

## **3.2\_Mạch tổ hợp**

# **Arithmetic Circuits**



# Nội dung

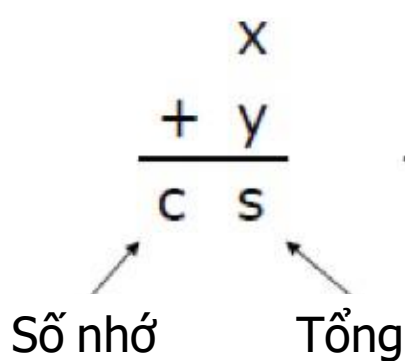
1. **Mạch cộng (Carry Ripple (CR)Adder)**
2. **Mạch cộng nhìn trước số nhớ - (Carry Look- Ahead (CLA)Adder)**
3. **Mạch cộng/ mạch trừ**
4. **Đơn vị tính toán luận lý (Arithmetic Logic Unit)**



# 1. Mạch cộng Carry Ripple (CR)

# Mạch cộng bán phần (Half Adder)

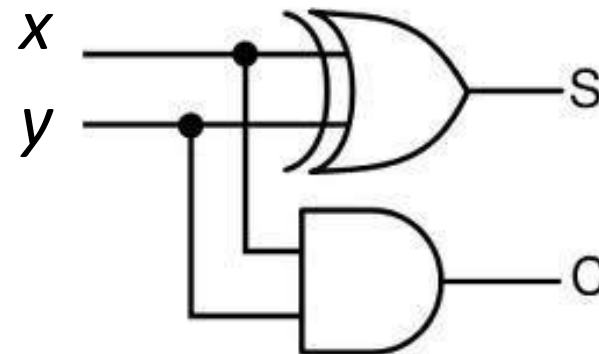
- Cộng 2 số 1 bit có 4 trường hợp



$x$	$0$	$0$	$1$	$1$
$+ y$	$+ 0$	$+ 1$	$+ 0$	$+ 1$
$\hline$	$\hline$	$\hline$	$\hline$	$\hline$
$c \quad s$	$0 \quad 0$	$0 \quad 1$	$0 \quad 1$	$1 \quad 0$

$x$	$y$	$c$	$s$
$0$	$0$	$0$	$0$
$0$	$1$	$0$	$1$
$1$	$0$	$0$	$1$
$1$	$1$	$1$	$0$

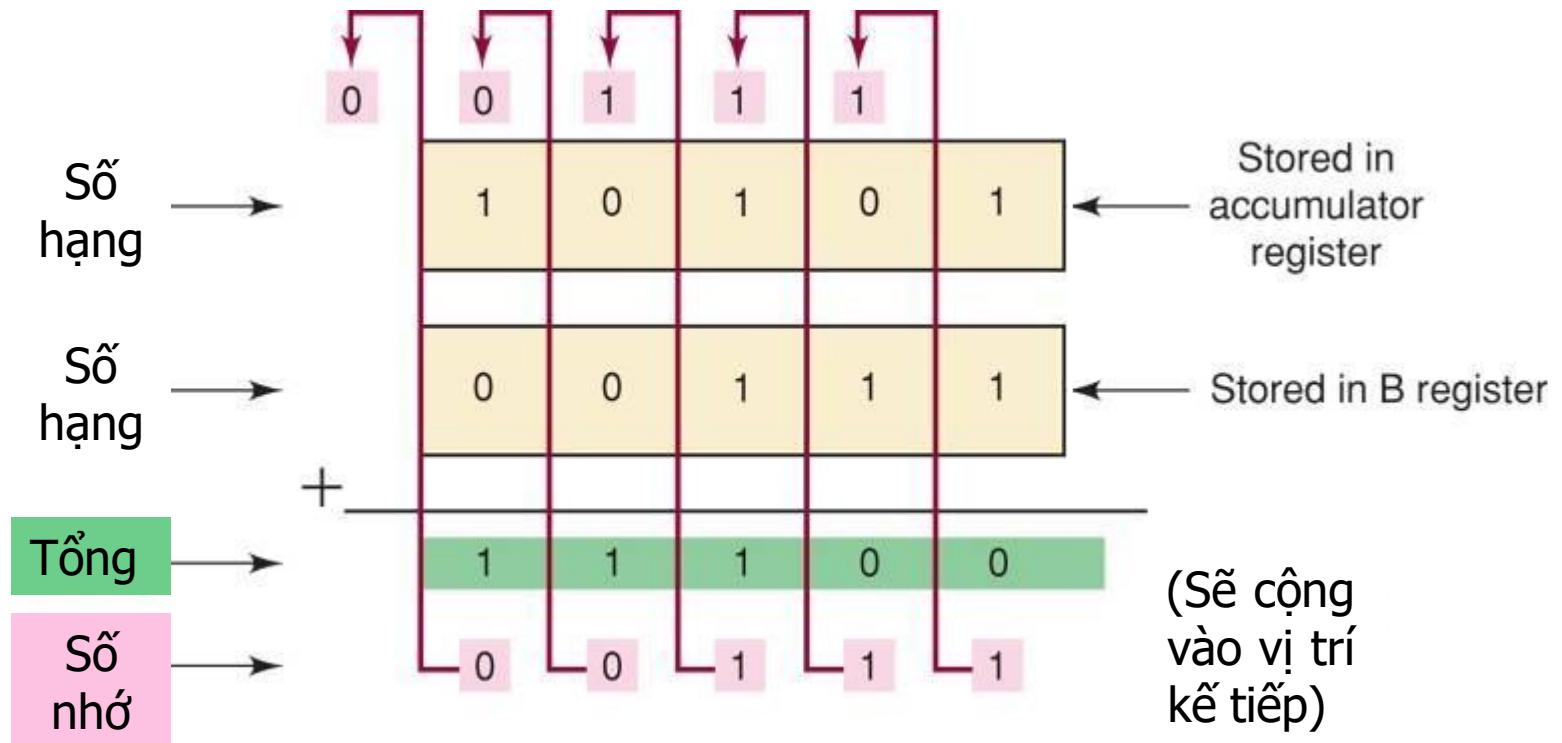
Mạch cộng 1 bit có tổng và số nhớ như thế này được gọi là mạch cộng bán phần (**HA**)



Sơ đồ mạch

# Mạch cộng nhị phân song song

- Cộng những số có 2 hoặc nhiều bit
  - Cộng từng cặp bit bình thường
  - Nhưng ở vị trí cặp bit  $i$ , có thể có carry-in từ bit  $i-1$



# Thiết kế một bộ cộng toàn phần (Full Adder)



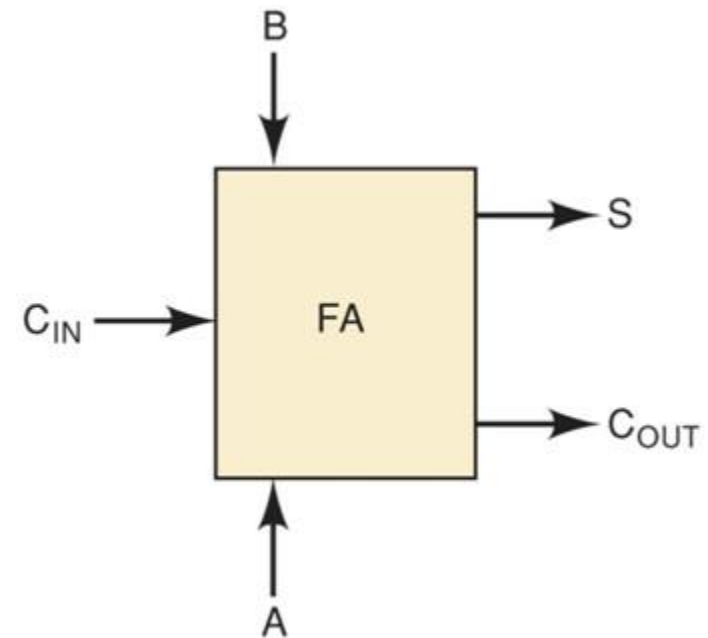
## Bộ cộng toàn phần (**FA**)

- **3 ngõ vào** (2 ngõ vào cho 2 số 1-bit cần tính tổng, và 1 ngõ vào cho số nhớ đầu vào (**carry-in**))
- **2 ngõ ra** (1 ngõ ra cho tổng và 1 cho số nhớ đầu ra (**carry-out**))

# Thiết kế một bộ cộng toàn phần (Full Adder)

## Bảng sự thật

Augend bit input	Addend bit input	Carry bit input	Sum bit output	Carry bit output
A	B	$C_{IN}$	S	$C_{OUT}$
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



Biểu tượng

# Thiết kế một bộ cộng toàn phần (Full Adder)

Augend bit input	Addend bit input	Carry bit input	Sum bit output	Carry bit output
A	B	$C_{IN}$	S	$C_{OUT}$
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

Bảng sự thật

$c \backslash xy$	00	01	11	10
0		1		1
1	1		1	

$$S_i = x_i \oplus y_i \oplus c_i$$

$c \backslash xy$	00	01	11	10
0			1	
1		1	1	1

$$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$C_i = C_{IN} \quad C_{i+1} = C_{OUT}$$

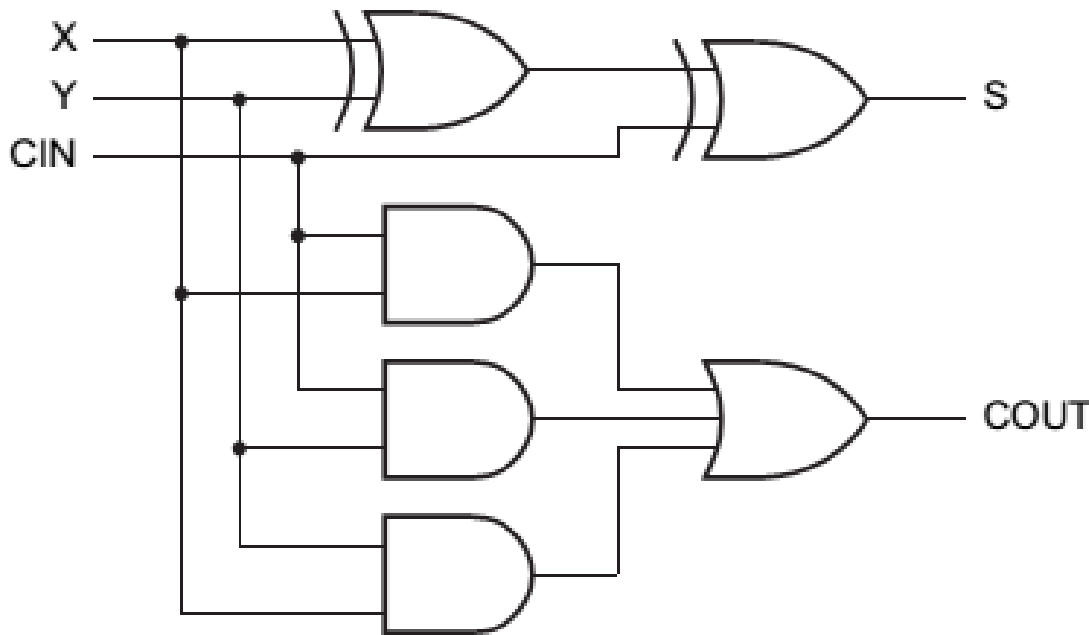


# Thiết kế một bộ cộng toàn phần (Full Adder)

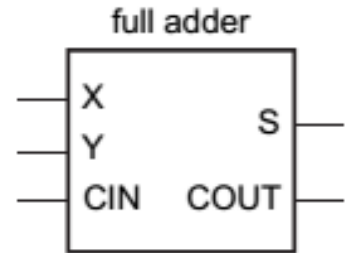
$$S_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i \quad c_i = c_{IN}$$

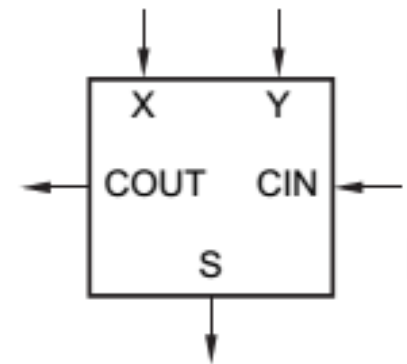
$$c_{i+1} = c_{OUT}$$



Sơ đồ mạch



Biểu tượng

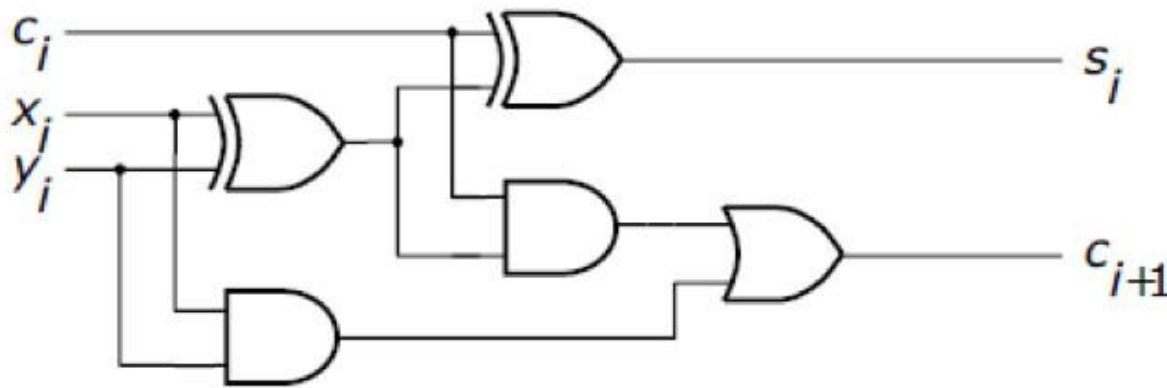


Biểu tượng khác

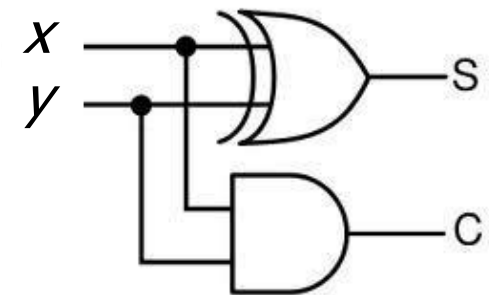
# Thiết kế một bộ cộng toàn phần (Full Adder)

