


數位系統導論實驗 LAB 5

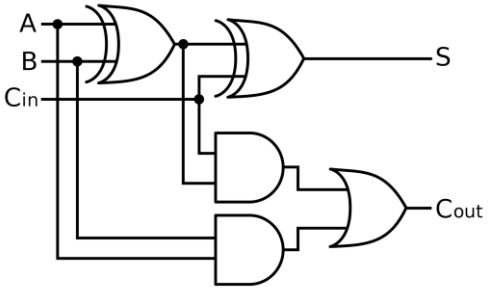
梁郁珮

2022/04/18

2022/04/20

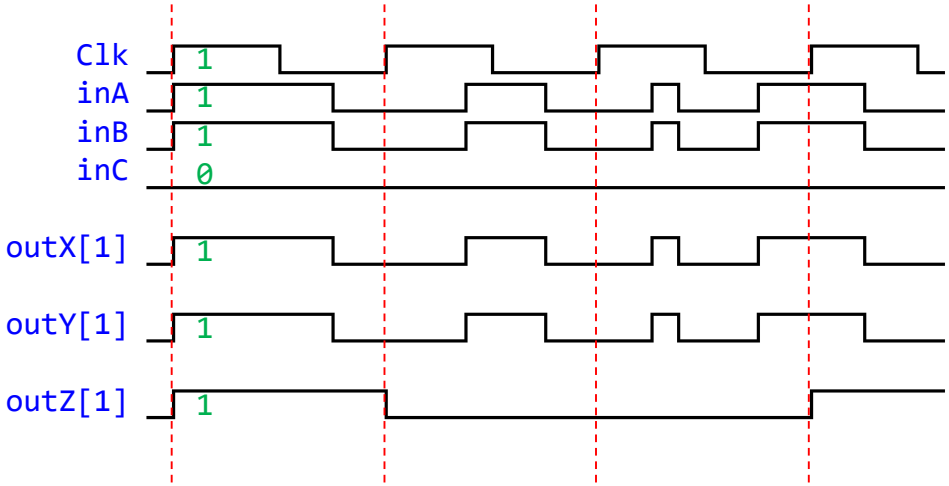
VHDL Modelling Styles

- Structure Description
- Dataflow Description
- Behavior Description

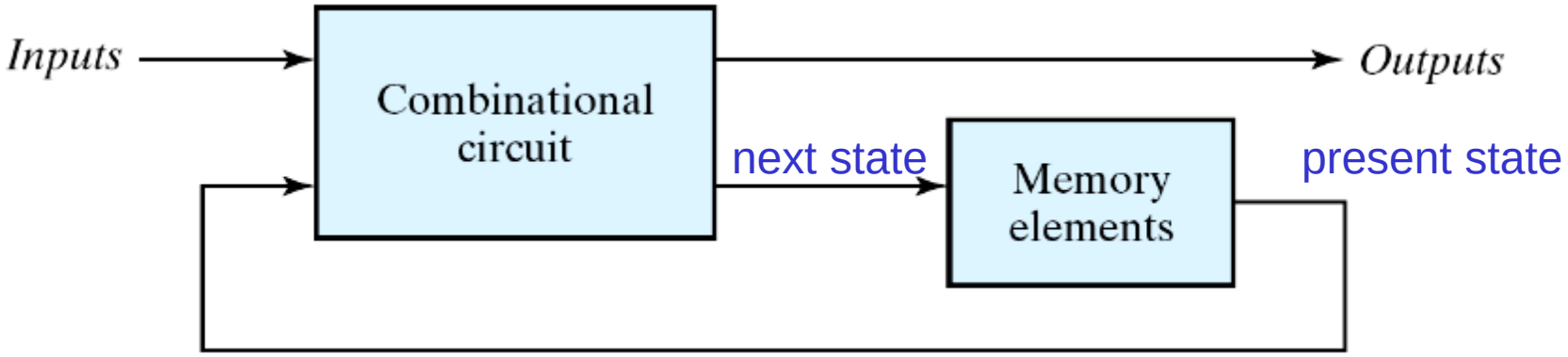


FA

```
module LAB(clk, inA, inB, inC, outY, outX, outZ);
//-------Input ports-------
input  clk, inA, inB, inC;
//-----Structure-----
output [1:0]outX;
FA   x0(.A(inA), .B(inB), .Cin(inC), .S(outX[0]), .Cout(outX[1]));

//-----Dataflow -----
output [1:0]outY;
assign outY = inA + inB + inC;

//-----Behavior -----
