




Kiến trúc máy tính

- ThS. Phạm Văn Phước
- <https://sites.google.com/site/phuocphamuit/ktmt>
- phuocpv@uit.edu.vn

Một số quy tắc




Đánh Giá:

- Giữa kỳ: 30%
- Cuối kỳ: 70%

Quy tắc:

- Vào lớp trật tự.
- Không mở điện thoại
- Điềm danh khi lớp vắng
- Cấm thi nếu vắng quá 4 buổi học.

Nội Dung

- 
- Chương 0 - Tổng quan nhập môn mạch số
 - Chương 1 - Máy tính - các khái niệm và công nghệ
 - Chương 2 – Assembly MIPS
 - Chương 3 - Phép toán số học trên máy tính
 - Chương 4 - Đường dữ liệu


KIẾN TRÚC MÁY TÍNH

Chương 0



Tổng Quan Nhập Môn Mạch Số

Nội Dung

- 
1. Giới thiệu các hệ thống số
 2. Chuyển đổi giữa các hệ thống số
 3. Các cổng Logics cơ bản
 4. Mạch Logic
 5. Mạch tích hợp



1. Giới thiệu các hệ thống số

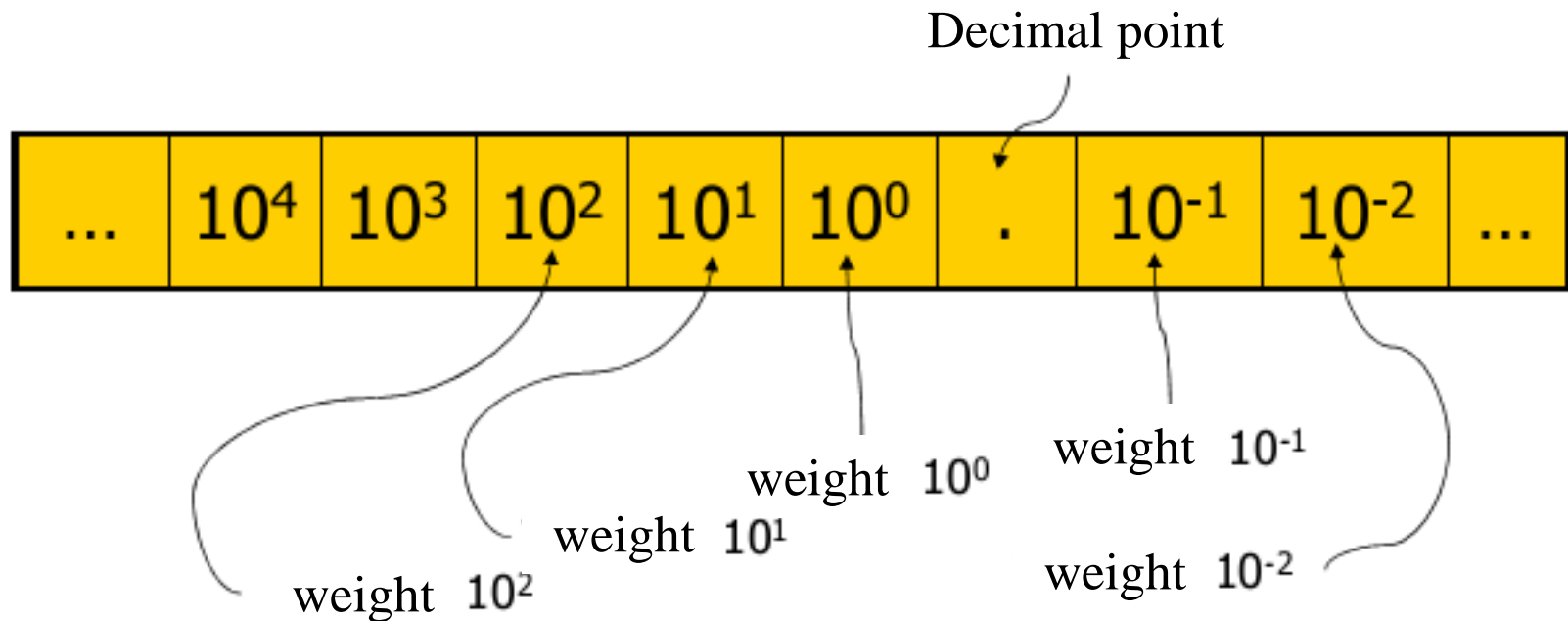
- **Số Thập Phân**
- **Số Nhị Phân**
- **Số Thập Lục Phân**
- **Số Bát Phân**

Các Hệ thống số cơ bản

| Hệ thống số | Cơ số | Chữ số |
|-------------|-------|--|
| Thập Phân | 10 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| Nhị Phân | 2 | 0, 1 |
| Bát Phân | 8 | 0, 1, 2, 3, 4, 5, 6, 7 |
| Thập Lục | 16 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D, E, F |

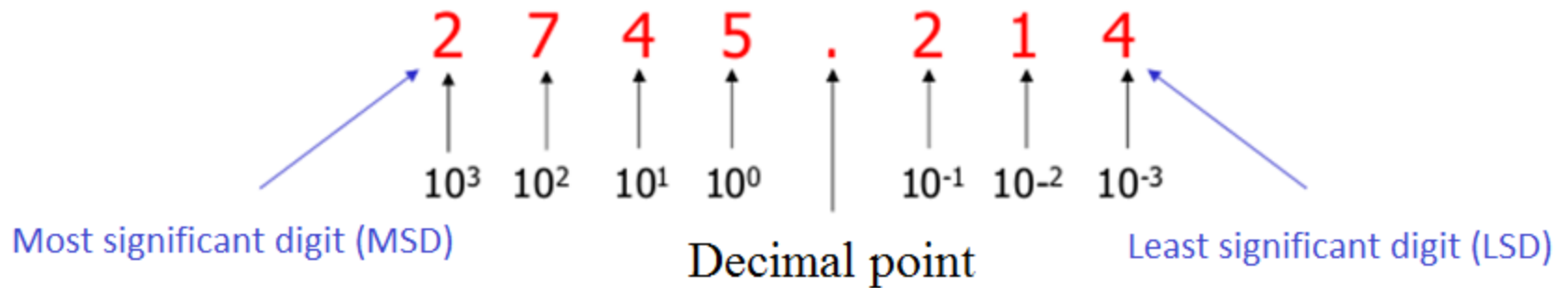
Số Thập Phân

Ví dụ: 2745.214₁₀



Số Thập Phân

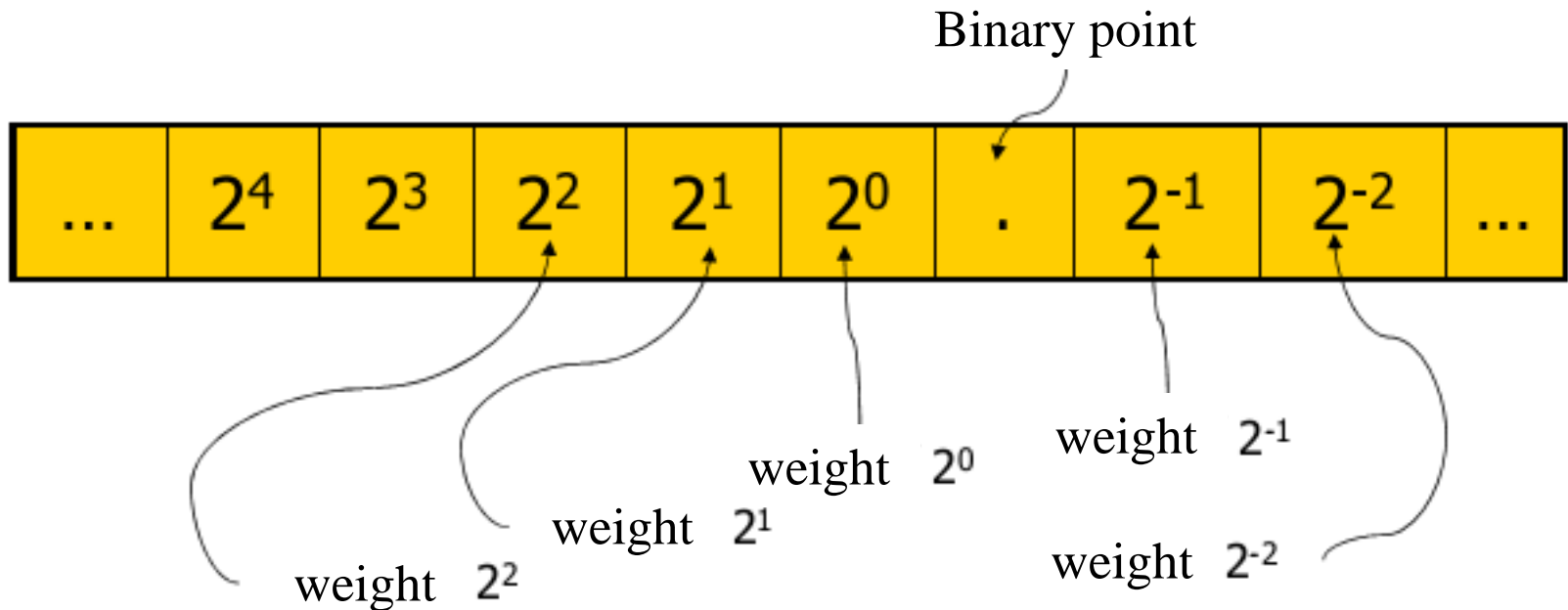
- Phân tích số thập phân : 2745.214_{10}



- $2745.214_{10} =$
 $2 * 10^3 + 7 * 10^2 + 4 * 10^1 + 5 * 10^0 +$
 $2 * 10^{-1} + 1 * 10^{-2} + 4 * 10^{-3}$

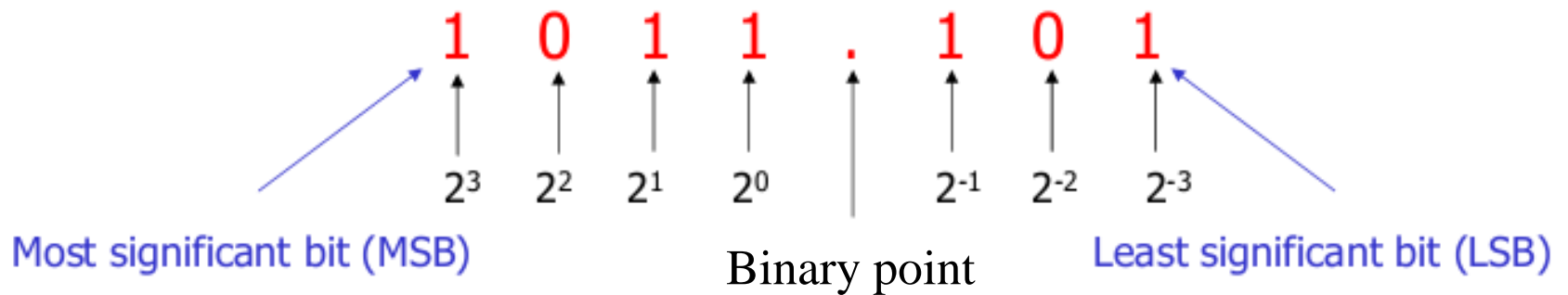
Số Nhị Phân

Ví dụ: 1011.101_2




Số Nhị Phân

- Phân tích số nhị phân 1011.101_2



- $$\begin{aligned} 1011.101_2 &= 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 + \\ &\quad 1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} \\ &= 11.625_{10} \end{aligned}$$


Số Bát Phân



| | | | | | | | | | |
|-----|-------|-------|-------|-------|-------|---|----------|----------|-----|
| ... | 8^4 | 8^3 | 8^2 | 8^1 | 8^0 | . | 8^{-1} | 8^{-2} | ... |
|-----|-------|-------|-------|-------|-------|---|----------|----------|-----|