- Sử dụng lại HA

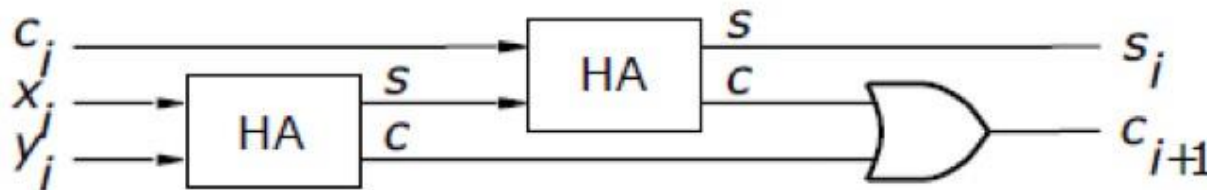
$$S_i = x_i \oplus y_i \oplus c_i \quad c_{i+1} = x_i y_i + c_i (x_i \oplus y_i)$$



Sơ đồ chi tiết



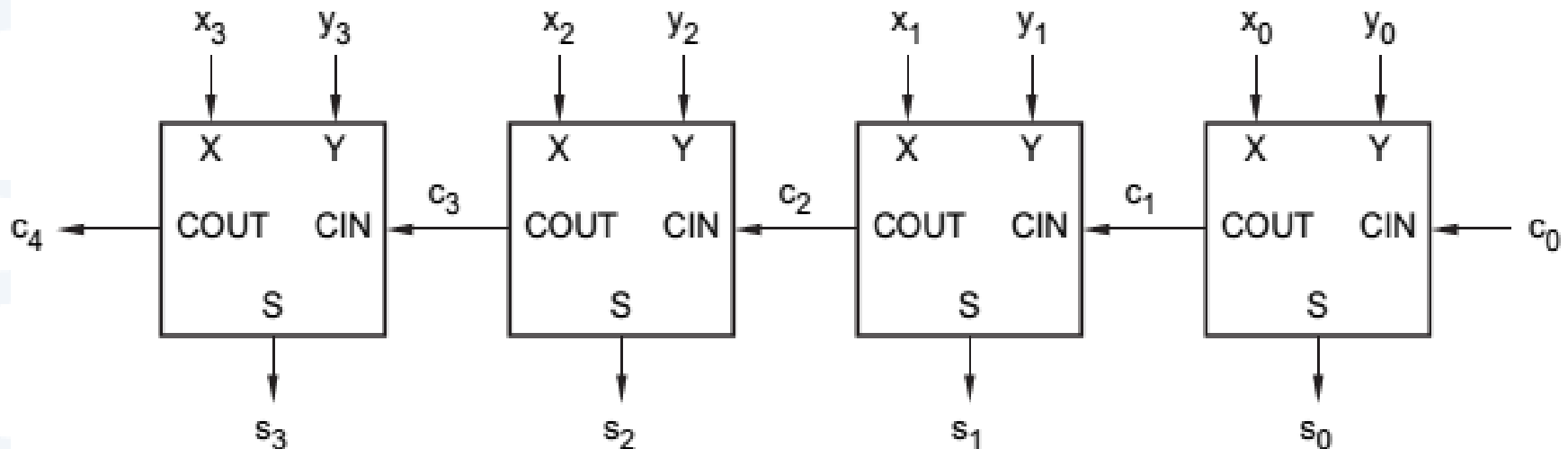
Sơ đồ mạch HA



Sơ đồ khối

# Mạch cộng Carry Ripple (CR)

- Sơ đồ biểu diễn mạch cộng 4 bit song song sử dụng full adder



# Mạch cộng Carry Ripple

- Mạch FA bắt đầu với việc cộng các cặp bit từ LSB đến MSB
  - Nếu carry xuất hiện ở vị trí bit  $i$ , nó được cộng thêm vào phép cộng ở vị trí bit thứ  $i+1$
- Việc kết hợp như vậy thường được gọi là mạch cộng carry-ripple
  - vì carry được “ripple” từ FA này sang các FA kế tiếp
  - Tốc độ phép cộng bị giới hạn bởi quá trình truyền số nhớ

# Mạch cộng Carry Ripple

- Mỗi FA có một khoảng trễ (delay), giả sử là
- Độ trễ phụ thuộc vào số lượng bit
  - Carry-out ở FA đầu tiên  $C_1$  có được sau
  - Carry-out ở FA đầu tiên  $C_2$  có được sau

$\Rightarrow C_n$  được tính toán sau
- Mô hình carry look ahead (CLA) thường được sử dụng để cải thiện tốc độ



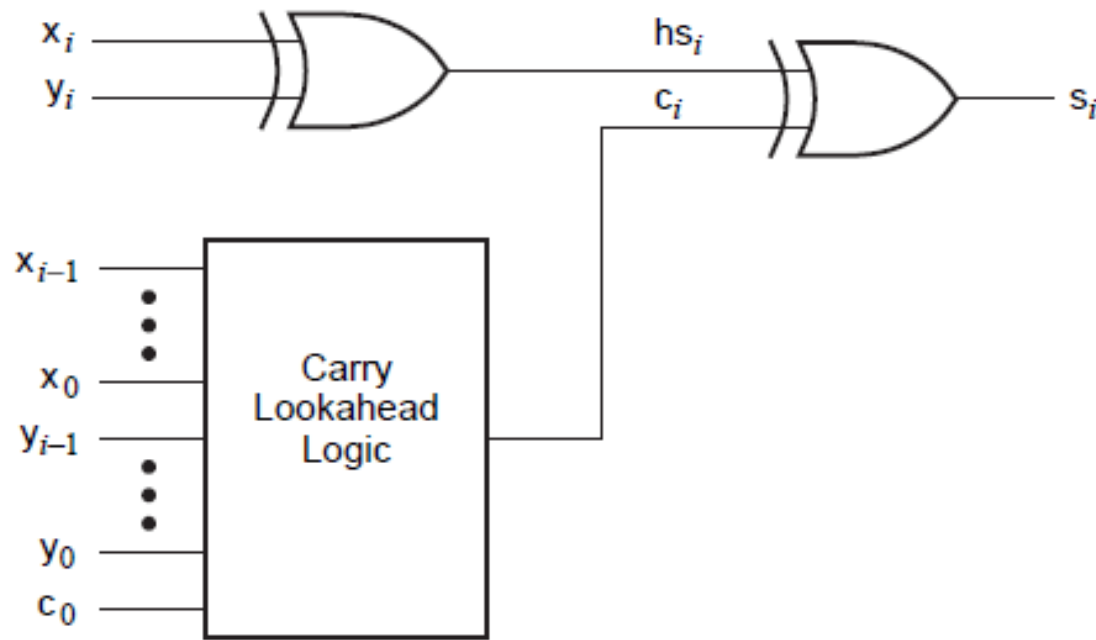
## 2. Mạch cộng nhìn trước số nhớ Carry Look-Ahead (CLA) Adder

# Hiệu năng

- Tốc độ của mạch bị giới hạn bởi độ trễ lớn nhất dọc theo đường nối trong mạch
  - Độ trễ lớn nhất được gọi là **critical-path-delay**
  - Đường nối gây ra độ trễ đó gọi là **critical path**

# Carry Look-Ahead (CLA) Adder

- Cải thiện tốc độ mạch cộng
  - Xác định nhanh giá trị carry-out ở mỗi lần cộng với carry-in ở lần cộng trước sẽ có giá trị 0 hay 1



- Mục tiêu: giảm critical-path-delay



# Carry Look-Ahead (CLA) Adder

- Hàm xác định carry-out ở lần cộng thứ i

$$C_{i+1} = x_i y_i + x_i C_i + y_i C_i = x_i y_i + (x_i + y_i) C_i$$

- Đặt  $g_i = x_i y_i$  và  $p_i = x_i + y_i \Rightarrow C_{i+1} = g_i + p_i C_i$

- $g_i = 1$  khi cả  $x_i$  và  $y_i$  đều bằng 1, không quan tâm  $c_i$ 
  - ❖  $g$  được gọi là **hàm generate**, carry-out luôn được generate ra
- $p_i = 1$  khi  $x_i = 1$  hoặc  $y_i = 1$ ; carry-out =  $c_i$ 
  - ❖  $p$  được gọi là **hàm propagate**, vì carry-in = 1 được propagate (truyền) ở lần cộng thứ i

# Carry Look-Ahead (CLA) Adder

- Xác định carry-out của mạch cộng n bit

$$C_n = g_{n-1} + p_{n-1}C_{n-1}$$

Mà  $C_{n-1} = g_{n-2} + p_{n-2}C_{n-2}$

$$C_n = g_{n-1} + p_{n-1}(g_{n-2} + p_{n-2}C_{n-2})$$

$$C_n = g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}C_{n-2}$$

- Tiếp tục khai triển đến lần cộng đầu tiên

$$C_n = g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}g_{n-3} + \dots + p_{n-1}p_{n-2} \dots p_1g_0 + p_{n-1}p_{n-2} \dots p_1p_0C_0$$

# Carry Look-Ahead (CLA) Adder

Carry generated in stage  $n-2$ ,  
and propagated through the  
remaining stages

Carry generated in stage 0,  
and propagated through  
the remaining stages

$$c_n = \underbrace{g_{n-1}}_{\substack{\uparrow \\ \text{Carry generated} \\ \text{in last stage}}} + \underbrace{p_{n-1}g_{n-2}}_{\substack{\uparrow \\ \text{Carry generated in stage } n-3, \\ \text{and propagated through the} \\ \text{remaining stages}}} + \underbrace{p_{n-1}p_{n-2}g_{n-3}}_{\substack{\uparrow \\ \text{Carry generated in stage } n-3, \\ \text{and propagated through the} \\ \text{remaining stages}}} + \dots + \underbrace{p_{n-1}p_{n-2}\dots p_1g_0}_{\substack{\swarrow \\ \text{Carry generated in stage 0,} \\ \text{and propagated through} \\ \text{the remaining stages}}} + \underbrace{p_{n-1}p_{n-2}\dots p_1p_0c_0}_{\substack{\uparrow \\ \text{Input carry } c_0 \\ \text{propagated through} \\ \text{all stages}}}$$

Carry generated  
in last stage

Carry generated in stage  $n-3$ ,  
and propagated through the  
remaining stages

