



數位邏輯實習

栓鎖器、正反器

國立台北科技大學 電機工程系 吳昭正



Outline

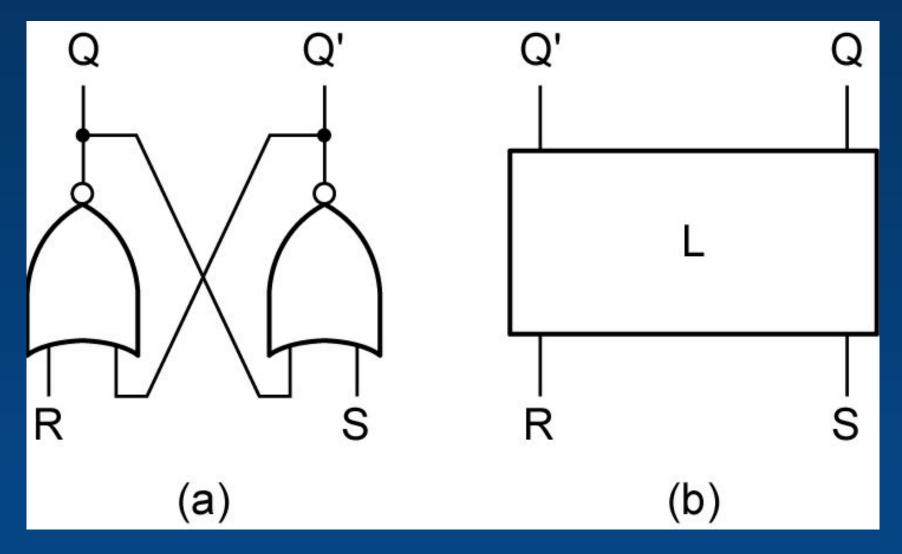


- 栓鎖器(Latch)
- 正反器(Flip-Flop)
- Digital的正反器與時脈產生器
- VHDL語法介紹
- 作業題



S-R Latch(1/2)

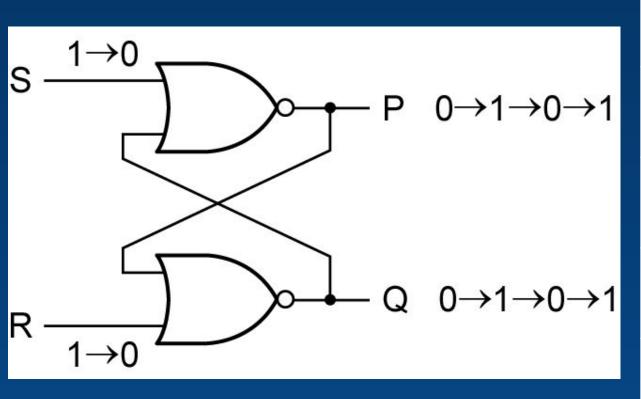






S-R Latch(2/2)



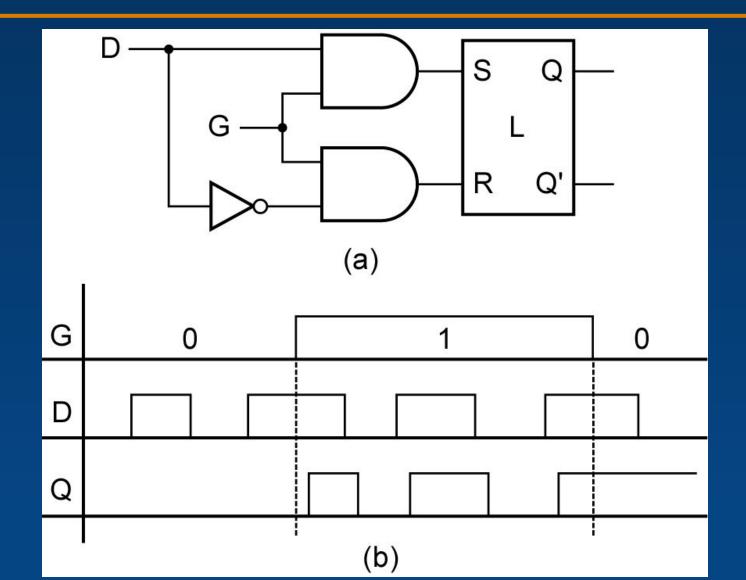


SR	Q ⁺
0 0	Q
0 1	0
10	1
11	Not allowed



D Latch(1/2)