- Số Bát Phân : 372_8
- $372_8 = 3 * 8^2 + 7 * 8^1 + 2 * 8^0$
 $= 250_{10}$

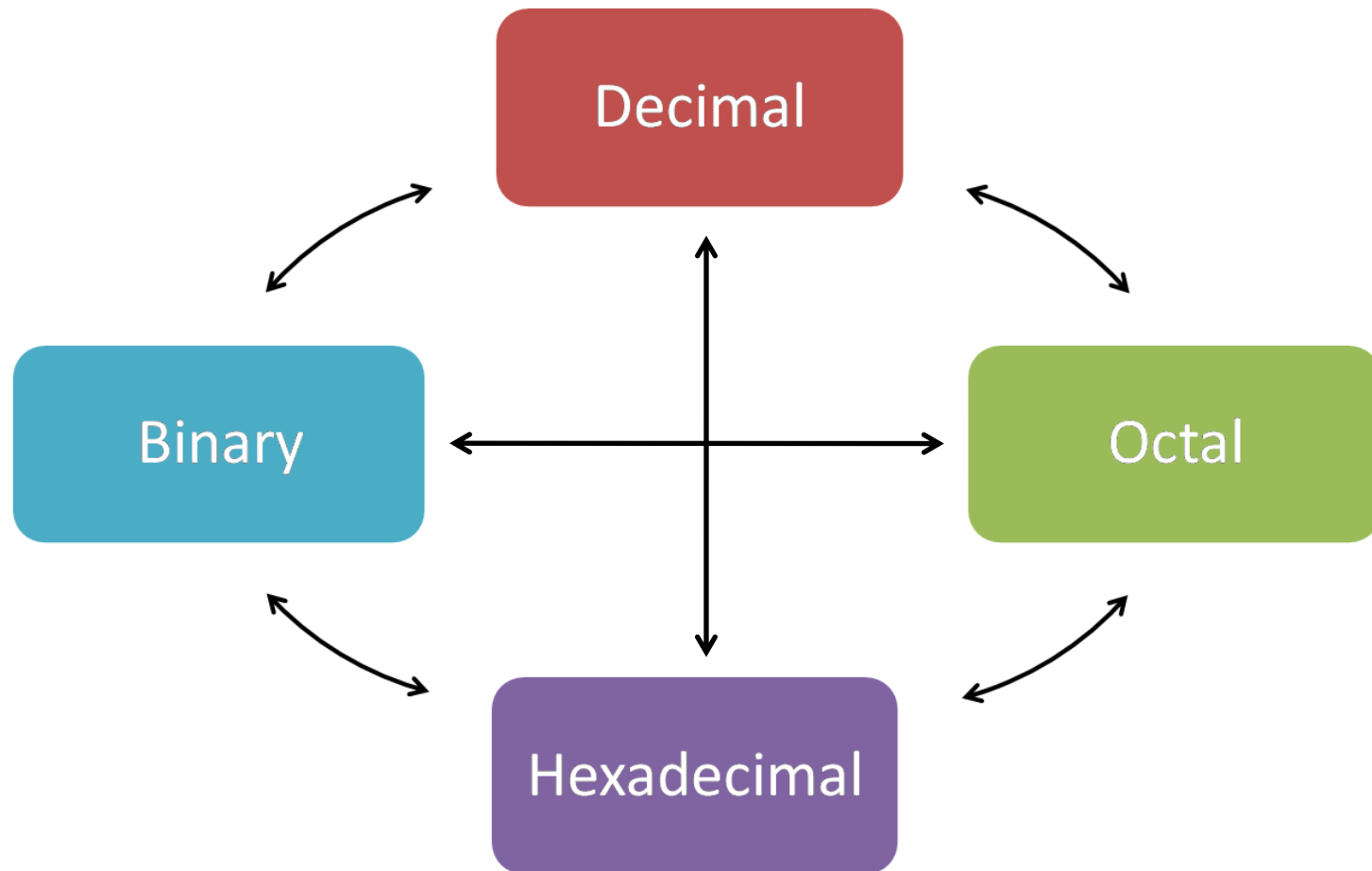
Số Thập Lục Phân



| | | | | | | | | | |
|-----|--------|--------|--------|--------|--------|---|-----------|-----------|-----|
| ... | 16^4 | 16^3 | 16^2 | 16^1 | 16^0 | . | 16^{-1} | 16^{-2} | ... |
|-----|--------|--------|--------|--------|--------|---|-----------|-----------|-----|

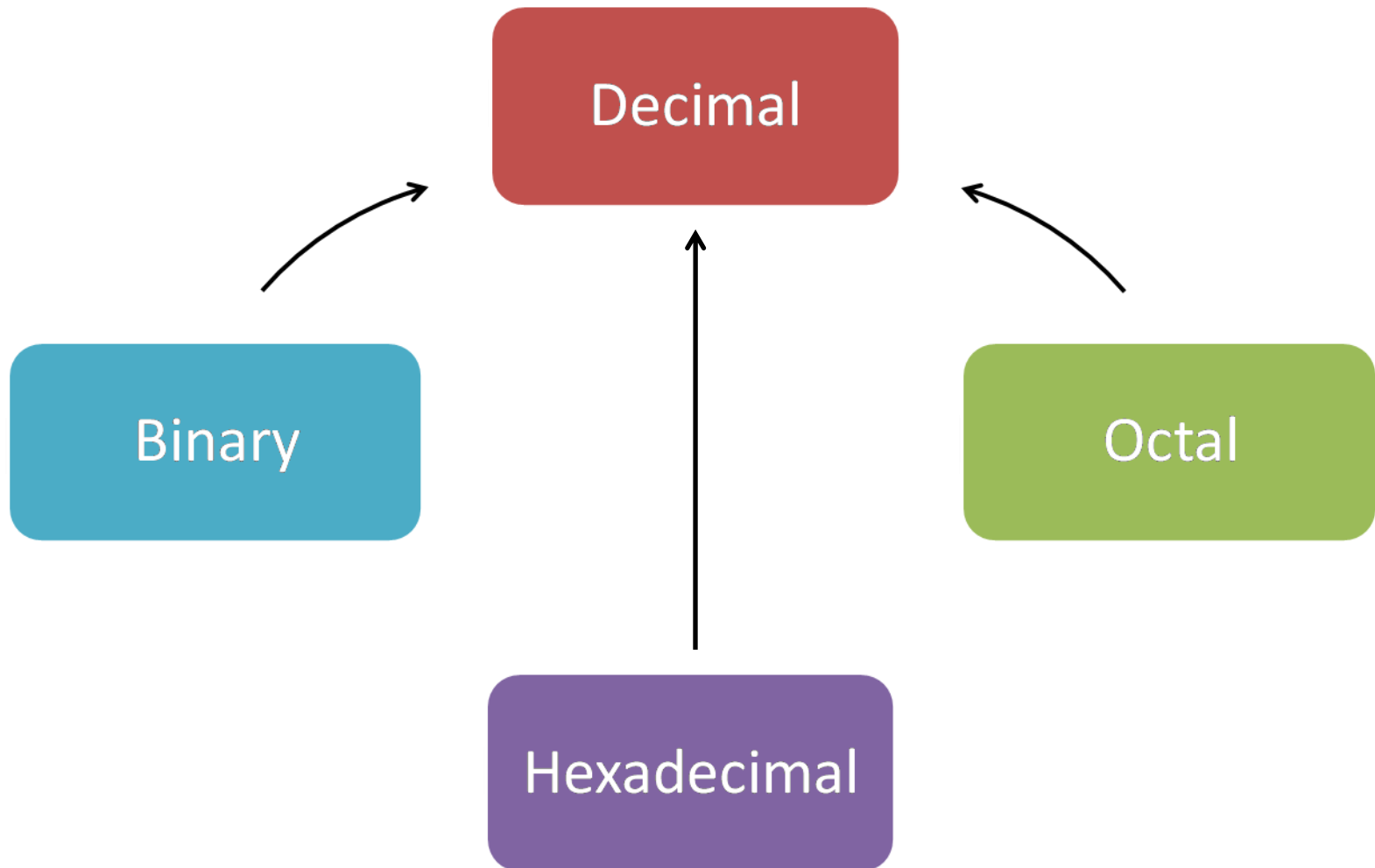
- Phân tích số thập lục phân : **3BA**₁₆
- $3BA_{16} = 3 * 16^2 + 11 * 16^1 + 10 * 16^0$
 $= 954_{10}$

2. Chuyển đổi giữa các hệ thống số



Chuyển đổi sang số thập phân

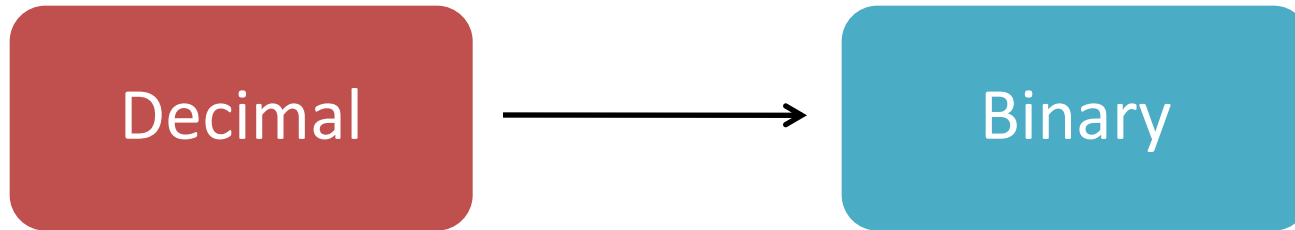
- Nhân mỗi chữ số (digit) với trọng số (weight)



Ví Dụ

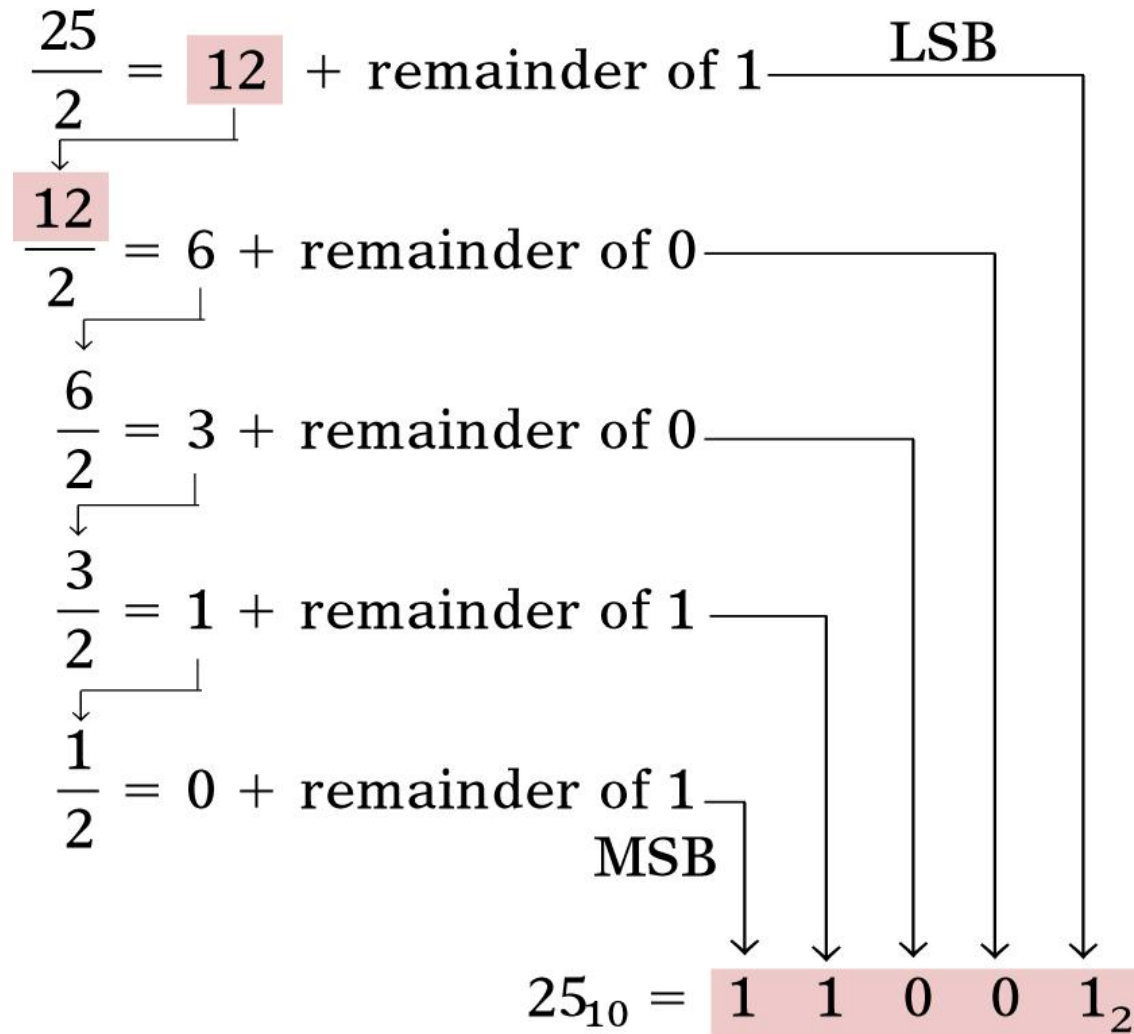
- Biểu diễn 3702_8 sang số thập phân
- Biểu diễn $1A2F_{16}$ sang số thập phân

Số Thập Phân => Số Nhị Phân

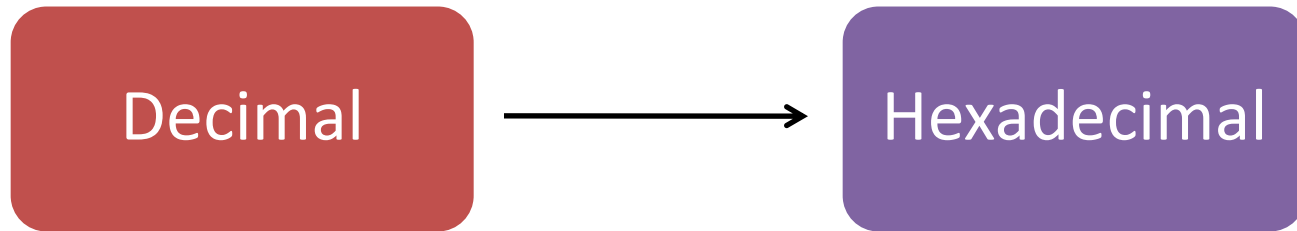


- **Chia số thập phân với 2 và sau đó viết ra phần dư còn lại**
 - Chia cho đến khi có thương số là 0.
- **Phần số dư đầu tiên gọi là LSB** (Bit có trọng số thấp nhất)
- **Phần số dư cuối cùng gọi là MSB** (Bit có trọng số cao nhất)

Ví dụ : 25₁₀ => Số Nhị Phân



Số Thập Phân => Số Thập Lục Phân

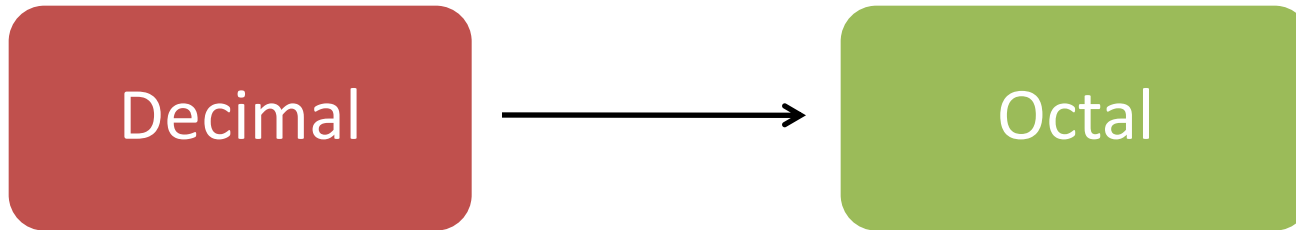


- **Chia số thập phân cho 16 và viết ra phần dư còn lại**
 - Chia cho đến khi có thương số là 0.
- **Phần số dư đầu tiên gọi là LSD (Số có trọng số thấp nhất)**
- **Phần số dư cuối cùng gọi là MSD (Số có trọng số cao nhất)**