Input carry  $c_0$   
propagated through  
all stages

# Carry Look-Ahead (CLA) Adder

- Ví dụ: Trường hợp cộng 4 bit

$$C_1 = G_0 + P_0.C_0$$

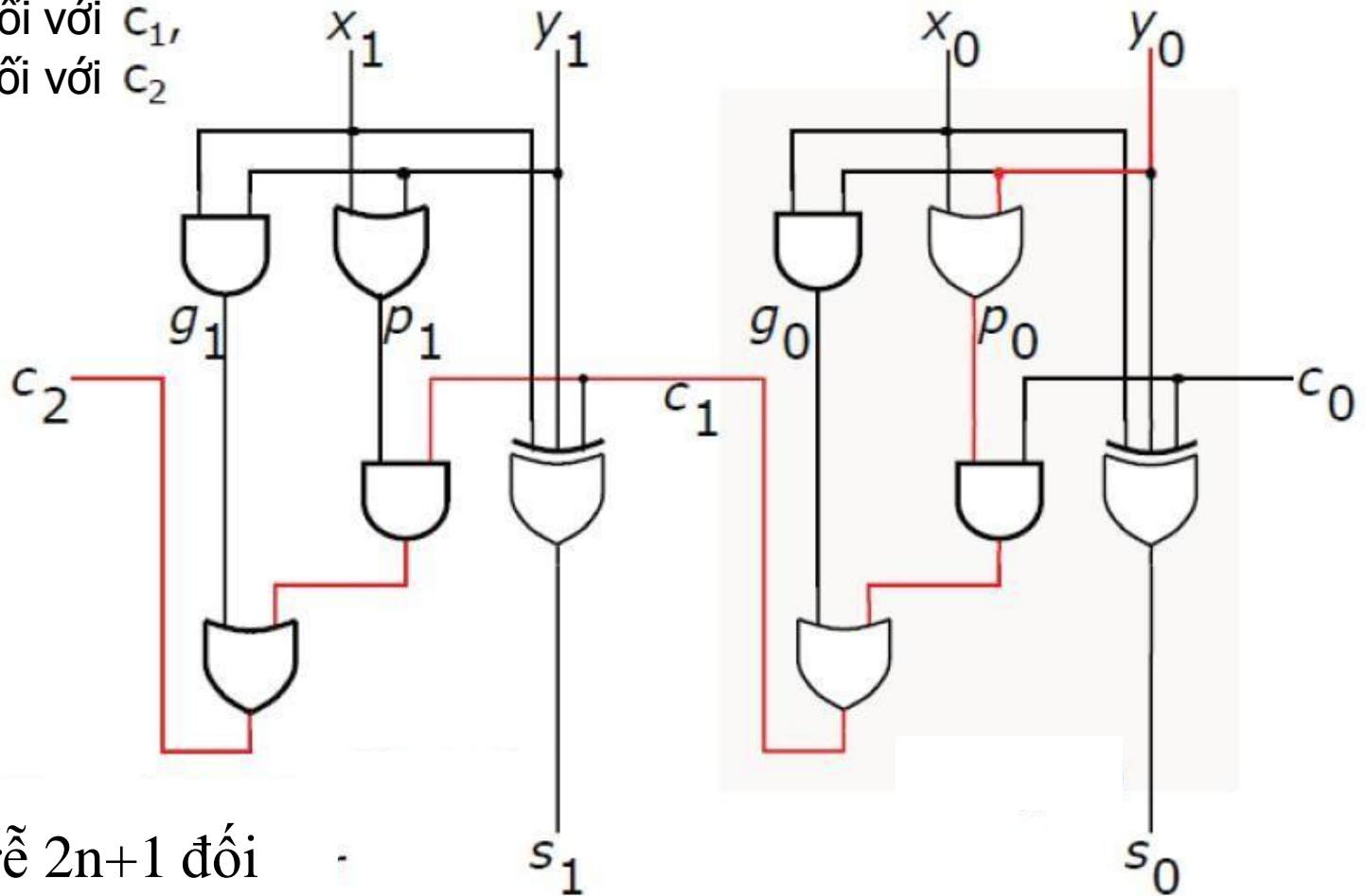
$$C_2 = G_1 + P_1.G_0 + P_1.P_0.C_0$$

$$C_3 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.C_0$$

$$C_4 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.C_0$$

# Mạch cộng CR - critical path

Độ trễ 3 cổng đối với  $c_1$ ,  
Độ trễ 5 cổng đối với  $c_2$



Tóm lại, Độ trễ  $2n+1$  đối  
với mạch cộng Carry  
Ripple n-bit

# Mạch cộng CLA - critical path

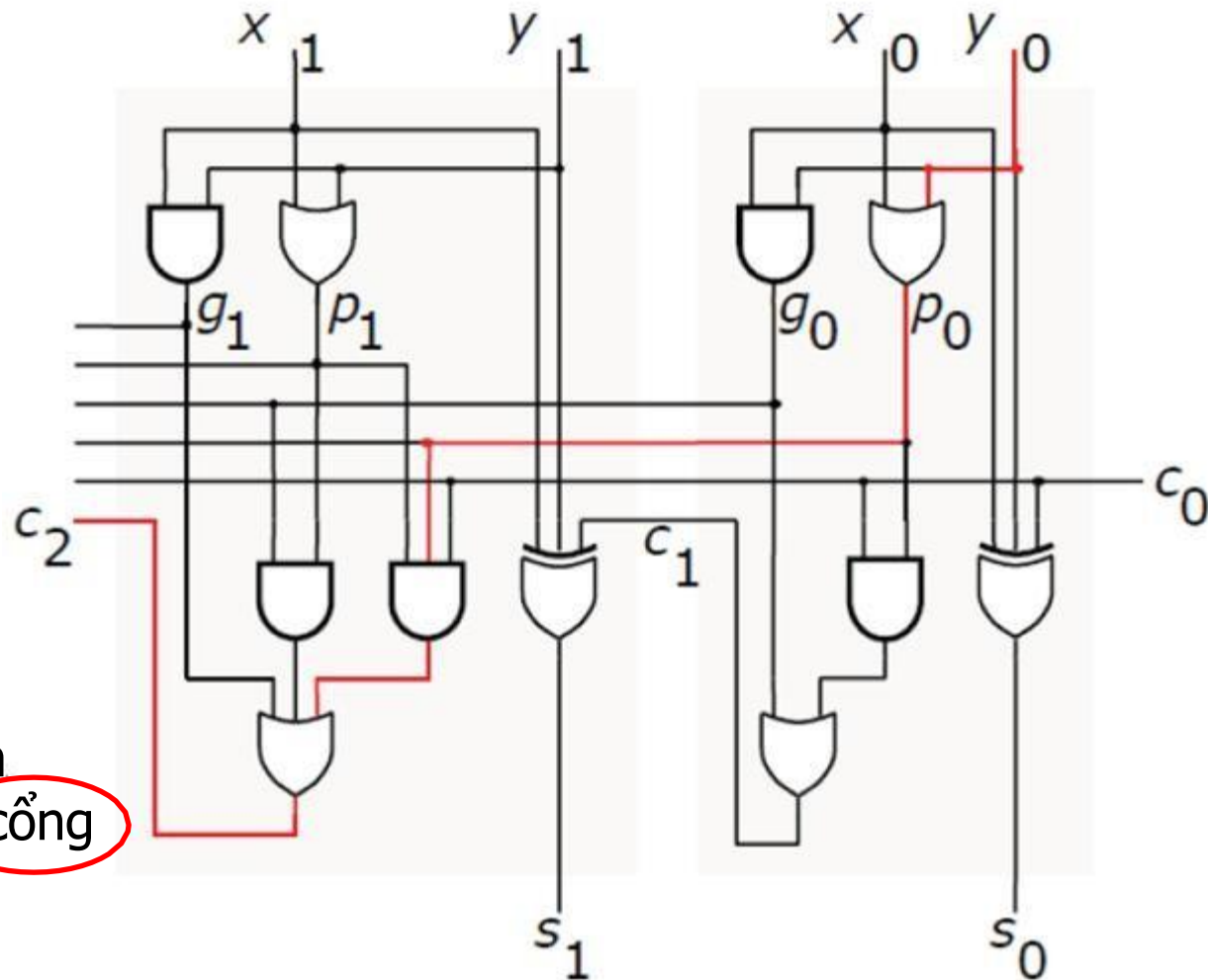
Độ trễ 3 cổng đối với  $c_1$ ,  
 Độ trễ 3 cổng đối với  $c_2$   
 Độ trễ 3 cổng đối với  $c_n$

$$C_1 = G_0 + P_0.C_0$$

$$C_2 = G_1 + P_1.G_0 + P_1.P_0.C_0$$

Độ trễ tổng cộng cho mạch cộng CLA n-bit là độ trễ **4 cổng**

- $g_i, p_i$ : độ trễ 1 cổng
- $C_i$ : độ trễ 2 cổng
- Độ trễ 1 cổng còn lại là do tính tổng  $s$



# Giới hạn của CLA

- Biểu thức tính carry trong mạch cộng CLA

$$C_n = g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}g_{n-3} + \dots + p_{n-1}p_{n-2}\dots p_1g_0 + p_{n-1}p_{n-2}\dots p_1p_0C_0$$

CLA là giải pháp tốc độ cao (2 level AND-OR)

- Độ phức tạp tăng lên nhanh chóng khi n lớn
  - Hierarchical approach để giảm độ phức tạp
- Fan-in issue có thể hạn chế tốc độ của CLA
  - Thiết bị có vấn đề với fan-in issue (vd: FPGA) thường kèm mạch riêng để hiện thực mạch cộng nhanh

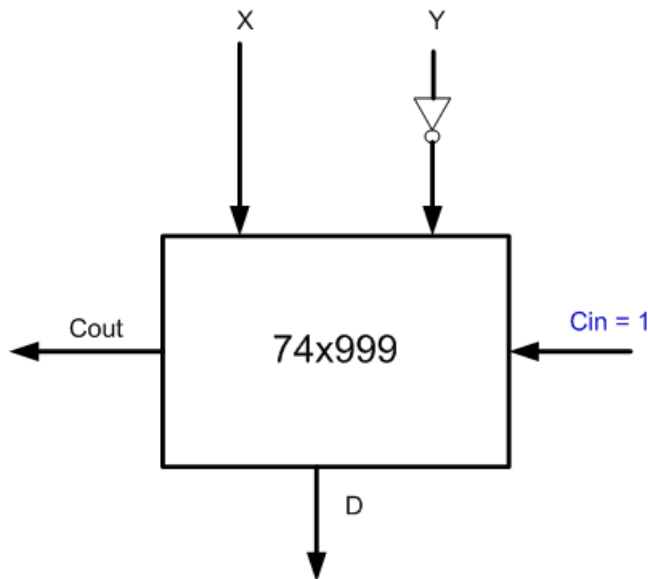


## 3 Adder/ Subtractor



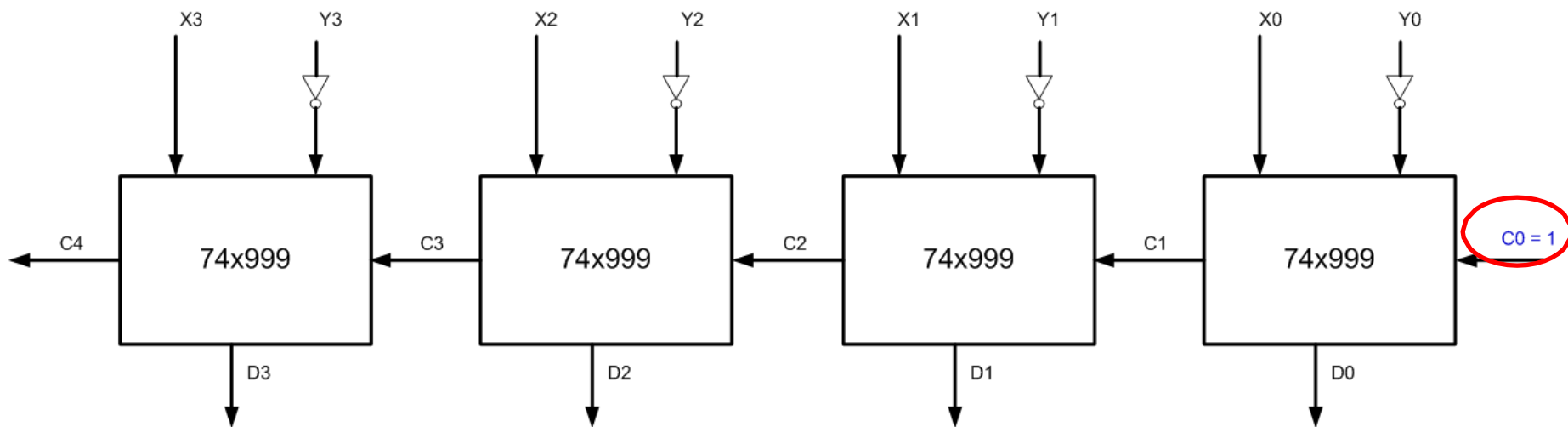
# Mạch cộng/ trừ

- X, Y là 2 số không dấu n-bit
- Phép cộng:  $S = X + Y$
- Phép trừ:  $D = X - Y = X + (-Y) =$   
 $= X + (\text{Bù 2 của } Y)$   
 $= X + (\text{Bù 1 của } Y) + 1$   
 $= X + Y' + 1$



# Mạch trừ

- Mạch cộng Carry Ripple có thể được dùng để xây dựng mạch trừ Carry Ripple bằng cách đảo Y và đặt số nhớ đầu tiên là 1



# Tràn (Arithmetic Overflow)

- **Overflow** là khi kết quả của phép toán vượt quá số bit biểu diễn phần giá trị
  - n bit biểu diễn được số từ  $-2^{n-1}$  đến  $+2^{n-1}-1$
  - Overflow luôn luôn cho ra 1 kết quả sai

+9 →	0	1001
+8 →	0	1000
	1	0001

incorrect sign

↑

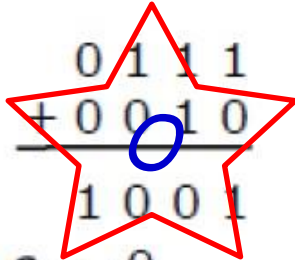
↑

incorrect magnitude

⇒ Mạch để xác định có overflow hay không

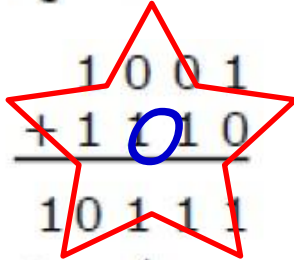
# Ví dụ về arithmetic overflow

- Với số 4 bit, 3 bit giá trị và 1 bit dấu

$$\begin{array}{r}
 (+7) \quad 0 \ 1 \ 1 \ 1 \\
 + (+2) \quad + \ 0 \ 0 \ 1 \ 0 \\
 \hline
 (+9) \quad 1 \ 0 \ 0 \ 1 \\
 c_4 = 0 \\
 c_3 = 1
 \end{array}$$


$$\begin{array}{r}
 (-7) \quad 1 \ 0 \ 0 \ 1 \\
 + (+2) \quad + \ 0 \ 0 \ 1 \ 0 \\
 \hline
 (-5) \quad 1 \ 0 \ 1 \ 1 \\
 c_4 = 0 \\
 c_3 = 0
 \end{array}$$

$$\begin{array}{r}
 (+7) \quad 0 \ 1 \ 1 \ 1 \\
 + (-2) \quad + \ 1 \ 1 \ 1 \ 0 \\
 \hline
 (+5) \quad 1 \ 0 \ 1 \ 0 \ 1 \\
 c_4 = 1 \\
 c_3 = 1
 \end{array}$$

$$\begin{array}{r}
 (-7) \quad 1 \ 0 \ 0 \ 1 \\
 + (-2) \quad + \ 1 \ 1 \ 1 \ 0 \\
 \hline
 (-9) \quad 1 \ 0 \ 1 \ 1 \ 1 \\
 c_4 = 1 \\
 c_3 = 0
 \end{array}$$


