

Chương 2:

MẠCH LOGIC

Giới Thiệu

- ❖ Các hoạt động của máy tính số được dựa trên việc lưu trữ và xử lý dữ liệu nhị phân.
- ❖ Chương này sẽ xem xét các thành phần cơ bản mà từ đó tạo nên tất cả máy tính.
- ❖ Bắt đầu bằng đại số Boolean, tiếp theo là giới thiệu về một số cổng logic. Cuối cùng là thiết kế một số mạch tổ hợp và mạch tuần tự.

Đại Số Boolean

- ❖ Đại số Boolean nghiên cứu các phép toán thực hiện trên các biến nhị phân.
- ❖ Đại số Boolean bao gồm một số luật và chỉ có ba phép toán luận lý: AND, OR và NOT.
- ❖ Chúng được ký hiệu như sau:

$$A \text{ AND } B = A . B$$

$$A \text{ OR } B = A + B$$

$$\text{NOT } A = \bar{A}$$

Bảng 2.1: Các phép toán Boolean cho hai biến P và Q.

P	Q	NOT P (\bar{P})	P AND Q ($P.Q$)	P OR Q ($P + Q$)
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Đại Số Boolean

❖ Các luật của đại số Boolean:

1. Luật giao hoán (*Commutative law*):

$$A \cdot B = B \cdot A$$

$$A + B = B + A$$

2. Luật phân phối (*Distributive law*):

$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

3. Phần tử đồng nhất (*Identity element*):

$$1 \cdot A = A$$

$$0 + A = A$$

4. Phần tử bù/nghịch đảo (*Inverse element*):

$$A \cdot \bar{A} = 0$$

$$A + \bar{A} = 1$$

Đại Số Boolean

5. Luật trội/nuốt/null (*Null law*) :

$$0 . A = 0$$

$$1 + A = 1$$

6. Luật lũy đẳng (*Idempotent law*):

$$A . A = A$$

$$A + A = A$$

7. Luật kết hợp (*Associative law*):

$$A . (B . C) = (A . B) . C$$

$$A + (B + C) = (A + B) + C$$

8. Luật bù kép:

$$\overline{\overline{A}} = A$$

9. Luật hấp thụ (*Absorption law*):

$$A . (A + B) = A \text{ và } A + (A . B) = A$$

Đại Số Boolean

10. Định luật DeMorgan:

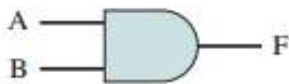
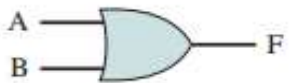
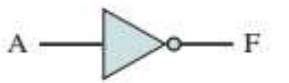

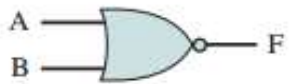
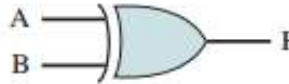
$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

Các Cổng Logic

- ❖ Cổng là một mạch điện tử mà nó tạo tín hiệu xuất là một phép toán Boolean từ các tín hiệu vào của nó.
- ❖ Các cổng cơ bản được sử dụng trong logic số là AND, OR, NOT, NAND, NOR và XOR. Hình 2.1 mô tả sáu cổng này.
- ❖ Tất cả các cổng ngoại trừ NOT có thể có nhiều hơn hơn hai ngõ vào. Do vậy, để tính giá trị biểu thức $(A.B.C)$, chỉ cần sử dụng cổng AND ba ngõ vào.

Các Cổng Logic

Name	Graphical Symbol	Algebraic Function	Truth Table															
AND		$F = A \cdot B$ or $F = AB$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \bar{A}$ or $F = A'$	<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
NAND		$F = \overline{AB}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		$F = A \oplus B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

Hình 2.1: Các cổng logic cơ bản.

Các Cổng Logic

❖ Bất kỳ biểu thức Boolean nào cũng có thể được thực hiện bằng cách chỉ sử dụng các cổng trong mỗi tập sau:

1. AND, OR, NOT

2. AND, NOT

3. OR, NOT

4. NAND

5. NOR

❖ Ví dụ: Phép toán OR có thể được tạo ra bằng cách sử dụng phép toán AND và NOT trong tập thứ hai bằng định lý DeMorgan:

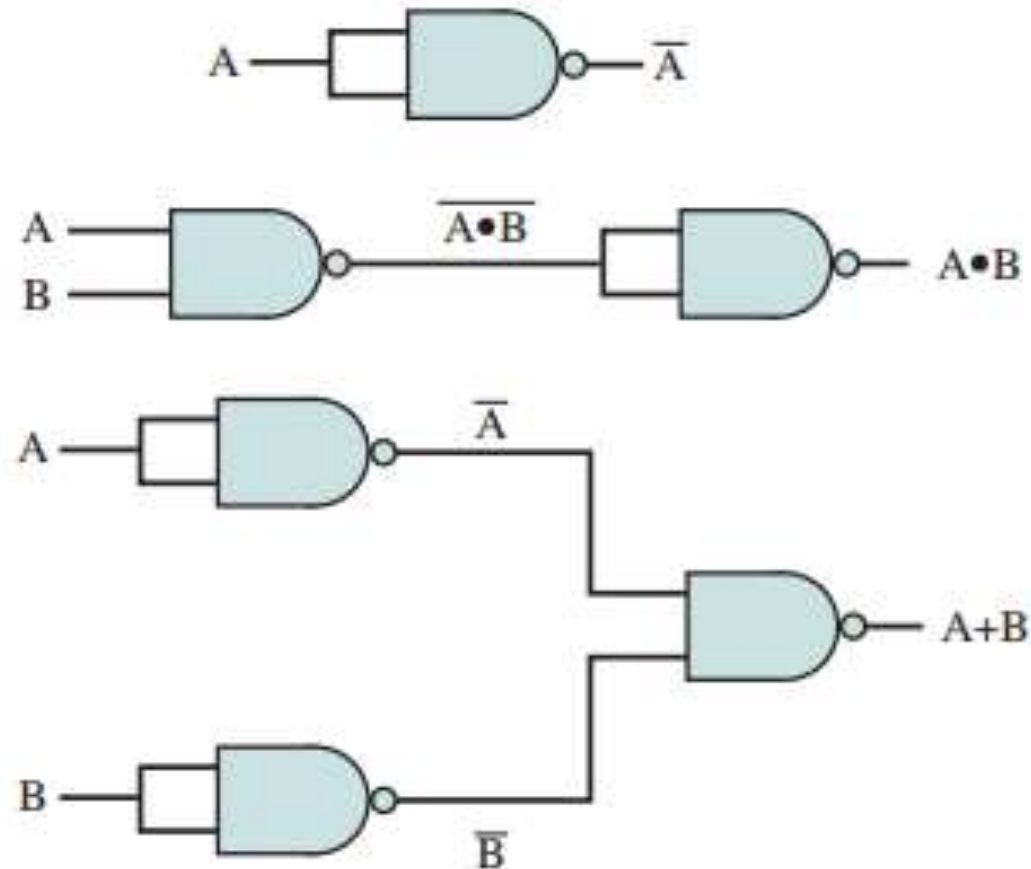
$$A + B = \overline{\overline{A} \cdot \overline{B}}$$

