數位邏輯實習

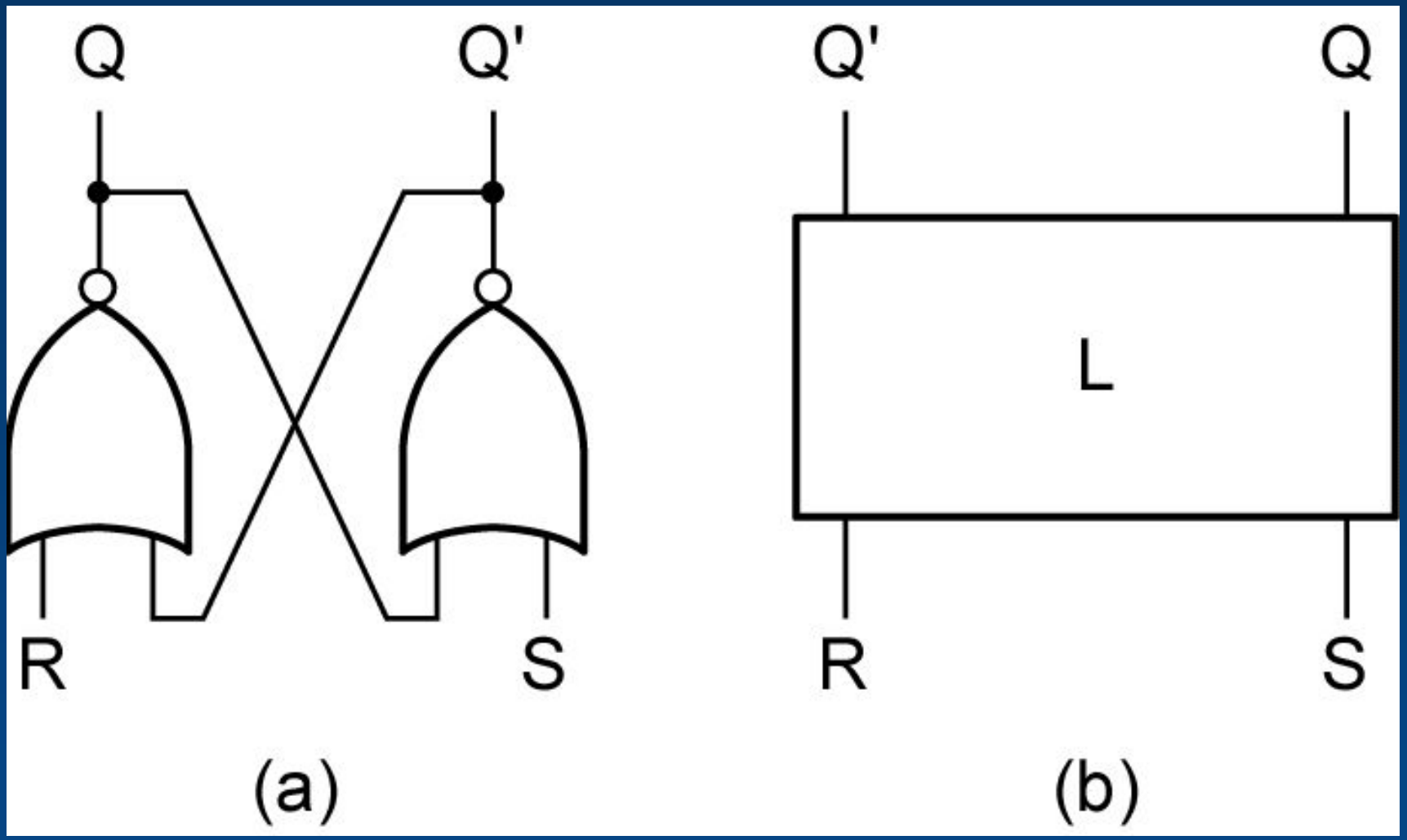
栓鎖器、正反器

國立台北科技大學
電機工程系
吳昭正

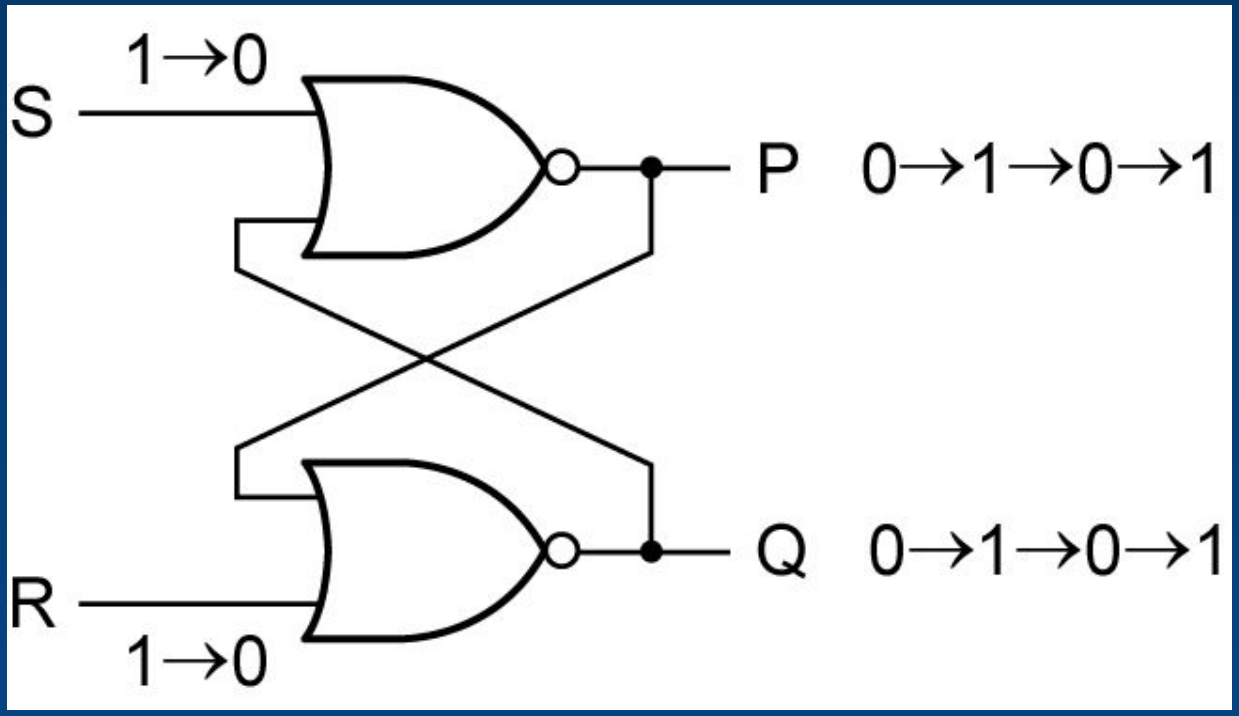
Outline

- 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)

