



Khoa  
**CÔNG NGHỆ THÔNG TIN**  
ĐH Khoa học Tự nhiên TP HCM

# Bài 06: Mạch số

**Phạm Tuấn Sơn**

**[ptson@fit.hcmus.edu.vn](mailto:ptson@fit.hcmus.edu.vn)**



# Mô hình phân tầng việc xử lý của máy tính

High Level Language  
Program (e.g., C)

*Compiler*

Assembly Language  
Program (e.g., MIPS)

*Assembler*

Machine Language  
Program (MIPS)

*Machine  
Interpretation*

Hardware Architecture Description  
(e.g. block diagrams)

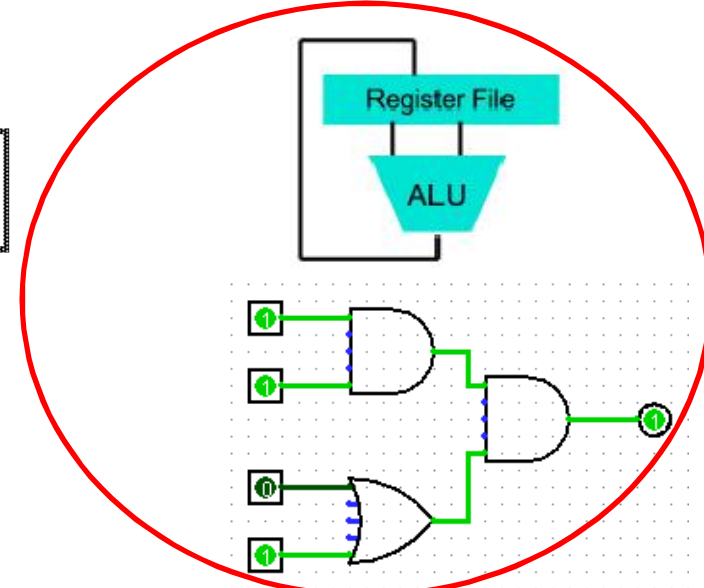
*Architecture  
Implementation*

Logic Circuit Description  
(Circuit Schematic Diagram)

```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

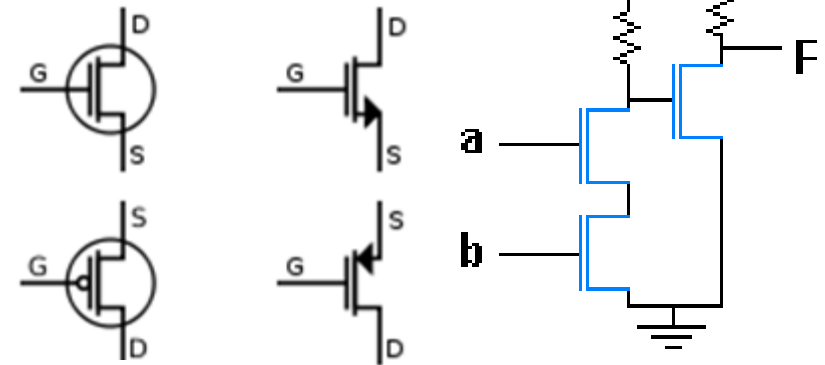
```
lw    $t0, 0($2)  
lw    $t1, 4($2)  
sw    $t1, 0($2)  
sw    $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```



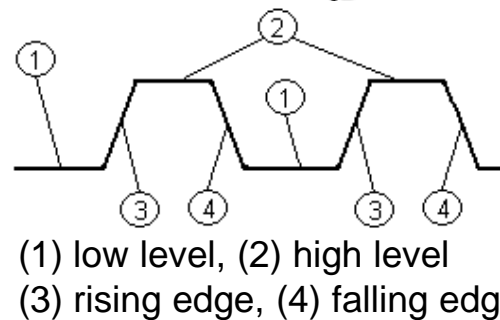
# Mạch số

- Transistor là linh kiện điện tử làm từ chất bán dẫn dùng để khuếch đại và chuyển tín hiệu điện
- MOSFET (metal-oxide-semiconductor field-effect transistor)
  - NMOSFET (n-type)
    - Nếu hiệu điện thế giữa G và S đủ lớn, thì D và S sẽ được nối (transistor sẽ có trạng thái "on")
  - PMOSFET (p-type)
    - Ngược lại NMOSFET
- Mạch số là thiết bị điện tử kết nối các linh kiện điện tử (như transistor) hoạt động ở 2 mức điện áp: cao và thấp



NMOS  
AND gate

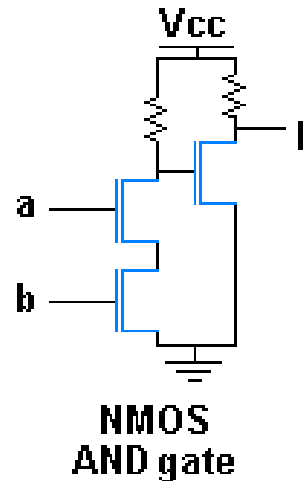
a	b	F
low	low	low
high	low	low
low	high	low
high	high	high










Công nghệ	Điện áp thấp	Điện áp cao	Ghi chú
CMOS	0V à $V_{CC}/2$	$V_{CC}/2$ à $V_{CC}$	$V_{CC}$ điện áp nguồn
TTL	0V à 0.8V	2V à $V_{CC}$	$V_{CC}$ : 4.75V à 5.25V
ECL	-1.175V à $-V_{EE}$	.75V à 0V	$V_{EE}$ : -5.2V $V_{CC}$ =Nối đất

# Cổng logic

- Các linh kiện điện tử thường kết nối với nhau thành các khối cơ bản
- Khối cơ bản nhất là các cổng logic với các giá trị luận lý (qui ước) 1 và 0 tương ứng với 2 mức điện thế cao và thấp



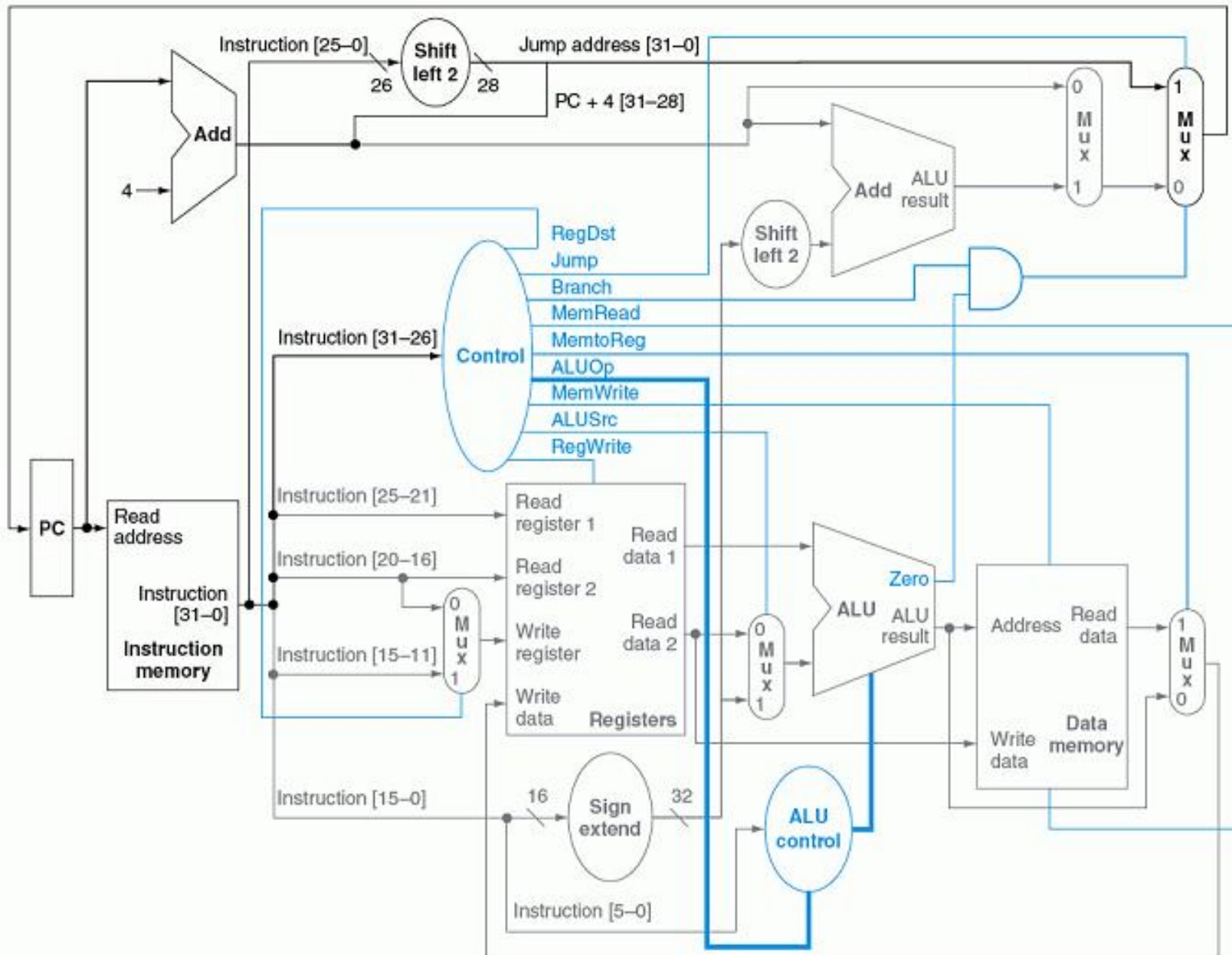
a	b	F (AND)
low (0)	low (0)	low (0)
high(1)	low (0)	low (0)
low (0)	high(1)	low (0)
high(1)	high(1)	high(1)

Tên cổng	Hình vẽ	Ký hiệu
AND		$x.y$
OR		$x+y$
XOR		$x \oplus y$
NOT		$\bar{x}$ (hoặc $x'$ )
NAND		$\overline{x.y}$
NOR		$\overline{x+y}$
NXOR		$\overline{x \oplus y}$

# Thiết kế logic

- Các cổng logic thường được kết nối với nhau thành các khối cao cấp hơn
- Các mạch cao cấp này gồm 2 loại
  - Mạch tổ hợp: kết nối các cổng logic sao cho kết quả của mạch chỉ phụ thuộc vào giá trị đầu vào tại thời điểm đang xét. Ví dụ: mạch adder, decoder, multiplexor, ALU,...
  - Mạch tuần tự: kết nối các cổng logic sao cho kết quả của mạch không chỉ phụ thuộc vào giá trị đầu vào tại thời điểm đang xét mà còn phụ thuộc vào trạng thái tại thời điểm trước đó của mạch. Ví dụ: mạch lật RS, JK, T, D,...
- Thông thường các mạch số (như mạch xử lý) được thiết kế ở mức logic (kết nối các khối cao cấp và các khối cơ bản), sau đó có thể sử dụng các kỹ thuật khác nhau để chuyển thành mạch số ở mức các linh kiện điện tử

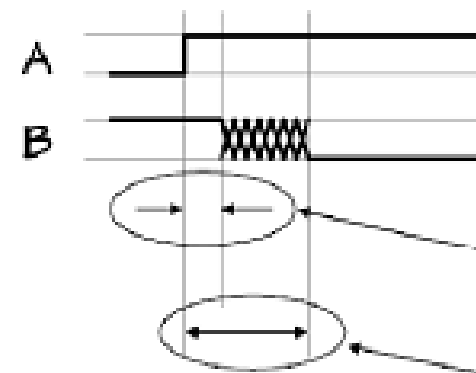
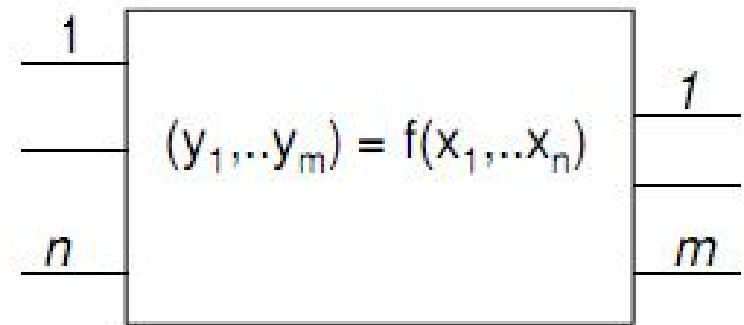
# Ví dụ mạch số thiết kế ở mức logic