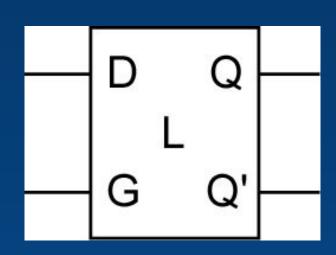






D Latch(2/2)



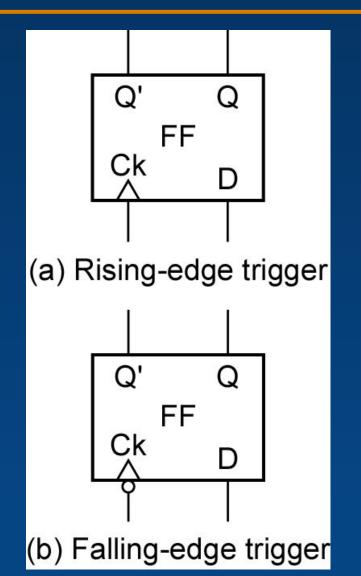


G	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



D $\overline{\text{Flip-Flop}(1/2)}$





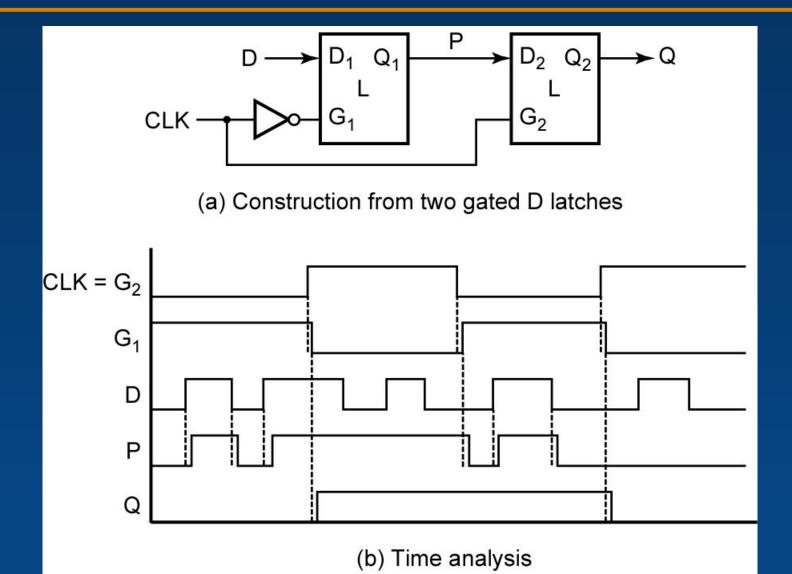
DQ	Q^+		
0 0	0		
0 1	0		
1 0	1		
1 1 1			
(c) Truth table			
O ⁺ - D			

$$Q^{\dagger} = D$$



D Flip-Flop(2/2)

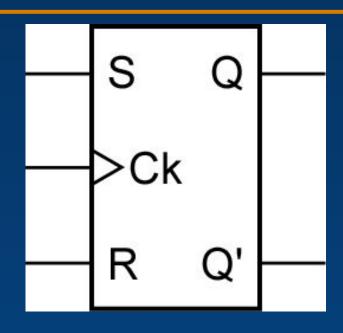






S-R Flip-Flop(1/2)





Operation summary:

S = R = 0 no state change

S = 1, R = 0 set Q to 1 (after active Ck edge)