- Overflow không xuất hiện khi cộng 2 số trái dấu

# Arithmetic overflow

- Overflow có thể phát hiện được (từ ví dụ ở slide trước)

$$\text{Overflow} = c_3 + c_4$$

$$\text{Overflow} = c_3 \oplus c_4$$

- Với n bit

$$\text{Overflow} = c_{n-1} \oplus c_n$$

- Mạch cộng/ trừ có thể bổ sung mạch kiểm tra overflow với 1 cổng XOR

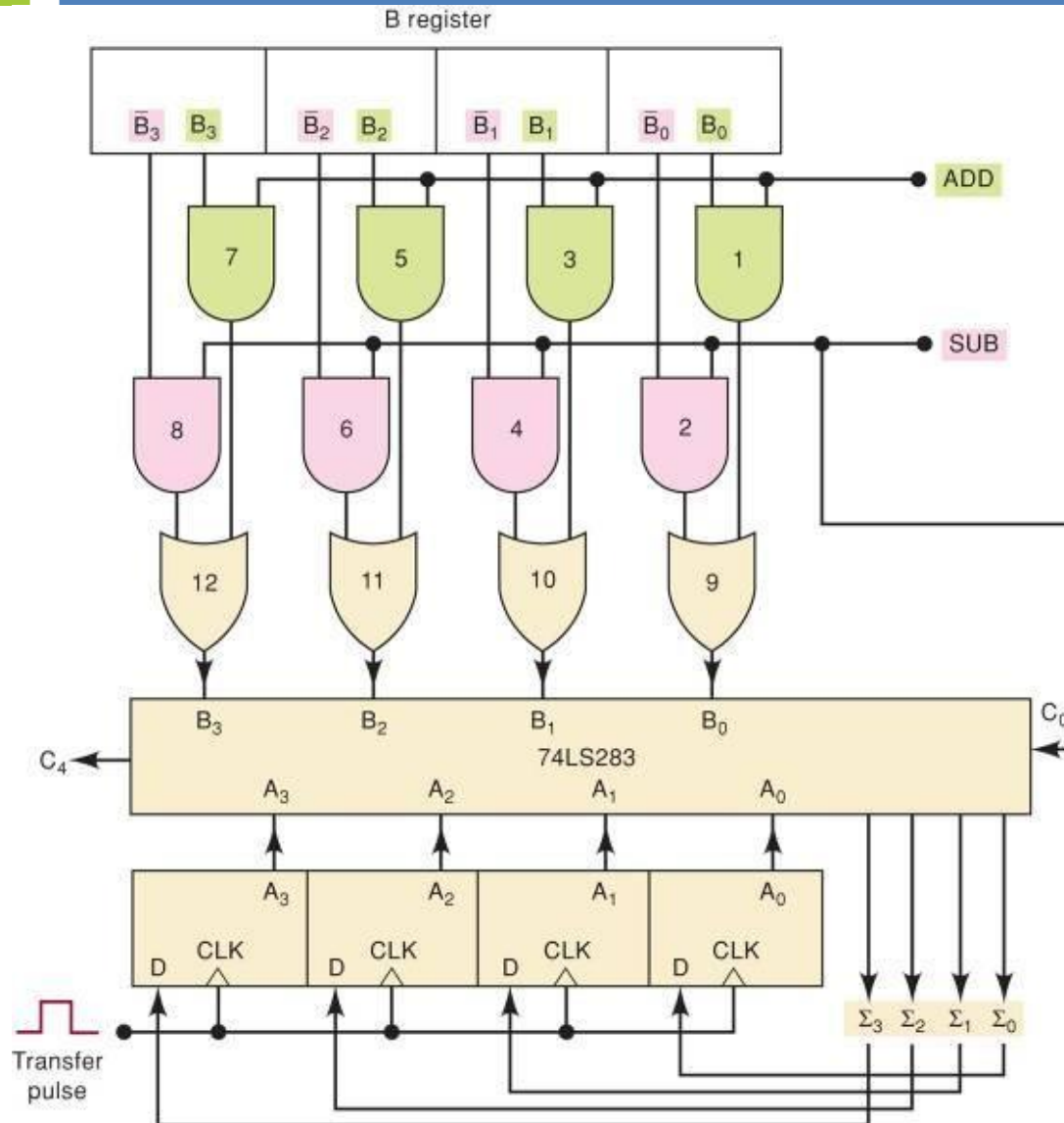
# Ví dụ

- Thiết kế một mạch cộng/ trừ với 2 ngõ điều khiển ADD và SUB
  - $ADD = 1$ : mạch cộng 2 số trong 2 thanh ghi A và B
  - $SUB = 1$ : mạch thực hiện phép trừ số B-A

## Chú ý:

Trong một lúc chỉ một trong hai ngõ ADD, SUB bằng 1

# Ví dụ



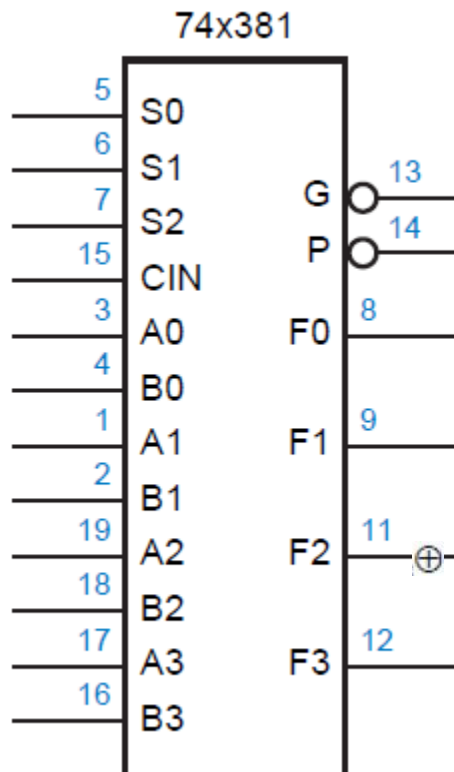


## 4 Arithmetic Logic Unit (ALU)



# ALU

- ALUs có thể thực thi nhiều toán tử và hàm logic khác nhau
  - Các toán tử và hàm được xác định bởi một mã ngõ vào



Inputs			Function
S2	S1	S0	
0	0	0	$F = 0000$
0	0	1	$F = B - A - 1 + Cin$
0	1	0	$F = A - B - 1 + Cin$
0	1	1	$F = A + B + Cin$
1	0	0	$F = A \quad B$
1	0	1	$F = A + B$
1	1	0	$F = A * B$
1	1	1	$F = 1111$



**Any question?**



# **3.4\_Mạch tuần tự: Chốt và Flip-flop**

(Sequential circuit: Latches and Flip-flop)



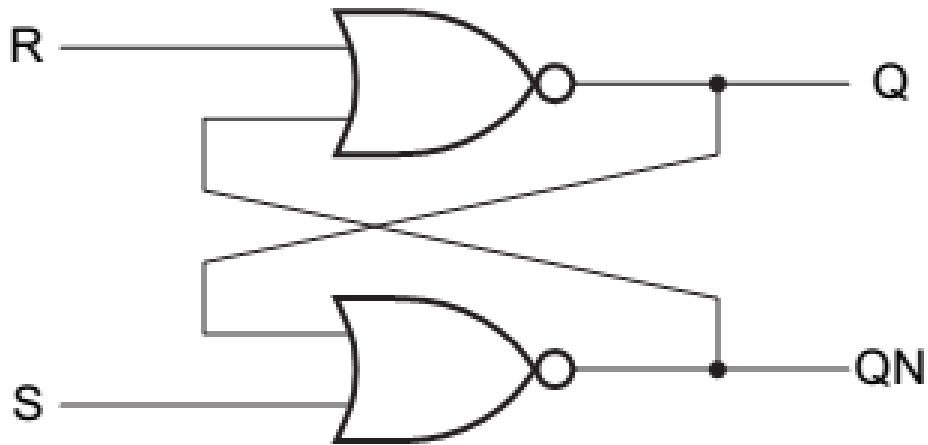
# Nội dung

- 
1. S-R chốt (latch)
  2. D chốt
  3. D Flip-flop
  4. T Flip-flop
  5. S-R Flip-flop
  6. J-K Flip-flop
  7. Scan Flip-flop



# 1. S-R chốt (Set-Reset latch)

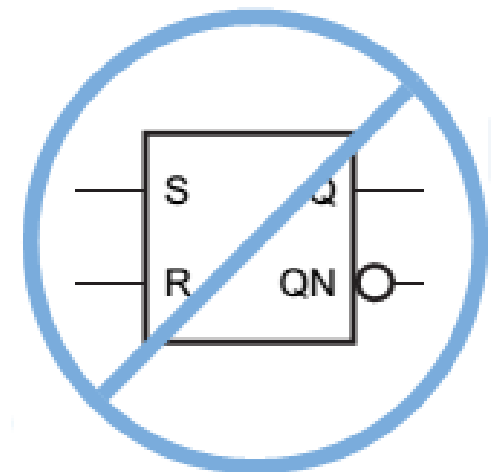
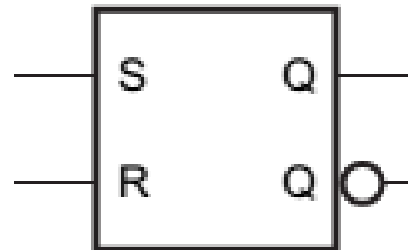
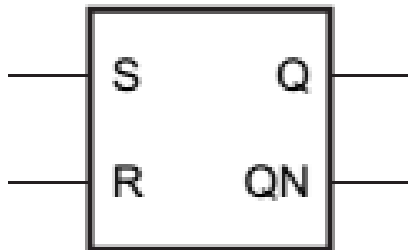
# S-R chốt dùng cổng NOR



Mạch logic

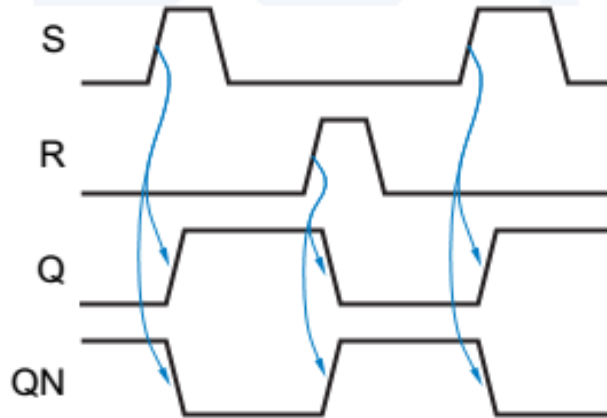
S	R	Q	QN
0	0	last Q	last QN
0	1	0	1
1	0	1	0
1	1	0	0

Bảng chức năng



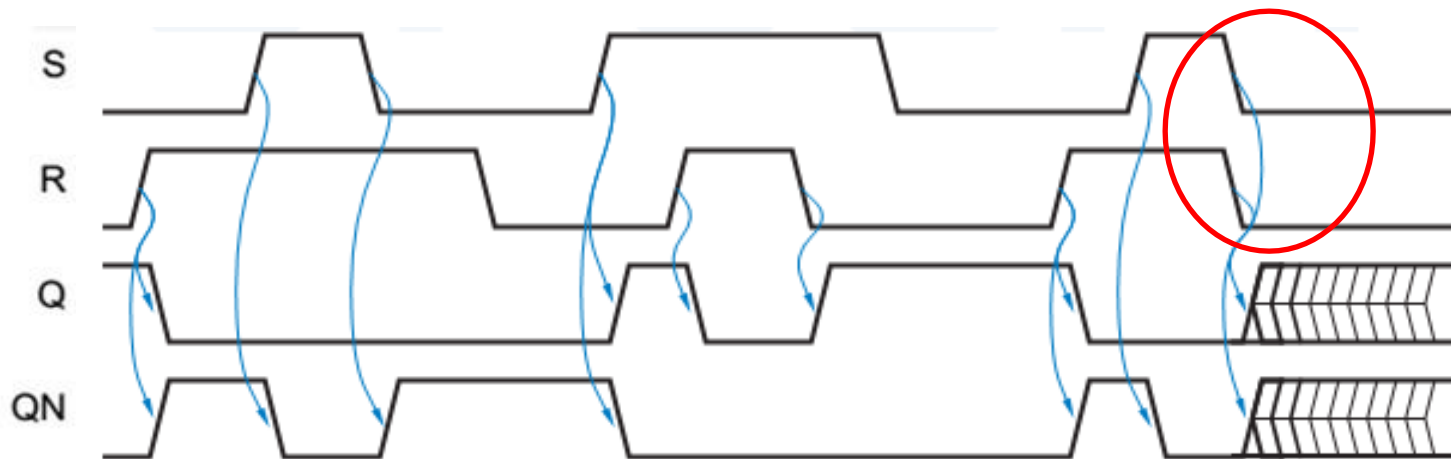
Ký hiệu

# S-R chốt dùng cổng NOR



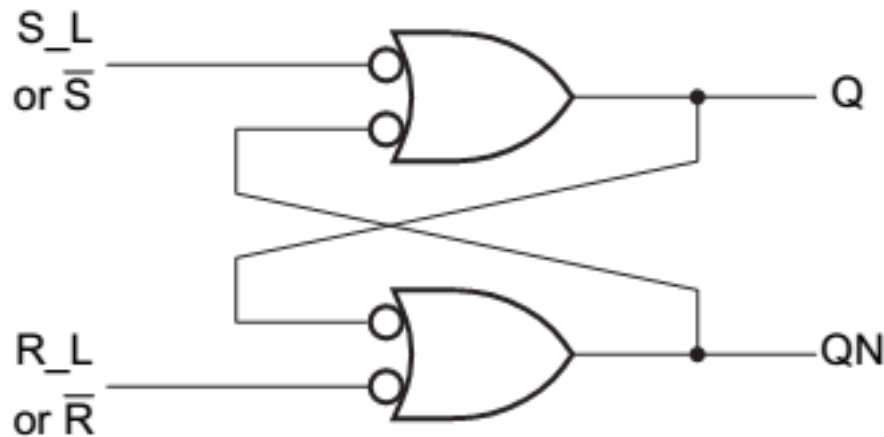
S	R	Q	QN
0	0	last Q	last QN
0	1	0	1
1	0	1	0
1	1	0	0

