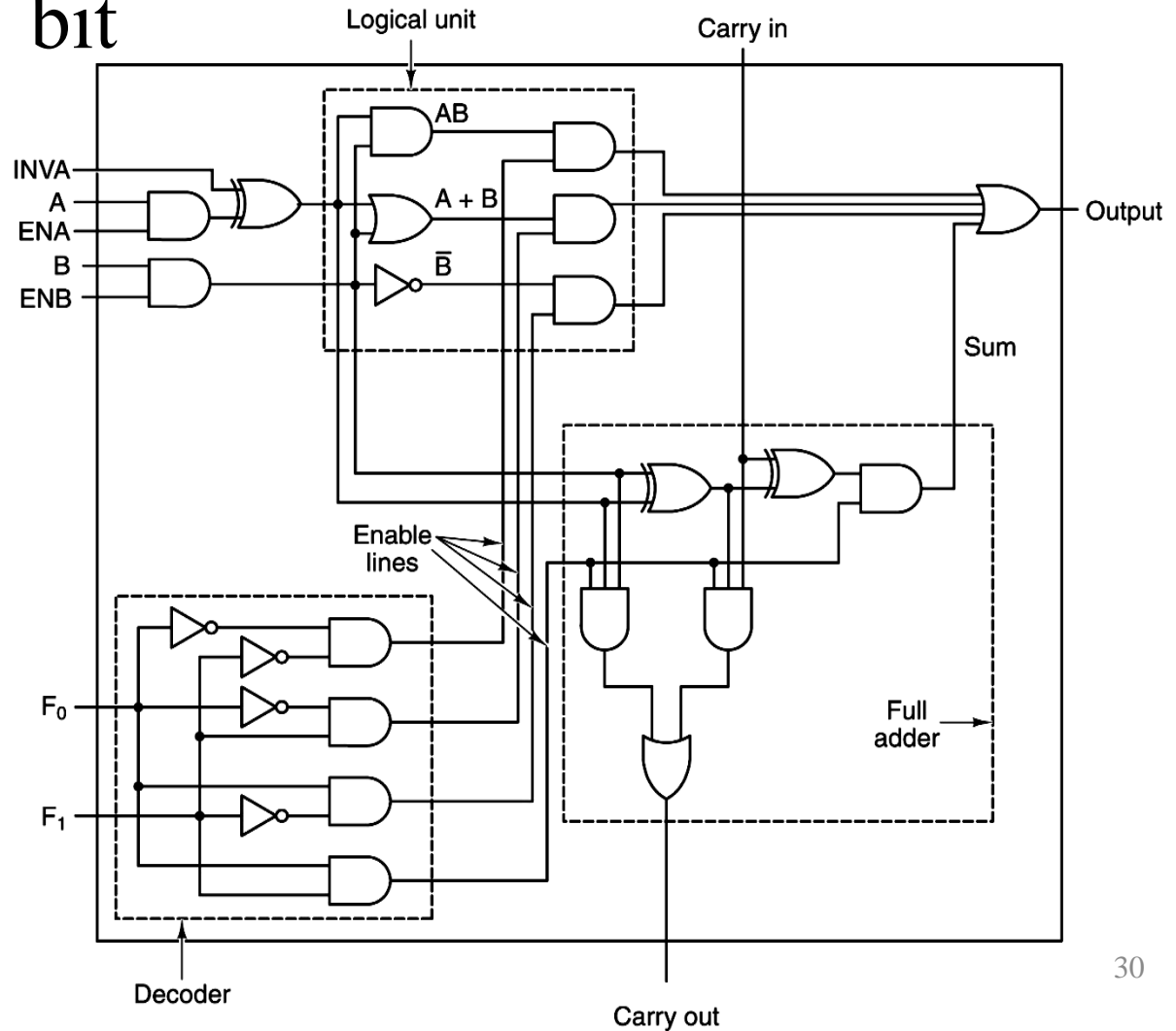
$F_0F_1$	Functions
00	<b>A AND B</b>
01	<b>A OR B</b>
10	<b><math>\bar{B}</math></b>
11	<b>A + B</b>

Điều kiện  
bình thường

ENA=1

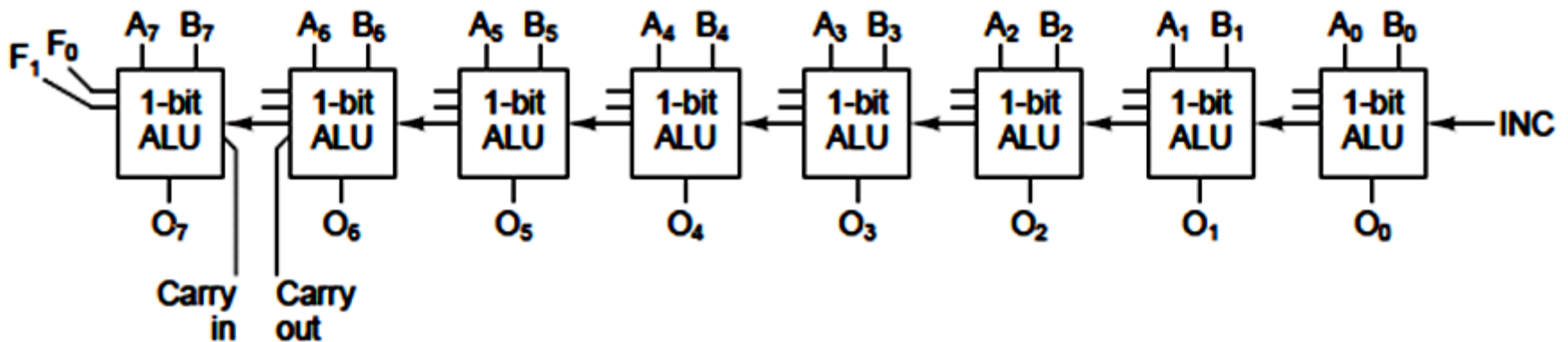
ENB=1

INVA=0



# Mạch tính toán

- ALU 8 bit
  - Ví dụ tạo 1 mạch ALU 8 bit bằng cách ghép 8 bộ ALU 1 bit ở ví dụ trước



