

NHẬP MÔN MẠCH SỐ


Chương 5 – phần 2

**Mạch tổ hợp:
Các loại mạch khác**

Nội dung

1. Mạch cộng (Carry Ripple (CR) Adder)
2. Mạch cộng nhìn trước số nhớ - (Carry Look-Ahead (CLA) Adder)
3. Mạch cộng/ mạch trừ
4. Đơn vị tính toán luận lý (Arithmetic Logic Unit)
5. Mạch giải mã (Decoder)/ Mạch mã hoá (Encoder)
6. Mạch dồn kênh (Multiplexer)/ Mạch chia kênh (Demultiplexer)
7. Mạch tạo Parity/ Mạch kiểm tra Parity
8. Mạch so sánh (Comparator)

Nội dung

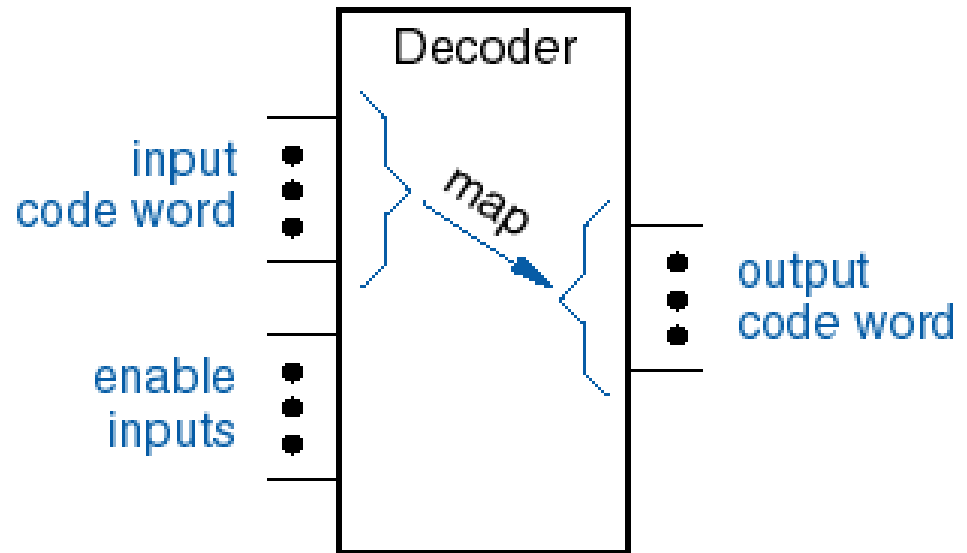
- 
1. Mạch cộng (Carry Ripple (CR) Adder)
 2. Mạch cộng nhìn trước số nhớ - (Carry Look-Ahead (CLA) Adder)
 3. Mạch cộng/ mạch trừ
 4. Đơn vị tính toán luận lý (Arithmetic Logic Unit)
 5. Mạch giải mã (Decoder)/ Mạch mã hoá (Encoder)
 6. Mạch dồn kênh (Multiplexer)/ Mạch chia kênh (Demultiplexer)
 7. Mạch tạo Parity/ Mạch kiểm tra Parity
 8. Mạch so sánh (Comparator)



5. Decoder/ Encoder

Mạch giải mã (Decoder)

- Nhiều ngõ vào/ nhiều ngõ ra
- Ngõ vào (n) thông thường ít hơn ngõ ra (m)
- Chuyển mã ngõ vào thành mã ngõ ra
- **Ảnh xạ 1-1:**
 - Mỗi mã ngõ vào chỉ tạo ra một mã ngõ ra
- Các mã ngõ vào:
 - Mã nhị phân
 - Your Code!
- Các mã ngõ ra:
 - 1-trong-m
 - Gray Code
 - BCD Code



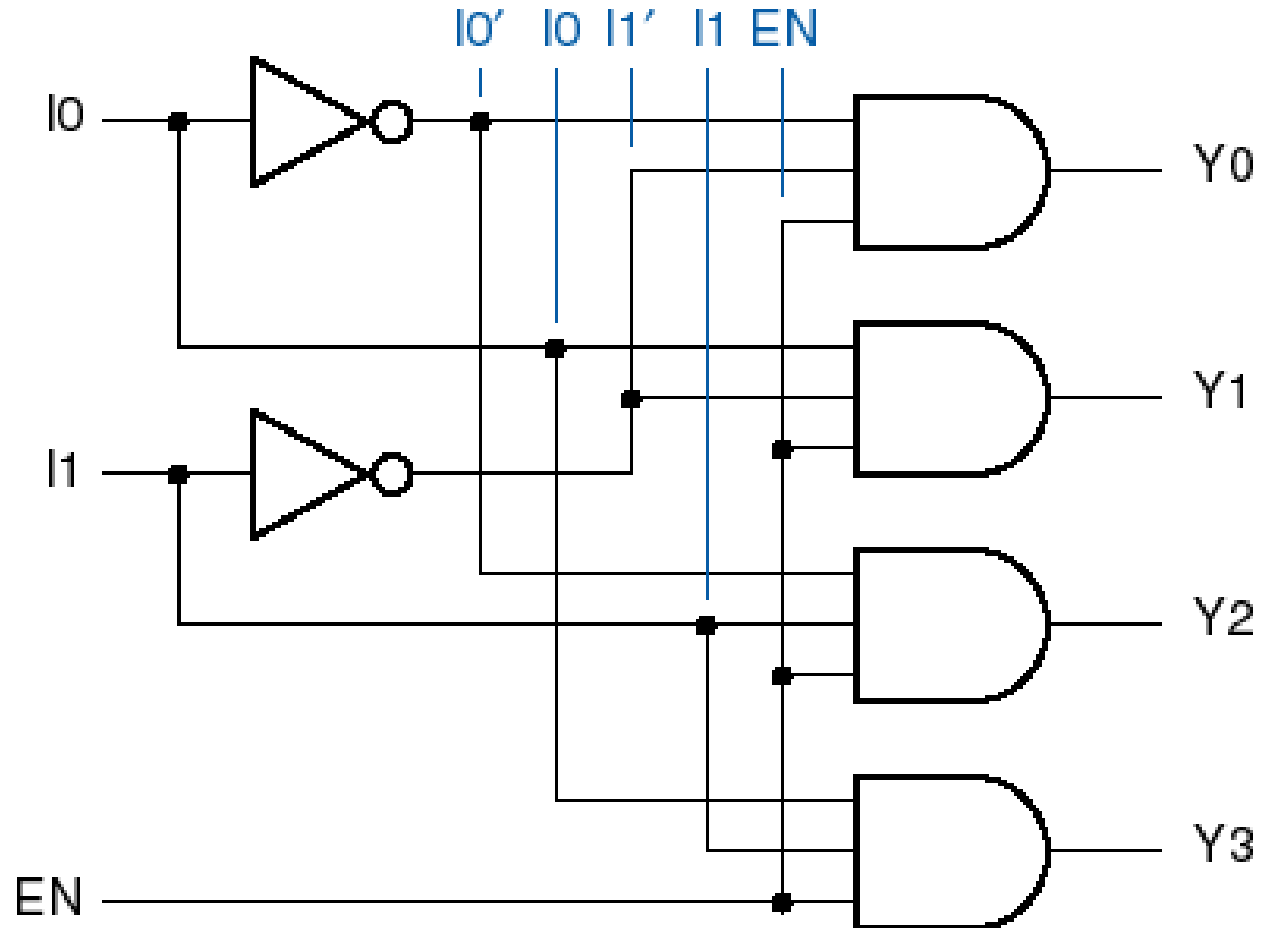
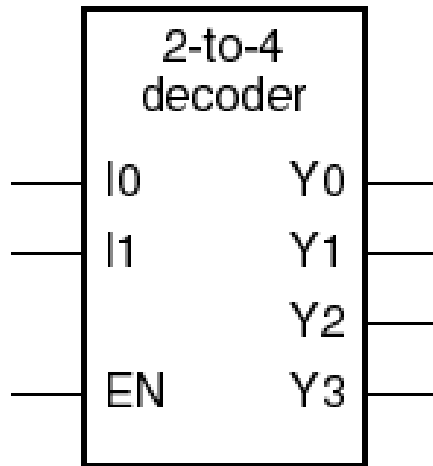
Mạch giải mã nhị phân (Binary Decoders)

- Mạch giải mã **n-ra- 2^n** : n ngõ vào và 2^n ngõ ra
 - Mã đầu vào: n bit nhị phân
 - Mã đầu ra: 1-trong- 2^n
- Ví dụ: n=2, mạch giải mã 2-ra-4

<i>Inputs</i>			<i>Outputs</i>			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

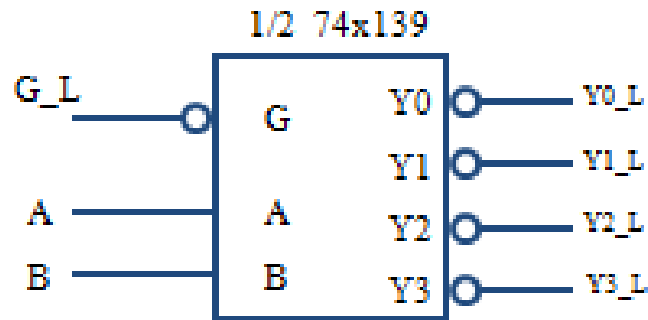
Chú ý “x” (kí hiệu ngõ vào don’t care)

Giải mã nhị phân 2-ra-4



Chip 74x139: giải mã nhị phân 2-to-4

- Tính hiệu Enable tích cực mức thấp và ngõ ra tích cực mức thấp

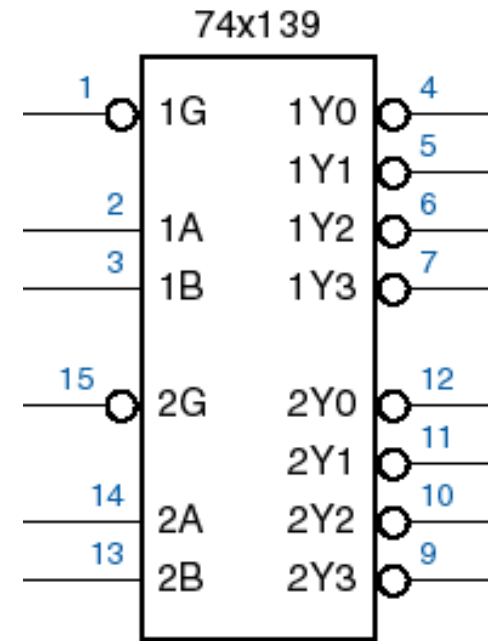
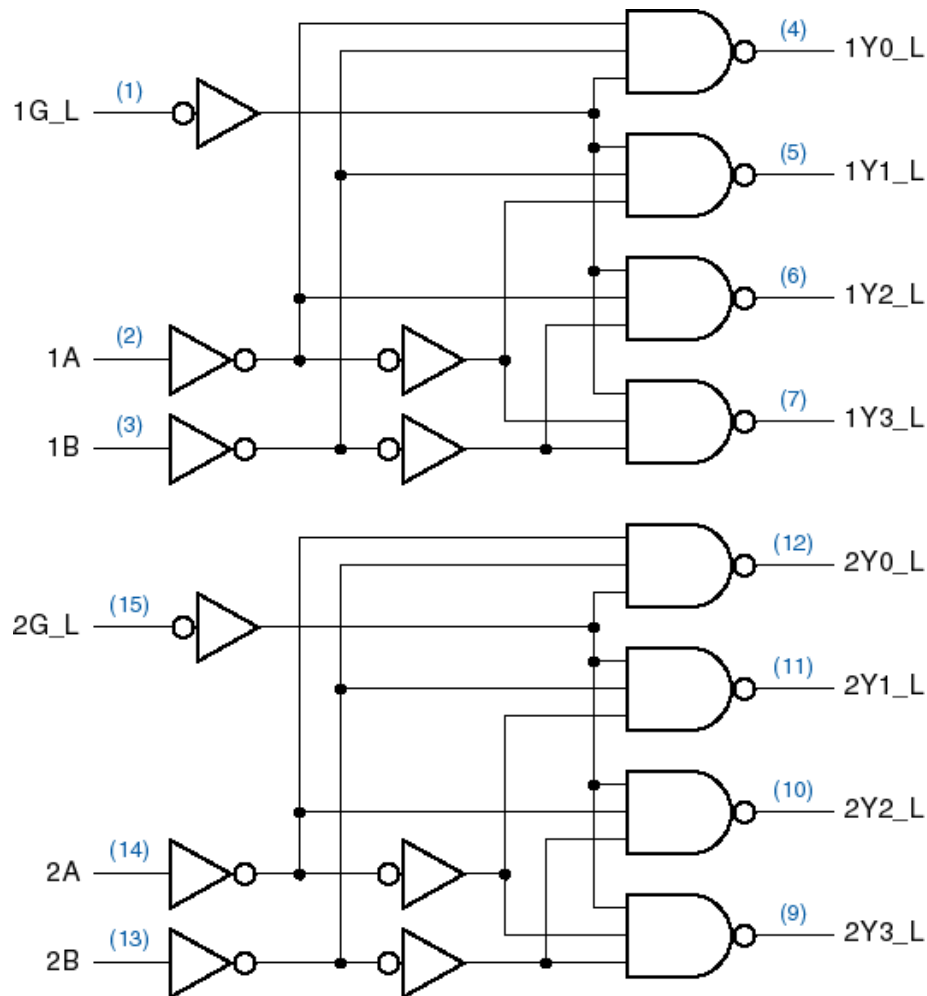


Ký hiệu
luận lý

Inputs			Outputs			
G_L	B	A	Y3-L	Y2-L	Y1-L	Y0_L
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Bảng sự thật

Mạch giải mã hoàn chỉnh 74x139



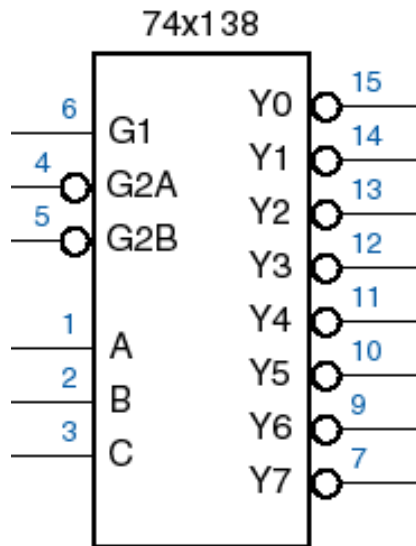
Chip 74x138: Giải mã nhị phân 3-to-8



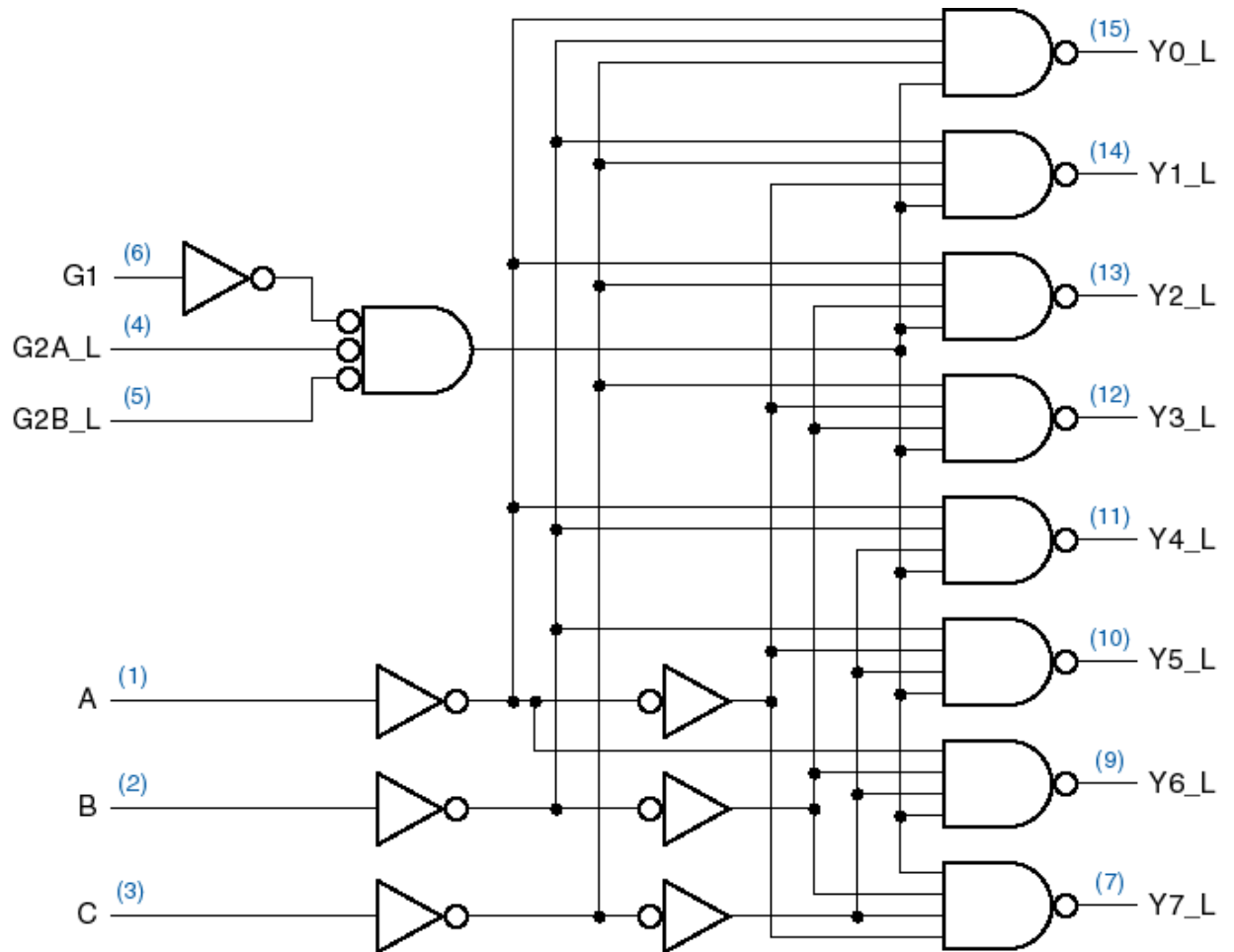
<i>Inputs</i>						<i>Outputs</i>							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

