



# Mạch logic


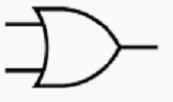
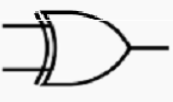
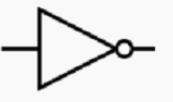

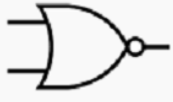

**Môn học: Kiến trúc máy tính & Hợp ngữ**

# Mạch số

- Là thiết bị điện tử hoạt động với 2 mức điện áp:
  - Cao: thể hiện bằng giá trị luận lý (quy ước) là 1
  - Thấp: thể hiện bằng giá trị luận lý (quy ước) là 0
- Được xây dựng từ những thành phần cơ bản là cổng luận lý (logic gate)
  - Cổng luận lý là thiết bị điện tử gồm 1 / nhiều tín hiệu đầu vào (input) - 1 tín hiệu đầu ra (output)
  - $\text{output} = F(\text{input}_1, \text{input}_2, \dots, \text{input}_n)$
  - Tùy thuộc vào cách xử lý của hàm F sẽ tạo ra nhiều loại cổng luận lý
- Hiện nay linh kiện cơ bản để tạo ra mạch số là transistor



# Cổng luận lý (Logic gate)

Tên cổng	Hình vẽ đại diện	Hàm đại số Bun
AND		$x.y$ hay $xy$
OR		$x + y$
XOR		$x \oplus y$
NOT		$x'$ hay $\bar{x}$
NAND		$(x.y)'$ hay $\overline{x.y}$
NOR		$(x + y)'$ hay $\overline{x + y}$
NXOR		$(x \oplus y)'$ hay $\overline{x \oplus y}$

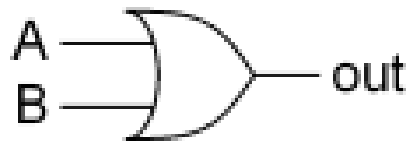
# Bảng chân trị

**AND**



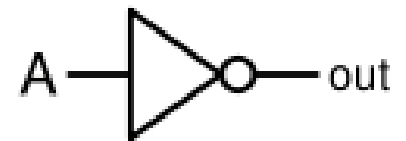
A	B	out
0	0	0
0	1	0
1	0	0
1	1	1

**OR**



A	B	out
0	0	0
0	1	1
1	0	1
1	1	1

**NOT**



A	out
0	1
1	0

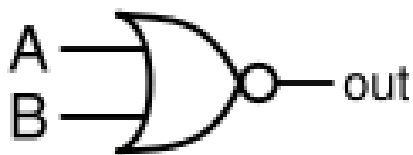
# Bảng chân trị

**NAND**



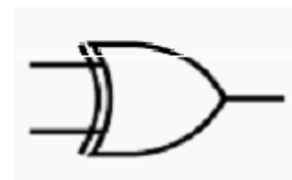
A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

**NOR**



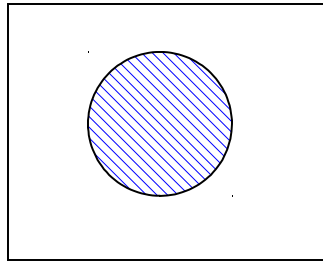
A	B	out
0	0	1
0	1	0
1	0	0
1	1	0

**XOR**

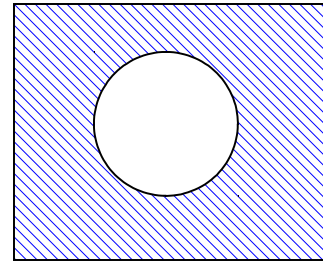


A	B	out
0	0	0
0	1	1
1	0	1
1	1	0

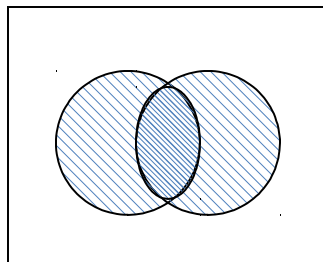
# Lược đồ Venn



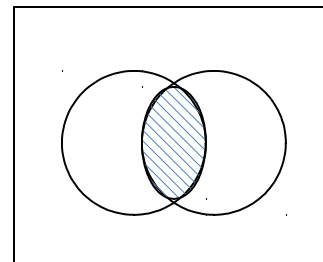
$A$



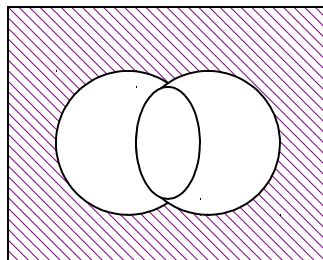
$\overline{A}$



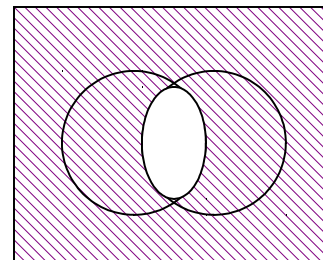
$A+B$



$A.B$

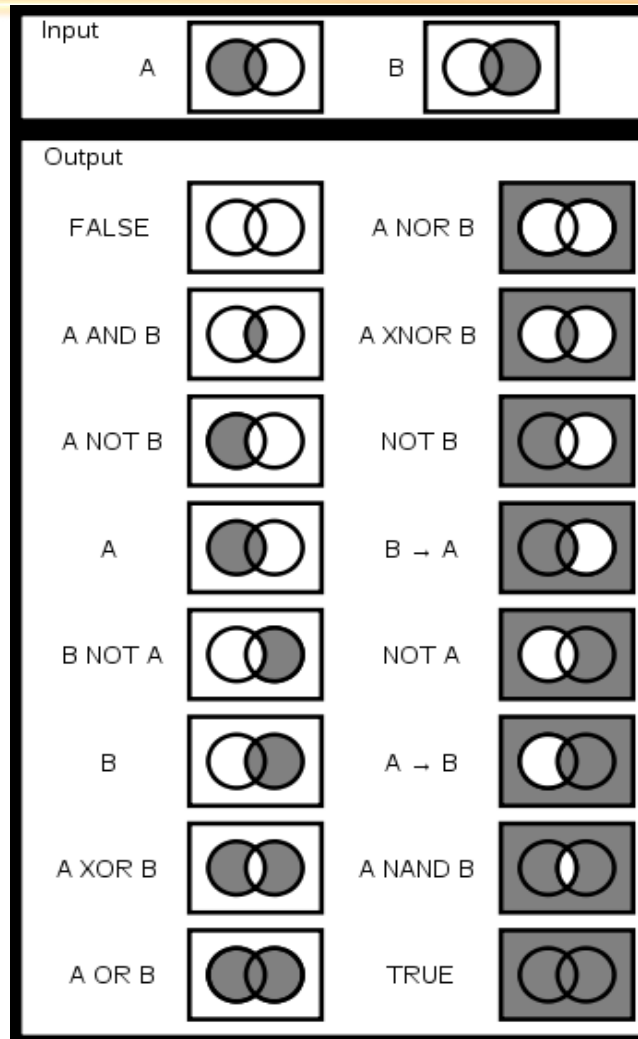


$\overline{A.B}$

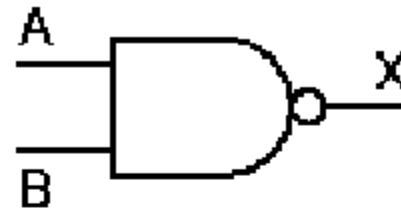
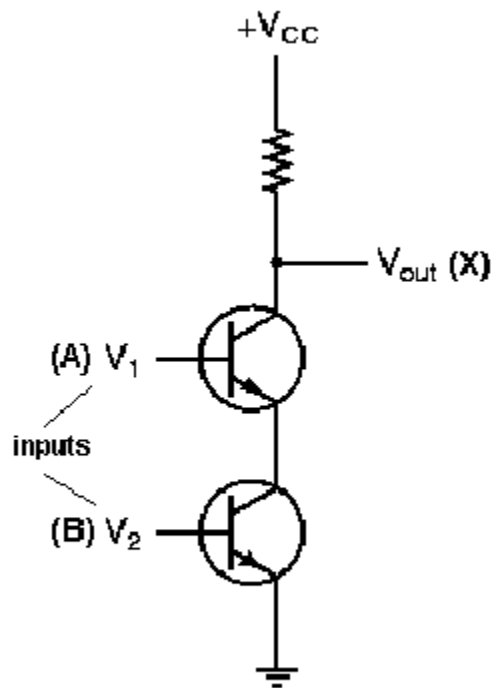


$\overline{A+B}$

# Lược đồ Venn



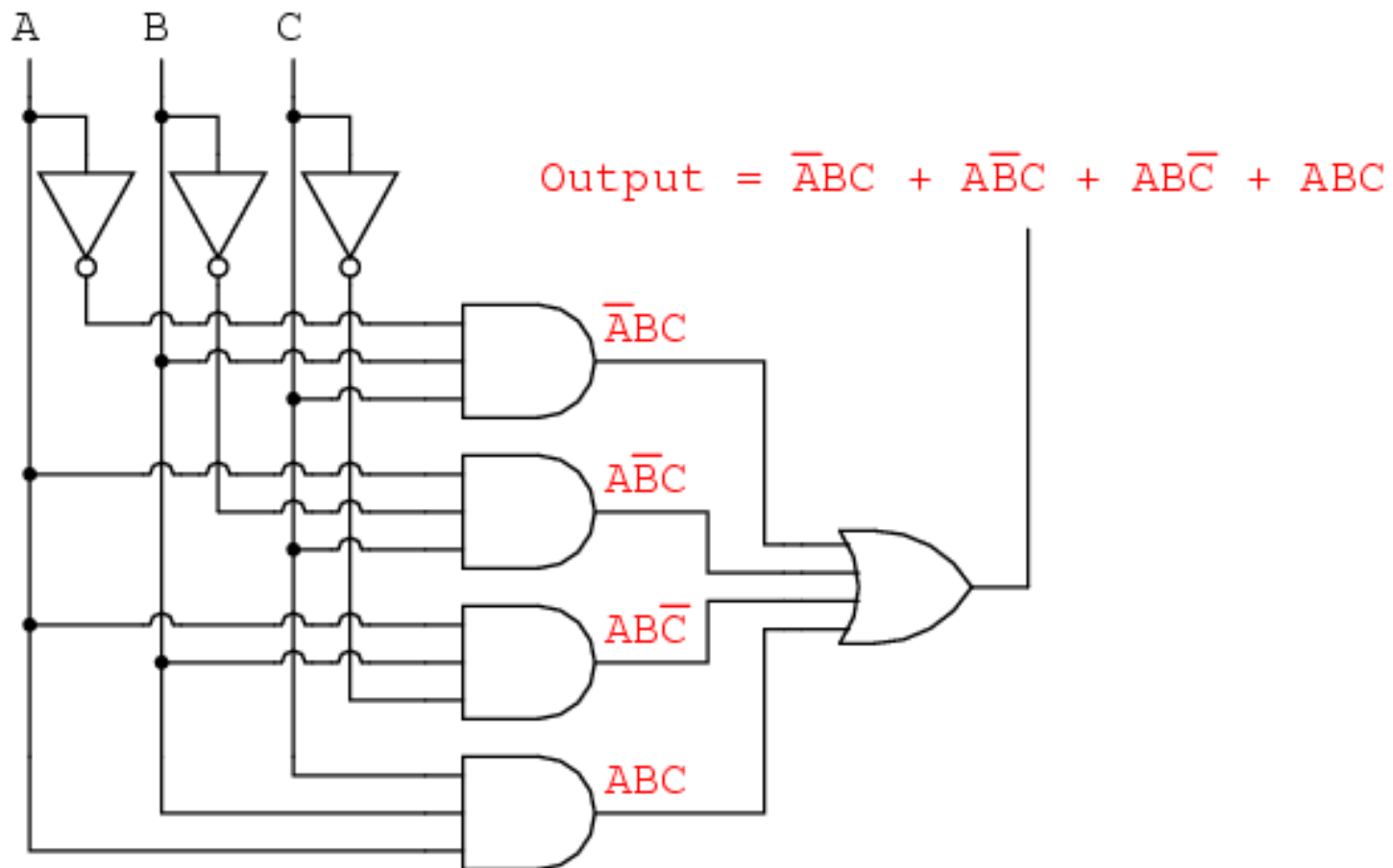
# Ví dụ công luận lý



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0



# Ví dụ mạch số

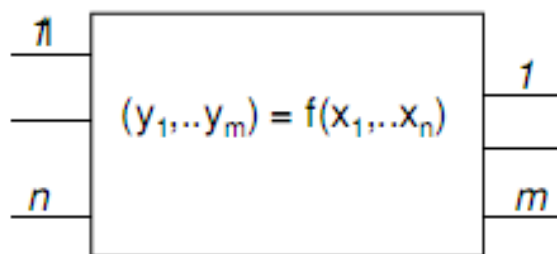


# Một số đẳng thức cơ bản

$x + 0 = x$	$x \cdot 0 = 0$
$x + 1 = 1$	$x \cdot 1 = x$
$x + x = x$	$x \cdot x = x$
$x + x' = 1$	$x \cdot x' = 0$
$x + y = y + x$	$xy = yx$
$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
$(x + y)' = x' \cdot y'$ (De Morgan)	$(xy)' = x' + y'$ (De Morgan)
$(x')' = x$	

# Mạch tổ hợp (tích hợp)

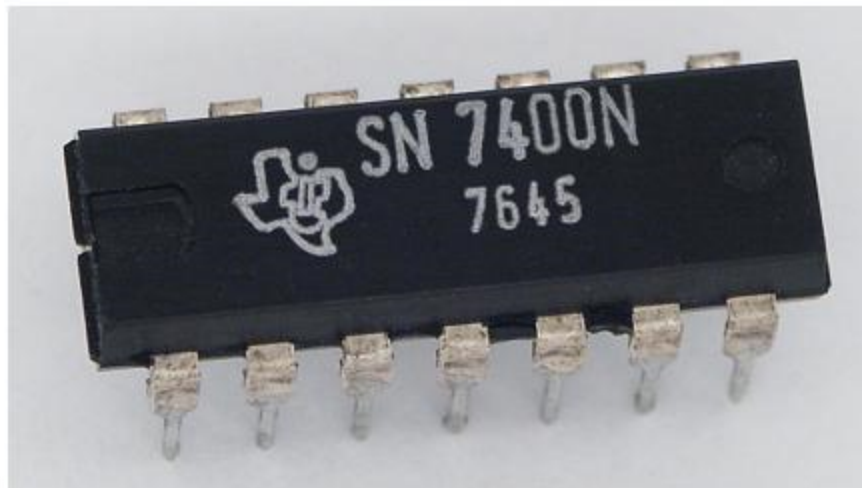
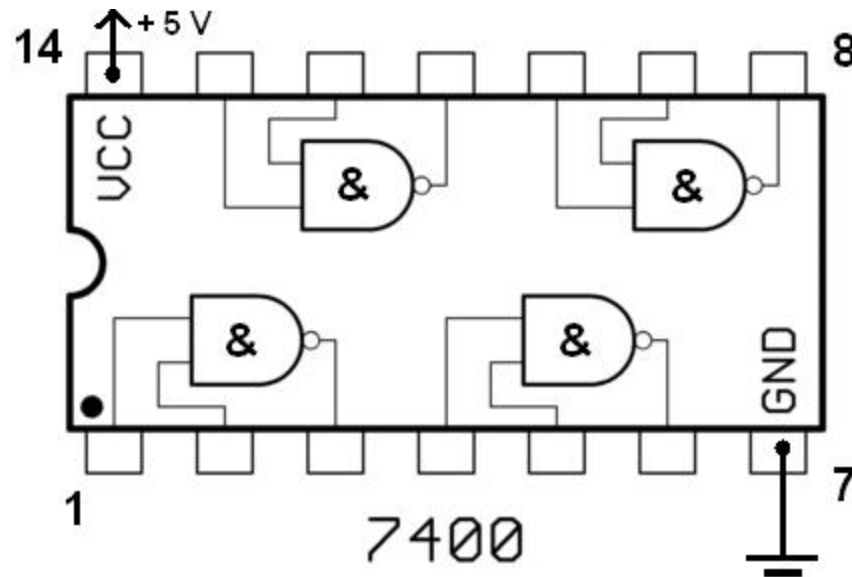
- Gồm  $n$  ngõ vào (input);  $m$  ngõ ra (output)
  - Mỗi ngõ ra là 1 hàm luận lý của các ngõ vào



- Mạch tổ hợp không mang tính ghi nhớ:  
Ngõ ra chỉ phụ thuộc vào Ngõ vào hiện tại,  
không xét những giá trị trong quá khứ



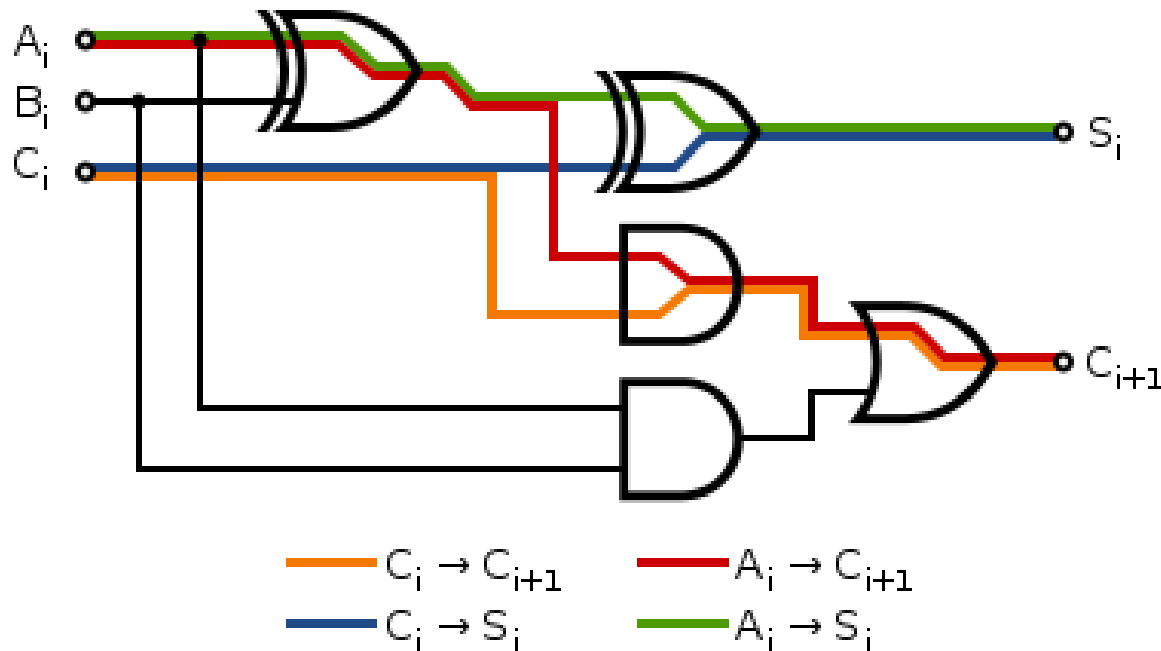
# Ví dụ mạch tổ hợp



- The 7400 chip, containing four NAND gate
- The two additional pins supply power (+5 V) and connect the ground.