Ví Dụ: $423_{10} \Rightarrow$ Thập Lục Phân

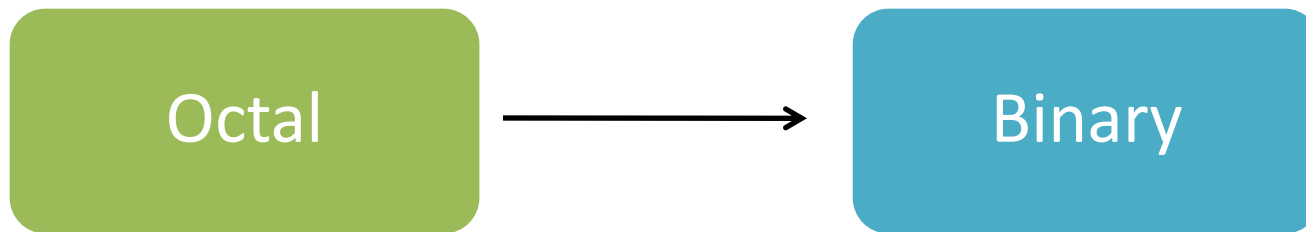
$$\begin{array}{l} \frac{423}{16} = 26 + \text{remainder of } 7 \\ \downarrow \\ \frac{26}{16} = 1 + \text{remainder of } 10 \\ \downarrow \\ \frac{1}{16} = 0 + \text{remainder of } 1 \end{array}$$
$$423_{10} = 1A7_{16}$$

Thập Phân => Bát Phân



- **Chia số thập phân cho 8 và viết ra phần dư còn lại**
 - Chia cho đến khi có thương số là 0.
- **Phần số dư đầu tiên gọi là LSD** (Số có trọng số thấp nhất)
- **Phần số dư cuối cùng gọi là MSD** (Số có trọng số lớn nhất)

Bát Phân => Nhị Phân

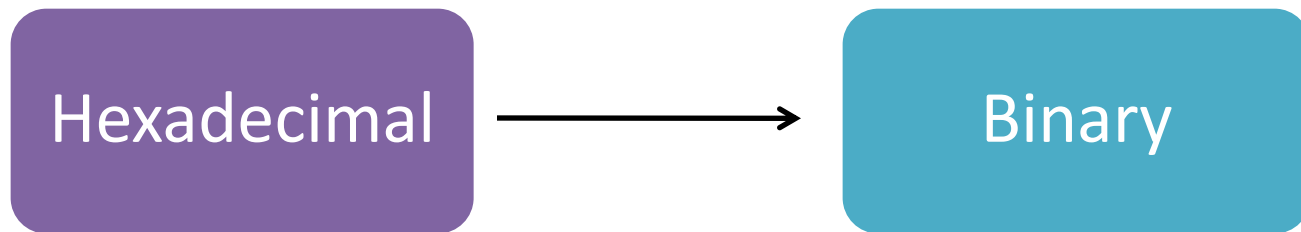


- Chuyển đổi lần lượt mỗi chữ số ở dạng Bát Phân sang nhóm 3 bits Nhị Phân

| Octal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|
| Binary | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

- VD:

Thập Lục Phân => Nhị Phân



- Chuyển đổi lần lượt mỗi chữ số ở dạng Thập Lục Phân sang nhóm 4 bits Nhị Phân

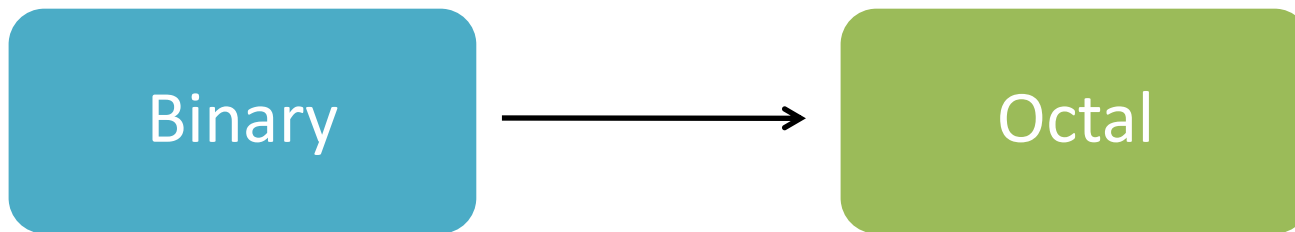
- VD:

5 6 A E 6 A

101 0110 1010 1110 0110 1010

| Hex | Bin |
|-----|------|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

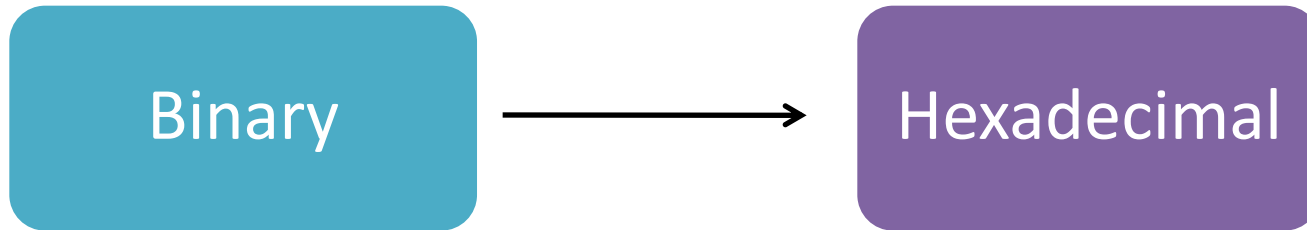
Nhị Phân \Rightarrow Bát Phân



- **Nhóm 3 bits bắt đầu từ ngoài cùng bên phải của số**
- **Chuyển đổi mỗi nhóm trên sang dạng chữ số của Bát Phân**
- VD: $1011010111_2 \Rightarrow$ Bát Phân

1327_8

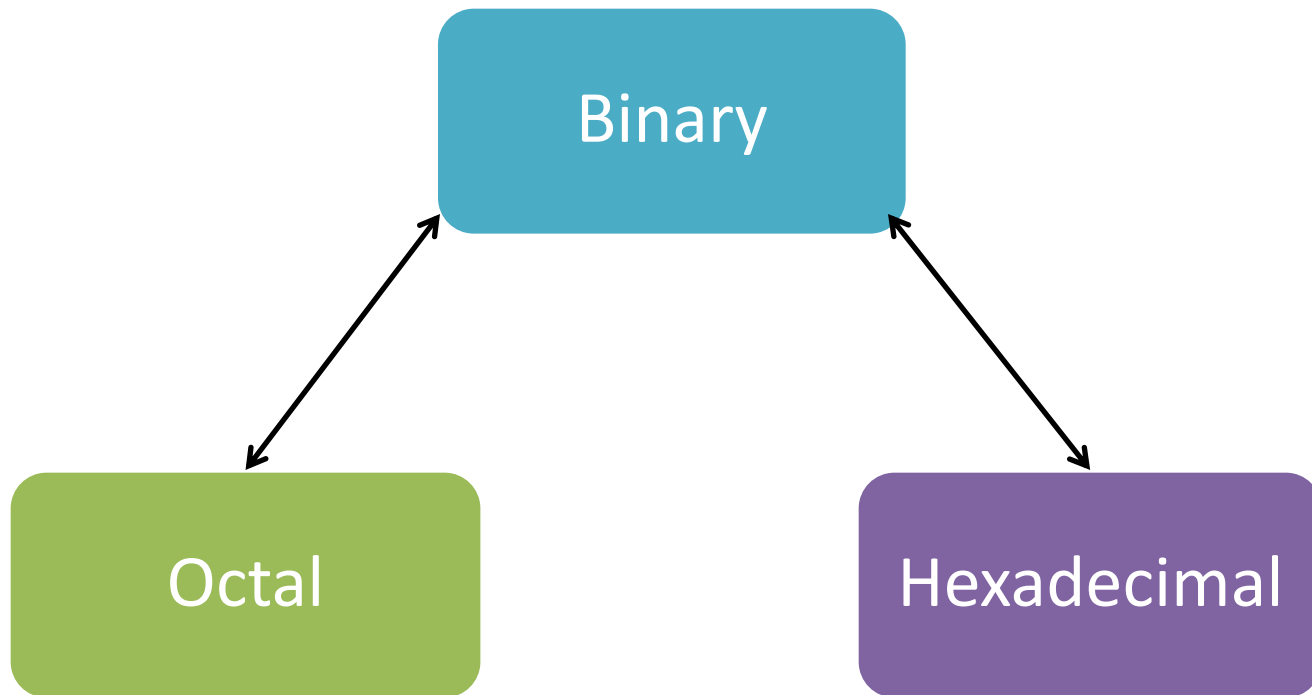
Nhị Phân => Thập Lục Phân



- **Nhóm 4 bits từ phía ngoài cùng bên phải của số**
- **Chuyển đổi mỗi nhóm trên sang 1 chữ số Thập Lục**
- **VD: $10101101010111001101010_2 \Rightarrow$ Thập Lục Phân**

$56AE6A_{16}$

Bát Phân <=> Thập Lục Phân



- Chuyển đổi thông qua trung gian là số Nhị Phân

Ví dụ: $1F0C_{16} \Rightarrow$ Bát Phân


Chuyển đổi từ Thập Lục Phân sang Nhị Phân

$$1F0C_{16} = 1_1111_0000_1100_2$$

Chuyển đổi từ Nhị Phân sang Bát Phân

$$1_111_100_001_100_2 = 17414_8$$

Ví Dụ: $1076_8 \Rightarrow$ Thập Lục phân



Chuyển đổi từ Bát Phân sang Nhị Phân

$$1076_8 = 1_000_111_110_2$$

Chuyển đổi từ Nhị Phân sang Thập Lục Phân

$$10_0011_1110_2 = 23E_{16}$$

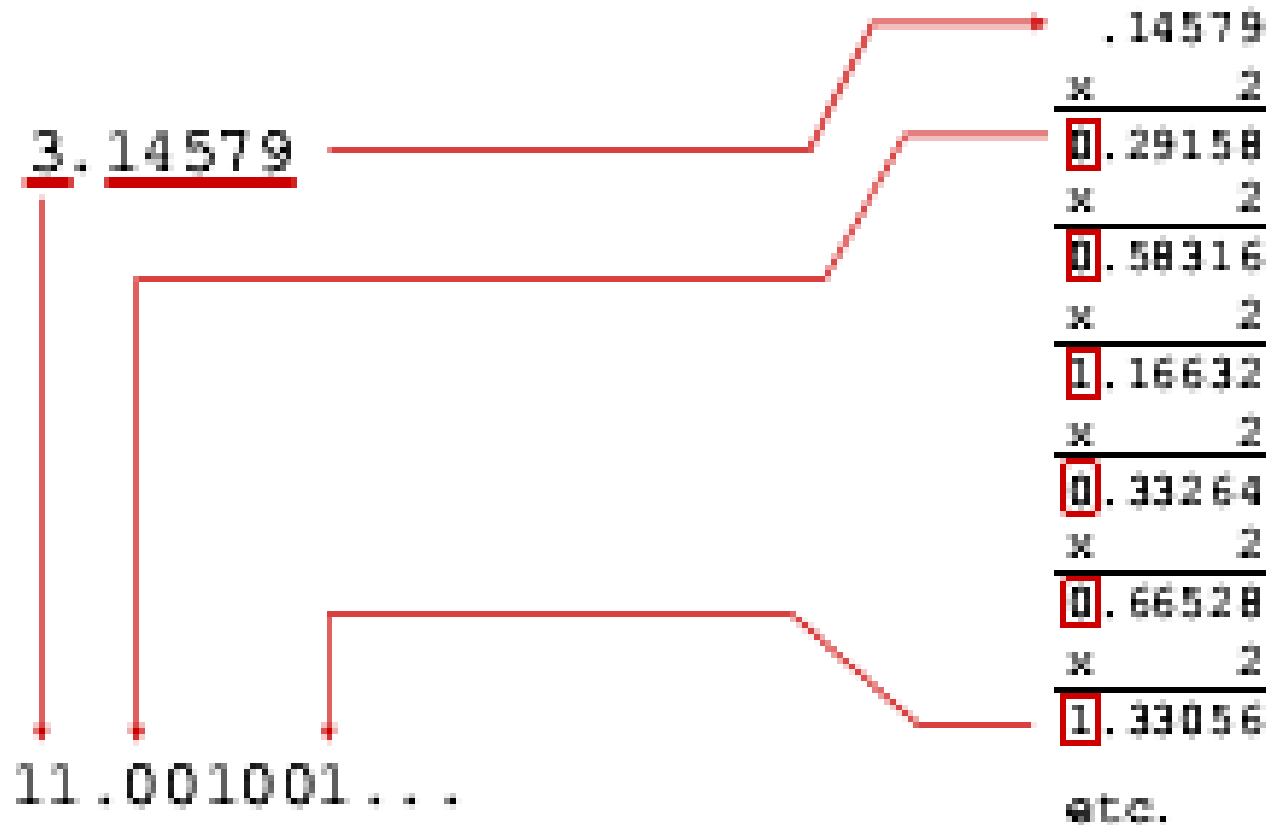
Ví Dụ

- Thực hiện phép chuyển đổi giữa các hệ thống số

| Decimal | Binary | Octal | Hexadecimal |
|---------|---------|-------|-------------|
| 35 | | | |
| | 1101101 | | |
| | | 712 | |
| | | | 1AF |