# Mạch tuần tự

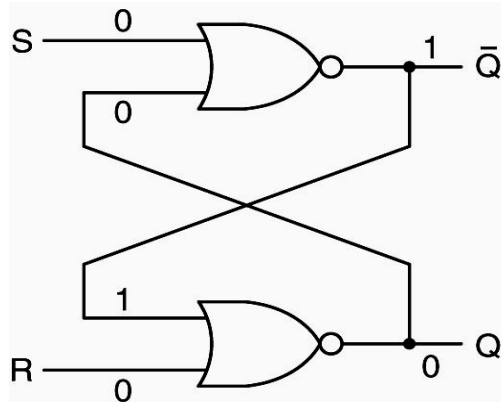
- Khái niệm
  - Mạch tuần tự (sequential circuit) là mạch logic trong đó tín hiệu ra phụ thuộc tín hiệu vào ở hiện tại và quá khứ
  - Là mạch có nhớ, được thực hiện bằng phần tử nhớ (Latch, Flip-Flop) và có thể kết hợp với các cổng logic cơ bản
  - Ứng dụng làm bộ nhớ, thanh ghi, mạch đếm,... trong máy tính



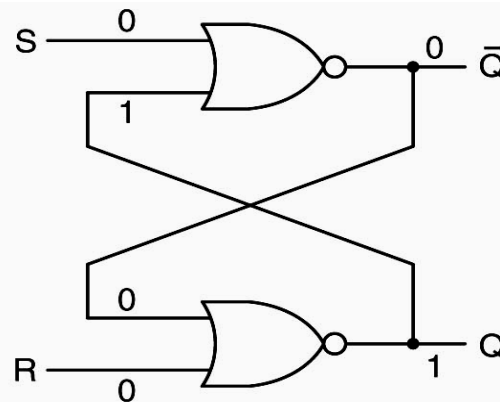
# Mạch tuần tự

- Mạch chốt (Latch)

- Dùng 2 cổng NOR mắc hồi tiếp với nhau. S, R là ngõ vào,  $\bar{Q}$  và Q là ngõ ra.
- Đây là mạch chốt SR. Nó có thể ở 1 trong 2 trạng thái  $Q=1$  hoặc  $Q=0$  khi  $S=R=0$ .
- Khi  $S=1 \rightarrow Q=1$  bất kể trạng thái trước đó (set)
- Khi  $R=1 \rightarrow Q=0$  bất kể trạng thái trước đó (reset)



IT-FIT (a)



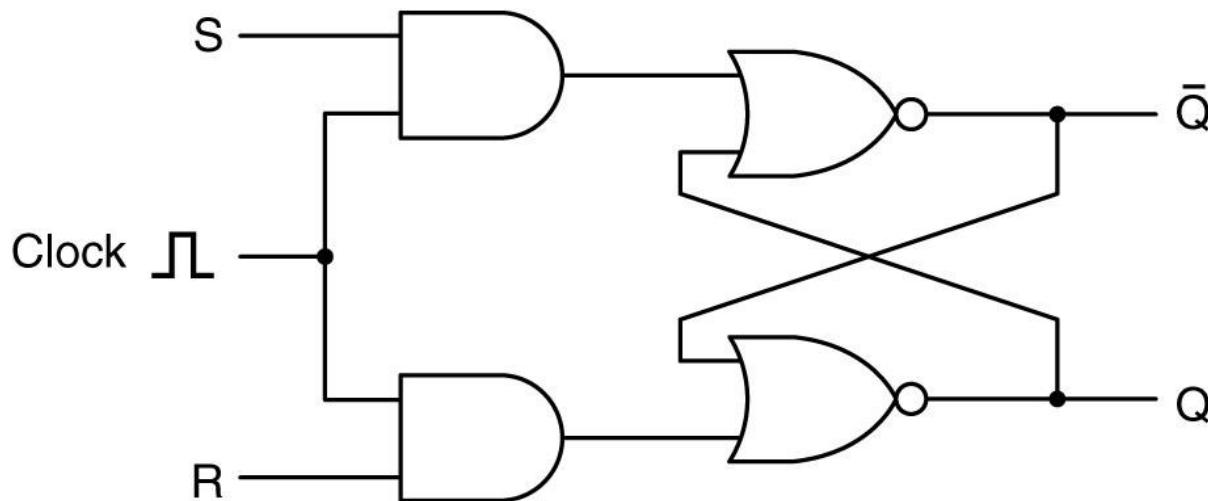
(b)

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

(c)