$$A \text{ OR } B = \text{NOT}((\text{NOT } A) \text{ AND } (\text{NOT } B))$$

Các Cổng Logic

- ❖ Bài tập tại lớp: Hãy sinh phép toán AND từ phép toán OR và NOT trong tập thứ 3.
- ❖ Các cổng AND, OR, NOT có thể được tạo ra bởi các cổng NAND trong tập thứ 4 (Hình 2.2) hoặc các cổng NOR. Vì lý do này, thường thì các mạch số chỉ sử dụng các cổng NAND hay NOR.
- ❖ Bài tập tại lớp: Sử dụng các cổng NOR trong tập thứ 5 để tạo ra AND, OR, NOT (Hình 2.3).

Các Cổng Logic

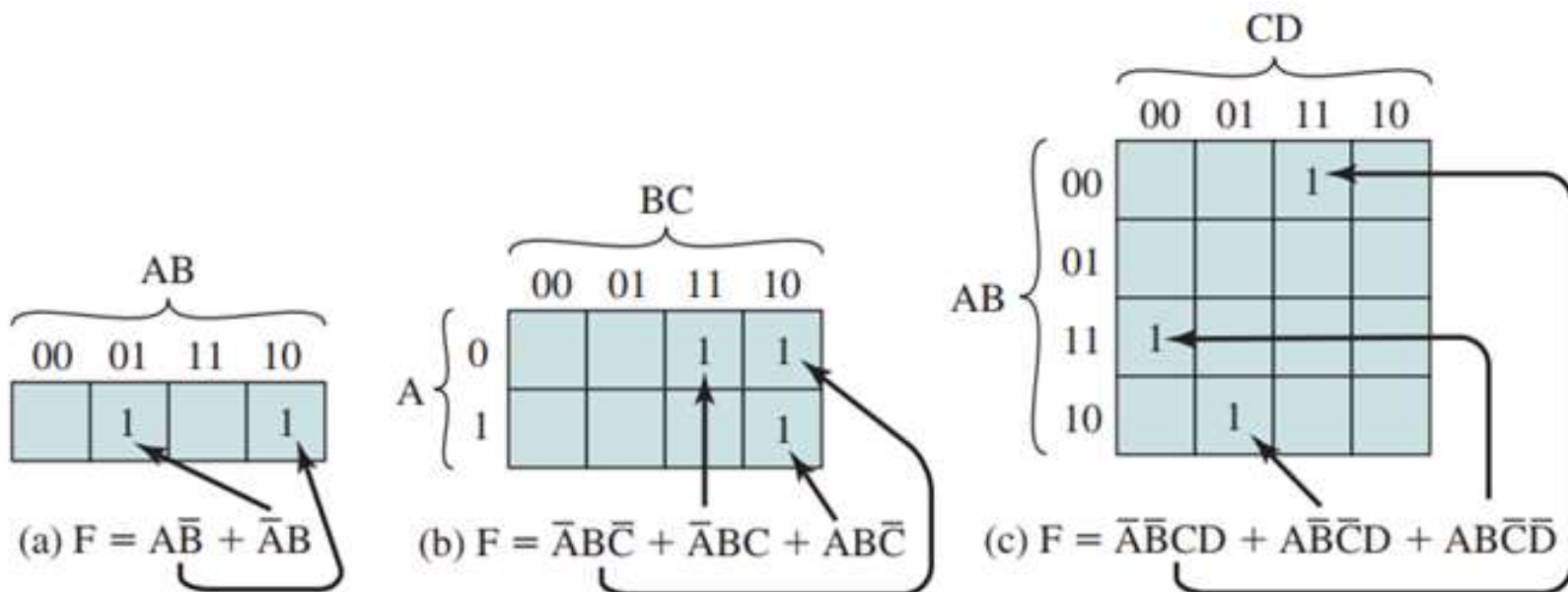


Hình 2.2: Sử dụng các cổng NAND để tạo ra AND, OR, NOT.

Bản Đồ Karnaugh

- ❖ Bản đồ Karnaugh (*Karnaugh map*) được sử dụng để đơn giản hoá biểu thức Boolean với số biến nhỏ.
- ❖ Bản đồ là một ma trận có 2^n ô vuông, thể hiện cho tất cả tổ hợp giá trị có thể của n biến nhị phân.
- ❖ Hình 2.4a cho thấy bản đồ là ma trận có một hàng và bốn cột được sử dụng cho biểu thức Boolean có hai biến. Bởi vì tổ hợp các giá trị của hai biến nhị phân A và B là: 00 ($\bar{A}\bar{B}$), 01 ($\bar{A}B$), 11 (AB) và 10 ($A\bar{B}$).
- ❖ Ba biến tạo thành tám ô vuông (Hình 2.4b) và bốn biến, cần 16 ô vuông (Hình 2.4c).

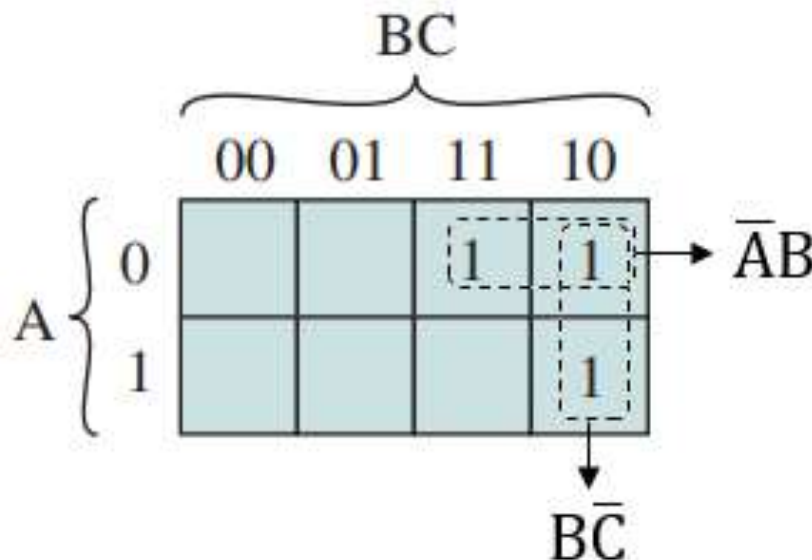
Bản Đồ Karnaugh



Hình 2.4: Sử dụng bảng đồ Karnaugh để biểu diễn biểu thức Boolean.