Chuyển phần thập phân sang Nhị Phân

- Phần thập phân \Rightarrow Số Nhị Phân



Ví dụ: $189.023_{10} \Rightarrow$ Số Nhị Phân

| | | | | |
|---------|-----|---------|--------------------------|------|
| $189/2$ | $=$ | 94 dư 1 | $0.023 \times 2 = 0.046$ | dư 0 |
| $94/2$ | $=$ | 47 dư 0 | $0.046 \times 2 = 0.092$ | dư 0 |
| $47/2$ | $=$ | 23 dư 1 | $0.092 \times 2 = 0.184$ | dư 0 |
| $23/2$ | $=$ | 11 dư 1 | $0.184 \times 2 = 0.368$ | dư 0 |
| $11/2$ | $=$ | 5 dư 1 | $0.368 \times 2 = 0.736$ | dư 0 |
| $5/2$ | $=$ | 2 dư 1 | $0.736 \times 2 = 1.472$ | dư 1 |
| $2/2$ | $=$ | 1 dư 0 | $0.472 \times 2 = 0.944$ | dư 0 |
| $1/2$ | $=$ | 0 dư 1 | ... | |

$$189.023 = 10111101.0000010_2$$

Ví Dụ

- Thực hiện phép chuyển đổi giữa các hệ thống số

| Decimal | Binary | Octal | Hexadecimal |
|---------|----------|-------|-------------|
| 29.8 | | | |
| | 110.1101 | | |
| | | 3.07 | |
| | | | C.82 |



Các phép tính số nhị phân

- **Phép Cộng**
- **Phép Nhân**
- **Phép Trừ**

Phép Cộng

- Cộng 2 số nhị phân 1-bit

| A | B | A + B |
|---|---|-------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 10 |

Phép Cộng

- Phép cộng 2 số nhị phân không dấu

| | | | | | |
|----|-------------|------------|----|----------------|----------------|
| a) | 11 | (3) | b) | 11.011 | (3.375) |
| | <u>+110</u> | <u>(6)</u> | | <u>+10.110</u> | <u>(2.750)</u> |
| | 1001 | (9) | | 110.001 | (6.125) |

Phép Nhân

- Nhân 2 số nhị phân 1-bit

| A | B | $A * B$ |
|---|---|---------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Phép Nhân

- Phép nhân 2 số nhị phân không dấu

$$\begin{array}{r} 1110 \\ \times 1011 \\ \hline 1110 \\ 1110 \\ 0000 \\ 1110 \\ \hline 10011010 \end{array}$$

Phép Trừ

- Quy tắc thực hiện phép trừ như sau:

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

$$[1]0 - 1 = 1 \text{ Mượn 1}$$


- VD: Thực hiện phép trừ 2 số nhị phân 5 bits: 00111 từ 10101

$$\begin{array}{r} 10101 \\ - 00111 \\ \hline 01110 \end{array} \quad \begin{array}{r} 21 \\ - 7 \\ \hline = 14 \end{array}$$



3. Các cổng Logics Cơ Bản

NỘI DUNG

- 
- Cổng Logic cơ bản AND, OR, NOT
 - Mạch Logic \Rightarrow Biểu thức Đại Số

Tổng Quát

- Đại Số Boolean chỉ xử lý 2 giá trị duy nhất (2 trạng thái logic): **0** và **1**

| Logic 0 | Logic 1 |
|-------------|---------------|
| False | True |
| Off | On |
| LOW | HIGH |
| No | Yes |
| Open switch | Closed switch |

- 3 cổng logic cơ bản:
 - **OR**, **AND** và **NOT**

Bảng Sự thật / Chân trị

- Mô tả các mối quan hệ giữa inputs và outputs của một mạch logic



| Inputs | | Output |
|--------|---|--------|
| A | B | x |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- Số lượng các mục tương ứng với số inputs
 - A 2-input bảng sẽ có

| |
|---|
| ? |
|---|

 mục
 - A 3-input bảng sẽ có

| |
|---|
| ? |
|---|

 mục

Cổng Logic OR

- Biểu thức Boolean cho cổng logic **OR** có hoạt động:
 - $X = A + B$ — Đọc là “X bằng A OR B”

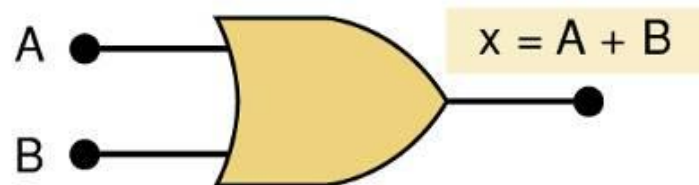
Dấu + không có nghĩa là phép cộng thông thường, mà là ký hiệu cho cổng logic OR

- Bảng sự thật và biểu diễn cổng logic OR có 2 inputs:

OR

| A | B | $x = A + B$ |
|---|---|-------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(a)



OR Gate

(b)

Cổng Logic AND

- Cổng logic AND thực hiện tương tự như phép nhân:

– $X = A \bullet B$ — Đọc là “X bằng A AND B”

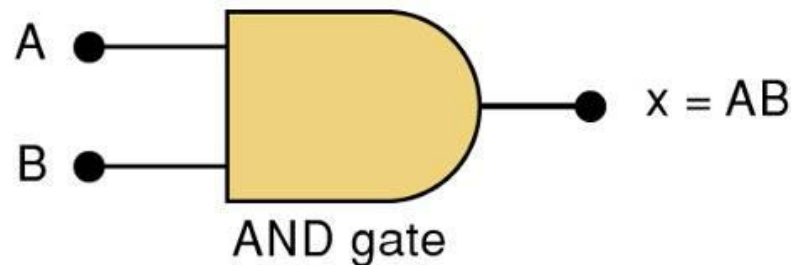
Dấu \bullet không có nghĩa là phép nhân thông thường ,
mà là ký hiệu cho cổng logic AND

- Bảng sự thật và biểu diễn cổng logic AND có 2 inputs:

AND

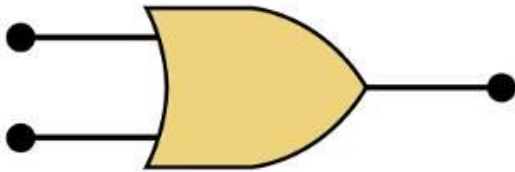
| A | B | $x = A \bullet B$ |
|---|---|-------------------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(a)



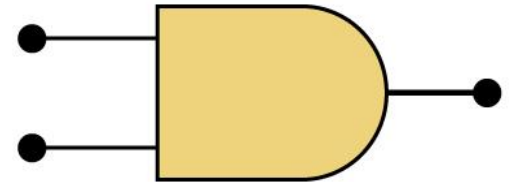
(b)

OR vs AND



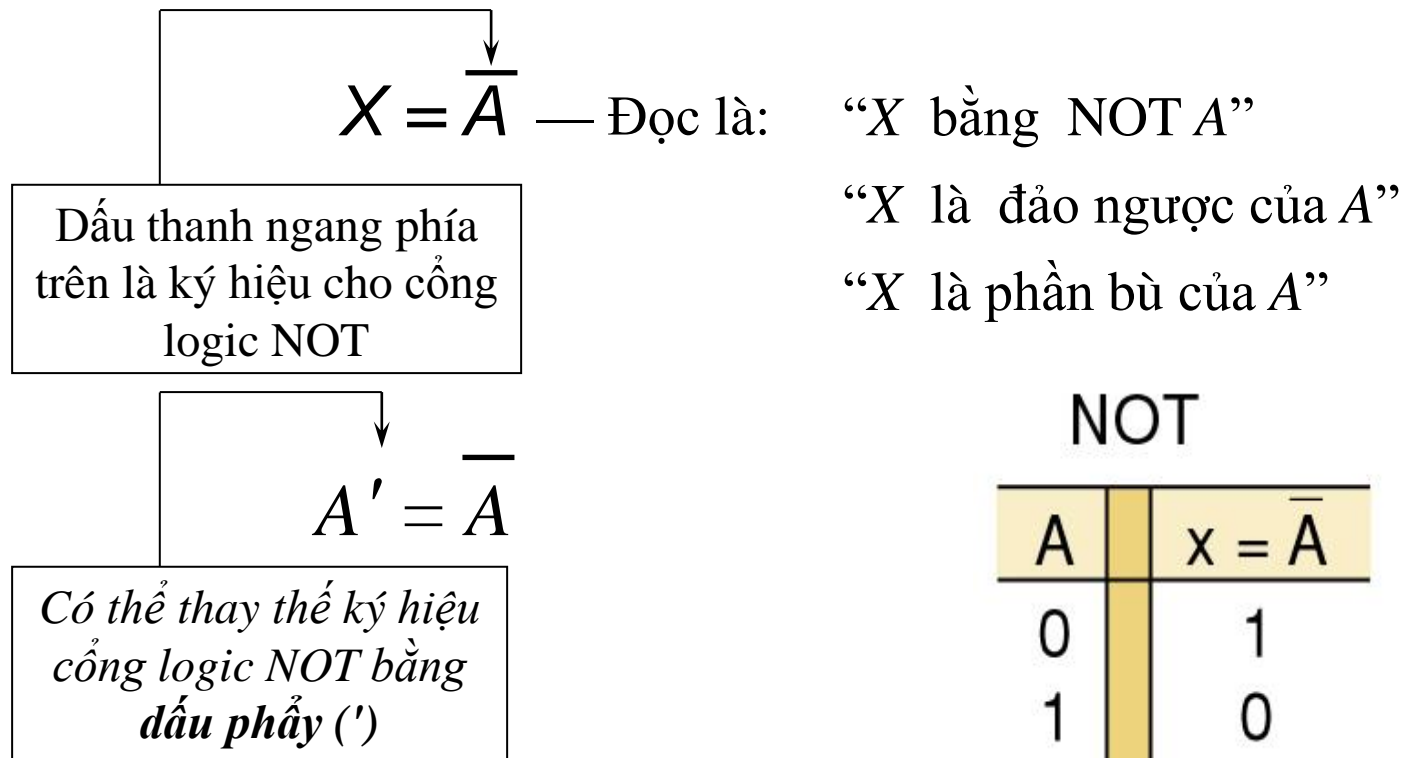
Ký hiệu của cổng logic OR có nghĩa là output sẽ có trạng thái là HIGH khi có bất kỳ input nào có trạng thái là HIGH

Ký hiệu của cổng logic AND có nghĩa là output sẽ có trạng thái là HIGH khi tất cả các input đều có trạng thái là HIGH



Cổng Logic NOT

- Biểu thức Boolean đối với cổng logic **NOT**

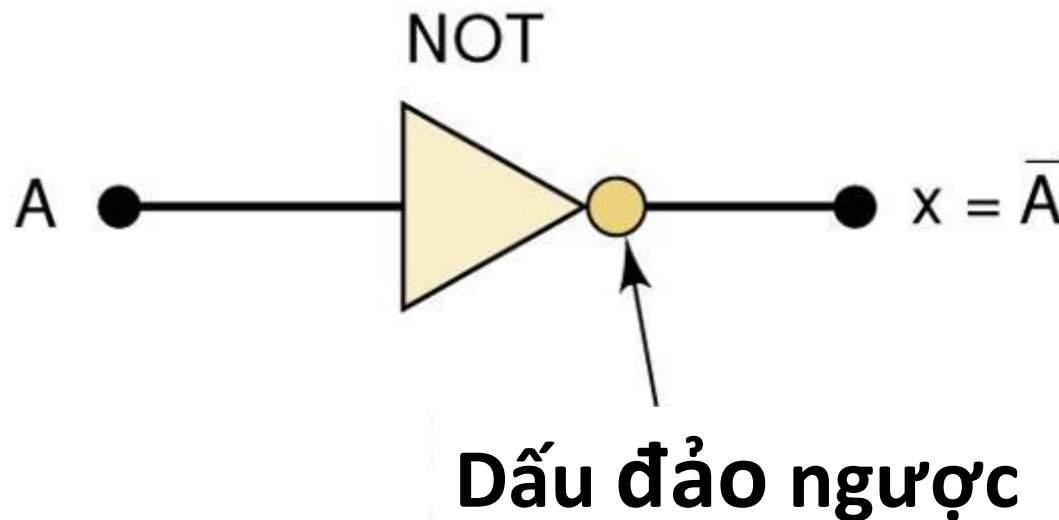


| NOT | | |
|-----|--|--------------------|
| A | | $x = \overline{A}$ |
| 0 | | 1 |
| 1 | | 0 |

Bảng sự thật cổng
Logic NOT

Cổng Logic NOT

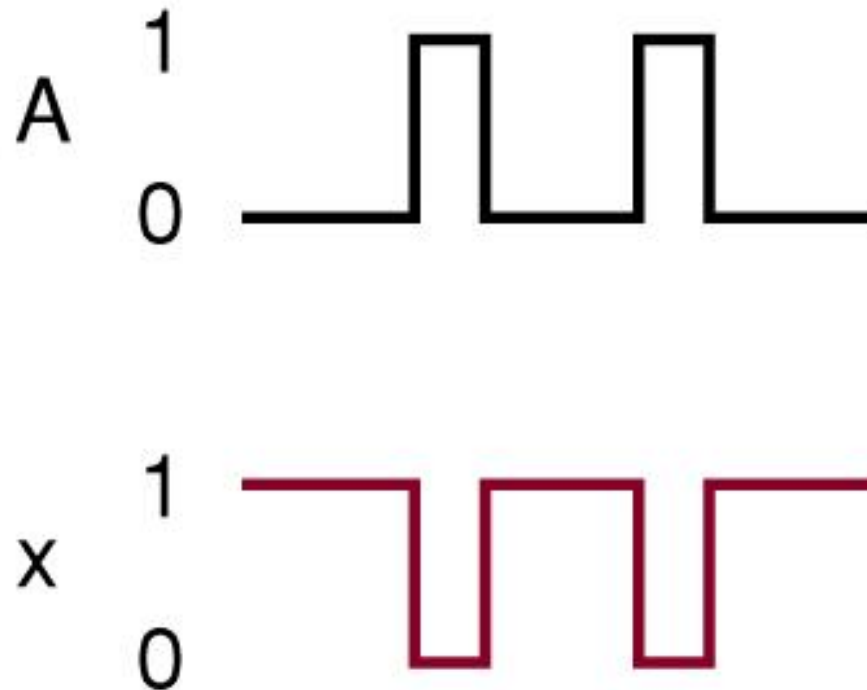
- Cổng logic NOT có thể gọi chung là *INVERTER*