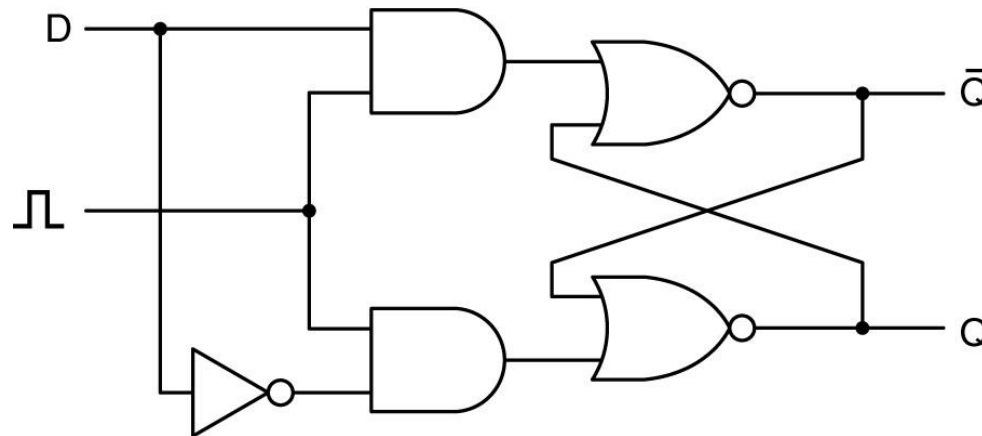
# Mạch tuần tự

- Mạch chốt SR có xung Clock
  - Thêm vào mạch chốt SR 2 cổng AND nối với xung đồng hồ để điều khiển trạng thái mạch chốt tại thời điểm xác định
  - Tín hiệu vào chỉ có tác dụng khi xung clock=1 (mức cao)



# Mạch tuần tự

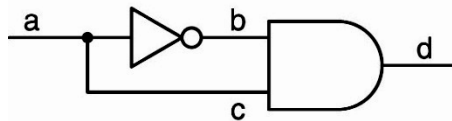
- Mạch chốt D có xung Clock
  - Mạch chốt SR sẽ ở trạng thái không xác định khi  $S=R=1$
  - Khắc phục bằng cách chỉ dùng 1 tín hiệu vào và đầu nối R với S qua cổng NOT
  - Đây chính là mạch bộ nhớ 1 bit với D là ngõ vào, Q là ngõ ra



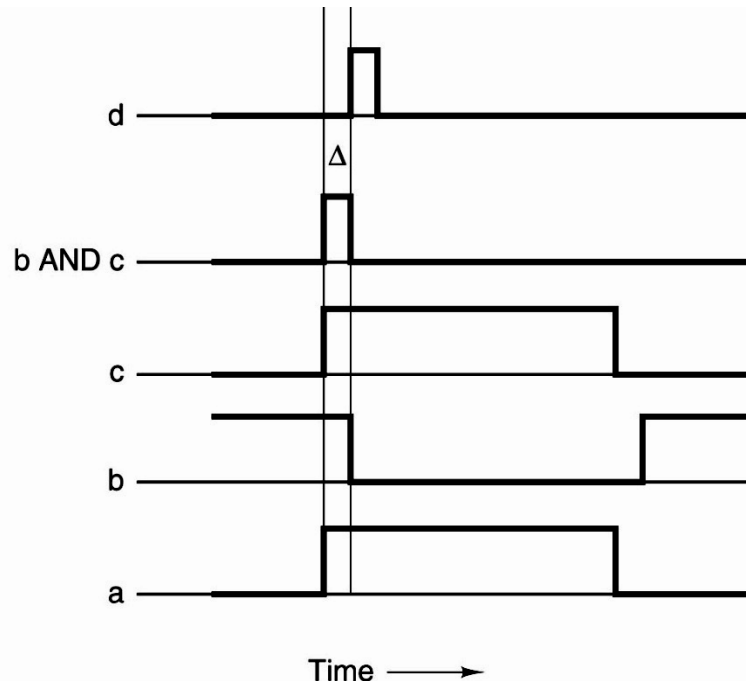
# Mạch bộ nhớ

- Flip-Flop

- Trong thực tế ta muốn bộ nhớ chỉ được ghi trong 1 khoảng thời gian nhất định → cần thiết kế mạch xung Clock tác dụng theo cạnh (lên hoặc xuống)



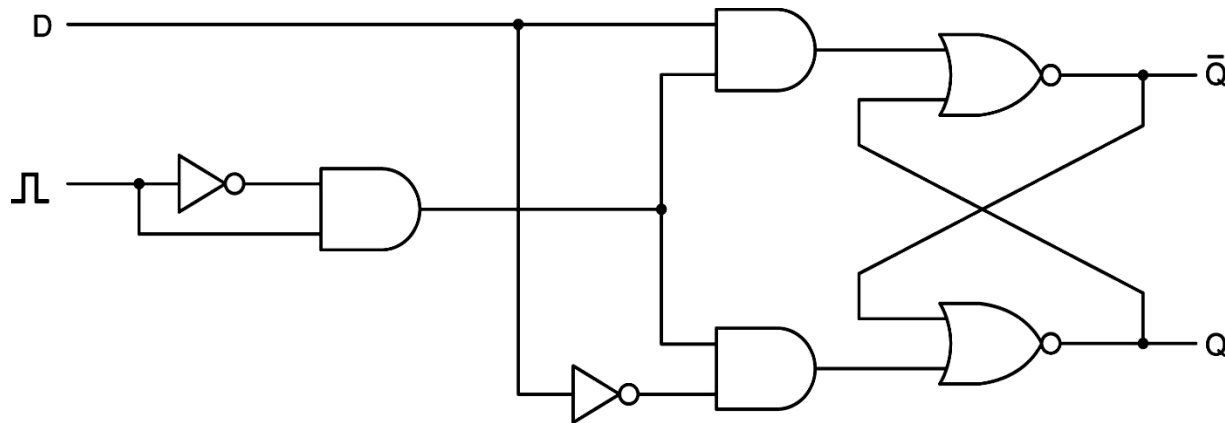
(a)



(b)