Bản Đồ Karnaugh

- ❖ Với mỗi toán hạng của một tổng, ô tương ứng trong bản đồ được ghi là 1.
- ❖ Sau khi ghi vào bản đồ, chúng ta nhóm các ô có giá trị 1 kề nhau trong cùng hàng hoặc cùng cột và chỉ giữ lại biến chung.
- ❖ Ví dụ, nhóm các ô kề nhau trên Hình 2.4b như Hình 2.5.



Hình 2.5: Nhóm các ô trên cùng hàng hoặc cùng cột.

Bản Đồ Karnaugh

❖ Vậy biểu thức:

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} = \bar{A}B + B\bar{C}$$

❖ Bài tập tại lớp: Chứng minh kết quả của biểu thức này bằng đại số Boolean.

Mạch Tổ Hợp

- ❖ Mạch số được chia làm hai loại: Mạch tổ hợp (*combinational circuit*) và mạch tuần tự (*sequential circuit*).
- ❖ Mạch tổ hợp bao gồm n ngõ vào và m ngõ ra.
- ❖ Các mạch cộng và trừ được trình bày sau đây là các mạch tổ hợp.

Mạch Cộng Bán Phần

- ❖ Mạch cộng bán phần (*half adder* - HA) là mạch cộng 1 bit không có nhớ. Chúng ta có thể xây dựng mạch cộng này như sau:
 - Gọi A, B là các số nhị phân 1 bit cần cộng, S là bit tổng và C là bit nhớ. Ta có:

A	0	0	1	1
	+	+	+	+
B	0	1	0	1
	<hr/>	<hr/>	<hr/>	<hr/>
S	0	1	1	0
C	0	0	0	1

Mạch Cộng Bán Phần

- **Bảng sự thật:**

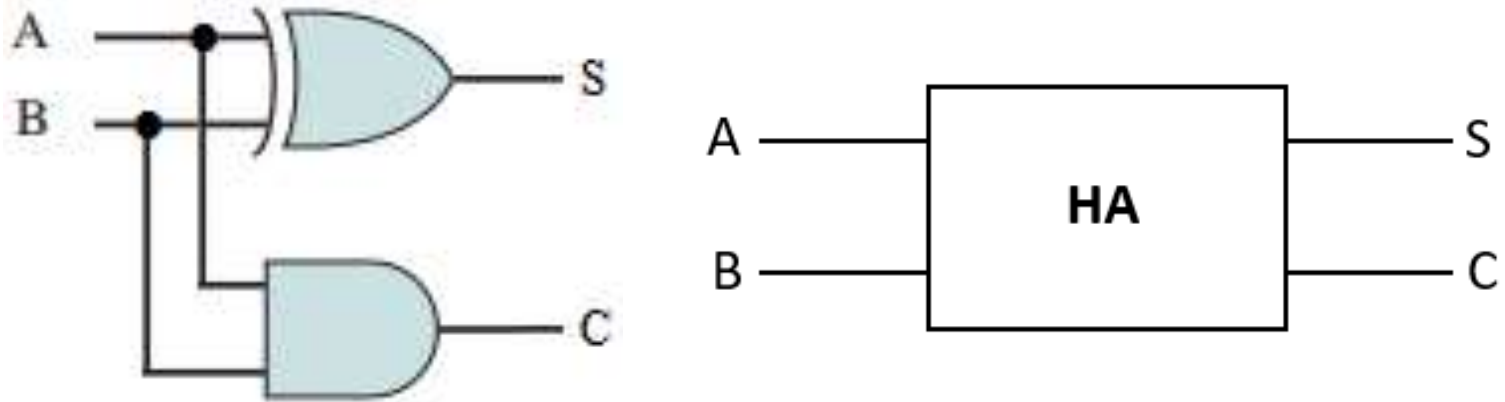
Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- **Từ bảng sự thật trên, ta có hệ thức và mạch logic sau:**

$$S = \bar{A}B + A\bar{B} = A \oplus B$$

$$C = AB$$

Mạch Cộng Bán Phần



Hình 2.6: Mạch cộng bán phần.

Mạch Cộng Toàn Phần

- ❖ Để xây dựng mạch cộng nhiều bit, trước hết chúng ta phải xây dựng mạch cộng 1 bit có nhớ được gọi là mạch cộng toàn phần (*full adder* - FA).
- ❖ Mạch cộng toàn phần sẽ tính:
 - $S = A + B + C_0$
 - C là bit nhớ của phép tính tổng S.

Mạch Cộng Toàn Phần

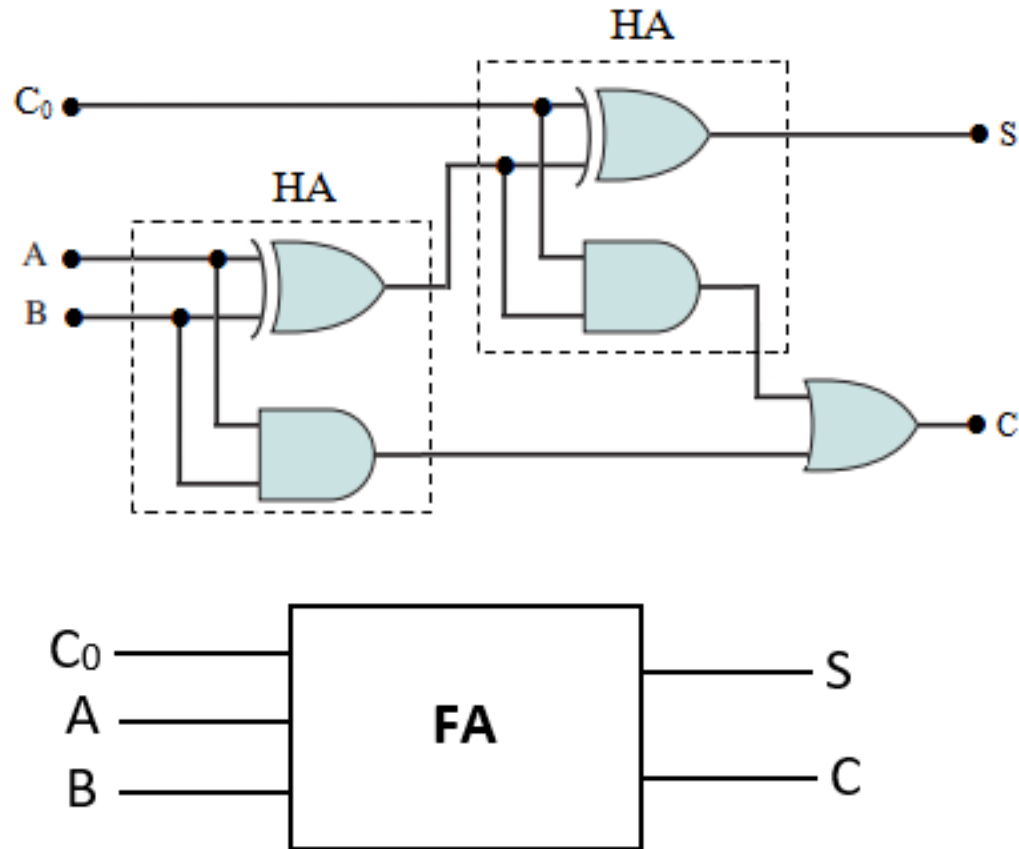
■ Bảng sự thật:

Input			Output	
C_0	A	B	S	C
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

Mạch Cộng Toàn Phần

- Từ bảng sự thật, để có được biểu thức logic tối giản của S và C, sử dụng một trong hai cách sau:
 - Cách 1: Sử dụng đại số Boolean.
$$S = C_0 \oplus (A \oplus B)$$
$$C = AB + C_0 (A + B)$$
 - Cách 2: Sử dụng bản đồ Karnaugh.
- Bài tập tại lớp: Sinh viên hãy chứng minh biểu thức của cách 1 và thực hiện cách 2 để cho thấy chúng có cùng một kết quả.

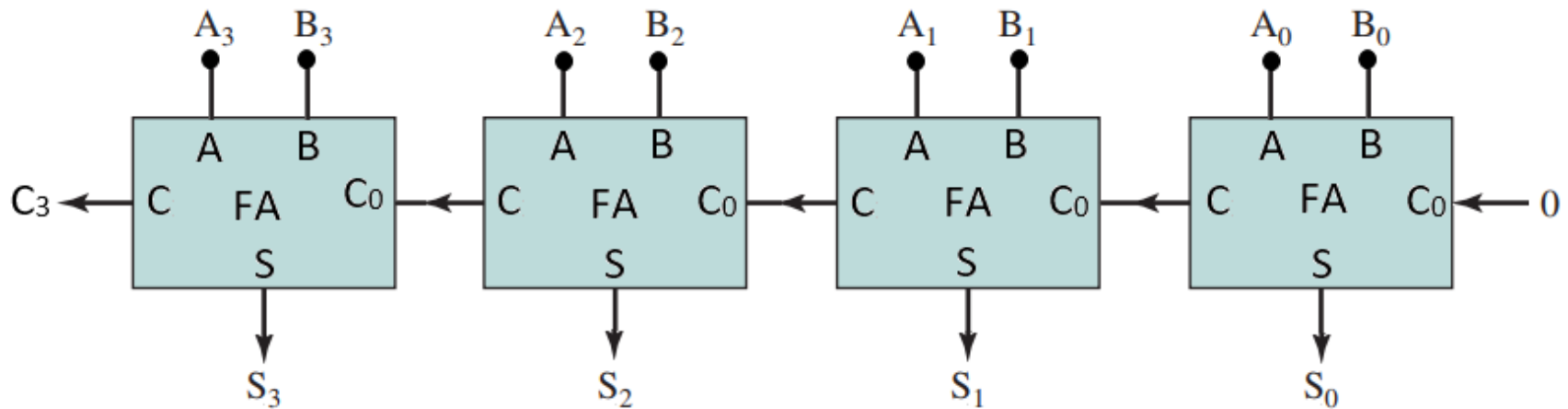
Mạch Cộng Toàn Phần



Hình 2.7: Mạch cộng toàn phần 1 bit.

Mạch Cộng Nhiều Bit

- ❖ Phép cộng của bit LSB có thể sử dụng mạch cộng bán phần hoặc mạch cộng toàn phần với ngõ vào bit nhớ $C_0 = 0$.
- ❖ Phép cộng của các bit cao hơn phải là mạch cộng toàn phần.



Hình 2.8: Mạch cộng 4 bit.

Mạch Trừ Bán Phần

❖ Mạch trừ bán phần (*half subtractor* - HS) là mạch trừ 1 bit không có mượn.

- Gọi A, B là các số nhị phân 1 bit cần thực hiện phép trừ, D (*difference*) là bit hiệu và Br (*borrow*) là bit mượn. Ta có:

$$\begin{array}{r} A \quad 0 \quad 0 \quad 1 \quad 1 \\ \quad - \quad - \quad - \quad - \\ B \quad 0 \quad 1 \quad 0 \quad 1 \\ \hline D \quad 0 \quad 1 \quad 1 \quad 0 \\ Br \quad 0 \quad 1 \quad 0 \quad 0 \end{array}$$

- **Bảng sự thật:**

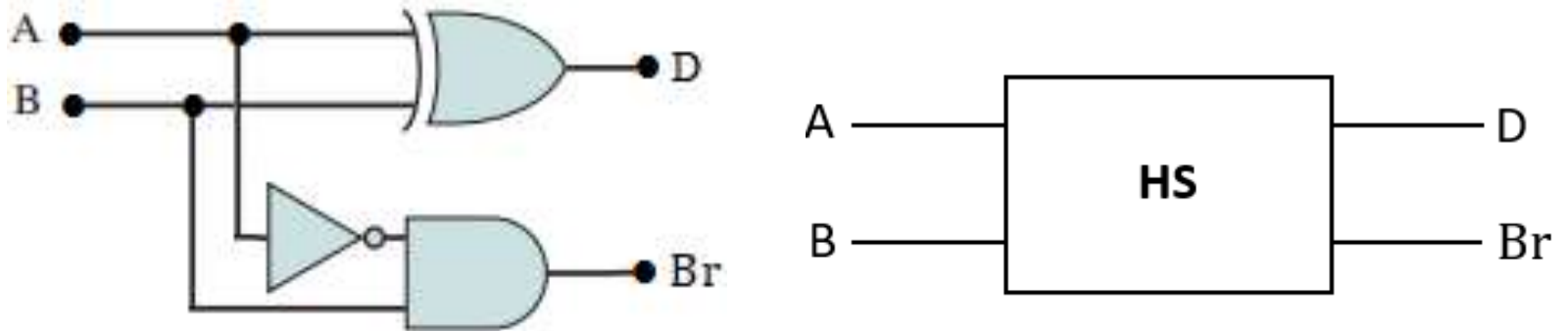
Input		Output	
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Mạch Trừ Bán Phần

- Hệ thức và mạch logic:

$$D = A \oplus B$$

$$Br = \bar{A}B$$



Hình 2.9: Mạch trừ bán phần.