# Mạch tổ hợp

- Kết nối các cổng logic sao cho kết quả của mạch tại một thời điểm chỉ phụ thuộc vào giá trị đầu vào tại thời điểm đó
- Gồm  $n$  ngõ vào,  $m$  ngõ ra. Ngõ ra là một hàm luận lý của các ngõ vào
- Luôn có một độ trễ giữa thời điểm tín hiệu vào ổn định với thời điểm tín hiệu ra ổn định (propagation delay)





# Thiết kế mạch tổ hợp

- 3 bước

- Lập bảng chân trị từ yêu cầu
- Xây dựng hàm luận lý từ bảng chân trị
- Vẽ sơ đồ mạch luận lý và thử nghiệm



# Lập bảng chân trị

- Từ yêu cầu, thiết lập tất cả các giá trị có thể có của các đầu vào (n giá trị đầu vào sẽ có  $2^n$  trường hợp) và giá trị tương ứng của các giá trị đầu ra cho từng trường hợp

- Ví dụ

- Yêu cầu: thiết kế mạch tổ hợp có 3 đầu vào và 1 đầu ra, sao cho giá trị logic ở đầu ra là giá trị nào chiếm đa số trong các đầu vào
- Gọi x, y, z là các đầu vào; f là đầu ra
- Xét tất cả các khả năng có thể có của x, y, z

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

- Khó. Đôi khi phải tự đặt thêm nhiều biến trạng thái mới có thể mô hình hóa được yêu cầu



# Xây dựng hàm luận lý (1/2)

- SOP (Sum Of Products)

– Với bảng chân trị cho mạch n đầu vào  $x_1, \dots, x_n$  và một 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 bằng 1, viết một tích dạng  $u_1.u_2\dots u_n$  trong đó

$$u_i = \begin{cases} x_i & \text{nếu } x_i = 1 \\ \overline{x_i} & \text{nếu } x_i = 0 \end{cases}$$

- Cộng các tích lại thành tổng, đó chính là công thức của  $f$



# Ví dụ SOP



x	y	z	f	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	à $\bar{x}.y.z$
1	0	0	0	
1	0	1	1	à $x.\bar{y}.z$
1	1	0	1	à $x.y.\bar{z}$
1	1	1	1	à $x.y.z$

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

# Xây dựng hàm luận lý (2/2)

- POS (Product Of Sums)

– Trường hợp số hàng có giá trị đầu ra bằng 1 nhiều hơn bằng 0, có thể đặt biến  $g = \text{NOT}(f)$ , sau đó viết công thức dạng SOP cho  $g$ , rồi lấy NOT để có công thức dạng POS (tích của 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

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

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



# Rút gọn hàm luận lý (1/2)

- Mục đích: đơn giản hóa hàm logic để sơ đồ mạch sử dụng ít cổng hơn
- Phương pháp đại số Bool

Các luật cơ bản		
$A \bullet B = B \bullet A$	$A + B = B + A$	Luật giao hoán
$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$	$A + (B \bullet C) = (A + B) \bullet (A + C)$	Luật phân phối
$1 \bullet A = A$	$0 + A = A$	Luật đồng nhất
$A \bullet \bar{A} = 0$	$A + \bar{A} = 1$	Phần tử nghịch đảo
Other laws		
$0 \bullet A = 0$	$1 + A = 1$	Luật kết hợp Luật DeMorgan
$A \bullet A = A$	$A + A = A$	
$\overline{A \bullet (B \bullet C)} = \bar{A} + \bar{B} + \bar{C}$	$\overline{A + (B + C)} = \bar{A} \bullet \bar{B} \bullet \bar{C}$	
$\overline{A \bullet B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \bullet \bar{B}$	



## Ví dụ rút gọn bằng đại số Bool

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

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

$$= \quad \quad y.z \quad \quad \quad x.z \quad \quad \quad x.y$$

$$= z.(x+y) + x.y$$

# Rút gọn hàm luận lý (2/2)

## • Phương pháp biểu đồ Karnaugh

- Biểu đồ Karnaugh là một ma trận gồm  $2^n$  ô, tượng trưng cho tất cả các trường hợp có thể có của n đầu vào
- Mỗi ô của biểu đồ Karnaugh sẽ mang giá trị hàm luận lý của trường hợp tương ứng
- Phương pháp biểu đồ Karnaugh được sử dụng trong trường hợp  $n \leq 4$

AB			
00	01	11	10
	1		1

(a)  $F = A\bar{B} + \bar{A}B$

A	BC			
	00	01	11	10
0			1	1
1				1

(b)  $F = \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$

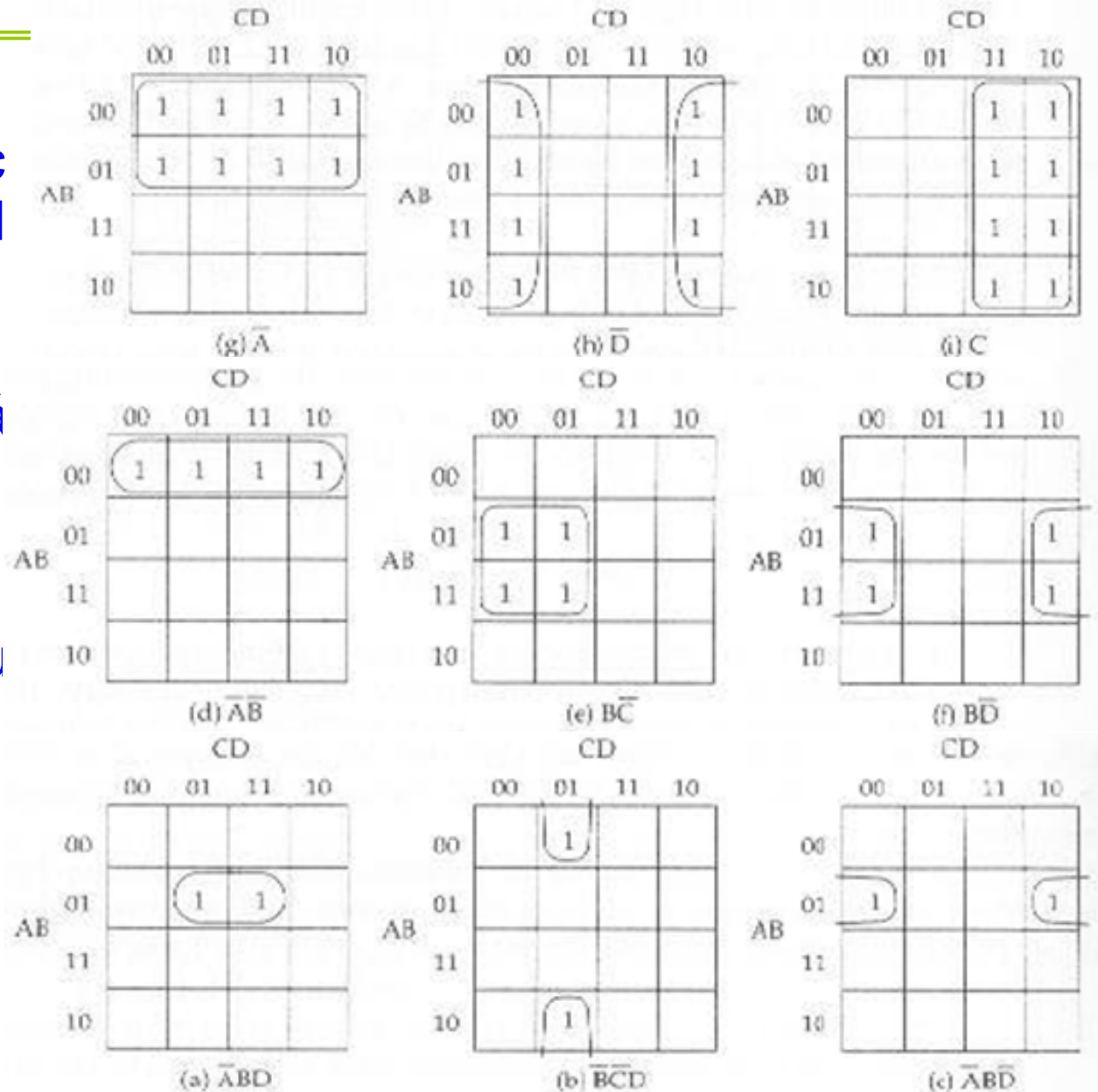
	CD			
	00	01	11	10
00			1	
01				
11	1			
10		1		

(c)  $F = \bar{A}\bar{B}CD + A\bar{B}\bar{C}D + AB\bar{C}\bar{D}$



# Nguyên tắc rút gọn biểu đồ Karnaugh

1. Gom nhóm ( $2^n$ , 8, 4, 2) các ô mang giá trị 1
2. ...cho đến khi nào không còn ô mang giá trị 1 nào chưa được gom nhóm (1 ô có thể thuộc nhiều nhóm khác nhau)
3. Kết quả gom nhóm cuối cùng có thể khác nhau





# Cơ sở của việc gom nhóm

- Bất kỳ 2 ô nào lân cận nhau đều chỉ khác nhau giá trị 1 đầu vào
- Do đó, nếu 2 ô lân cận nhau đều mang giá trị 1 thì có thể gom lại và bỏ đi đầu vào khác giá trị
- Ví dụ:  $\overline{A}B\overline{C}D + \overline{A}BCD = \overline{A}BD$
- Tương tự cho 4, 8, ...,  $2^n$  ô lân cận nhau

## Ví dụ rút gọn bằng biểu đồ Karnaugh








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

x\yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1

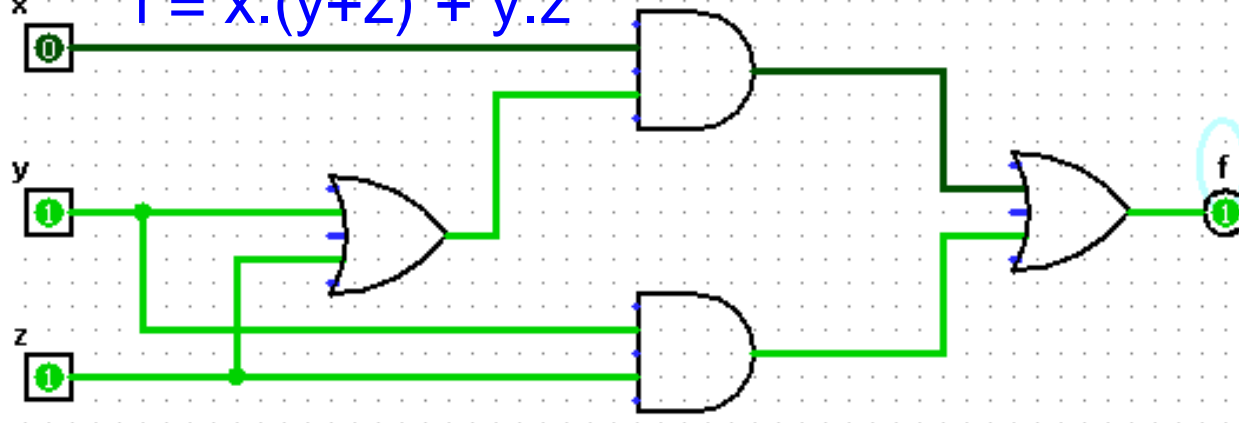
$$f = x.z + x.y + y.z = x.(y + z) + y.z$$