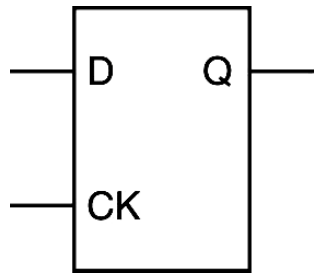
# Mạch bộ nhớ

- D Flip-Flop
  - Là mạch chốt D có xung Clock điều khiển bằng Flip-flop
  - Phân biệt:
    - Flip-flop: edge triggered
    - Latch: level triggered

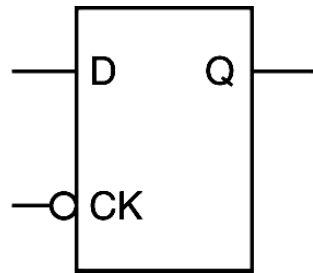


# Mạch bộ nhớ

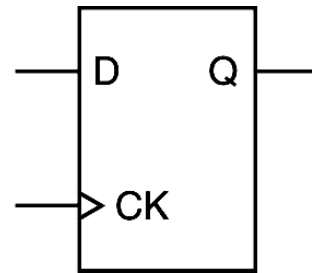
- Ký hiệu mạch chốt và Flip-Flop



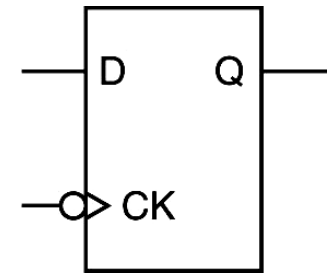
(a)



(b)



(c)

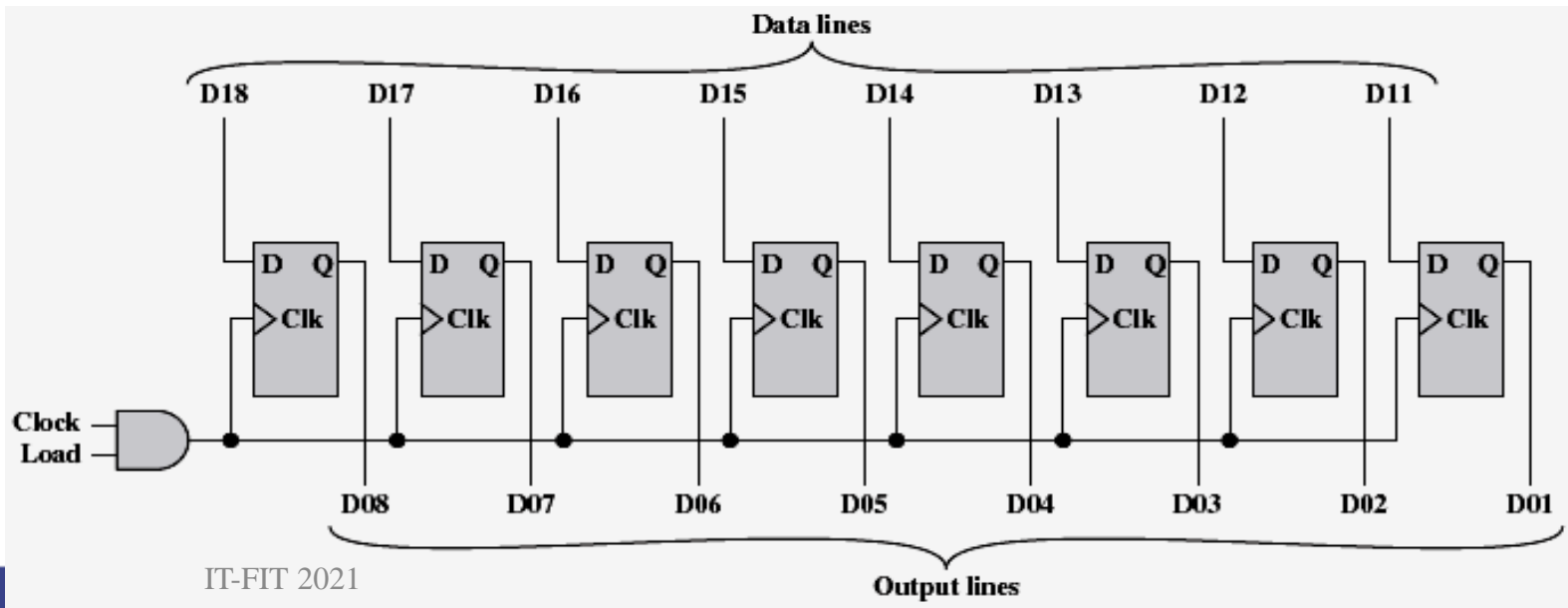
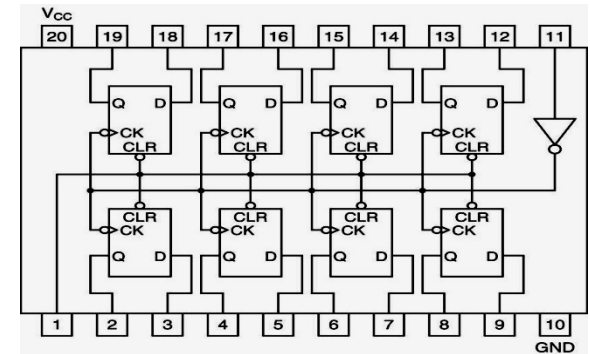


(d)

- a) Mạch chốt D tác động theo mức 1 (clock=1)
- b) Mạch chốt D tác động theo mức 0 (clock=0)
- c) Flip-flop D tác động theo cạnh lên (clock= 0 → 1)
- d) Flip-flop D tác động theo cạnh xuống (clock= 1 → 0)