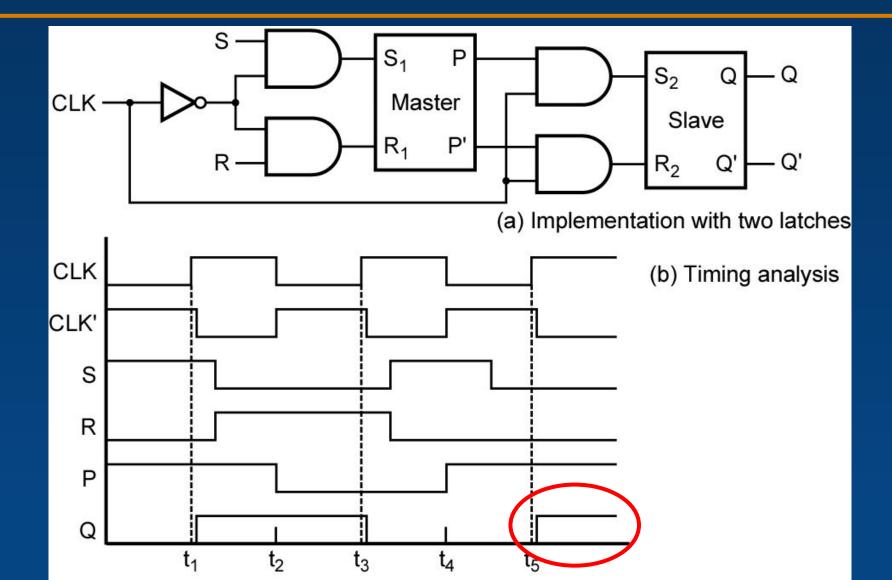
S = 0, R = 1 reset Q to 0 (after active Ck edge)

S = R = 1 not allowed



S-R Flip-Flop(2/2)

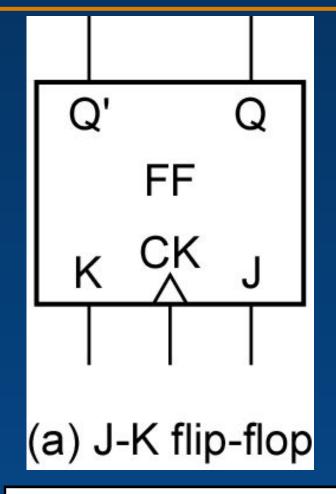






J-K Flip-Flop(1/2)





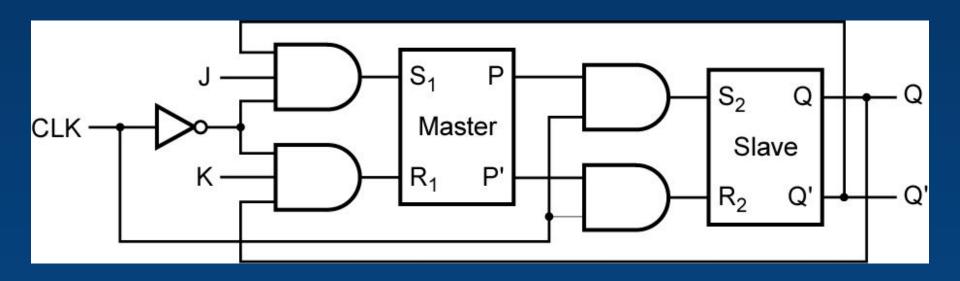
$$Q' = JQ' + K'Q$$

JK	Q	Q^+
0 0	0	0
00	1	1
0 1	0	0
0 1	1	0
10	0	1
10	1	1
1 1	0	1
11	1	0



J-K Flip-Flop(2/2)

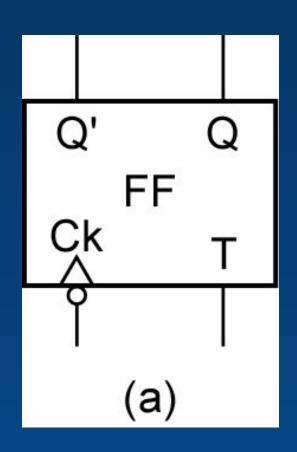






T Flip-Flop(1/2)