Bảng sự thật

74x138



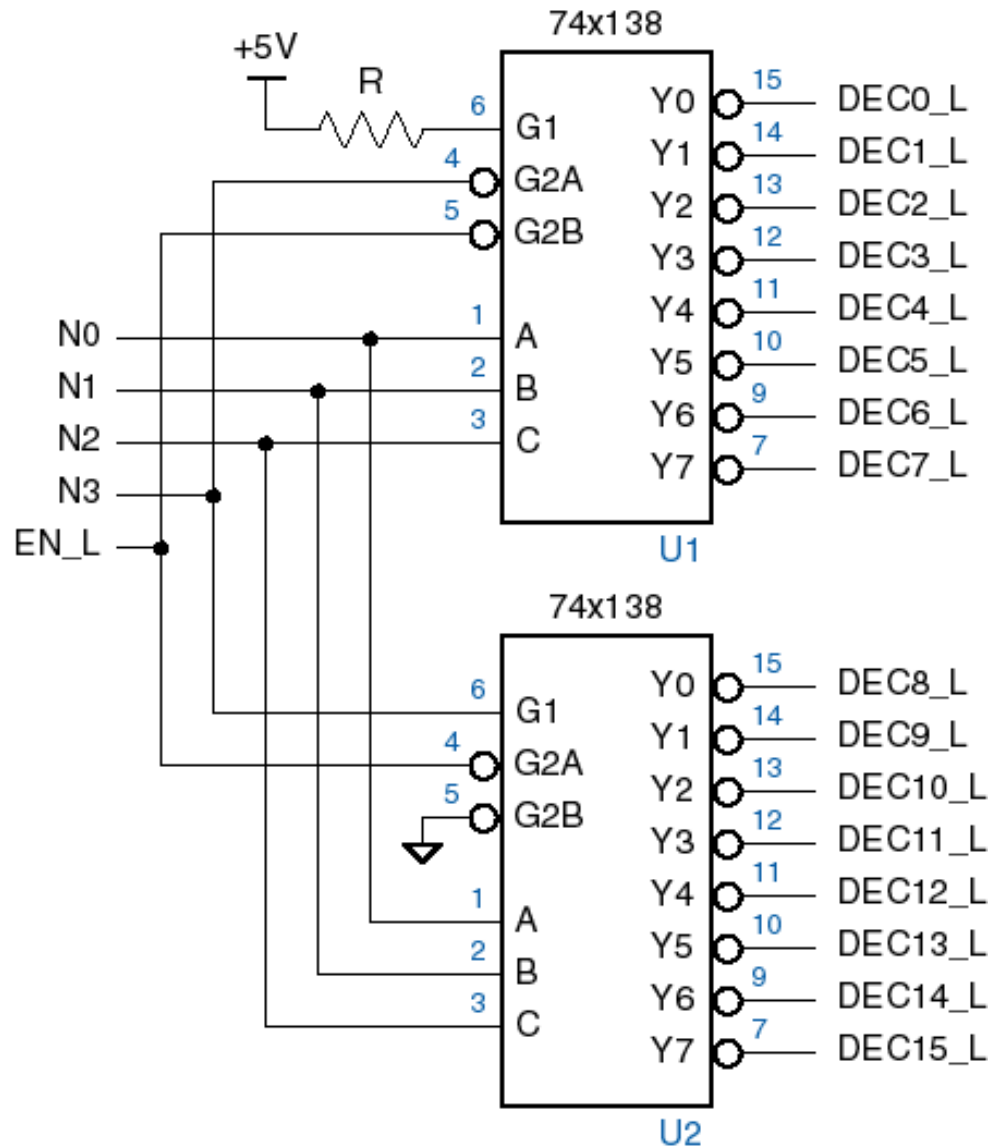
Ký hiệu
luận lý



Mạch luận lý

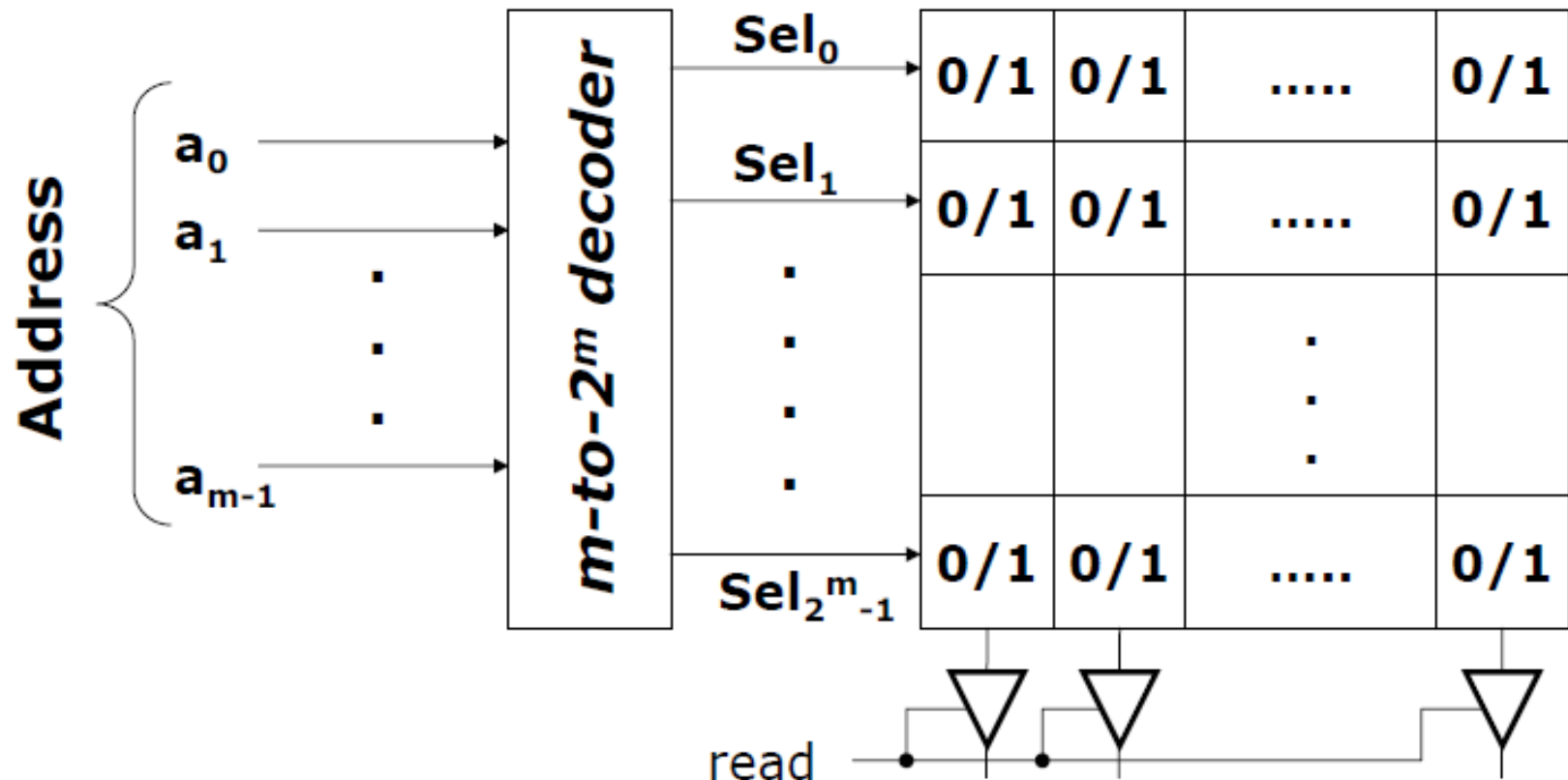
Ghép mạch giải mã

Mạch giải mã
4-to-16

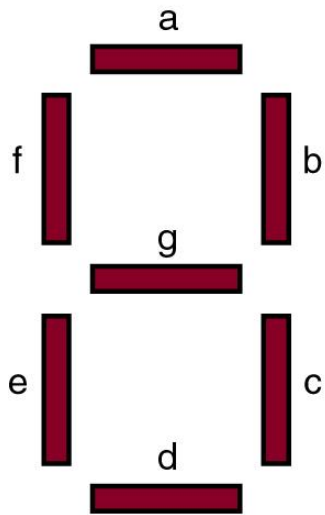


Ứng dụng của mạch giải mã

- Một ứng dụng phổ biến là giải mã địa chỉ cho các chip nhớ

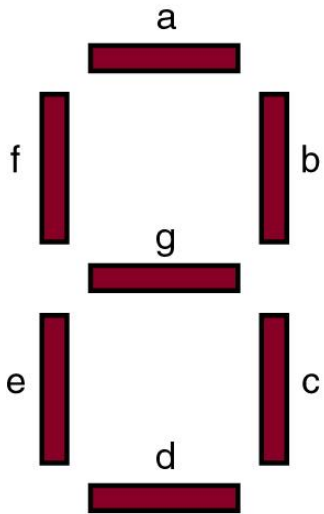


Giải mã BCD ra LED 7 đoạn



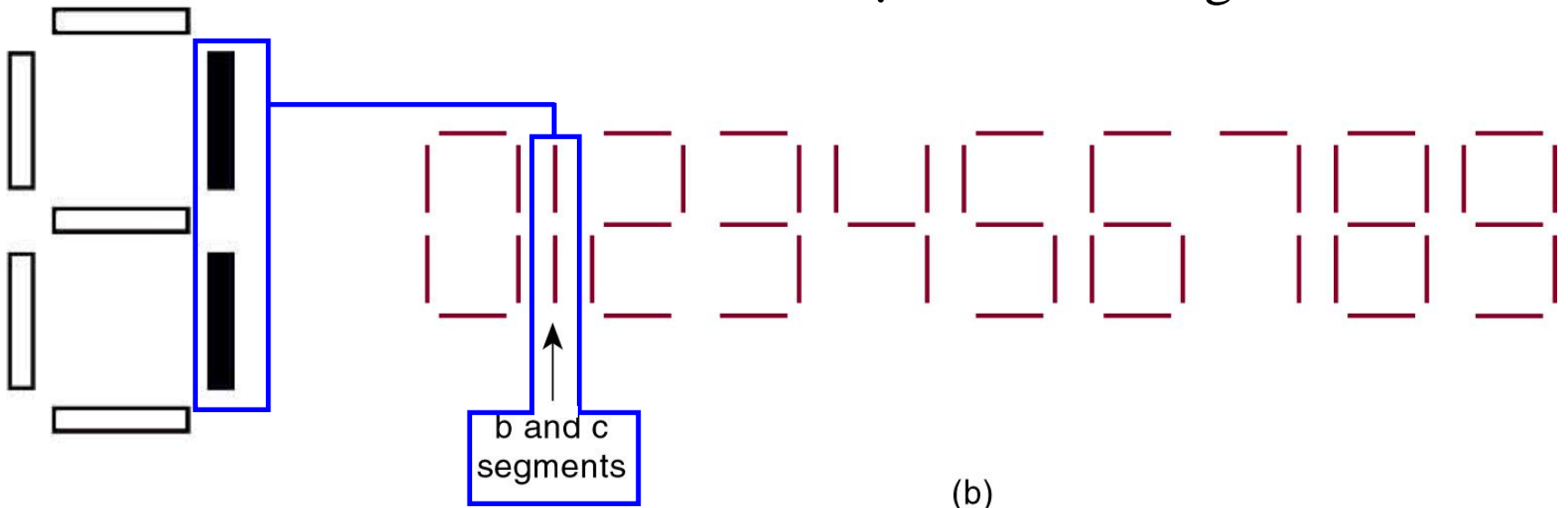
LED 7 đoạn (7-segment display)

- LED 7 đoạn là cách phổ biến để hiển thị số thập phân hoặc số thập lục phân
 - Sử dụng LED cho mỗi đoạn



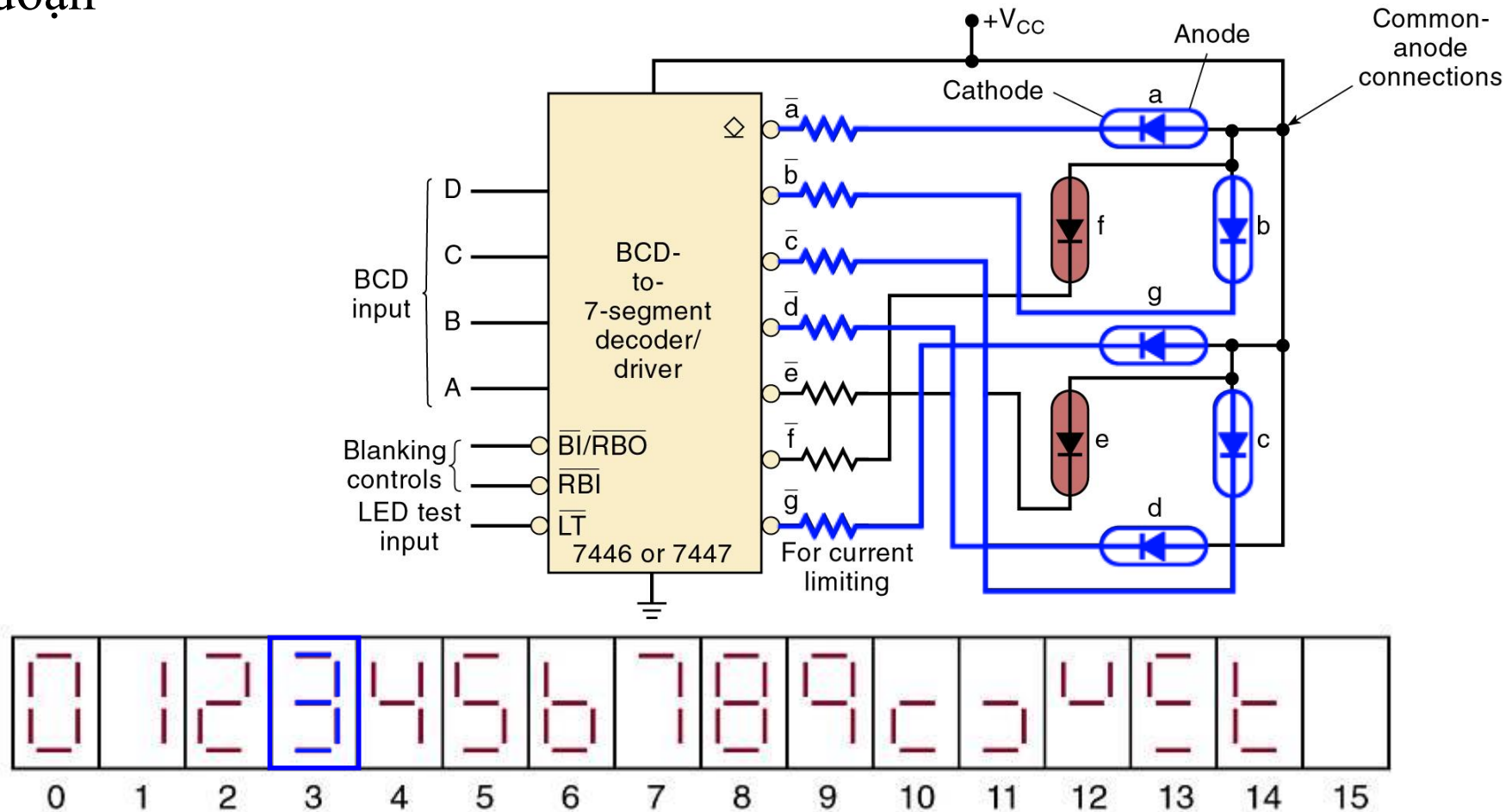
LED 7 đoạn (7-segment display)

