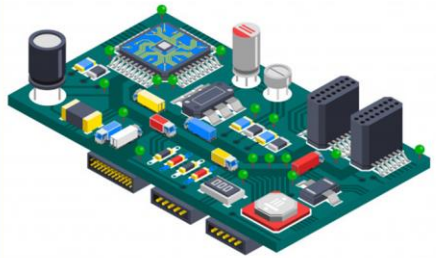
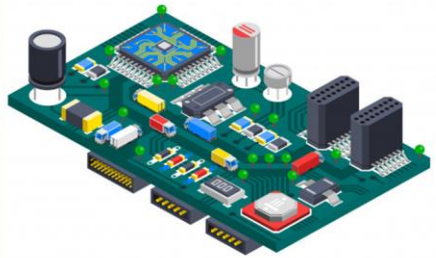
PHENIKAA UNIVERSITY



Các bước thiết kế mạch logic tổ hợp

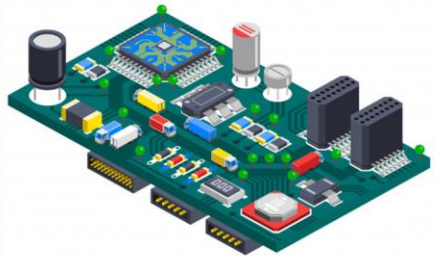
- B1: Đặt các biến ngõ vào và ngõ ra
- B2: Lập bảng sự thật
- B3: Viết biểu thức hàm logic liên hệ ngõ vào và ngõ ra
- B4: Rút gọn hàm logic theo Boole hoặc bìa Kanaugh
- B5: Vẽ mạch

PHENIKAA UNIVERSITY



6. Vi mạch tổ hợp

- 6.1 Mạch cộng cơ bản
- 6.2 Mạch cộng nhị phân song song
- 6.3 Truyền số nhớ và thấy trước số nhớ
- 6.4 Mạch so sánh
- 6.5 Mạch giải mã
- 6.6 Mạch mã hóa
- 6.7 Mạch biến đổi mã
- 6.8 Mạch ghép kênh (chọn kênh)
- 6.9 Mạch giải ghép kênh (phân đường)
- 6.10 Mạch kiểm tra chẵn lẻ



6.1 Mạch cộng cơ bản

- Mạch cộng bán phần (không đầy đủ)(half-adder)

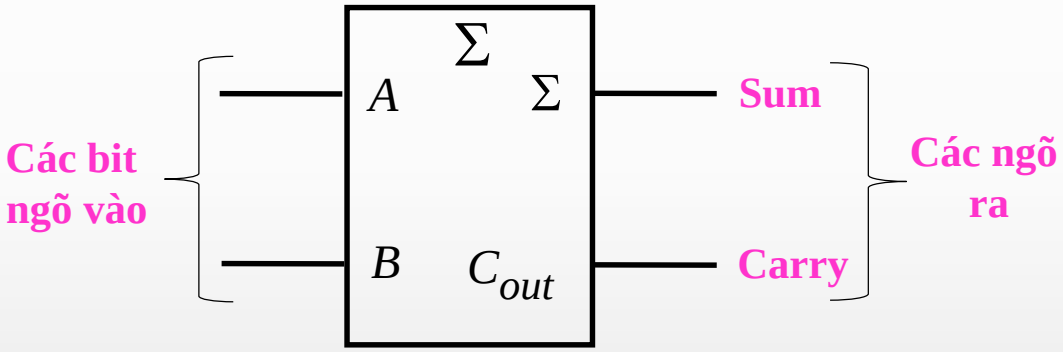
Cộng nhị phân

0 + 0 = 0

0 + 1 = 1

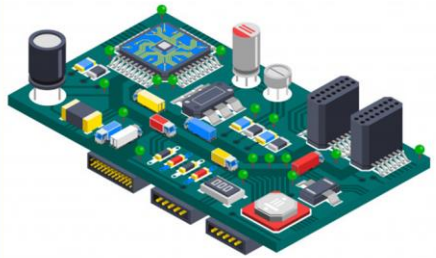
1 + 0 = 1

1 + 1 = 10



Mạch cộng bán phần HA

Mạch cộng bán phần (không đầy đủ) nhận hai bit (digit) nhị phân trên các ngõ vào và tạo ra hai bit (digit) nhị phân trên những ngõ ra, bit tổng (sum bit) và bit nhớ (carry bit).



6.1 Mạch cộng cơ bản

- Mạch cộng bán phần (không đầy đủ)

A	B	C_{out}	∑
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

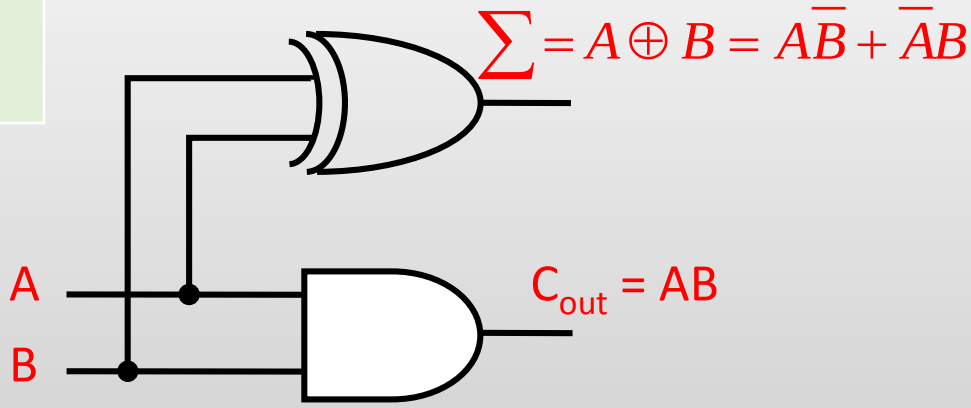
∑ = tổng
C_{out} = số nhớ ngõ ra
A và B = các biến ngõ vào (toán hạng)

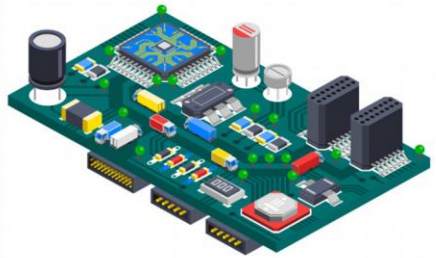
Viết các hàm theo dạng chính tắc 1:

$$\sum = \overline{A}B + A\overline{B}$$

$$\sum = A \oplus B$$

C_{out} = AB

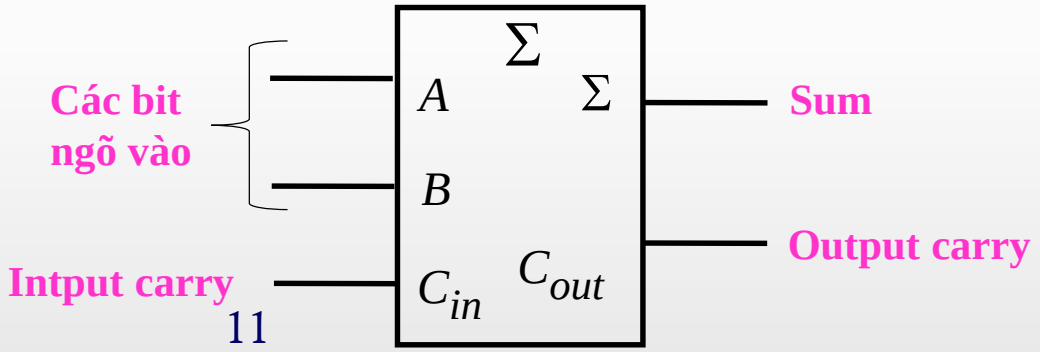




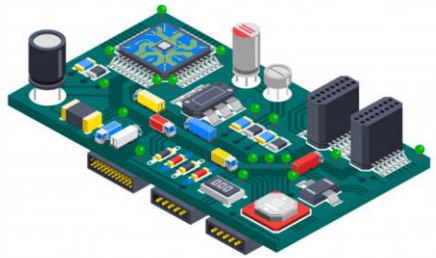
6.1 Mạch cộng cơ bản

- Mạch cộng toàn phần FA (đầy đủ)(full-adder)

Mạch cộng toàn phần (đầy đủ) nhận hai bit (digit) dữ liệu ngõ vào và bit nhớ ngõ vào (input carry), tạo ra hai bit (digit) trên những ngõ ra, bit tổng (sum bit) và bit nhớ ngõ ra (output carry).



11
1011
+0110
10001



6.1 Mạch cộng cơ bản

- Mạch cộng toàn phần (đầy đủ)

A	B	C_{In}	C_{Out}	∑
0	0	0	0	0
0	0	1	0	1
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	1

C_{in}= số nhớ ngõ vào, đôi khi ký hiệu là CI

C_{out}= số nhớ ngõ ra, đôi khi ký hiệu là CO

∑ = tổng

A và B = các biến ngõ vào (các toán hạng)

$$\sum = \overline{A}\overline{B}C_{in} + \overline{A}B\overline{C_{in}} + A\overline{B}\overline{C_{in}} + ABC_{in}$$

$$\sum = (\overline{A}B + A\overline{B})\overline{C_{in}} + (\overline{A}\overline{B} + AB)C_{in}$$

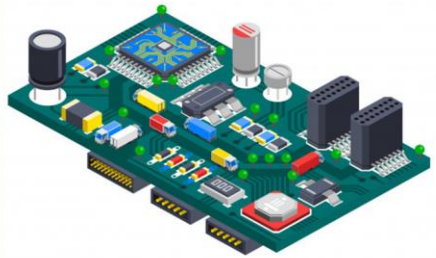
$$\sum = (A \oplus B)\overline{C_{in}} + (\overline{A \oplus B})C_{in}$$

$$\boxed{\sum = (A \oplus B) \oplus C_{in}}$$

$$C_{out} = \overline{A}BC_{in} + A\overline{B}C_{in} + AB\overline{C_{in}} + ABC_{in}$$

$$C_{out} = AB + (\overline{A}B + A\overline{B})C_{in}$$

$$\boxed{C_{out} = AB + (A \oplus B)C_{in}}$$



6.1 Mạch cộng cơ bản

- Lưu ý:

  Cần ghi nhớ:

  $C_{out} = A.B + (A \oplus B).C_{in}$

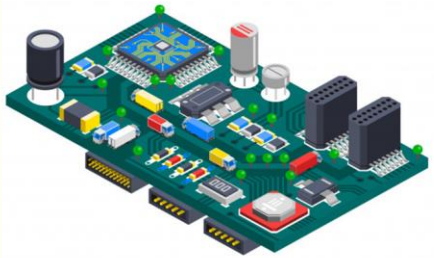
  $C_{out} = A.B + (A \oplus B).C_{in} + A.B.C_{in}$