# Vẽ sơ đồ mạch luận lý và thử nghiệm

- Ánh xạ các hàm bool thành các cổng logic tương ứng

Hàm Bool	Tên cổng	Hình vẽ
$x.y$	AND	
$x+y$	OR	
$x \oplus y$ ( $\overline{x}.y + x.\overline{y}$ )	XOR	
$\overline{x}$ (hoặc $x'$ )	NOT	
$\overline{x.y}$	NAND	
$\overline{x+y}$	NOR	
$\overline{x \oplus y}$	NXOR	

- Ví dụ:  $f = x.(y+z) + y.z$





# Một số mạch tổ hợp

- Adder
- Encoder & Decoder
- Multiplexor & Demultiplexor
- ALU

# Mạch cộng (1/3)

- Mạch nửa cộng (Half - adder)

- Bảng chân trị

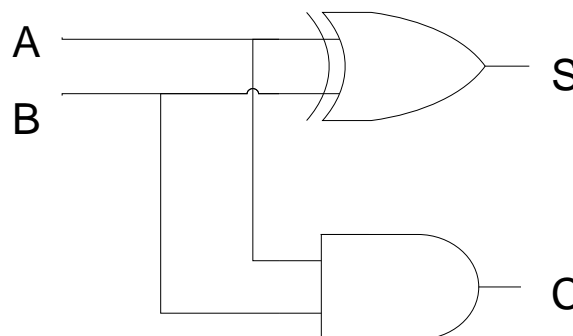
Cộng 1 bit			
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Hàm luận lý

- $\text{Sum} = \bar{A}B + A\bar{B}$

- $\text{Carry} = AB$

- Sơ đồ mạch



# Mạch cộng (2/3)

## • Mạch toàn cộng (Full - adder)

### – Bảng chân trị

Cộng có nhớ				
$C_i$	A	B	Sum	$C_o$
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

### – Hàm luận lý

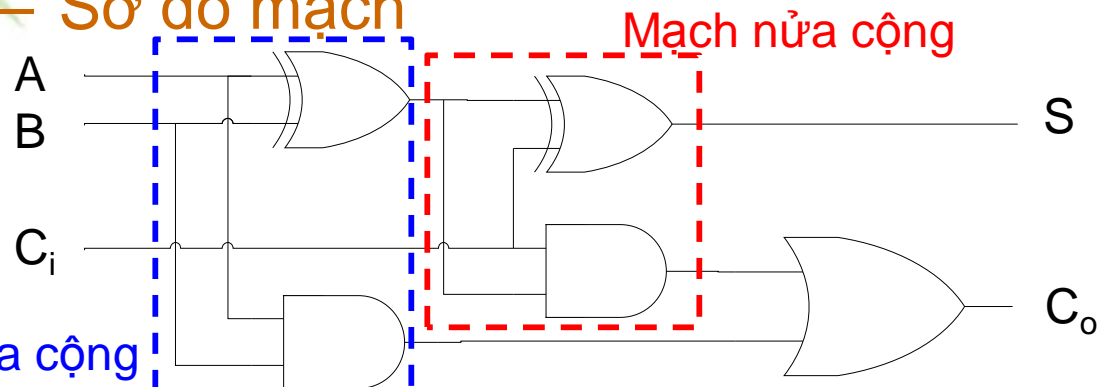
$S$

$C/AB$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$C_o$

$C/AB$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

### – Sơ đồ mạch



$$\text{Sum} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}C + A\bar{B}\bar{C}$$

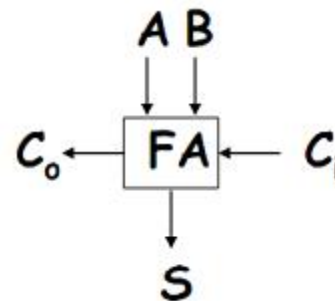
$$= A \oplus B \oplus C$$

$$C_o = AB + AC_i + BC_i$$

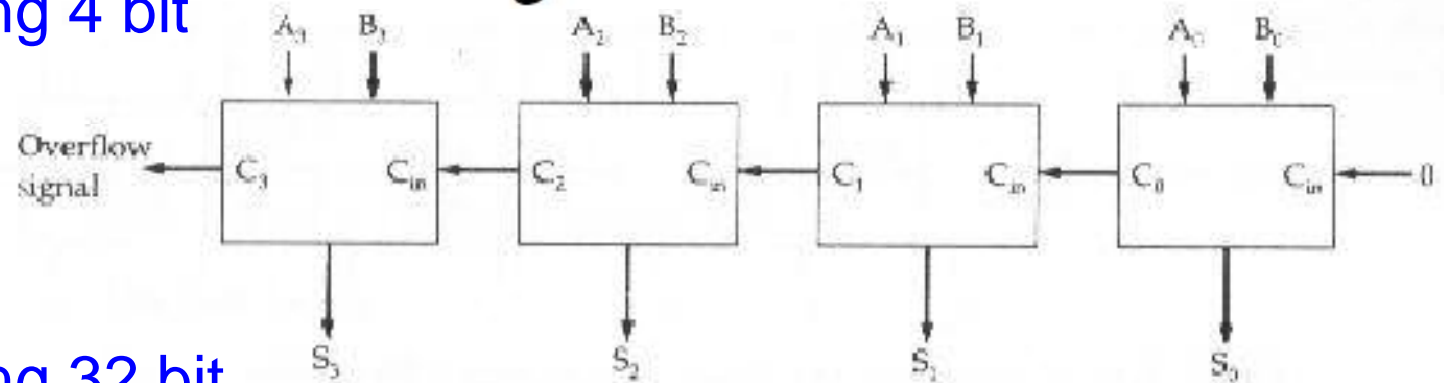


# Mạch cộng (3/3)

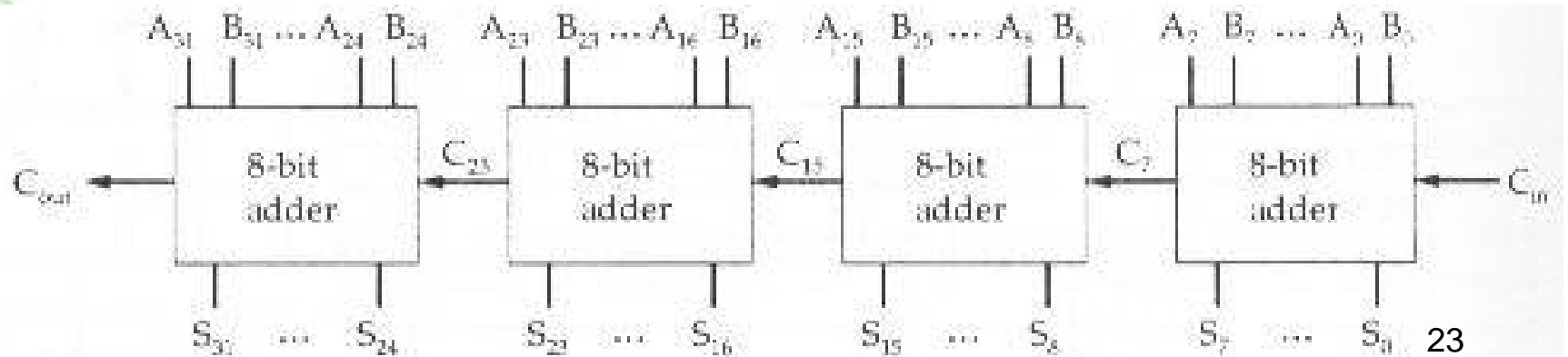
- Mạch cộng 1 bit



- Mạch cộng 4 bit

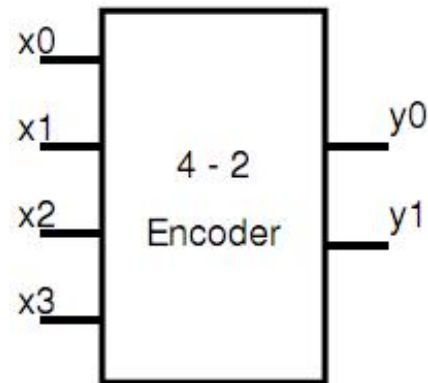


- Mạch cộng 32 bit



# Mạch mã hóa (1/3)

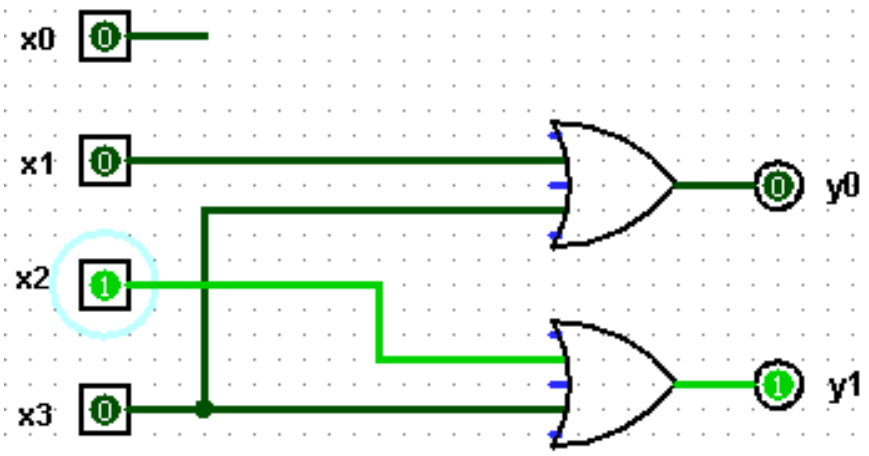
- Chỉ có một ngõ vào được bật, tổ hợp giá trị các ngõ ra sẽ cho biết ngõ vào nào được bật
- Mạch mã hóa 4-2 (4-2 Encoder)



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

$$y0 = x1 + x3$$

$$y1 = x2 + x3$$



# Mạch mã hóa (2/3)

## Mạch mã hóa ưu tiên 4-3 (4-3 Priority Encoder)

- Các đầu vào được xem như có độ ưu tiên. Ví dụ, nếu đầu vào  $x_2$  có giá trị 1 thì giá trị của  $x_1$  và  $x_0$  không được xét đến
- Để  $x_0$  tham gia vào mạch, qui định trường hợp tất cả đầu vào bằng 0 sẽ cho ra tất cả đầu ra bằng 0 à cần thêm 1 giá trị đầu ra

$x_0$	$x_1$	$x_2$	$x_3$	$y_2$	$y_1$	$y_0$
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

$x_0 \ x_1$	$x_2 \ x_3$	00	01	11	10
01	0	0	0	0	0
11	0	0	0	0	0
10	1	1	1	1	1
00	0	0	0	0	1

$$y_0 = (x_2 + x_0.\overline{x_1}).\overline{x_3}$$

$x_0 \ x_1$	$x_2 \ x_3$	00	01	11	10
01	0	0	0	0	0
11	0	0	0	0	0
10	1	1	1	1	1
00	0	1	1	1	0