# Độ trễ mạch

- Độ trễ mạch (**Propagation delay / gate delay**) = Thời điểm tín hiệu ra ổn định - thời điểm tín hiệu vào ổn định
  - Mục tiêu thiết kế mạch: làm giảm thời gian độ trễ mạch



# Mô tả mạch tổ hợp

- Bảng ngôn ngữ
- Bảng bảng chân trị
  - $n$  input –  $m$  output
  - $2^n$  hàng –  $(n + m)$  cột
- Bảng công thức (hàm luận lý)
- Bảng sơ đồ



# Thiết kế

- Thường trải qua 3 bước:

- Lập bảng chân trị

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

- Viết hàm luận lý

$$F = (AB)'$$

- Vẽ sơ đồ mạch và thử nghiệm



# SOP – Sum of Products

- Giả sử đã có bảng chân trị cho mạch  $n$  đầu vào  $x_1, \dots, x_n$  và 1 đầu ra  $f$
- Ta dễ dàng thiết lập công thức (hàm) logic theo thuật toán sau:
  - Ứng với mỗi hàng của bảng chân trị có đầu ra = 1 ta tạo thành 1 tích có dạng  $u_1 \cdot u_2 \dots u_n$  với:
$$u_i = \begin{cases} x_i & \text{nếu } x_i = 1 \\ (x_i)' & \text{nếu } x_i = 0 \end{cases}$$
  - Cộng các tích tìm được lại thành tổng  $\rightarrow$  công thức của  $f$





# Ví dụ

$x_1$	$x_2$	$x_3$	$f$	
0	0	0	0	
0	0	1	1	$\rightarrow \bar{x}_1 \cdot \bar{x}_2 \cdot x_3$
0	1	0	1	$\rightarrow \bar{x}_1 \cdot x_2 \cdot \bar{x}_3$
0	1	1	0	
1	0	0	0	
1	0	1	1	$\rightarrow x_1 \cdot \bar{x}_2 \cdot x_3$
1	1	0	0	
1	1	1	0	

$$f = \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 x_3$$

# POS – Product of Sum

- Trường hợp số hàng có giá trị **đầu ra = 1**  
**nhều hơn = 0**, ta có thể đặt  $g = (f)'$
- Viết công thức dạng **SOP** cho  $g$
- Lấy  $f = (g)' = (f')'$  để có công thức dạng POS (Tích các tổng) của  $f$



# Ví dụ

x	y	z	f	g
0	0	0	1	0
0	0	1	1	0
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	0

$$g = \bar{x}.y.\bar{z} + x.\bar{y}.\bar{z}$$

$$f = \bar{g} = (x + \bar{y} + z)(\bar{x} + y + z)$$

# Đơn giản hoá hàm logic

- Sau khi viết được hàm logic, ta có thể vẽ sơ đồ của mạch tổ hợp từ những cộng luận lý cơ bản
  - Ví dụ:  $f = xy + xz$
- Tuy nhiên ta có thể viết lại hàm logic sao cho sơ đồ mạch sử dụng **ít cổng hơn**
  - Ví dụ:  $f = xy + xz = x(y + z)$
- Cách đơn giản hoá hàm tổng quát? Một số cách phổ biến:
  - Dùng đại số Bun (Xem lại bảng 1 số đẳng thức cơ bản để áp dụng)
  - Dùng bản đồ Karnaugh (Cac-nô)



# Đại số Bun

- Dùng các phép biến đổi đại số Bun để lược giản hàm logic
- Khuyết điểm:
  - Không có cách làm tổng quát cho mọi bài toán
  - Không chắc kết quả cuối cùng đã tối giản chưa
- Ví dụ: Đơn giản hoá các hàm sau
  - $F(x,y,z) = xyz + x'yz + xy'z + xyz'$



# Bản đồ Karnaugh

- Mỗi tổ hợp biến trong bảng chân trị gọi là bộ trị (tạm hiểu là 1 dòng)
  - Biểu diễn hàm có  $n$  biến thì sẽ cho ra tương ứng  $2^n$  bộ trị, với vị trí các bộ trị được đánh số từ 0
  - Thông tin trong bảng chân trị có thể **cô đọng** bằng cách:
    - Liệt kê vị trí các bộ trị (**minterm**) với giá trị đầu ra = **1** (**SOP**)
    - Liệt kê vị trí các bộ trị (**maxterm**) với giá trị đầu ra = **0** (**POS**)



# Ví dụ

- $F(x,y,z) = m_1 + m_4 + m_5 + m_6 + m_7 = \Sigma(1,4,5,6,7)$
- $F(x,y,z) = M_0 M_2 M_3 = \Pi(0,2,3)$

Vị trí	x	y	z	minterm	maxterm	F
0	0	0	0	$m0 = x'y'z'$	$M0 = x + y + z$	0
1	0	0	1	$m1 = x'y'z$	$M1 = x + y + z'$	1
2	0	1	0	$m2 = x'yz'$	$M2 = x + y' + z$	0
3	0	1	1	$m3 = x'yz$	$M3 = x + y' + z'$	0
4	1	0	0	$m4 = xy'z'$	$M4 = x' + y + z$	1
5	1	0	1	$m5 = xy'z$	$M5 = x' + y + z'$	1
6	1	1	0	$m6 = xyz'$	$M6 = x' + y' + z$	1
7	1	1	1	$m7 = xyz$	$M7 = x' + y' + z'$	1

# Các dạng bản đồ Karnaugh cơ bản

24

		B	
A \	B	0	1
	0	0	1
A {	1	2	3

		B			
A \ BC	00	01	11	10	
0	0	1	3	2	
1	4	5	7	6	

<div>AB \ CD</div>		C			
		00	01	11	10
A	00	0	1	3	2
	01	4	5	7	6
	11	12	13	15	14
	10	8	9	11	10
		D			





# Ví dụ

- $F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$

		<div style="display: flex; justify-content: space-around;"> <span>BC</span> <span>00</span> <span>01</span> <span>11</span> <span>10</span> </div>			
A	0	0	1	0	0
	1	1	1	1	1

C

=

		<div style="display: flex; justify-content: space-around;"> <span>BC</span> <span>00</span> <span>01</span> <span>11</span> <span>10</span> </div>			
A	0		1		
	1	1	1	1	1

C

# Nhận xét

- Bộ trị giữa 2 ô liền kề trong bản đồ chỉ khác nhau 1 biến
  - Biến đó bù 1 ô, không bù ở ô kế hoặc ngược lại
- Các ô đầu / cuối của các dòng / cột là các ô liền kề
- 4 ô nằm ở 4 góc bản đồ cũng coi là ô liền kề



# Đơn giản hàm theo dạng SOP

- Hàm logic F biểu diễn bảng chân trị được đưa vào bản đồ bằng các trị 1 tương ứng
- Các ô liên kề có giá trị 1 được gom thành nhóm sao cho mỗi nhóm sau khi gom có tổng số ô là lũy thừa của 2 (2, 4, 8,...)
- Các nhóm có thể dùng chung ô có giá trị 1 để tạo thành nhóm lớn hơn. Cố gắng tạo những nhóm lớn nhất có thể
- Nhóm 2/4/8 ô sẽ đơn giản bớt 1/2/3 biến trong số hạng
- Mỗi nhóm biểu diễn 1 số hạng nhân (Product), Cộng (Sum – OR)  
các số hạng này ta sẽ được biểu thức tối giản của hàm logic F



# Ví dụ 1

- $F(A, B, C) = \Sigma(3, 4, 6, 7)$

A \ BC	B			
	00	01	11	10
0			1	
1	1	1	1	1

C

A \ BC	B			
	00	01	11	10
0			1	
1	1		1	1

C

$$F(A, B, C) = BC + AC'$$

# Ví dụ 2

- $F(A, B, C) = \Sigma(0, 2, 4, 5, 6)$

		B			
		00	01	11	10
A	0	1			1
	1	1	1		1

Diagram illustrating the Karnaugh map for the function  $F(A, B, C) = \Sigma(0, 2, 4, 5, 6)$ . The map is a 2x4 grid with variables A and B. The columns are labeled 00, 01, 11, and 10. The rows are labeled 0 and 1. The values in the cells are 1, 0, 1, 1, 0, 1, 0, 1. A bracket labeled C is shown under the columns 01 and 11, indicating the group  $C'$ . A bracket labeled A is shown under the rows 0 and 1, indicating the group  $AB'$ .

		B			
		00	01	11	10
A	0	1			1
	1	1	1		1

Diagram illustrating the Karnaugh map for the function  $F(A, B, C) = \Sigma(0, 2, 4, 5, 6)$ . The map is a 2x4 grid with variables A and B. The columns are labeled 00, 01, 11, and 10. The rows are labeled 0 and 1. The values in the cells are 1, 0, 1, 1, 0, 1, 0, 1. A bracket labeled C is shown under the columns 01 and 11, indicating the group  $C'$ . A bracket labeled A is shown under the rows 0 and 1, indicating the group  $AB'$ . The cells (0,0), (1,0), (0,1), and (1,1) are highlighted in yellow, representing the group  $AB'$ . The cells (1,0) and (1,1) are highlighted in green, representing the group  $C'$ .

$$F(A, B, C) = C' + AB'$$

# Ví dụ

- $F(A, B, C, D) = \Sigma(0, 1, 2, 6, 8, 9, 10)$

AB \ CD		C			
		00	01	11	10
A	00	1	1		1
	01				1
	11				
	10	1	1		1

Brackets in the original image indicate groupings: a vertical bracket on the right labeled 'B' groups rows 00, 01, 11, and 10; a horizontal bracket at the bottom labeled 'D' groups columns 00, 01, and 10.

AB \ CD		C			
		00	01	11	10
A	00	1	1		1
	01				1
	11				
	10	1	1		1

Brackets in the original image indicate groupings: a vertical bracket on the right labeled 'B' groups rows 00, 01, 11, and 10; a horizontal bracket at the bottom labeled 'D' groups columns 00, 01, and 10. In this version, the 1s in the 00 and 10 rows for columns 00 and 01 are highlighted with yellow boxes, and the 1s in the 00 and 01 columns for rows 00 and 10 are highlighted with blue boxes. The 1 in row 01, column 10 is highlighted with a green box.

$$F(A, B, C, D) = B'D' + B'C' + A'CD'$$

# Đơn giản hàm theo dạng POS

- Đôi khi biểu diễn dạng tổng các tích (SOP) sẽ khó làm khi **số bộ trị có đầu ra = 1 < số bộ trị có đầu ra = 0**

→ Dùng phương pháp tích các tổng (POS)

- Hoàn toàn giống phương pháp đơn giản hàm theo dạng SOP, chỉ khác ta **nhóm các ô liền kề = 0** thay vì 1

→ Tìm được  $F'$

→  $F = (F')'$



# Ví dụ 3

- $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$

AB \ CD	C			
	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

Groupings:   
 - **A** (Red bracket on the left): Rows 11 and 10.  
 - **B** (Yellow bracket on the right): Rows 01, 11, and 10.  
 - **D** (Green bracket at the bottom): Columns 00, 01, and 10.

AB \ CD	C			
	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

Groupings:   
 - **A** (Red bracket on the left): Rows 11 and 10.  
 - **B** (Yellow bracket on the right): Rows 01, 11, and 10.  
 - **D** (Green bracket at the bottom): Columns 00, 01, and 10.  
 - **CD** (Yellow vertical bar): Column 11.  
 - **BD'** (Orange vertical bars): Columns 00 and 01 for rows 01 and 11.  
 - **AB** (Green horizontal bar): Row 11 for columns 00 and 01.

$$F'(A, B, C) = CD + BD' + AB$$

$$F = (F')' = (A' + B')(C' + D')(B' + D)$$



# Điều kiện không cần / tùy chọn

- Trong 1 số trường hợp ta **không cần quan tâm** đến giá trị ngõ ra của 1 số bộ trị nào đó (**1 hay 0 đều được**)
- Trong bản đồ ta sẽ ghi tương ứng những ô đó là x (gọi là giá trị tùy chọn /không cần)
- **x có thể dùng để gom nhóm với các ô liền kề nhằm đơn giản hàm**
- Lưu ý: Không được gom nhóm bao gồm toàn những ô có giá trị x



# Ví dụ

- $F(A, B, C) = \Sigma(0, 2, 6)$
- $d(A, B, C) = \Sigma(1, 3, 5)$

Vị trí	A	B	C	F
0	0	0	0	1
1	0	0	1	x
2	0	1	0	1
3	0	1	1	x
4	1	0	0	0
5	1	0	1	x
6	1	1	0	1
7	1	1	1	0



# Ví dụ

□  $F(A, B, C) = \Sigma(0, 2, 6)$

□  $d(A, B, C) = \Sigma(1, 3, 5)$

BC \ A		B			
		00	01	11	10
A	0	1	x	x	1
	1		x		1

C

BC \ A		B			
		00	01	11	10
A	0	1	x	x	1
	1		x		1

C

$$F(A, B, C) = A' + BC'$$

# Bài tập thiết kế mạch tổ hợp

- Yêu cầu: Thiết kế mạch tổ hợp 3 ngõ vào, 1 ngõ ra, sao cho giá trị logic ở ngõ ra là giá trị nào chiếm đa số trong các ngõ vào



# Bước 1: Lập bảng chân trị

- Gọi các ngõ vào là  $x, y, z$  - ngõ ra là  $f$

x	y	z	f
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

$$f(x, y, z) = \Sigma(3, 5, 6, 7)$$



# Bước 2: Viết hàm logic

□  $f(x, y, z) = \Sigma(3, 5, 6, 7)$

$\begin{array}{c} yz \\ \backslash x \end{array}$		$y$			
		00	01	11	10
0				1	
1			1	1	1

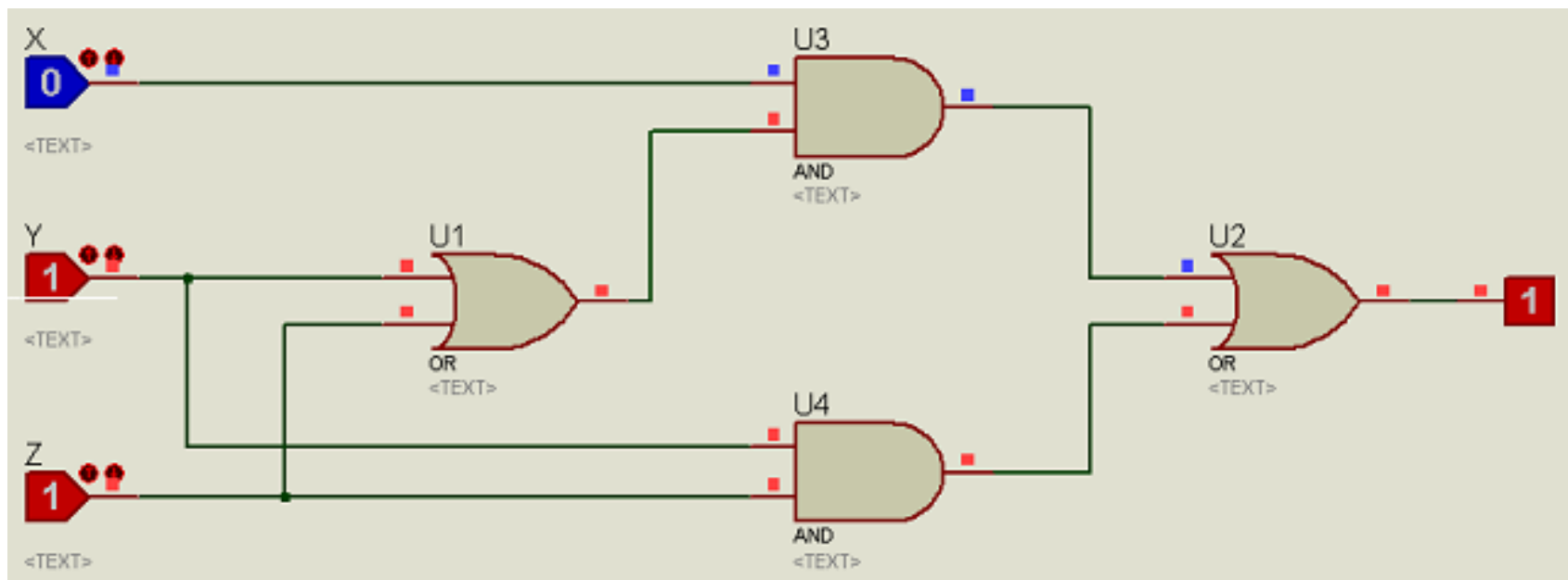
$z$

$\begin{array}{c} yz \\ \backslash x \end{array}$		$y$			
		00	01	11	10
0				1	
1			1	1	1