Ngõ vào thông thường



S và R chuyển từ mức 1 xuống mức 0 đồng thời

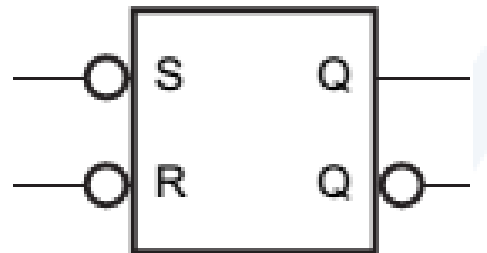
# $\bar{S}$ - $\bar{R}$ chốt dùng cổng NAND



Mạch logic

S_L	R_L	Q	Q $\bar{N}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	last Q	last Q $\bar{N}$

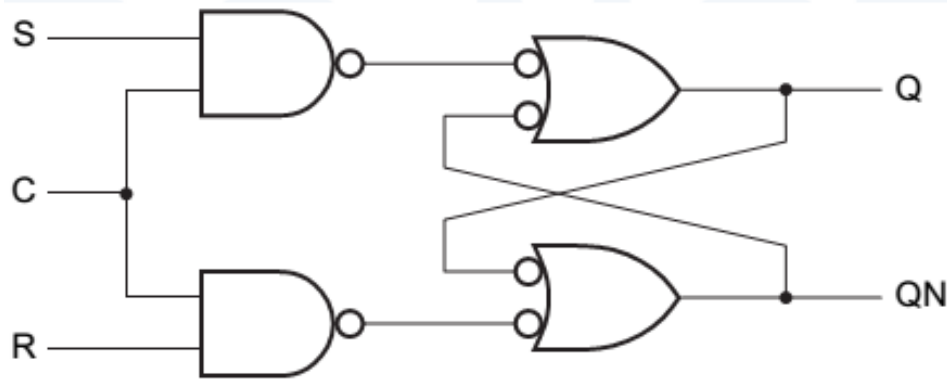
Bảng chức năng



Ký hiệu



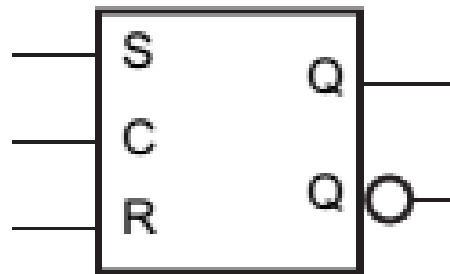
# S-R chốt với ngõ vào cho phép (Enable)



Mạch logic

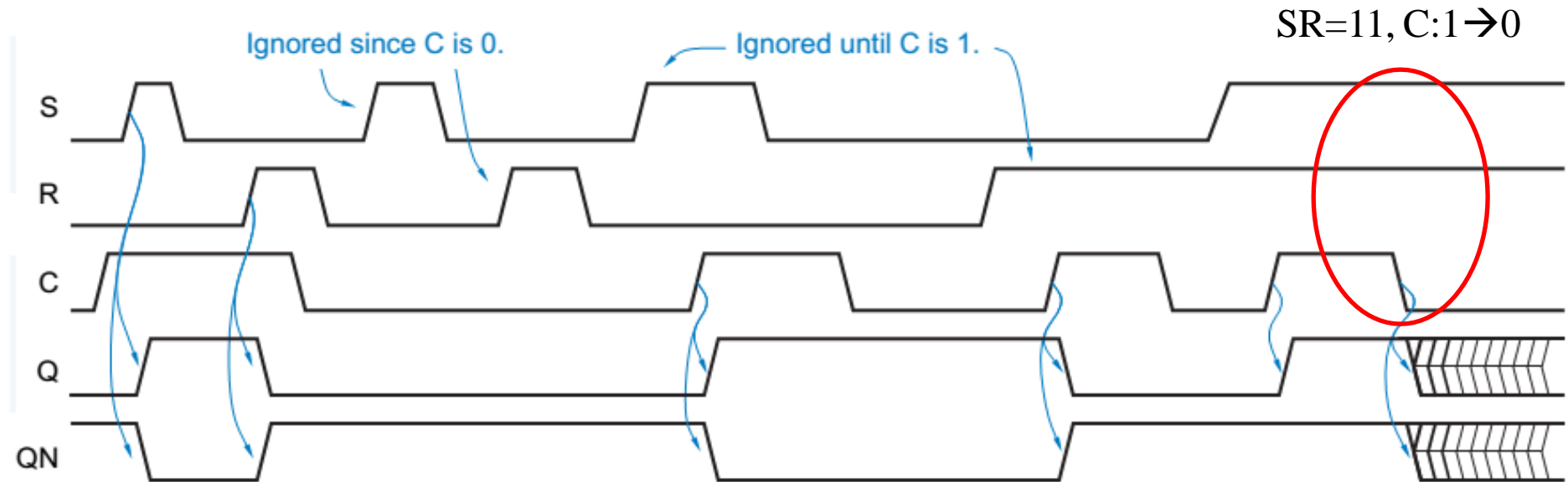
S	R	C	Q	QN
0	0	1	last Q	last QN
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1
x	x	0	last Q	last QN

Bảng chức năng



Ký hiệu

# S-R chốt với ngõ vào cho phép (Enable)

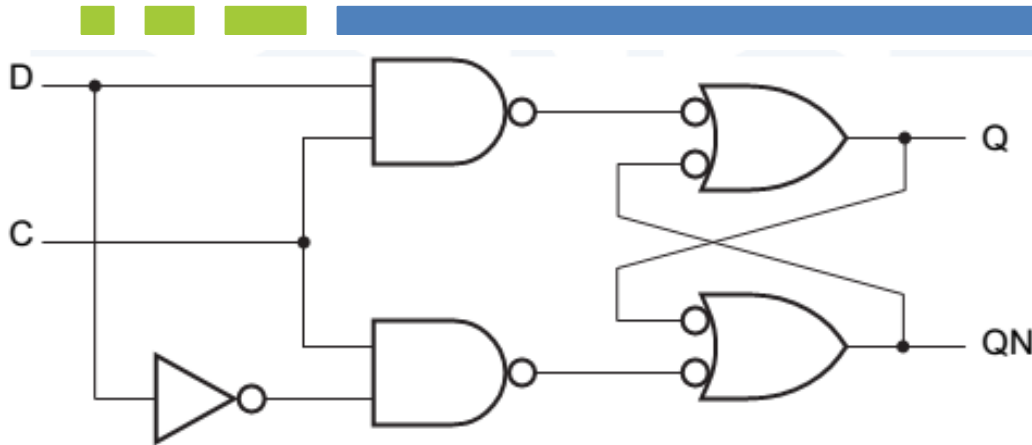


Hoạt động của S-R chốt



## 2. D chốt (Data Latch)

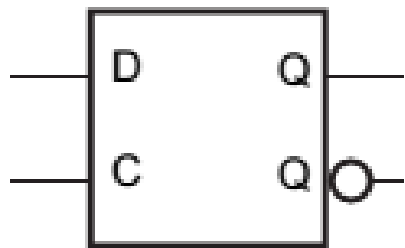
# D chốt



Mạch logic

C	D	Q	QN
1	0	0	1
1	1	1	0
0	x	last Q	last QN

Bảng chức năng



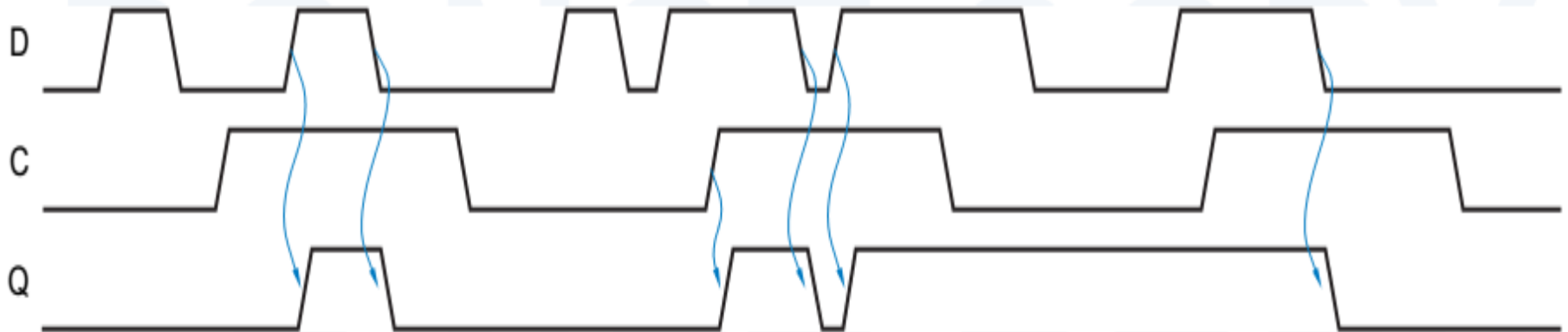
Ký hiệu

- Loại bỏ những hạn chế trong S-R chốt khi mà S và R chuyển từ 1 xuống 0 đồng thời
- Ngõ vào điều khiển C thỉnh thoảng được gọi là ngõ vào cho phép (enable)
- Khi C tích cực,  $Q = D \rightarrow$  chốt mở/trong suốt (transparent latch)
- C không tích cực, Q giữ giá trị trước đó  $\rightarrow$  chốt đóng (close latch)

# D chốt

C	D	Q	QN
1	0	0	1
1	1	1	0
0	x	last Q	last QN

Bảng chức năng



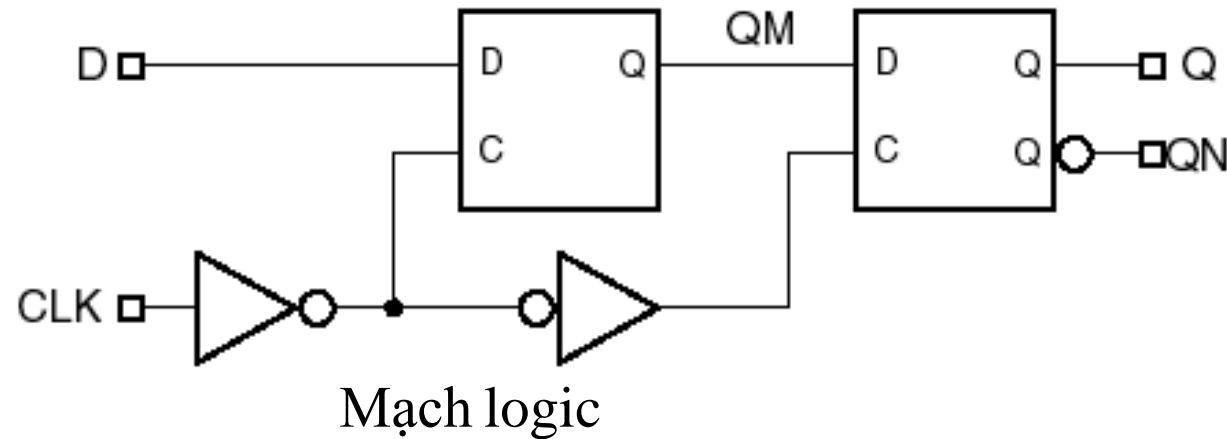
Hoạt động của D chốt





## 3. D (Data) Flip-flop

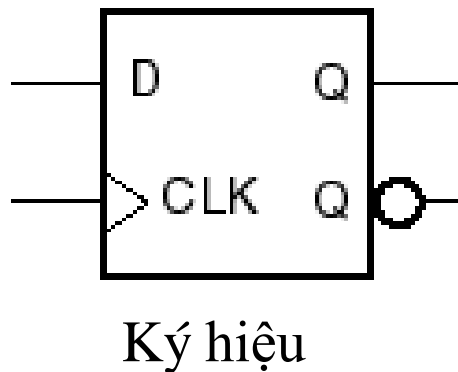
# D flip-flop kích cạnh lên

## (Positive-edge-triggered D flip-flop)



D	CLK	Q	QN
0		0	1
1		1	0



Bảng chức năng



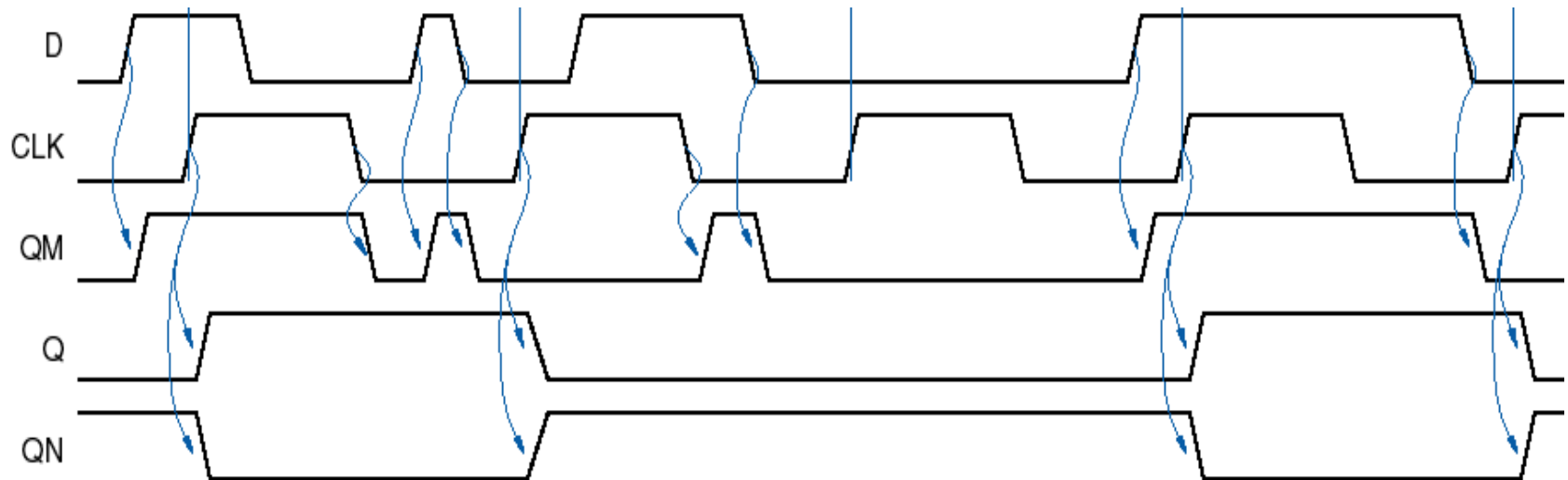
- Một D-FF kích cạnh lên bao gồm một cặp D chốt kết nối sao cho dữ liệu truyền từ ngõ vào D đến ngõ ra Q mỗi khi có cạnh lên của xung Clock (CLK)
- D chốt (latch) đầu tiên gọi là Chủ (master), nó hoạt động khi xung CLK bằng 0
- D chốt thứ hai gọi là Tớ (slave), nó hoạt động khi CLK bằng 1