  $C_{out} = A.B + (A \oplus B + A.B)C_{in}$

  $C_{out} = A.B + (A.B + \overline{A}.B + A.\overline{B}).C_{in}$

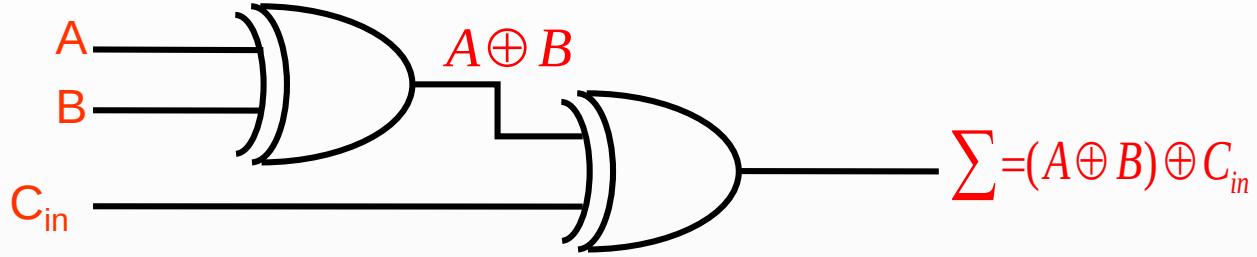
  $C_{out} = A.B + (A.B + \overline{A}.B + A.B + A.\overline{B}).C_{in}$

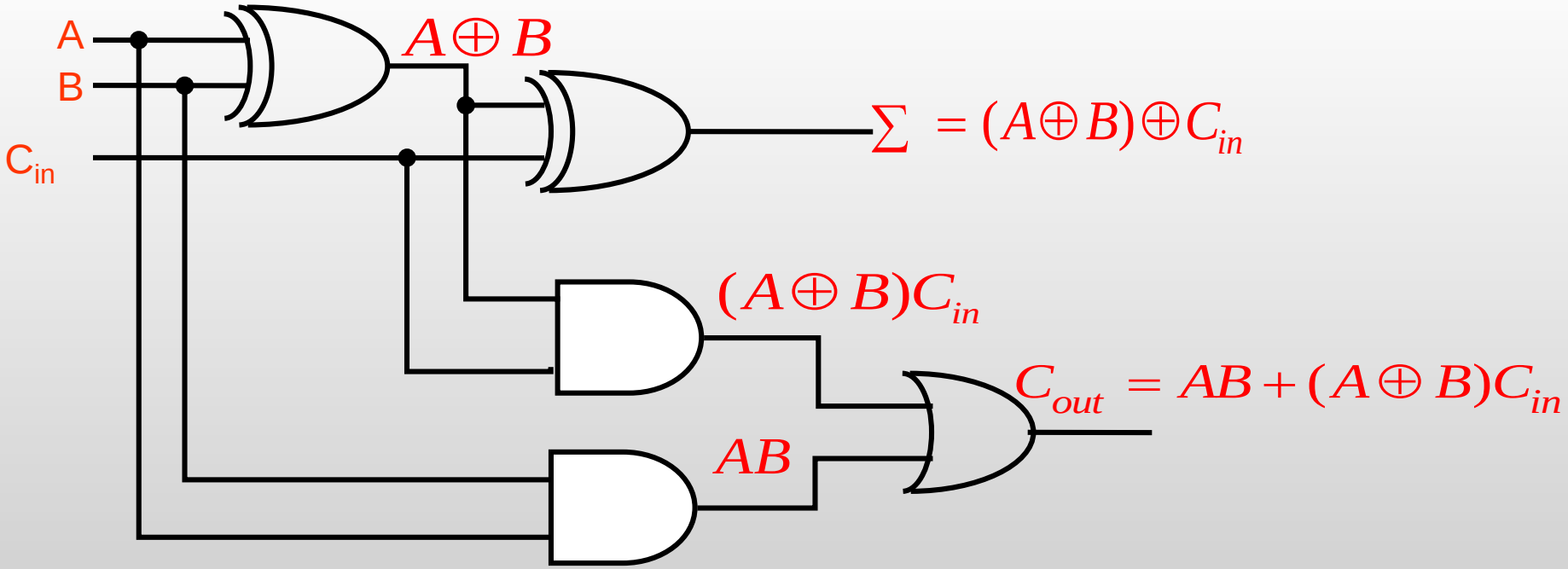
  $\boxed{C_{out} = A.B + (A + B).C_{in}}$

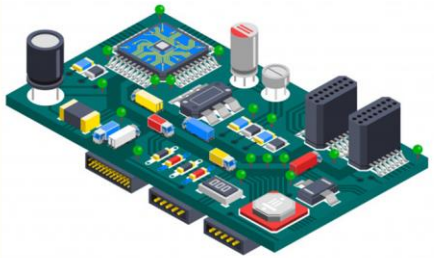


6.1 Mạch cộng cơ bản

- Mạch cộng toàn phần (đầy đủ)

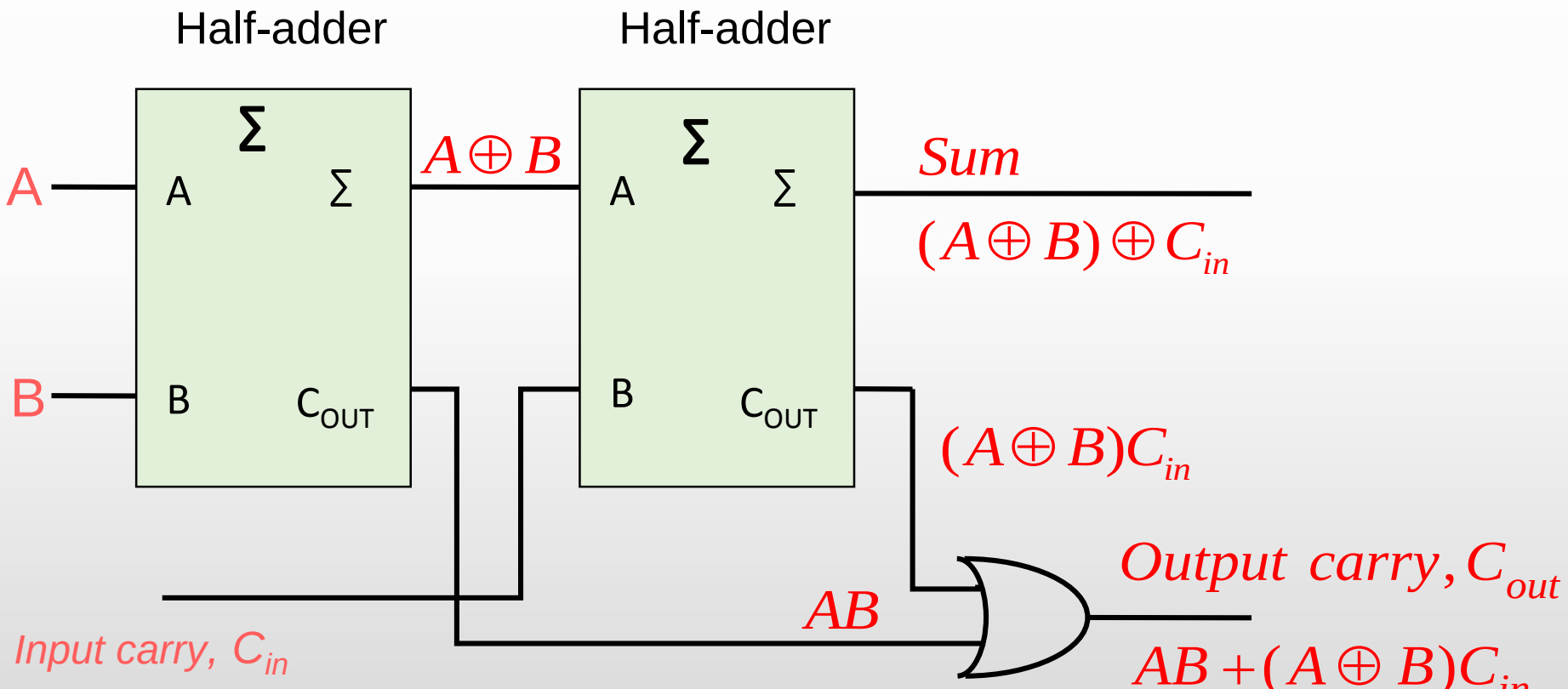


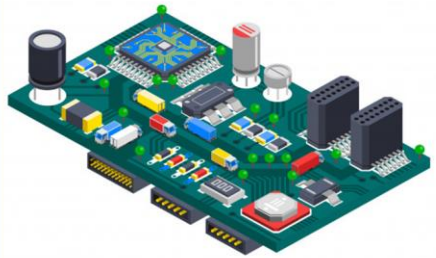




6.1 Mạch cộng cơ bản

- Mạch cộng toàn phần (đầy đủ)



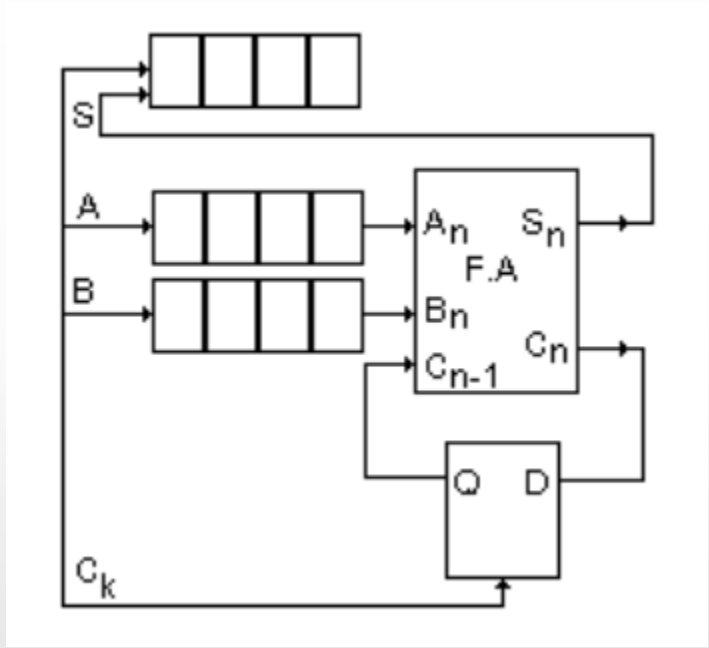


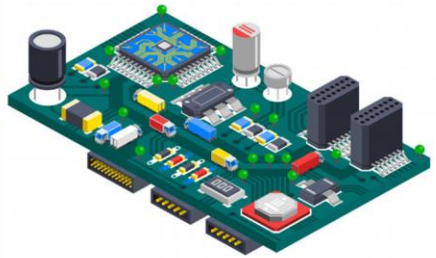
6.2 Mạch cộng nhị phân song song

- Giới thiệu

Mạch cộng nhị phân nối tiếp

Trong cách cộng nối tiếp, người ta dùng các ghi dịch để chuyển các bit vào một mạch cộng toàn phần duy nhất, số nhớ từ ngã ra C_n được làm trễ một bit nhờ FF D và đưa vào ngã vào C_{n-1}. Như vậy tốc độ của phép cộng tùy thuộc vào tần số xung C_K và số bit phải thực hiện.