$z$

$$f(x, y, z) = xz + xy + yz = x.(y+z) + yz$$

# Bước 3: Vẽ sơ đồ mạch và Thử nghiệm



## Phần 2: Một số mạch tổ hợp cơ bản

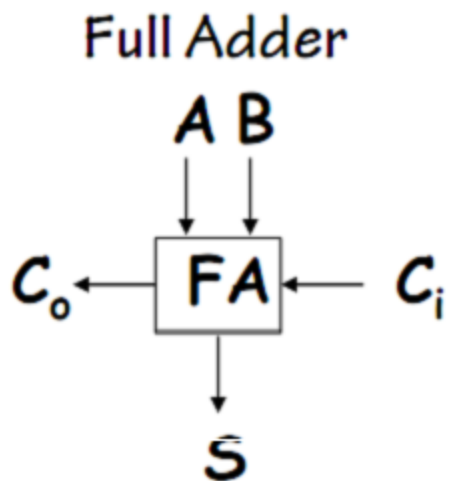
- Mạch toàn cộng (Full adder)
- Mạch giải mã (Decoder)
- Mạch mã hoá (Encoder)





# Mạch toàn cộng (Full adder - FA)

- Mạch tổ hợp thực hiện phép **cộng số học 3 bit**
- Gồm **3 ngõ vào** (A, B: bit cần cộng –  $C_i$ : bit nhớ) và **2 ngõ ra** (kết quả có thể từ 0 đến 3 với giá trị 2 và 3 cần 2 bit biểu diễn – S: ngõ tổng,  $C_o$ : ngõ nhớ)



A	B	$C_i$	S	$C_o$
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$S = F(A, B, C_i) \\ = \Sigma(1, 2, 4, 7)$$

$$C_o = F(A, B, C_i) \\ = \Sigma(3, 5, 6, 7)$$

# Bước 2: Viết hàm logic

		A			
		AB			
Ci	0	00	01	11	10
	1	00	01	11	10
		B			

$$S = F(A, B, Ci) = \Sigma(1, 2, 4, 7)$$

$$S = A'BCi' + AB'Ci' + A'B'Ci + ABCi$$

$$S = A \oplus B \oplus Ci$$

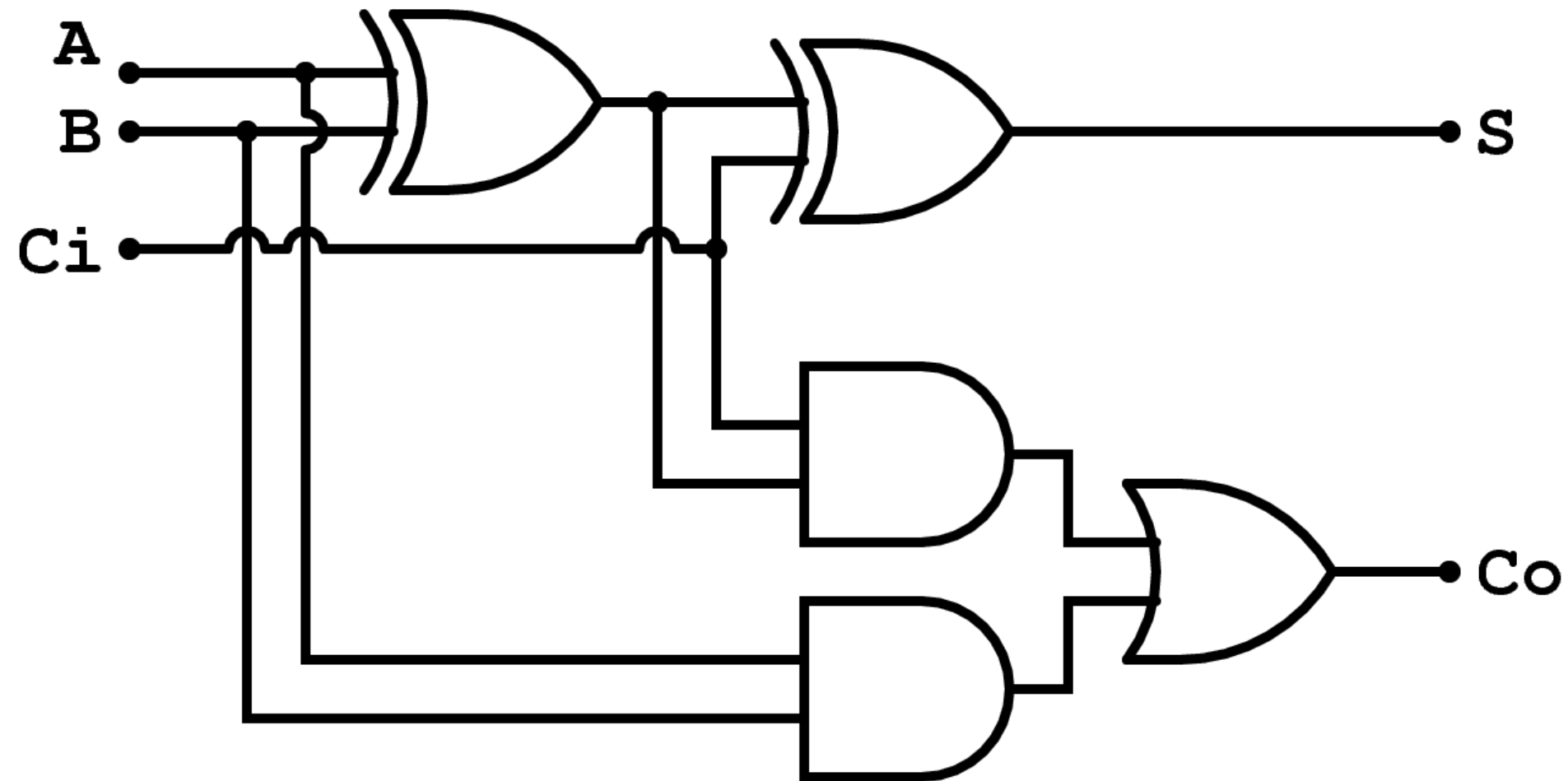
$$(\text{Lưu ý: } x \oplus y = x'y + xy')$$

		A			
		AB			
Ci	0	00	01	11	10
	1	00	01	11	10
		B			

$$C_0 = F(A, B, Ci) = \Sigma(3, 5, 6, 7)$$

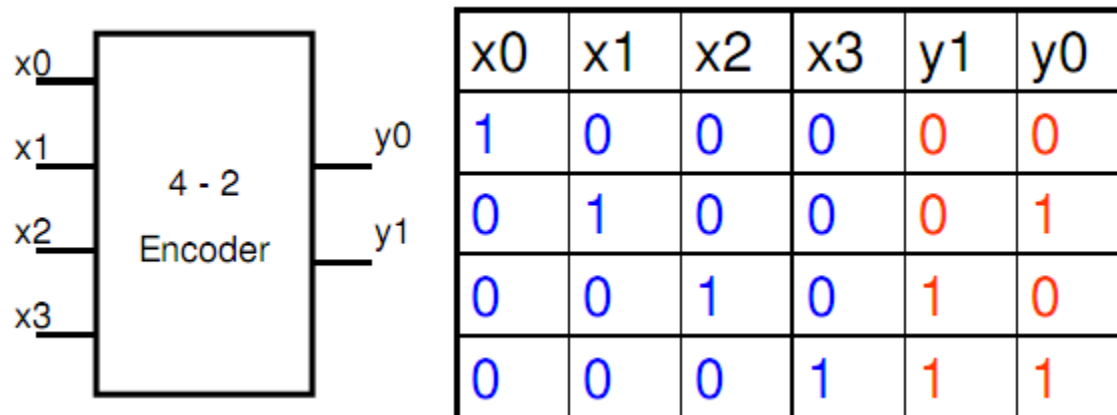
$$C_0 = AB + BCi + ACi$$

# Sơ đồ mạch Full adder



# Mạch mã hoá nhị phân (Binary Encoder)

- Có  $2^n$  (hoặc ít hơn) ngõ vào,  $n$  ngõ ra
- Quy định chỉ có **duy nhất một ngõ vào mang giá trị = 1** tại một thời điểm
- Nếu ngõ vào = 1 đó là ngõ thứ  $k$  thì các ngõ ra tạo thành số nhị phân có giá trị =  $k$

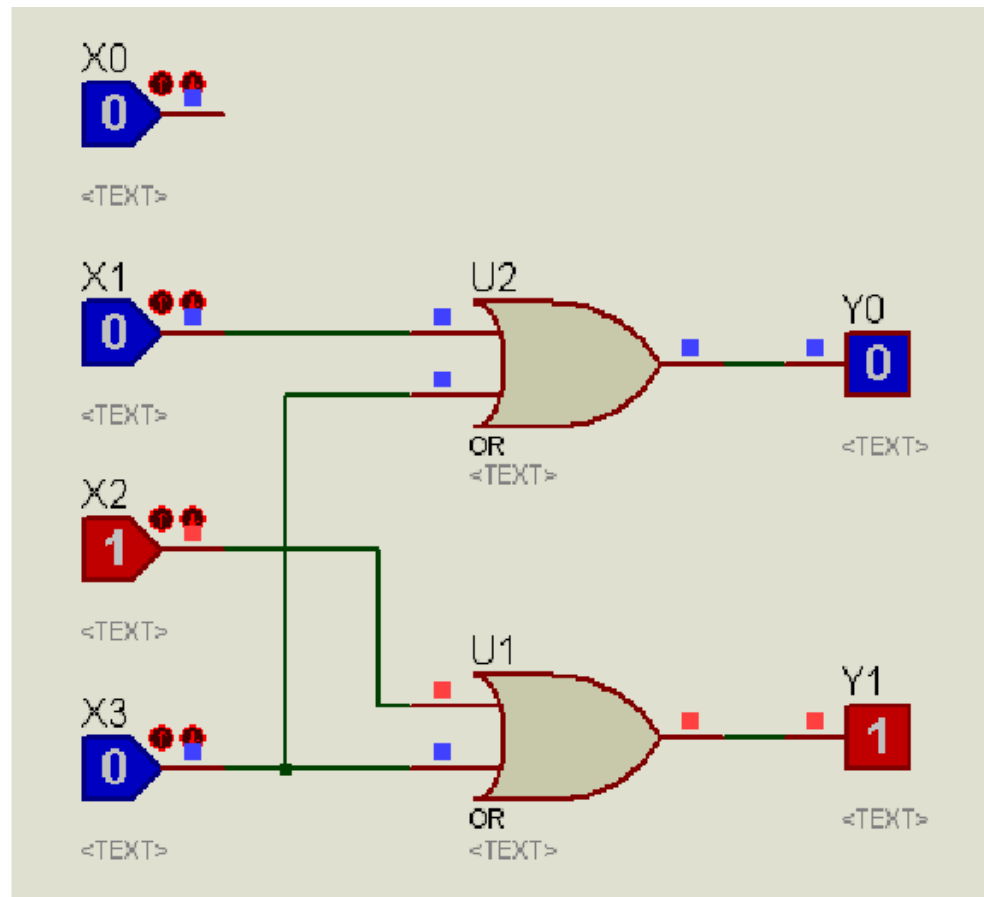


# Sơ đồ mạch 4-2 Binary Encoder

- Ngõ vào:  $X_0, X_1, X_2, X_3$
- Ngõ ra:  $Y_0, Y_1$

$$Y_0 = X_1 + X_3$$

$$Y_1 = X_2 + X_3$$



# Mạch mã hoá theo thứ tự (Priority Encoder)

- Các ngõ vào được xem như có độ ưu tiên
- Giá trị ngõ ra phụ thuộc vào các ngõ vào có độ ưu tiên cao nhất
- Ví dụ: Độ ưu tiên ngõ vào  $x_3 > x_2 > x_1 > x_0$

x0	x1	x2	x3	y2	y1	y0
0	0	0	0	0	0	0
1	0	0	0	0	0	1
x	1	0	0	0	1	0
x	x	1	0	0	1	1
x	x	x	1	1	0	0

$$y_0 = (x_2 + x_0x_1').x_3$$

$$y_1 = (x_2 + x_1).x_3'$$

$$y_2 = x_3$$

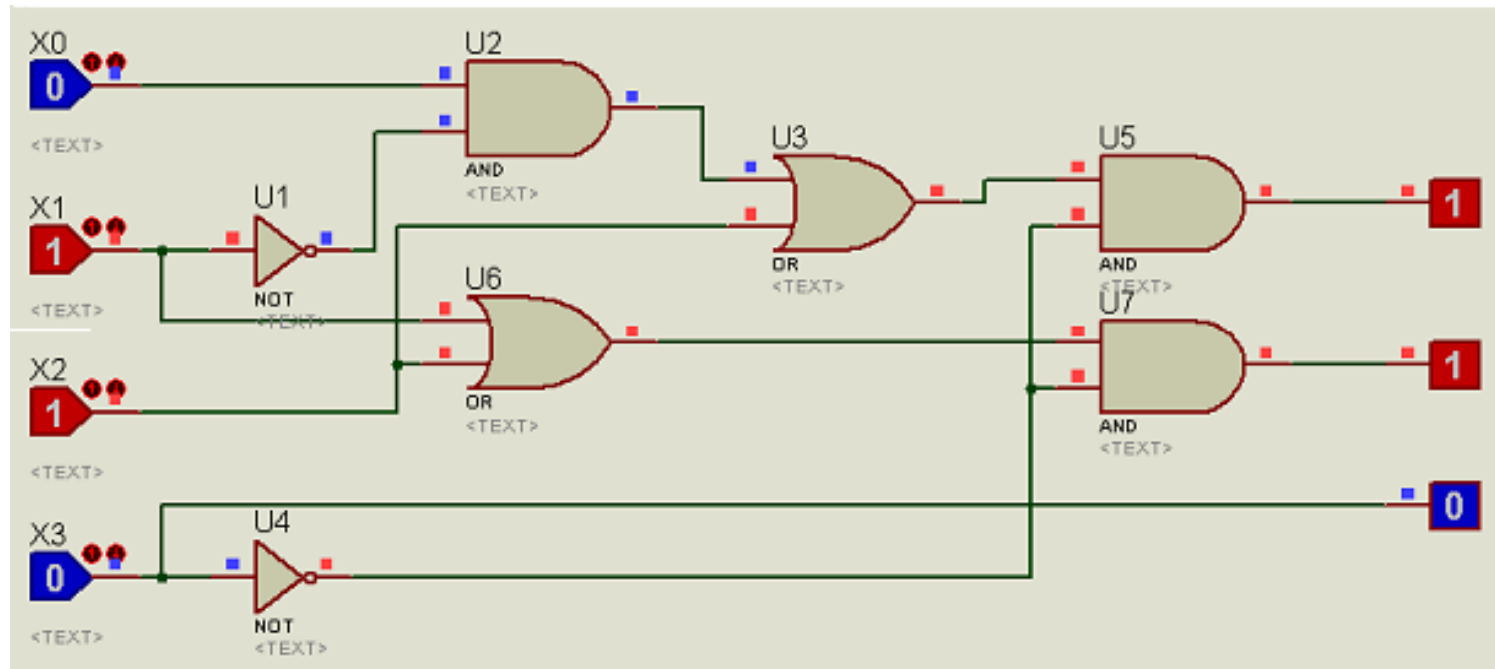


# Sơ đồ mạch 4-3 Priority Encoder

$$y_0 = (x_2 + x_0x_1').x_3$$

$$y_1 = (x_2 + x_1).x_3'$$

$$y_2 = x_3$$

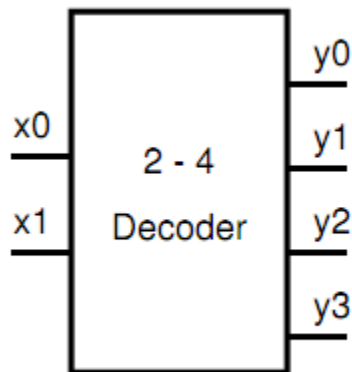






# Mạch giải mã (Decoder)

- Có  $n$  ngõ vào,  $2^n$  (hoặc ít hơn) ngõ ra
- Quy định chỉ có **duy nhất một ngõ ra mang giá trị = 1** tại một thời điểm
- Nếu các ngõ vào tạo thành số nhị phân có giá trị =  $k$  thì ngõ ra = 1 đó là ngõ thứ  $k$