# Mạch bộ nhớ

- Thanh ghi (Register)
  - Việc ghép nối nhiều ô nhớ 1 bit tạo thành các ô nhớ lớn hơn
  - Ví dụ : Vi mạch 74273 gồm 8 D flip-flop ghép nối lại tạo thành 1 thanh ghi 8 bit



# Mạch bộ nhớ

- Ví dụ : mạch bộ nhớ 4 ô x 3 bit

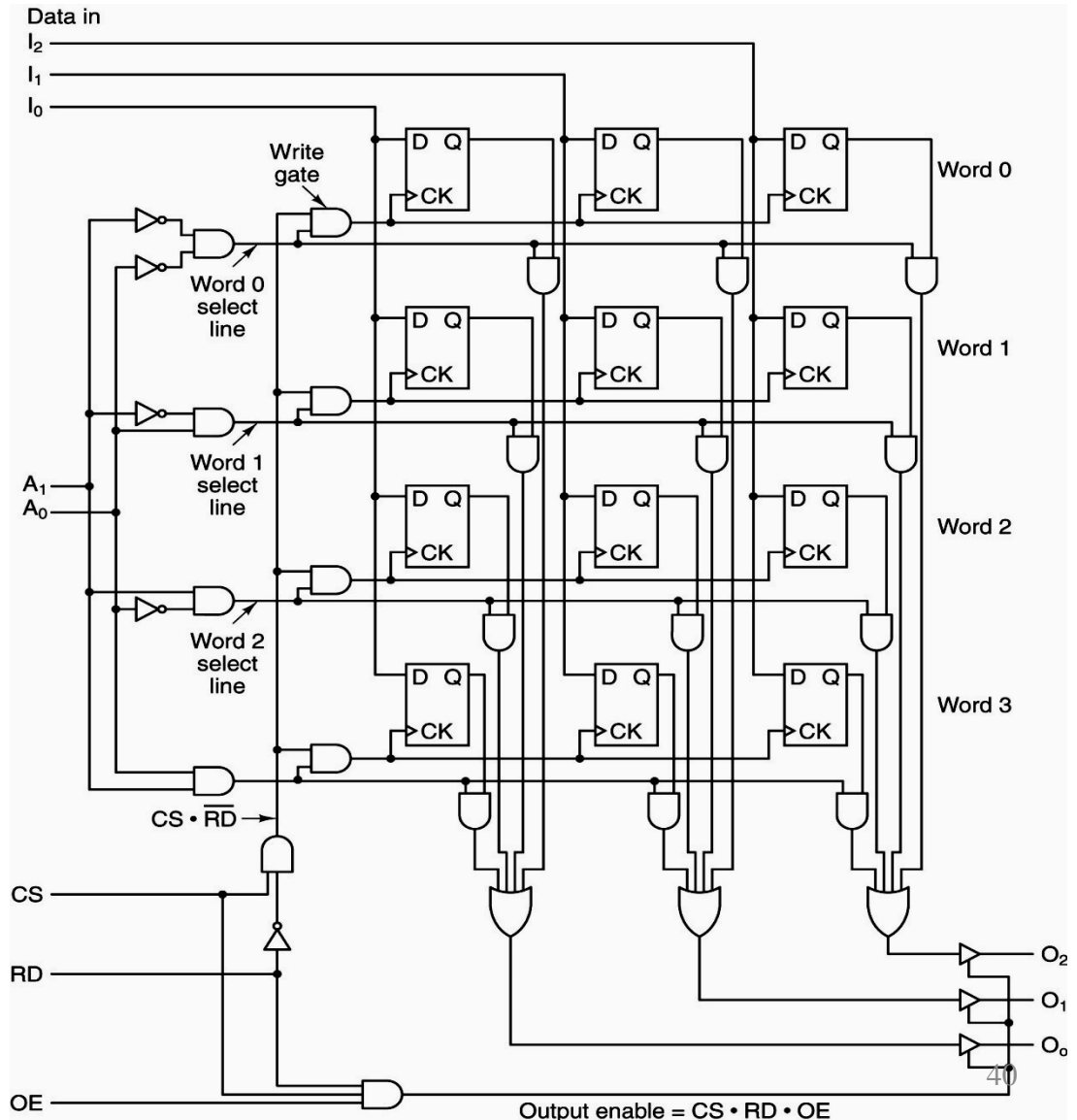
- A: Address
- I: Input data
- O: Output data
- CS: Chip select
- RD: Read/write
- OE: Output enable

Write:

CS=1, RD=0, OE=0

Read:

CS=1, RD=1, OE=1

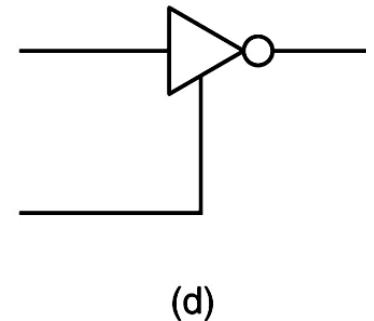
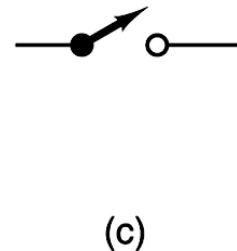
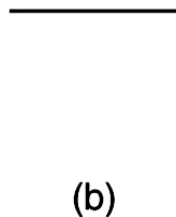
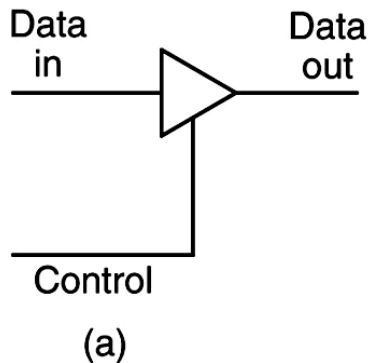