# D flip-flop kích cạnh lên

## (Positive-edge-triggered D flip-flop)

D	CLK	Q	QN
0		0	1
1		1	0

Bảng chức năng

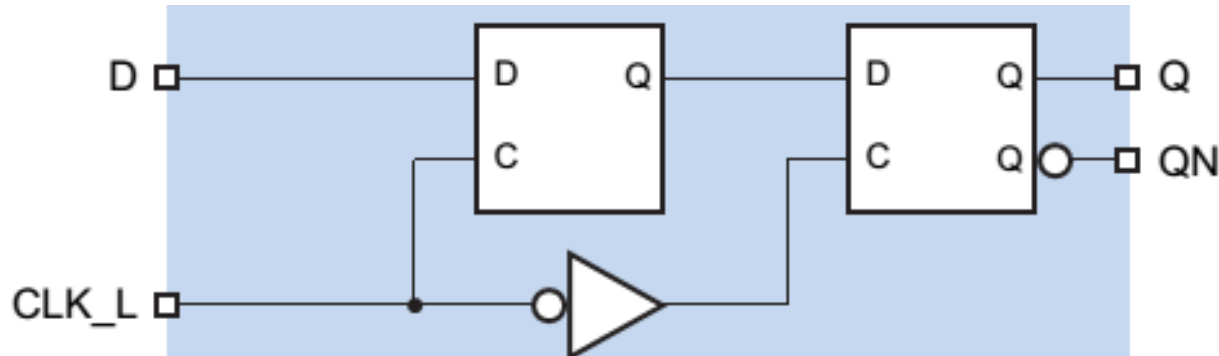


Hoạt động của D Flip-flop kích cạnh lên





# D Flip-flop kích cạnh xuống

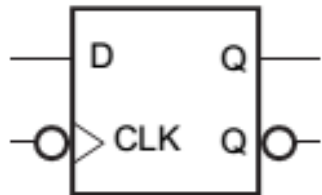
## (Negative-edge-triggered D flip-flop)



Mạch logic

D	CLK_L	Q	QN
0		0	1
1		1	0

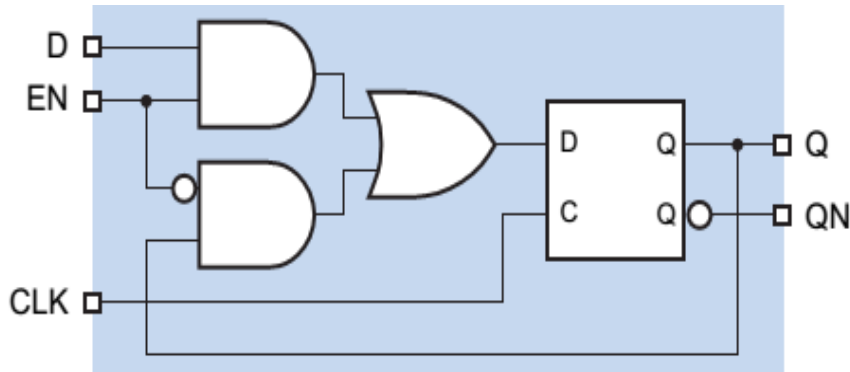
Bảng chức năng






Ký hiệu

- Một D-FF kích cạnh xuống thiết kế giống với D-FF kích cạnh lên, nhưng đảo ngõ vào xung Clock của 2 con D chốt

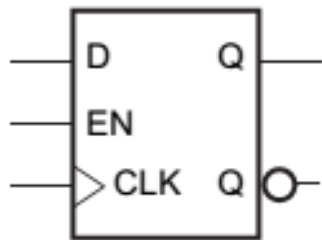
# D flip-flop với ngõ vào điều khiển



Mạch logic

D	EN	CLK	Q	QN
0	1		0	1
1	1		1	0
x	0		last Q	last QN

Bảng chức năng

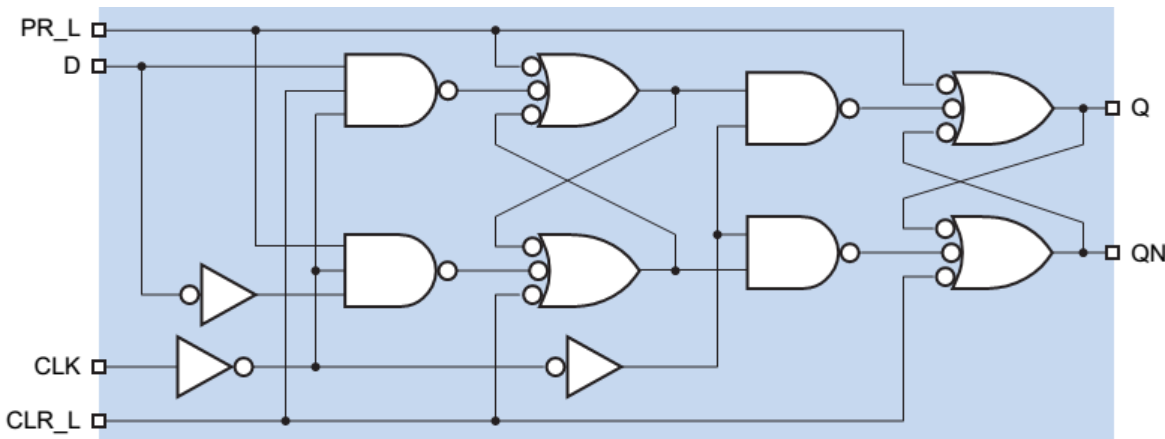


Ký hiệu

- Một chức năng mong muốn của D-FF là khả năng lưu giữ (store) dữ liệu sau cùng hơn là nạp vào (load) dữ liệu mới tại cạnh của xung Clock
- Để thực hiện được chức năng trên, ta thêm vào ngõ vào cho phép (enable input) của mỗi FF. Ngõ vào này thường ký hiệu là EN hoặc CE (chip enable)

# D-FF với ngõ vào bất đồng bộ

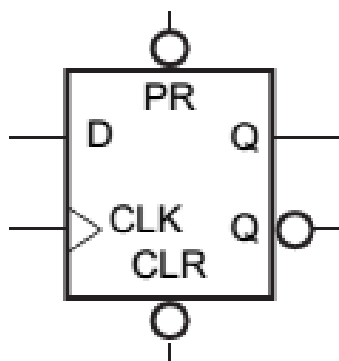
## (D-FF with asynchronous inputs)



Mạch logic

PR PRESET	CLR CLEAR	CLK CLOCK	D DATA	Q	$\bar{Q}$
1	1	$\uparrow$	0	0	1
1	1	$\uparrow$	1	1	0
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1	1

Bảng chức năng



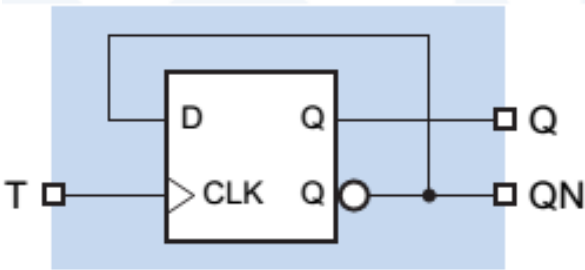
Ký hiệu

- Các ngõ vào bất đồng bộ (Asynchronous inputs) thường được sử dụng để ép ngõ ra Q và Q' (Q-bù) của D-FF đến một giá trị mong muốn mà không phụ thuộc vào ngõ vào D và xung CLK
- Những ngõ vào này thường ký hiệu **PR** (preset) và **CLR** (clear)
- Những ngõ vào **PR** và **CLR** thường được dùng để khởi tạo giá trị ban đầu cho các FF hoặc phục vụ cho mục đích kiểm tra hoạt động của mạch.

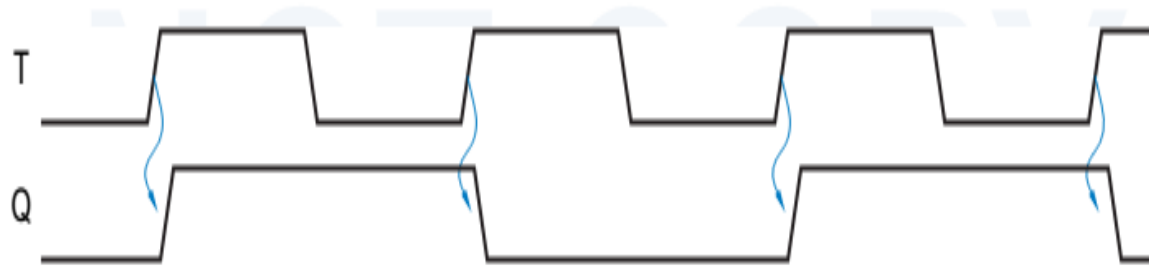


## 4. T (Toggle: lật) Flip-flop

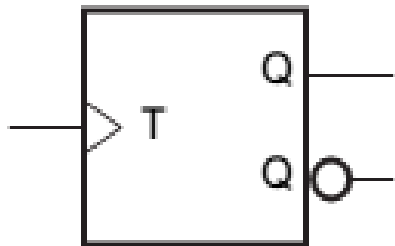
# T Flip-flop (T-FF)



T-FF được thiết kế từ D-FF



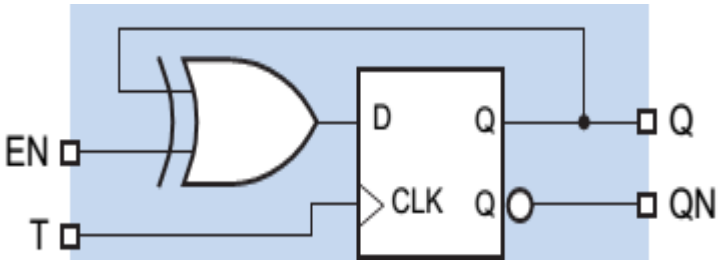
Hoạt động của T-FF tích cực cạnh lên của T



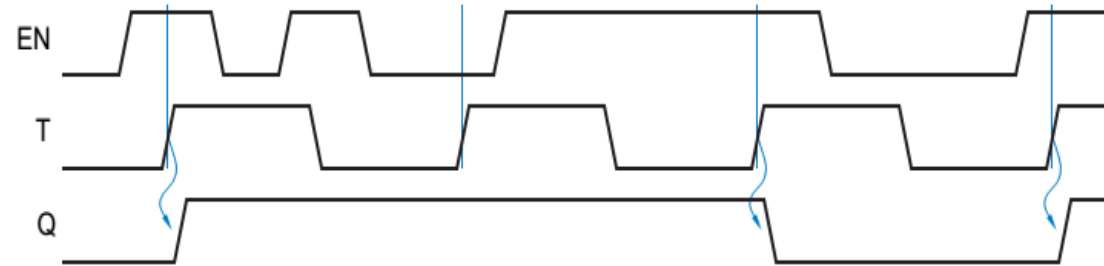
Ký hiệu

- Ngõ ra Q hoặc QN của T-FF sẽ đảo trạng thái mỗi khi có cạnh lên của xung T
- Ngõ ra Q có tần số bằng  $\frac{1}{2}$  tần số của ngõ vào T  
 → T-FF thường được sử dụng trong các bộ đếm hoặc bộ chia tần số

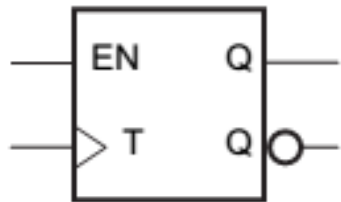
# T Flip-flop với ngõ vào cho phép



T-FF với ngõ vào cho phép  
En được thiết kế từ D-FF



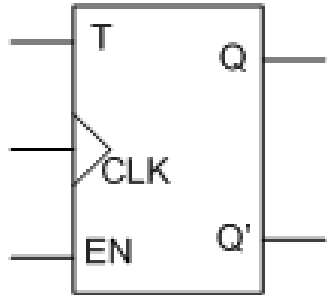
Hoạt động của T-FF tích cực cạnh lên của T và  
ngõ vào cho phép En (Enable) tích cực mức cao



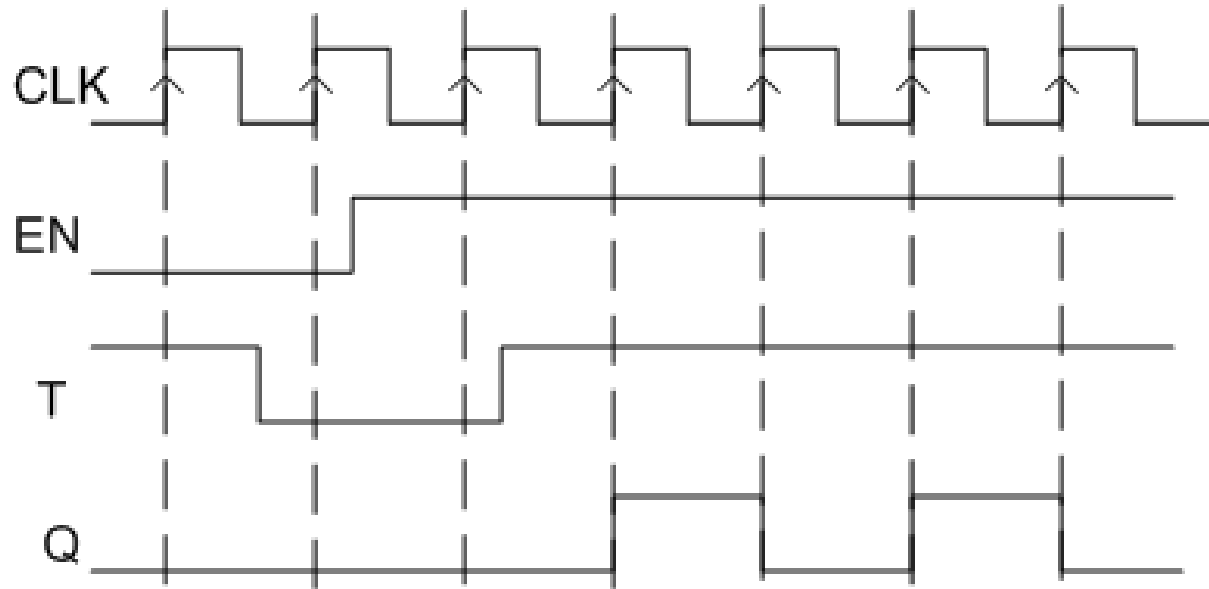
Ký hiệu

- Flip-flop thay đổi trạng thái tại cạnh lên của xung **T** chỉ khi ngõ vào cho phép EN (enable) tích cực.