x1	x0	y0	y1	y2	y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

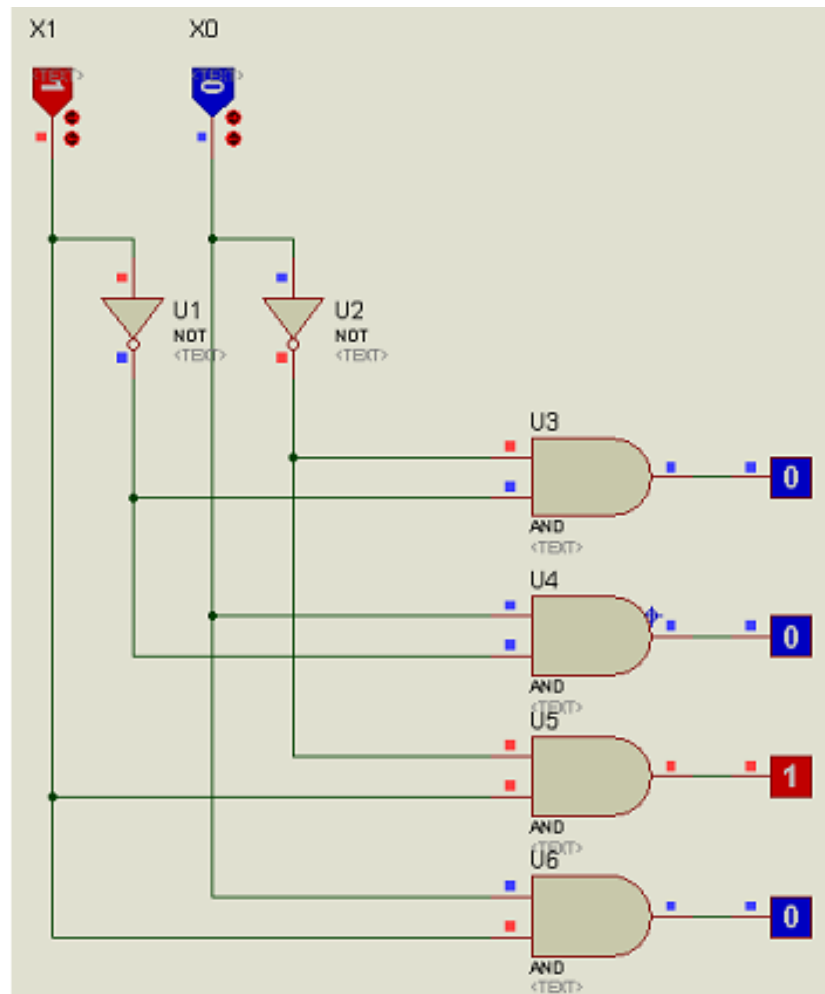
$$y0 = \overline{x1}.\overline{x0}$$

$$y1 = \overline{x1}.x0$$

$$y2 = x1.\overline{x0}$$

$$y3 = x1.x0$$

# Sơ đồ mạch 2-4 Decoder

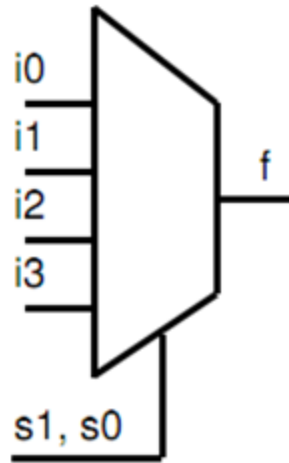
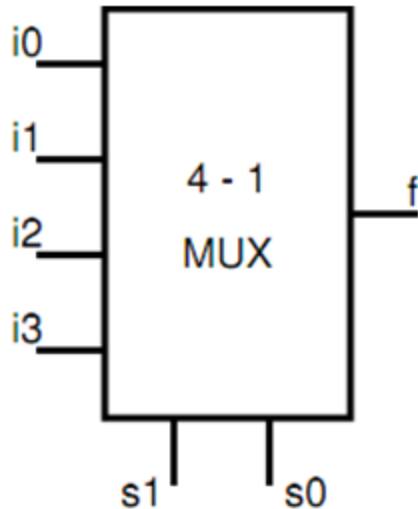


# Mạch dồn (Multiplexer - MUX)

- Còn gọi là mạch chọn dữ liệu
- Chọn  $n$  ngõ trong  $2^n$  ngõ vào để quyết định giá trị của duy nhất 1 ngõ ra
- Mạch dồn  $2^n - 1$  có  $2^n$  ngõ nhập, 1 ngõ xuất và  $n$  ngõ nhập chọn



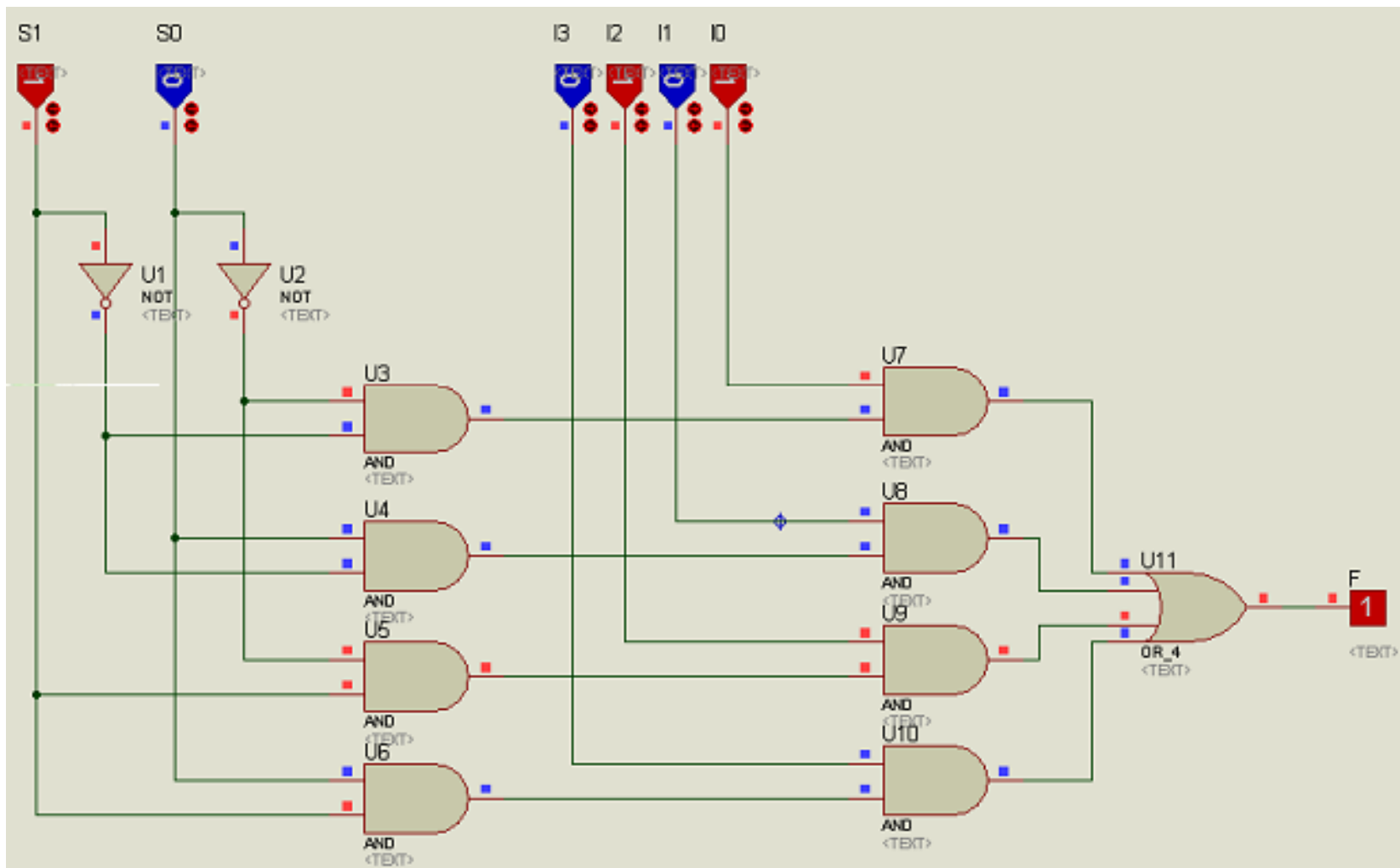
# Ví dụ: Mạch 4-1 MUX



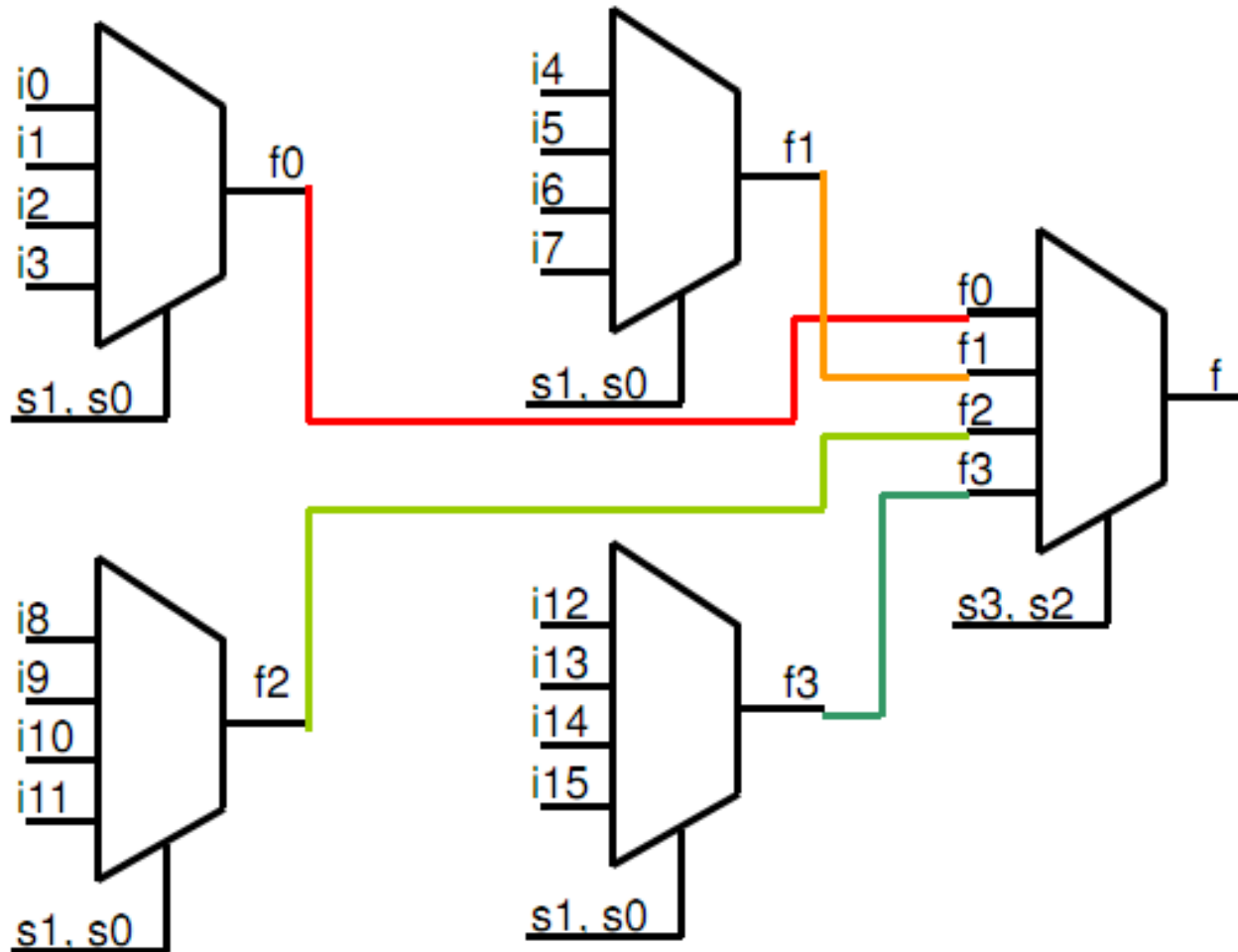
$s_1$	$s_0$	$f$
0	0	$i_0$
0	1	$i_1$
1	0	$i_2$
1	1	$i_3$

tín hiệu  $s_1$ ,  $s_0$  dùng để lựa chọn xem tín hiệu nào trong các ngõ vào  $i_0, i_1, i_2, i_3$  được chuyển đến ngõ ra  $f$

# Sơ đồ mạch 4-1 MUX



# 16-1 MUX

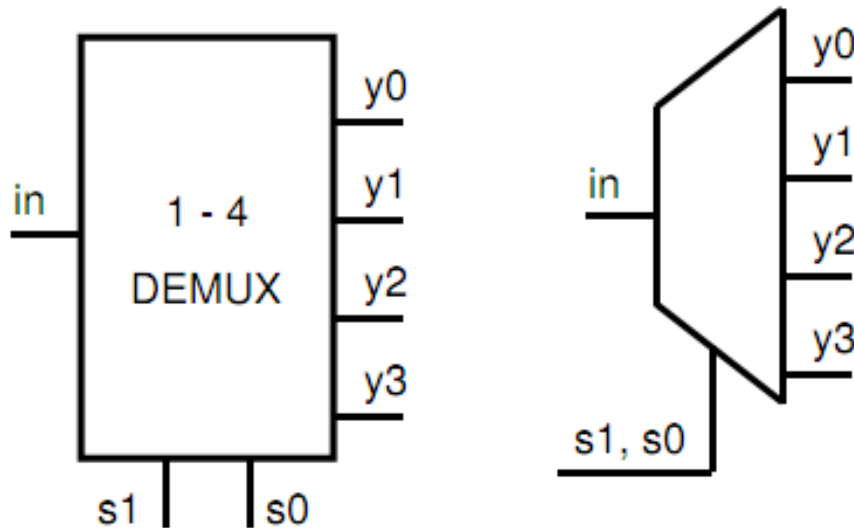


# Mạch tách Demultiplexer (DEMUX)

- Chọn  $n$  ngõ trong  $2^n$  ngõ vào để quyết định giá trị của duy nhất 1 ngõ ra
- Mạch DEMUX  $1-2^n$  có 1 ngõ nhập,  $2^n$  ngõ xuất và  $n$  ngõ nhập chọn



# Ví dụ: Mạch 1-4 DEMUX

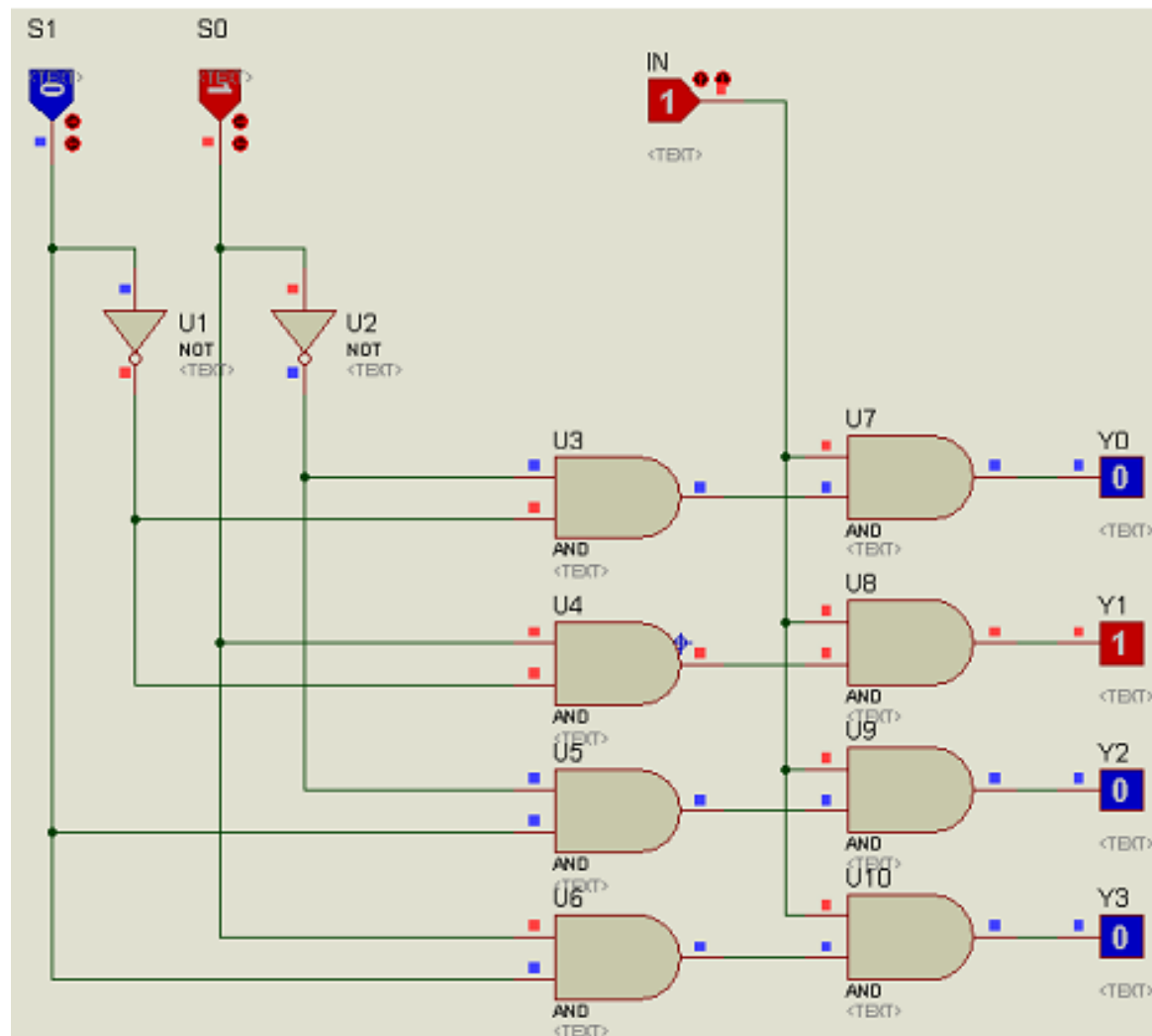


tín hiệu **s1**, **s0** dùng để lựa chọn xem tín hiệu vào **in** sẽ được chuyển đến ngõ nào trong các ngõ ra **y0,y1,y2,y3**

s1	s0	y0	y1	y2	y3
0	0	in	0	0	0
0	1	0	in	0	0
1	0	0	0	in	0
1	1	0	0	0	in



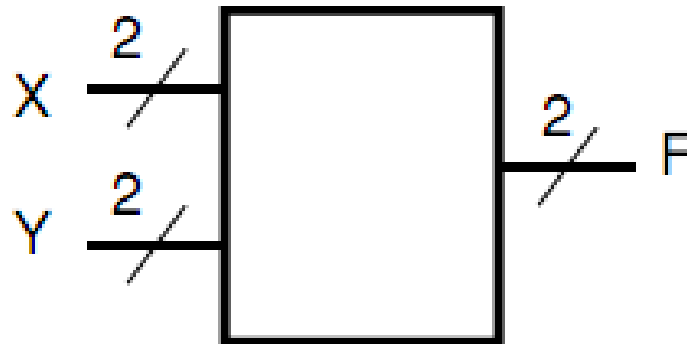
# Sơ đồ mạch 1-4 DEMUX



# Bài tập: Thiết kế mạch ALU

- $F = (5X + 2Y) \% 4$
- Input: X (2 bit), Y (2 bit)
- Output: F (2 bit)