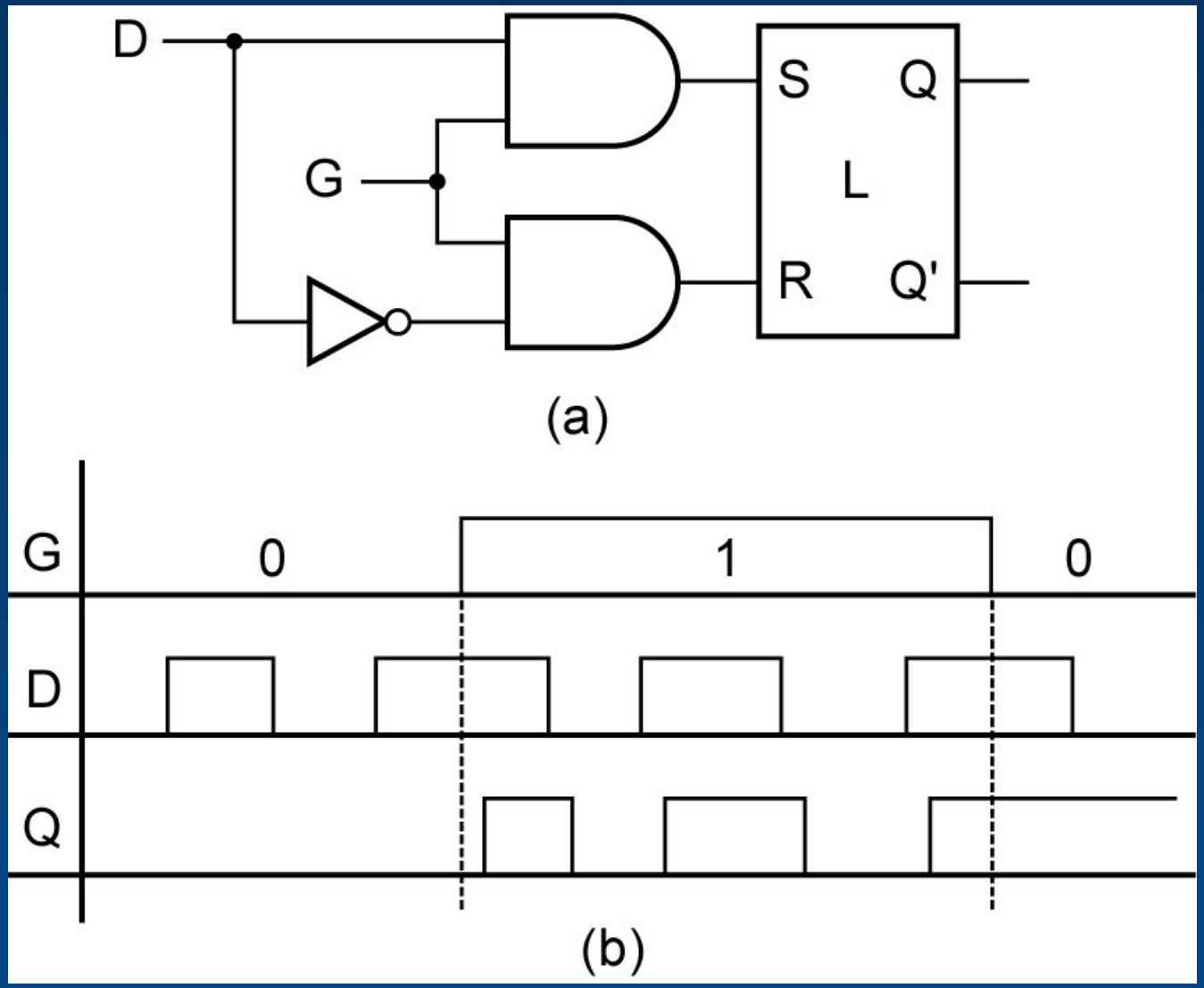


S-R Latch(2/2)

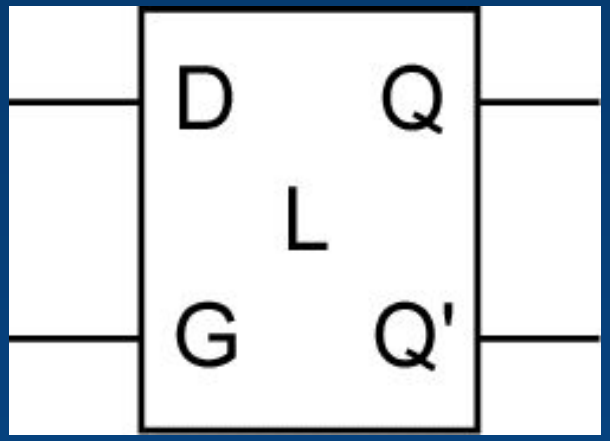


S R	Q^+
0 0	Q
0 1	0
1 0	1
1 1	Not allowed

D Latch(1/2)

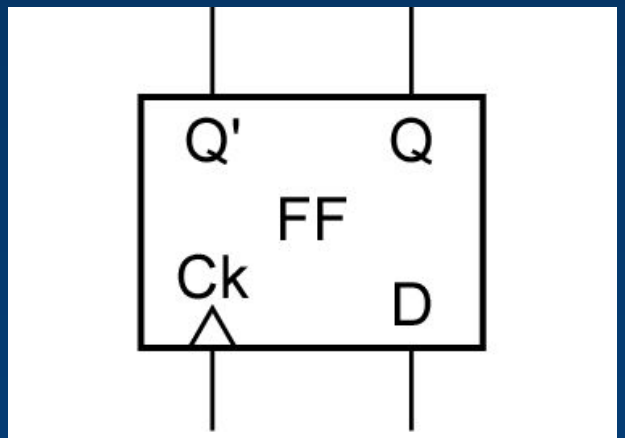


D Latch(2/2)

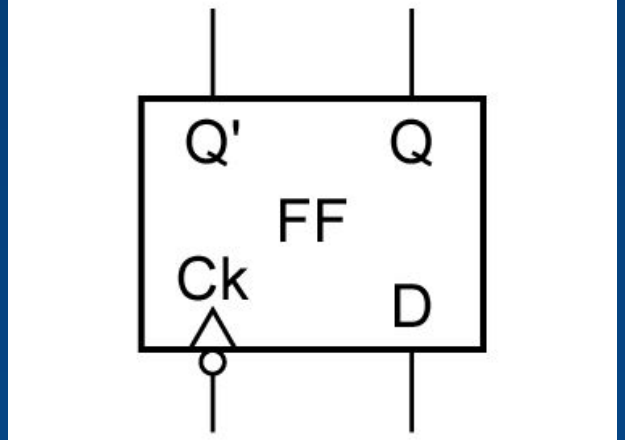


G	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

D Flip-Flop(1/2)