# Mạch bộ nhớ

- Mạch đệm (Buffer)

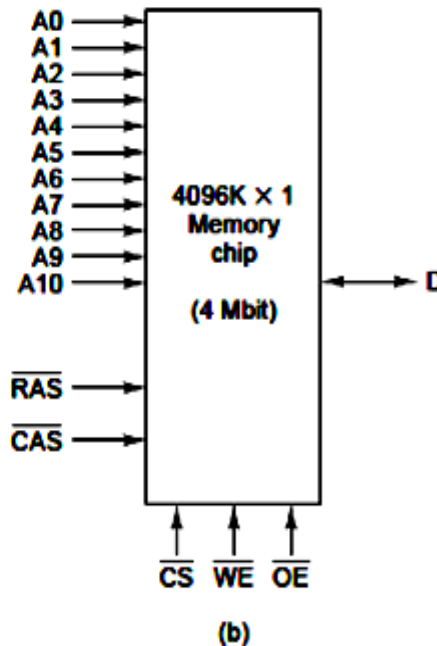
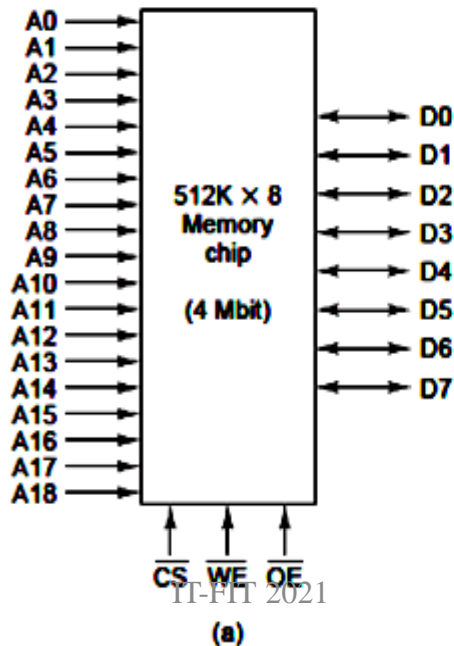
- Dùng để đọc dữ liệu đồng bộ trên nhiều đường tín hiệu bằng 1 đường điều khiển riêng.
- Sử dụng các cổng 3 trạng thái (tri-state devices)



- (a) Buffer không đảo.
- (b) Khi control ở mức cao (=1).
- (c) Khi control ở mức thấp (=0).
- (d) Buffer đảo.

# Mạch bộ nhớ

- Chip bộ nhớ
  - Bộ nhớ thường gồm nhiều ô nhớ ghép lại
  - Ví dụ 1: Chip bộ nhớ 4Mbit có thể tạo thành từ 512K ô 8 bit hoặc ma trận 2048x2048 ô 1 bit