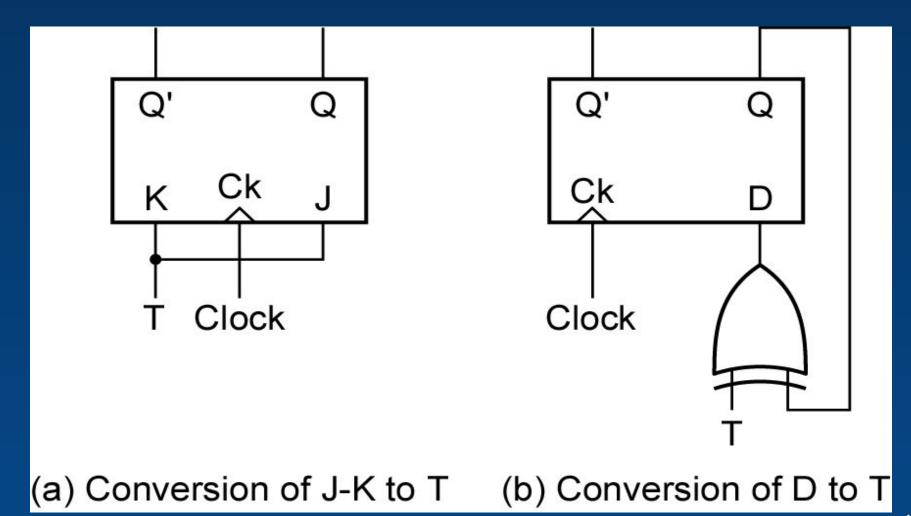
	T Q	Q^+			
	0 0	0			
	0 1	1			
	1 0	1			
	1 1	0			
+	= T'Q + TQ' = Q ⊕ T				

$$Q^{\dagger} = T'Q + TQ' = Q \oplus T$$



T Flip-Flop(2/2)







Boolean Algebra for Flip-Flops



D flip-flop
$$Q^+ = D$$
 (13-1)
D-CE flip-flop $Q^+ = D \cdot CE + Q \cdot CE'$ (13-2)
T flip-flop $Q^+ = T \cdot Q$ (13-3)
S-R flip-flop $Q^+ = S + R'Q$ (13-4)
J-K flip-flop $Q^+ = JQ' + K'Q$ (13-5)



Digital內建的正反器



		Digital
File Edit View Simulation Analysis Components	Windows Help	Digital
CO COGIC IO		
Wires Plexers		<u>.</u>
Flip-Flops Memory	RS-Flip-flop	
Arithmetic) Switches	RS-Flip-flop, clocked	
Misc. Library	JK-Flip-flop	
	D-Flip-flop	
	T-Flip-Flop	
	JK-Flip-flop, asynchronous	
	JK-AS Set Q D	
	D-Flip-flop, asynchronous	
	R O Monoflop	
nodes		



TECH Digital內的時脈產生器 REGISTRIAN EN LEGISTRE AND RESTRICT TECH Digital內的時態



			Digit	al					
File Edit View Simulation Analysis Comp	onents Wi	ndows Help	-	2070					
Logic					ID O				
	• •	Output		External O	<u>c</u> <u>d</u>				
Wires		-							
Plexe		LED							
		Input						00 (8 00) (8 00) (8 0	
	lops	Clock Input							
Memo	··, · ·								
	metic) 💾	A clock sig	nal. Its possible to contro	l it by a real-time	clock.			100 10 100 10 100 10 10	
Switc	hes 🕨 🔳	DIPDepending	on the complexity of the	circuit, the clock	frequency				
Misc.	▶ Text	Ter achieved h	nay be less than the selec	ted value. If the 1	requency is				
Libra	ry 🕨 🖓	Prolonger be a	an 50Hz, the graphic repr updated at every clock cyc	esentation of the	re colors will no				
	-	longer be t	ipdated. If the real-time c	lock is not activa	ted, the clock				
		Da can be con	updated. If the real-time c trolled by mouse clicks. Is	also used to ass	ign an pin				
		number, if	code for a CPLD or FPGA i	s generated.	•				
	· · · · · · · · ·	-							
		Displays 🕨							
		Mechanical)							
		Peripherals >						era era era e	
								(a) (a) (a) (a) (a) (b)	
								en a lei a lei a l	
								(a) (a) (a) (a) (a) (a) (b)	
						· pElo	Clock	Input	7 0 X
						- AC*	CIUCK	mput	
						Basic	Advanced		
						Label:			
						(3)	1		
						Star	t real time o	Clock	
						191			
						Freque	ncy/Hz: 1		-
1 nodes						1 1 0 1 0 0			
						<u> </u>	22	544	
							Help	Cancel	ок
							The state of the s		