Mạch Trừ Toàn Phần

- ❖ Mạch trừ 1 bit có mượn được gọi là mạch trừ toàn phần (*full subtractor* - FS).
- ❖ Mạch trừ toàn phần sẽ tính:
 - $D = A - (B + Br_0)$.
 - Br_0 là bit mượn của phép tính hiệu D.
 - Bảng sự thật:

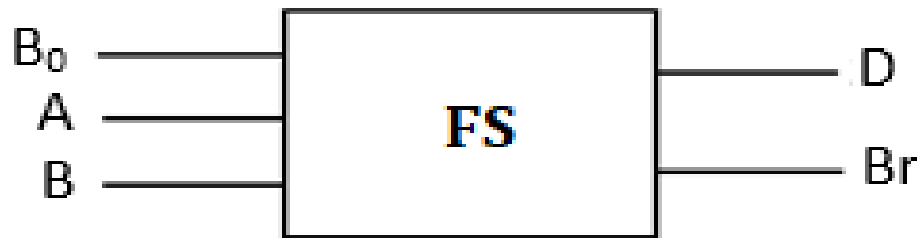
Mạch Trừ Toàn Phần

Input			Output	
Br_0	A	B	D	Br
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

- Hệ thức và mạch logic:

Mạch Trừ Toàn Phần

$$D = Br_0 \oplus A \oplus B$$
$$Br = \overline{A}B + Br_0 (\overline{A \oplus B})$$



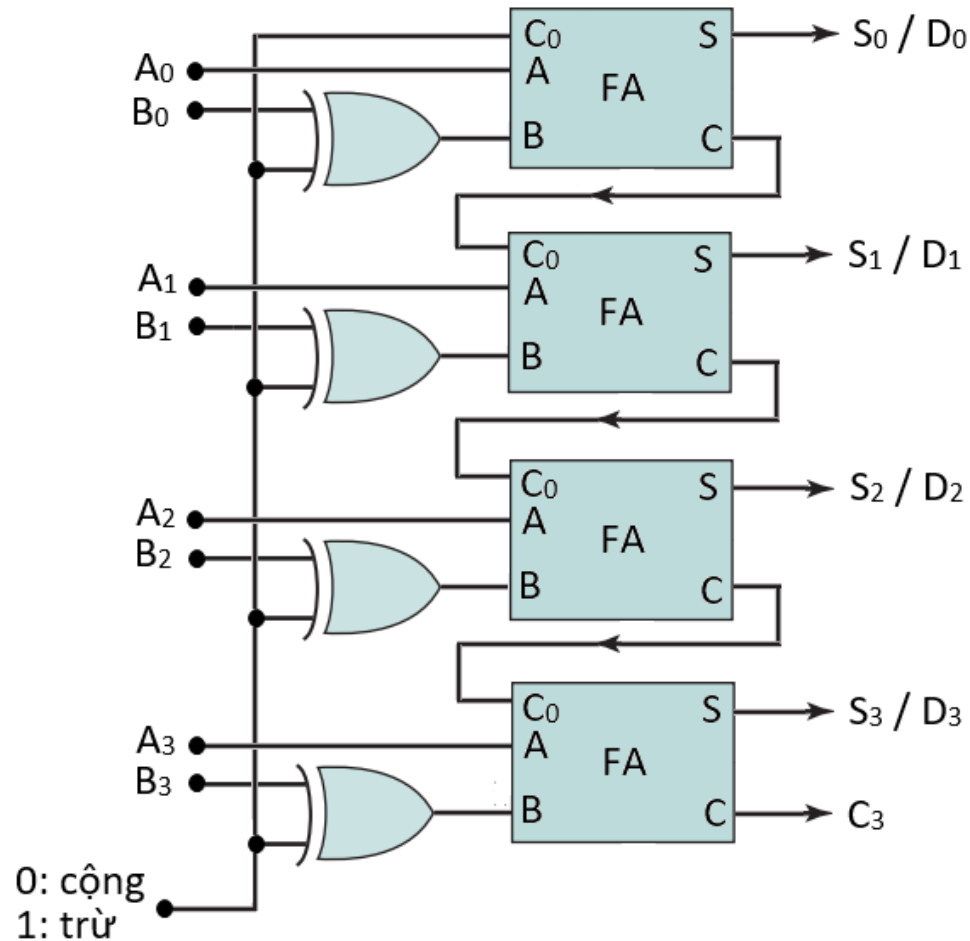
Hình 2.10: Mạch trừ toàn phần 1 bit.

- Bài tập tại lớp: Sinh viên vẽ mạch trừ toàn phần 1 bit dựa vào mạch trừ bán phần.

Mạch Cộng và Trừ Dùng Số Bù 2

- ❖ Thông thường, phép trừ hai số nhị phân nhiều bit $A - B$ được đổi thành phép cộng $A + (-B)$.
- ❖ Để biểu diễn số âm $(-B)$, người ta sử dụng số bù 1 hoặc bù 2 của B .
- ❖ Hình sau là mạch cộng/trừ 4 bit dùng số bù 2.

Mạch Cộng và Trừ Dùng Số Bù 2



Hình 2.11: Mạch cộng/trừ 4 bit dùng số bù 2.