output [1:0]outZ;
always@(posedge clk)
begin
  outZ = inA + inB + inC;
end

endmodule
```



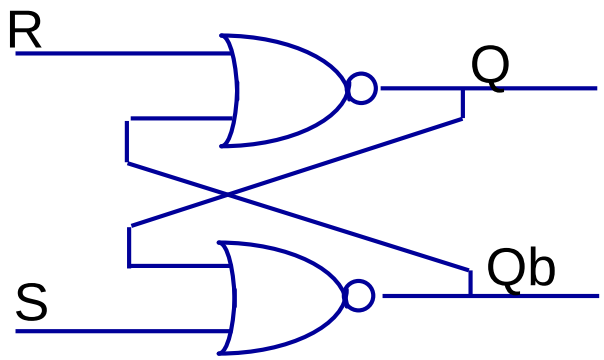
Sequential



- A feedback path
- The state of the sequential circuit
- (inputs, current state) $\Rightarrow$ (outputs, next state)
- **Synchronous**: the transition happens at discrete instants of time
- **Asynchronous**: at any instant of time

SR latch

NOR:
'0' if either of its inputs are '1'

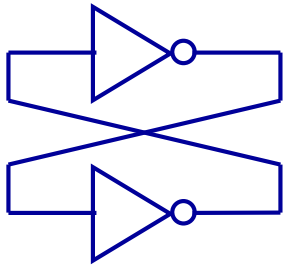


S = set
R = reset
Q = output

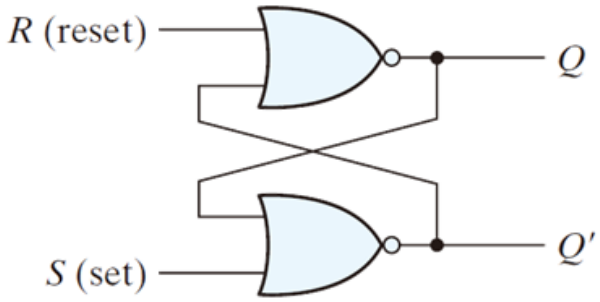
S	R	Q	Qb
1	0	1	0
0	1	0	1
0	0	hold	hold
1	1	0	0