→ Có 4 ngõ vào, 2 ngõ ra (mỗi ngõ có 1 tín hiệu biểu diễn cho 1 bit)



# Bước 1: Lập bảng chân trị

X	Y	F
0 (00)	0 (00)	0 (00)
0 (00)	1 (01)	2 (10)
0 (00)	2 (10)	0 (00)
0 (00)	3 (11)	2 (10)
1 (01)	0 (00)	1 (01)
1 (01)	1 (01)	3 (11)
1 (01)	2 (10)	1 (01)
1 (01)	3 (11)	3 (11)
2 (10)	0 (00)	2 (10)
2 (10)	1 (01)	0 (00)
2 (10)	2 (10)	2 (10)
2 (10)	3 (11)	0 (00)
3 (11)	0 (00)	3 (11)
3 (11)	1 (01)	1 (01)
3 (11)	2 (10)	3 (11)
3 (11)	3 (11)	1 (01)



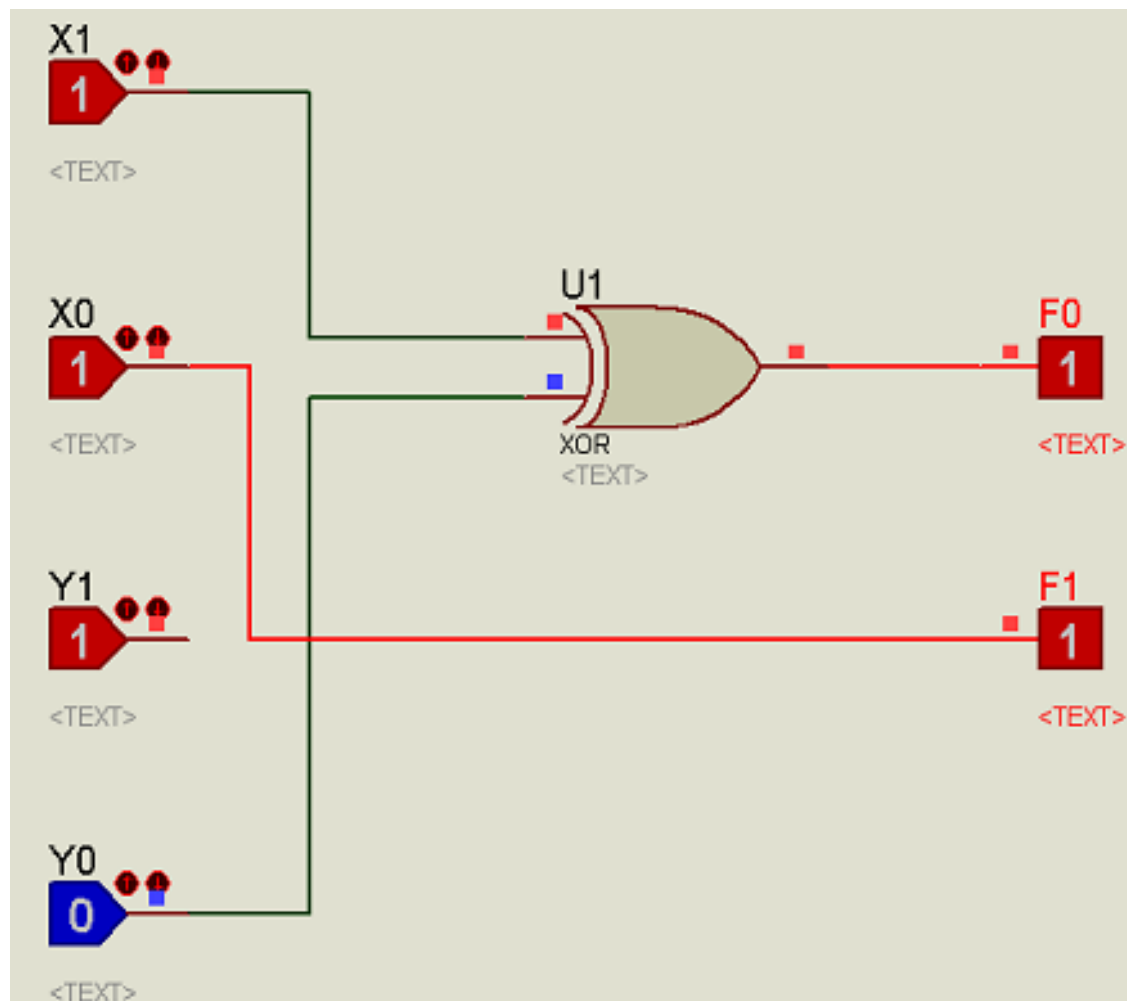
## Bước 2: Xác định hàm

		Y1 Y0			
		00	01	11	10
X1 X0	00	0	1	1	0
	01	0	1	1	0
	11	1	0	0	1
	10	1	0	0	1

$$F1 = X1.Y0' + X1'.Y0$$

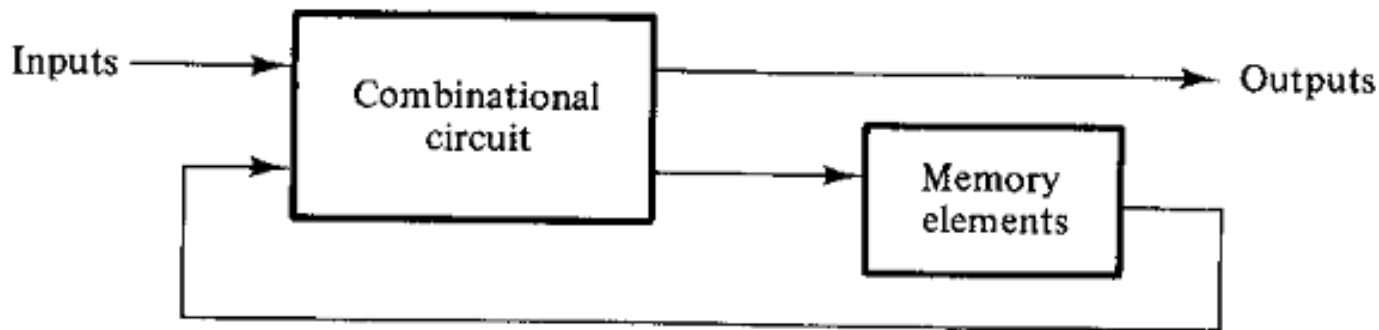
$$F0 = X0$$

# Bước 3: Vẽ mạch

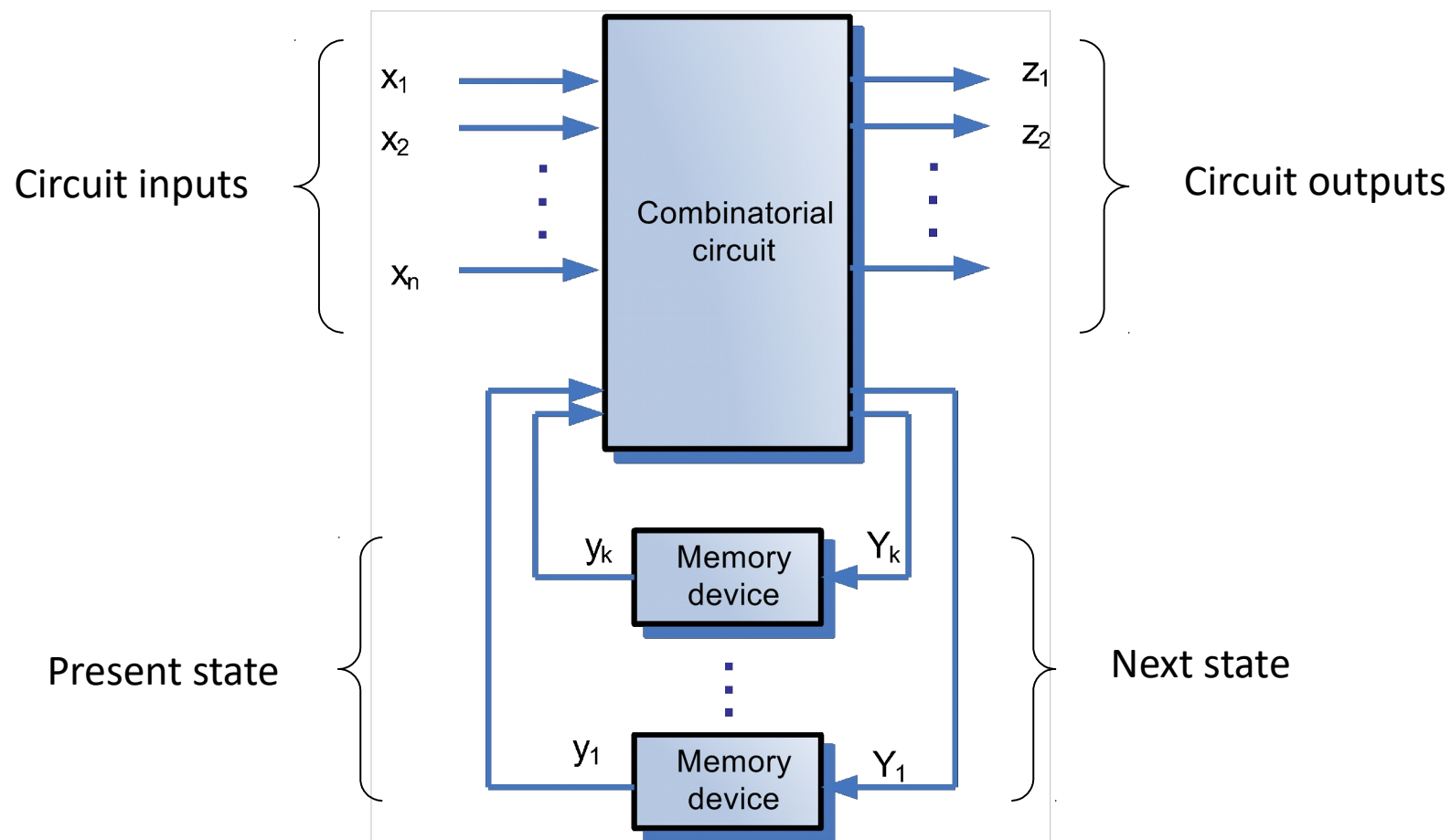


# Phần 3: Mạch tuần tự

- Khác với mạch tổ hợp, ở mạch tuần tự thì ngõ ra không chỉ phụ thuộc vào giá trị hiện thời của ngõ vào, mà còn phụ thuộc giá trị quá khứ
- Mạch tuần tự có khả năng “ghi nhớ các trạng thái trong quá khứ”



# Mạch tuần tự



# Mạch lật

- Là 1 thành phần cấu thành mạch tuần tự
- Có chức năng lưu trữ 1 bit nhị phân
- Có nhiều loại mạch lật, sự khác nhau ở chỗ số ngõ vào và cách thức các ngõ vào tác động đến trạng thái bit nhị phân

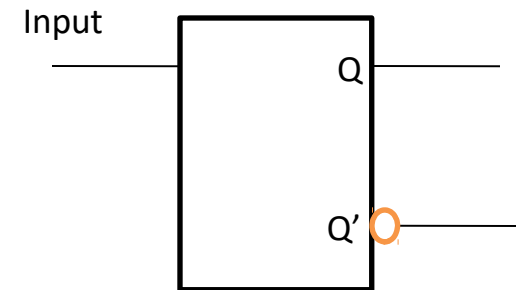




# Phân loại mạch lật

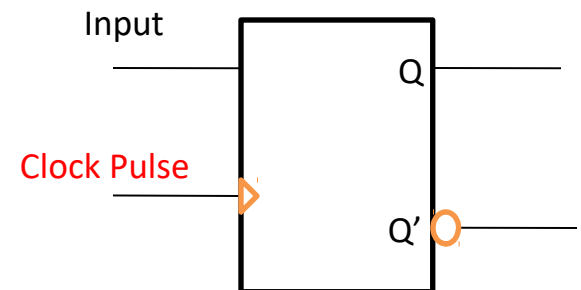
## □ Latch

- Ngõ ra thay đổi trạng thái khi ngõ vào thay đổi giá trị
- Độ trễ mạch (delayed gate) giá trị mới của ngõ ra được xác định bằng độ trễ giữa ngõ vào và ngõ ra
- Được sử dụng như 1 thành phần nhớ của mạch tuần tự **bất đồng bộ**

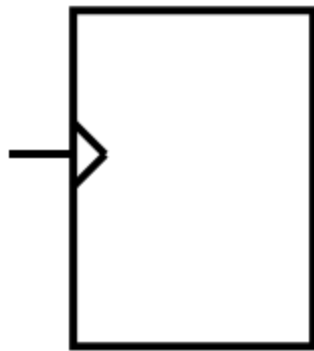


## □ Flip-Flop

- Bên cạnh những ngõ vào thông thường thì luôn có 1 ngõ vào kích hoạt (trigger input), gọi là clock
- Trạng thái của ngõ ra chỉ có thể thay đổi khi ngõ vào kích hoạt (clock) thay đổi xung đồng hồ (clock pulse) của nó (0 → 1 hoặc 1 → 0)
- Được sử dụng như 1 thành phần nhớ của mạch tuần tự **đồng bộ**

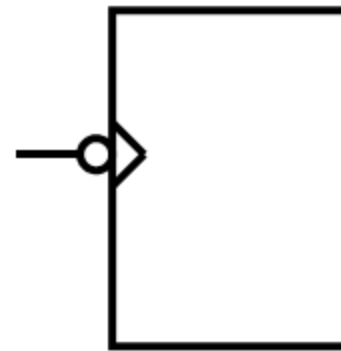



# Tín hiệu lễ xung đồng hồ - Clock edge



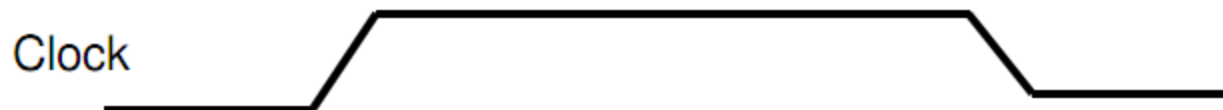
Clk 

Chuyển tiếp lễ dương ( $0 \rightarrow 1$ )



Clk 

Chuyển tiếp lễ âm ( $1 \rightarrow 0$ )



Positive edge

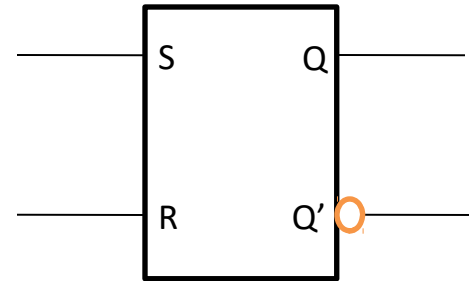
Negative edge

# RS Latch (SR Latch)

- Có 2 ngõ vào:

- S (Set): Đặt

- R (Reset): Khởi động

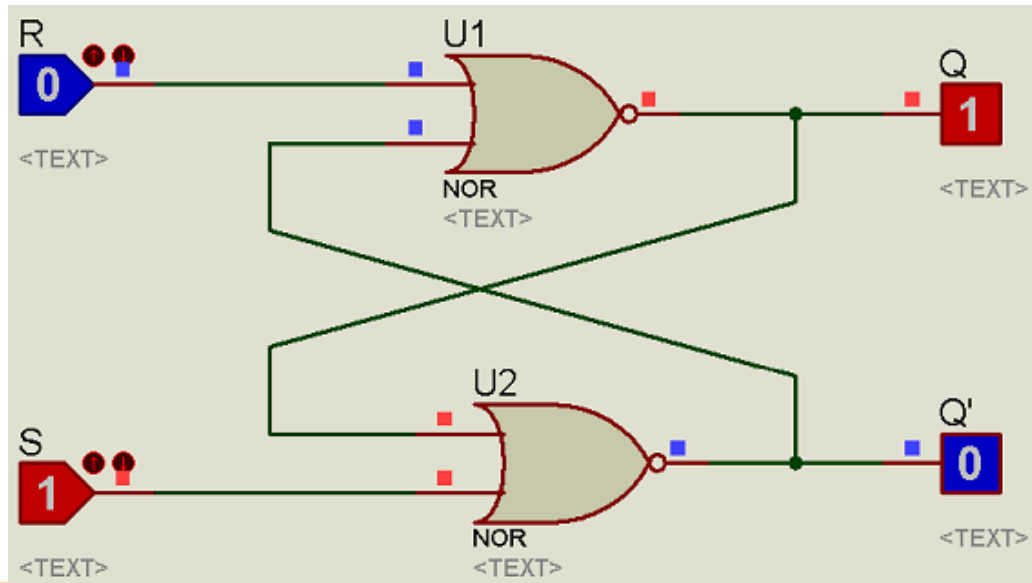


- Có 2 ngõ ra Q và Q' (tín hiệu đảo của Q)
- Trạng thái ngõ ra  $Q_{\text{next}} = Q(t+1)$  phụ thuộc vào trạng thái ngõ vào S, R và tình trạng hiện tại của mạch  $Q_{\text{current}} = Q(t)$

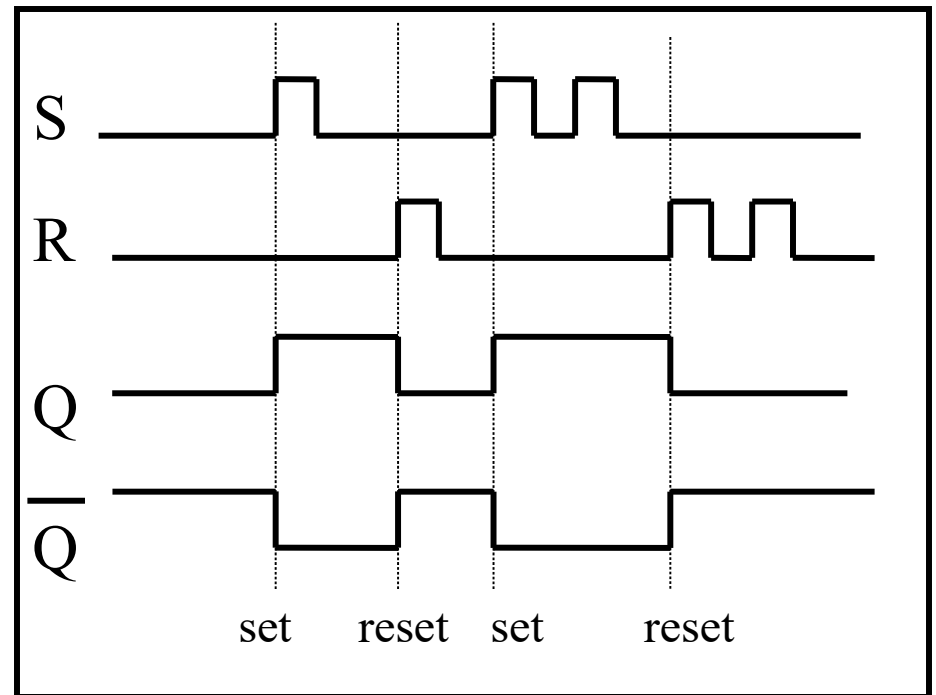
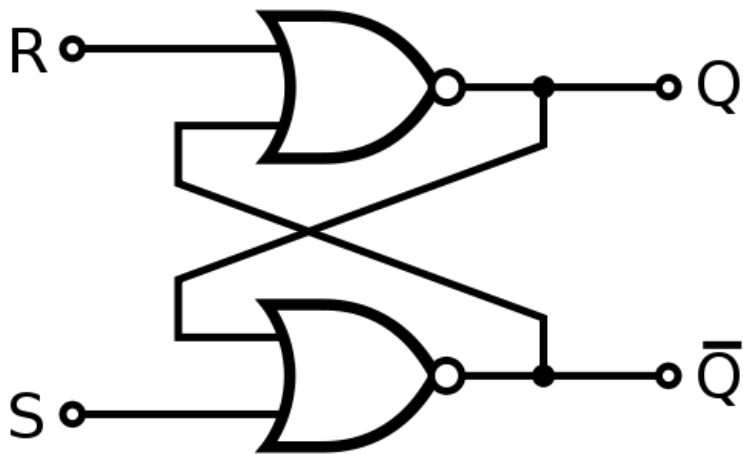


# RS Latch

S	R	$Q = Q(t+1)$	$Q'$	Ý nghĩa
0	0	$Q(t)$	$(Q(t))'$	Không đổi
0	1	0	1	= 0
1	0	1	0	= 1
1	1	undefined	undefined	Không xác định

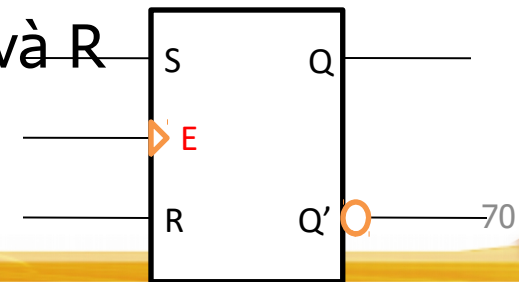


# Timing chart

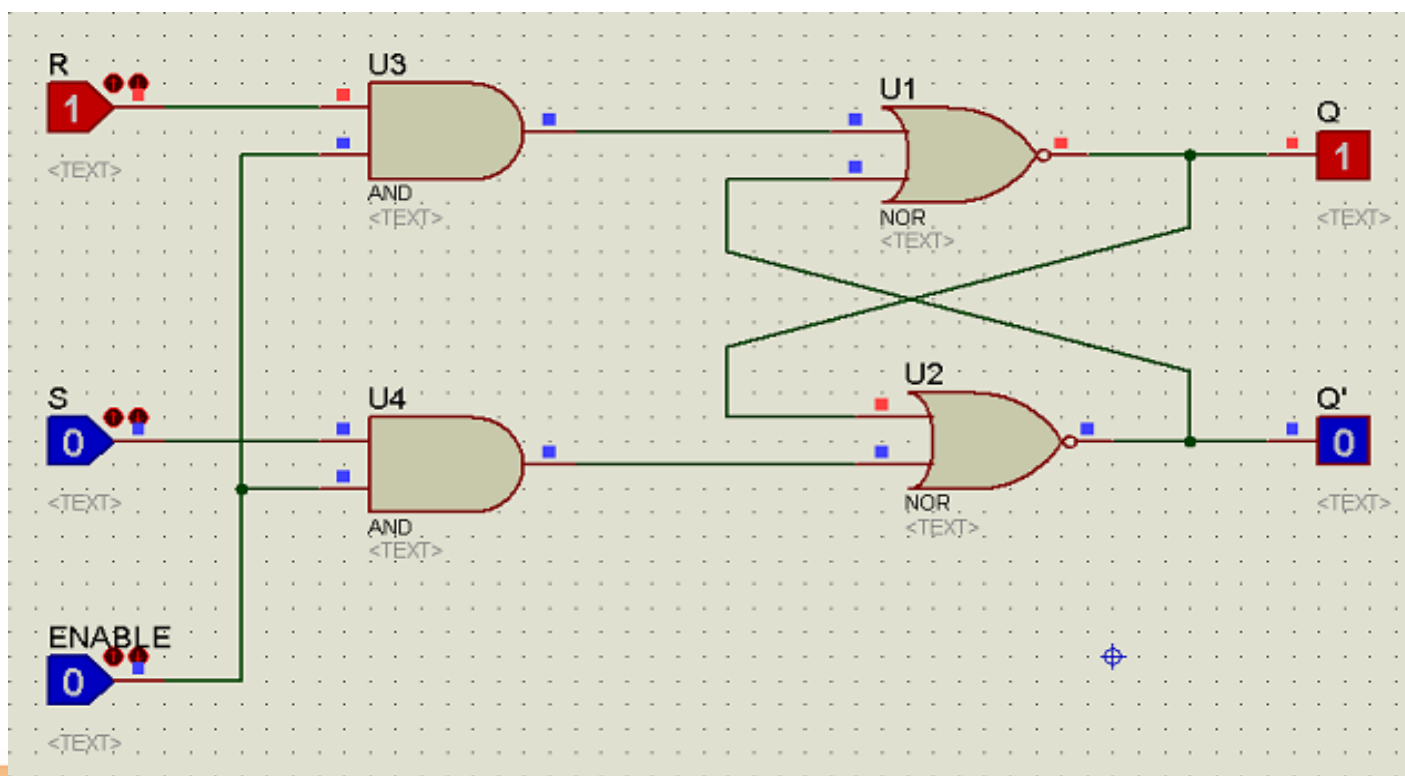


# RS Flip-Flop

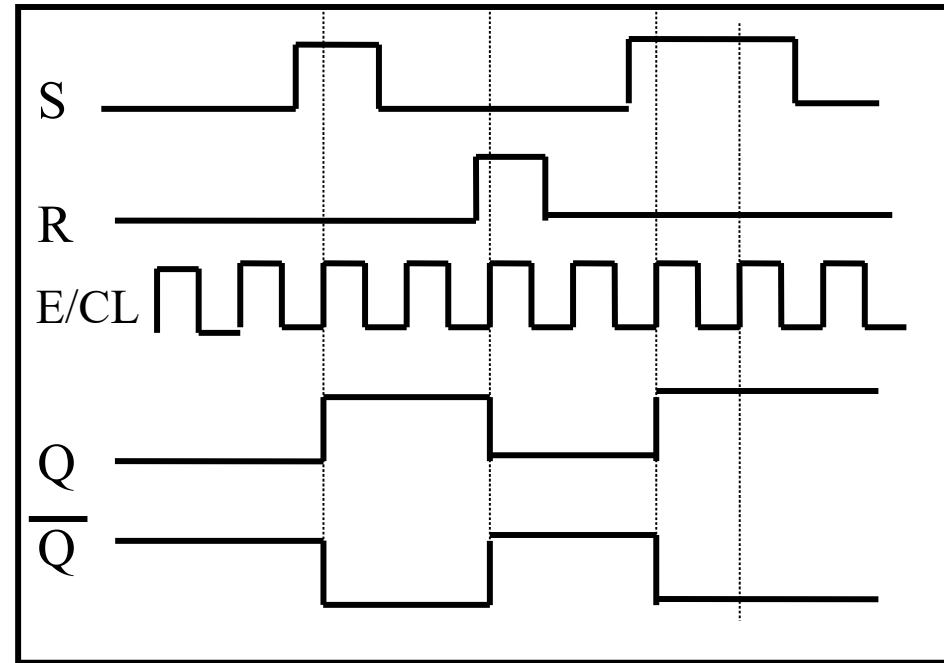
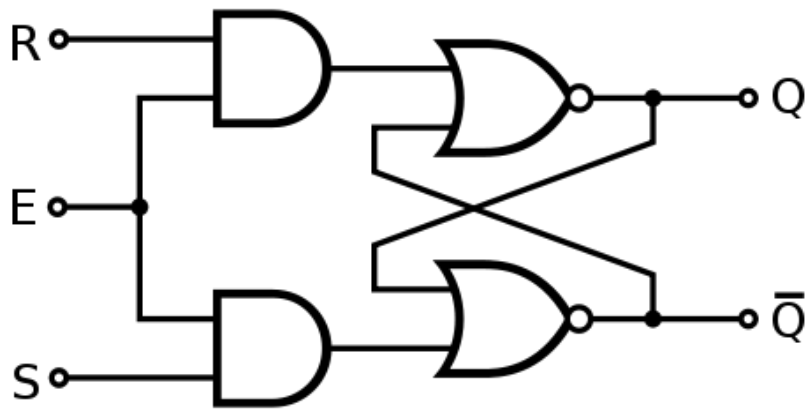
- Dùng thêm 1 tín hiệu ngõ vào kích hoạt “Enabled” (thường là tín hiệu xung đồng hồ Clock - C) để điều khiển mạch
    - Enabled = 1 (Positive Clock Edge): mạch hoạt động như mạch lật RS Latch
    - Enabled = 0 (Negative Clock Edge): mạch bị vô hiệu hoá,  
→ Q giữ nguyên giá trị →  $Q(t+1) = Q(t)$
- Chỉ khi tín hiệu Enabled đổi từ 0 sang 1 (positive edge triggered), ngõ ra mới có thể bị ảnh hưởng, nếu không thì không thể thay đổi bất chấp giá trị của S và R



E	S	R	$Q = Q(t+1)$	$Q'$	Ý nghĩa
0	x	x	$Q(t)$	$(Q(t))'$	Không đổi
1	0	0	$Q(t)$	$(Q(t))'$	Không đổi
1	0	1	0	1	= 0
1	1	0	1	0	= 1
1	1	1	undefined	undefined	Không xác định



# Timing chart



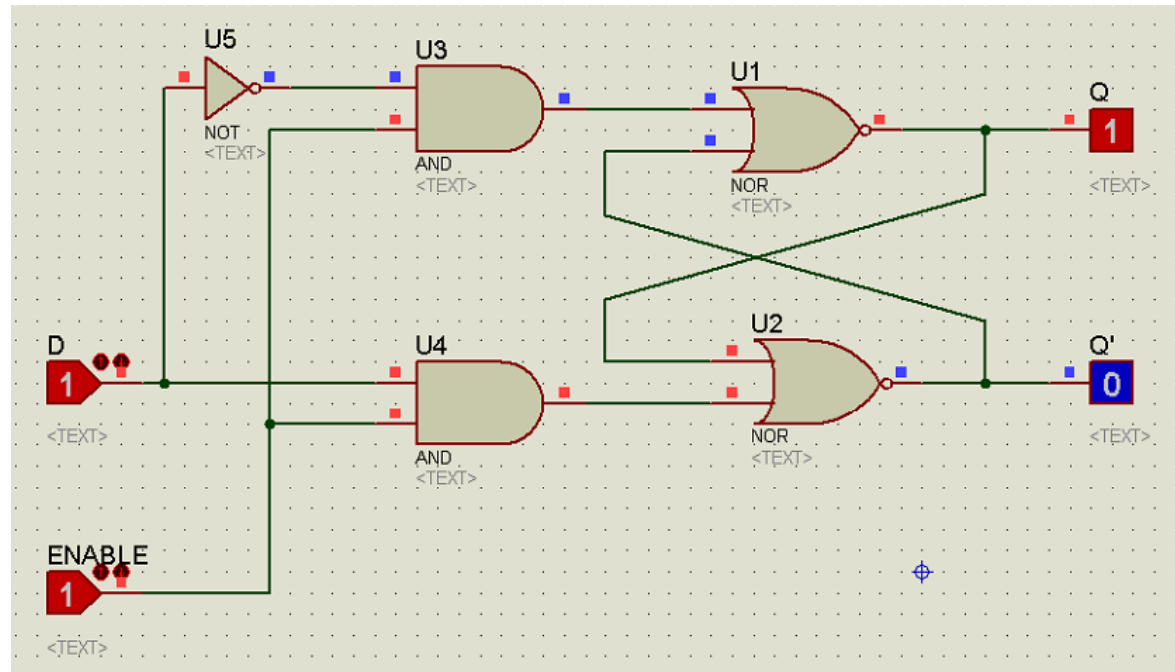
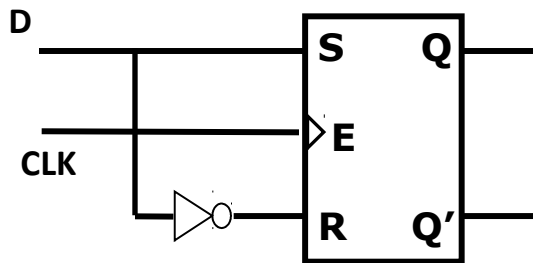
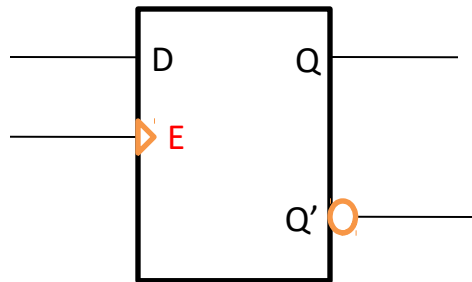