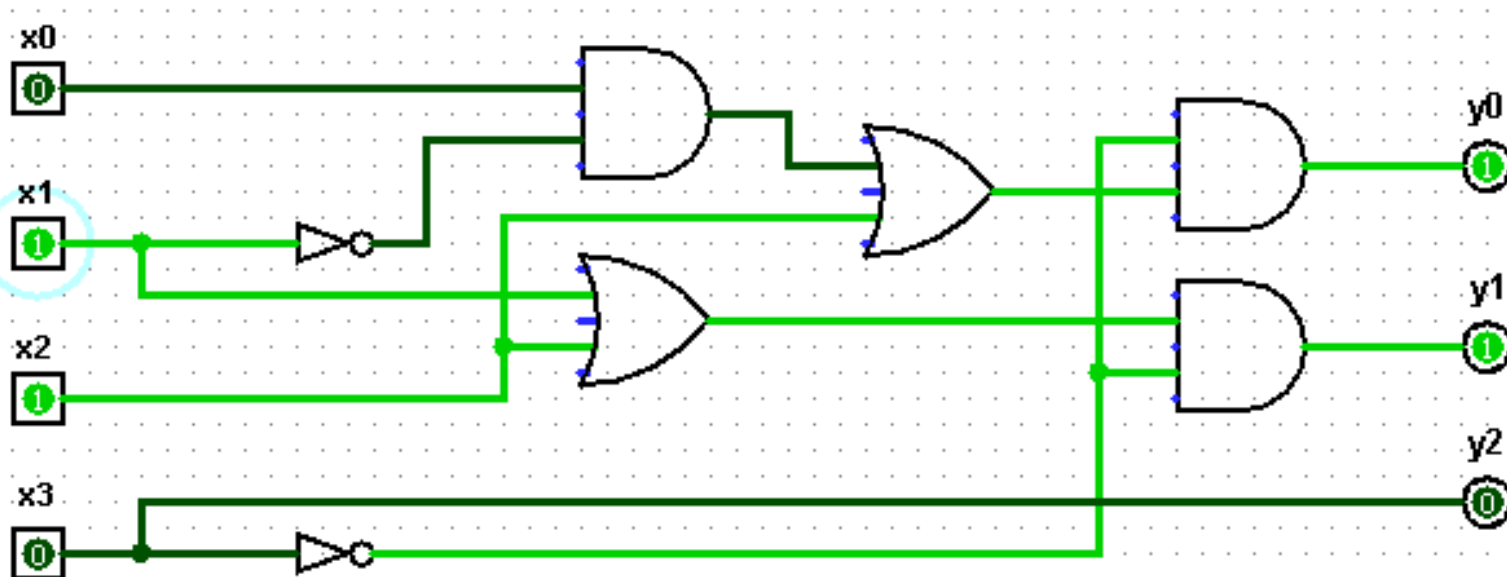
$$y_1 = (x_2 + x_1).\overline{x_3}$$

$$y_2 = x_3$$

# Mạch mã hóa (3/3)

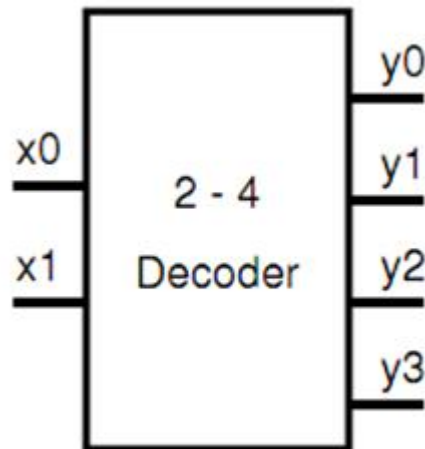
## • Mạch mã hóa ưu tiên 4-3 (4-3 Priority Encoder)

- $y_0 = (x_2 + x_0.\overline{x_1}).\overline{x_3}$
- $y_1 = (x_2 + x_1).\overline{x_3}$
- $y_2 = x_3$



## Mạch giải mã (1/3)

- Ngược lại với mạch mã hóa, ứng với mỗi tổ hợp của các ngõ vào sẽ cho biết duy nhất ngõ ra nào được bật
- Mạch giải mã 2-4 (2-4 decoder)



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

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

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

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

$$y3 = x1.x0$$



# Mạch giải mã (2/3)

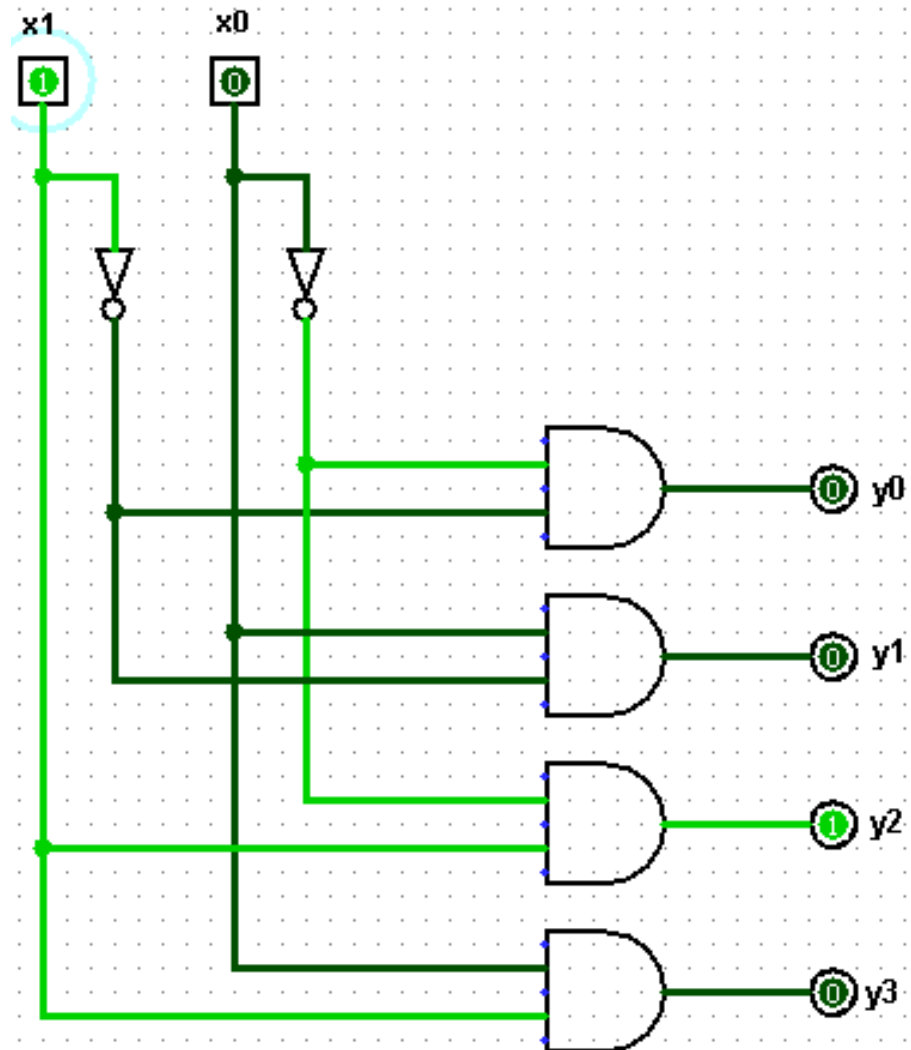
- Mạch giải mã 2-4  
(2-4 decoder)

$$y_0 = \overline{x_1} \cdot \overline{x_0}$$

$$y_1 = \overline{x_1} \cdot x_0$$

$$y_2 = x_1 \cdot \overline{x_0}$$

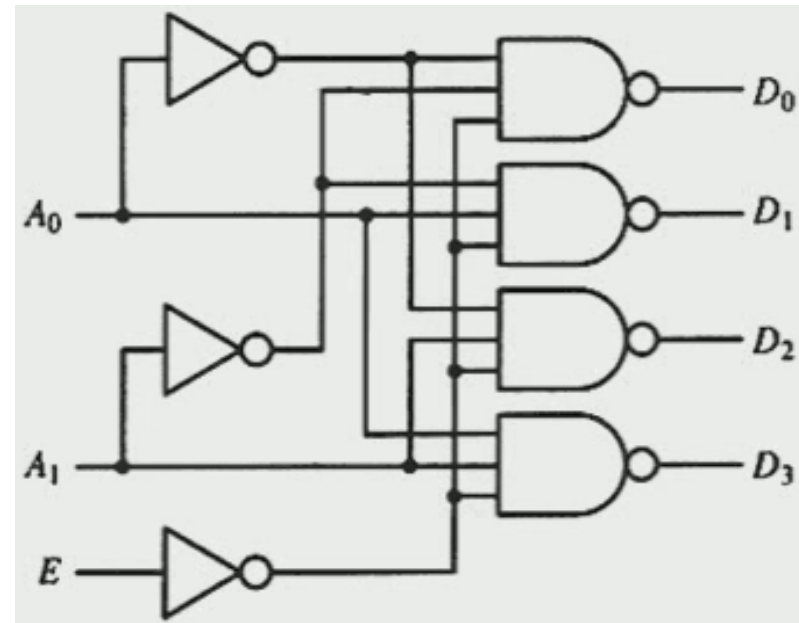
$$y_3 = x_1 \cdot x_0$$



# Mạch giải mã (3/3)

- Mạch giải mã 2-4 với tín hiệu Enable

$E$	$A_1$	$A_0$	$D_0$	$D_1$	$D_2$	$D_3$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

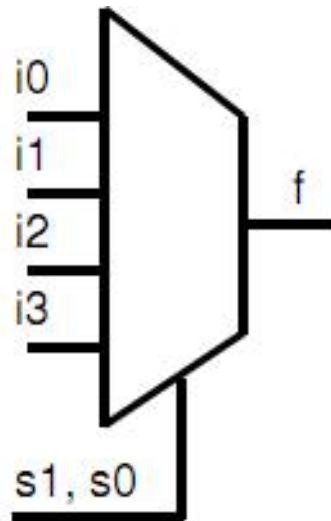
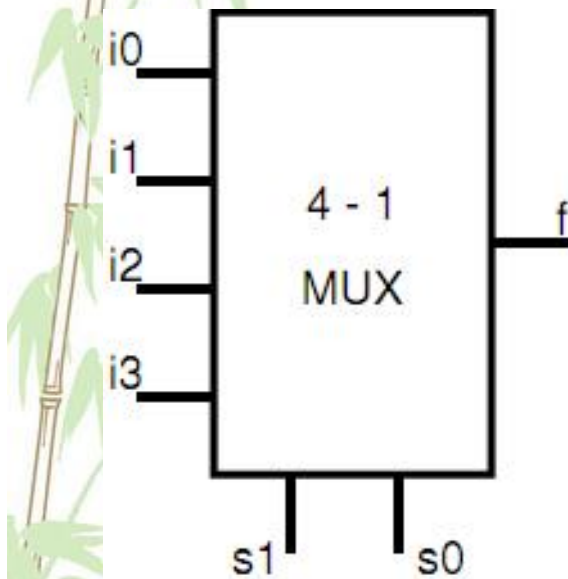




# MUX – Multiplexor (1/4)

- 4-1 multiplexor

– Tín hiệu  $s1$  và  $s0$  dùng để lựa chọn xem tín hiệu nào trong các ngõ vào được chuyển đến ngõ ra  $f$