TAIPEI VHDL語法介紹(01/14) RSWIP Modeling Flip-Flops Using Processes

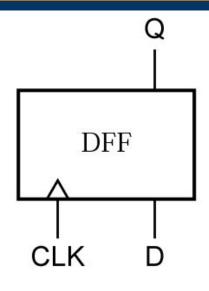
A flip-flop can change state either on the rising or on the falling edge of the clock input. This type of behavior is modeled in VHDL by a process.

Section 17.1 (p. 554)

VHDL語法介紹(02/14)



Modeling Flip-Flops Using Processes



```
process (CLK)
begin
  if CLK'event and CLK = '1' -- rising edge of CLK
     then Q <= D;
  end if;
end process;</pre>
```

TAIPEI VHDL語法介紹(03/14) RESIDENT Modeling Flip-Flops Using Processes

A basic process has the following form:

```
process(sensitivity-list)
begin
  sequential-statements
end process;
```

The basic if statement has the form:

VHDL語法介紹(04/14) RSMP



TECH Modeling Flip-Flops Using Processes

```
process (G,D)
begin
   if G = '1' then Q \le D; end if;
end process;
```

Figure 17-2: VHDL Code for a Transparent Latch

VHDL語法介紹(05/14)



Modeling Flip-Flops Using Processes

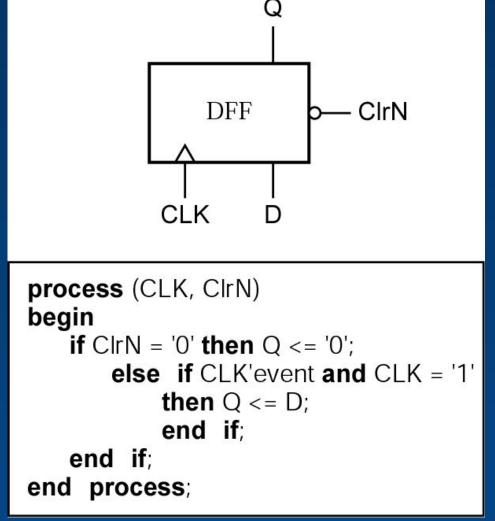


Figure 17-3: VHDL Code for a D Flip-Flop with Asynchronous Clear

TAIPEI VHDL語法介紹(06/14) RESIDENT Modeling Flip-Flops Using Processes

The most general form of the **if** statement is

if condition then
 sequential statements
{elsif condition then
 sequential statements}
 -- 0 or more elsif clauses may be included
[else sequential statements]
end if;

The curly brackets indicate that any number of **elsif** clauses may be included, and the square brackets indicate that the **else** clause is optional.

Section 17.1 (p. 554)

VHDL語法介紹(07/14)