Bằng cách điều khiển dòng điện qua mỗi LED, một số đoạn sẽ sáng và một số tắt, từ đó tạo nên số mong muốn



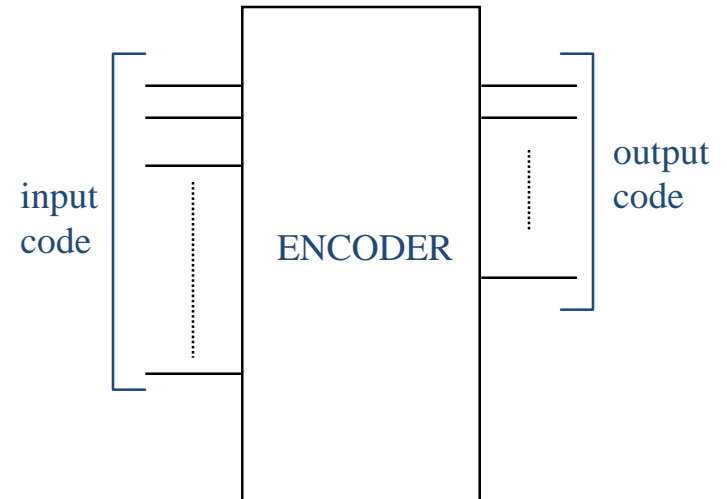
Giải mã BCD ra LED 7 đoạn

Chuyển số BCD sang thông tin thích hợp để hiển thị trên đèn 7 đoạn

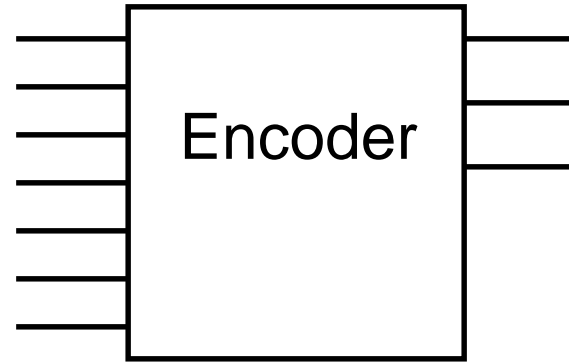
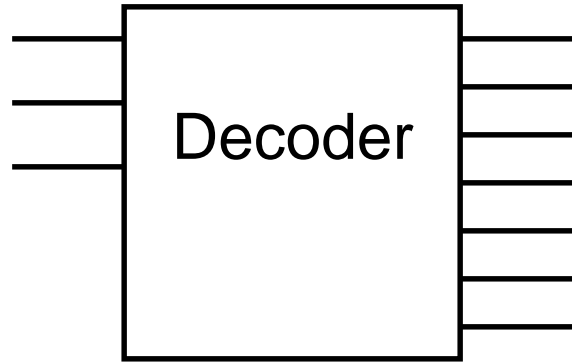


Mạch mã hoá (Encoder)

- Nhiều ngõ vào/ nhiều ngõ ra
- Chức năng ngược lại với mạch giải mã
- Outputs (m) ít hơn inputs (n)
- Chuyển mã ngõ vào thành mã ngõ ra



Encoders vs. Decoders



decoders/encoders nhị phân

- $n \rightarrow 2^n$
- Input code: Mã nhị phân
- Output code: 1-trong- 2^n

- $2^n \rightarrow n$
- Input code: 1-trong- 2^n
- Output code: Mã nhị phân

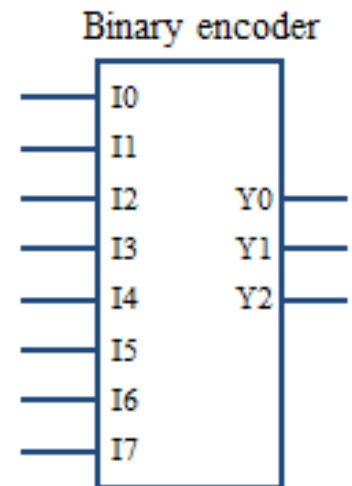
Mạch mã hoá nhị phân (Binary Encoder)

- **2^n -ra-n encoder:** 2^n ngõ vào và n ngõ ra
 - Input code: 1-trong- 2^n
 - Output code: Mã nhị phân

Ví dụ: $n=3$, mạch mã hóa 8-ra-3

<u>Inputs</u>								<u>Outputs</u>		
I0	I1	I2	I3	I4	I5	I6	I7	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Bảng sự thật



Ký hiệu

Hiện thực mạch mã hóa 8-ra-3

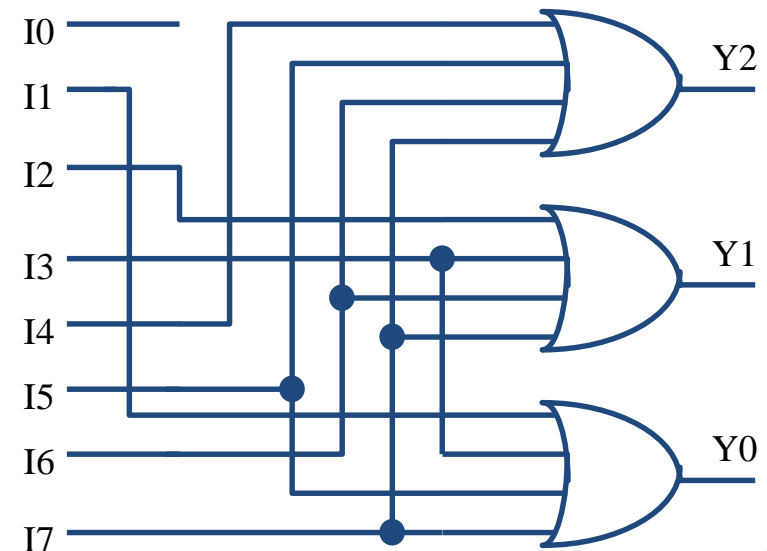
Ngõ vào								Ngõ ra		
I0	I1	I2	I3	I4	I5	I6	I7	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- Rút gọn:

$$Y0 = I1 + I3 + I5 + I7$$

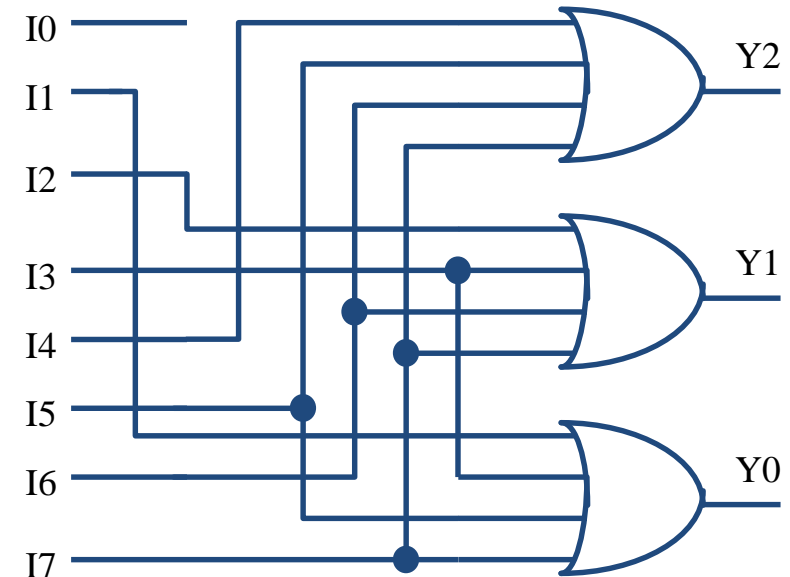
$$Y1 = I2 + I3 + I6 + I7$$

$$Y2 = I4 + I5 + I6 + I7$$

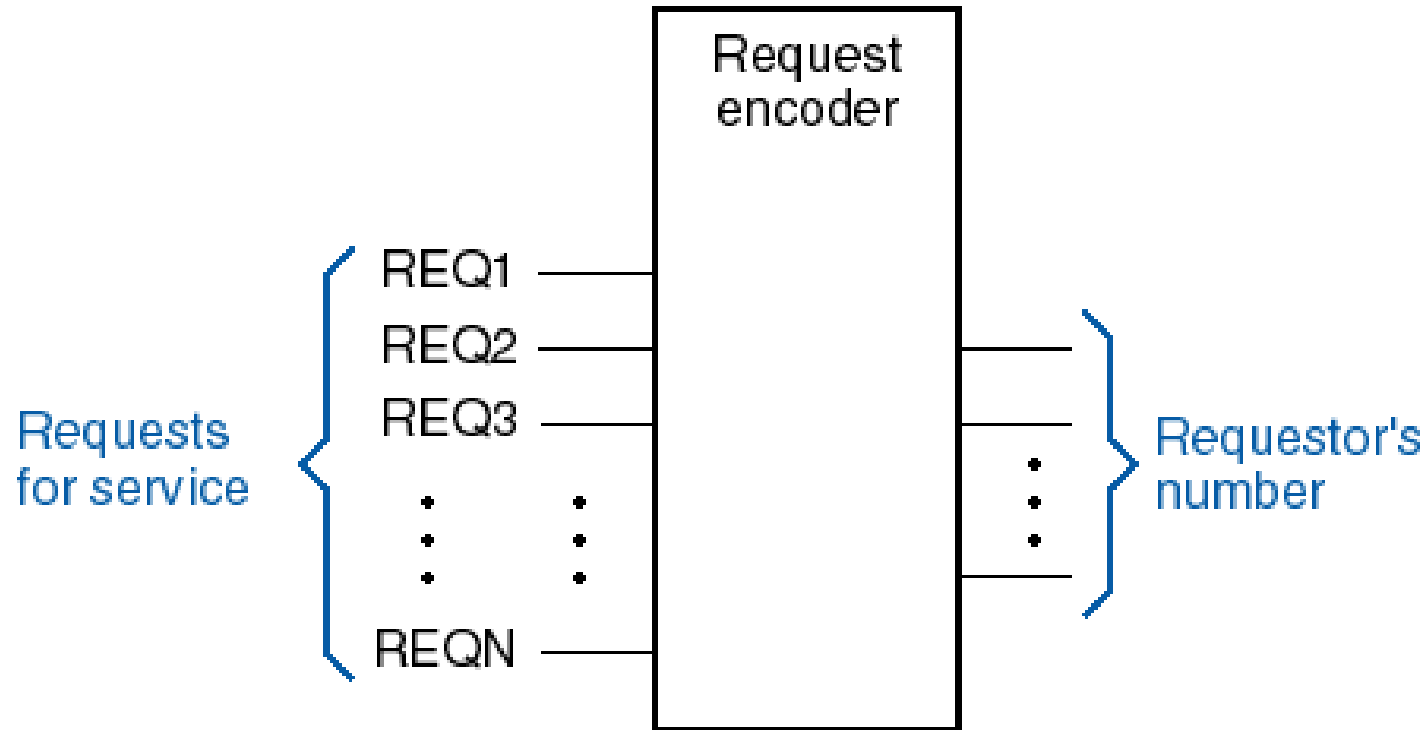


Hiện thực mạch mã hóa 8-ra-3

- Giới hạn:
 - **I0** không ảnh hưởng đến ngõ ra
 - Chỉ một ngõ nhập được kích hoạt tại một thời điểm
- Ứng dụng:
 - Giải quyết những yêu cầu từ *nhiều thiết bị*, nhưng không phải là những yêu cầu đồng thời.
 - Trong trường hợp có nhiều thiết bị yêu cầu cùng lúc thì có thể thiết lập mức độ ưu tiên để giải quyết vấn đề này.



Cần có độ ưu tiên trong hầu hết các ứng dụng



Mạch mã hoá có độ ưu tiên (Priority Encoder)

- Gán độ ưu tiên cho các ngõ vào
- Khi có nhiều hơn 1 ngõ vào tích cực, ngõ ra tạo ra mã của ngõ vào có độ ưu tiên cao nhất.

- Priority Encoder:

$H7=I7$ (Độ ưu tiên cao nhất)