$s1$	$s0$	$f$
0	0	$i0$
0	1	$i1$
1	0	$i2$
1	1	$i3$



# MUX – Multiplexor (2/4)

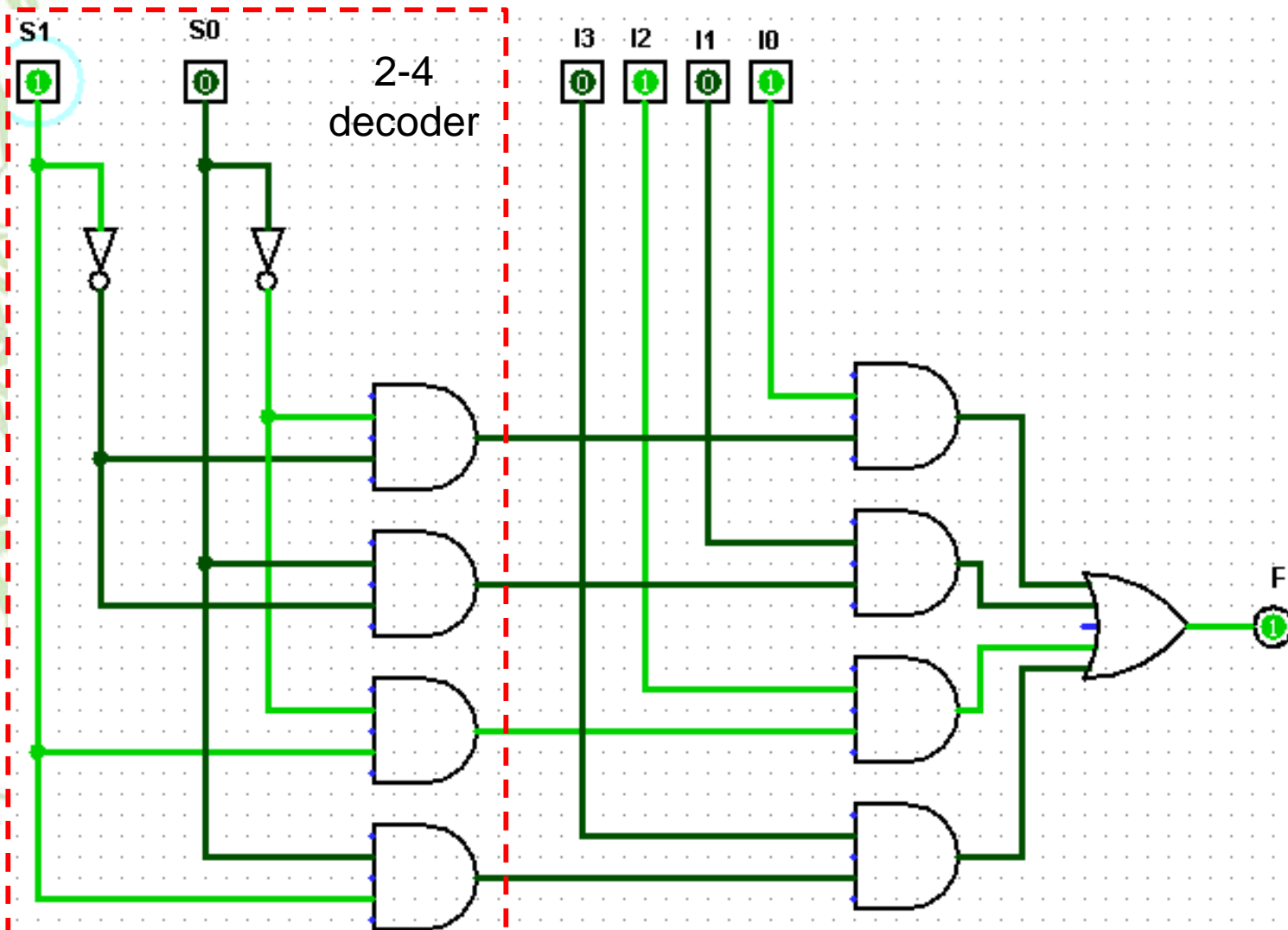
- 4-1 multiplexor

$S_1 S_0$	$I_3$	$I_2$	$I_1$	$I_0$	$F$	$S_1 S_0$	$I_3$	$I_2$	$I_1$	$I_0$	$F$	$S_1 S_0$	$I_3$	$I_2$	$I_1$	$I_0$	$F$	$S_1 S_0$	$I_3$	$I_2$	$I_1$	$I_0$	$F$
0 0	0	0	0	0	0	0 1	0	0	0	0	0	1 0	0	0	0	0	0	1 1	0	0	0	0	0
	0	0	0	1	1		0	0	0	1	0		0	0	0	1	0		0	0	0	1	0
	0	0	1	0	0		0	0	1	0	1		0	0	1	0	0		0	0	1	0	0
	0	0	1	1	1		0	0	1	1	1		0	0	1	1	0		0	0	1	1	0
	0	1	0	0	0		0	1	0	0	0		0	1	0	0	1		0	1	0	0	0
	0	1	0	1	1		0	1	0	1	0		0	1	0	1	1		0	1	0	1	0
	0	1	1	0	0		0	1	1	0	1		0	1	1	0	1		0	1	1	0	0
	0	1	1	1	1		0	1	1	1	1		0	1	1	1	1		0	1	1	1	0
	1	0	0	0	0		1	0	0	0	0		1	0	0	0	0		1	0	0	0	1
	1	0	0	1	1		1	0	0	1	0		1	0	0	1	0		1	0	0	1	1
	1	0	1	0	0		1	0	1	0	1		1	0	1	0	0		1	0	1	0	1
	1	0	1	1	1		1	0	1	1	1		1	0	1	1	0		1	0	1	1	1
	1	1	0	0	0		1	1	0	0	0		1	1	0	0	1		1	1	0	0	1
	1	1	0	1	1		1	1	0	1	0		1	1	0	1	1		1	1	0	1	1
	1	1	1	0	0		1	1	1	0	1		1	1	1	0	1		1	1	1	0	1
	1	1	1	1	1		1	1	1	1	1		1	1	1	1	1		1	1	1	1	1

# MUX – Multiplexor (3/4)

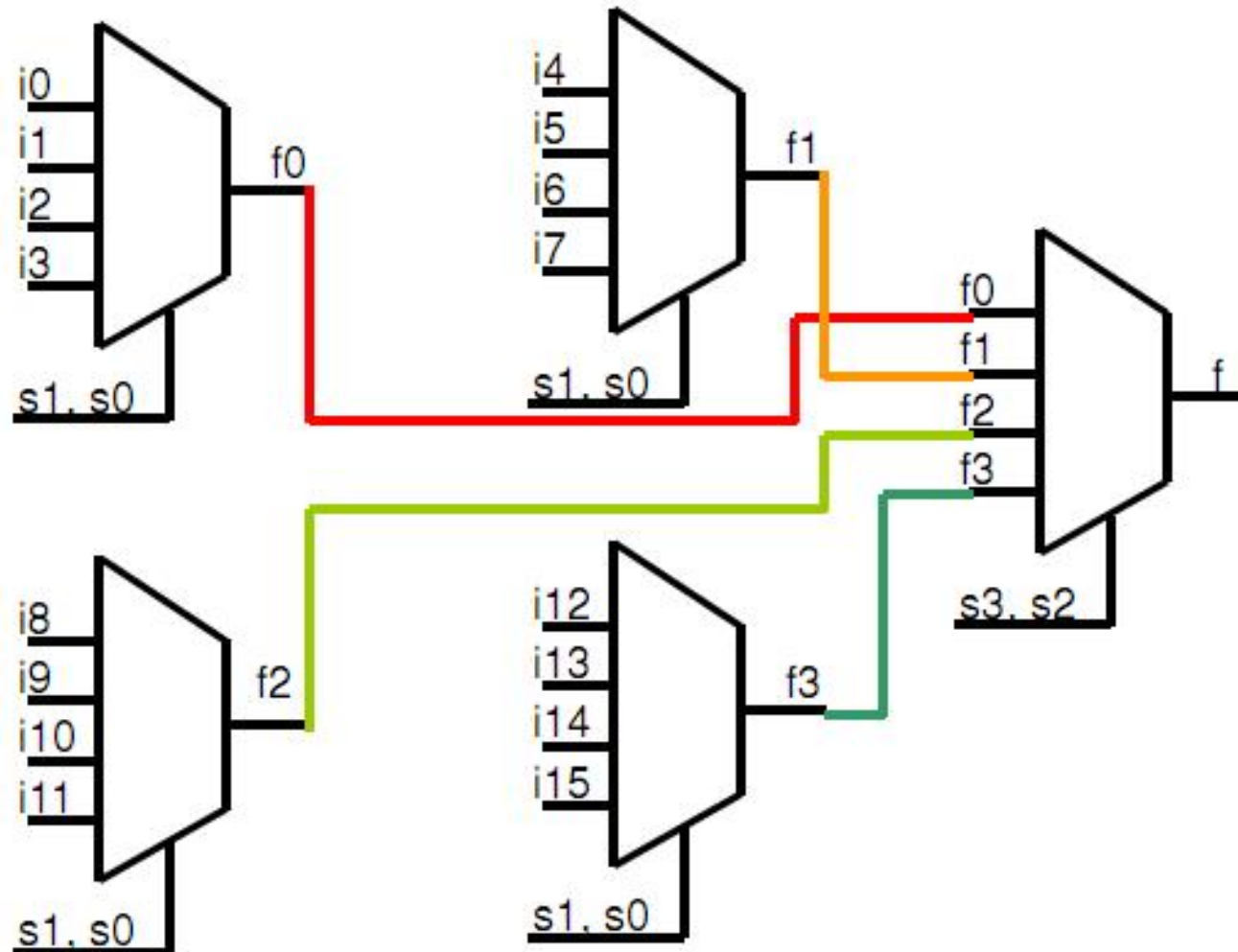
- 4-1 multiplexor

$$F = \overline{S_1}\overline{S_0}I_0 + \overline{S_1}S_0I_1 + S_1\overline{S_0}I_2 + S_1S_0I_3$$



# MUX – Multiplexor (4/4)

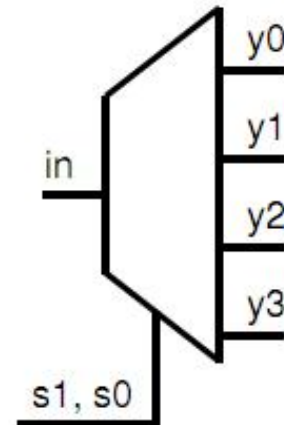
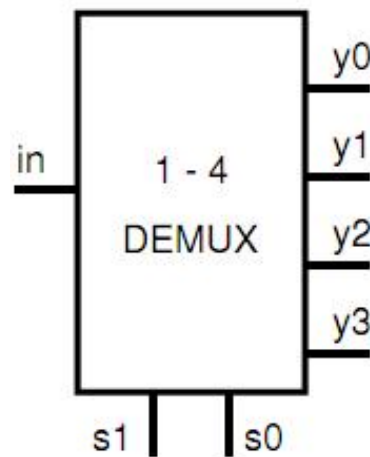
- 16-1 MUX