(a) Rising-edge trigger



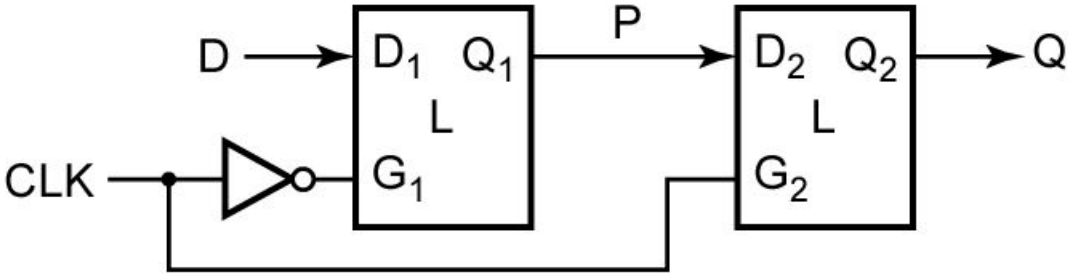
(b) Falling-edge trigger

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

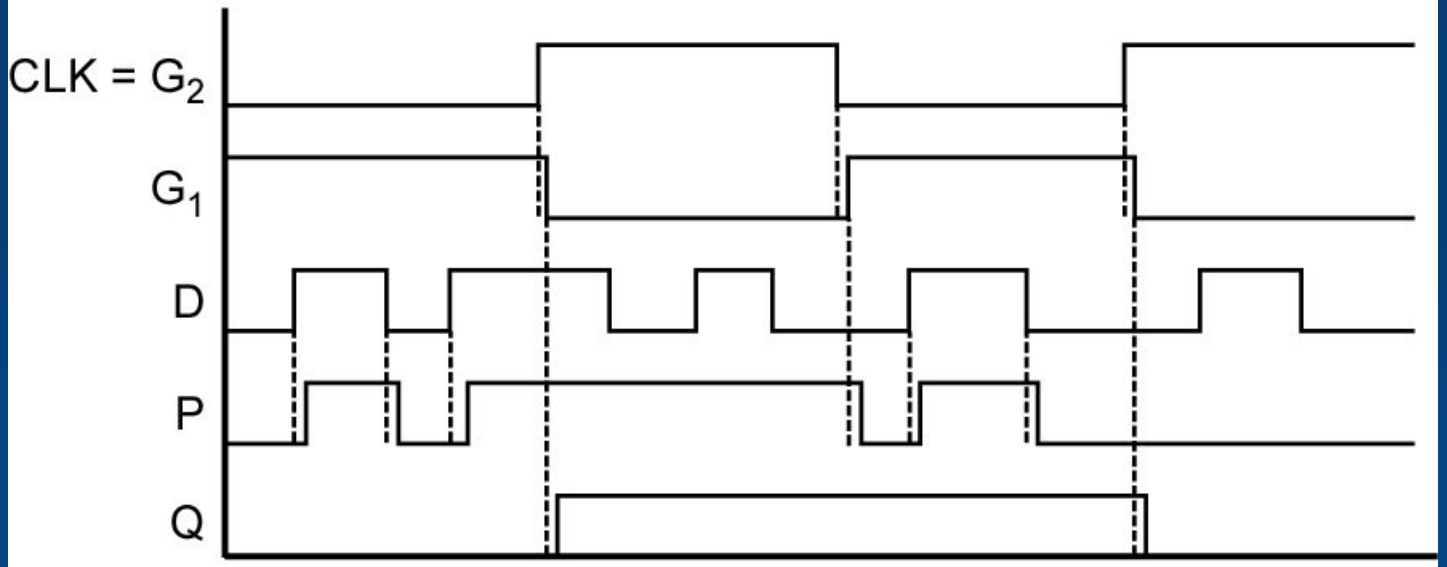
(c) Truth table

$Q^+ = D$

D Flip-Flop(2/2)

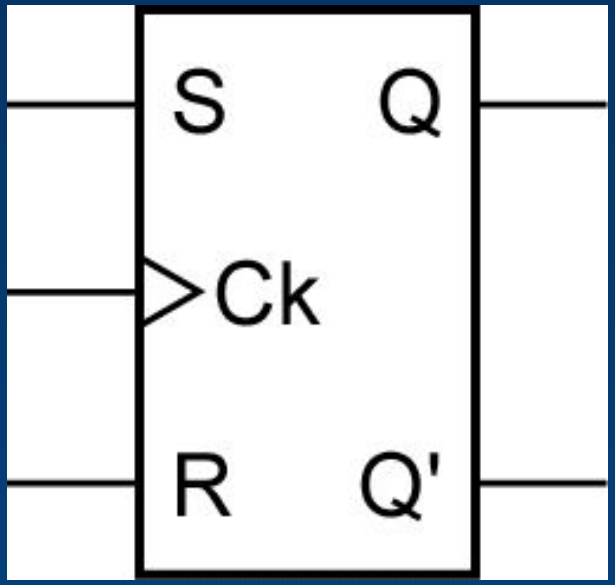


(a) Construction from two gated D latches



(b) Time analysis

S-R Flip-Flop(1/2)



Operation summary:

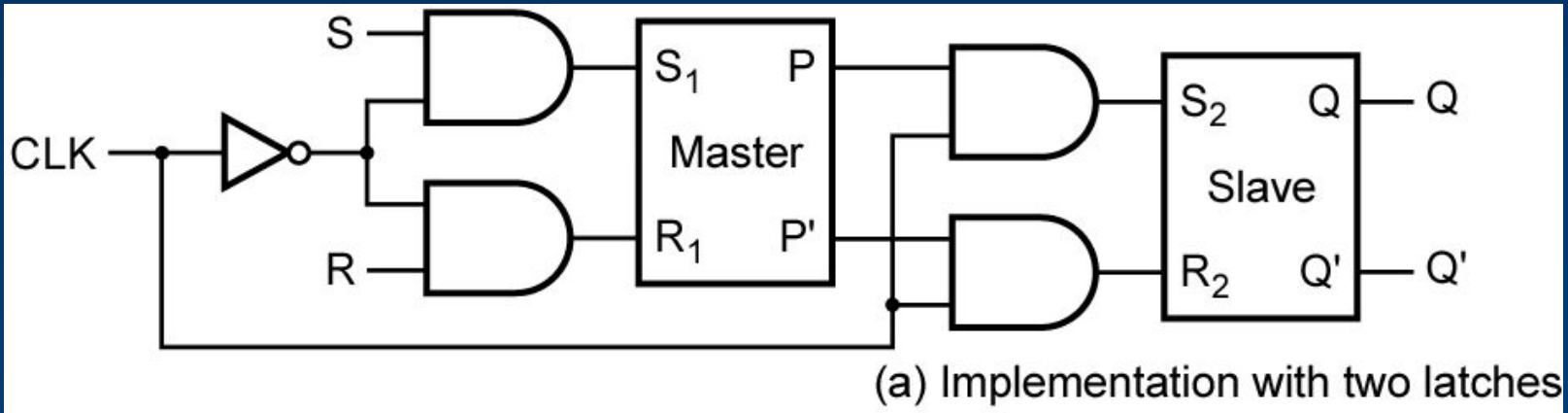
S = R = 0 no state change

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

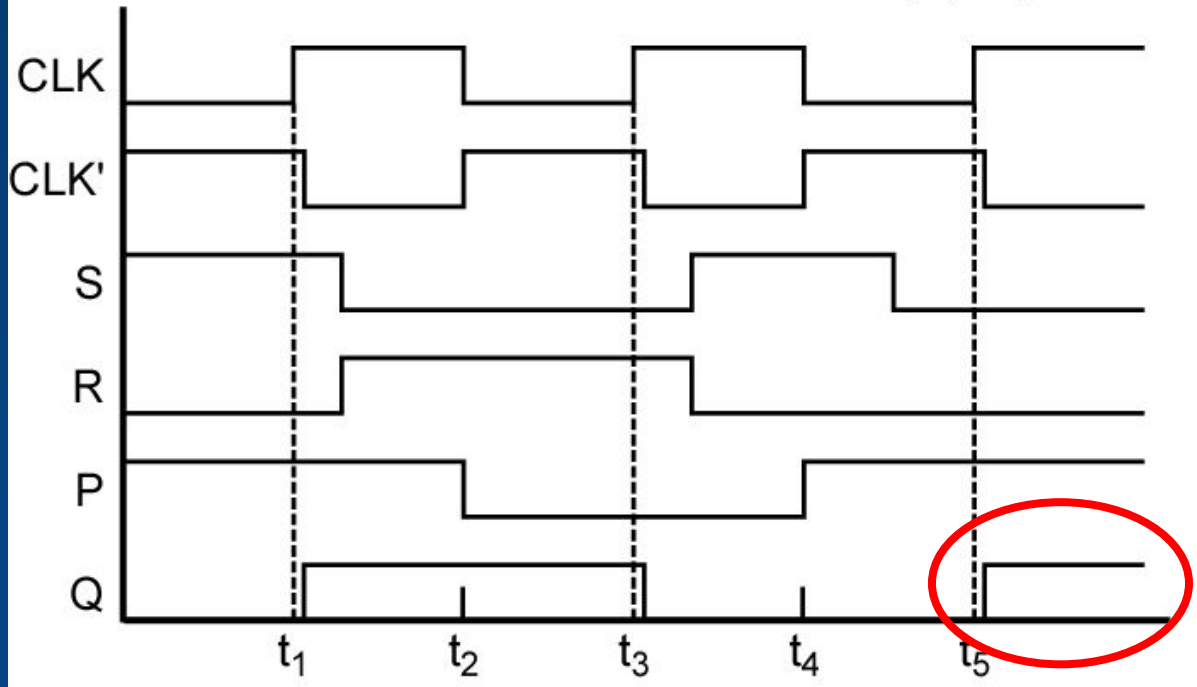
S = 0, R = 1 reset Q to 0 (after active Ck edge)