6.2 Mạch cộng nhị phân song song

- Giới thiệu

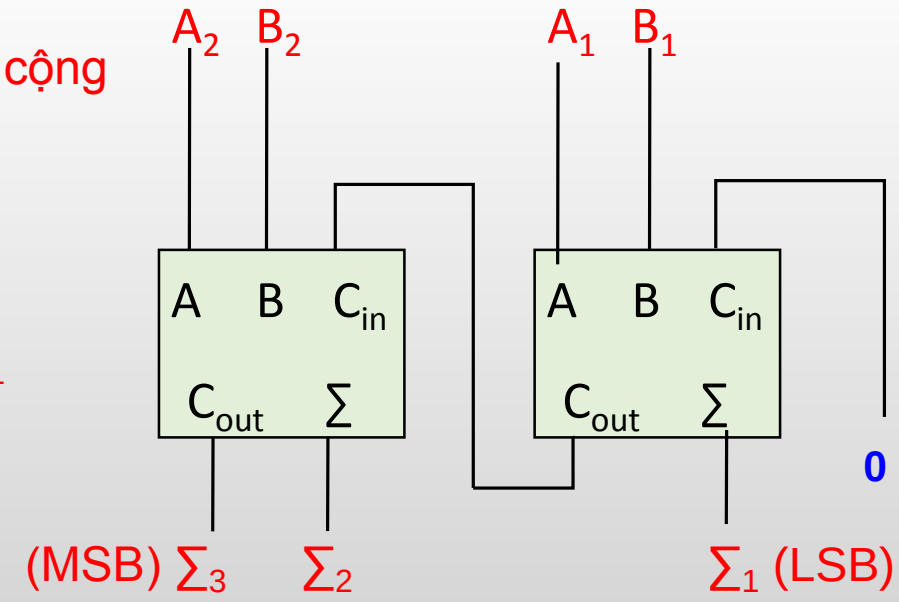
Mạch cộng 2-bit song song

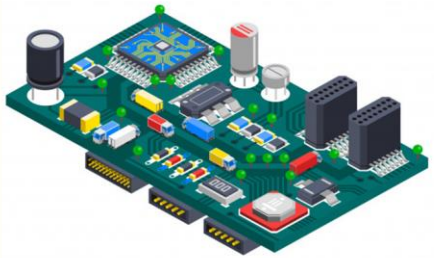
Trong cách cộng song song, các bit được đưa đồng thời vào các mạch cộng toàn phần và số nhớ của kết quả ở bit thấp được đưa lên bit cao hơn

Dạng tổng quát, cộng hai số 2-bit:

$A_2\ A_1$
$+\ B_2\ B_1$

$\Sigma_3\ \Sigma_2\ \Sigma_1$

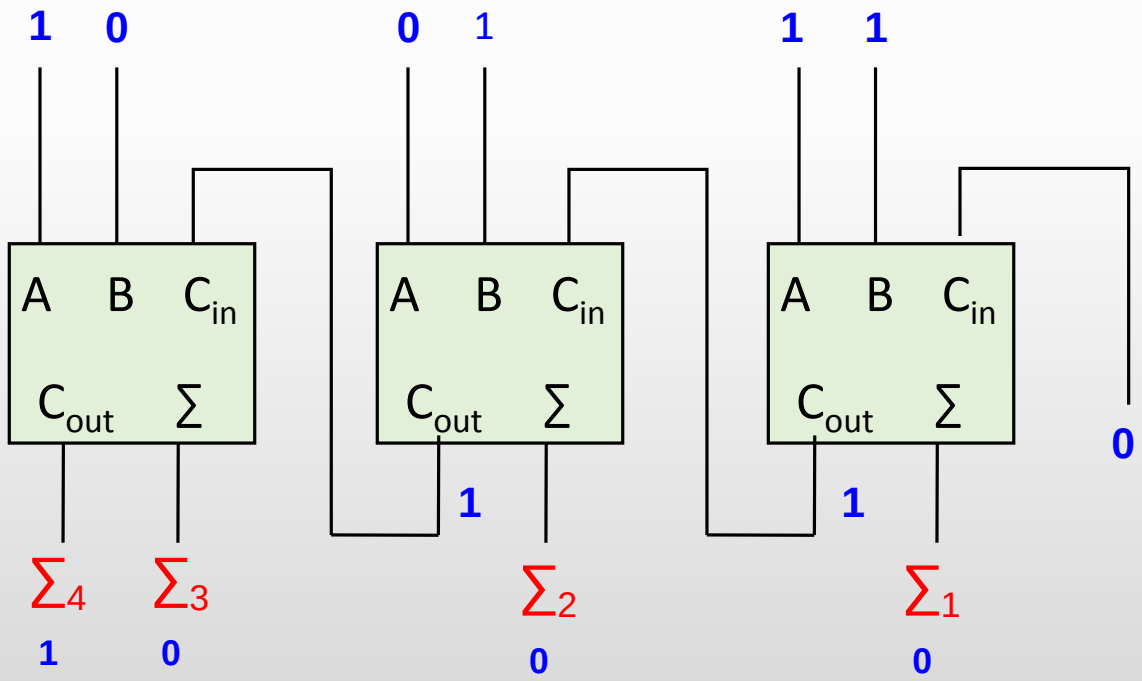


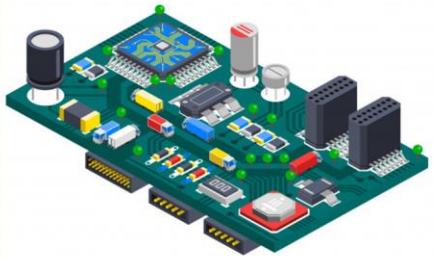


6.2 Mạch cộng nhị phân song song

- Giới thiệu

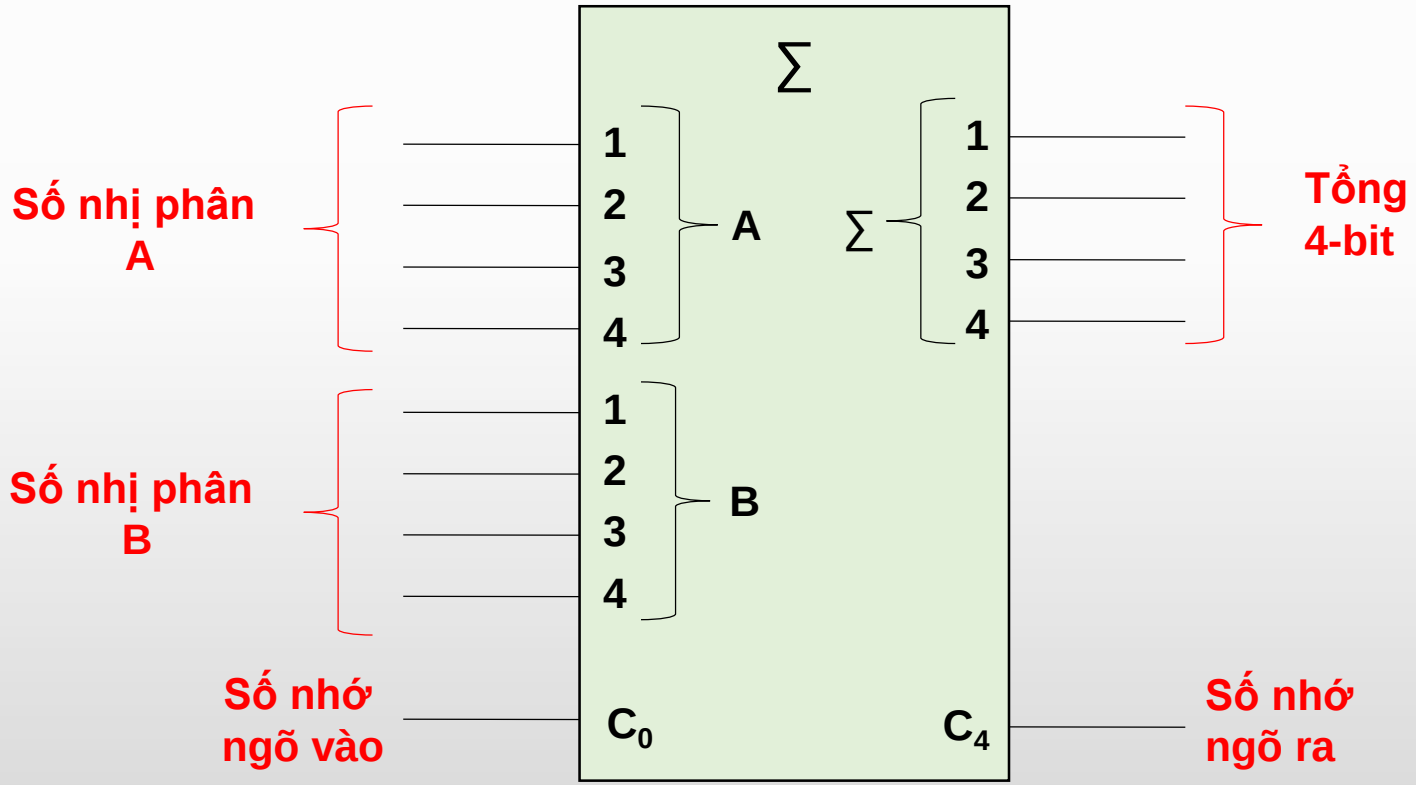
  Mạch cộng 3-bit song song

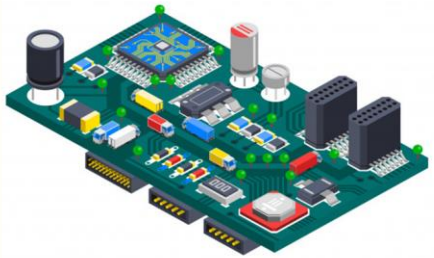




6.2 Mạch cộng nhị phân song song

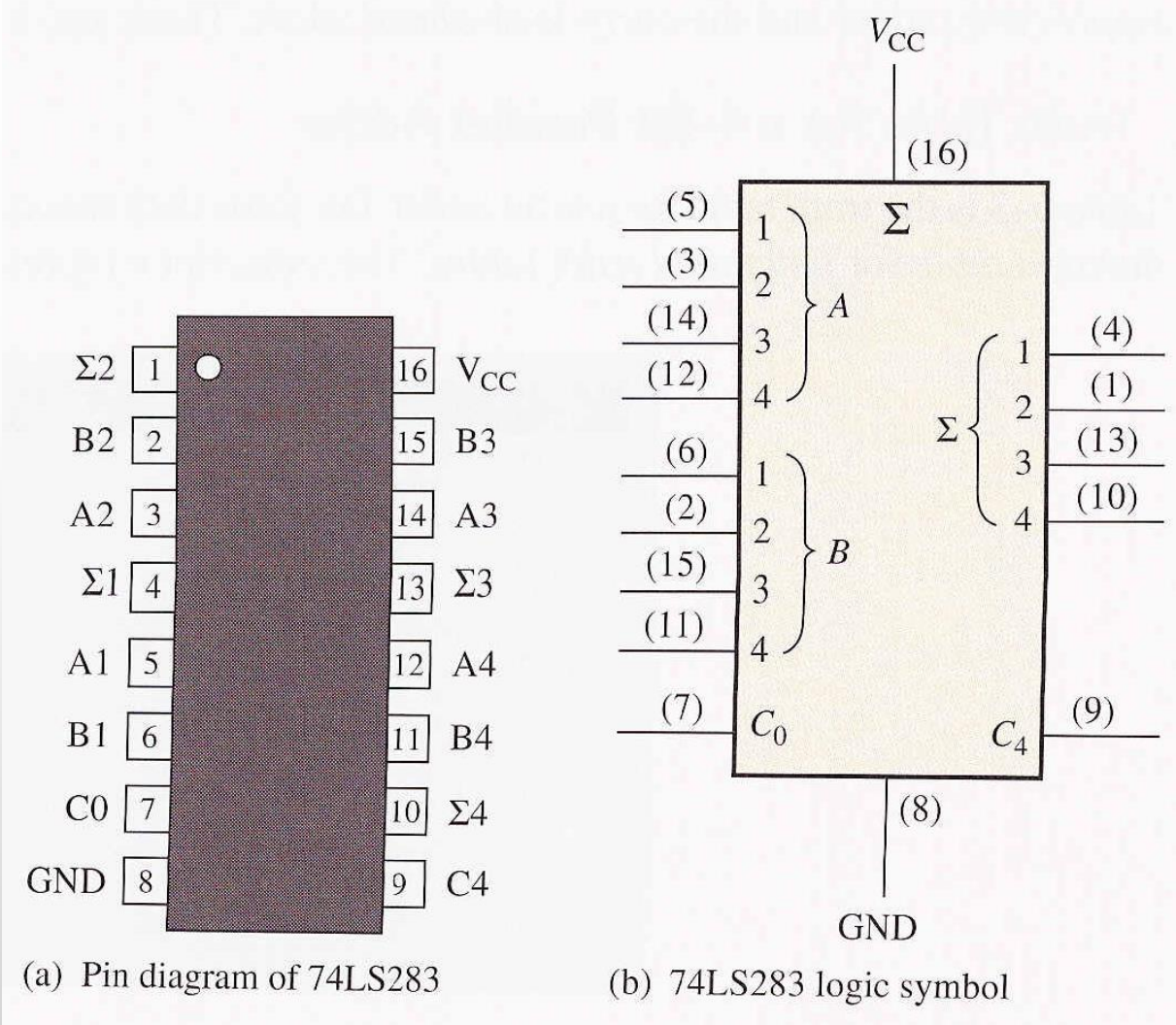
- Mạch cộng 4-bit song song





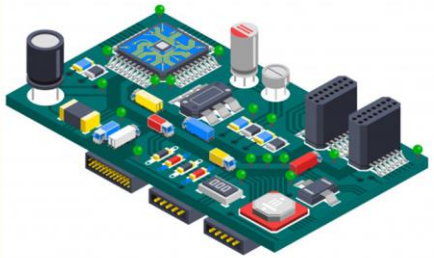
6.2 Mạch cộng nhị phân song song

- IC 74LS283



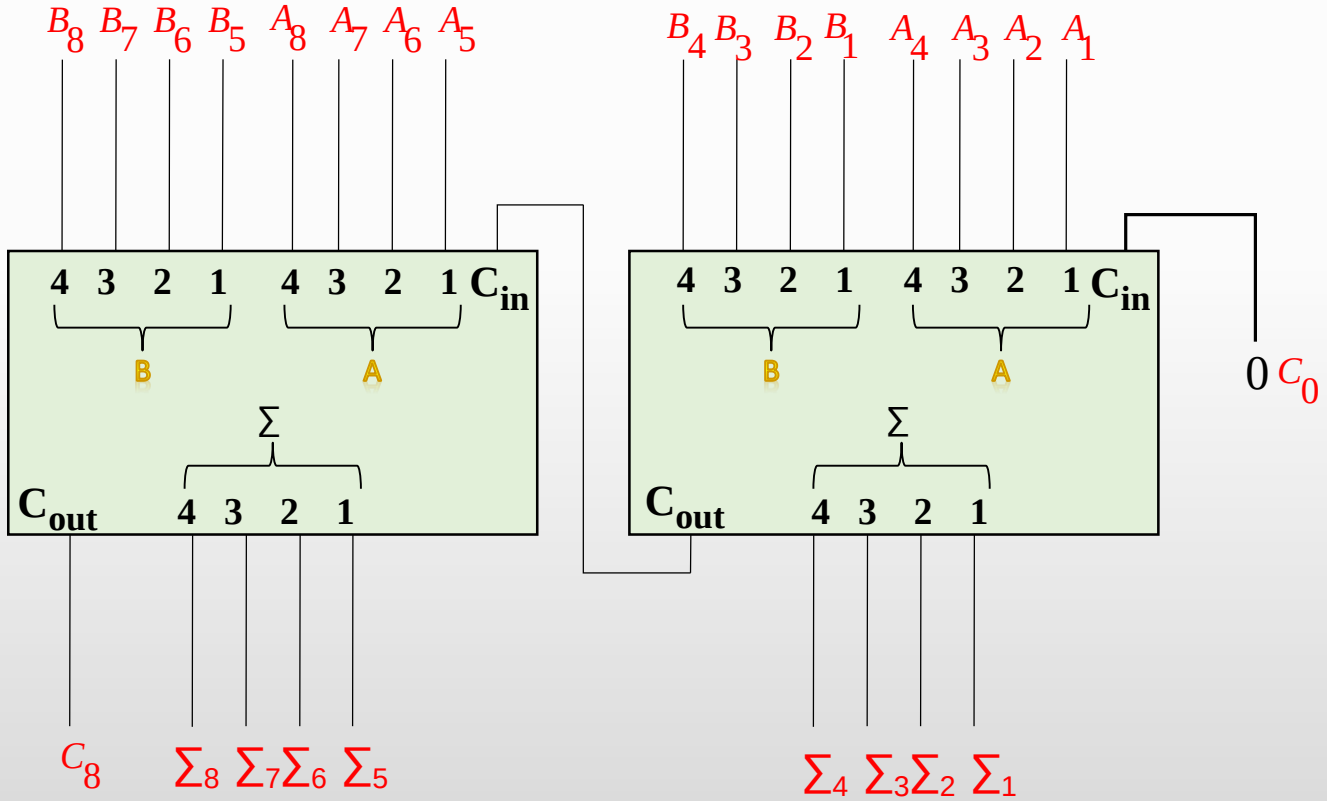
(a) Pin diagram of 74LS283

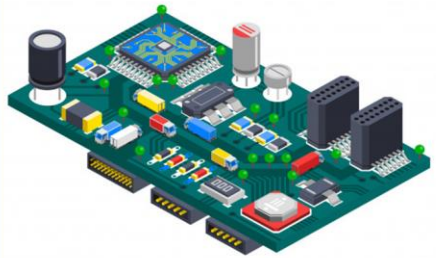
(b) 74LS283 logic symbol



6.2 Mạch cộng nhị phân song song

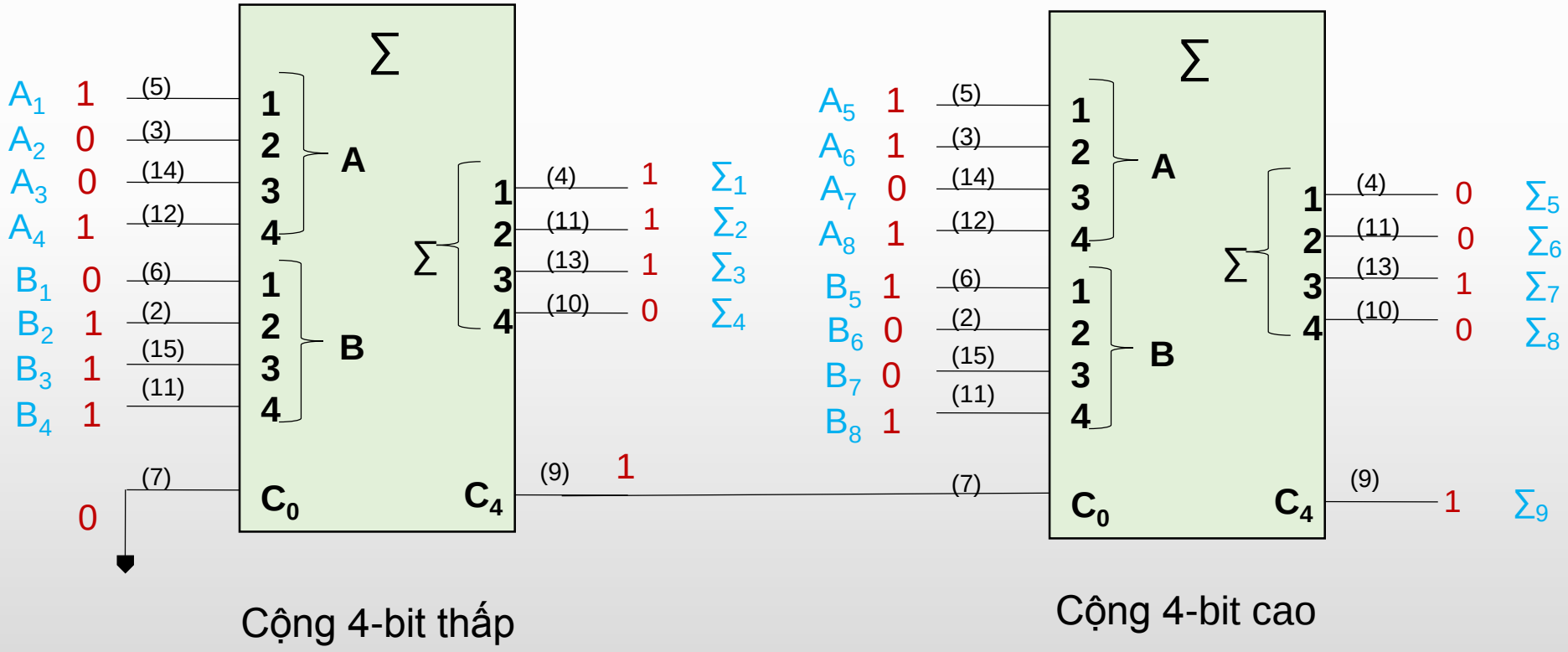
- Mở rộng thành mạch cộng 8-bit song song