Cổng logic này luôn luôn chỉ có duy nhất 1 input, và trạng thái của output sẽ đối nghịch với trạng thái của input

Cổng Logic NOT

Cổng INVERTER nghịch đảo (*phản bù*) trạng thái tín hiệu của các inputs tại các điểm trong cùng bước sóng



Bất cứ khi nào có: input = 0, output = 1, và ngược lại

Cổng Logic Cơ Bản

OR

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

AND

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

NOT

$$\overline{0} = 1$$

$$\overline{1} = 0$$

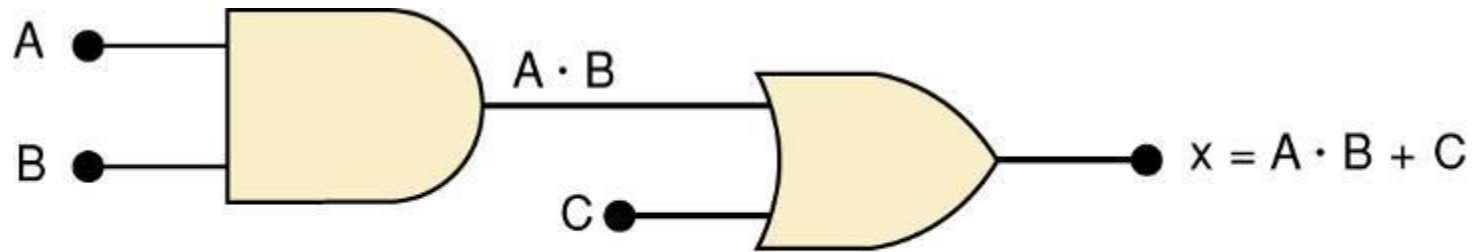
Ba cổng logic Boolean cơ bản có thể mô tả được bất kỳ mạch logic nào



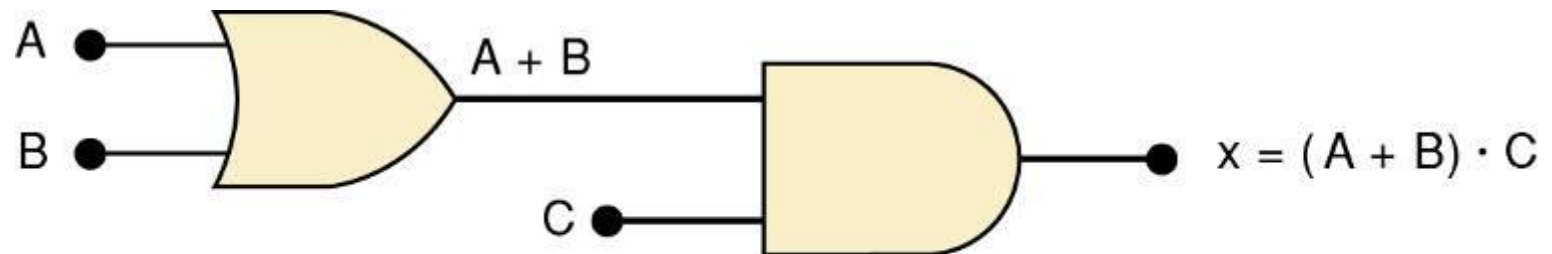
Mạch Logic \Rightarrow Biểu thức đại số

Mô tả mạch logic đại số

- Nếu một biểu thức có chứa cả hai cổng Logic **AND** và **OR**, thì cổng logic **AND** sẽ được thực hiện trước :

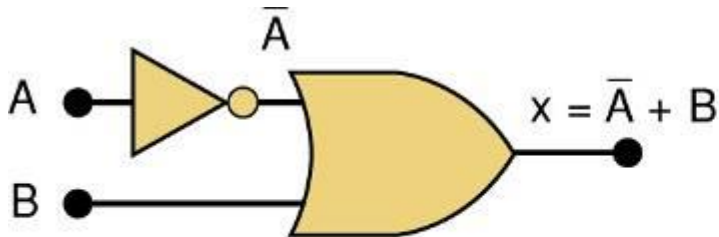


- Trừ khi có một dấu ngoặc trong biểu thức

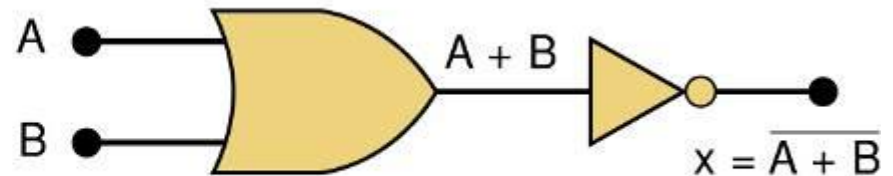


Mô tả mạch logic đại số

- Bất cứ khi nào có sự xuất hiện của cổng logic INVERTER trong mạch, output sẽ có giá trị tương đương với input, kèm theo dấu thanh ngang — trên đầu của output
 - Input A qua một inverter sẽ có output là \bar{A}

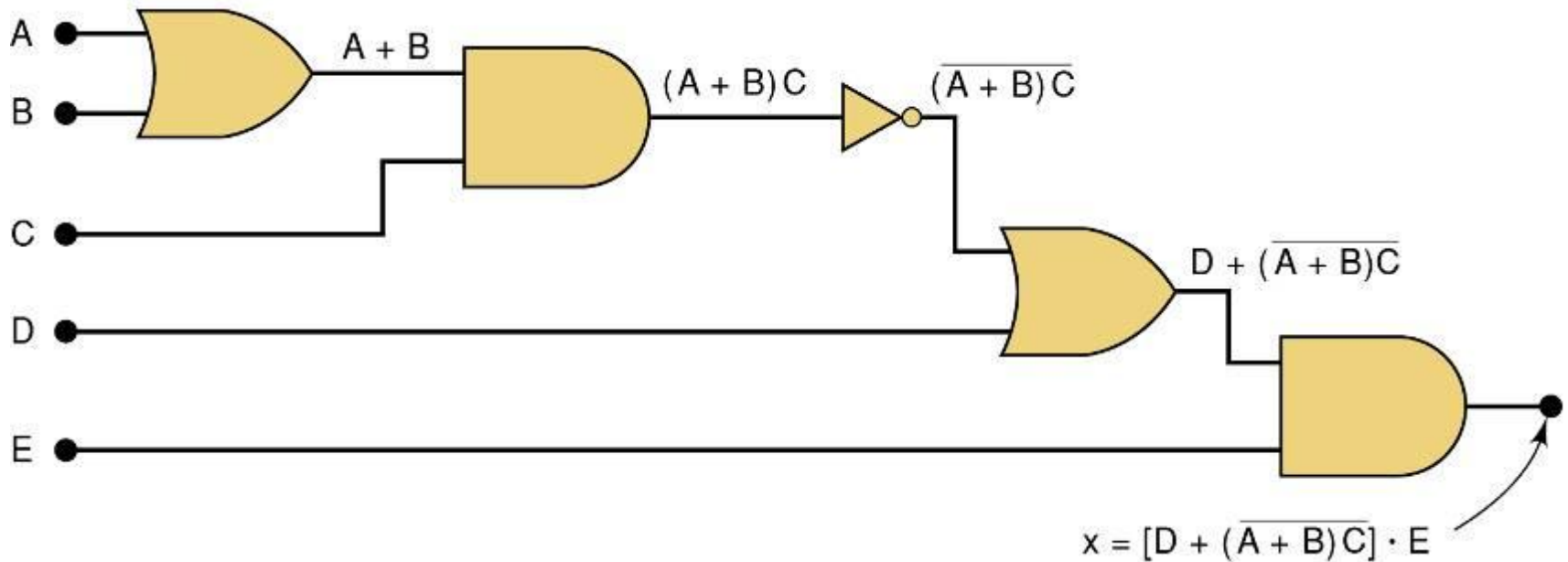
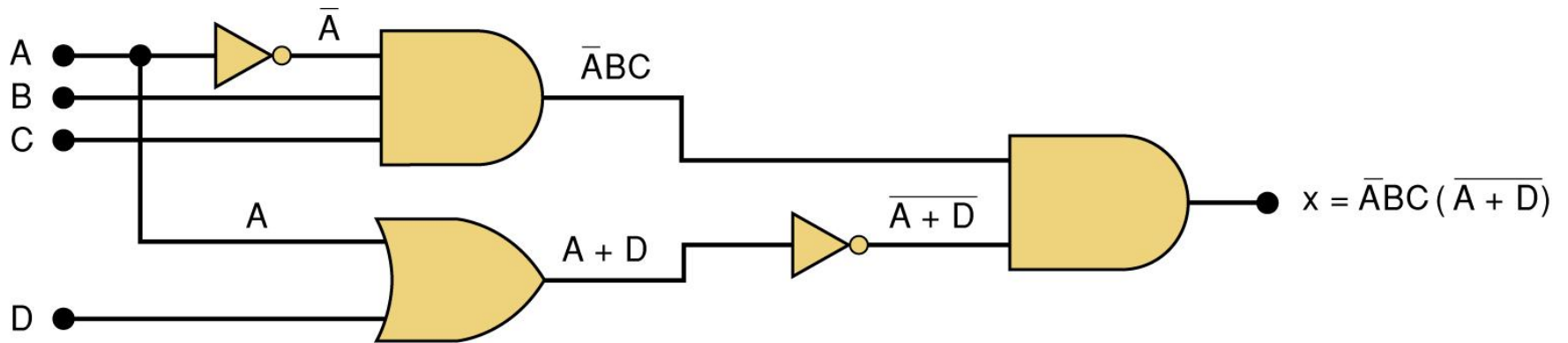


(a)



(b)

Ví Dụ



Đánh giá OUTPUTs của mạch logic

• Ex:
$$X = \overline{A\overline{B}C(D + \overline{E}) + FG}$$

Quy tắc đánh giá một biểu thức Boolean:

- Thực hiện tất cả đảo ngược đối với các inputs đơn trước
- Thực hiện xử lý tất cả các phép tính trong ngoặc trước
- Thực hiện xử lý cổng logic AND trước rồi mới đến cổng logic OR, trừ khi trường hợp cổng logic OR ở trong ngoặc trước
- Nếu cả một biểu thức có thanh ngang trên đầu, thực hiện các phép tính bên trong biểu thức trước, và sau đó đảo ngược kết quả lại

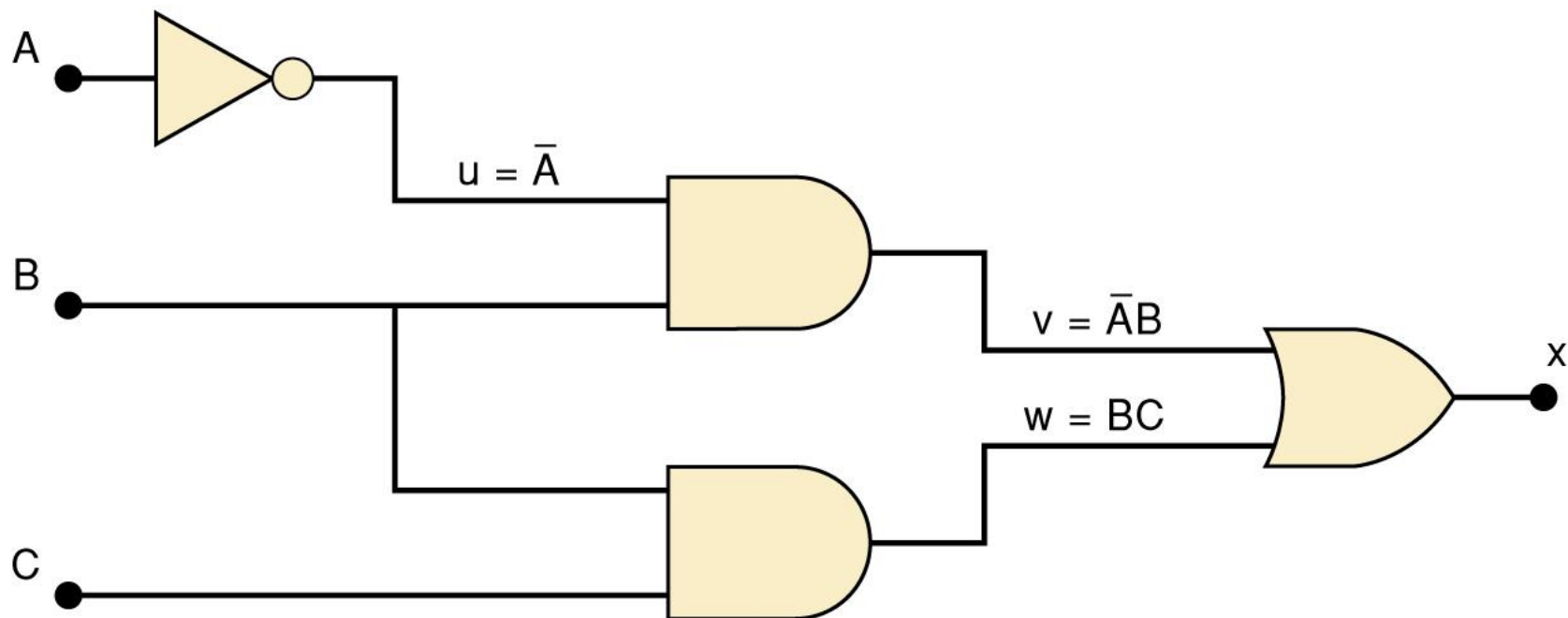
Đánh giá OUTPUTs của mạch logic



- Cách tốt nhất để phân tích một mạch gồm có nhiều cổng logic khác nhau là sử dụng **bảng sự thật**
 - Cho phép chúng ta có thể phân tích một cổng hoặc một tổ hợp các cổng logic có trong mạch **cùng một lúc**
 - Cho phép chúng ta dễ dàng kiểm tra lại hoạt động của mạch logic tổ hợp một cách chính xác nhất
 - Bảng sự thật giúp ích trong việc phát hiện và xử lý lỗi hay sự cố xuất hiện có trong mạch logic tổ hợp