# D (Data) Flip-Flop

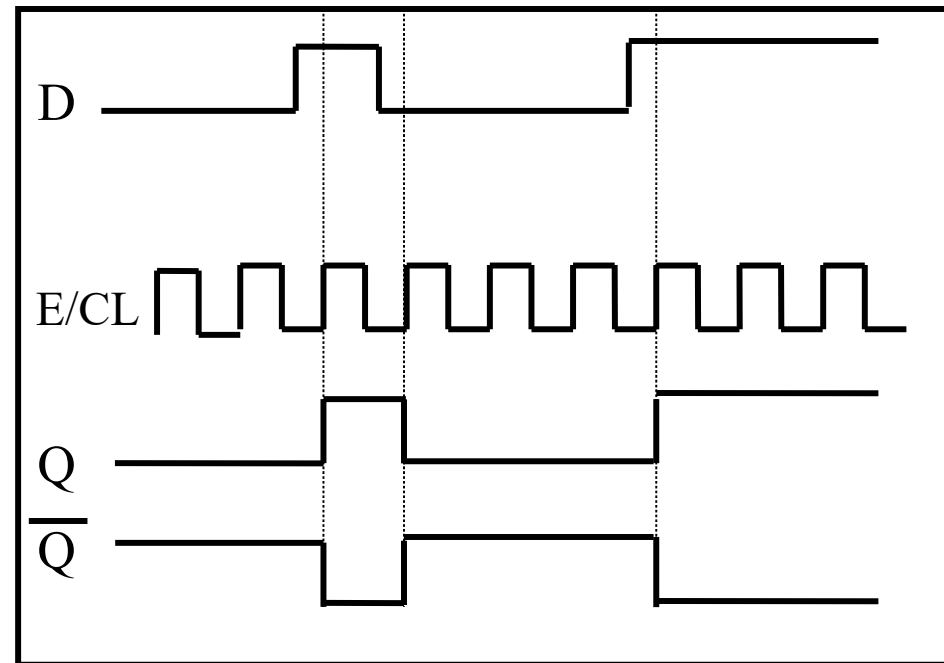
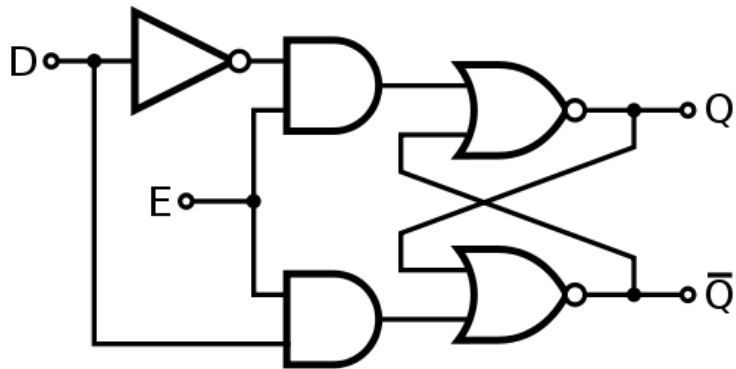
- Để tránh trường hợp  $R = S = 1$  trong RS Flip-Flop, trong mạch lật D Flip-Flop ta chỉ dùng **1 ngõ vào D** nhưng tách ra 2 tín hiệu, 1 trong 2 tín hiệu sẽ đi qua cổng NOT để tạo tín hiệu đảo của D
  - Không bao giờ xảy ra trường hợp 2 tín hiệu vào mạch đều bằng 1
  - Nhưng bên cạnh đó cũng không bao giờ xảy ra 2 tín hiệu vào mạch đều bằng 0 ☹️
  - Ta không thể giữ nguyên trạng thái tín hiệu ngõ ra  $Q(t + 1) = Q(t)$
  - ***Để khắc phục ta sẽ dùng tín hiệu xung đồng hồ để vô hiệu hoá mạch khi cần, lúc đó trạng thái tín hiệu ngõ ra sẽ không đổi***



E	D	$Q = Q(t+1)$	$Q'$	Ý nghĩa
0	x	$Q(t)$	$(Q(t))'$	Không đổi
1	0	0	1	= 0
1	1	1	0	= 1

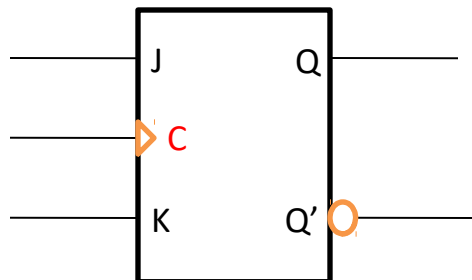


# Timing chart



# JK Flip-Flop

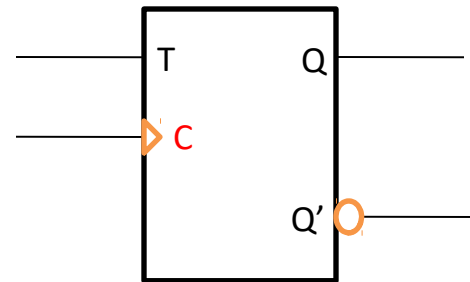
- Là 1 cải tiến của mạch RS Flip-Flop đối với trường hợp  $R = S = 1$
- Nguyên tắc:
  - $J = S$
  - $K = R$
  - Nếu  $J = K = 1$  thì khi đó với 1 chuyển tiếp của tín hiệu xung đồng hồ sẽ chuyển tín hiệu ngõ ra Q sang trạng thái bù  $Q'$



J	K	$Q = Q(t+1)$	$Q'$	Ý nghĩa
0	0	$Q(t)$	$(Q(t))'$	Không đổi
0	1	0	1	= 0
1	0	1	0	= 1
1	1	$Q'(t)$	$Q(t)$	Đảo bit

# Mạch lật T

- Xuất phát từ mạch JK Flip-Flop với sự kết hợp 2 ngõ vào J, K thành duy nhất 1 ngõ vào T ( $T = J = K$ )
  - $T = 0: Q(t + 1) = Q(t)$
  - $T = 1: Q(t + 1) = (Q(t))'$

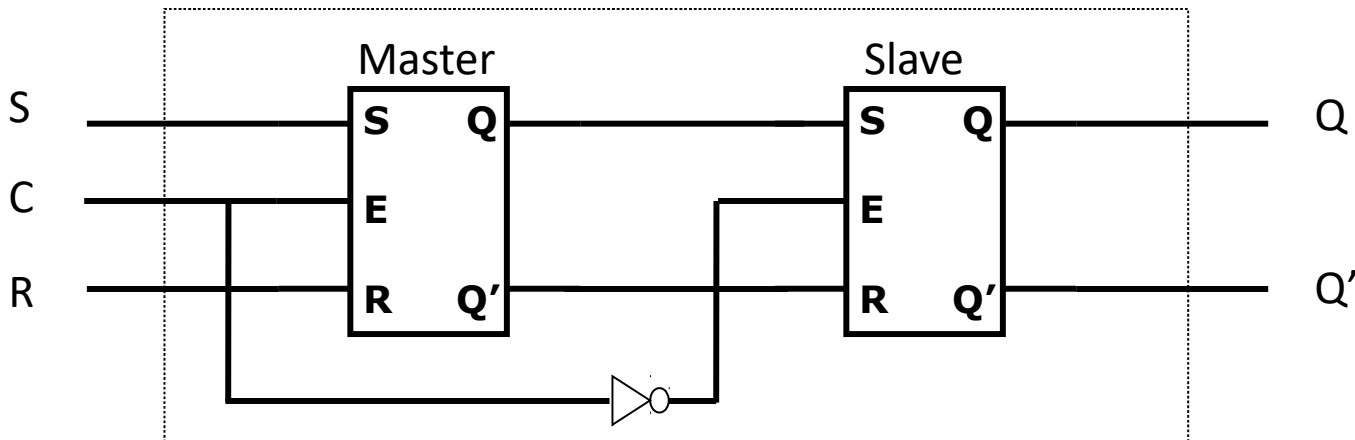


T	$Q = Q(t+1)$	$Q'$	Ý nghĩa
0	$Q(t)$	$(Q(t))'$	Không đổi
1	$(Q(t))'$	$Q(t)$	Đảo bit



# Master-Slave Flip-Flop

- Bao gồm 2 bản mạch flip-flop tuần tự nối với nhau (master – slave)
- Tín hiệu ngõ ra Q phụ thuộc vào giá trị của những ngõ vào tại những chuyển tiếp lề âm / dương của xung đồng hồ (clock edge)

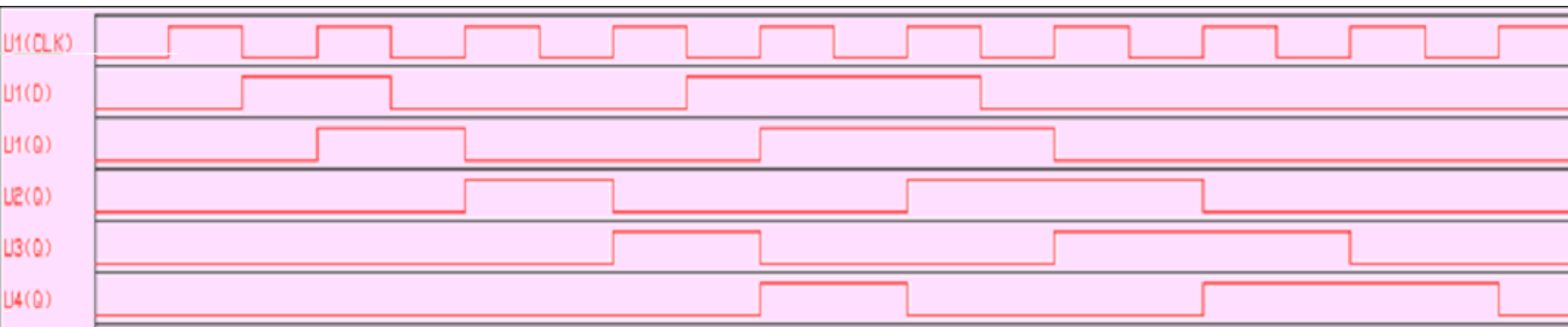
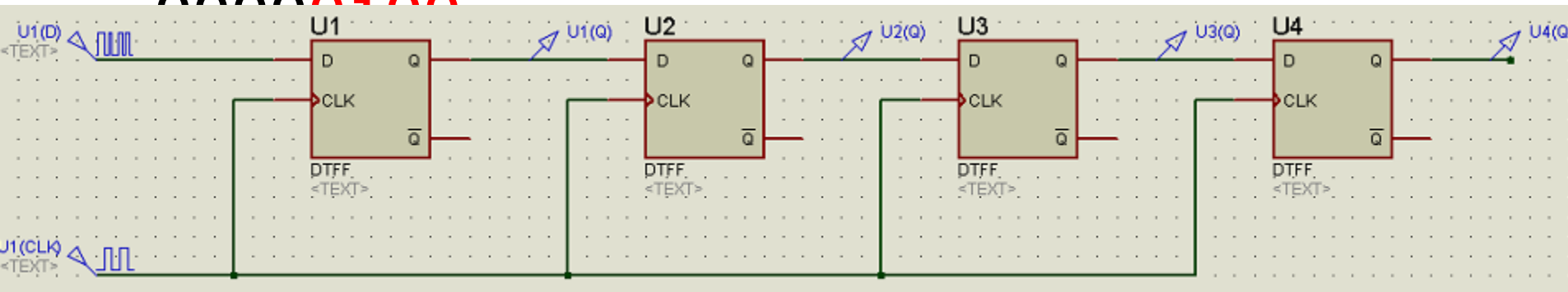


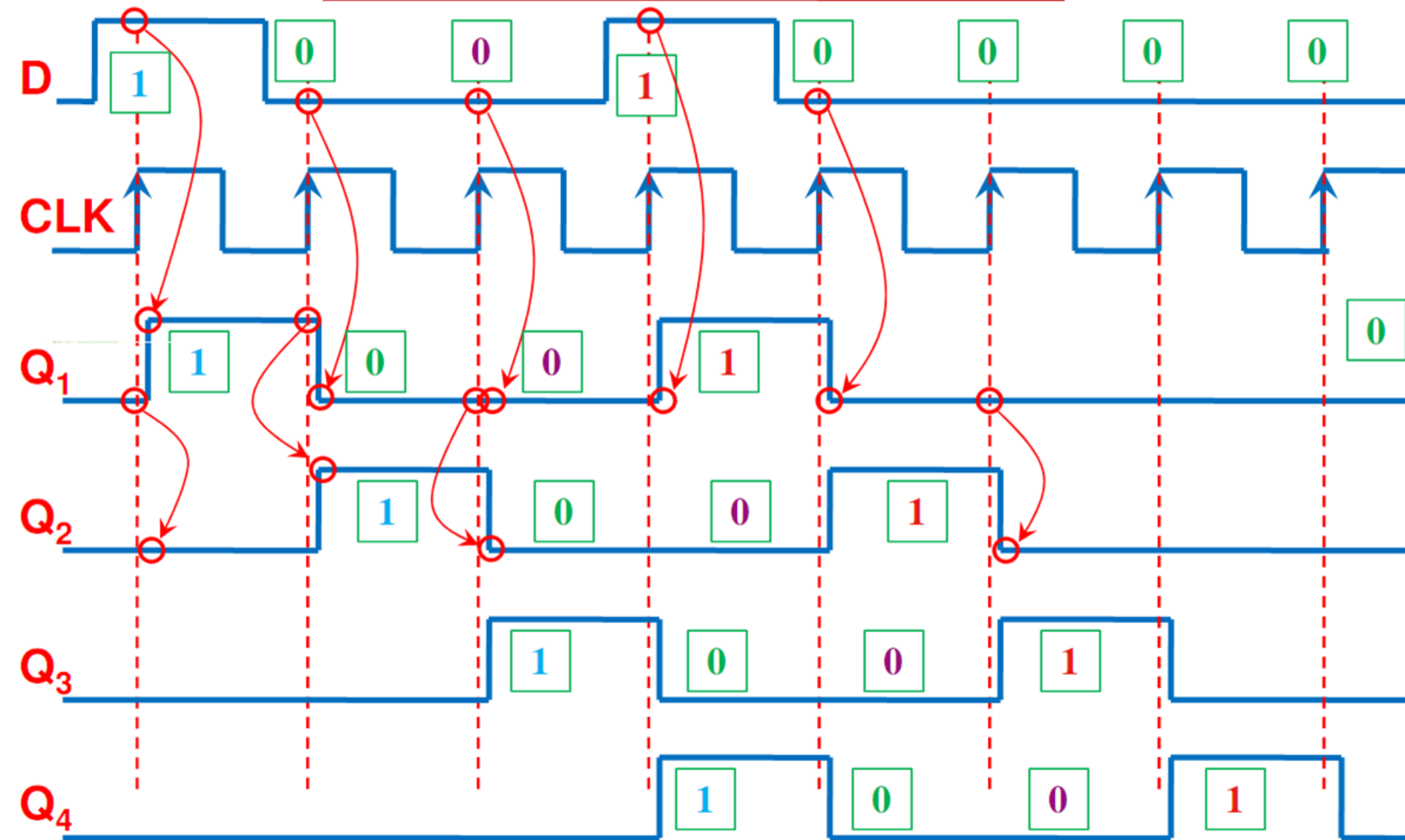
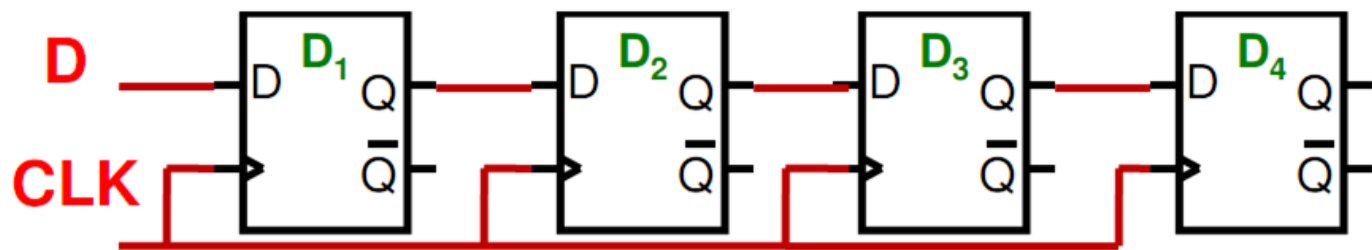
Master works when C=1

Slave works when C=0

# Thanh ghi dịch (Shift Register)

- Thanh ghi dịch 4 bit
- $U1(D) = 01001100\dots \rightarrow U4(Q) =$   
 $00000100$