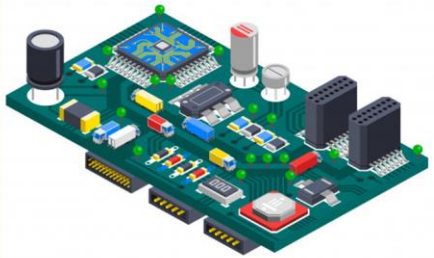


6.2 Mạch cộng nhị phân song song

- Ghép 2 IC 74LS283 để có mạch cộng 8-bit

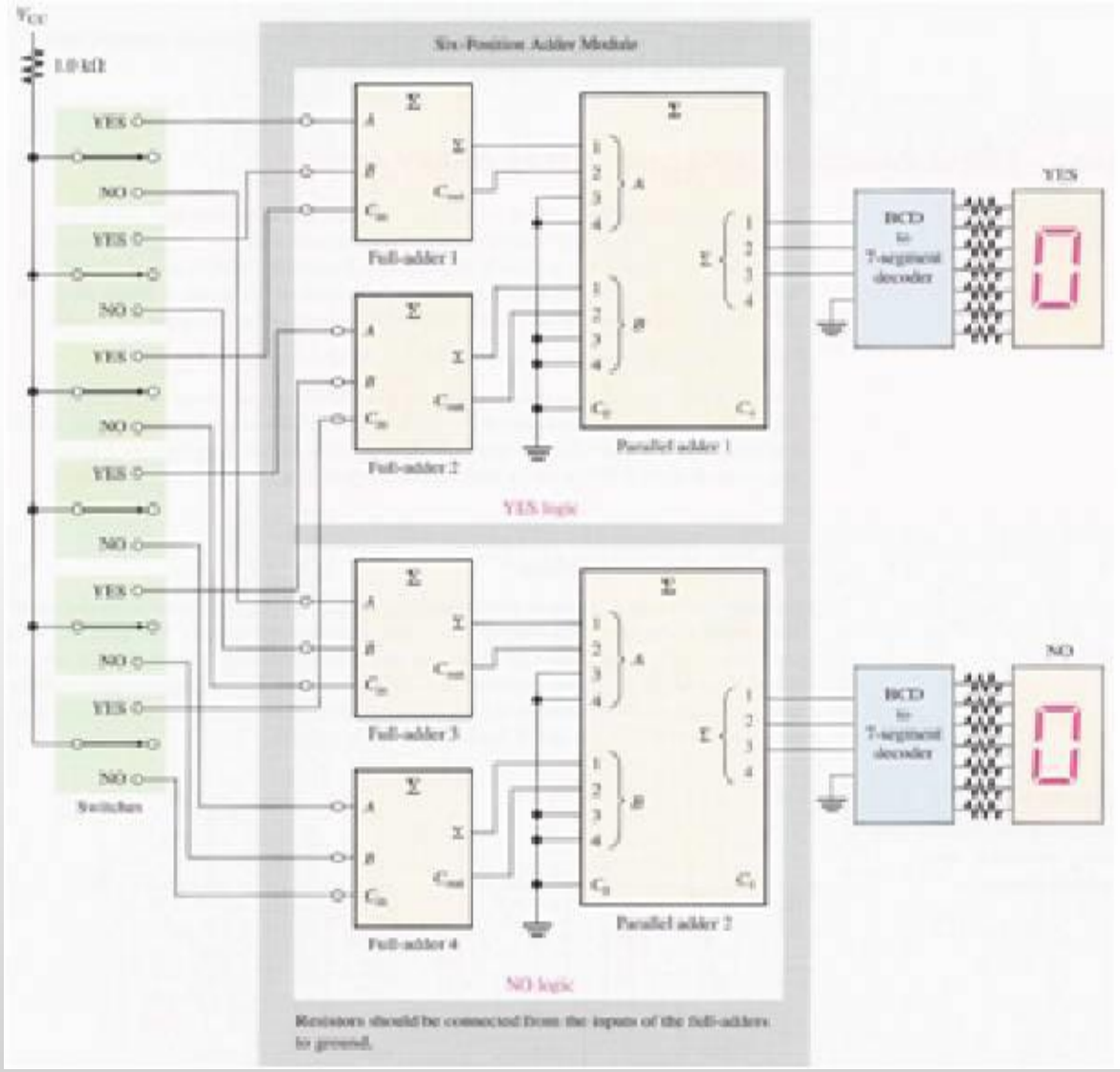


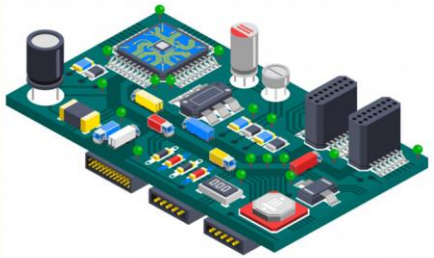
PHENIKAA UNIVERSITY



6.2 Mạch cộng nhị phân song song

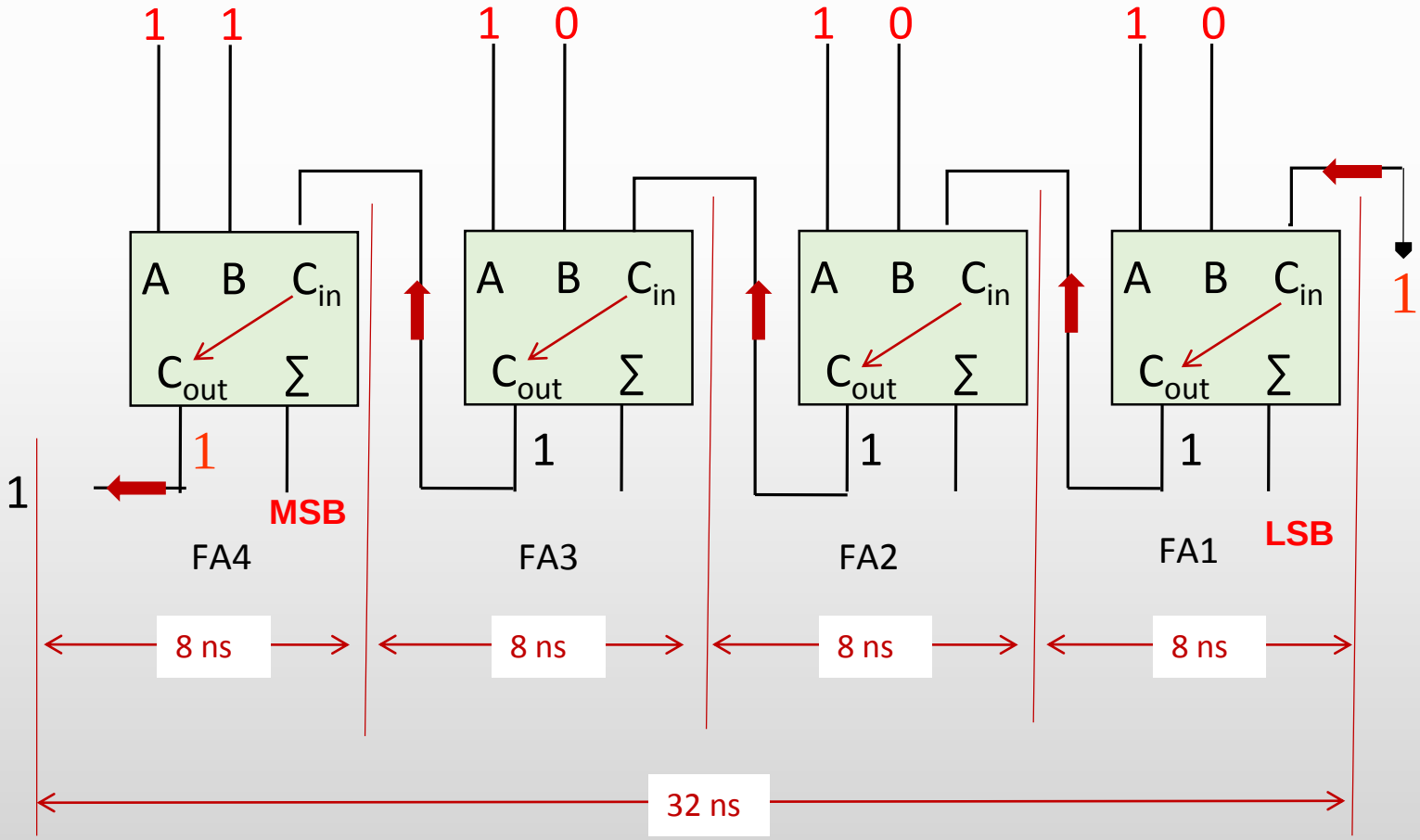
Ứng dụng

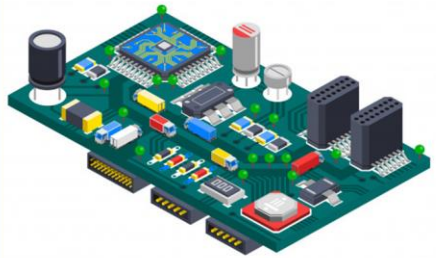




6.3 Truyền và thấy trước số nhớ

- Mạch cộng truyền số nhớ (ripple carry adder)





6.3 Truyền và thấy trước số nhớ

- Mạch cộng truyền số nhớ (ripple carry adder)