Mạch Tuần Tự

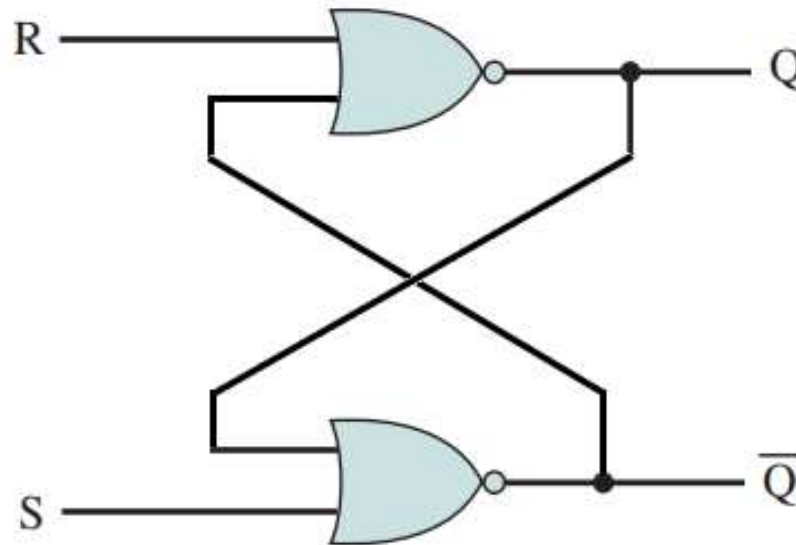
- ❖ Trong mạch tuần tự, trạng thái các ngõ ra không những phụ thuộc vào trạng thái các ngõ vào mà còn phụ thuộc vào trạng thái trước đó của các ngõ ra.
- ❖ Mạch tuần tự có tính nhớ và tính đồng bộ.

Mạch Chốt RS và Flip-Flop RS

- ❖ Dạng đơn giản nhất của mạch tuần tự là mạch chốt (mạch cài) và flip-flop (mạch lật).
- ❖ Có nhiều loại mạch chốt và flip-flop, chúng có hai thuộc tính:
 - Có hai trạng thái ổn định và duy trì trạng thái đó khi trạng thái ngõ vào chưa thay đổi. Do đó, chúng có chức năng như là 1 bit nhớ.
 - Có hai ngõ ra có giá trị ngược nhau, thường được gọi là Q và \bar{Q} . Kết quả được tính theo ngõ Q .

Mạch Chốt RS

- ❖ Hình 2.12 là mạch chốt RS với hai cổng NOR được mắc hồi tiếp với nhau (có thể dùng cổng NAND).
- ❖ Mạch này có hai ngõ vào là R và S, hai ngõ ra là Q và \bar{Q} .



Hình 2.12: Mạch chốt RS.

Mạch Chốt RS

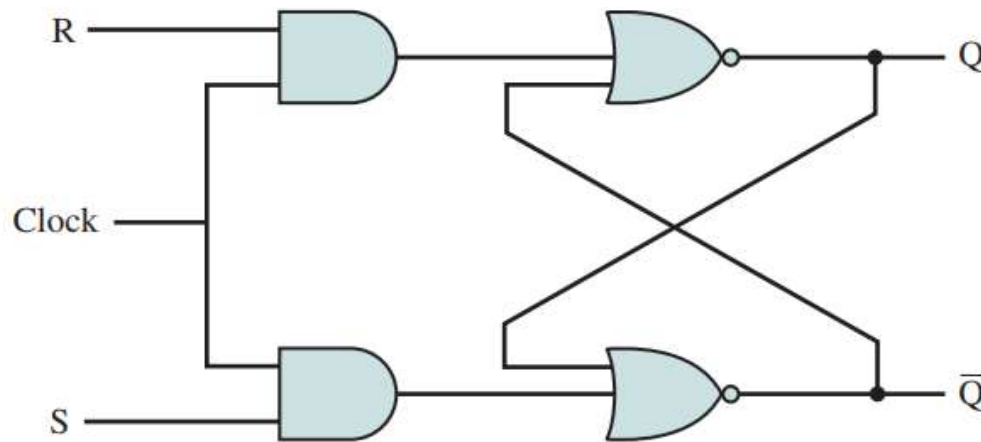
❖ Trong bảng 2.2 các ngõ vào $S = 1, R = 1$ thì không được phép.

Input		Output
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Cấm

Bảng 2.2: Bảng sự thật của mạch chốt RS.

Flip-Flop RS

- ❖ Nhược điểm của mạch chốt RS: Sinh viên tự nhận xét để tìm nhược điểm.
- ❖ Để khắc phục các nhược điểm của mạch chốt RS, sử dụng flip-flop RS.



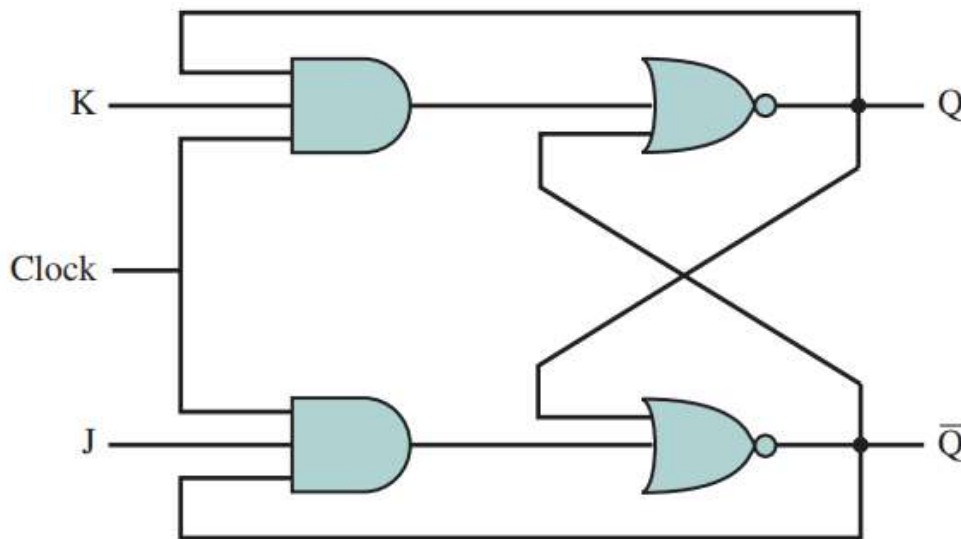
Hình 2.13: Flip-flop RS.

Input			Output
S	R	CLK	Q_{n+1}
0	0	↑	Q_n
0	1	↑	0
1	0	↑	1
1	1	↑	Cấm

Bảng 2.3: Bảng sự thật của flip-flop RS.

Flip-Flop JK

❖ Flip-flop JK khắc phục được trạng thái cấm của flip-flop RS.