$H6=I6.I7'$

$H5=I5.I6'.I7'$

$H4=I4.I5'.I6'.I7'$

$H3=I3.I4'.I5'.I6'.I7'$

$H2=I2.I3'.I4'.I5'.I6'.I7'$

$H1=I1.I2'.I3'.I4'.I5'.I6'.I7'$

$H0=I0.I1'.I2'.I3'.I4'.I5'.I6'.I7'$

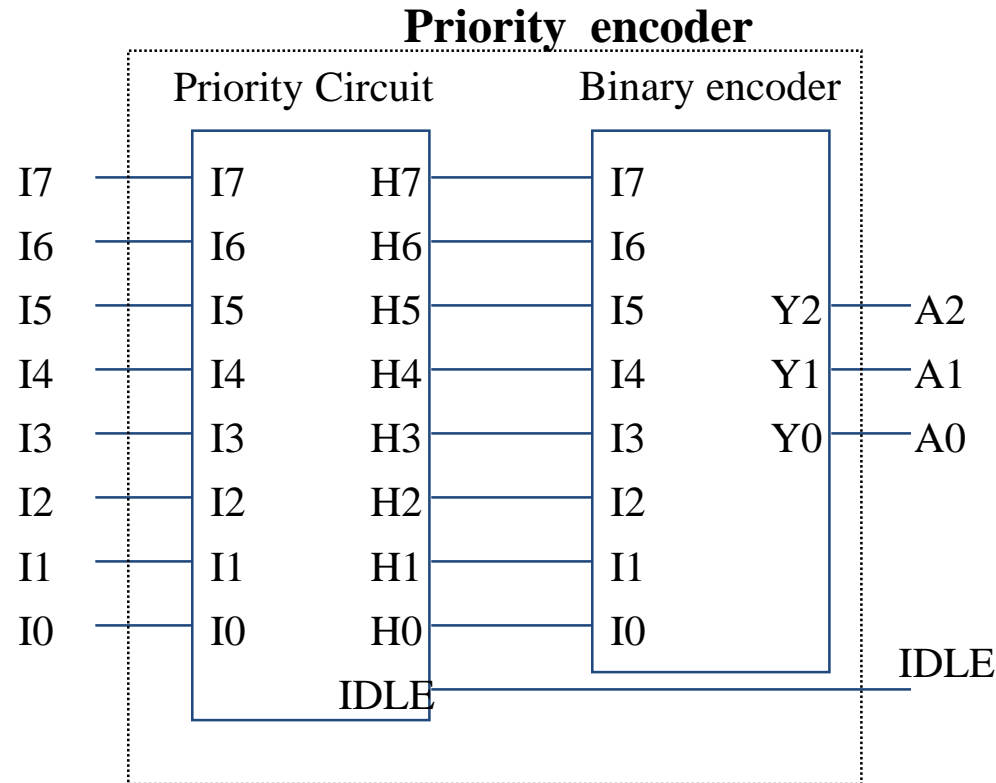
$IDLE= I0'.I1'.I2'.I3'.I4'.I5'.I6'.I7'$

- Encoder

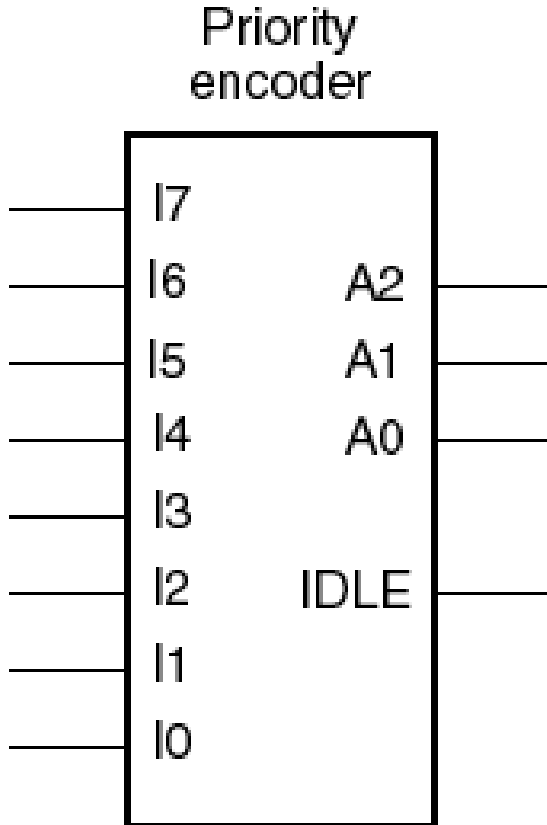
$A0=Y0 = H1 + H3 + H5 + H7$

$A1=Y1 = H2 + H3 + H6 + H7$

$A2=Y2 = H4 + H5 + H6 + H7$

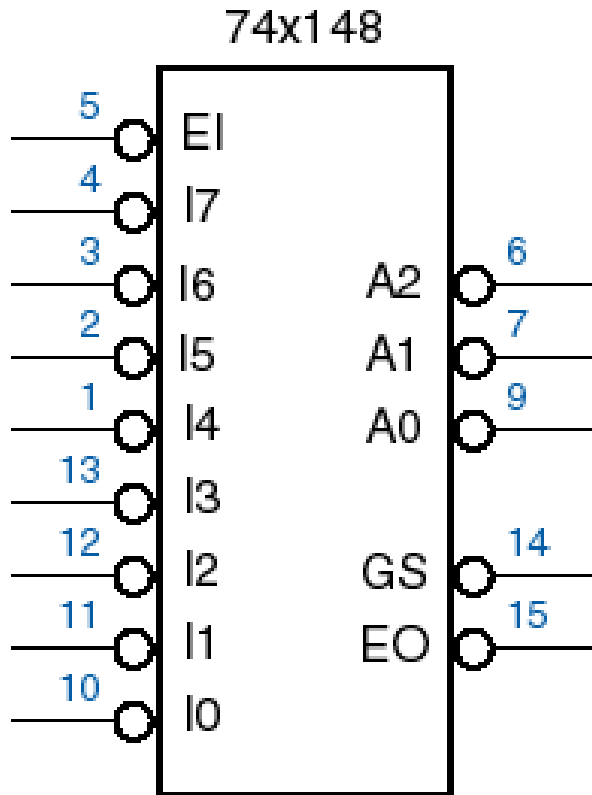


8-input priority encoder



- I7 có độ ưu tiên cao nhất, I0 thấp nhất
- A2-A0 chứa số thứ tự của ngõ vào có độ ưu tiên cao nhất đang tích cực
- IDLE tích cực nếu không có ngõ vào nào tích cực

74x148 8-input priority encoder



– Các pin trên chip đều tích cực mức thấp

GS: Got Something/Group Select
EO: Enable Output

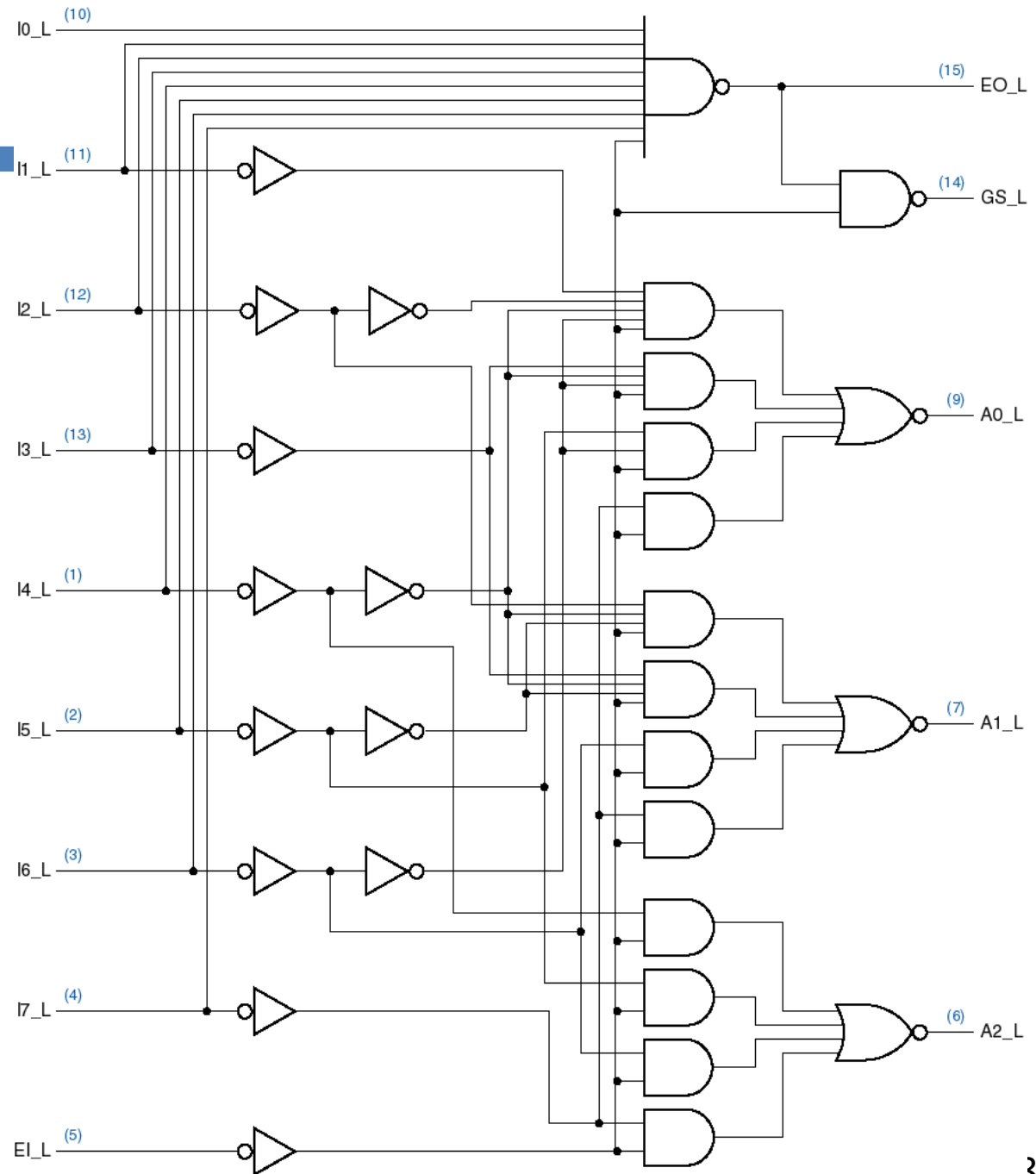
74x148 – Bảng sự thật

<i>Inputs</i>									<i>Outputs</i>				
E_L	I0_L	I1_L	I2_L	I3_L	I4_L	I5_L	I6_L	I7_L	A2_L	A1_L	A0_L	GS_L	EO_L
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

74x148

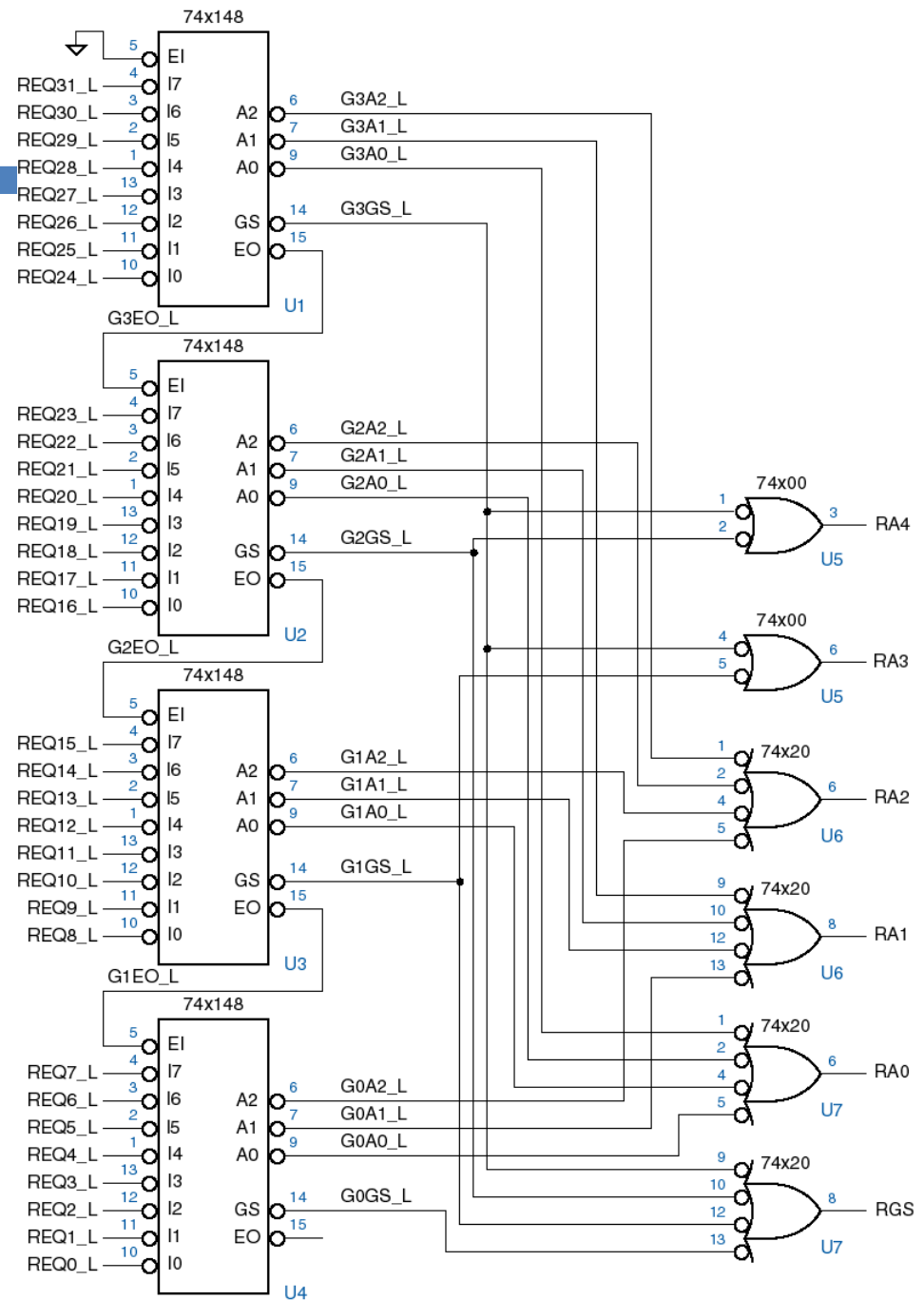


Sơ đồ luận lý



Ghép các priority encoders

Mạch mã hoá có độ ưu tiên
32-ngõ vào được ghép từ 4
mạch mã hóa ưu tiên 8-ngõ
vào

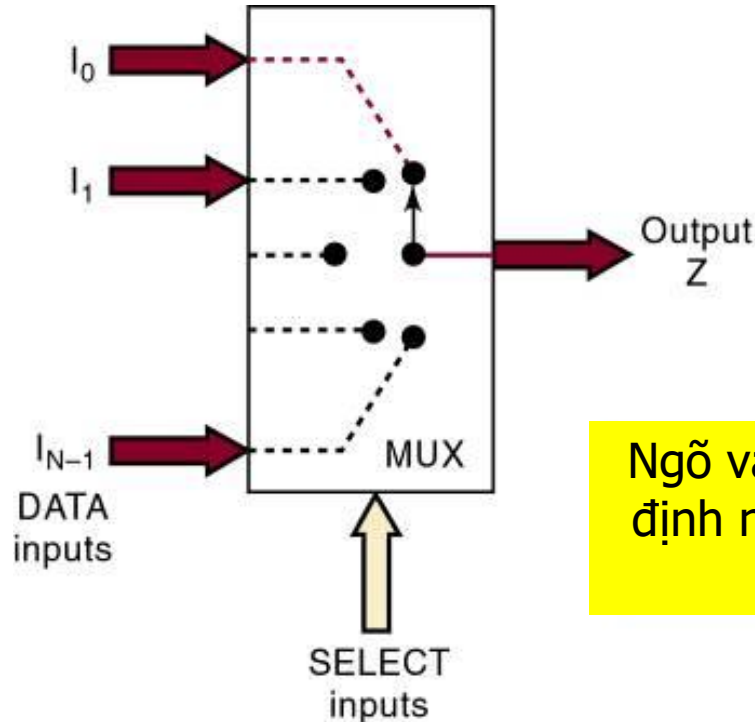




6. Multiplexer (MUX)/ Demultiplexer (DeMUX)

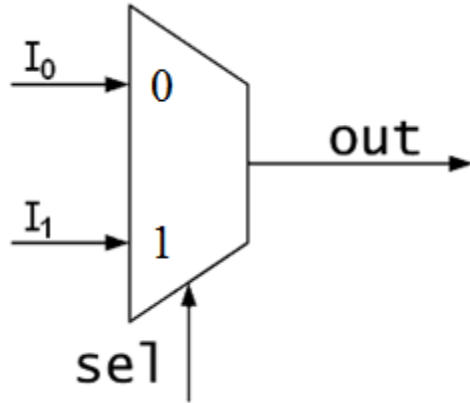
Multiplexer (MUX)

- Một **MUX** truyền một trong những ngõ vào của nó ra ngõ ra dựa trên tín hiệu Select



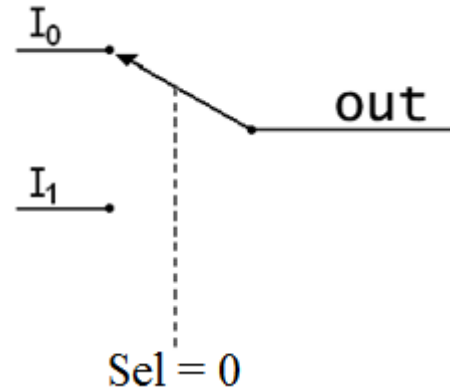
Ngõ vào SELECT sẽ xác định ngõ vào nào được truyền ra Z