Modeling Flip-Flops Using Processes

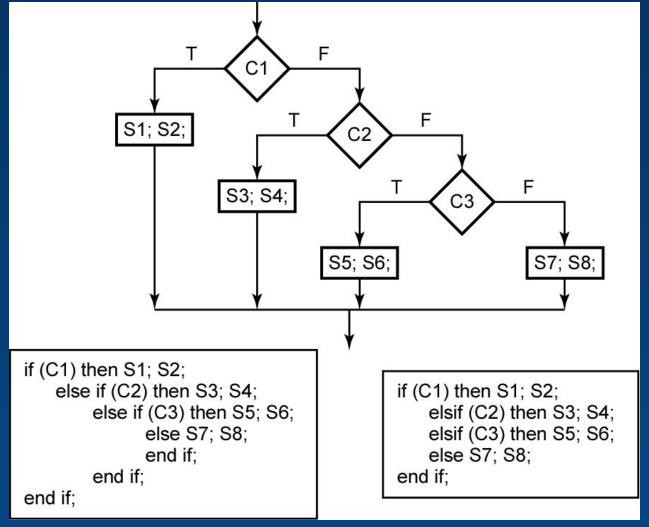
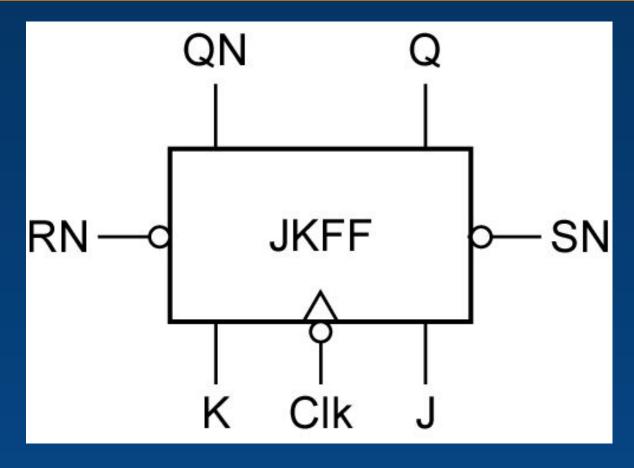


Figure 17-4: Equivalent Representations of a Flow Chart Using Nested Ifs and Elsifs

VHDL語法介紹(08/14)



Modeling Flip-Flops Using Processes



 $Q^+ = JQ' + K'Q$

Figure 17-5: J-K Flip-Flop

TAIPEI VHDL語法介紹(09/14) REMOTE SENSING AND MEDICAL MAGE PROCESSING ** Modeling Flip-Flops Using Processes

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
entity JKFF is
port (SN, RN, J, K, CLK: in std_logic;
    Q, QN: out std_logic);
end JKFF;
```

TAIPEI VHDL語法介紹(10/14) RSMITTECH Modeling Flip-Flops Using Processes

```
architecture JKFF1 of JKFF is
signal Qint: std_logic; -- internal value of Q
begin
    Q <= Qint; -- output Q and QN to port
    QN <= not Qint;
    process (SN, RN, CLK)
    begin
         if RN = '0' then Qint <= '0'; -- RN='0' will clear the FF
         elsif SN = '0' then Qint <= '1'; -- SN='0' will set the FF
         elsif CLK'event and CLK = '0' then -- falling edge of CLK
             Qint <= (J and not Qint) or (not K and Qint);
         end if;
     end process;
end JKFF1;
```

VHDL語法介紹(11/14) RSMP



Modeling Flip-Flops Using Processes

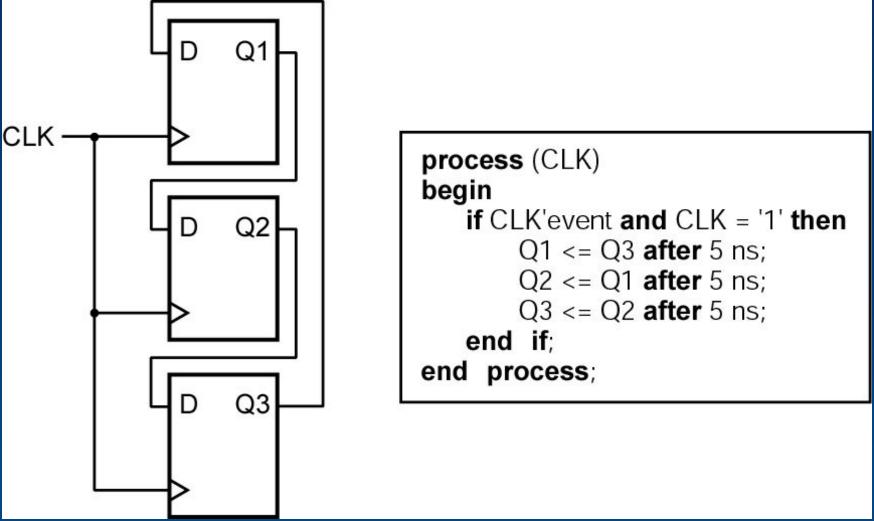
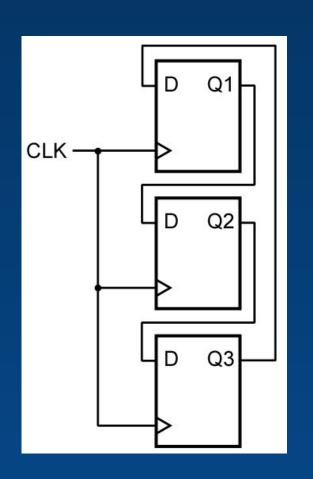


Figure 17-7: Cyclic Shift Register

VHDL語法介紹(12/14)



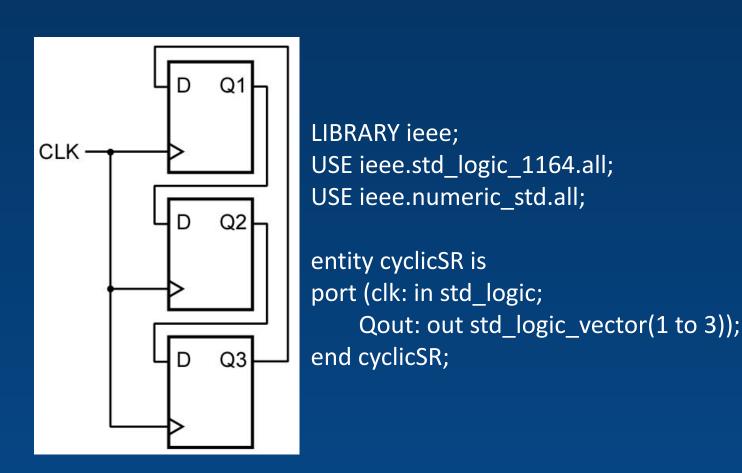
Modeling Flip-Flops Using Processes