$C_{out} = A.B + (A + B).C_{in}$ Đặt $C_g = A.B$ và $C_p = A + B$

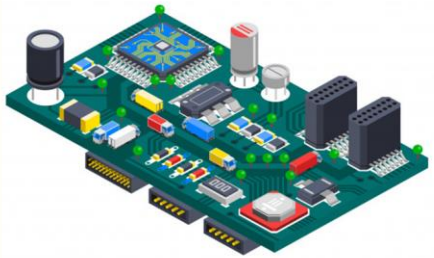
Ta có: $C_{out} = C_g + C_p.C_{in}$

A	B	C_{in}	C_{out}	S	carry status
0	0	0	0	0	diệt
0	0	1	0	1	diệt
0	1	0	0	1	truyền
0	1	1	1	0	truyền
1	0	0	0	1	truyền
1	0	1	1	0	truyền
1	1	0	1	0	tạo
1	1	1	1	1	tạo

→ C_p

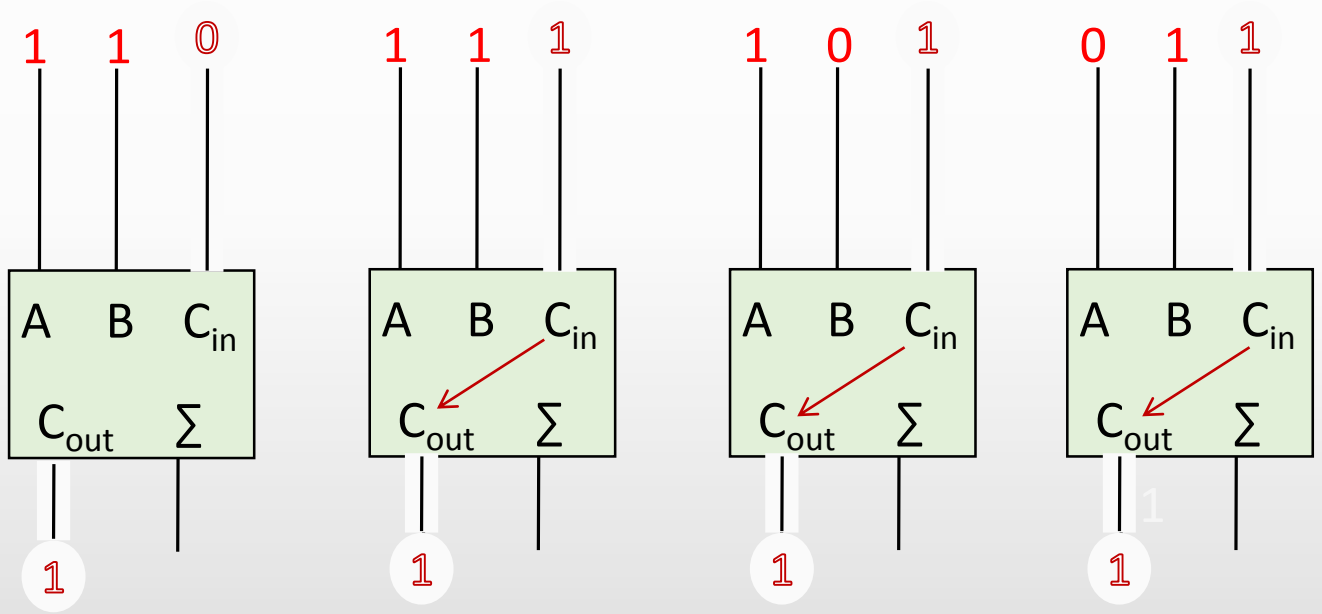
→ C_g

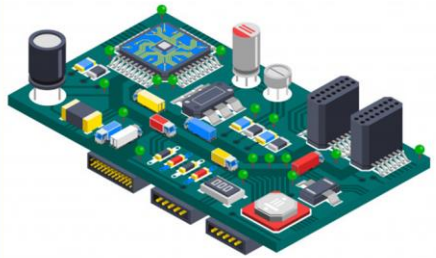
PHENIKAA UNIVERSITY



6.3 Truyền và thấy trước số nhớ

- Mạch cộng thấy trước số nhớ (look-ahead adder)





6.3 Truyền và thấy trước số nhớ

- Mạch cộng thấy trước số nhớ

$$C_{out} = C_g + C_p C_{in}$$

$$C_{out1} = C_{g1} + C_{p1} C_{in1}$$

$$C_{in2} = C_{out1}$$

$$C_{out2} = C_{g2} + C_{p2} C_{in2} = C_{g2} + C_{p2} C_{out1} = C_{g2} + C_{p2}(C_{g1} + C_{p1} C_{in1})$$

$$= C_{g2} + C_{p2} C_{g1} + C_{p2} C_{p1} C_{in1} \longrightarrow$$ chỉ phụ thuộc c_{in1}

$$C_{in3} = C_{out2}$$

$$C_{out3} = C_{g3} + C_{p3} C_{in3} = C_{g3} + C_{p3} C_{out2} = C_{g3} + C_{p3}(C_{g2} + C_{p2} C_{g1} + C_{p2} C_{p1} C_{in1})$$

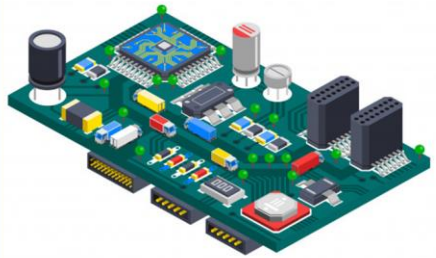
$$= C_{g3} + C_{p3} C_{g2} + C_{p3} C_{p2} C_{g1} + C_{p3} C_{p2} C_{p1} C_{in1} \longrightarrow$$ chỉ phụ thuộc c_{in1}

$$C_{in4} = C_{out3}$$

$$C_{out4} = C_{g4} + C_{p4} C_{in4} = C_{g4} + C_{p4} C_{out3}$$

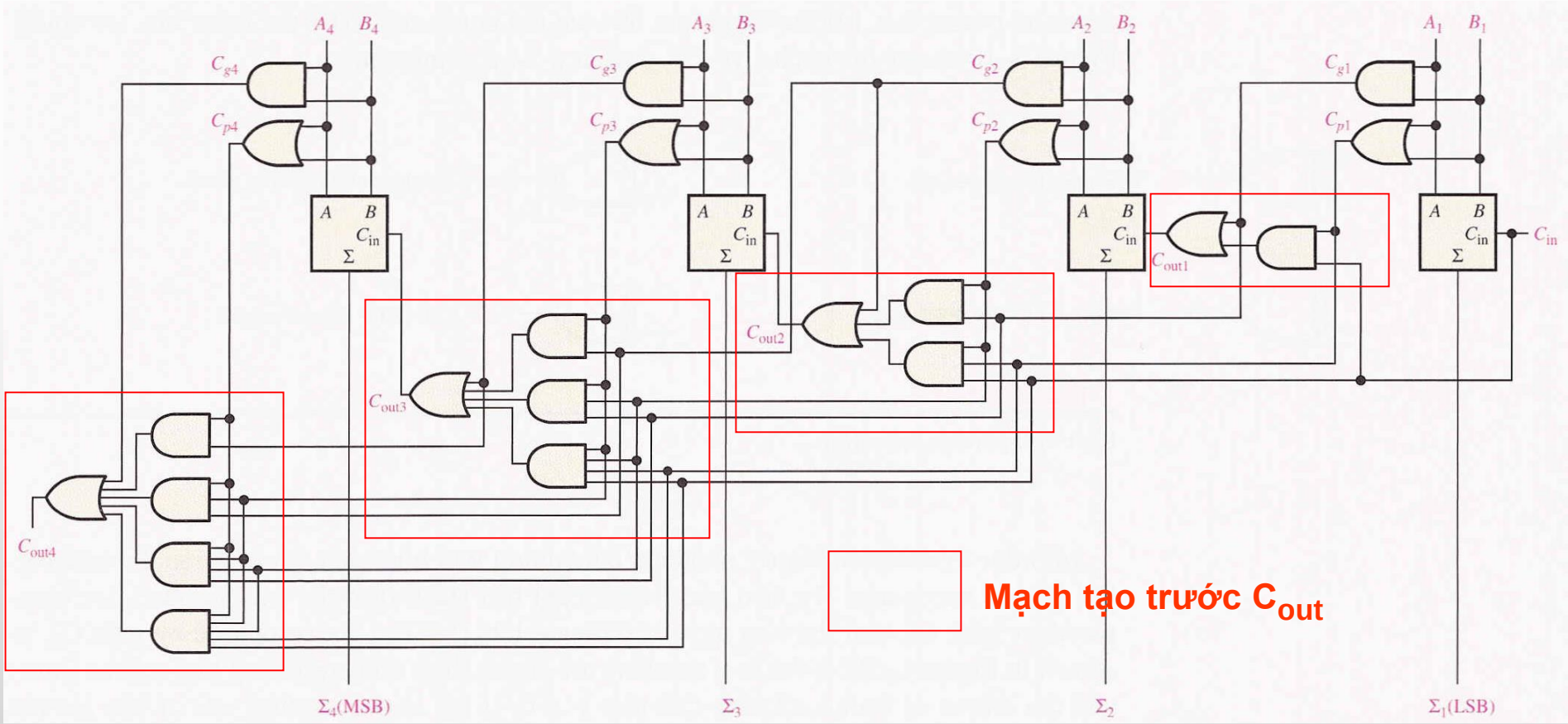
$$= C_{g4} + C_{p4}(C_{g3} + C_{p3} C_{g2} + C_{p3} C_{p2} C_{g1} + C_{p3} C_{p2} C_{p1} C_{in1})$$

$$= C_{g4} + C_{p4} C_{g3} + C_{p4} C_{p3} C_{g2} + C_{p4} C_{p3} C_{p2} C_{g1} + C_{p4} C_{p3} C_{p2} C_{p1} C_{in1} \longrightarrow$$ chỉ phụ thuộc c_{in1}