Đánh giá OUTPUTs của mạch logic

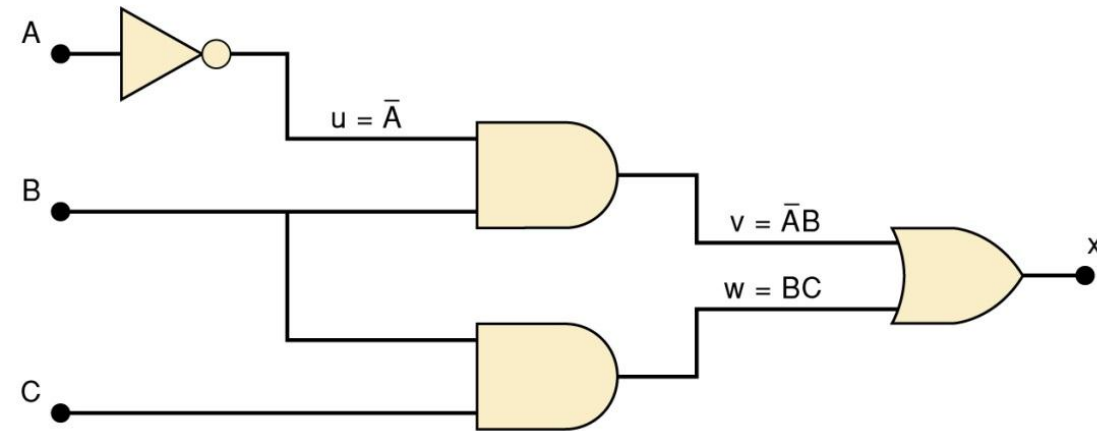
- Đánh giá outputs của mạch logic sau:



Đánh giá OUTPUTs của mạch logic

- Bước 1: Liệt kê tất cả các inputs có trong mạch logic tổ hợp
- Bước 2: Tạo ra một cột trong bảng sự thật cho mỗi tín hiệu

trung gian (node)

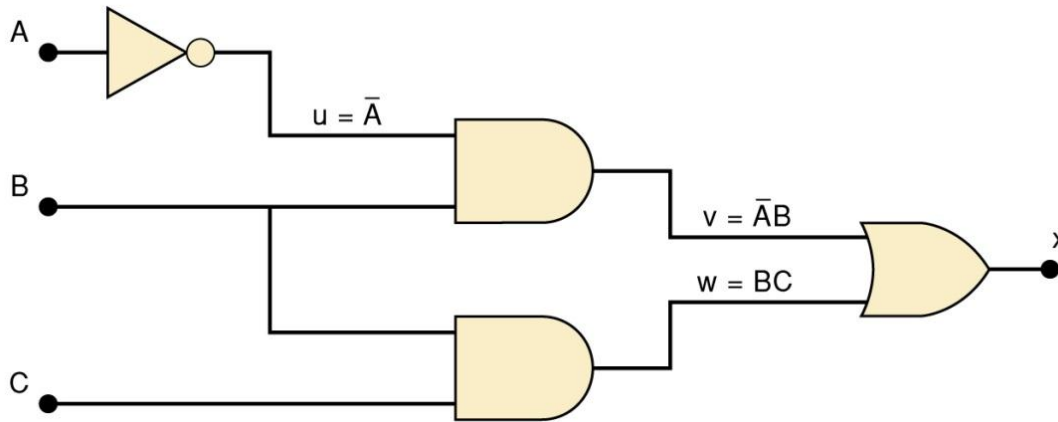


| A | B | C | $u = \bar{A}$ | $v = \bar{A}B$ | $w = BC$ | $x = v + w$ |
|---|---|---|---------------|----------------|----------|-------------|
| 0 | 0 | 0 | 1 | | | |
| 0 | 0 | 1 | 1 | | | |
| 0 | 1 | 0 | 1 | | | |
| 0 | 1 | 1 | 1 | | | |
| 1 | 0 | 0 | 0 | | | |
| 1 | 0 | 1 | 0 | | | |
| 1 | 1 | 0 | 0 | | | |
| 1 | 1 | 1 | 0 | | | |

Node u đã được điền vào như là kết quả của phần bù của tín hiệu input A

Đánh giá OUTPUTs của mạch logic

- Bước 3: điền vào các giá trị tín hiệu của cột node v



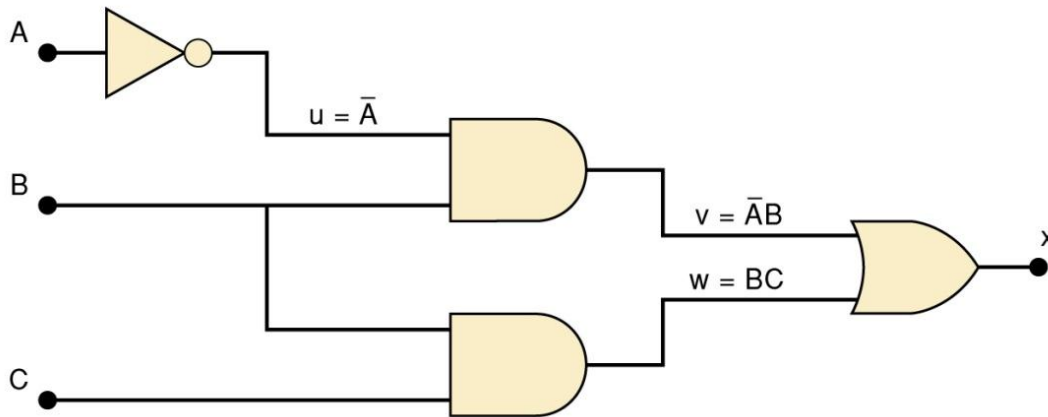
| A | B | C | $u = \bar{A}$ | $v = \bar{A}B$ | $w = BC$ | $x = v + w$ |
|---|---|---|---------------|----------------|----------|-------------|
| 0 | 0 | 0 | 1 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | | |
| 0 | 1 | 0 | 1 | 1 | | |
| 0 | 1 | 1 | 1 | 1 | | |
| 1 | 0 | 0 | 0 | 0 | | |
| 1 | 0 | 1 | 0 | 0 | | |
| 1 | 1 | 0 | 0 | 0 | | |
| 1 | 1 | 1 | 0 | 0 | | |

$v = \bar{A}B$ — Node v sẽ có giá trị HIGH

Khi A (node u) là HIGH và B là HIGH

Đánh giá OUTPUTs của mạch logic

- Bước 4: Dự đoán trước giá trị tín hiệu của node w là outputs của cổng logic BC

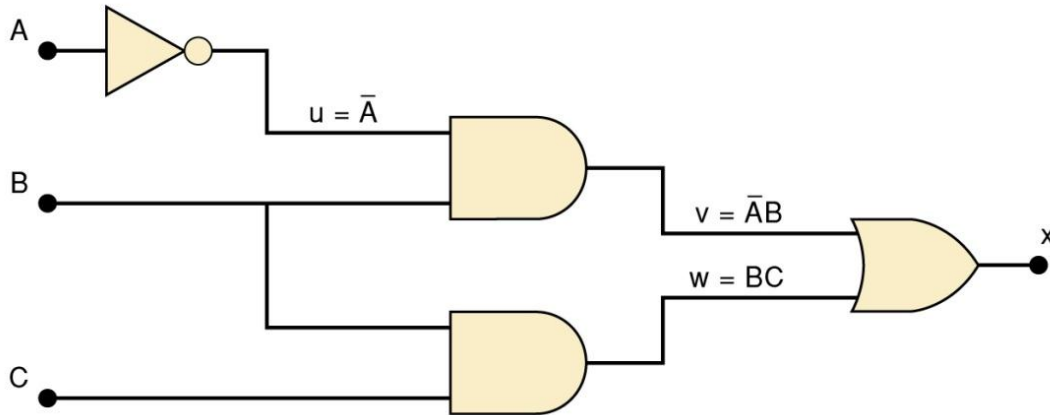


| A | B | C | $u = \bar{A}$ | $v = \bar{A}B$ | $w = BC$ | $x = v + w$ |
|---|---|---|---------------|----------------|----------|-------------|
| 0 | 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 0 | 0 | 0 | |
| 1 | 1 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 1 | 0 | 0 | 1 | |

Cột này là HIGH khi và chỉ khi B là HIGH và cả C là HIGH

Đánh giá OUTPUTs của mạch logic

- Bước cuối cùng: kết hợp một cách logic 2 cột \underline{v} và \underline{w} để dự đoán cho output \underline{x}

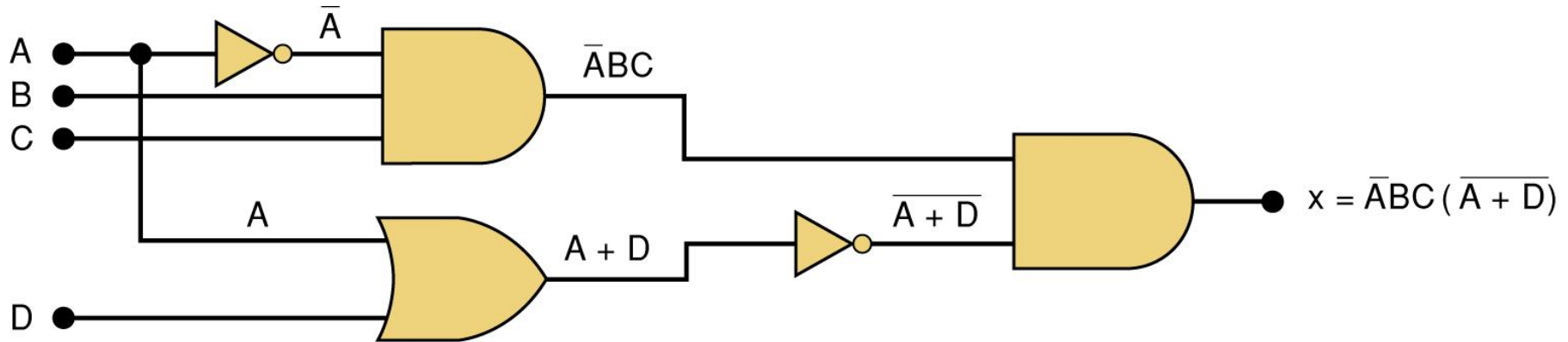


| A | B | C | $\underline{v} = \bar{A}$ | $\underline{w} = \bar{A}B$ | $w = BC$ | $x = v + w$ |
|---|---|---|---------------------------|----------------------------|----------|-------------|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |

Từ biểu thức $x = v + w$, thì x output sẽ là HIGH khi v OR w là HIGH

Đánh giá OUTPUTS của mạch logic

- Ví dụ:



Ví Dụ

- Vẽ sơ đồ mạch logic với output như sau: $y = AC + B\bar{C} + \bar{A}BC$



5. Mạch Tổ Hợp

Nội dung



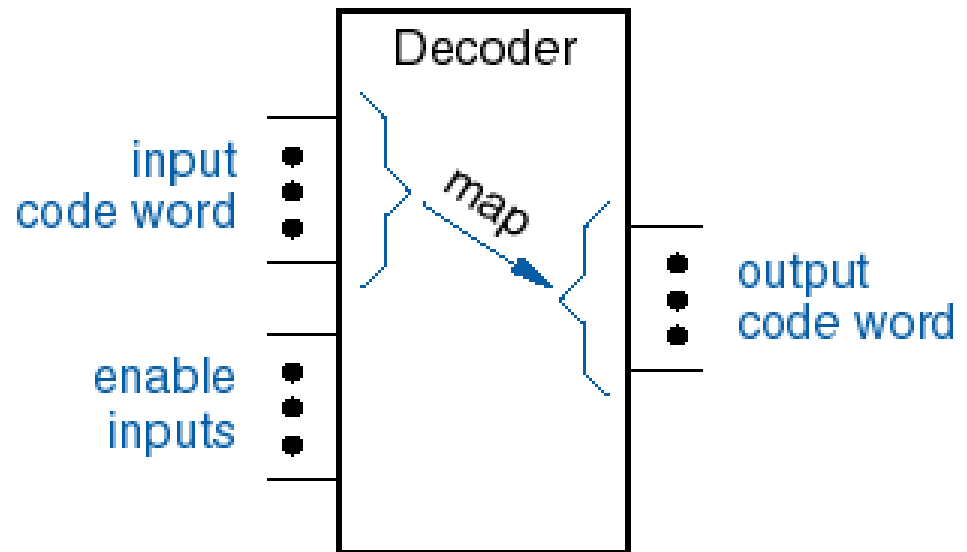
- Mạch giải mã (Decoder)/ Mạch mã hoá (Encoder)
- Mạch dồn kênh (Multiplexer)/ Mạch chia kênh (Demultiplexer)