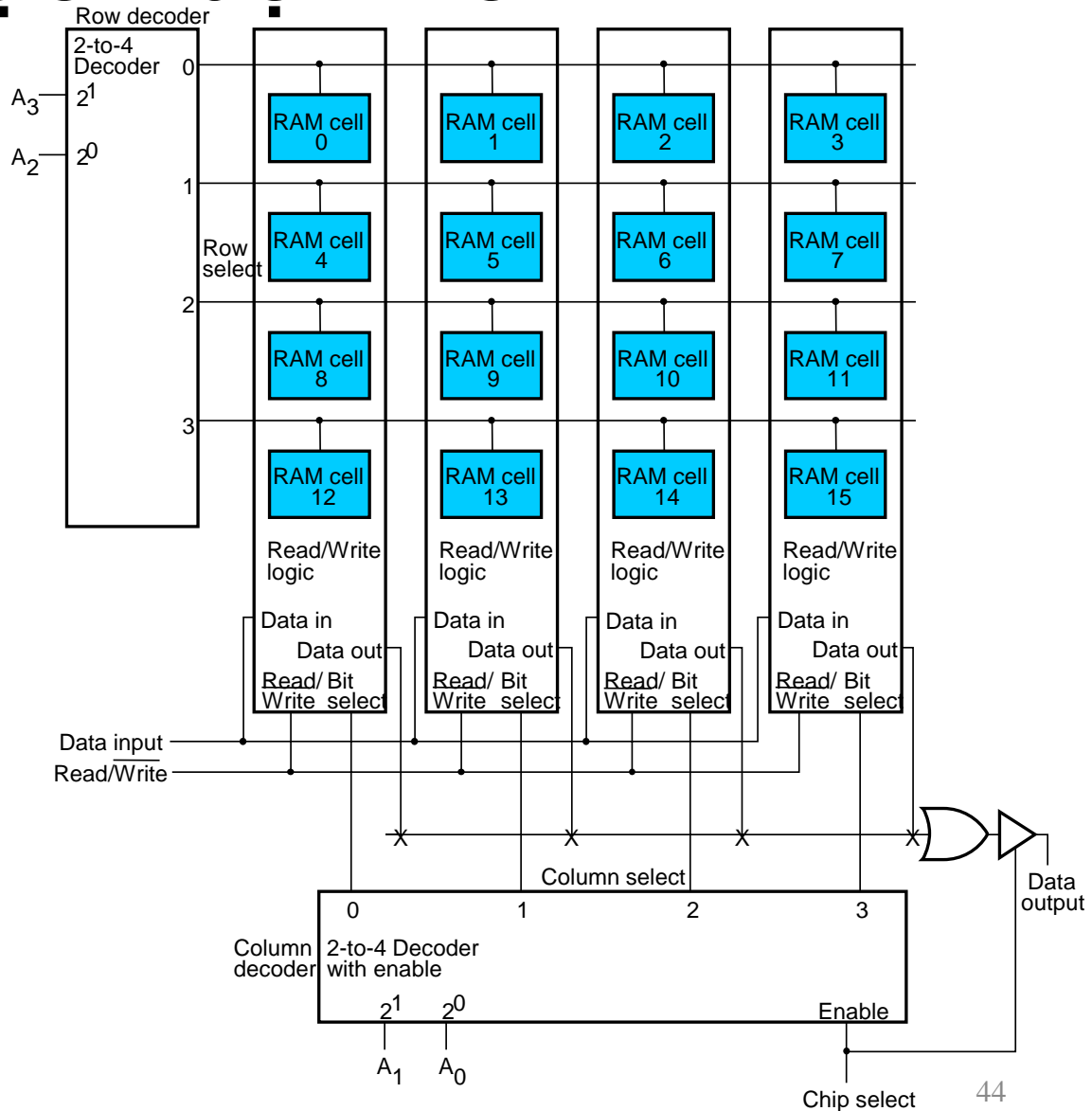
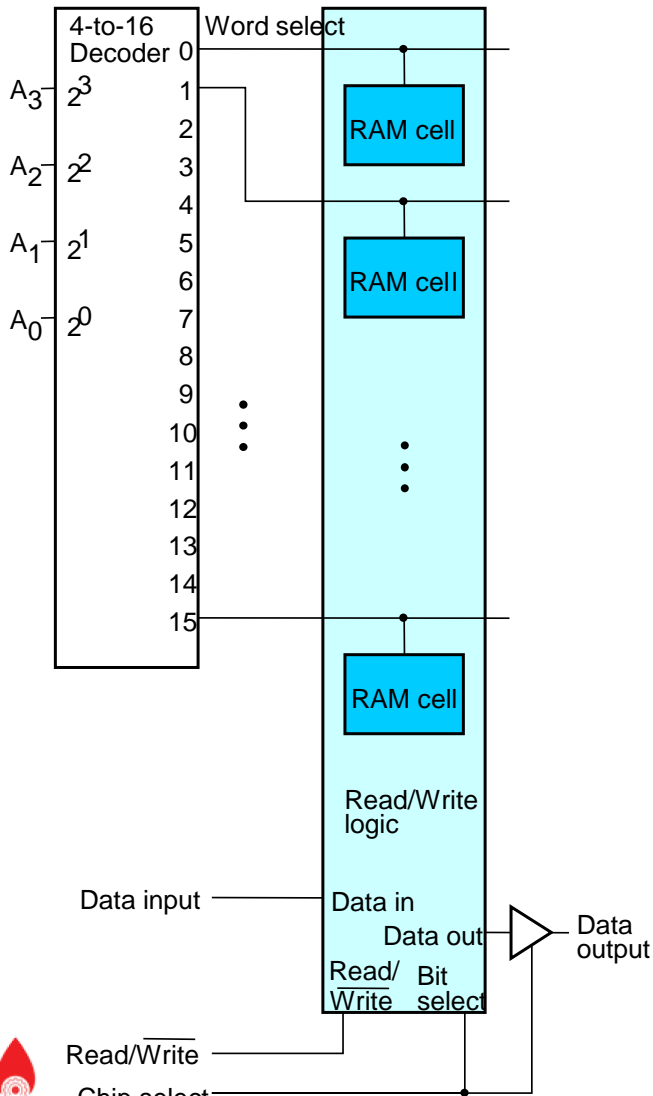
## Ghi chú:

RAS: Row Address Strobe  
CAS: Column Address Strobe  
CS: Chip select  
WE: Write enable  
OE: Output enable  
D: Data  
A: Address

# Mạch bộ nhớ

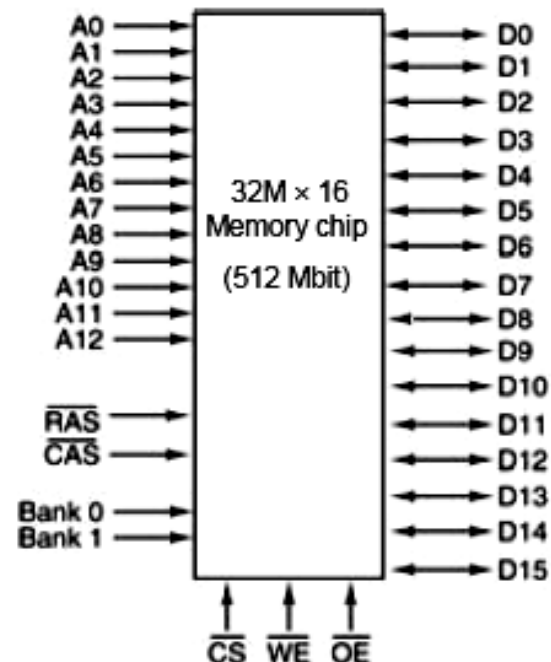
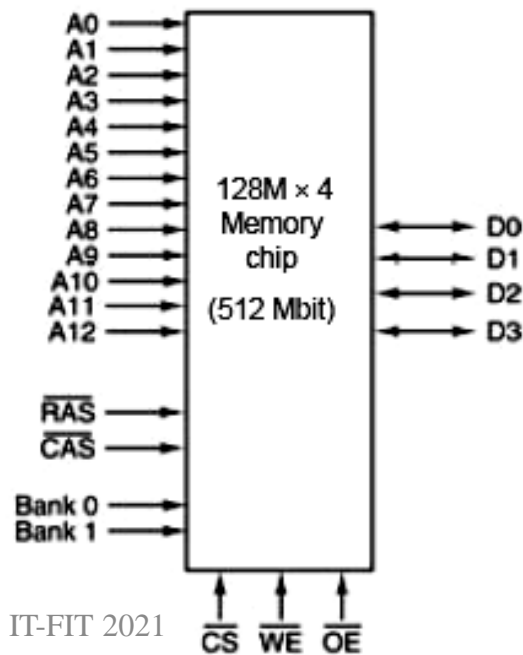
- Chip bộ nhớ (tiếp)
  - Mạch giải mã địa chỉ  $n$  bit có thể giải mã cho  $2^n$  ô nhớ  $\rightarrow$  cần  $n$  chân tín hiệu địa chỉ
  - Có thể giảm kích thước bộ giải mã còn  $\sqrt{n}$  bằng cách tổ chức thành ma trận các ô nhớ  $\rightarrow$  sử dụng 2 bộ giải mã cho hàng và cột riêng
  - Ví dụ: bộ nhớ 16 ô cần 4 bit địa chỉ có thể tổ chức thành ma trận  $4 \times 4 \rightarrow$  chỉ cần giải mã 2 bit cho hàng và 2 bit cho cột.
  - Có thể ghép địa chỉ hàng và cột chung 1 chân tín hiệu  $\rightarrow$  giảm số chân kết nối bus địa chỉ
  - Nhược điểm: cần gấp đôi thời gian truy cập bộ nhớ