2-to-1 Multiplexer



Ký hiệu

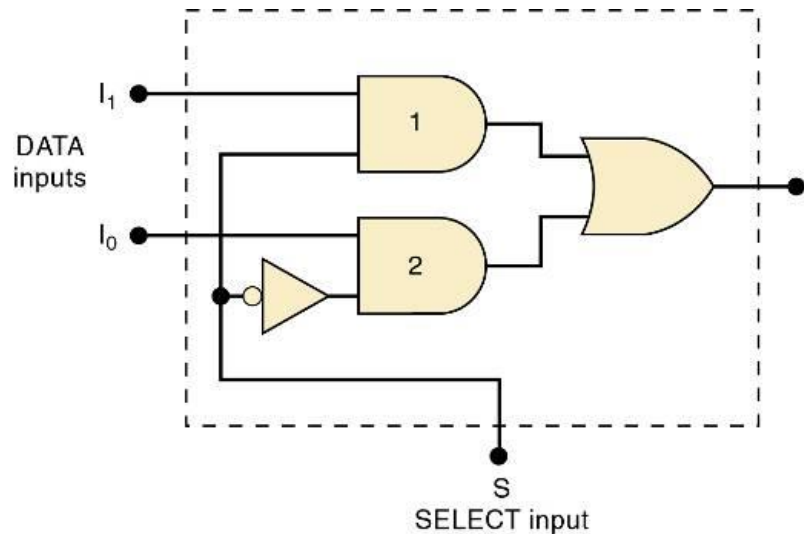
Minh họa
với Sel = 0



Sel	Out
0	I_0
1	I_1

$$\text{Out} = \overline{\text{Sel}} * I_0 + \text{Sel} * I_1$$

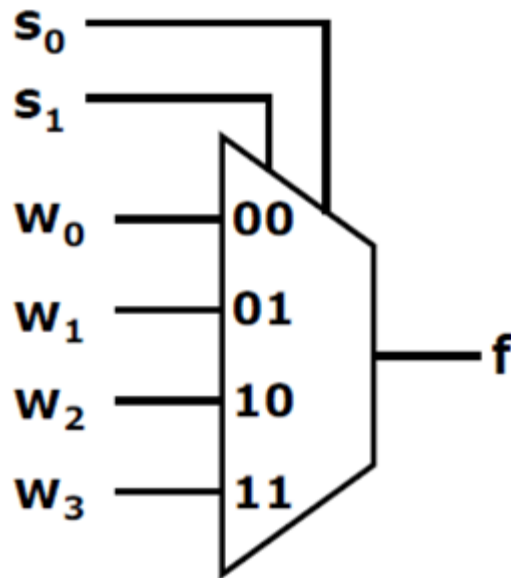
Biểu thức đại số



Mạch logic

4-to-1 Mux

- 4-to-1 Mux xuất ra một trong bốn ngõ vào dựa trên giá trị của 2 tín hiệu select



Ký hiệu

s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

Bảng sự thật

$$f = s_1' s_0' w_0 + s_1' s_0 w_1 + s_1 s_0' w_2 + s_1 s_0 w_3$$

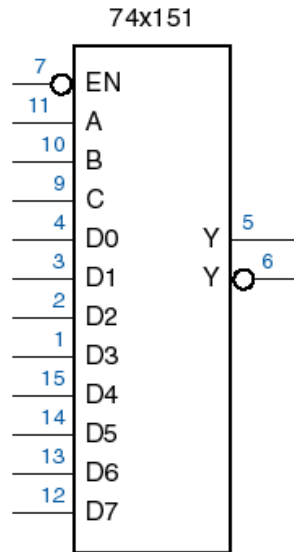
Biểu thức đại số

Xây dựng MUX 4-to-1

- Từ MUX 2-to-1

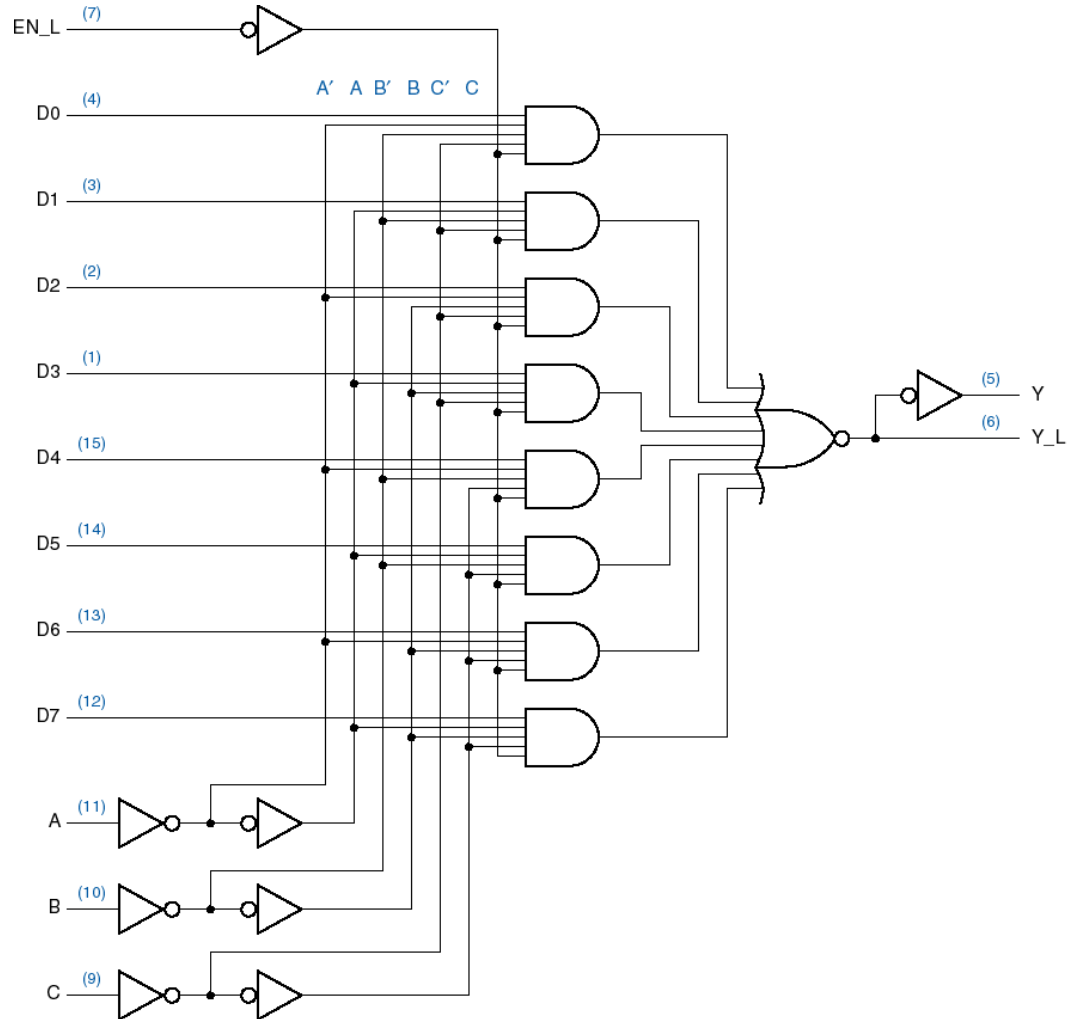
Chip 74x151: 1-bit multiplexer 8-to-1

Ký hiệu



Bảng sự thật

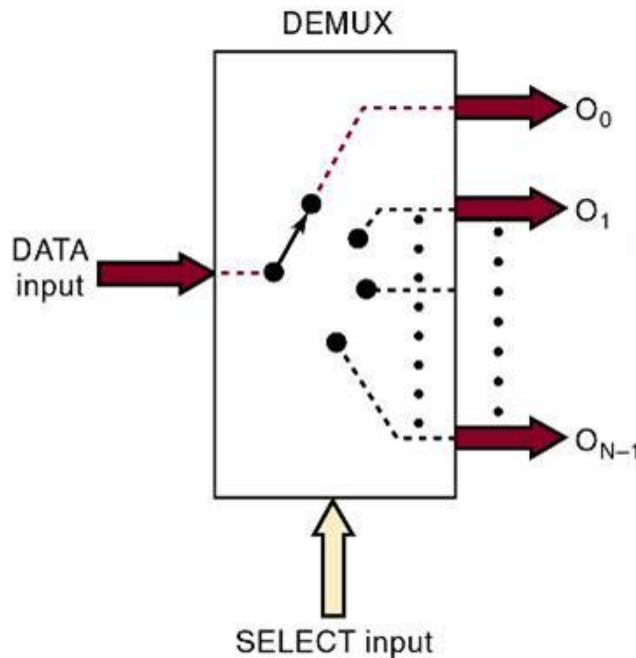
Inputs				Outputs	
EN_L	S2	S1	S0	Y	Y_L
1	x	x	x	0	1
0	0	0	0	D0	D0'
0	0	0	1	D1	D1'
0	0	1	0	D2	D2'
0	0	1	1	D3	D3'
0	1	0	0	D4	D4'
0	1	0	1	D5	D5'
0	1	1	0	D6	D6'
0	1	1	1	D7	D7'



Sơ đồ luận lý

Demultiplexer

- **Demultiplexer (DEMUX)** lấy ngõ vào duy nhất và phân phối nó ra một ngõ ra.
 - Mã ngõ vào SELECT sẽ xác định ngõ ra nào sẽ được kết nối với ngõ vào



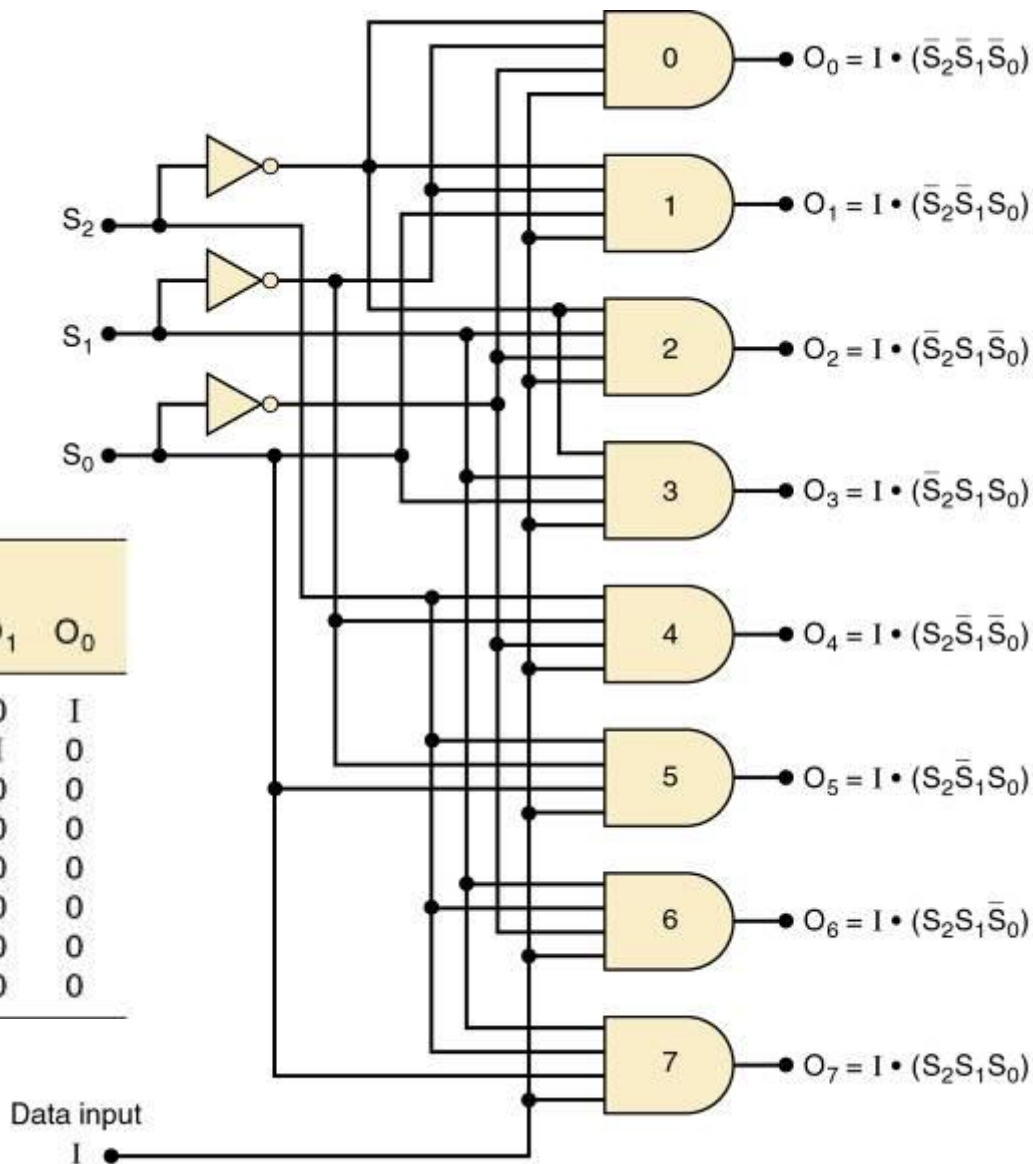
DATA được truyền ra *một* và *chỉ một* ngõ ra được xác định bởi mã của ngõ vào SELECT

DEMUX

1-to-8 demultiplexer

Select Code			Outputs							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Chú ý: I là ngõ vào DATA

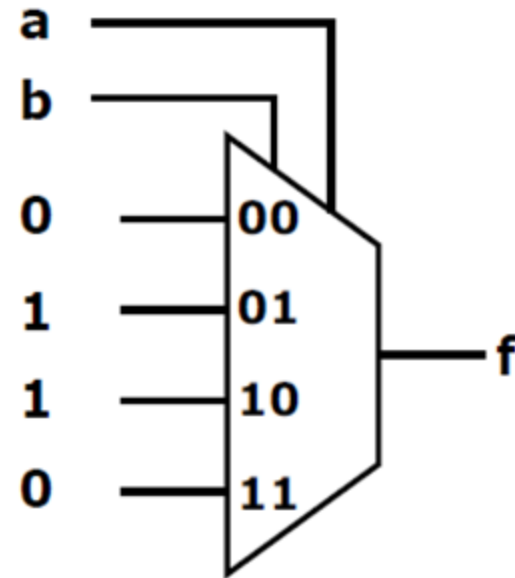


Tổng hợp các hàm logic từ MUX

- Cách hiện thực LUT (Look-up table)
 - Sử dụng MUX để chọn một giá trị (**hằng số**) từ 1 LUT



Ví dụ hàm XOR

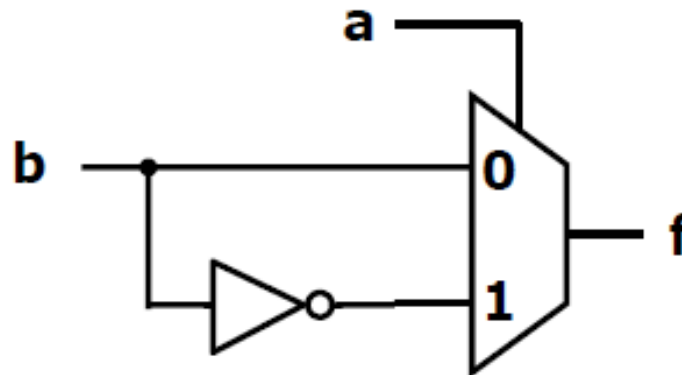
<i>a</i>	<i>b</i>	<i>f</i>
0	0	0
0	1	1
1	0	1
1	1	0



Tổng hợp các hàm logic từ MUX

- Giải pháp ở slide trước không hiệu quả vì phải sử dụng MUX 4-to-1
- Nhận xét:

a	b	f			a	f	
0	0	0	}	when $a=0, f=b$		0	b
0	1	1				0	b
1	0	1	}	when $a=1, f=b'$		1	b'
1	1	0				1	b'



Tổng hợp các hàm logic từ MUX

- Ví dụ: Hiện thực mạch với bảng sự thật sau bằng một **MUX** và các cổng khác

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

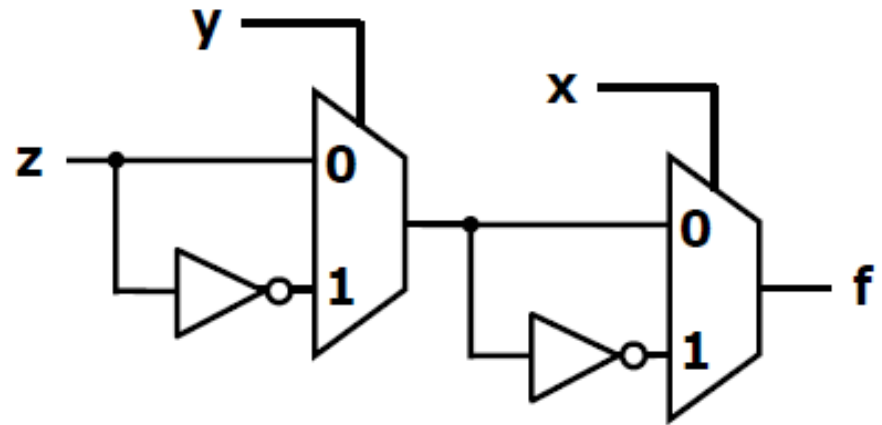
Tổng hợp các hàm logic từ MUX

- XOR 3 ngõ vào có thể hiện thực bằng 2 MUX 2-to-1

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

$y \oplus z$

$(y \oplus z)'$



Tổng hợp các hàm logic từ MUX

- **Ví dụ:** Hiện thực mạch với bảng sự thật sau bằng một MUX và các cổng logic khác