S = R = 1 not allowed

S-R Flip-Flop(2/2)

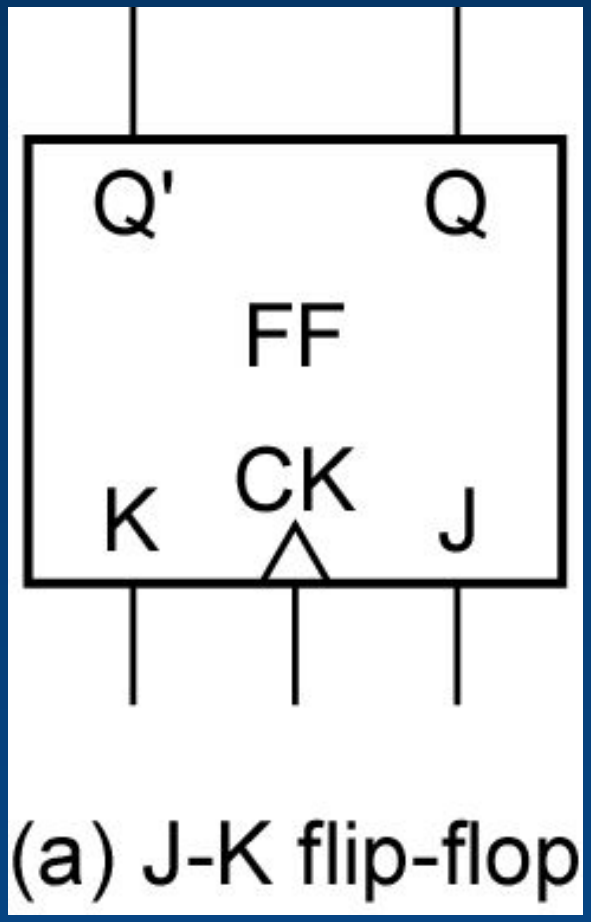


(a) Implementation with two latches



(b) Timing analysis

J-K Flip-Flop(1/2)

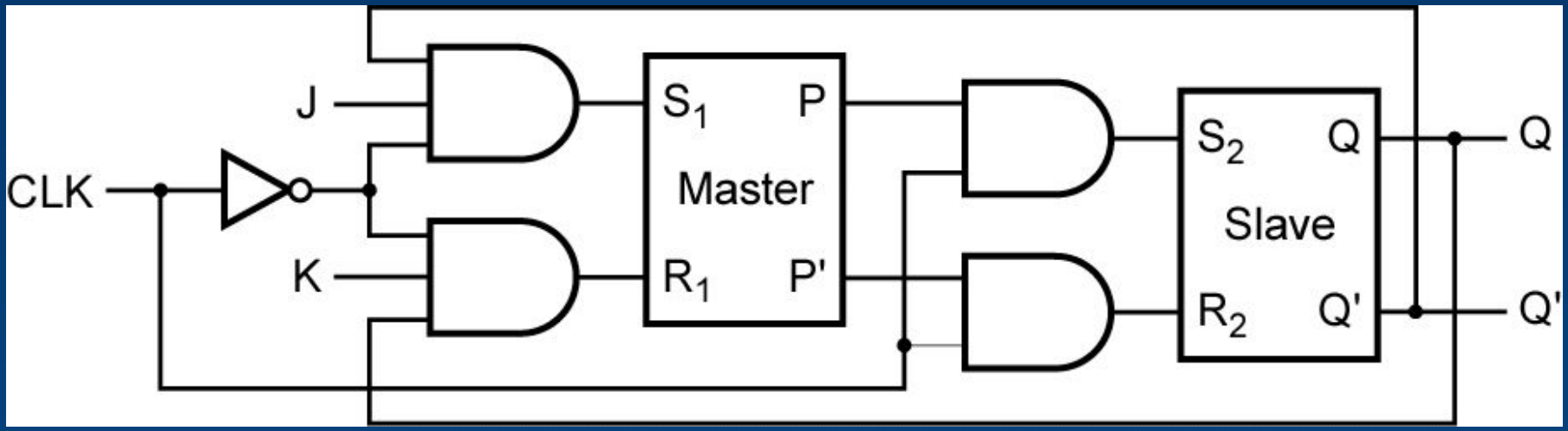


(a) J-K flip-flop

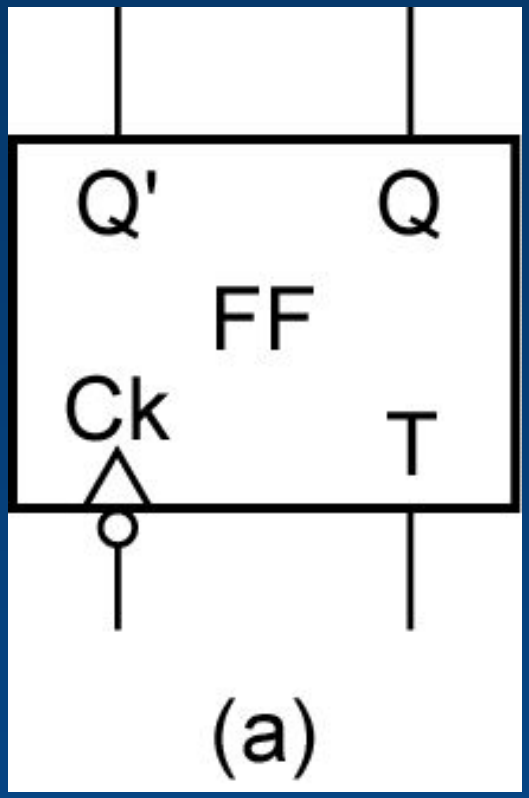
$$Q^+ = JQ' + K'Q$$

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

J-K Flip-Flop(2/2)