```
LIBRARY ieee;
USE ieee.std logic 1164.all;
USE ieee.numeric_std.all;
entity my DFF is
port (D, clk: in std_logic;
    Q: out std_logic);
end my DFF;
architecture DFF simple of my DFF is
begin
     process (clk)
     begin
          if clk'event and clk = '1' then
               Q \le D after 5 ns;
          end if;
     end process;
end DFF_simple;
```

Figure 17-8a: Structural VHDL Code for Cyclic Shift Register

TAPE VHDL語法介紹(13/14) RSM REDICAL MAGE PROCESSING AND MEDICAL MAGE PROCESSING AND MEDI



TAIPEI VHDL語法介紹(14/14) RSMUP REDICTION WAS PROCESSES WHO WE WITH A PROCESSES WHO WE WAS A PROCESSES WHO W

```
architecture cyclicSR3 of cyclicSR is
                        component my_DFF
                             port (D, clk: in std logic;
CLK
                             Q: out std_logic);
                        end component;
                        signal Q1, Q2, Q3: std_logic;
                         begin
                             FF1: my_DFF port map (Q3, clk, Q1);
                             FF2: my_DFF port map (Q1, clk, Q2);
                             FF3: my_DFF port map (Q2, clk, Q3);
                             Qout <= Q1 & Q2 & Q3;
                        end cyclicSR3;
```



作業題(1/2)



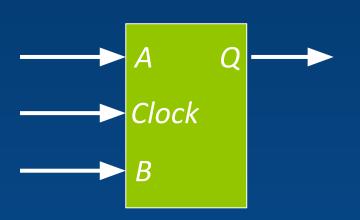
- 作業題1:請利用以下兩種方式實作S-R正反器
- 方法一:利用NOR邏輯閘實作投影片第3頁的 S-R Latch,將完成的S-R Latch模組化之後,實作 投影片第10與12頁的S-R正反器。
- 方法二:利用VHDL Process實作S-R正反器
- 比較投影片第10頁的波型圖與兩個方法實作S-R 正反器的波形圖。
- •請挑戰修正投影片第10頁中紅色區域的錯誤。



作業題(2/2)



- 作業題2:請利用以下兩種方式實作A-B正反器
- 方法一利用VHDL Process實作A-B正反器
- 方法二利用作業題1的S-R正反器實作A-B正反器



A B	Q ⁺
0 0	0
0 1	Q
1 0	Q'
11	1