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

Biểu thức Shannon

- Bất kì hàm Boolean $f(w_1, w_2, \dots, w_n)$ có thể được viết dưới dạng:

$$f(w_1, w_2, \dots, w_n) = \overline{w_1} * f(0, w_2, \dots, w_n) + w_1 * f(1, w_2, \dots, w_n)$$

Biểu thức Shannon

- Ví dụ 1:

$$f(w_1, w_2, w_3) = w_1 w_2 + w_1 w_3 + w_2 w_3$$

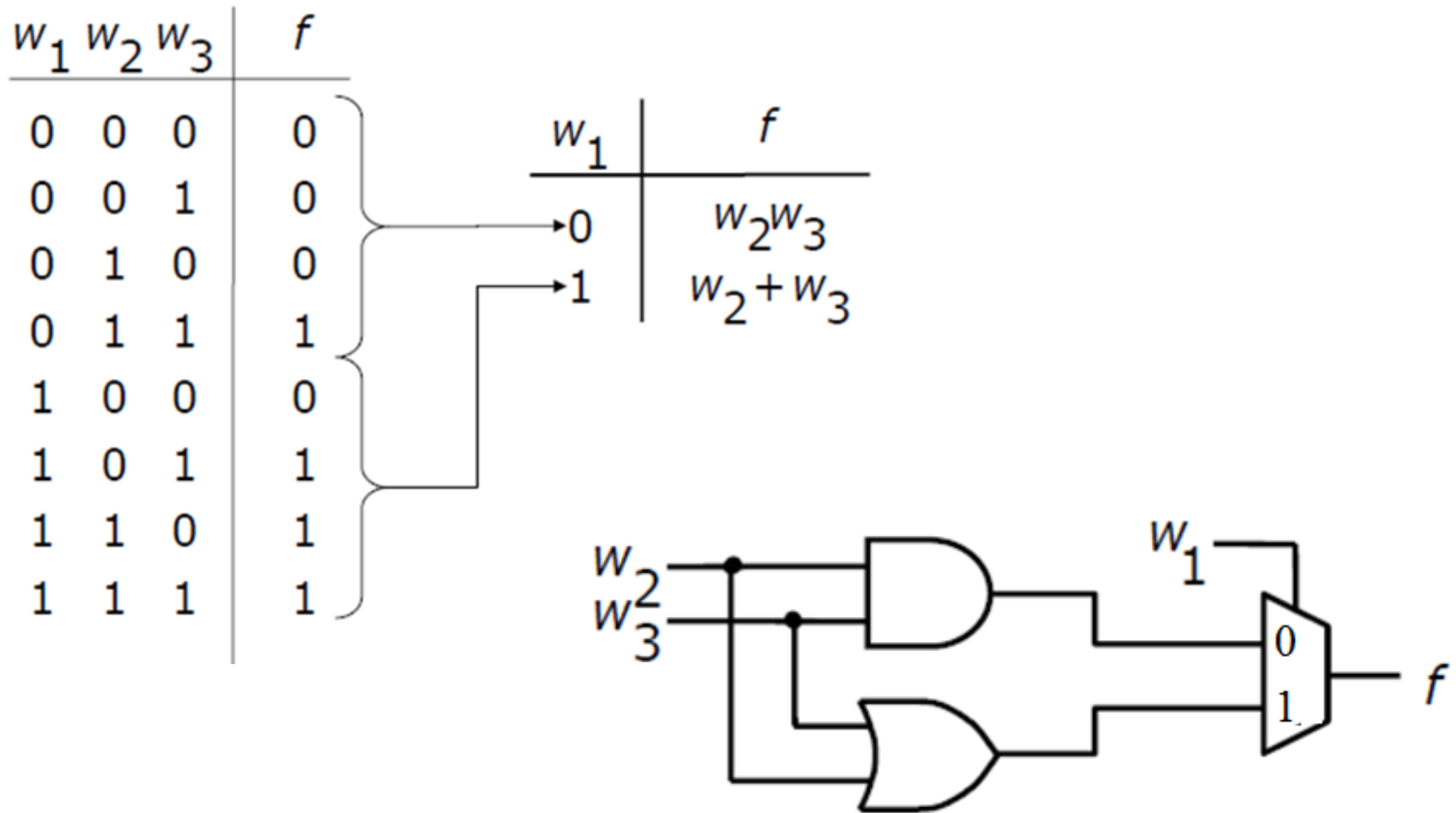
- Phân tích hàm này theo biến w_1 :

$$f(w_1, w_2, w_3) = \underbrace{w_1(w_2 + w_3)}_{\text{f khi } w_1=1} + \underbrace{\overline{w_1}(w_2 w_3)}_{\text{f khi } w_1=0}$$

f khi $w_1=1$

f khi $w_1=0$

Biểu thức Shannon



Biểu thức Shannon

- Ví dụ 2:

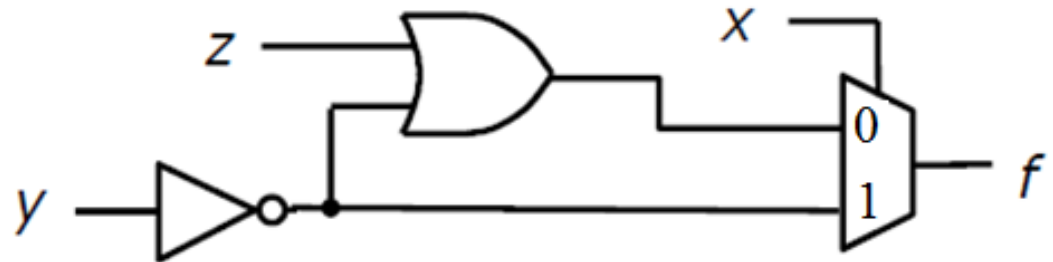
x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$f = x'y'z' + x'y'z + x'yz + xy'z' + xy'z$$

Chọn x làm biến mở rộng

$$f = x'(y'z' + y'z + yz) + x(y'z' + y'z)$$

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



Biểu thức Shannon

- Ví dụ 3:

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

$$f = x'y'z' + x'y'z + x'yz + xy'z' + xy'z$$

Chọn z làm biến mở rộng

Ví dụ

- Dùng MUX 4-to-1 và các cổng luận lý cần thiết để hiện thực hàm sau:

$$F(a, b, c, d) = \text{SOP}(1, 3, 5, 6, 8, 11, 15)$$

- Yêu cầu: *c* và *d* là các ngõ vào điều khiển của MUX 4-ra-1

f

Ví dụ

- Dùng MUX 4-to-1 và các công luận lý cần thiết để hiện thực hàm sau:

$$F(a, b, c, d) = \text{SOP}(1, 3, 5, 6, 8, 11, 15)$$

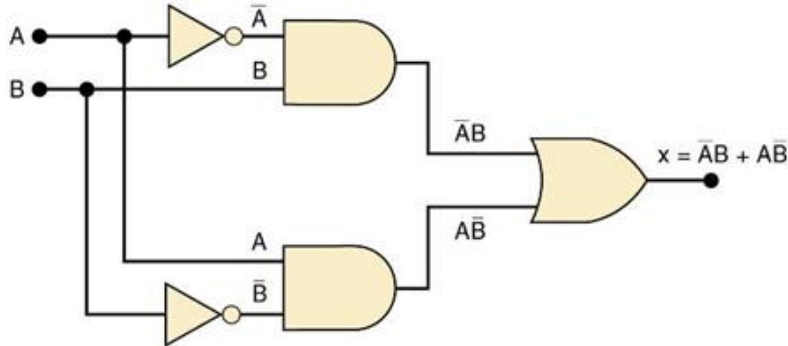
- Yêu cầu: ***b*** và ***c*** là các ngõ vào điều khiển của MUX 4-to-1



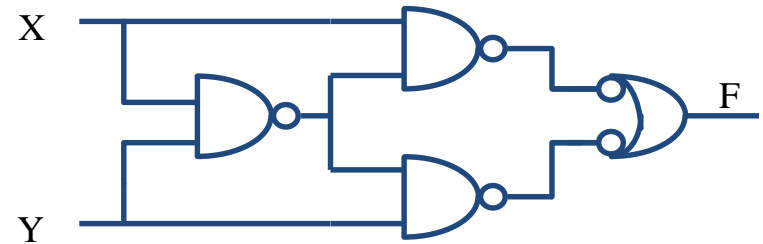
7. Parity Generator/ Checker

Cổng Exclusive OR và Exclusive NOR

$$\text{XOR: } X \oplus Y = X' \cdot Y + X \cdot Y'$$

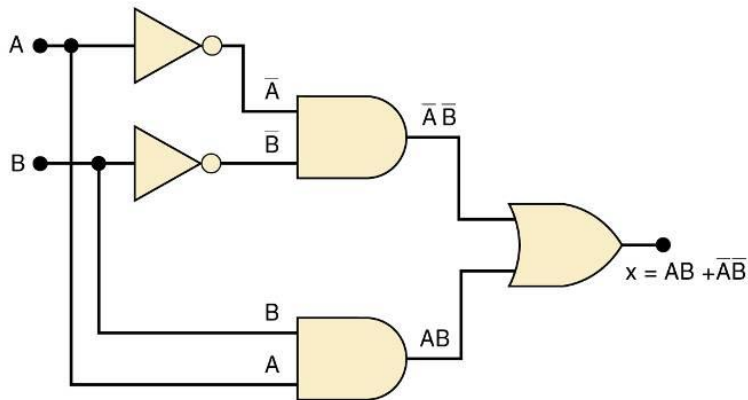


XOR: AND-OR



XOR: NAND 3 cấp

$$\text{XNOR: } (X \oplus Y)' = X \cdot Y + X' \cdot Y'$$



X	Y	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Bảng sự thật

XOR and XNOR Symbols

- Các biểu tượng tương đương của cổng XOR



- Các biểu tượng tương đương của cổng XNOR



Quy tắc tạo cổng XOR/XNOR tương đương:

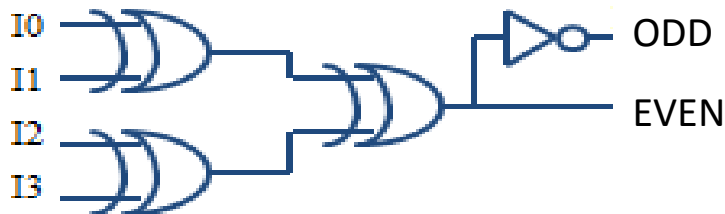
Lấy bù bất kì 2 trong 3 tín hiệu (cả input hoặc output) của cổng XOR/XNOR ban đầu.

Ứng dụng của XOR: Mạch Parity

- **Mạch Parity chẵn (EVEN):** tổng số bit 1 trong chuỗi bit (kể cả bit parity) là số chẵn.
 - Parity bit = 1, nếu số bit 1 trong chuỗi bit (không kể parity) là số lẻ
 - Parity bit = 0, nếu số bit 1 trong chuỗi bit (không kể parity) là số chẵn
- **Mạch Parity lẻ (ODD):** tổng số bit 1 trong chuỗi bit (kể cả bit parity) là số lẻ.
- Ví dụ: Mạch tạo Parity 4-bit