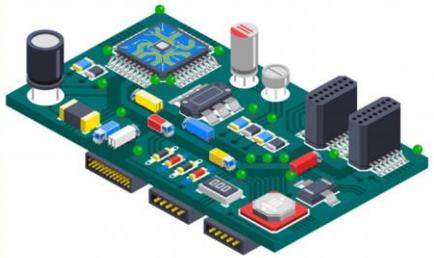


6.3 Truyền và thấy trước số nhớ

- Mạch cộng thấy trước số nhớ

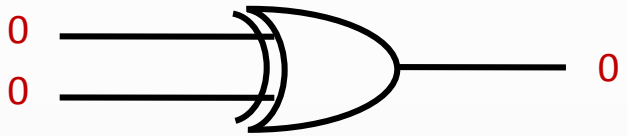


PHENIKAA UNIVERSITY

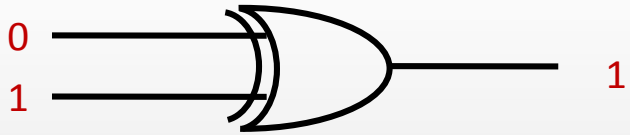


6.4 Mạch so sánh

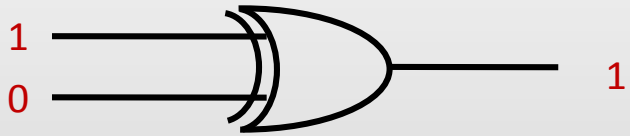
- So sánh hai bit



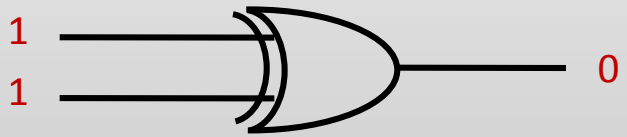
Các ngõ vào bằng nhau



Các ngõ vào không bằng nhau

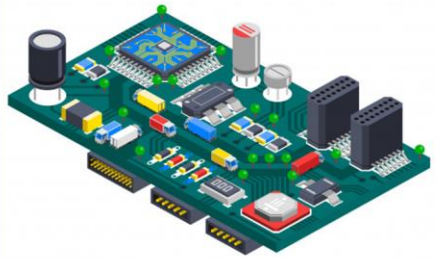


Các ngõ vào không bằng nhau



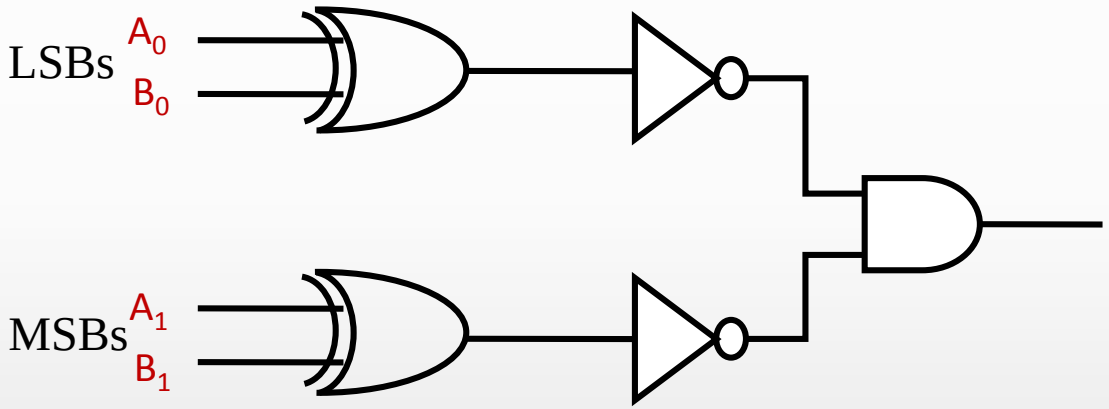
Các ngõ vào bằng nhau

PHENIKAA UNIVERSITY



6.4 Mạch so sánh

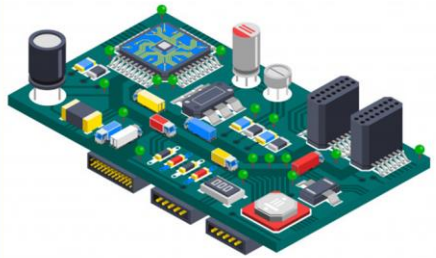
- So sánh bằng



Ngõ ra (A = B) ở logic 1 cho biết hai số A và B bằng nhau.

Dạng tổng quát: Số nhị phân A → A_1 A_0

Số nhị phân B → B_1 B_0

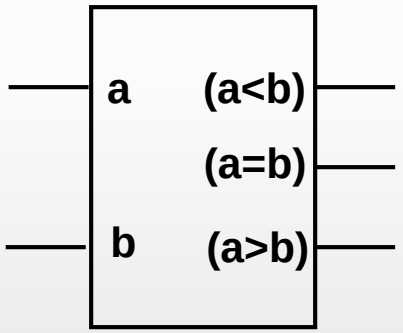


6.4 Mạch so sánh

- So sánh không bằng

  So sánh 1-bit

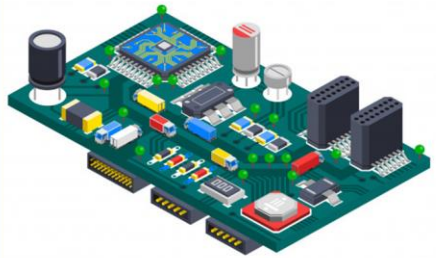
a	b	a<b	a=b	a>b
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



$a<b = \overline{a}.b$ Ngõ ra a<b bằng 1 cho biết a nhỏ hơn b.

$a=b = \overline{a \oplus b}$ Ngõ ra a=b bằng 1 cho biết a bằng b.

$a>b = a.\overline{b}$ Ngõ ra a>b bằng 1 cho biết a lớn hơn b.



6.4 Mạch so sánh

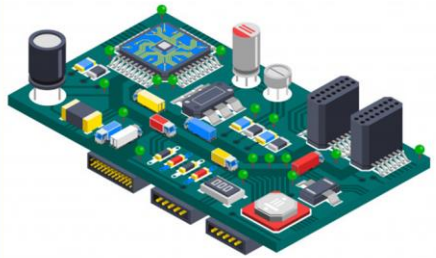
- So sánh không bằng

  So sánh 4-bit

  So sánh 2 số: $A = A_3A_2A_1A_0$ và $B = B_3B_2B_1B_0$

A_3B_3	A_2B_2	A_1B_1	A_0B_0	Kết quả
>	x	x	x	A>B
<	x	x	x	A<B
=	>	x	x	A>B
=	<	x	x	A<B
=	=	>	x	A>B
=	=	<	x	A<B
=	=	=	>	A>B
=	=	=	<	A<B
=	=	=	=	A=B

Kết quả so sánh A_3 và B_3 sẽ điều khiển việc so sánh A_2 với B_2, kết quả so sánh A_2 và B_2 sẽ điều khiển việc so sánh A_1 với B_1, v.v. . .



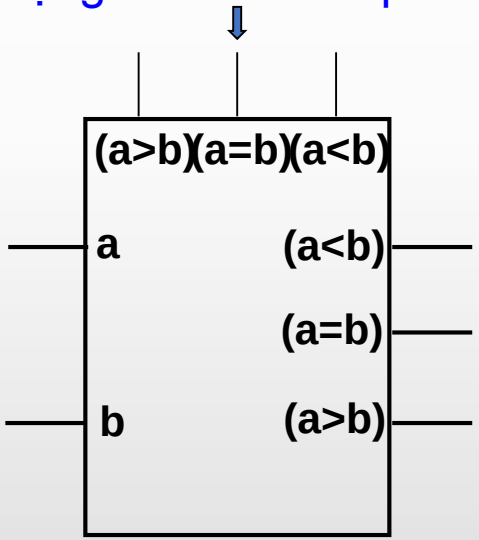
6.4 Mạch so sánh

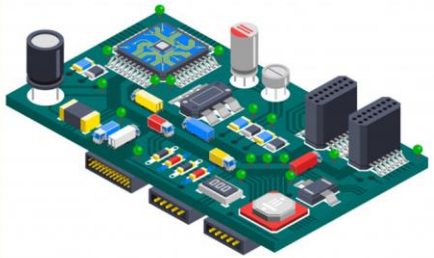
- So sánh không bằng

  So sánh 1-bit có điều khiển

Ngõ vào điều khiển					Ngõ ra so sánh		
a>b	a=b	a<b	a	b	a>b	a=b	a<b
1	0	0	x	x	1	0	0
0	0	1	x	x	0	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	0	0	1
0	1	0	1	0	1	0	0
0	1	0	1	1	0	1	0

Kết quả so sánh 2 bit có trọng số lớn kế tiếp.