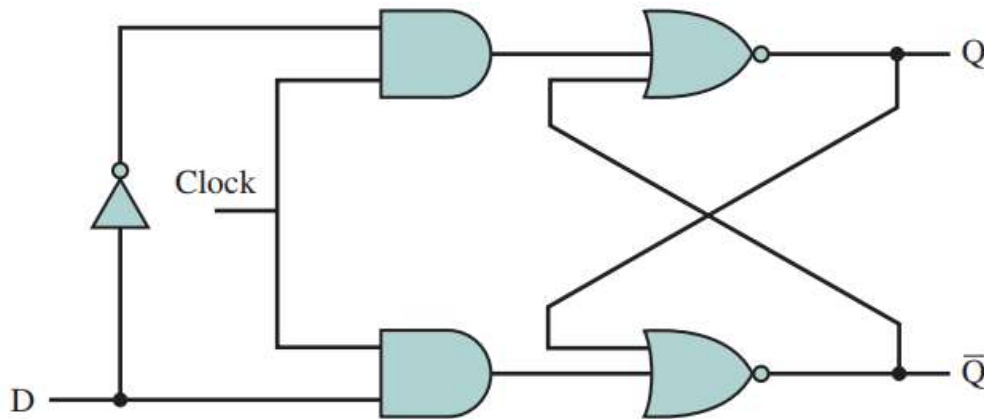
Hình 2.14: Flip-flop JK.

Input			Output
J	K	CLK	Q_{n+1}
0	0	↑	Q_n
0	1	↑	0
1	0	↑	1
1	1	↑	$\overline{Q_n}$

Bảng 2.4: Bảng sự thật của flip-flop JK.

Flip-Flop D

- ❖ Khi nối các ngõ vào của flip-flop RS (hoặc JK), chúng ta được flip-flop D.
- ❖ Ngõ ra của flip-flop D luôn bằng với giá trị đã gán trước khi có xung nhịp cho ngõ vào.



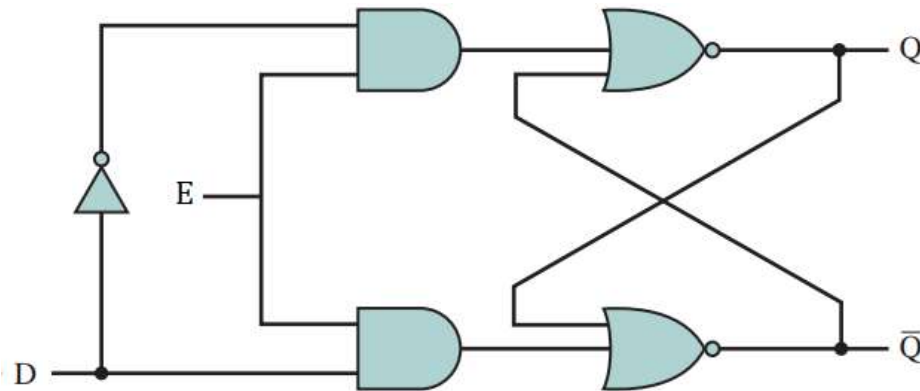
Hình 2.15: Flip-flop D.

Input		Output
D	CLK	Q_{n+1}
0	↑	0
1	↑	1

Bảng 2.5: Bảng sự thật của flip-flop D.

Chốt D

- ❖ Trong flip-flop D, nếu thay ngõ vào đồng hồ bằng ngõ vào cho phép E (*enable*) tác động ở mức cao, chúng ta được mạch chốt D.



Hình 2.16: Chốt D.

Input		Output
D	E	Q_{n+1}
0	1	0
1	1	1

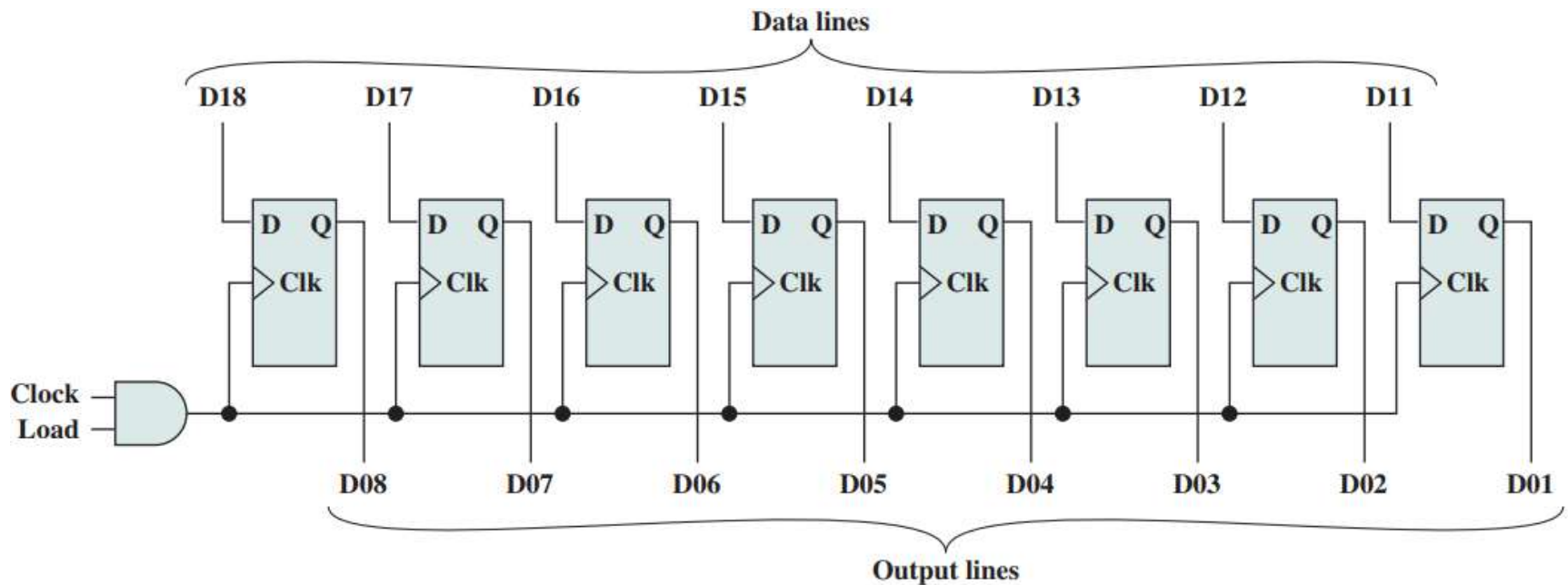
Bảng 2.6: Bảng sự thật của chốt D.

Thanh Ghi

- ❖ Thanh ghi là mạch số được dùng trong CPU để lưu trữ nhiều bit dữ liệu. Có hai loại thanh ghi thường sử dụng: Thanh ghi song song (*parallel register*) và thanh ghi dịch (*shift register*).