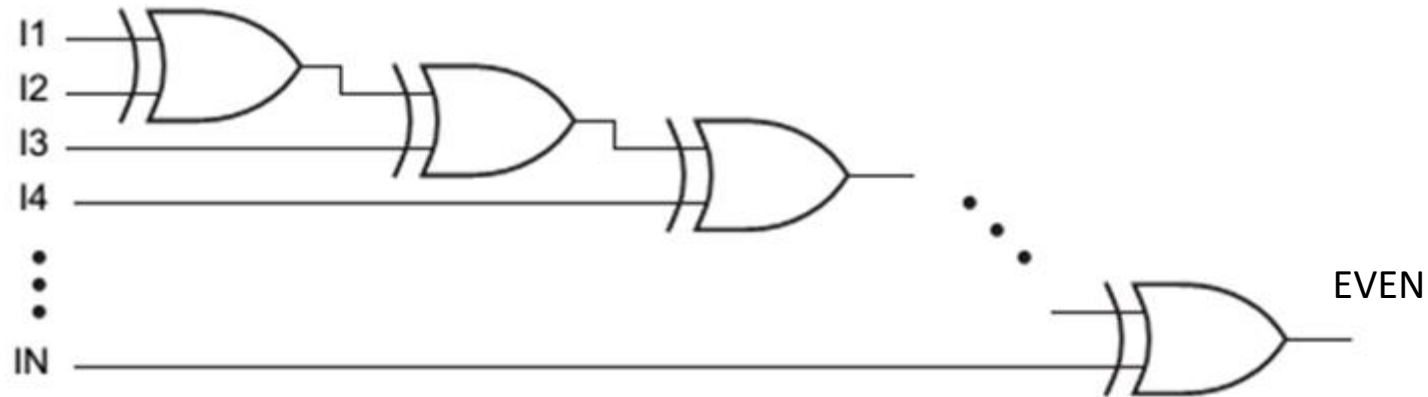
Daisy-Chain Structure



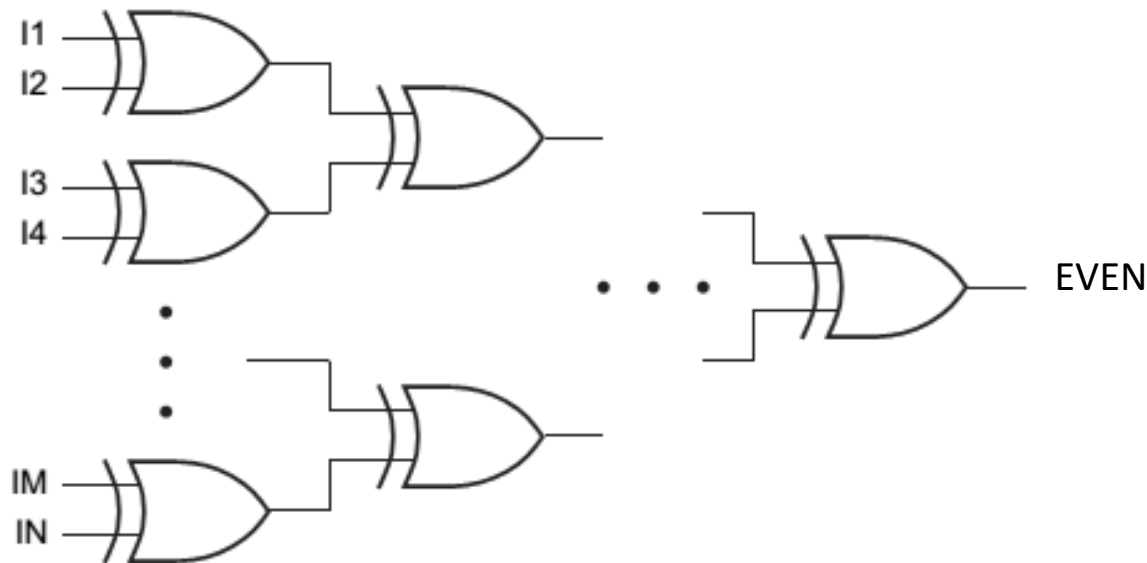
Tree structure

Input	Even Parity	Odd Parity
0000	0	1
0001	1	0
1101	1	0
1111	0	1
1100	0	1

Ứng dụng của XOR: Mạch tạo Parity



Daisy-Chain Structure



Tree structure

Ứng dụng của XOR: Mạch Parity

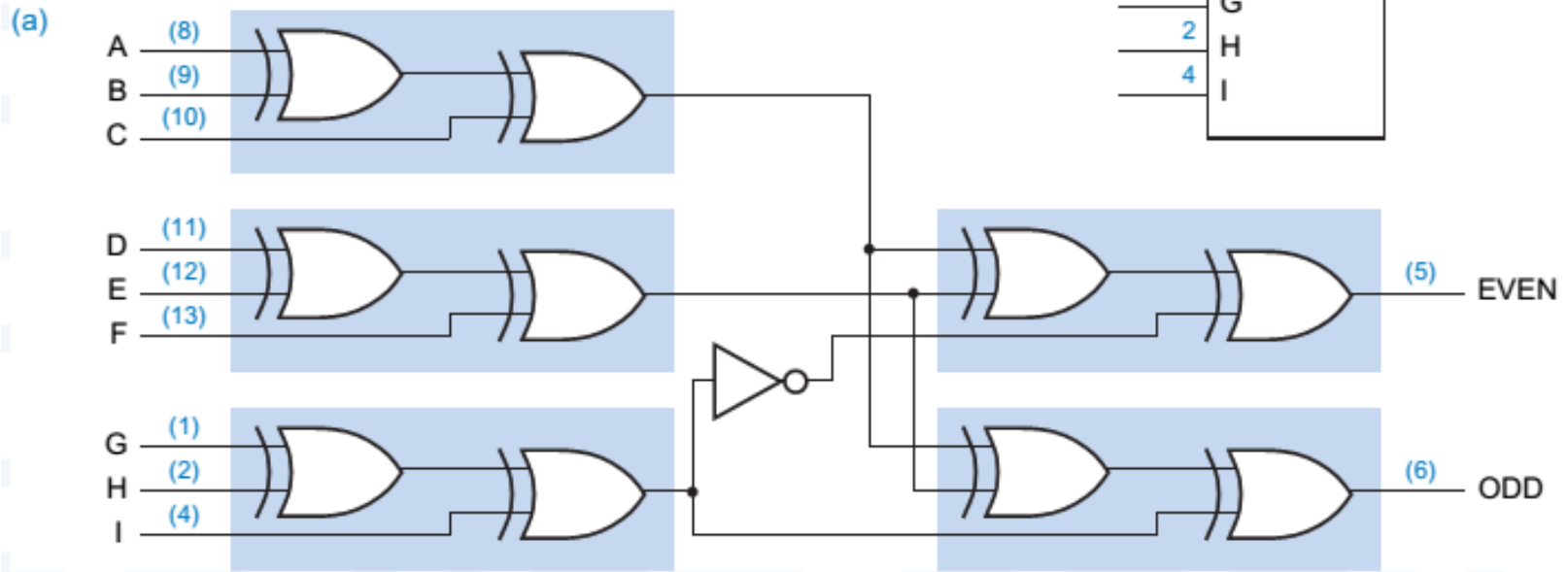
<i>Information Bits</i>	<i>Even-parity Code</i>	<i>Odd-parity Code</i>
000	000 0	000 1
001	001 1	001 0
010	010 1	010 0
011	011 0	011 1
100	100 1	100 0
101	101 0	101 1
110	110 0	110 1
111	111 1	111 0

7 bits of data	(count of 1 bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

Source: http://en.wikipedia.org/wiki/Parity_bit

Chip 74x280: tạo parity chẵn/lẻ từ dữ liệu 9-bit

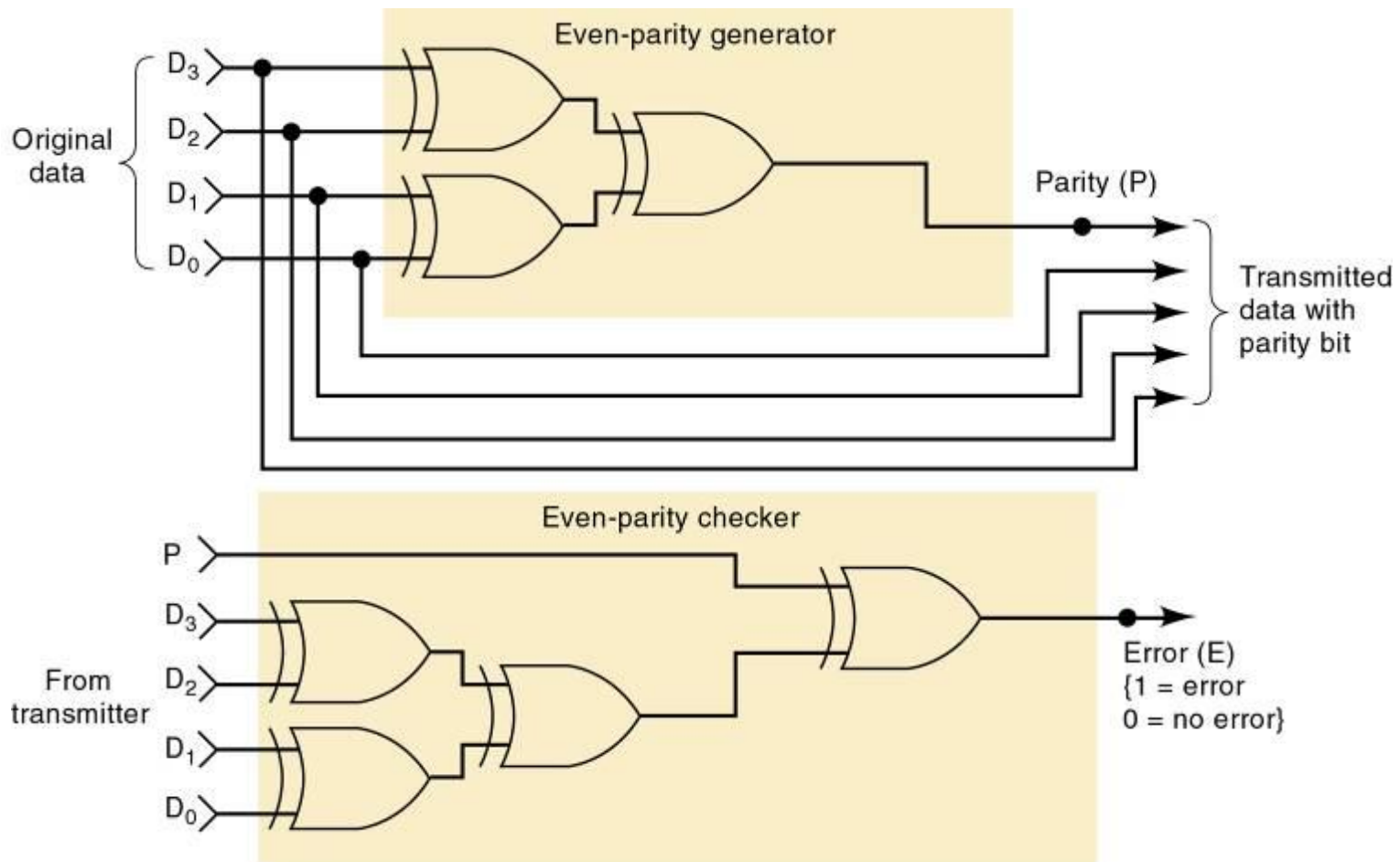
Chú ý: Chip 74x280 định nghĩa even/odd parity ngược lại với định nghĩa trong các slide trước
=> cẩn thận khi sử dụng chip này



The 74x280 9-bit odd/even parity generator: (a) logic diagram, including pin numbers for a standard 16-pin dual in-line package; (b) traditional logic symbol.

Bộ tạo và kiểm tra Parity (Parity generator and checker)

- Cổng XOR và XNOR rất hữu dụng trong các mạch với mục đích **tạo** (bộ phát) và **kiểm tra** (bộ nhận) parity bit



Ứng dụng của XOR: Mạch tạo Parity

Type of bit parity	Successful transmission scenario
Even parity	<p>A wants to transmit: 1001</p> <p>A computes parity bit value: $1 \wedge 0 \wedge 0 \wedge 1 = 0$</p> <p>A adds parity bit and sends: 10010</p> <p>B receives: 10010</p> <p>B computes parity: $1 \wedge 0 \wedge 0 \wedge 1 \wedge 0 = 0$</p> <p>B reports correct transmission after observing expected even result.</p>
Odd parity	<p>A wants to transmit: 1001</p> <p>A computes parity bit value: $\sim(1 \wedge 0 \wedge 0 \wedge 1) = 1$</p> <p>A adds parity bit and sends: 10011</p> <p>B receives: 10011</p> <p>B computes overall parity: $1 \wedge 0 \wedge 0 \wedge 1 \wedge 1 = 1$</p> <p>B reports correct transmission after observing expected odd result.</p>

Ứng dụng cổng XOR: kiểm tra Parity

Type of bit parity error	Failed transmission scenario
Even parity Error in the second bit	A wants to transmit: 1001 A computes parity bit value: $1^{\wedge}0^{\wedge}0^{\wedge}1 = 0$ A adds parity bit and sends: 10010 ...TRANSMISSION ERROR... B receives: 11010 B computes overall parity: $1^{\wedge}1^{\wedge}0^{\wedge}1^{\wedge}0 = 1$ B reports incorrect transmission after observing unexpected odd result.
Even parity Error in the parity bit	A wants to transmit: 1001 A computes even parity value: $1^{\wedge}0^{\wedge}0^{\wedge}1 = 0$ A sends: 10010 ...TRANSMISSION ERROR... B receives: 10011 B computes overall parity: $1^{\wedge}0^{\wedge}0^{\wedge}1^{\wedge}1 = 1$ B reports incorrect transmission after observing unexpected odd result.



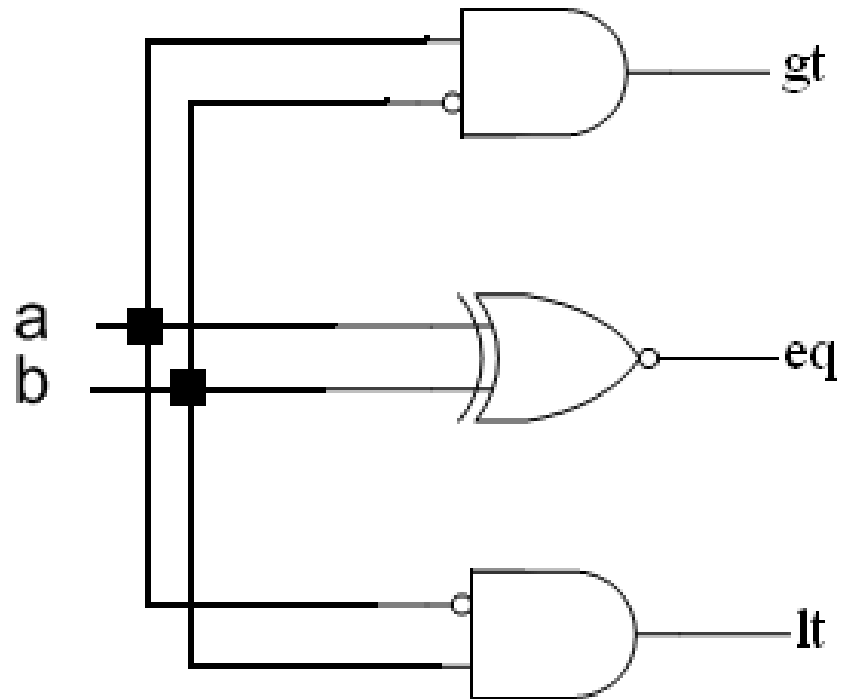
8. Comparator

Mạch so sánh (Comparator)

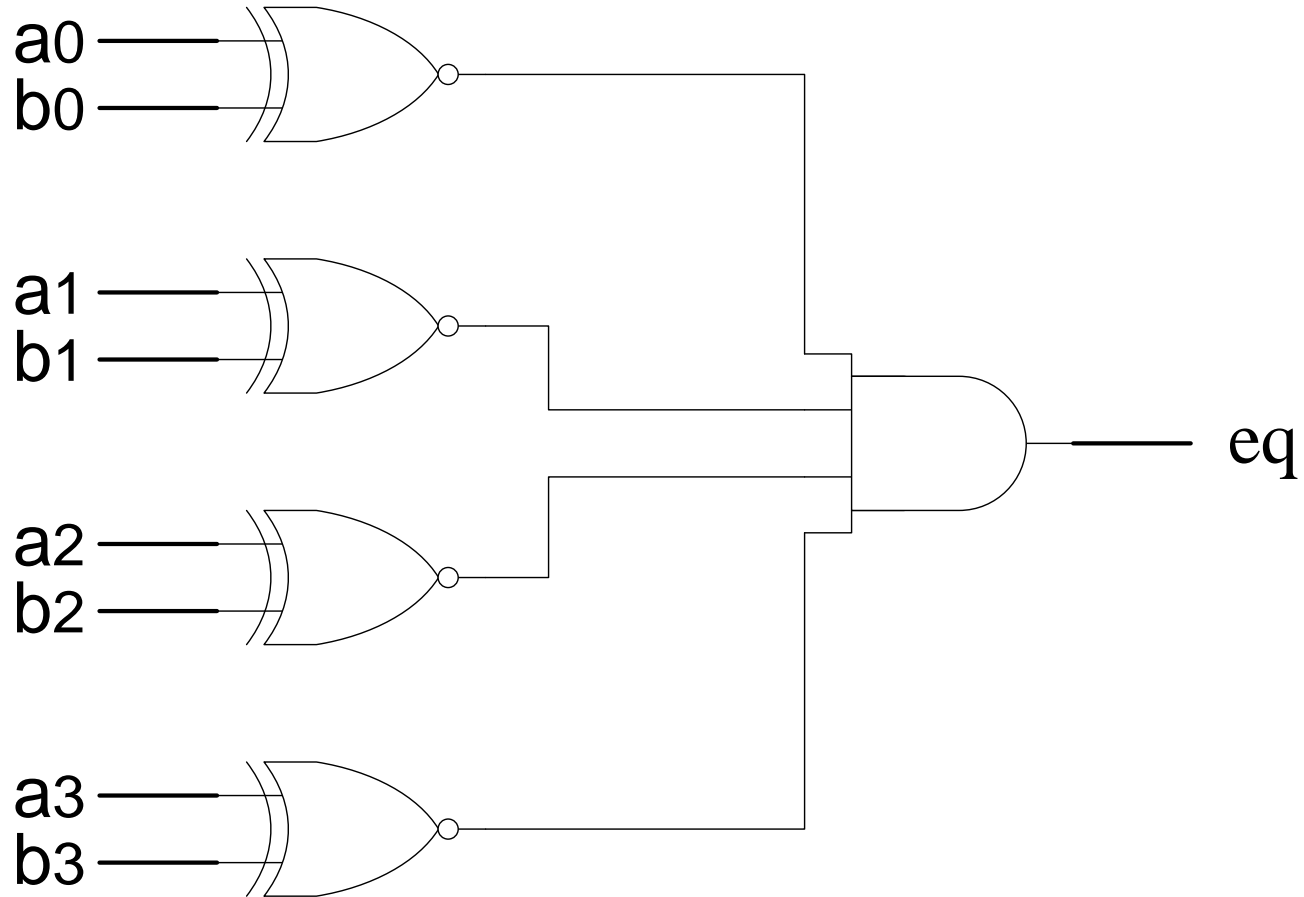
- Mạch so sánh 2 số
 - Xuất ra 1 nếu chúng bằng nhau
 - Xuất ra 0 nếu chúng khác nhau
- Dựa trên cổng **XOR**, trả về 0 nếu ngõ vào giống nhau và 1 nếu chúng khác nhau
- Dựa trên cổng **XNOR**, trả về 1 nếu ngõ vào giống nhau và 0 nếu chúng khác nhau