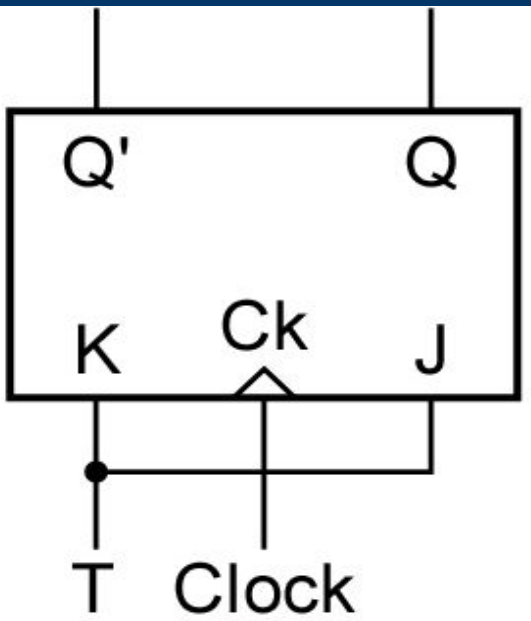
T Flip-Flop(1/2)



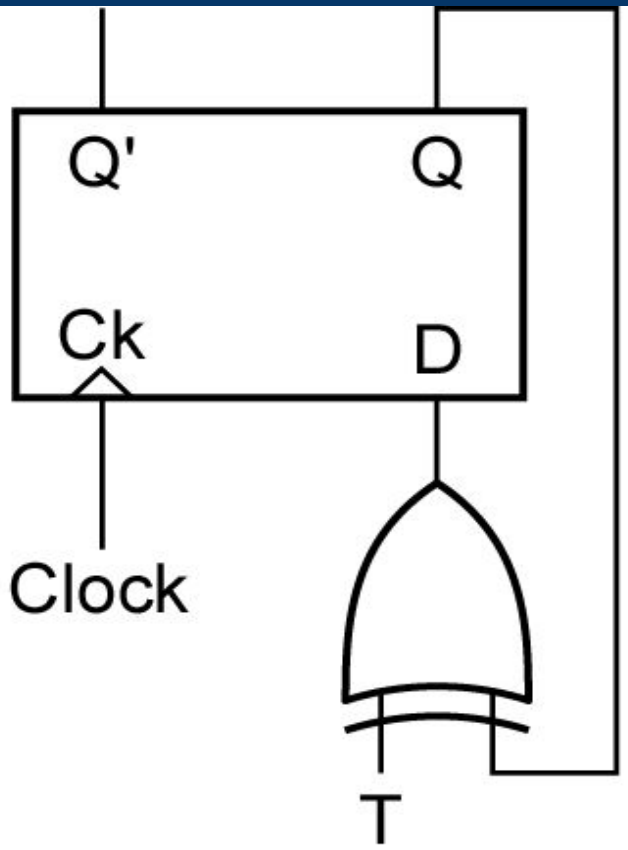
T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

$$Q^+ = T'Q + TQ' = Q \oplus T$$

T Flip-Flop(2/2)



(a) Conversion of J-K to T



(b) Conversion of D to T

Boolean Algebra for Flip-Flops

D flip-flop $Q^+ = D$ (13-1)

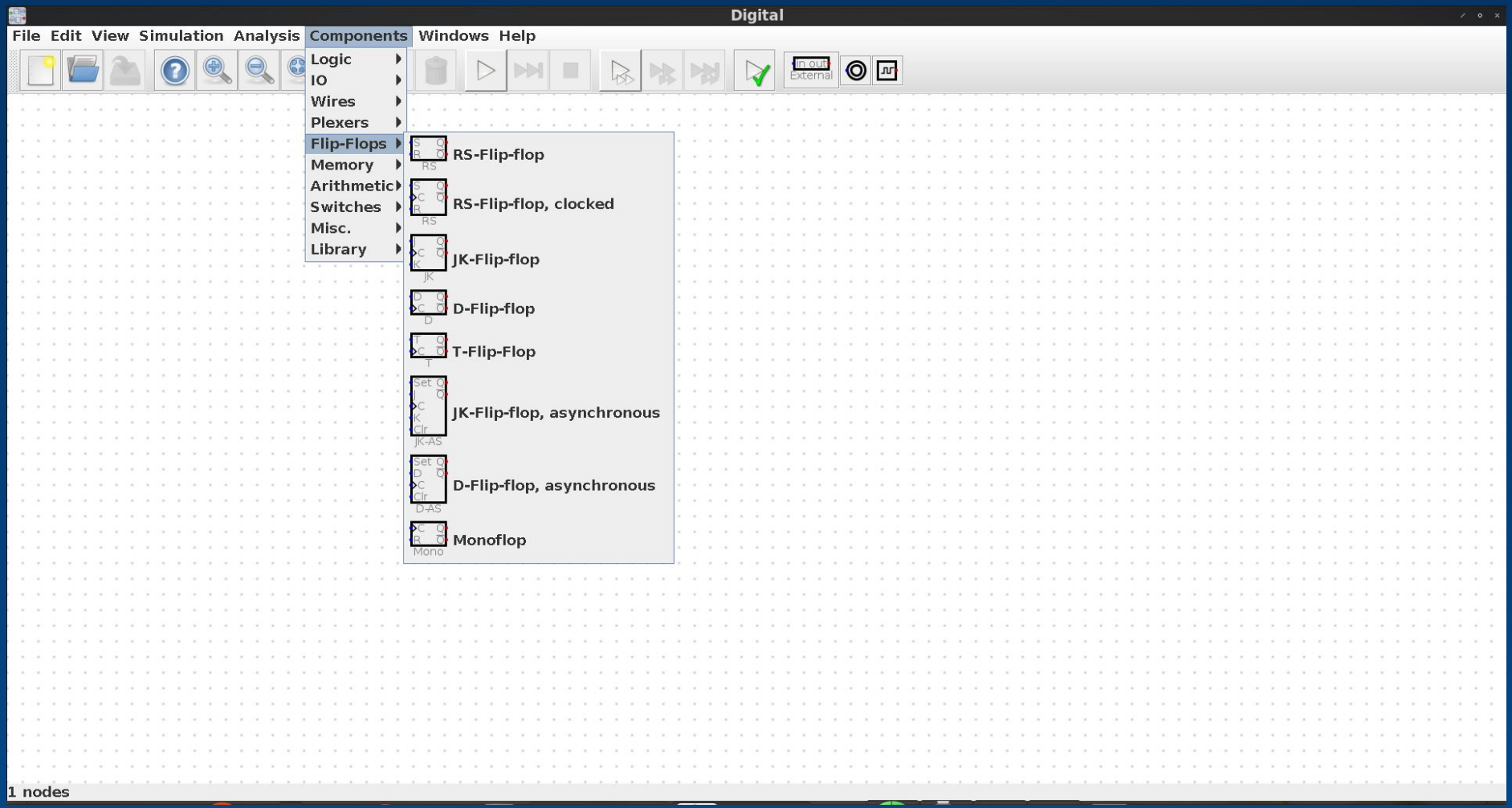
D-CE flip-flop $Q^+ = D \cdot CE + Q \cdot CE'$ (13-2)

T flip-flop $Q^+ = T \quad Q$ (13-3)