# DEMUX – Demultiplexor (1/3)

- 1-4 demultiplexor

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



# DEMUX – Demultiplexor (2/3)

## • 1-4 demultiplexor

$S_1$	$S_0$	$I_0$	$F_3$	$F_2$	$F_1$	$F_0$
0	0	0	0	0	0	0
		1	0	0	0	1

$S_1$	$S_0$	$I_0$	$F_3$	$F_2$	$F_1$	$F_0$
0	1	0	0	0	0	0
		1	0	0	1	0

$S_1$	$S_0$	$I_0$	$F_3$	$F_2$	$F_1$	$F_0$
1	0	0	0	0	0	0
		1	0	1	0	0

$S_1$	$S_0$	$I_0$	$F_3$	$F_2$	$F_1$	$F_0$
1	1	0	0	0	0	0
		1	1	0	0	0

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

$$y0 = \overline{s1} \cdot \overline{s0} \cdot in$$

$$y1 = \overline{s1} \cdot s0 \cdot in$$

$$y2 = s1 \cdot \overline{s0} \cdot in$$

$$y3 = s1 \cdot s0 \cdot in$$

# DEMUX – Demultiplexor (3/3)

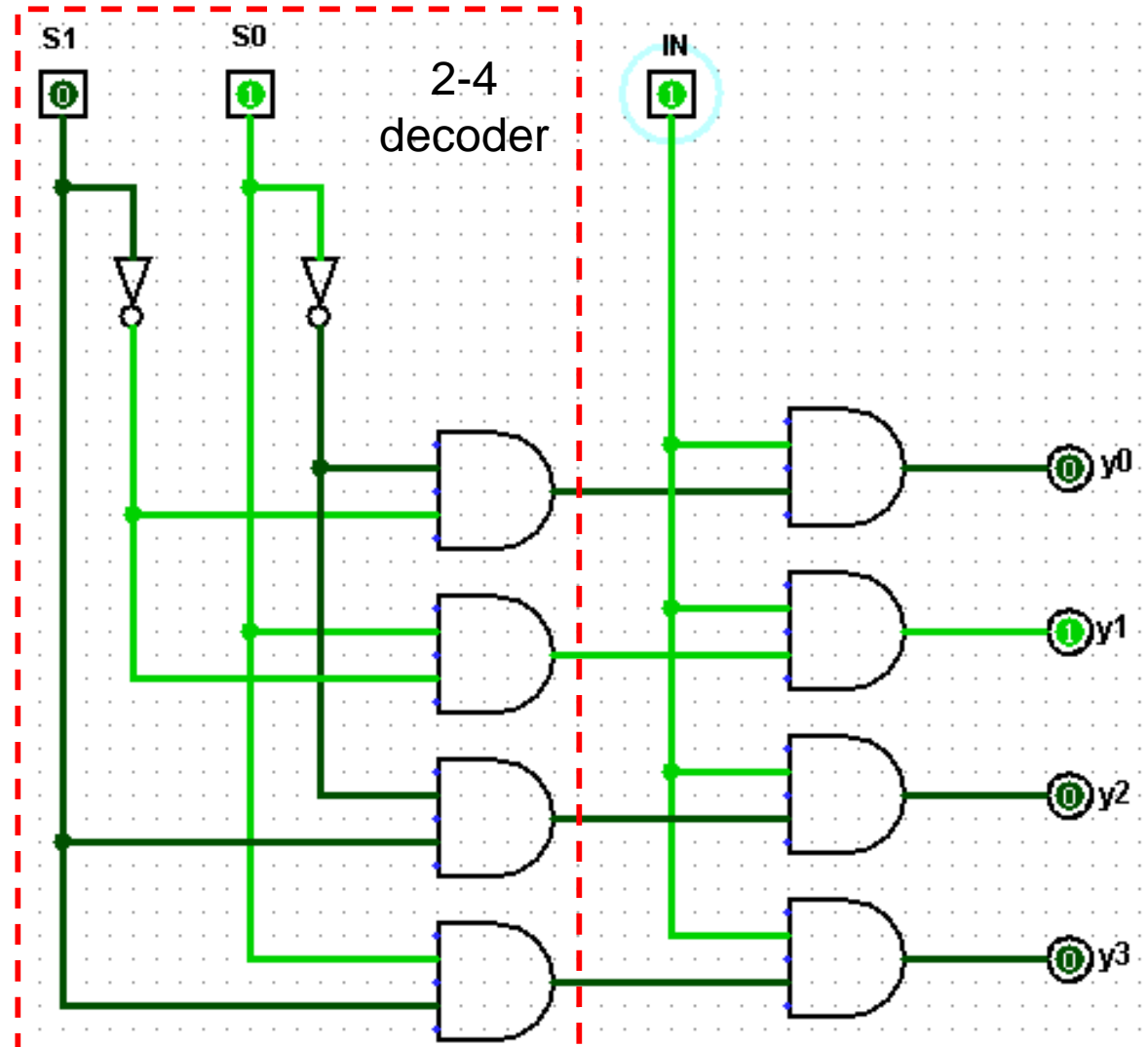
## • 1-4 demultiplexor

$$y0 = \overline{s1}.\overline{s0}.in$$

$$y1 = s1.\overline{s0}.in$$

$$y2 = s1.s0.in$$

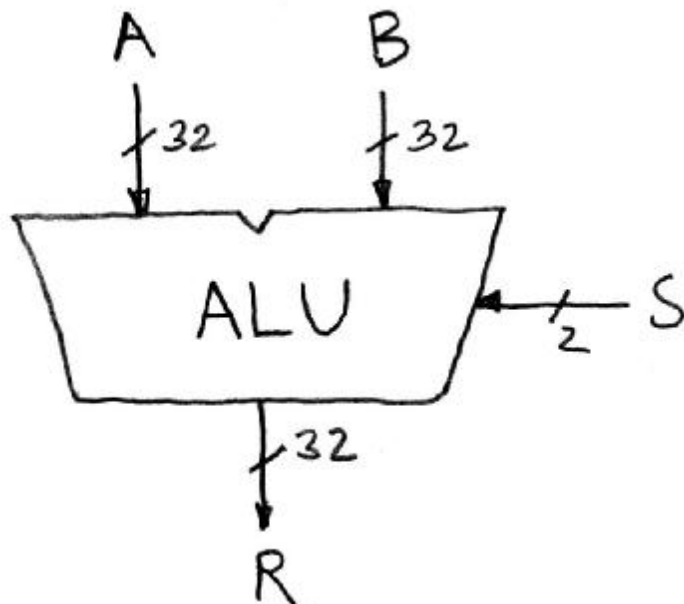
$$y3 = \overline{s1}.s0.in$$





# Thiết kế ALU đơn giản

- ALU đơn giản được thiết kế cho phép lựa chọn thực hiện các phép toán ADD, SUB, AND và OR