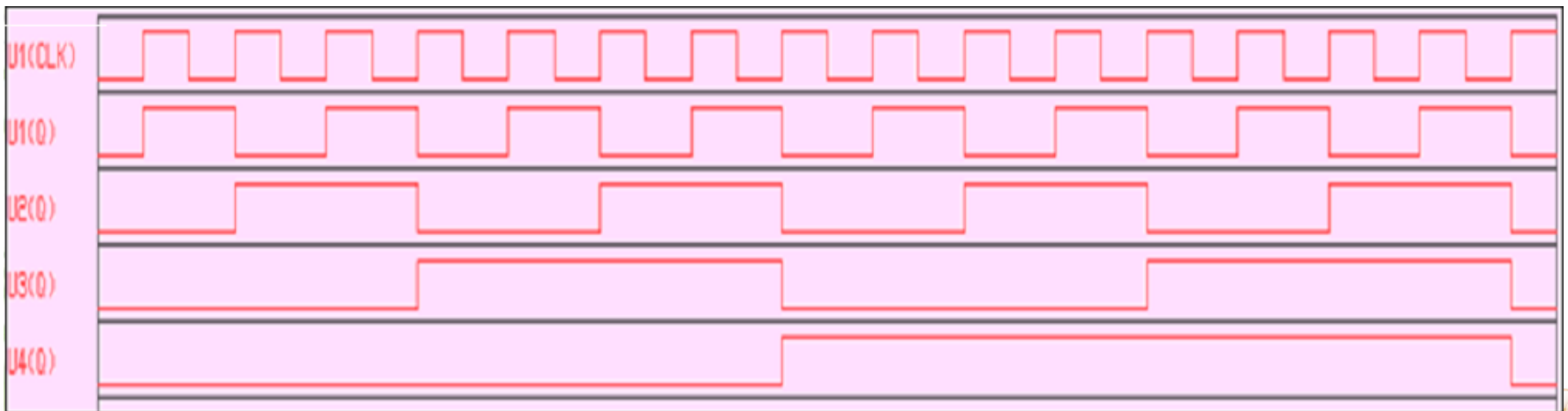
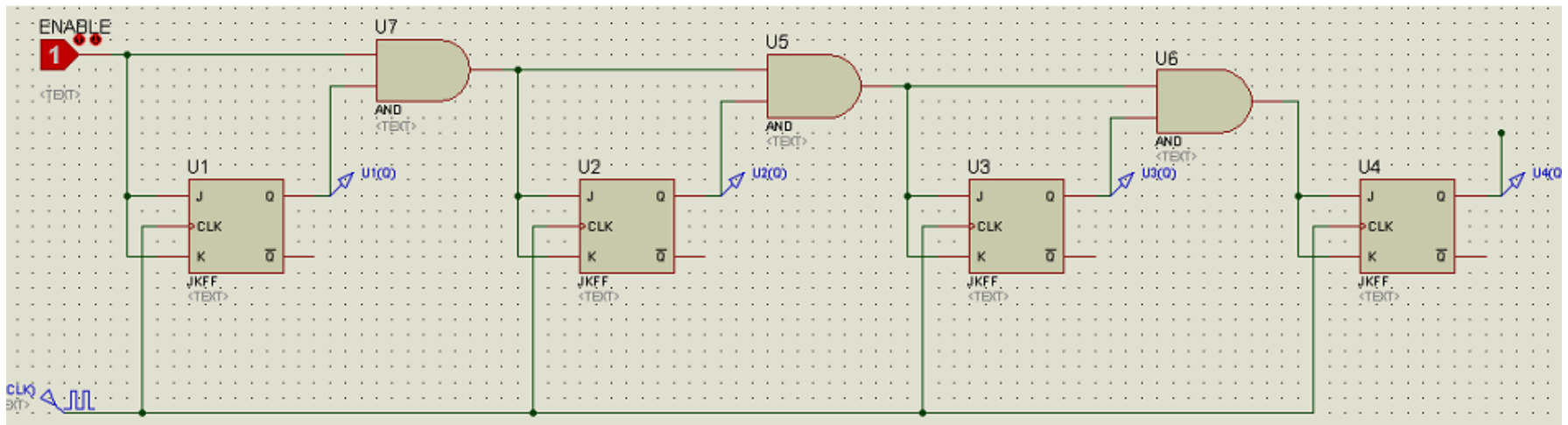






# Bộ đếm (Counter)

- Mạch đếm đồng bộ nhị phân 4 bit ( $0 \rightarrow 15$ )



Số xung vào	Ngõ ra sau khi có xung vào				Trị thập phân ra
	Q3	Q2	Q1	Q0	
Xoá	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	10
11	1	0	1	1	11
12	1	1	0	0	12
13	1	1	0	1	13
14	1	1	1	0	14
15	1	1	1	1	15
16	0	0	0	0	0
17	0	0	0	1	1

Ck —

Q<sub>0</sub> 0

Q<sub>1</sub> 0

Q<sub>2</sub> 0

Q<sub>3</sub> 0

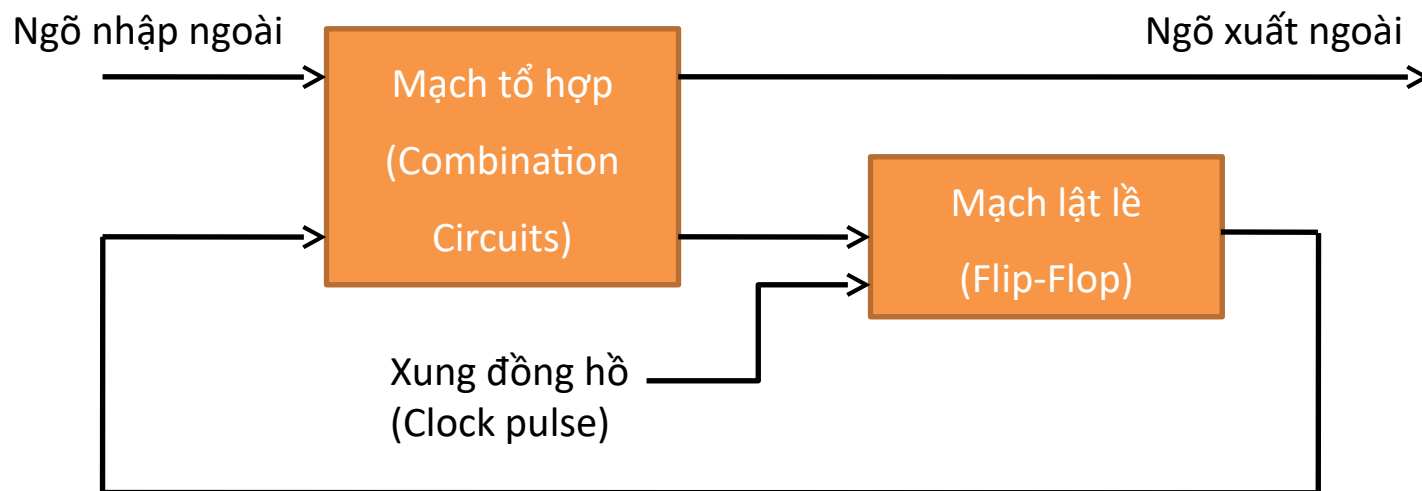
↑ 0

Xoá



# Mạch tuần tự đồng bộ

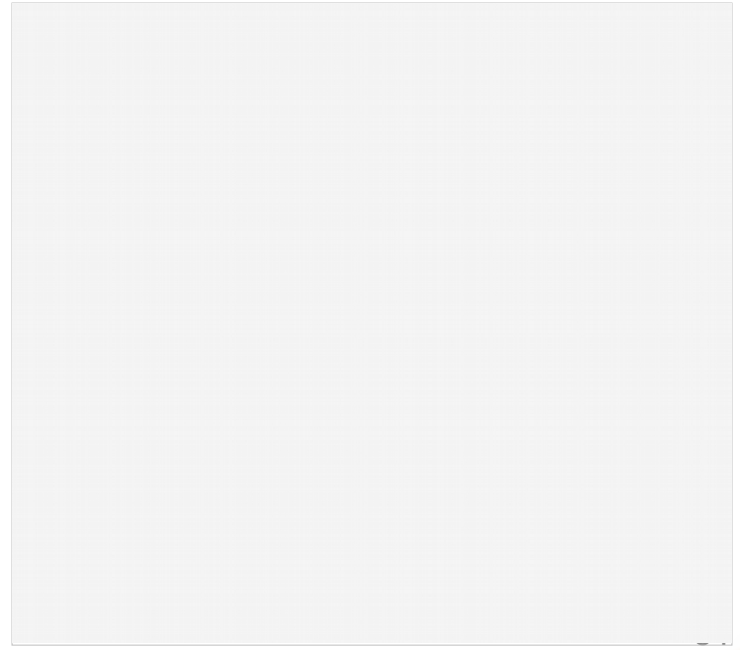
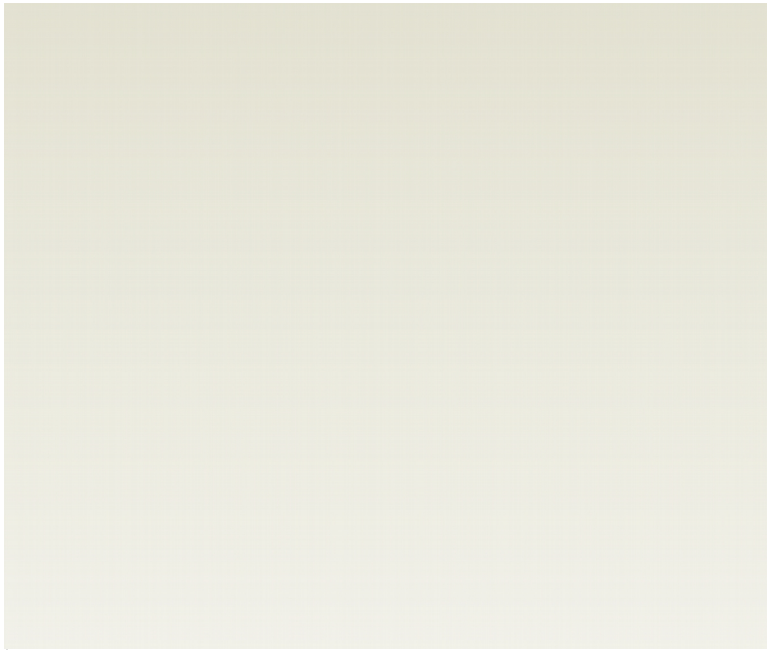
83



- Mạch tuần tự được xác định bởi:
  - Các ngõ nhập ngoài
  - Các ngõ xuất ngoài
  - Trạng thái nhị phân của mạch lật
- Trạng thái kế của mạch lật =  $F(\text{Trạng thái hiện tại}, \text{Các ngõ nhập ngoài})$
- **Thiết kế mạch tuần tự → Xác định dạng mạch lật và các Input của chúng**

# Thiết kế mạch tuần tự – Bước 1

- Đầu tiên phải **xác định dùng dạng mạch lật gì** (RS / JK / D / T)
- Lập **lược đồ các trạng thái** mạch lật dựa trên đặc tả mạch ban đầu
- Có 2 cách biểu diễn



# (Bước 1 – tiếp tục)

- Thay vì dùng lược đồ trạng thái, ta cũng có thể lập **bảng trạng thái** mạch lật

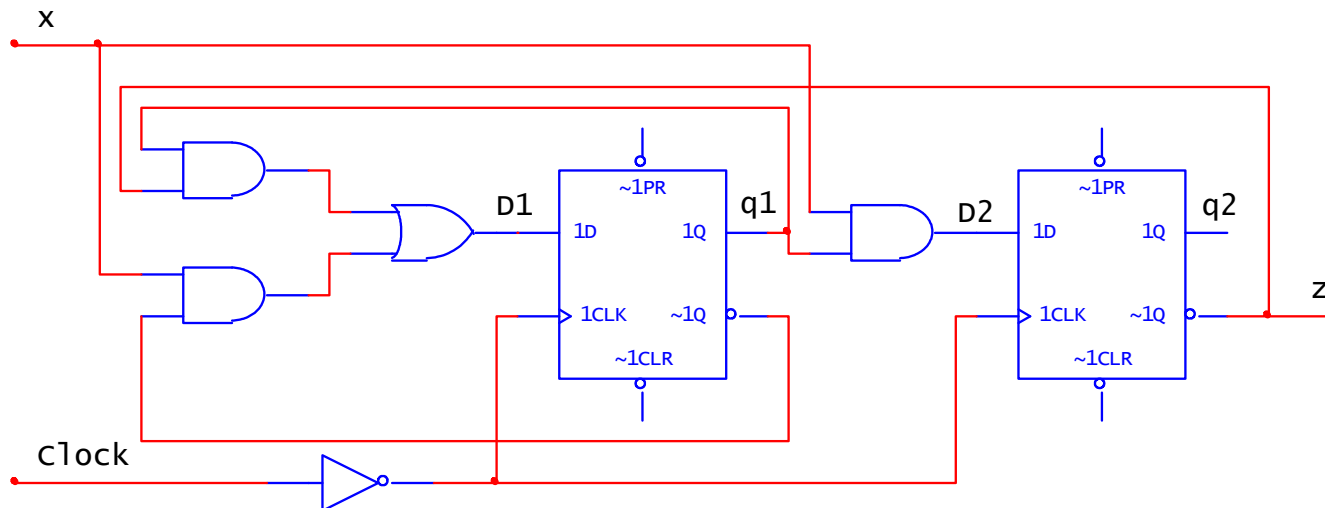
Trạng thái hiện tại $Q(t)$			Giá trị ngõ nhập ngoài	Trạng thái kế $Q(t+1)$		
Ngõ xuất mạch lật 1	...	Ngõ xuất mạch lật n	x	Ngõ xuất mạch lật 1	...	Ngõ xuất mạch lật n

- Trạng thái kế của mạch lật. Dựa trên mô tả để đại



# Thiết kế mạch tuần tự – Bước 2

- Lập **bảng kích thích**
  - Nhiệm vụ là phải xác định được **làm thế nào để có được ngõ nhập vào mạch lật từ ngõ nhập ngoài x**
  - Lưu ý ngõ nhập vào mạch lật  $\neq$  ngõ nhập ngoài
    - Ví dụ:  $x \neq D1, D2$



# Bảng kích thích

Mạch lật RS / SR			
Q(t)	Q(t+1)	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Mạch lật JK			
Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Mạch lật D		
Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Mạch lật T		
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0



# Thiết kế mạch tuần tự – Bước 3

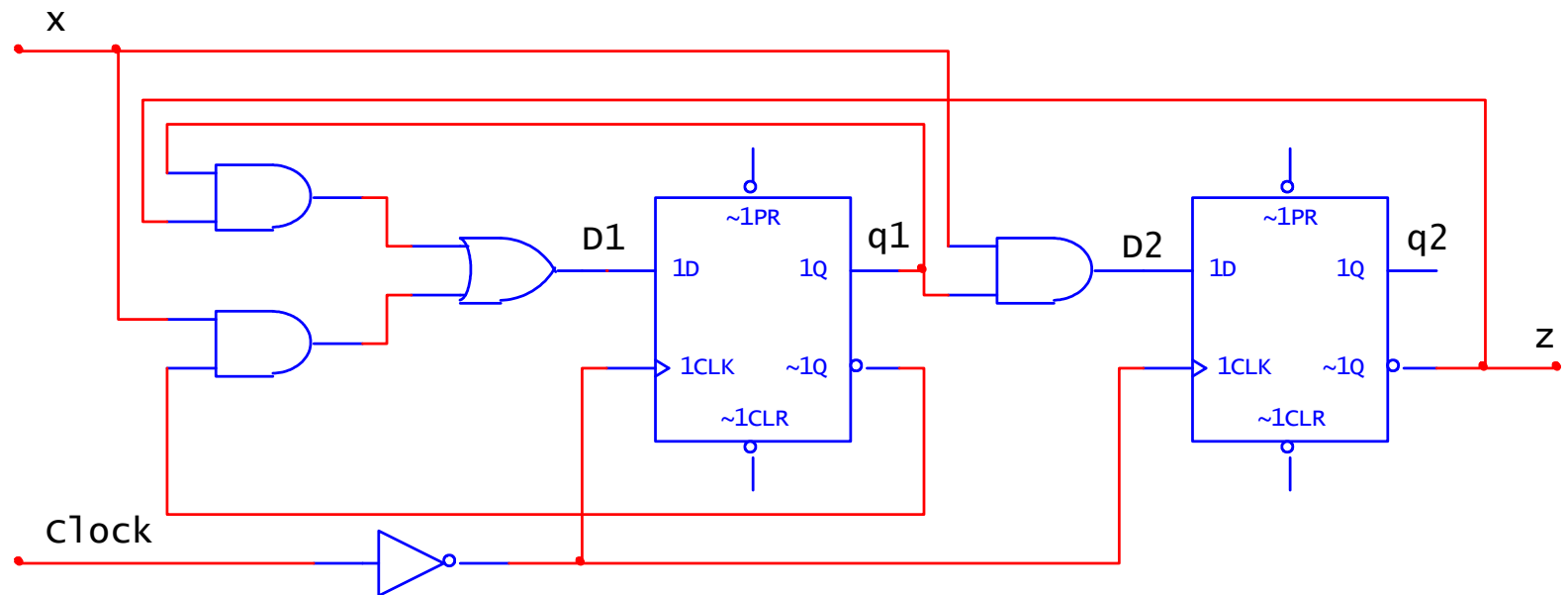
- Tìm phương trình đại số xác định ngõ nhập mạch lật từ bảng kích thích (Hàm ngược)
  - Có  $n$  mạch lật  $\rightarrow n$  ngõ ra mạch lật  $A_1 \dots A_n$
  - Suy ra phương trình ngõ nhập mạch lật có  $n + 1$  biến bao gồm:
    - $n$  biến  $A_1 \dots A_n$
    - 1 biến  $x$  (ngõ nhập ngoài)
  - Dùng biểu đồ Karnaugh + bảng kích thích để xác định phương trình hàm ngõ nhập mạch lật





# Thiết kế mạch tuần tự – Bước 4

- Vẽ sơ đồ mạch dựa trên phương trình hàm ngõ nhập



# Bài tập minh họa

- Xem ví dụ minh họa tại giáo trình “Kiến trúc máy tính” – Thầy Nguyễn Minh Tuấn, trang 42-45



# Một số bài tập thiết kế mạch

## Bài 1 – Digital Clock v.1

- Thiết kế đồng hồ với mặt số thể hiện các số từ 0 đến 7 và 2 nút bấm A, B. Nếu bấm nút A, số thể hiện tăng lên 1. Nếu bấm nút B, số thể hiện giảm đi 1
- Cần: Adder, MUX



# Một số bài tập thiết kế mạch

## Bài 2 – Digital Clock v.2

- Thiết kế đồng hồ bấm giây với mặt số thể hiện các số từ 00 đến 63 và 2 nút bấm A, B. Bấm nút A để start / stop. Khi đồng hồ đang ở trạng thái stop, bấm nút B sẽ xóa về 0
- Cần: Counter, MUX



# Một số bài tập thiết kế mạch

## Bài 3 – Digital Clock v.3

- Thiết kế đồng hồ bấm giây với mặt số thể hiện các số từ 00 đến 63 và 3 nút bấm A, B, C. Bấm nút A để start / stop. Khi đồng hồ đang ở trạng thái stop, bấm nút B sẽ tăng lên 1, bấm nút C sẽ giảm đi 1, bấm đồng thời B và C sẽ xoá về 00