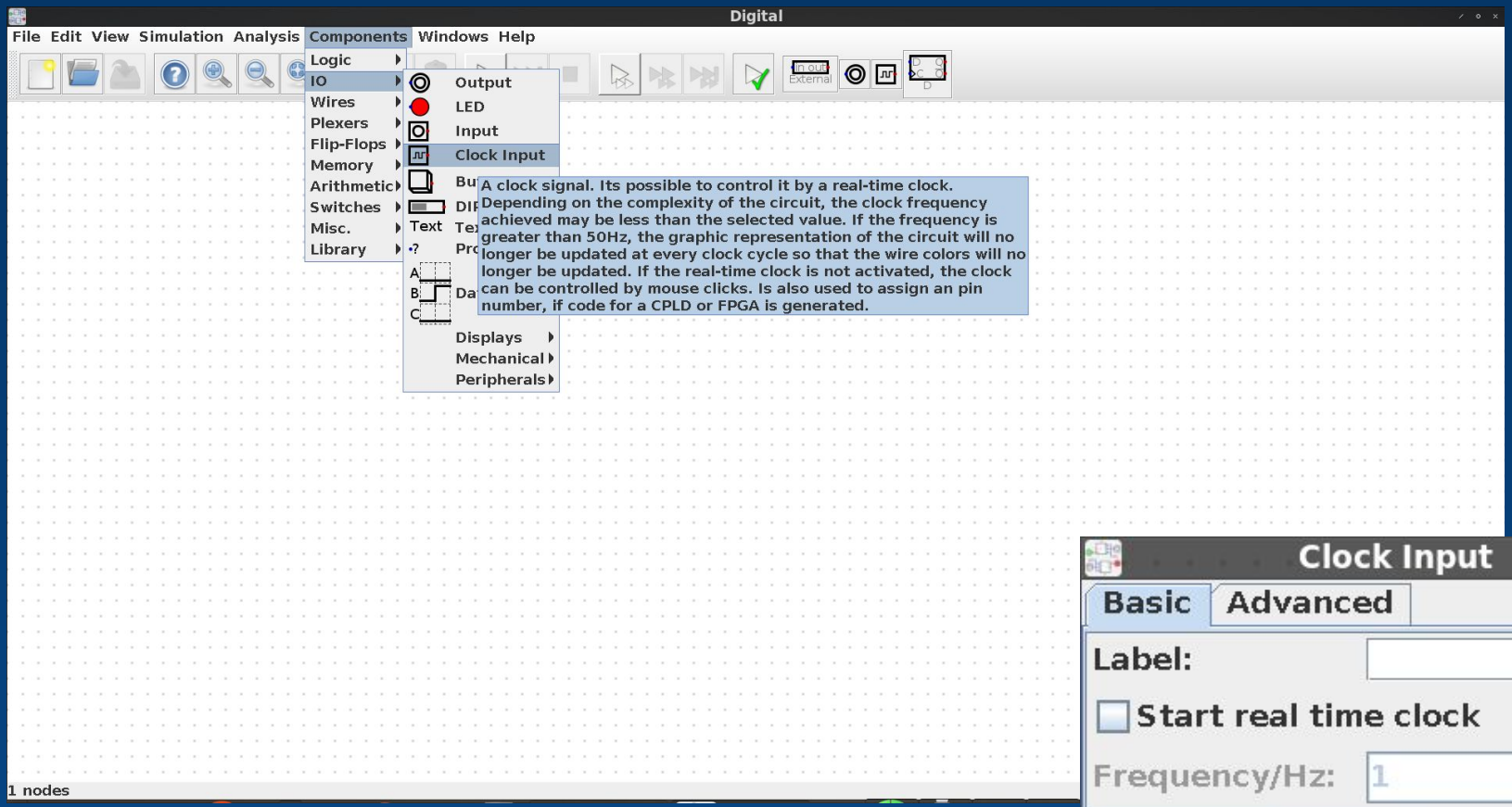
S-R flip-flop $Q^+ = S + R'Q$ (13-4)

J-K flip-flop $Q^+ = JQ' + K'Q$ (13-5)

Digital內建的正反器



Digital內的時脈產生器



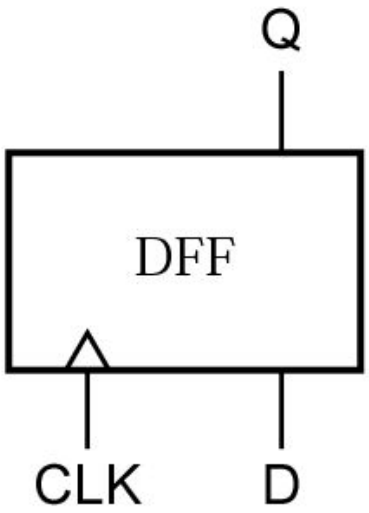
Modeling Flip-Flops Using Processes

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;
```

Figure 17-1: VHDL Code for a Simple D Flip-Flop

Modeling Flip-Flops Using Processes

A basic process has the following form:

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

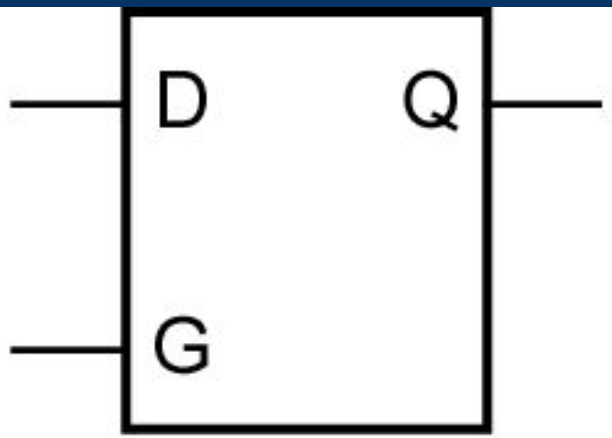
The basic **if** statement has the form:

```
if condition then
      sequential statements1
  else sequential statements2
end if;
```

Section 17.1 (p. 554)

VHDL語法介紹(04/14)

Modeling Flip-Flops Using Processes

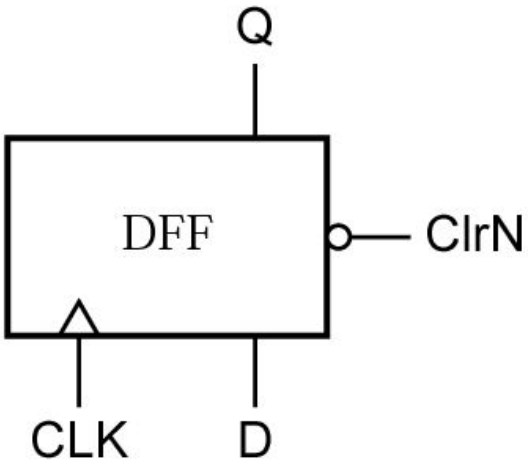



```
process (G,D)
begin
     if G = '1' then Q <= D; end  if;
end  process;
```

Figure 17-2: VHDL Code for a Transparent Latch

VHDL語法介紹(05/14)

Modeling Flip-Flops Using Processes