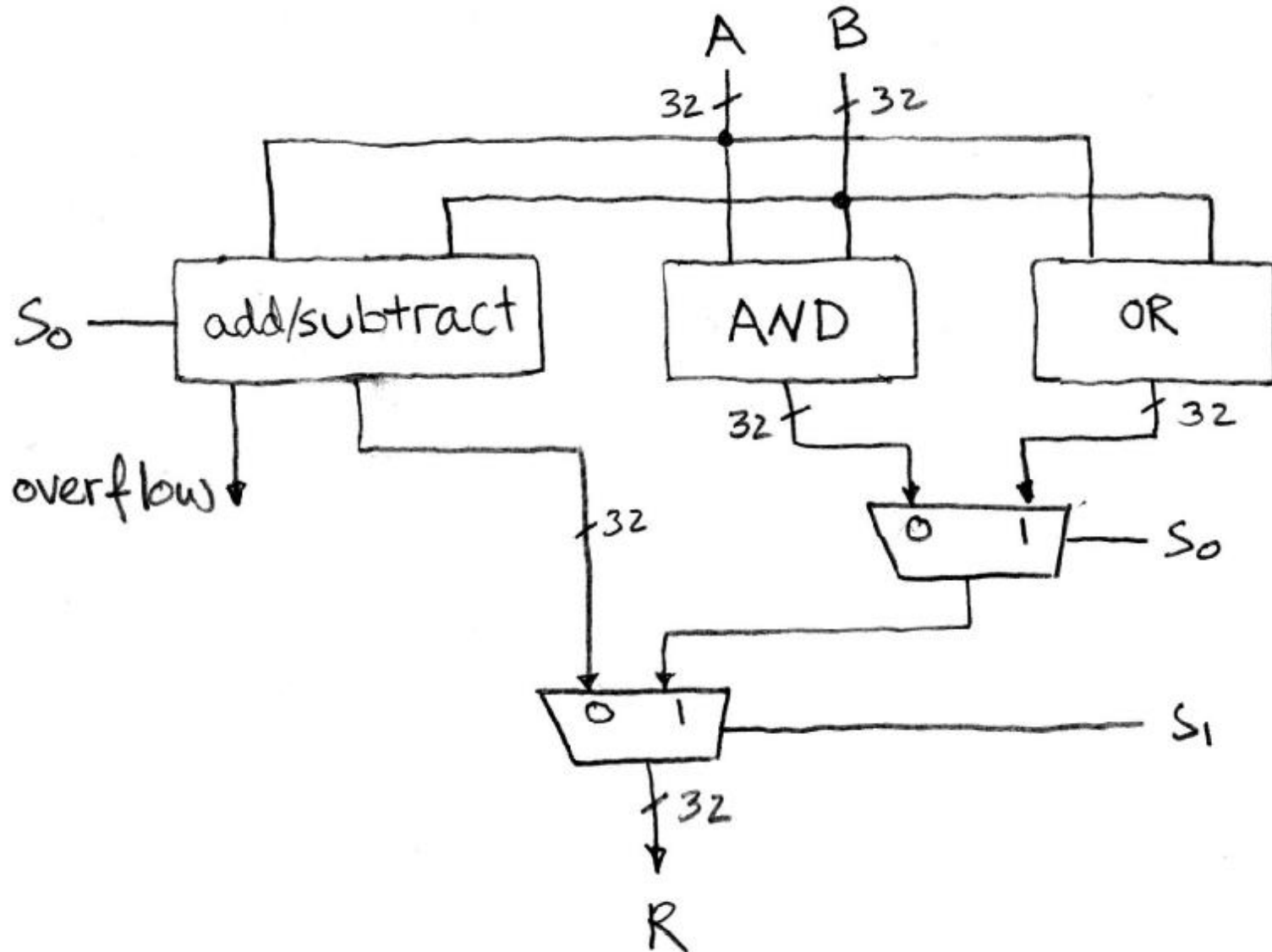
$S=00 \rightarrow R=A+B$

$S=01 \rightarrow R=A-B$

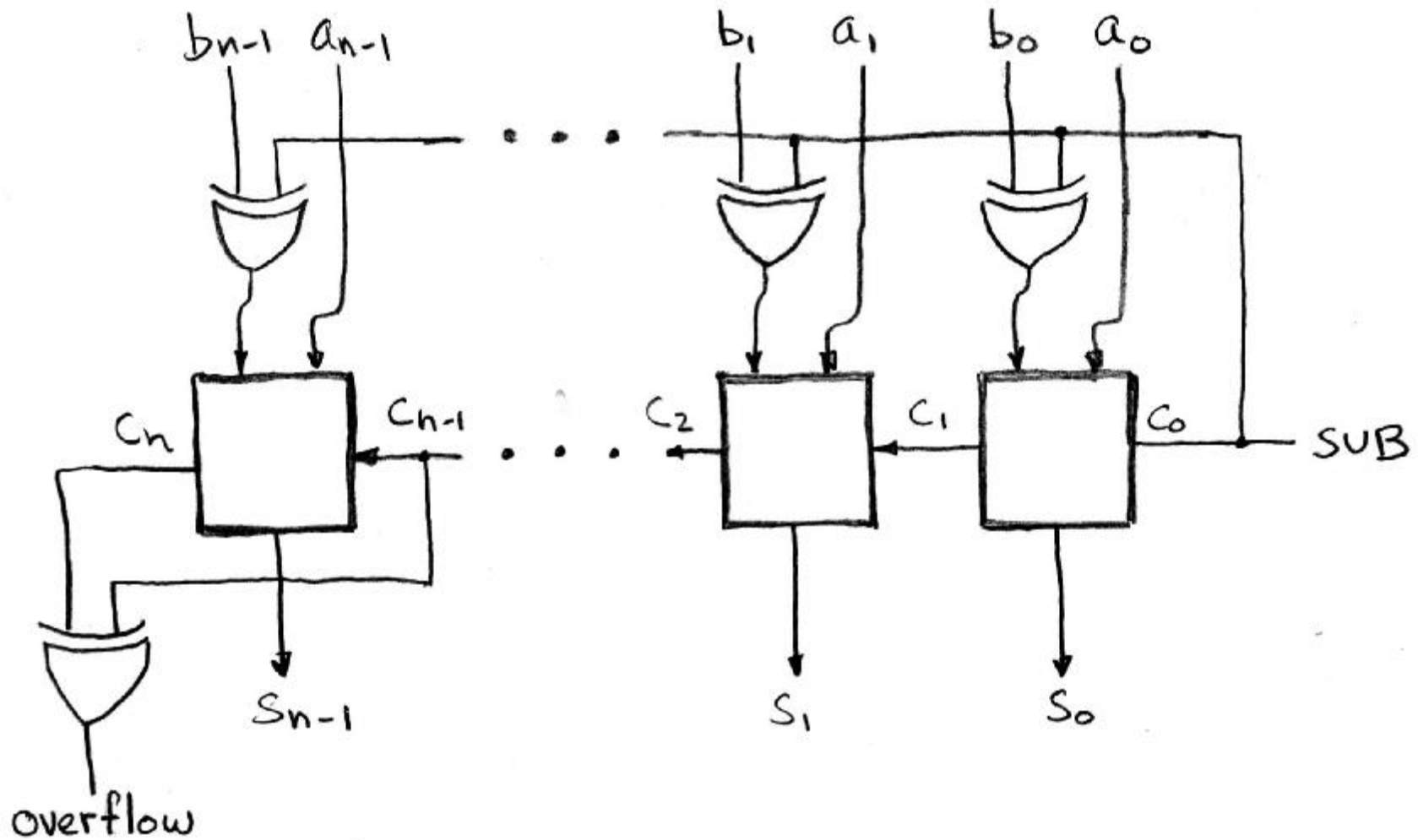
$S=10 \rightarrow R=A \& B$

$S=11 \rightarrow R=A | B$

# Thiết kế ALU đơn giản



# Thiết kế add/subtract

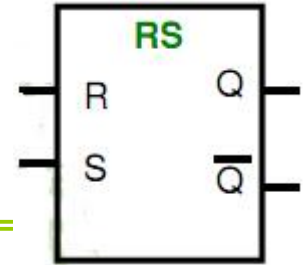




# Mạch tuần tự

- Kết nối các cổng logic sao cho kết quả của mạch không chỉ phụ thuộc vào giá trị đầu vào tại thời điểm đang xét mà còn phụ thuộc vào trạng thái tại thời điểm trước đó của mạch
- Mạch tuần tự có khả năng “ghi nhớ các trạng thái trong quá khứ”
- Mạch lật (Flip-flop) là mạch tuần tự cơ bản nhất
  - Mạch lật RS, JK, D, T,...
  - Có chức năng lưu trữ 1 bit nhớ

# Mạch lật RS



R	S	Q	Q'
0	0	Q	Q'
0	1	1	0
1	0	0	1
1	1	-	-

Q <sub>n</sub> \ RS	00	01	11	10
0		1		
1	1	1		

$$Q_{n+1} = \overline{R}.S + \overline{R}.Q_n$$

$$= \overline{R}.(S + Q_n)$$

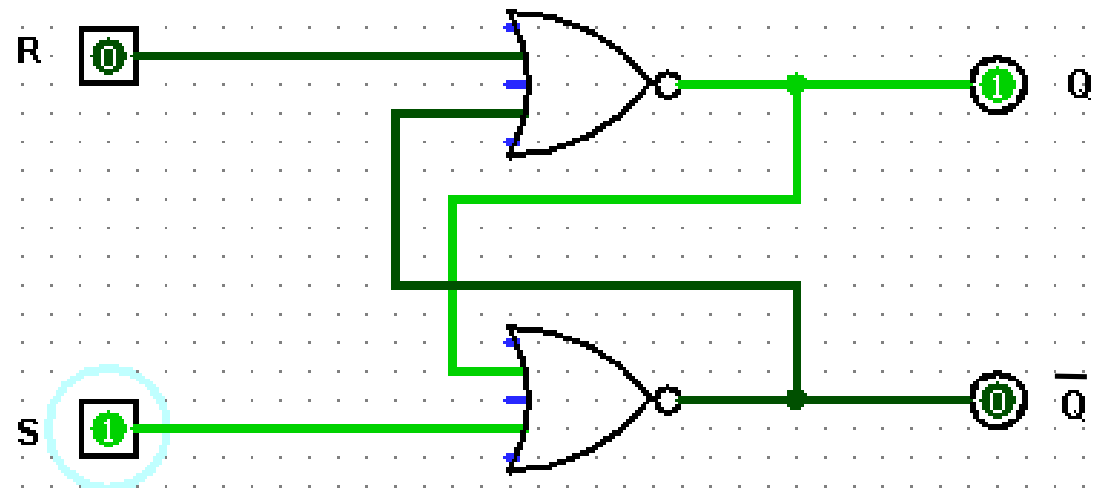
$$Q_{n+1} = \overline{S}.R + \overline{S}.Q_n$$

$$= \overline{S}.(R + Q_n)$$

$$= R + \overline{(S + Q_n)}$$

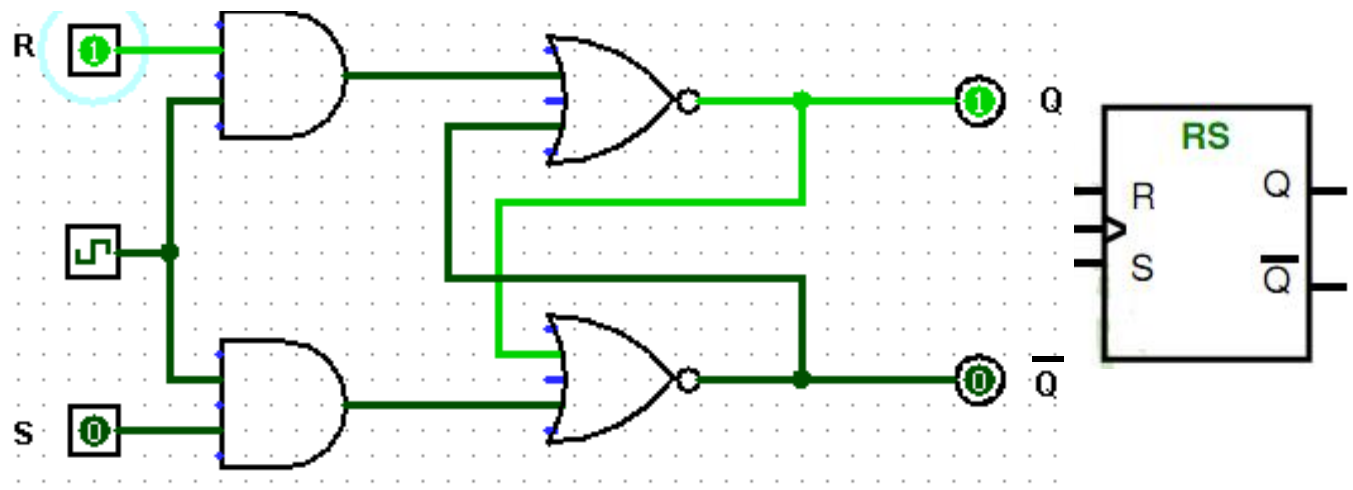
$$= S + \overline{(R + Q_n)}$$

Đầu vào hiện tại SR	Đầu ra hiện tại Q <sub>n</sub>	Đầu ra tiếp theo Q <sub>n+1</sub>
00	0	0
00	1	1
01	0	0
01	1	0
10	0	1
10	1	1
11	0	-
11	1	-

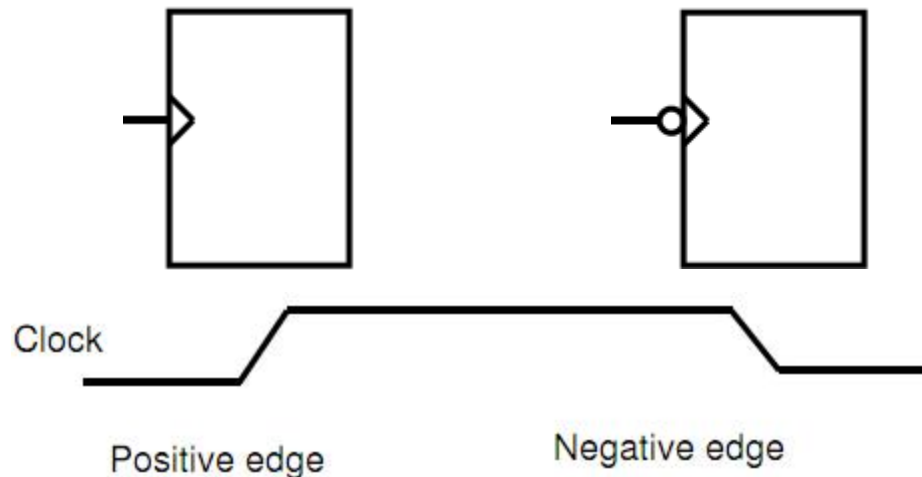


# Mạch lật RS có tín hiệu đồng bộ

Clk	R	S	Q	Q'
0	x	x	Q	Q'
1	0	0	Q	Q'
1	0	1	1	0
1	1	0	0	1
1	1	1	-	-

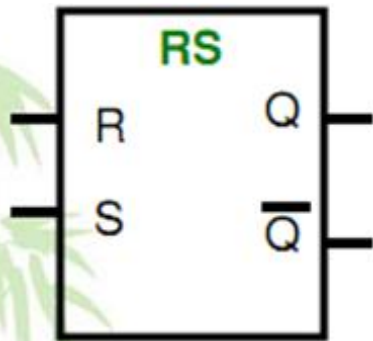


- Khi Clk = 1, mạch hoạt động như mạch lật RS
- Khi Clk = 0, 2 ngõ vào RS bị vô hiệu hóa, Q giữ nguyên giá trị

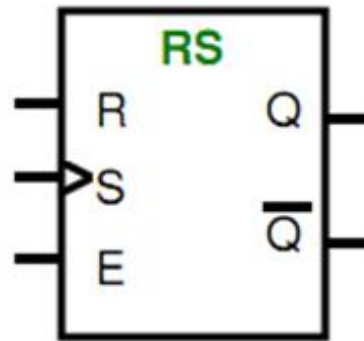




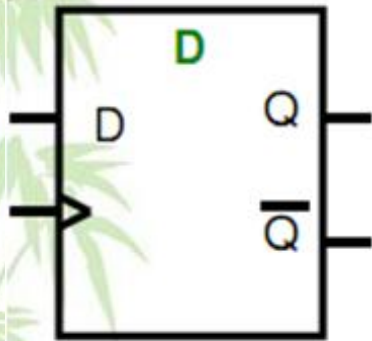
# Một số mạch lật



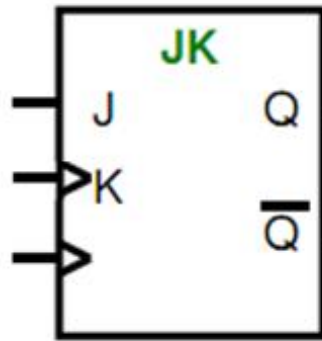
R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	-



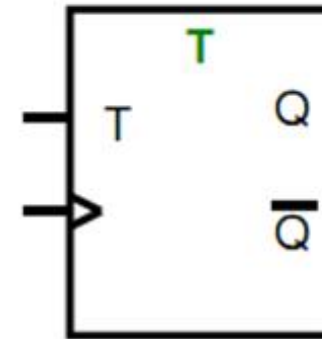
E	R	S	Q
0	x	x	Q
1	0	0	Q
1	0	1	1
1	1	0	0
1	1	1	-