6.4 Mạch so sánh

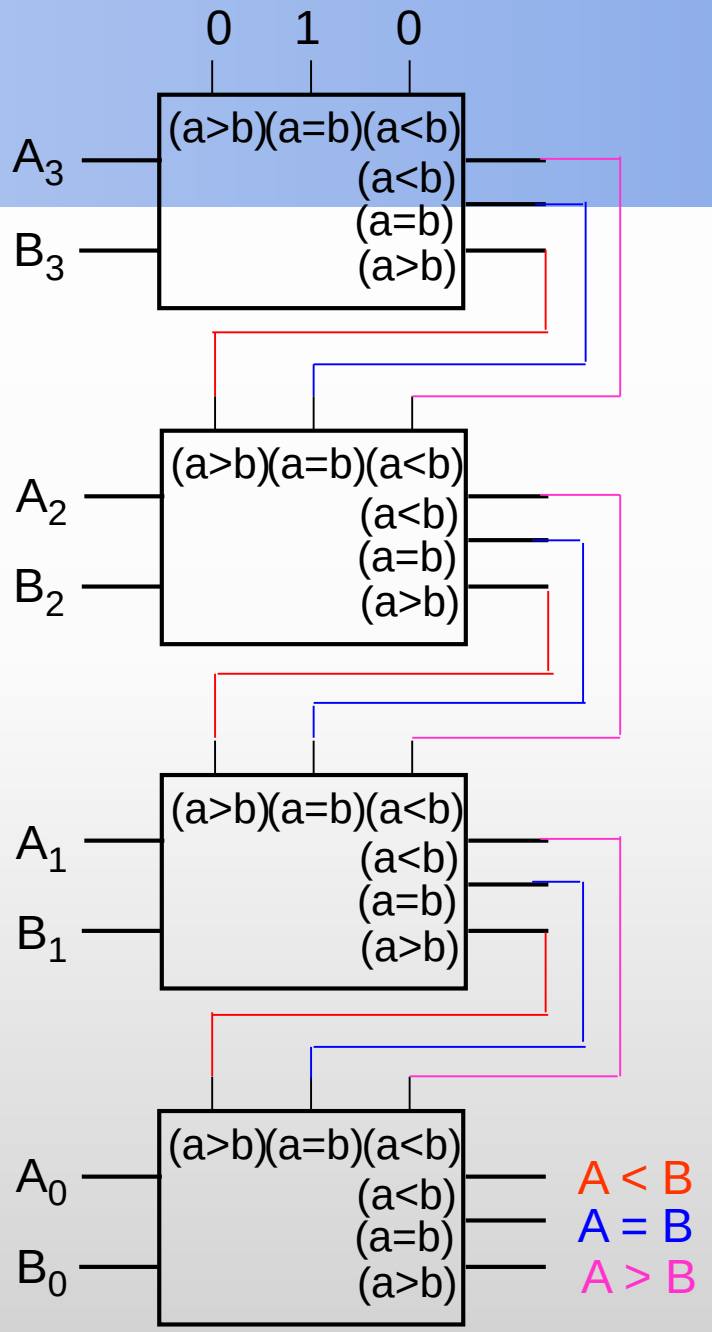
- So sánh không bằng

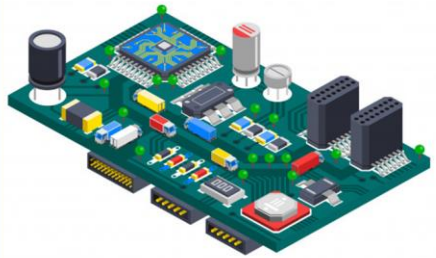
  So sánh 4-bit

  $A = A_3A_2A_1A_0$

  $B = B_3B_2B_1B_0$

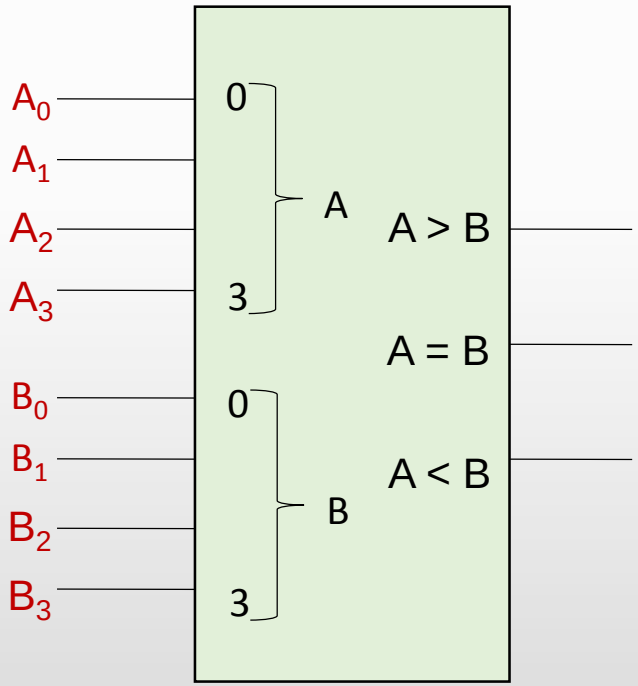
Ta ghép 4 mạch so sánh 1-bit có điều khiển để tạo thành mạch so sánh 4-bit.

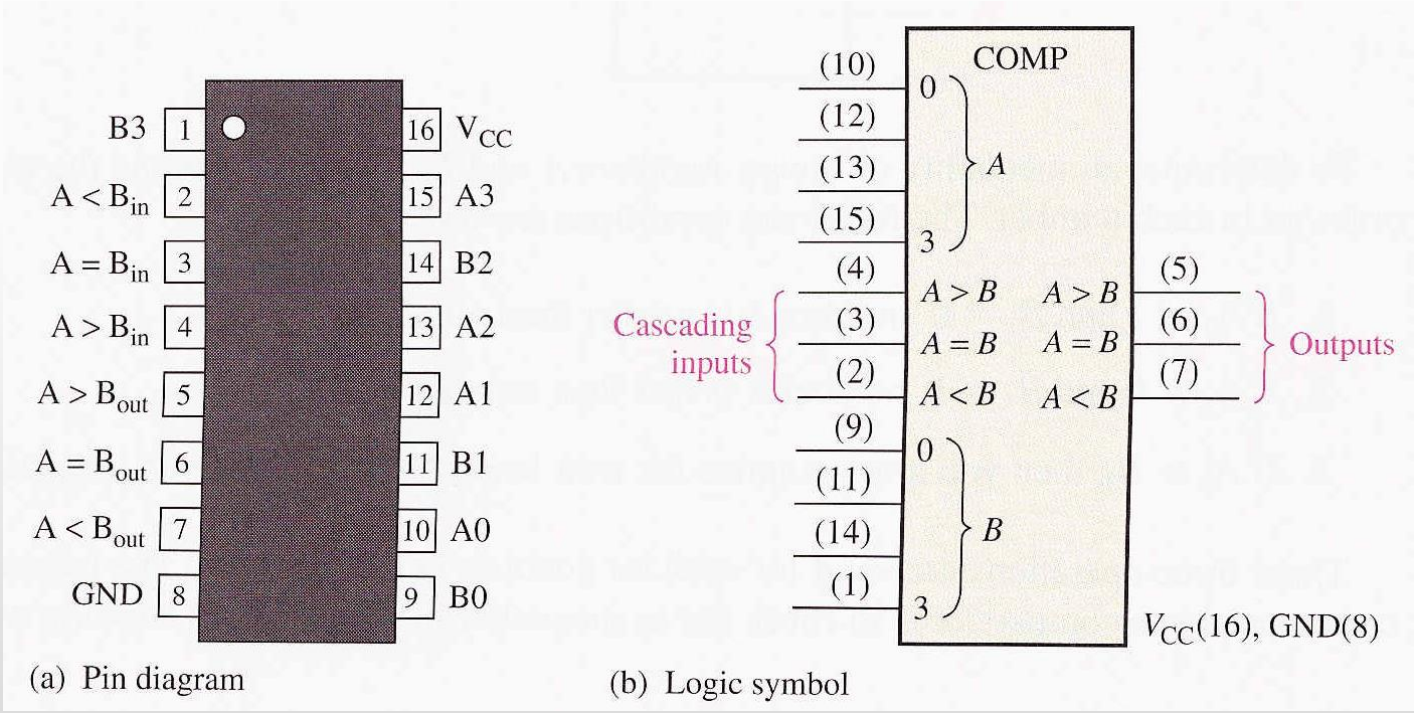


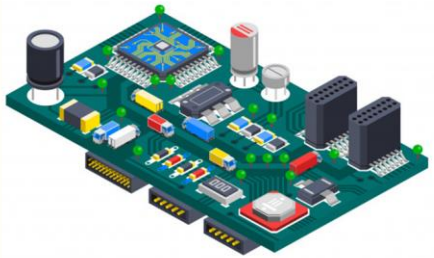


6.4 Mạch so sánh

- IC so sánh 4-bit 74HC85

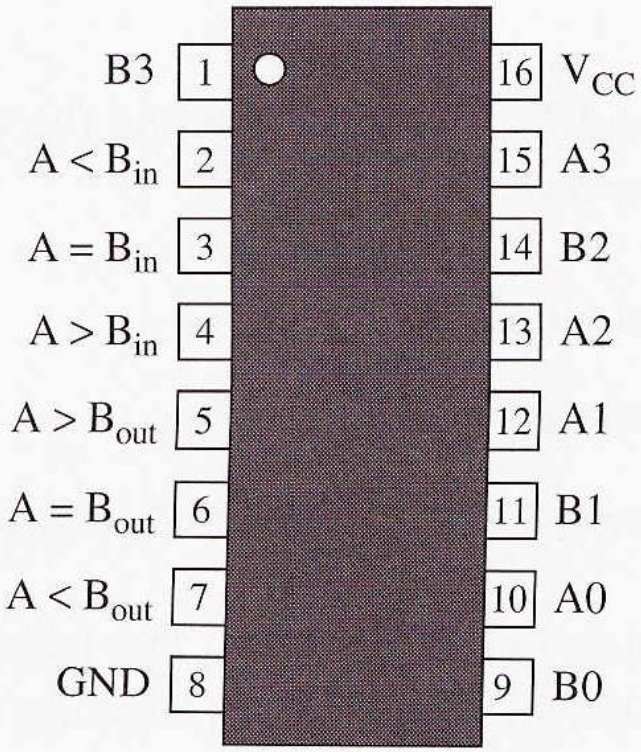




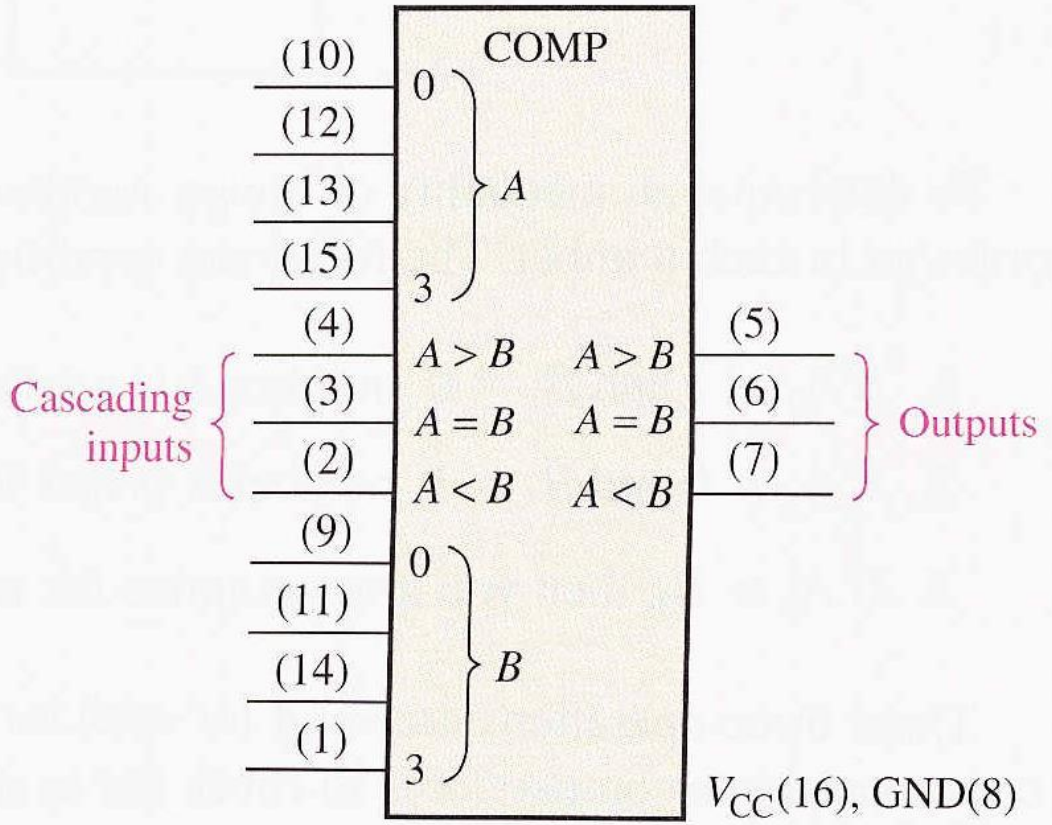


6.4 Mạch so sánh

- IC so sánh 4-bit 74HC85



(a) Pin diagram



(b) Logic symbol