# T Flip-flop với ngõ vào điều khiển và xung Clock



Ký hiệu



Hoạt động của T-FF tích cực  
cạnh lên của xung Clock

EN	T	CLK	Q	Q'
0	x	x	last Q	last Q'
1	0	f	last Q	last Q'
1	1	f	Q'	Q

Bảng chức năng

- Flip-flop thay đổi trạng thái tại cạnh lên của xung Clock (CLK) chỉ khi ngõ vào cho phép EN (enable) và ngõ vào T tích cực.

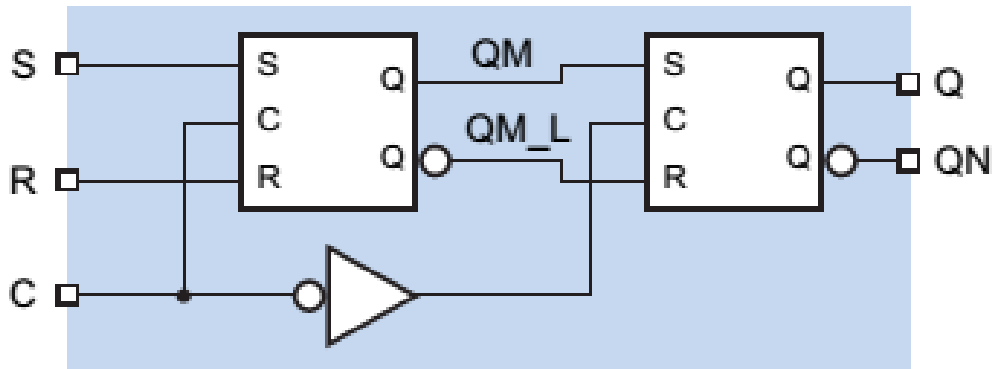






## 5. S-R (Set-Reset) Flip-flop



# S-R flip-flop dạng Chủ-Tớ

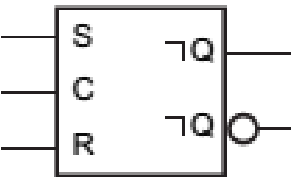
## (Master-Slave S-R flip-flop)



S	R	C	Q	QN
x	x	0	last Q	last QN
0	0		last Q	last QN
0	1		0	1
1	0		1	0
1	1		undef.	undef.

Bảng chức năng

Mạch logic



Ký hiệu

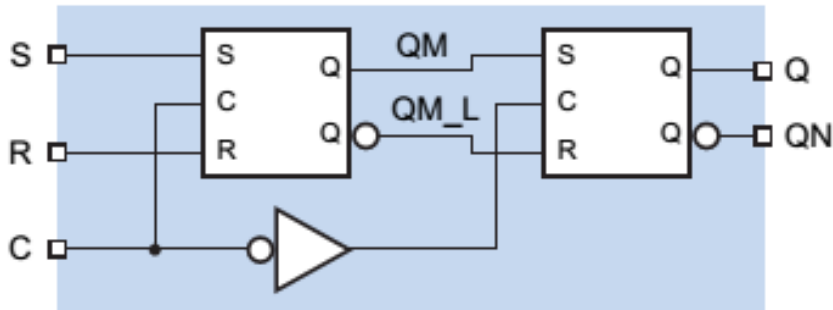
- Flip-flop thay đổi giá trị ngõ ra Q chỉ khi có cạnh xuống của ngõ vào điều khiển C
- Tuy nhiên, giá trị ngõ ra Q thay đổi không chỉ phụ thuộc vào cạnh xuống của ngõ vào C mà còn trong suốt thời gian ngõ vào C bằng 1 trước đó

→ Giá trị ở ngõ ra Q của FF khi có cạnh xuống của xung C phụ thuộc vào giá trị ngõ ra của chốt Chủ (Master latch) bằng 1 hoặc 0 khi ngõ vào C bằng 1 trước đó

- Không có ký hiệu dấu > tại chân C (**dynamic-input indicator**) vì FF này không thật sự được kích bằng cạnh
- Ký hiệu trì hoãn ngõ ra (**postponed-output indicator**) chỉ ra rằng tín hiệu ngõ ra không đổi cho đến khi ngõ vào C xuống mức 0

# S-R flip-flop dạng Chủ-Tớ

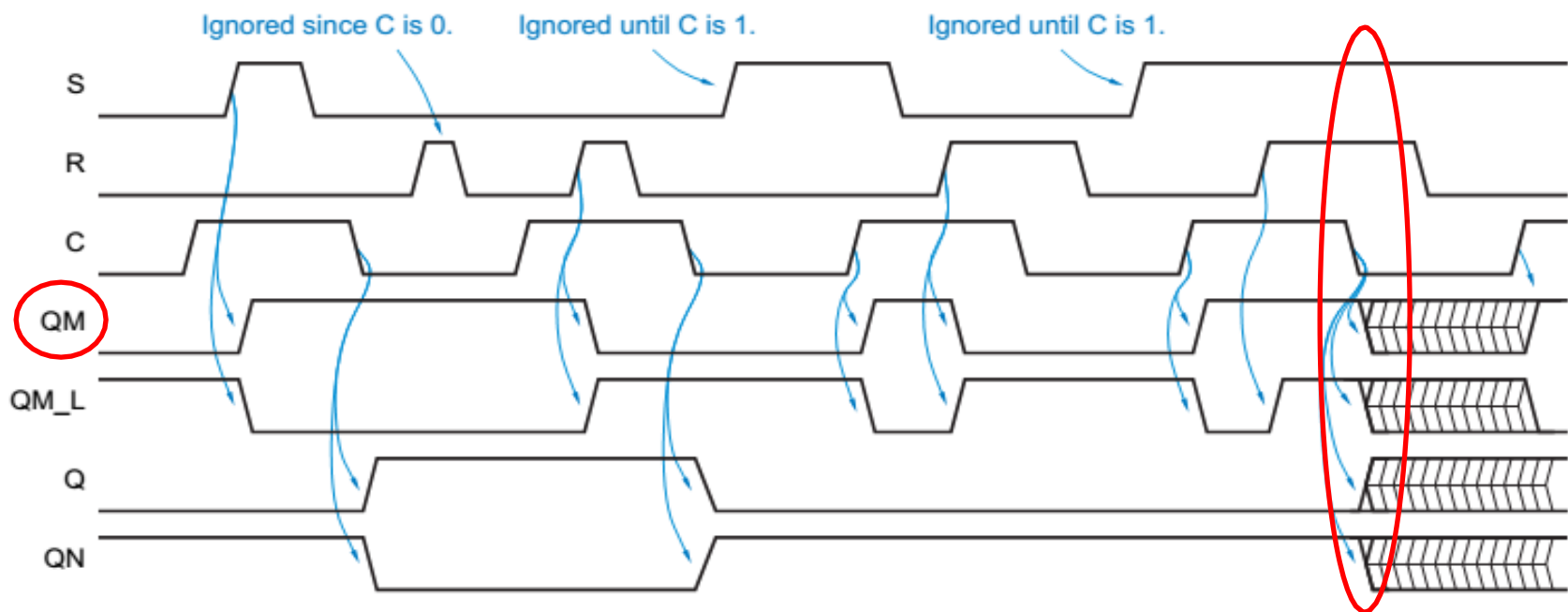
## (Master-Slave S-R flip-flop)



Mạch logic

S	R	C	Q	QN
x	x	0	last Q	last QN
0	0	↓	last Q	last QN
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	undef.	undef.

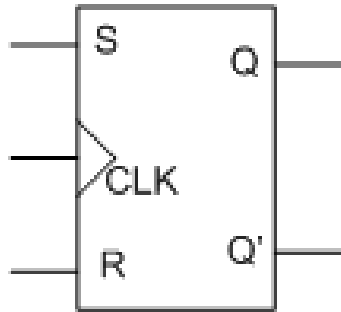
Bảng chức năng



Hoạt động của S-R FF dạng Chủ-Tớ

# S-R flip-flop kích cạnh lên

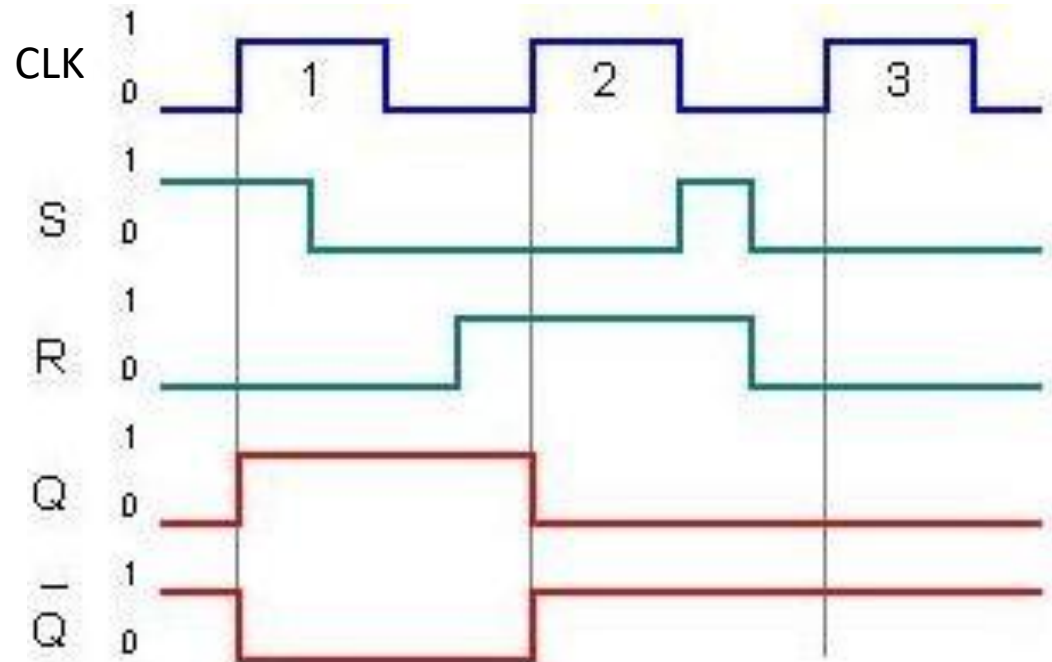
## (Positive-edge-triggered S-R flip-flop )



Ký hiệu

S	R	CLK	Q	Q'
0	0	f	last Q	last Q'
0	1	f	0	1
1	0	f	1	0
1	1	f	×	×

Bảng chức năng



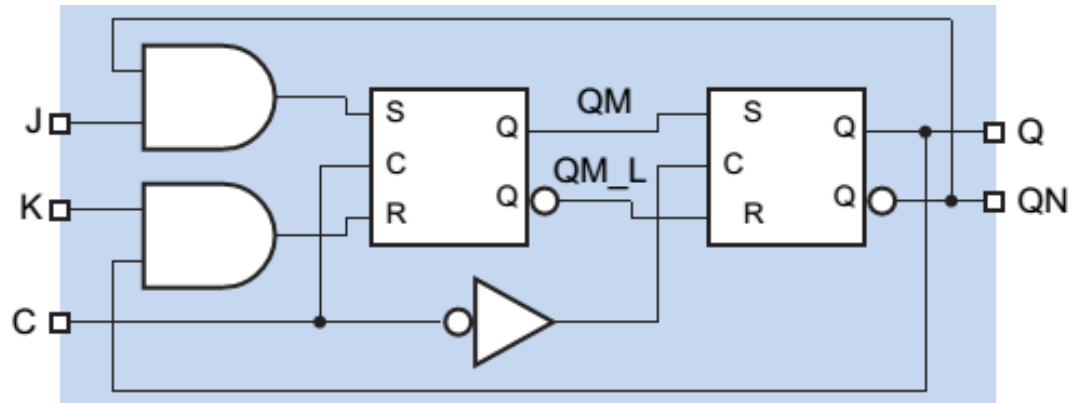
Hoạt động của S-R FF kích cạnh lên







## 6. J-K Flip-Flop

# J-K flip-flop dạng Chủ-Tớ

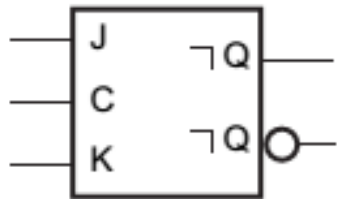
## (Master-Slave J-K flip-flop)



Mạch logic

J	K	C	Q	QN
x	x	0	last Q	last QN
0	0		last Q	last QN
0	1		0	1
1	0		1	0
1	1		last QN	last Q

Bảng chức năng



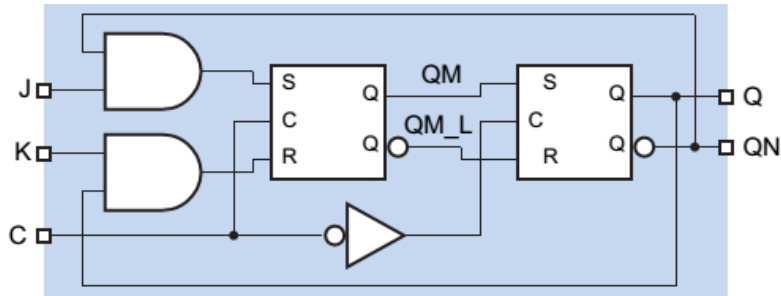
Ký hiệu

- Dấu > tại ngõ vào C (**dynamic-input indicator**) không được sử dụng
- Ký hiệu trì hoãn tại ngõ ra (**postponed-output indicator**) được sử dụng





- Ngõ vào J và K của J-K FF có chức năng tương tự với ngõ vào S và R của S-R FF
- Tuy nhiên, khác với S-R FF, J-K FF giải quyết được vấn đề J và K tích cực đồng thời .