D	Q
0	0
1	1



J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

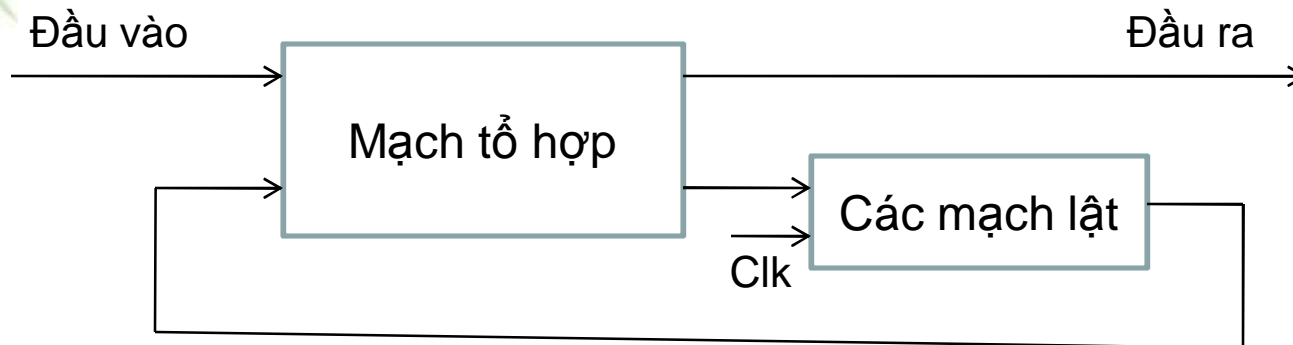


T	Q
0	Q
1	$\overline{Q}$



# Thiết kế mạch tuần tự

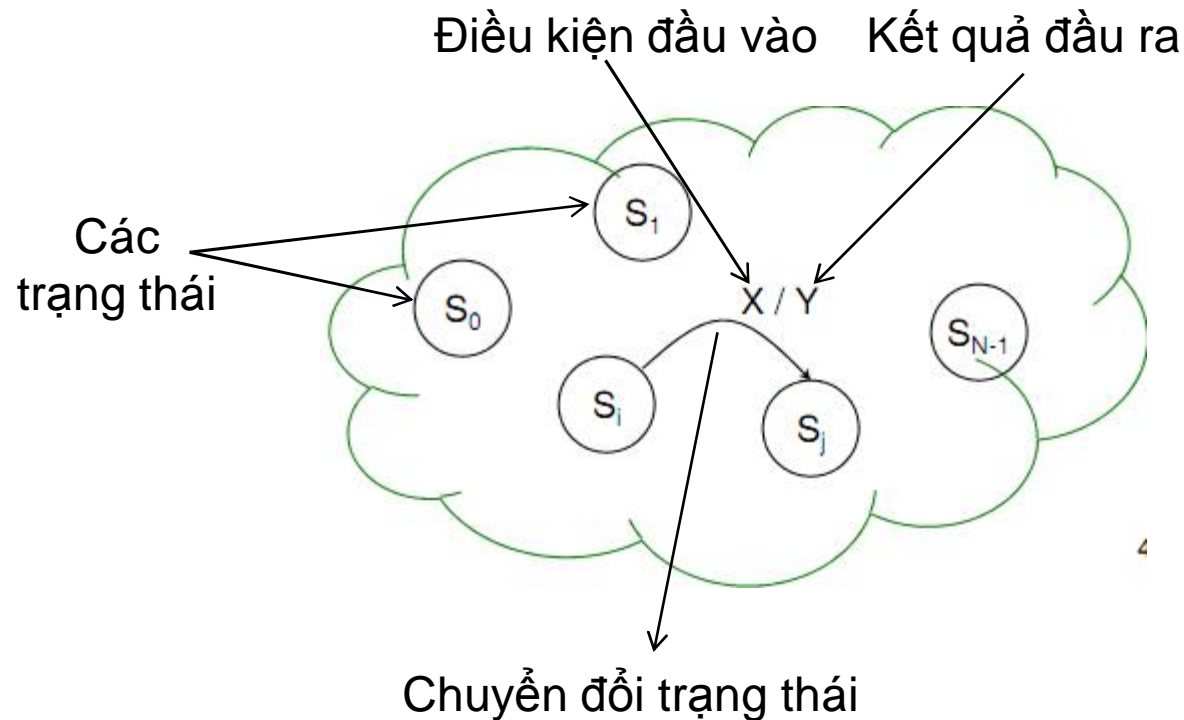
- Mạch tuần tự thường gồm mạch tổ hợp kết nối với các mạch lật. Còn được gọi là máy trạng thái hữu hạn (Finite State Machine - FSM). Thường có mô hình như sau



- 4 bước thiết kế mạch tuần tự
  - Vẽ lược đồ trạng thái từ yêu cầu
  - Lập bảng trạng thái từ sơ đồ trạng thái
  - Xây dựng hàm luận lý từ sơ đồ trạng thái
  - Vẽ sơ đồ mạch luận lý và thử nghiệm

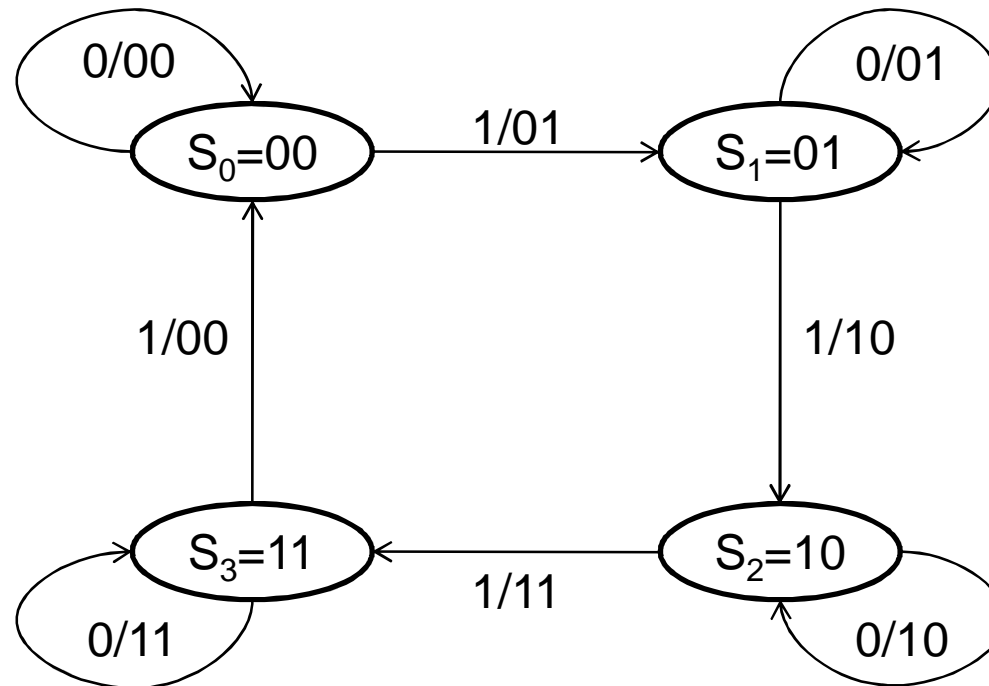
# Vẽ lược đồ trạng thái

- Từ yêu cầu bài toán, xác định các trạng thái cần thiết và các chuyển đổi trạng thái có thể xảy ra



# Ví dụ vẽ lược đồ trạng thái

- Yêu cầu: xây dựng mạch đếm nhị phân 2 bit với tín hiệu điều khiển  $i$ 
  - $i = 1$ : các trạng thái lần lượt biến đổi  
00 à 01 à 10 à 11 à 00
  - $i = 0$ : trạng thái không đổi
- Xây dựng lược đồ trạng thái



# Lập bảng trạng thái

- Để quản lý  $n$  trạng thái, cần  $\lceil \log_2 n \rceil$  mạch lật
- Lựa chọn loại mạch lật (D, JK,...)
- Ví dụ
  - Để quản lý 4 trạng thái (00, 01, 10, 11), cần 2 mạch lật
  - Chọn mạch lật JK

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Trạng thái hiện hành		Đầu vào i	Trạng thái kế tiếp ~ Đầu ra		Mạch lật 0		Mạch lật 1	
p1	p0		n1	n0	J0	K0	J1	K1
0	0	0	0	0	0	x	0	x
0	0	1	0	1	1	x	0	x
0	1	0	0	1	x	0	0	x
0	1	1	1	0	x	1	1	x
1	0	0	1	0	0	x	x	0
1	0	1	1	1	1	x	x	0
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

# Xây dựng hàm luận lý

- Có thể sử dụng phương pháp đại số Bool hoặc phương pháp bản đồ Karnaugh

Trạng thái hiện hành		Đầu vào i	Trạng thái kế ~ Đầu ra		Mạch lật 0		Mạch lật 1	
p1	p0		n1	n0	J0	K0	J1	K1
0	0	0	0	0	0	x	0	x
0	0	1	0	1	1	x	0	x
0	1	0	0	1	x	0	0	x
0	1	1	1	0	x	1	1	x
1	0	0	1	0	0	x	x	0
1	0	1	1	1	1	x	x	0
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

i \ p	0	0	1	1
	0	1	1	0
0		x	x	
1	1	x	x	1

$$J0 = i$$

i \ p	0	0	1	1
	0	1	1	0
0	x			x
1	x	1	1	x

$$K0 = i$$

i \ p	0	0	1	1
	0	1	1	0
0			x	x
1		1	x	x

$$J1 = p0.i$$

i \ p	0	0	1	1
	0	1	1	0
0	x	x		
1	x	x	1	

$$K1 = p0.i$$

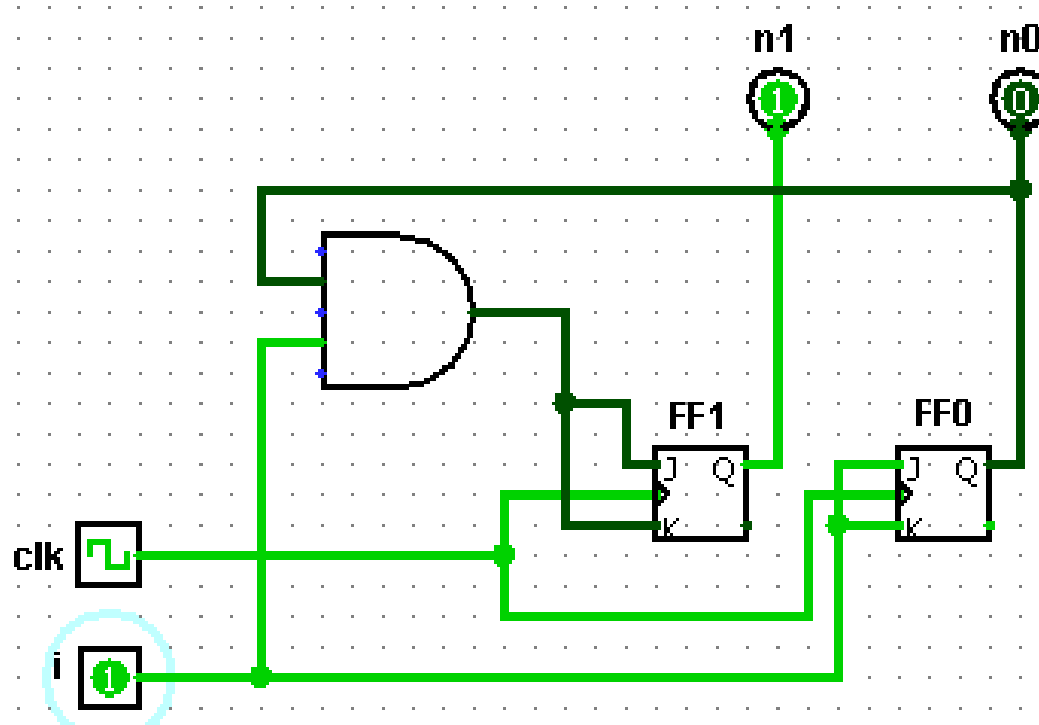
## Vẽ sơ đồ mạch luận lý và thử nghiệm

$$J0 = i$$

$$J1 = p0.i$$

$$K0 = i$$

$$K1 = p0.i$$





# Thanh ghi dịch

- Thanh ghi dịch (shift register) 4 bit được tạo từ 4 mạch lật D với dãy bit nạp 10010000