# Mạch bộ nhớ



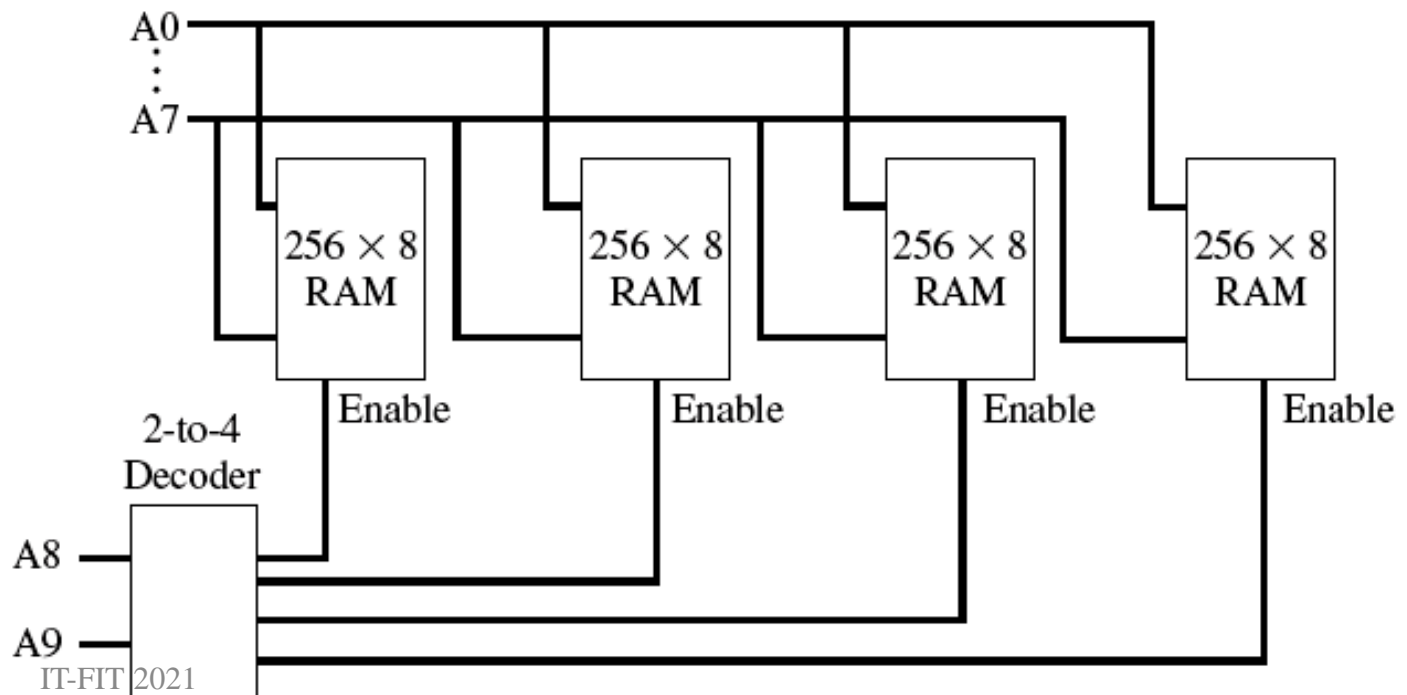
# Mạch bộ nhớ

- Chip bộ nhớ (tiếp)
  - Ví dụ 2: Chip bộ nhớ 512Mbit = 4 bank 128Mbit
    - Ma trận 13 bit hàng \* 12 bit cột \* ô nhớ 4 bit
    - Ma trận 13 bit hàng \* 10 bit cột \* ô nhớ 16 bit



# Mạch bộ nhớ

- Tổ chức bộ nhớ
  - Bộ nhớ thường gồm nhiều chip nhớ dung lượng nhỏ ghép lại
  - Dùng 1 mạch giải mã địa chỉ để chọn chip khi truy cập
  - Ví dụ: Bộ nhớ 1KB gồm 4 chip 256B ghép lại



# Câu hỏi