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



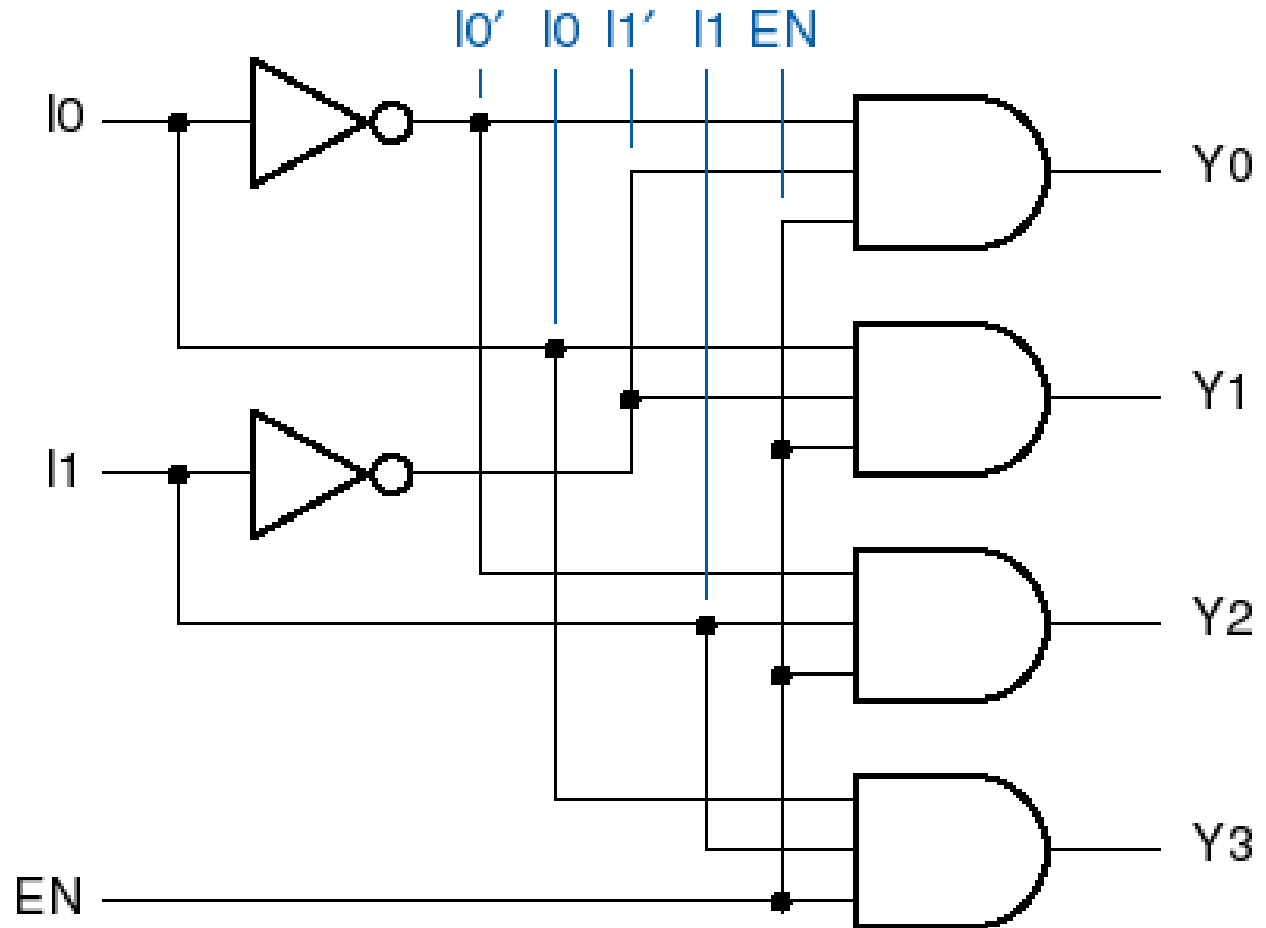
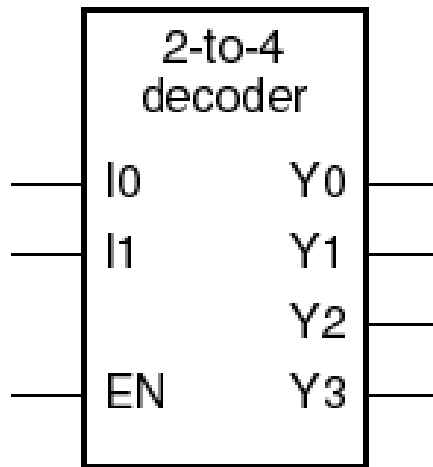
Mạch giải mã nhị phân (Binary Decoders)

- 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 |

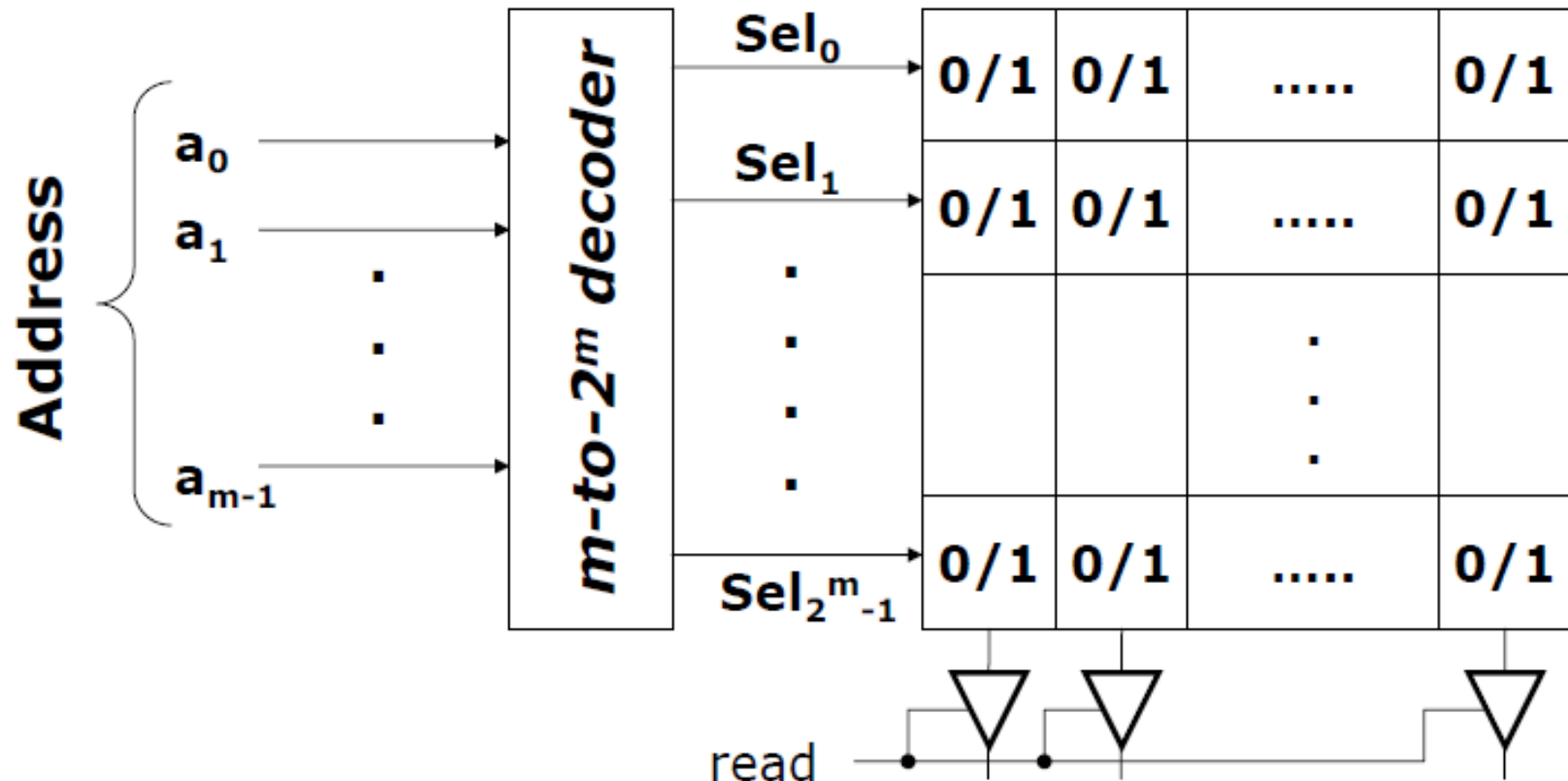
Chú ý “x” (kí hiệu ngõ vào don’t care)

Giải mã nhị phân 2-ra-4



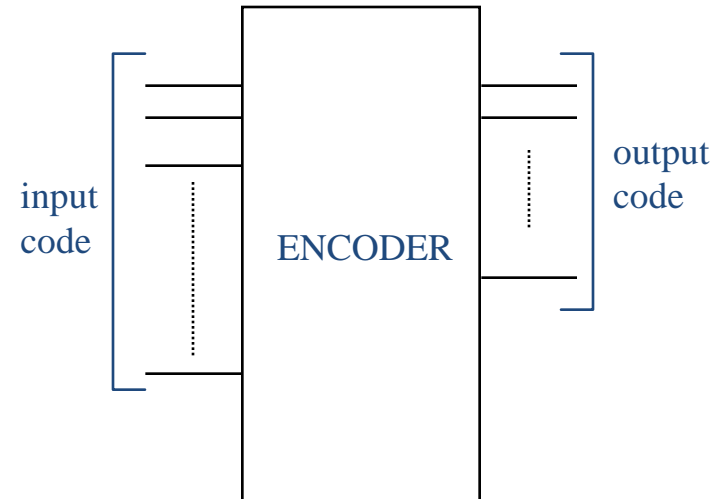
Ứng dụng của mạch giải mã

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

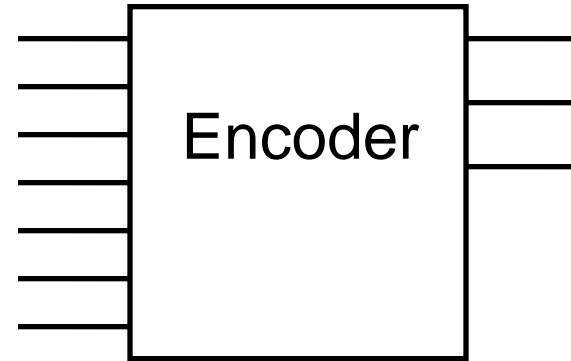
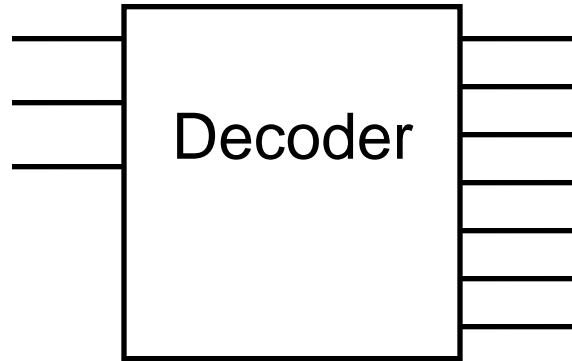


Mạch mã hoá

- 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

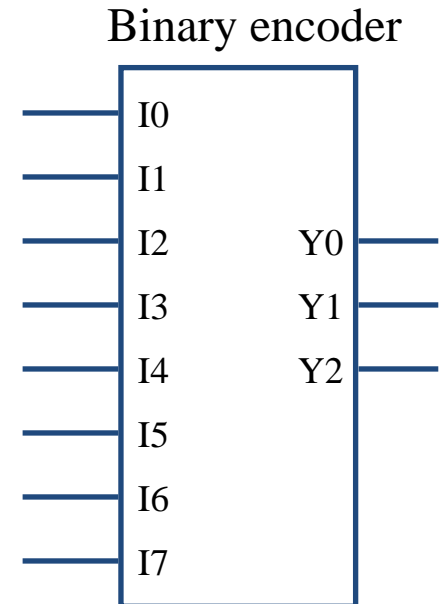
- $n \rightarrow 2^n$
- Input code: Mã nhị phân
- Output code: 1-trong- 2^n

- $2^n \rightarrow n$
- 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

| I0 | I1 | Ngõ vào | | I4 | I5 | I6 | I7 | Ngõ ra | | |
|----|----|---------|----|----|----|----|----|--------|----|----|
| | | I2 | I3 | | | | | 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 |



Hiện thực mạch mã hóa 8-ra-3

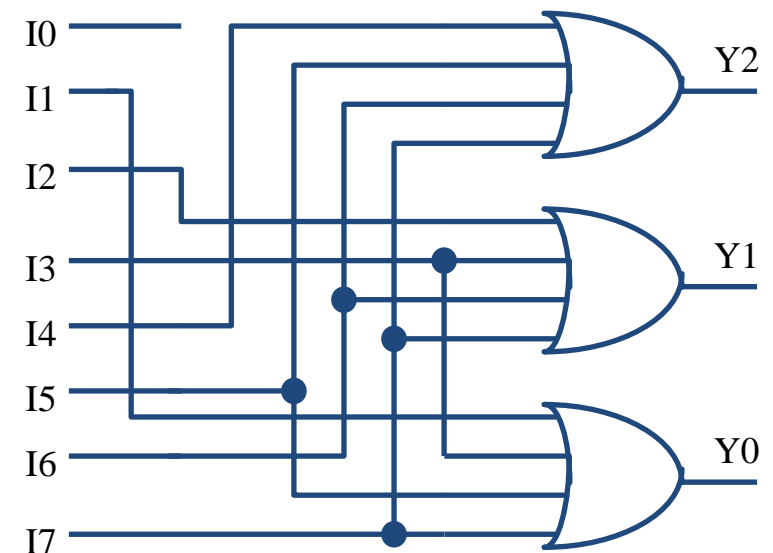
| Ngõ vào | | | | | | | | Ngõ ra | | |
|---------|----|----|----|----|----|----|----|--------|----|----|
| I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 | Y2 | Y1 | Y0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
|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

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

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

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

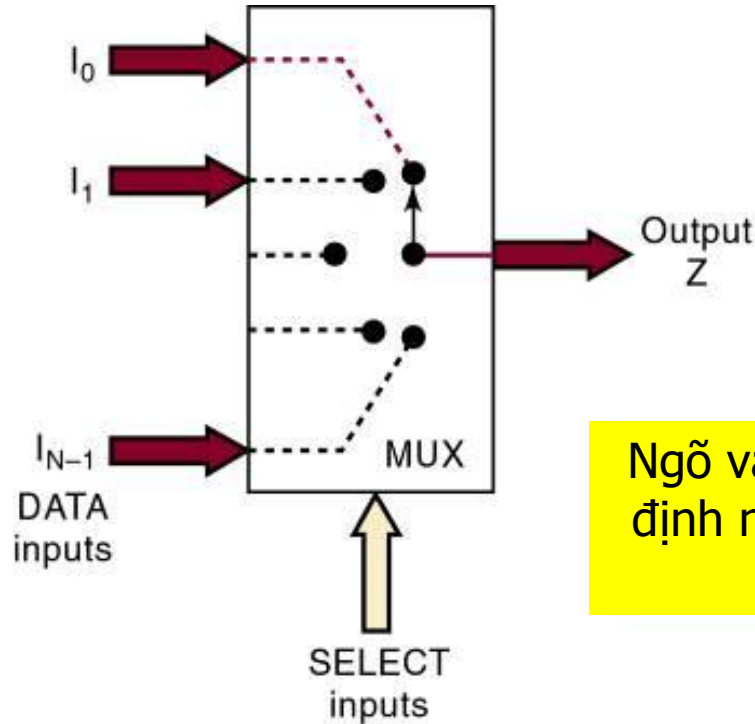




Multiplexer (MUX)/ Demultiplexer (DeMUX)