```
process (CLK, ClrN)
begin
    if ClrN = '0' then Q <= '0';
        else  if CLK'event and CLK = '1'
            then Q <= D;
            end  if;
    end  if;
end  process;
```

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

Modeling Flip-Flops Using Processes

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

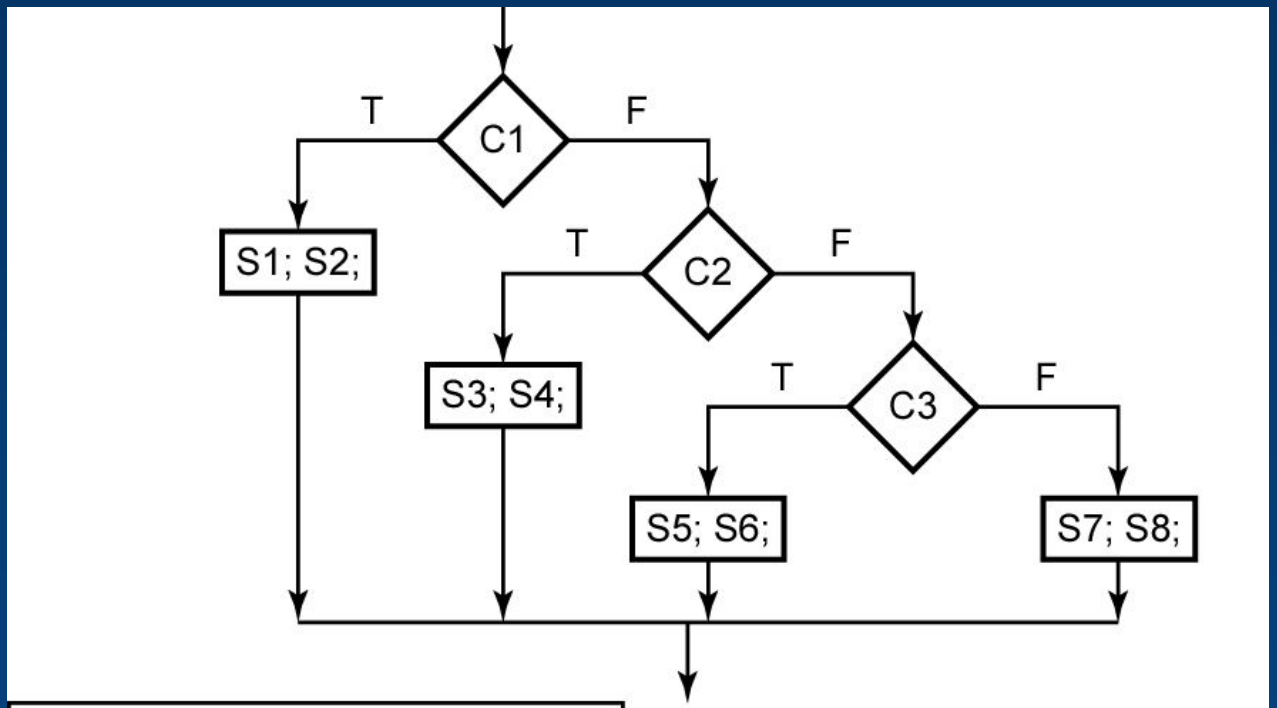
if condition **then**
    sequential statements
{**elsif** condition **then**
    sequential statements}
    -- 0 or more **elsif** clauses may be included
[**else** sequential statements]
end if;

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

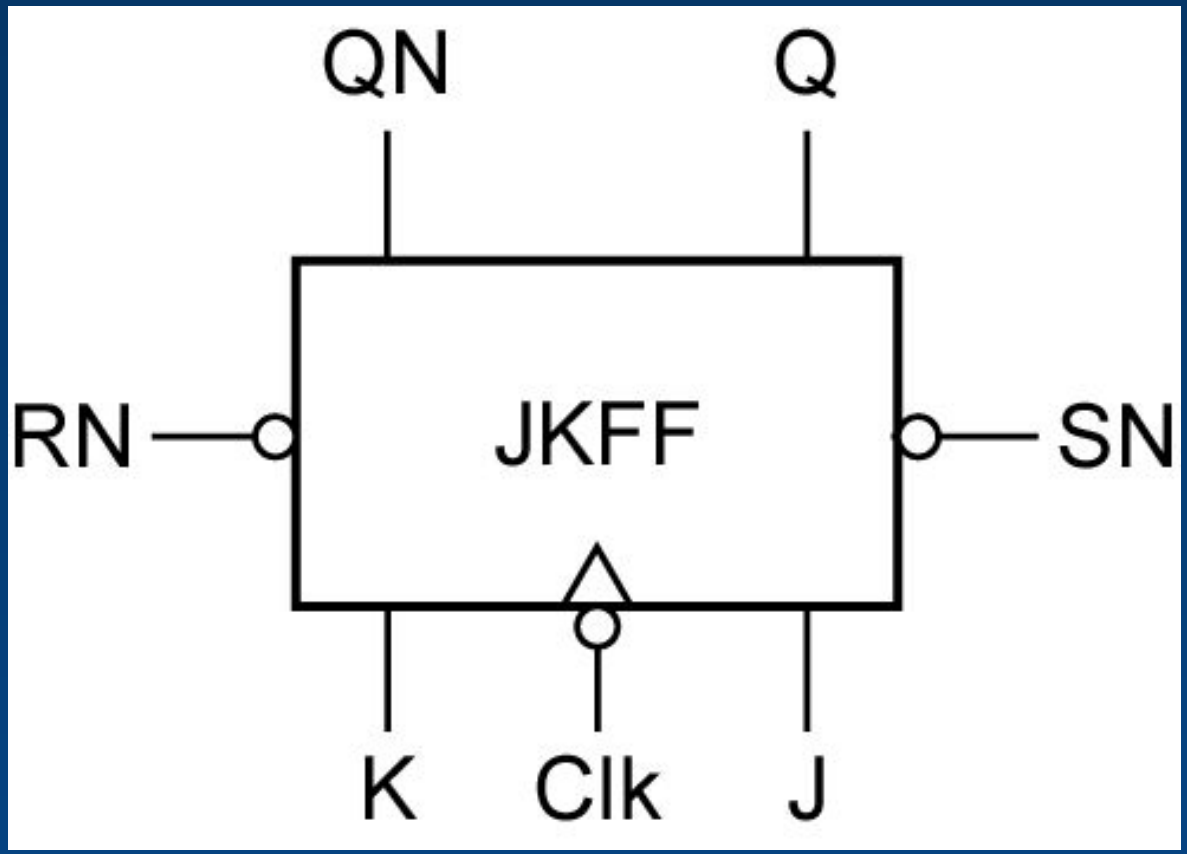



```
if (C1) then S1; S2;
    else if (C2) then S3; S4;
        else if (C3) then S5; S6;
            else S7; S8;
            end if;
        end if;
end if;
```

```
if (C1) then S1; S2;
    elsif (C2) then S3; S4;
    elsif (C3) then S5; S6;
    else S7; S8;
end if;
```

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

Modeling Flip-Flops Using Processes



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

Figure 17-5: J-K Flip-Flop

VHDL語法介紹(09/14)

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;
```

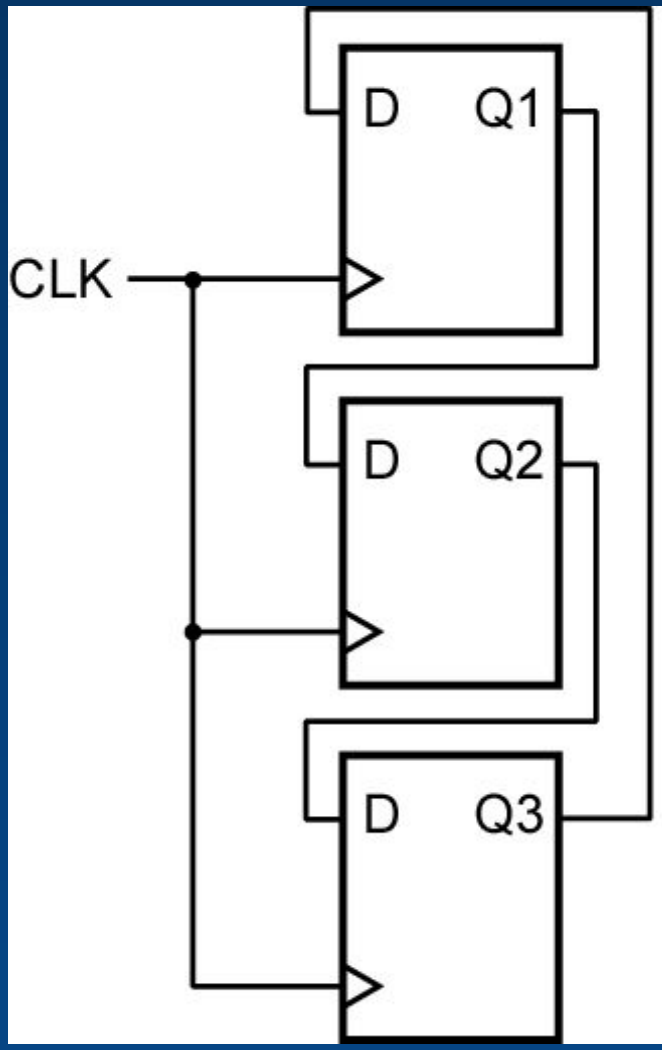
VHDL語法介紹(10/14)