Mạch so sánh 1 bit

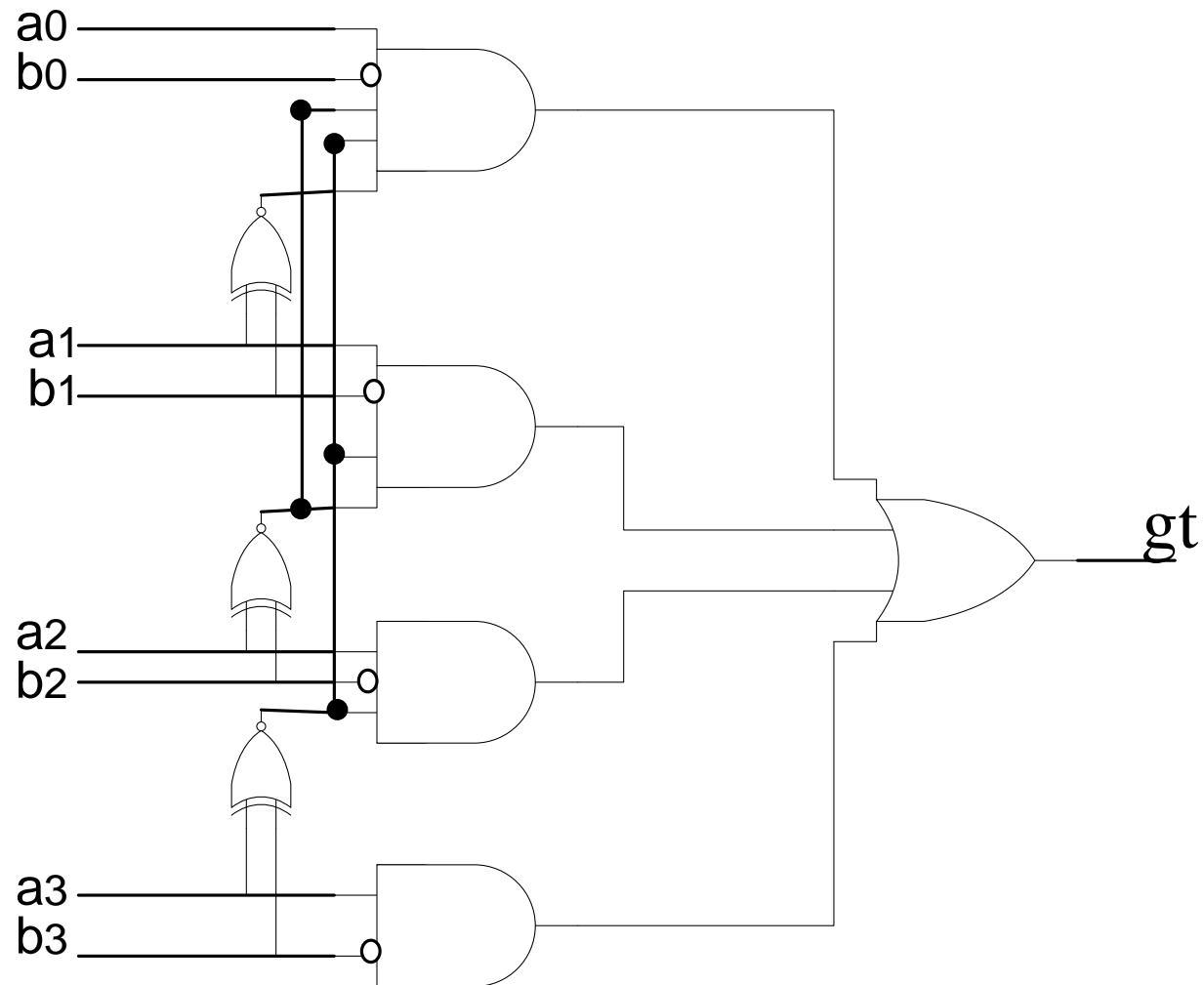
a	b	gt	eq	lt
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0



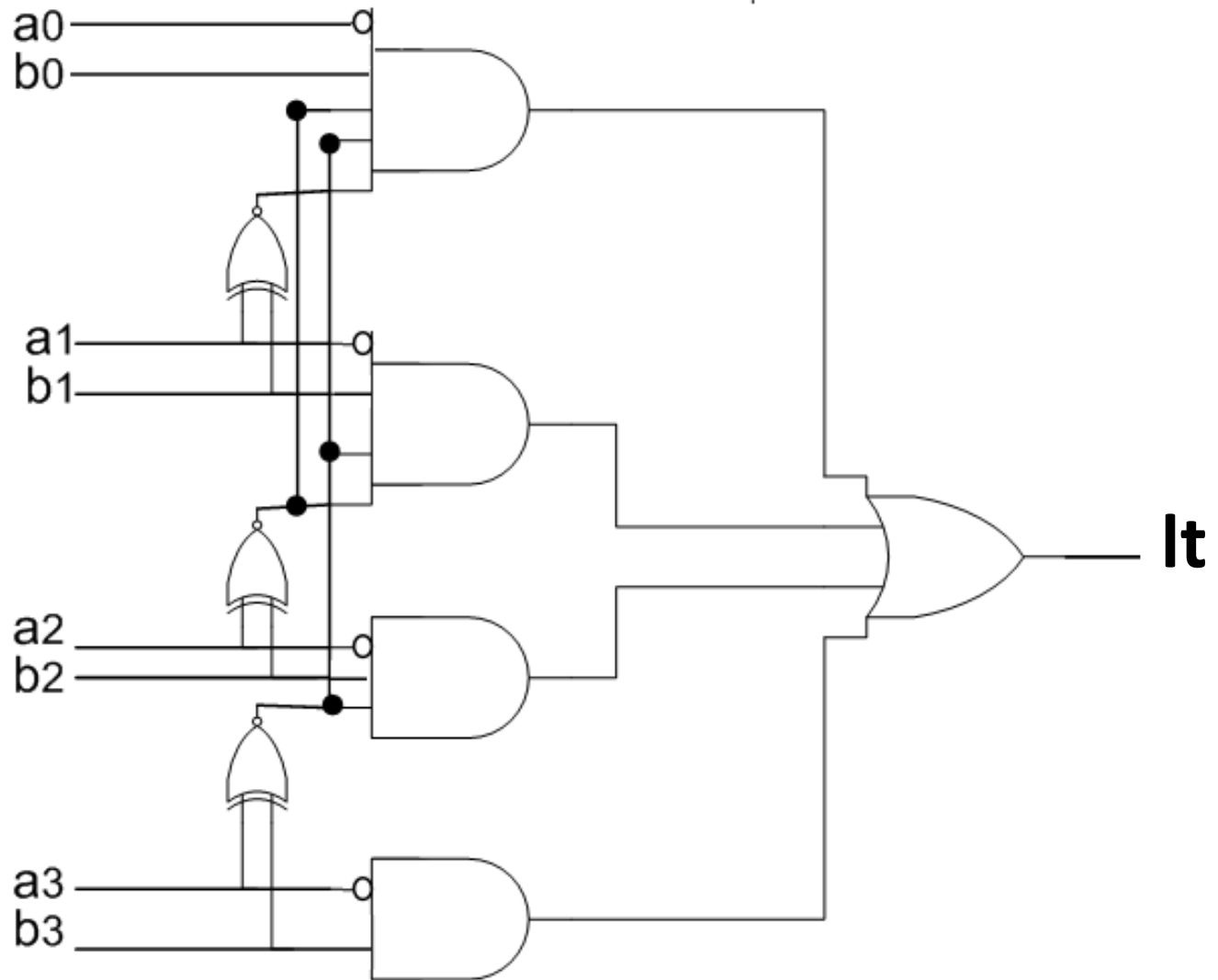
Mạch so sánh 4 bit



Mạch so sánh 4 bit



Mạch so sánh 4 bit



Mạch so sánh 4-bit

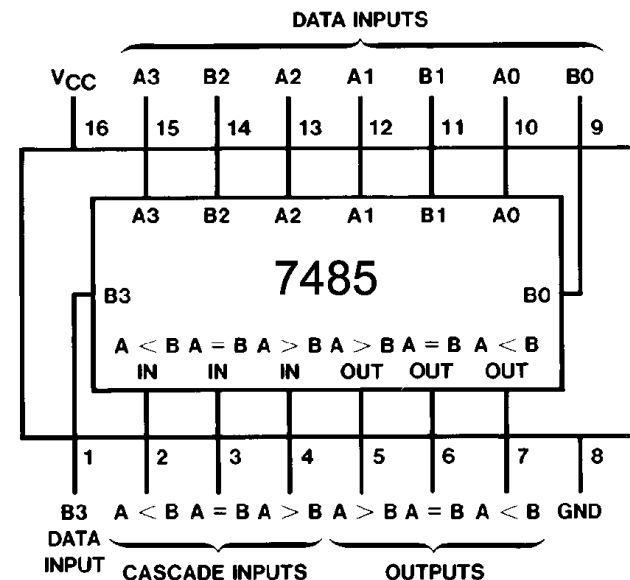
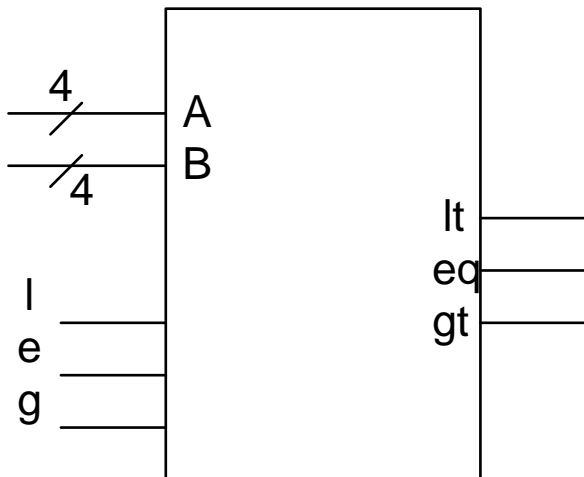
- **74x85** là mạch so sánh tiêu chuẩn với những đặc tính sau:

if (A>B) lt=0, eq=0, gt=1

if (A<B) lt=1, eq=0, gt=0

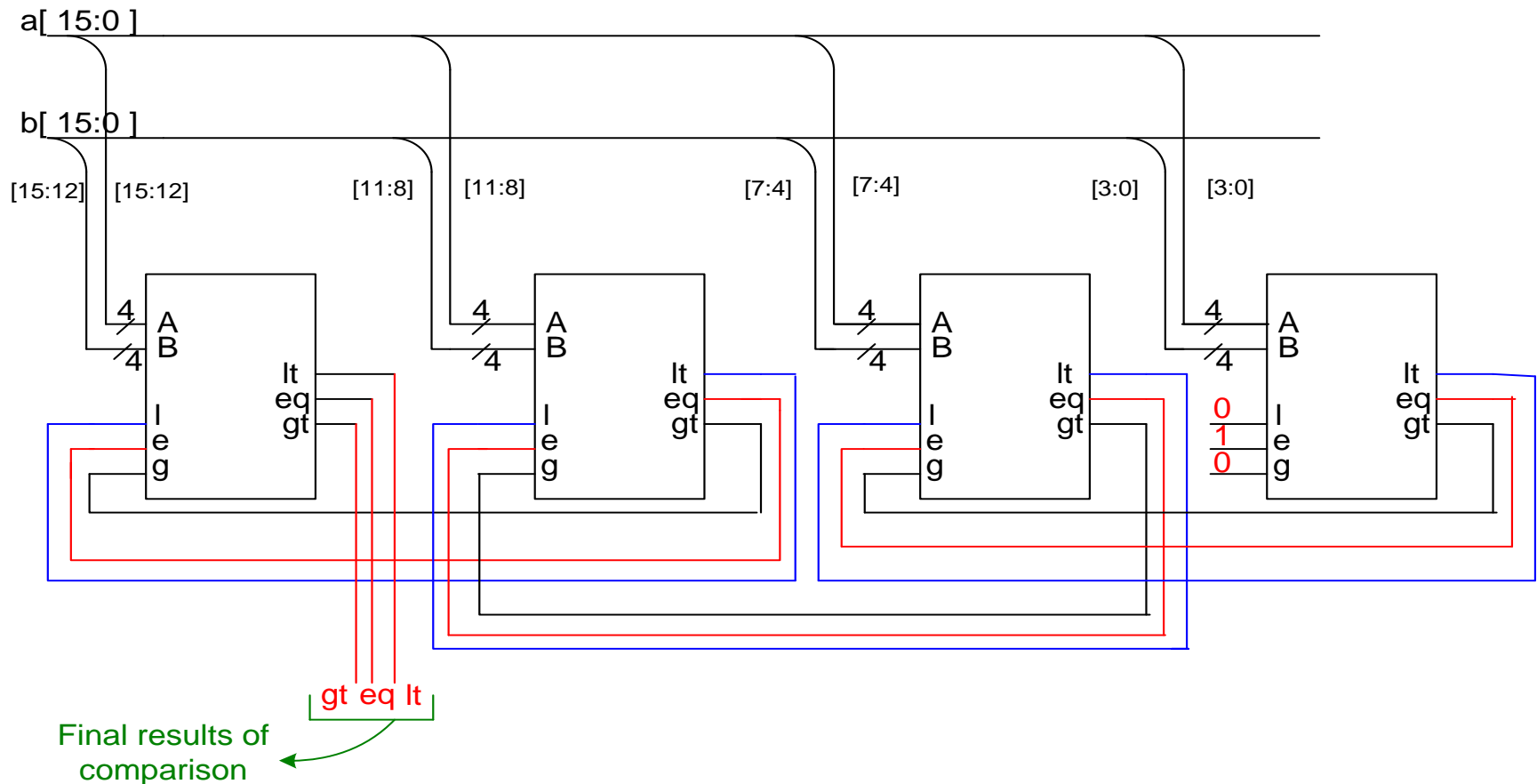
if (A=B) lt=**l**, eq=**e**, gt=**g**

- **Chú ý:** 3 ngõ vào **l, e và g** được sử dụng khi ghép nối để tạo mạch so sánh với số bit nhiều hơn



Mạch so sánh 16 bit

- Ghép 4 IC 74x85 để xây dựng một mạch so sánh 16 bit



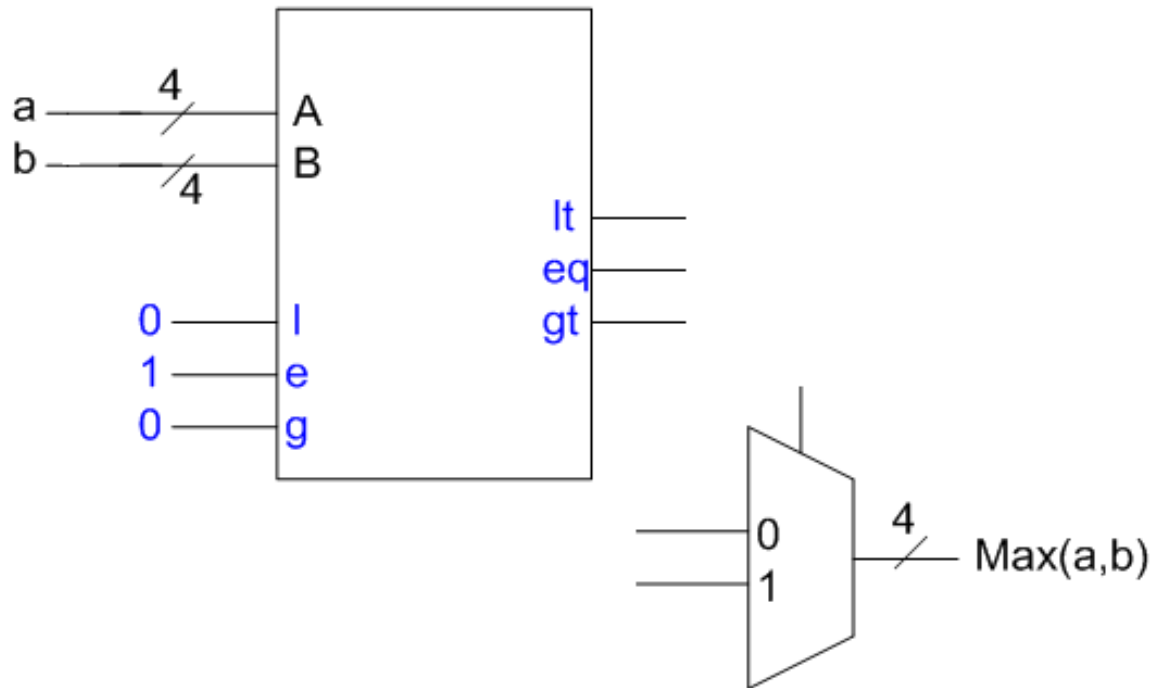
Mạch so sánh 16 bit



- Mạch sẽ đầu tiên so sánh 4 bit cao nhất ở 2 ngõ vào
- Trong bước đầu này:
 - ‘a’ nhỏ hơn ‘b’ nếu 4 MSB nhỏ hơn
 - ‘a’ lớn hơn ‘b’ nếu 4 MSB lớn hơn
 - Nếu 4 MSB của 2 số bằng nhau, kết quả của phép so sánh sẽ chờ cho kết quả so sánh ở vị trí các bit thấp hơn

Ví dụ

- Thiết kế mạch tìm số lớn nhất, số nhỏ nhất trong 4 số 4-bit sử dụng mạch so sánh và MUXs





Any question?