Thanh Ghi Song Song

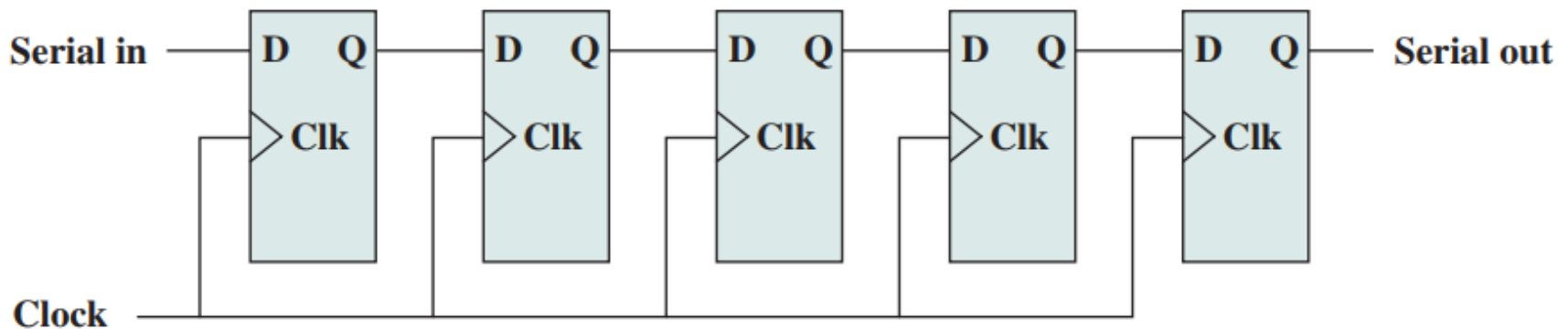
- ❖ Thanh ghi song song chứa các bit nhớ có thể đọc hoặc ghi đồng thời.



Hình 2.17: Thanh ghi song song 8 bit.

Thanh Ghi Dịch

- ❖ Thanh ghi dịch nhận hoặc chuyển thông tin tuần tự.
- ❖ Dữ liệu vào từ flip-flop bên trái nhất. Với mỗi chu kỳ xung nhịp, dữ liệu được dịch sang phải một vị trí và bit bên phải nhất được chuyển ra ngoài.



Hình 2.18: Thanh ghi dịch 5 bit.

Bài Tập

1. Lập bảng sự thật cho các biểu thức Boolean sau:

a) $ABC + \overline{A}\overline{B}\overline{C}$

b) $ABC + \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C$

c) $A(\overline{B}C + B\overline{C})$

d) $(A + B)(A + C)(\overline{A} + \overline{B})$

2. Đơn giản hoá các biểu thức sau theo luật giao hoán:

a) $A.\overline{B} + \overline{B}.A + C.D.E + \overline{C}.D.E + E.\overline{C}.D$

b) $A.B + A.C + B.A$

c) $(L.M.N)(A.B)(C.D.E)(M.N.L)$

d) $F.(K + R) + S.V + W.\overline{X} + V.S + \overline{X}.W + (R + K).F$

Bài Tập

3. Áp dụng định luật DeMorgan cho các phương trình sau:

a) $F = \overline{V + A + L}$

b) $F = \overline{A} + \overline{B} + \overline{C} + \overline{D}$

4. Đơn giản hoá các biểu thức sau:

a) $A = S.T + V.W + R.S.T$

b) $A = T.U.V + X.Y + Y$

c) $A = F.(E + F + G)$

d) $A = (P.Q + R + S.T)T.S$

e) $A = \overline{\overline{D} . \overline{D} . E}$

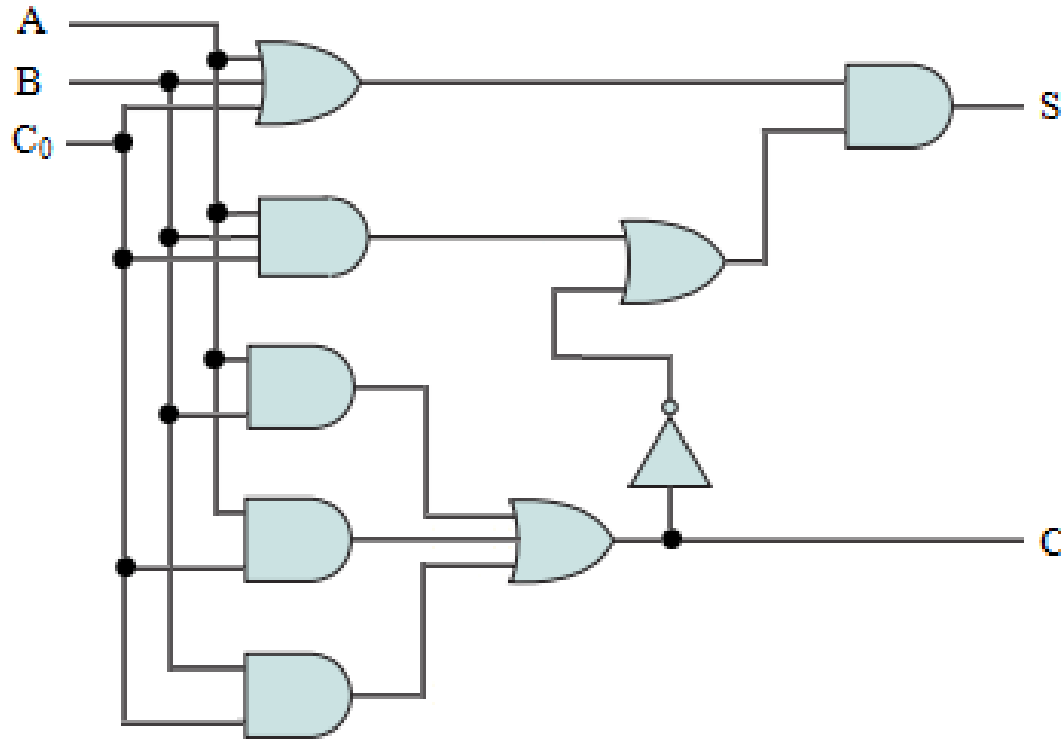
f) $A = Y.(W + X + \overline{\overline{Y} + \overline{Z}}).Z$

g) $A = (B.E + C + F).C$

Bài Tập

5. Tạo phép toán XOR từ các phép toán Boolean cơ bản AND, OR và NOT.
6. Cho một cổng NOR và các cổng NOT, vẽ sơ đồ logic để thực hiện chức năng cổng AND ba ngõ vào.
7. Viết biểu thức Boolean cho cổng NAND bốn ngõ vào.
8. Viết biểu thức của S và C theo A, B, C_0 và cho biết ý nghĩa của mạch sau:

Bài Tập



9. Vẽ mạch chốt RS bằng các cổng NAND thay vì các cổng NOR và lập bảng sự thật cho mạch chốt này.