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)

Modeling Flip-Flops Using Processes

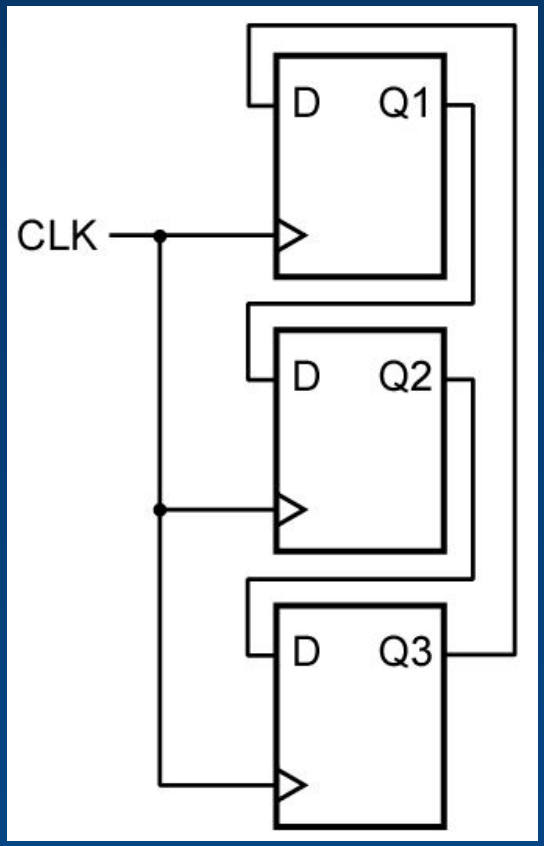



```
process (CLK)
begin
    if CLK'event and CLK = '1' then
        Q1 <= Q3 after 5 ns;
        Q2 <= Q1 after 5 ns;
        Q3 <= Q2 after 5 ns;
    end if;
end process;
```

Figure 17-7: Cyclic Shift Register

VHDL語法介紹(12/14)

Modeling Flip-Flops Using Processes

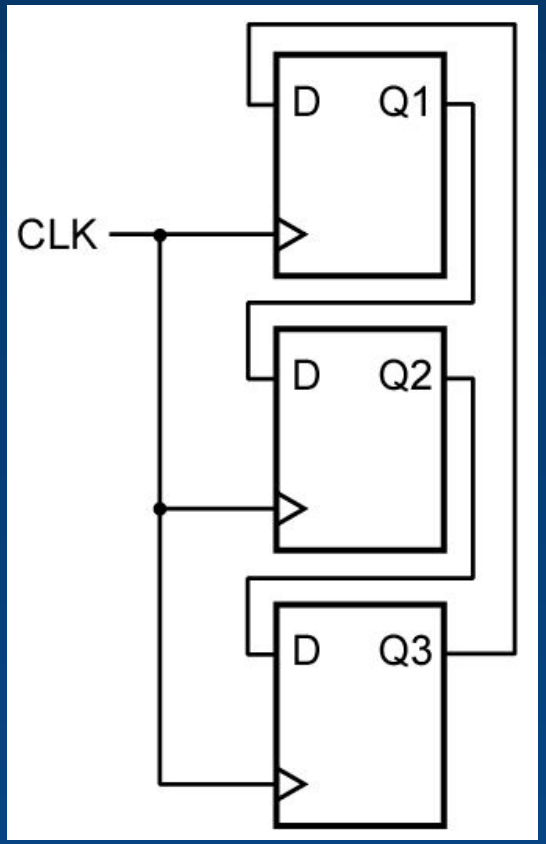



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

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 <= D after 5 ns;
            end if;
      end process;
end DFF_simple;
```

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

VHDL語法介紹(13/14)

Modeling Flip-Flops Using Processes

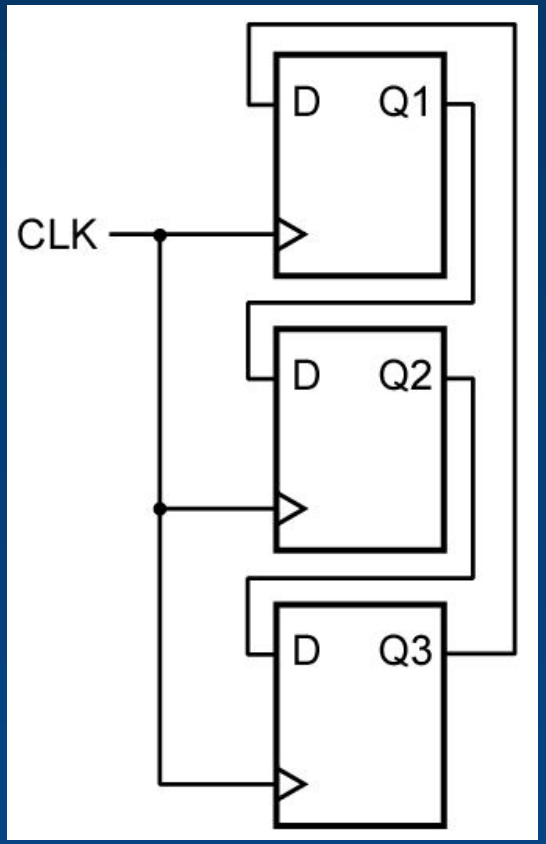



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity cyclicSR is
port (clk: in std_logic;
      Qout: out std_logic_vector(1 to 3));
end cyclicSR;
```

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

VHDL語法介紹(14/14)

Modeling Flip-Flops Using Processes




```
architecture cyclicSR3 of cyclicSR is
component my_DFF
        port (D, clk: in std_logic;
        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;
```

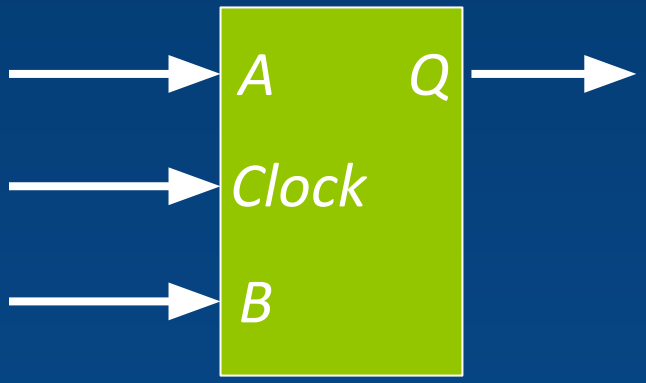
Figure 17-8b: Structural VHDL Code for Cyclic Shift Register

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’
1 1	1