# J-K flip-flop dạng Chủ-Tớ

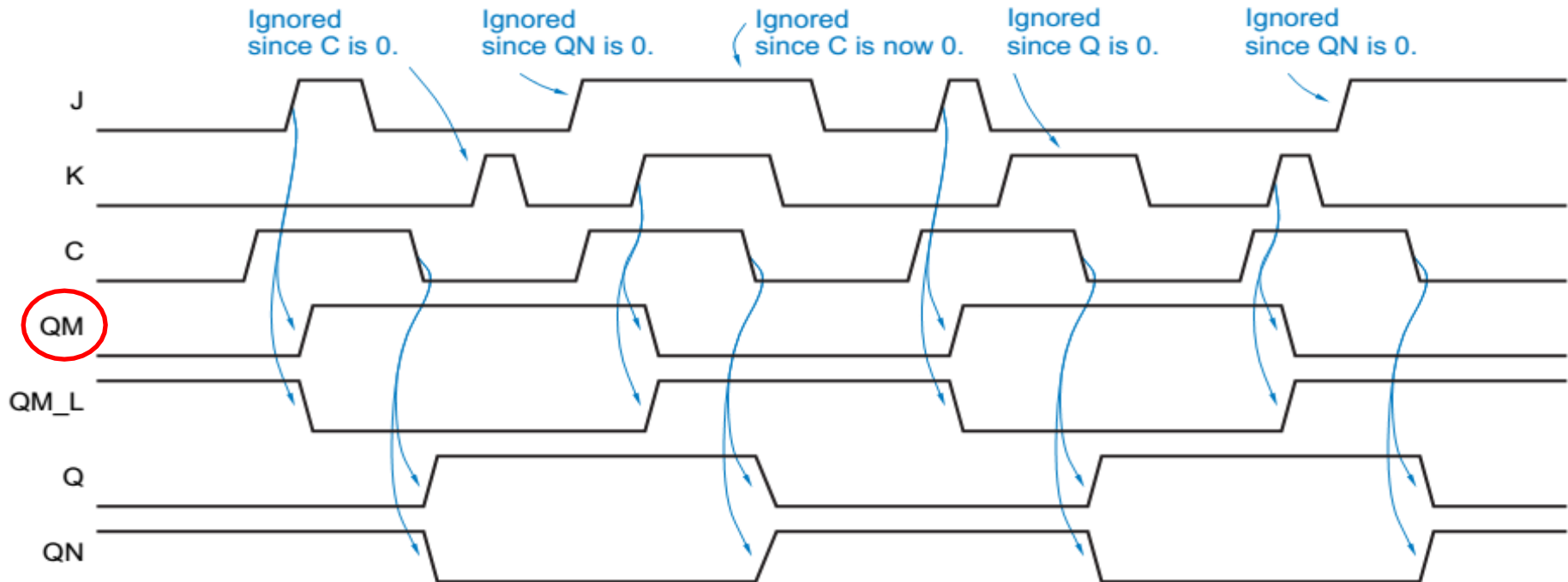
## (Master-Slave J-K flip-flop)



Mạch logic

J	K	C	Q	QN
x	x	0	last Q	last QN
0	0		last Q	last QN
0	1		0	1
1	0		1	0
1	1		last QN	last Q

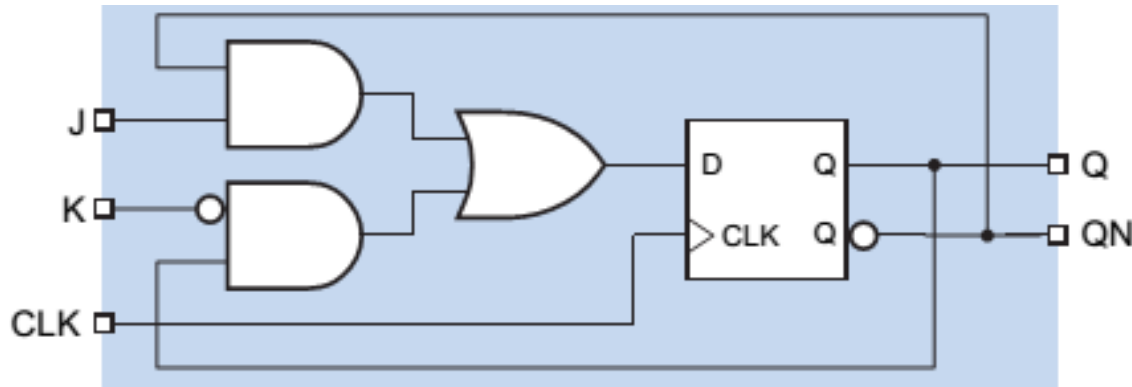
Bảng chức năng



Hoạt động của J-K FF dạng Chủ-Tớ

# J-K flip-flop kích cạnh lên

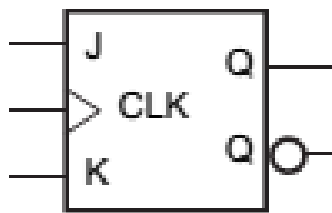
## (Edge-triggered J-K flip-flop)



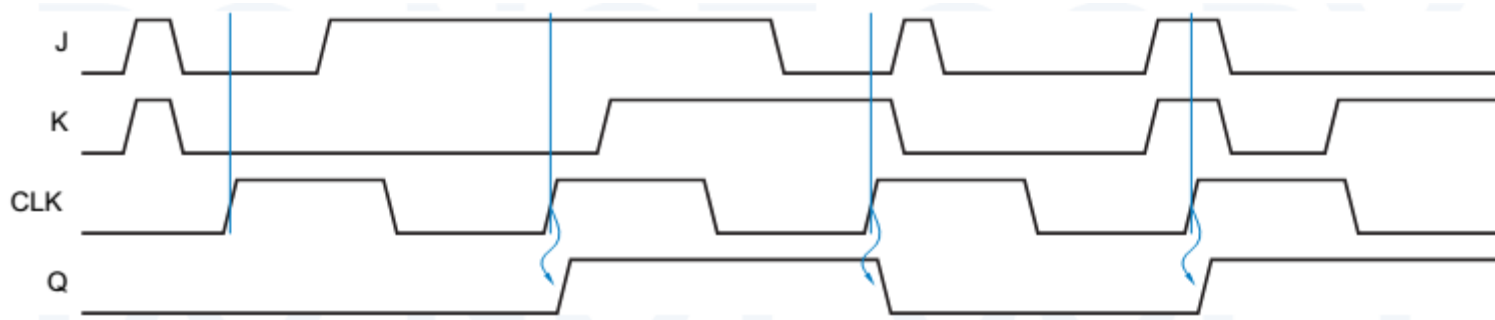
J-K FF kích cạnh lên được thiết kế từ D-FF kích cạnh lên

J	K	CLK	Q	QN
0	0		last Q	last QN
0	1		0	1
1	0		1	0
1	1		last QN	last Q

Bảng chức năng



Ký hiệu



Hoạt động của J-K FF kích cạnh lên







## 7. Scan Flip-Flop



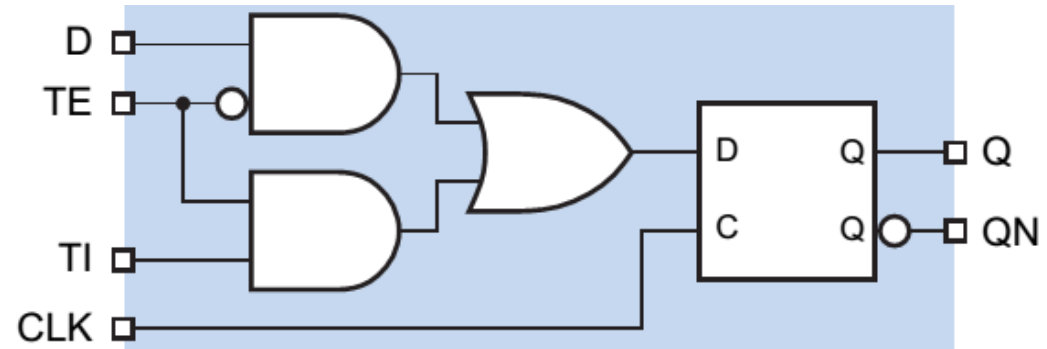
# Scan flip-flop

Chế độ  
bình thường

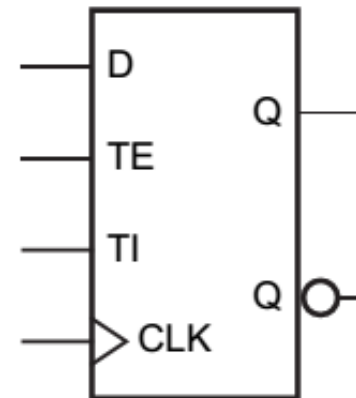
Chế độ  
kiểm tra

TE	TI	D	CLK	Q	QN
0	x	0		0	1
0	x	1		1	0
1	0	x		0	1
1	1	x		1	0
x	x	x	0	last Q	last QN
x	x	x	1	last Q	last QN

Bảng chức năng

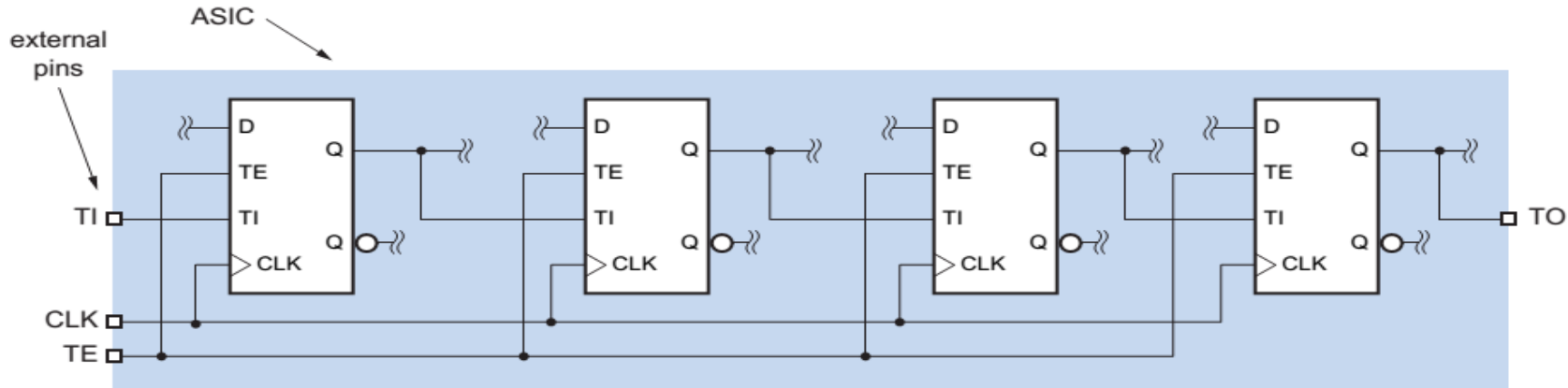


D-FF kích cạnh lên có chế độ Scan



Ký hiệu

# Scan flip-flop



Một chuỗi 4 FFs hoạt động trong chế độ Scan

- Một tính năng quan trọng của các FF được chế tạo ở mức **ASIC** là **khả năng Scan (khả năng kiểm tra)**  
Các ngõ vào phụ (TI, TE, TO) được kết nối đến tất cả các FF theo một chuỗi Scan để phục vụ cho mục đích kiểm tra
- Trong chế độ kiểm tra (testing mode), một chuỗi dữ liệu kiểm tra (test pattern) được đưa vào các FF thay thế cho chuỗi dữ liệu thông thường
- Sau khi các test pattern được đưa vào các FF, các FF sẽ quay trở lại chế độ hoạt động bình thường (normal mode)
- Sau một hay nhiều cạnh lên của xung Clock, các FF quay lại chế độ kiểm tra và kết quả kiểm tra được xuất ra ngoài tại ngõ ra của các FF

# Ghi chú

- Khi nguồn điện được đưa vào một Flip-flop (FF), nếu ngõ vào PRESET hoặc CLEAR không tích cực thì giá trị ngõ ra của FF này có thể rơi vào trạng thái không xác định (hoặc bằng 0 hoặc bằng 1)
- Để khởi tạo cho FF một giá trị mong muốn ban đầu, chúng ta phải tích cực ngõ vào PRESET (nếu muốn ngõ ra bằng 1) hoặc CLEAR (nếu muốn ngõ ra bằng 0).



# Thảo luận?