When a NOR has one input at '0', it inverts the other input

When S and R are 0



SR latch type

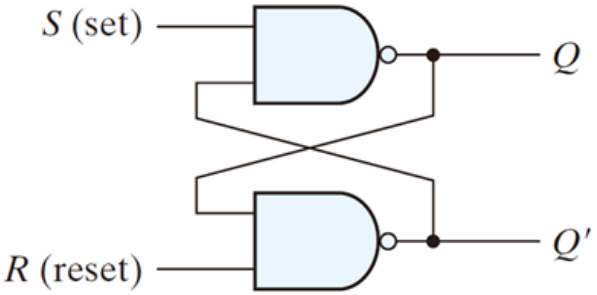


(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

FIGURE 5.3
SR latch with NOR gates



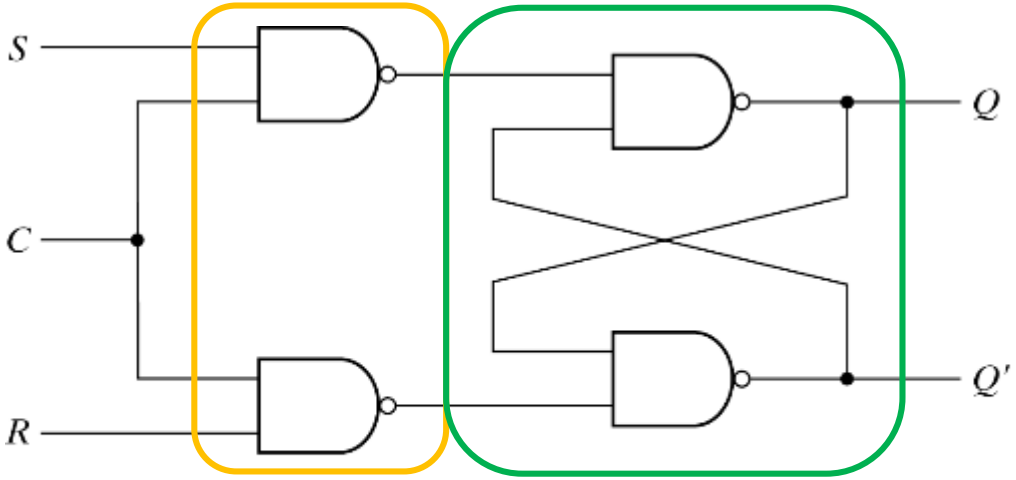
(a) Logic diagram

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

(b) Function table

FIGURE 5.4
***SR* latch with NAND gates**

SR latch w control



(a) Logic diagram

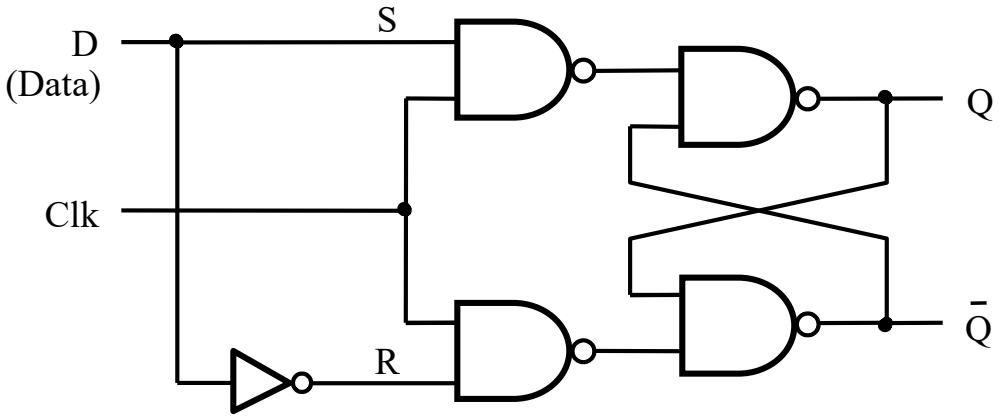
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; set state
1	1	1	Indeterminate

(b) Function table

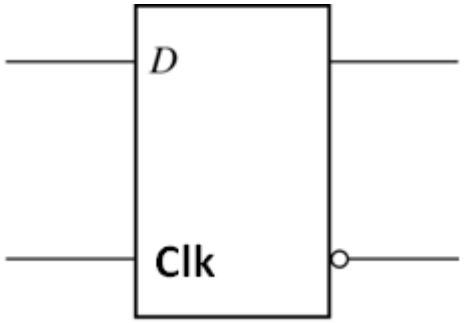
Fig. 5-5 SR Latch with Control Input

- Occasionally, desirable to avoid latch changes
- C = 0 disables all latch state changes
- Control signal enables data change when C = 1
- Right side of circuit same as ordinary S-R latch.