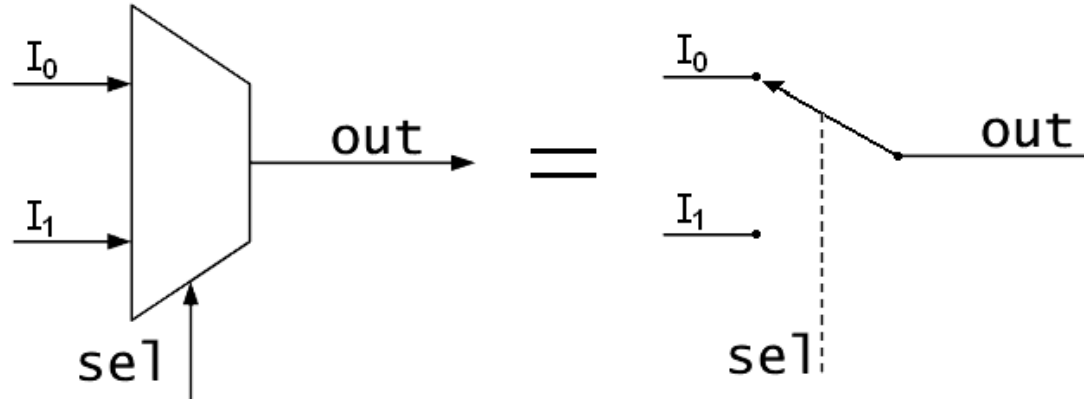
Multiplexer

- Multiplexer (MUX) truyền một trong những ngõ vào của nó làm ngõ ra dựa trên tín hiệu Select



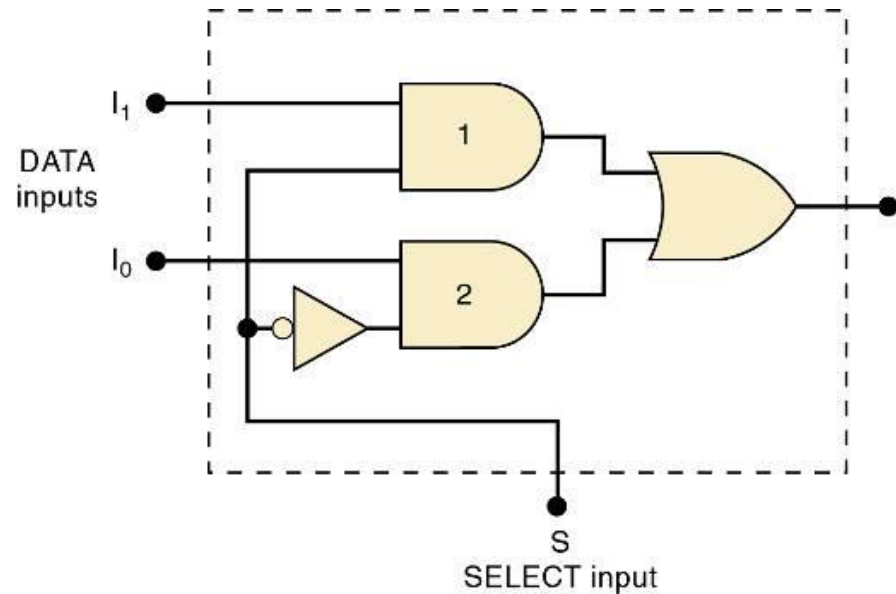
Ngõ vào SELECT sẽ xác định ngõ vào nào được truyền ra Z

2-to-1 Multiplexer



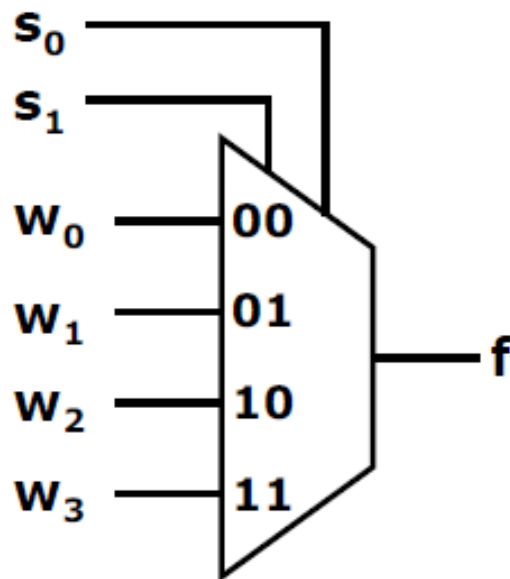
$$Out = I_0 * \overline{Sel} + I_1 * Sel$$

| Sel | Out |
|-----|-------|
| 0 | I_0 |
| 1 | I_1 |



4-ra-1 Mux

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

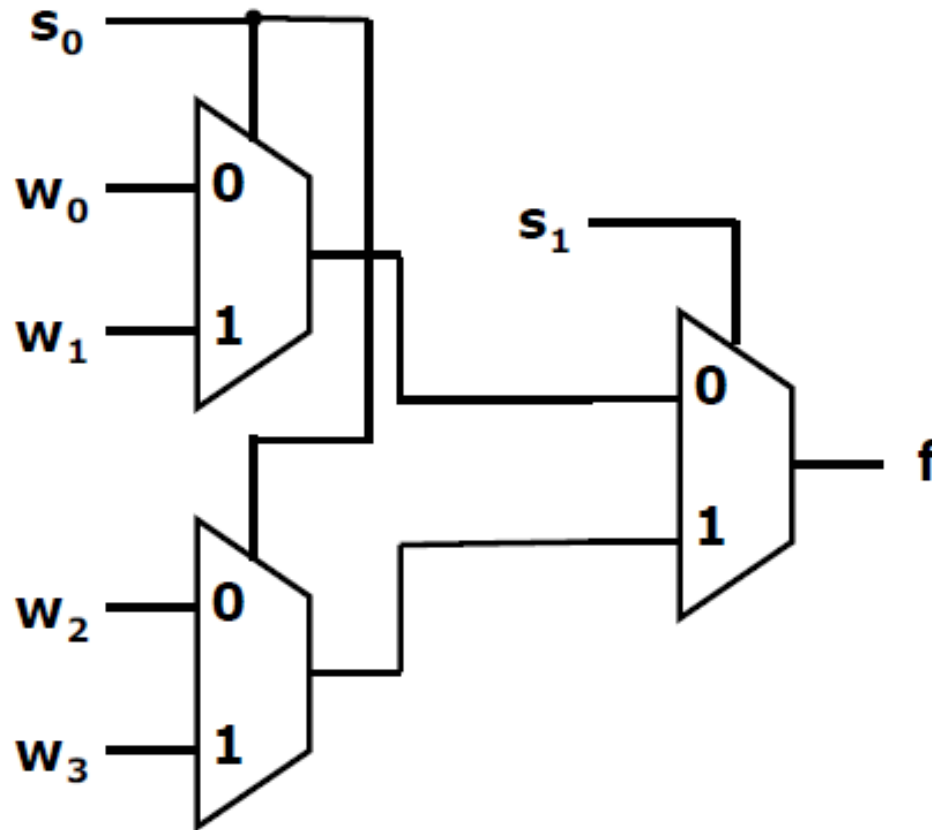


| s_1 | s_0 | f |
|-------|-------|-------|
| 0 | 0 | w_0 |
| 0 | 1 | w_1 |
| 1 | 0 | w_2 |
| 1 | 1 | w_3 |

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

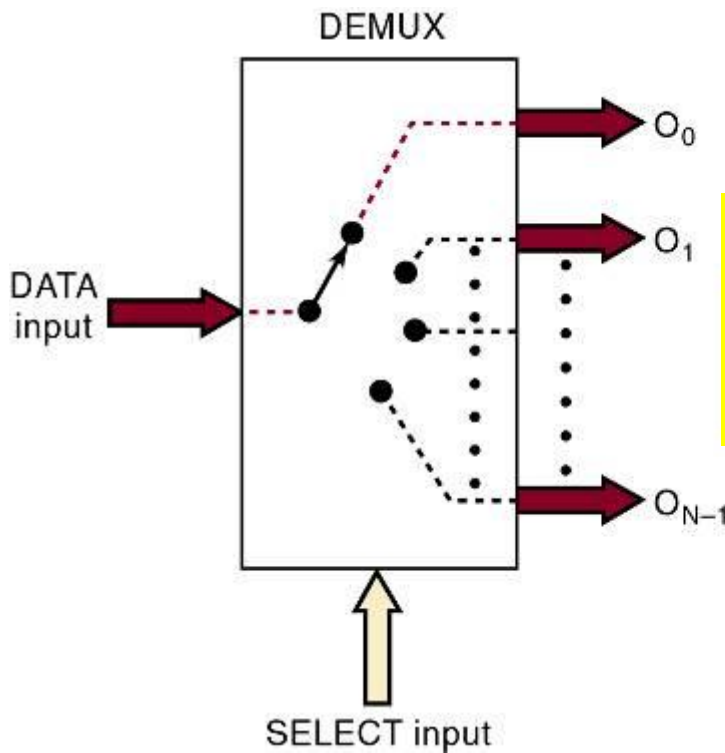
Xây dựng MUX 4-ra-1

- Từ MUX 2-ra-1



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 mà ngõ vào DATA sẽ truyền qua



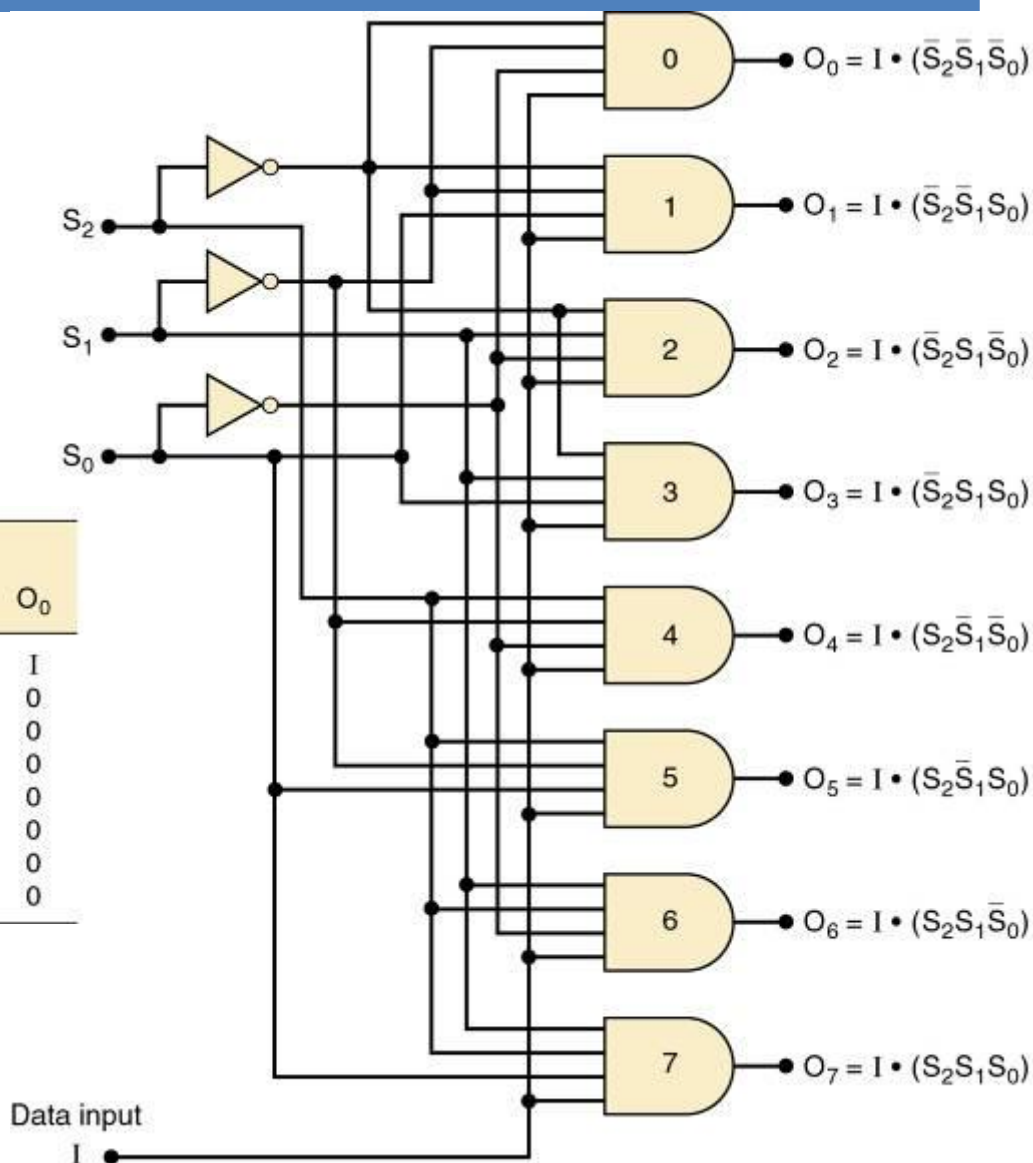
DATA được truyền ra một và chỉ một ngõ ra được xác định bởi mã của ngõ vào SELECT

DEMUX

1-ra-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 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Chú ý: I là ngõ vào
DATA



Bài tập

- Xây dựng bộ Mux 16 – 1, chỉ sử dụng các bộ mux 4 – 1
- Xây dựng bộ Mux 8 – 1, chỉ sử dụng các bộ mux 2 – 1
- Xây dựng bộ Mux 8 – 1, chỉ sử dụng 1 bộ mux 2 – 1, và 2 bộ mux 4 – 1.