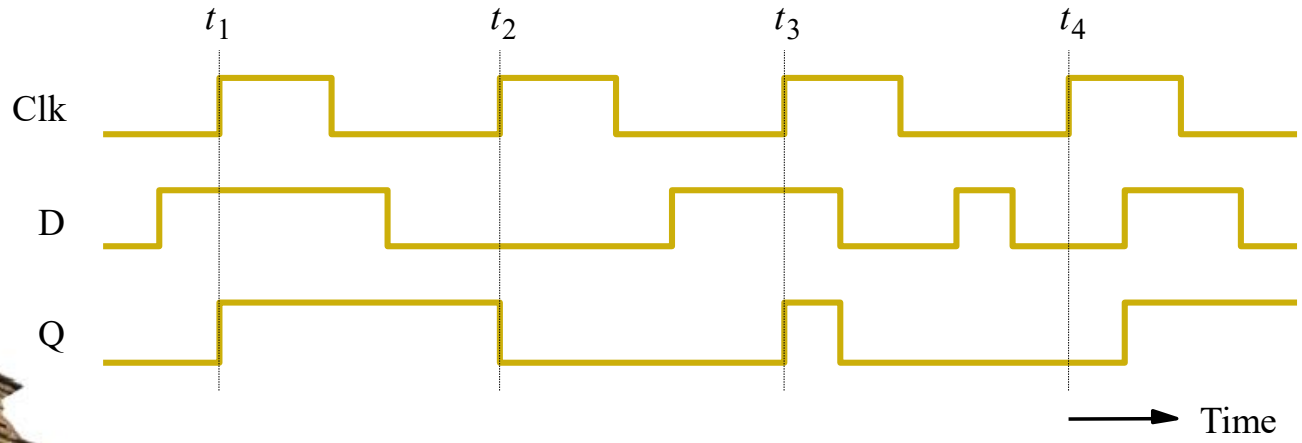
D-Latch



(a) Circuit

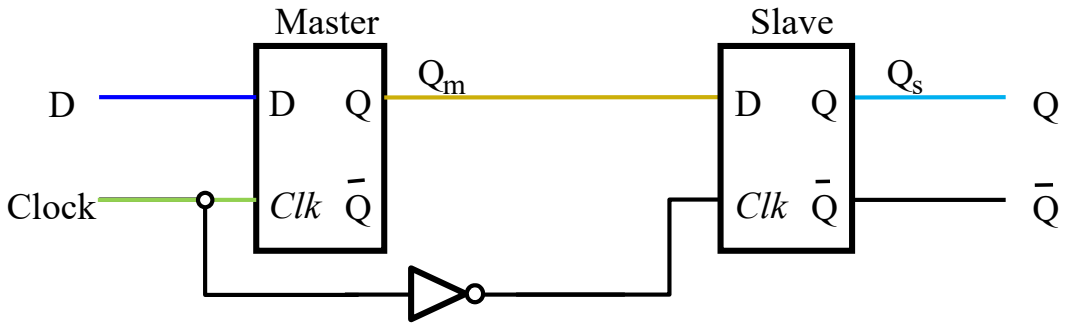


(b) Graphical symbol

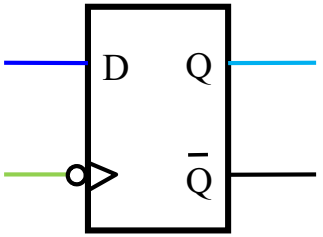


(c) Timing diagram

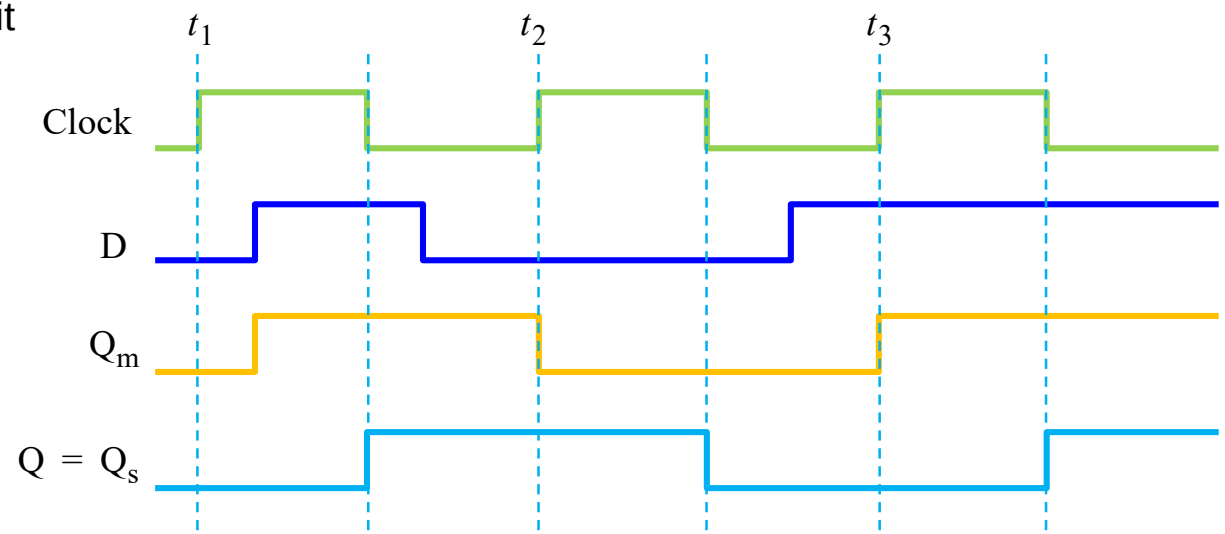
Master-Slave D Flip Flop (neg.)



(a) Circuit

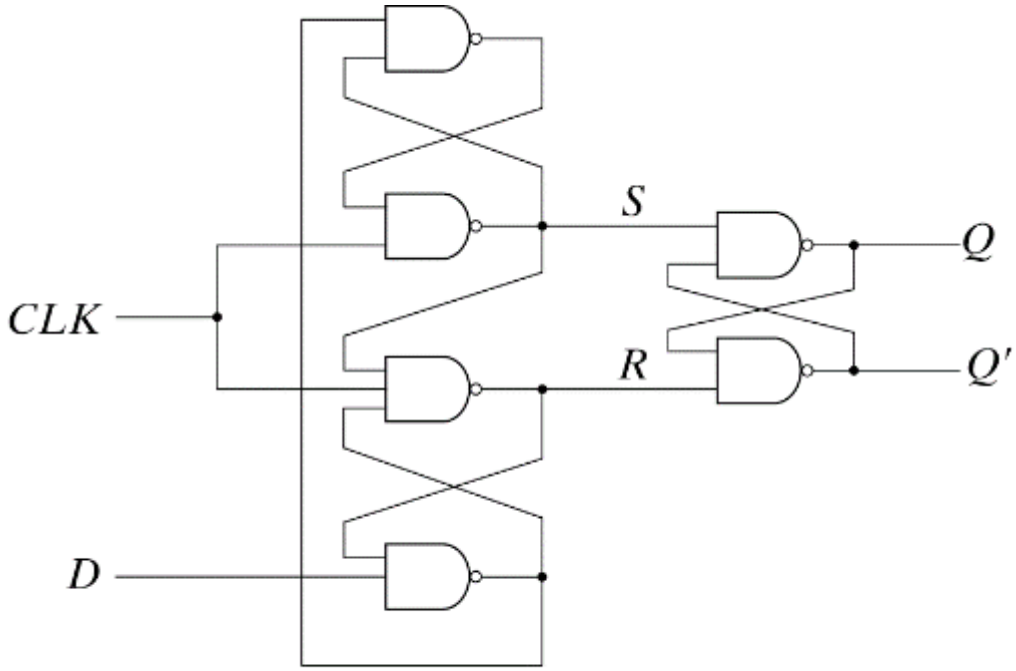


(b) Graphical symbol



(c) Timing diagram

Master-Slave D Flip Flop (pos.)



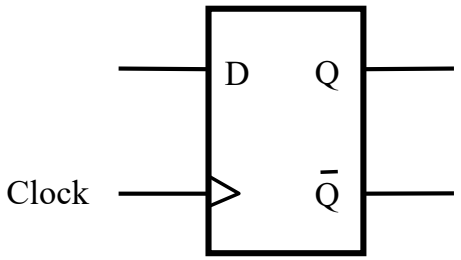
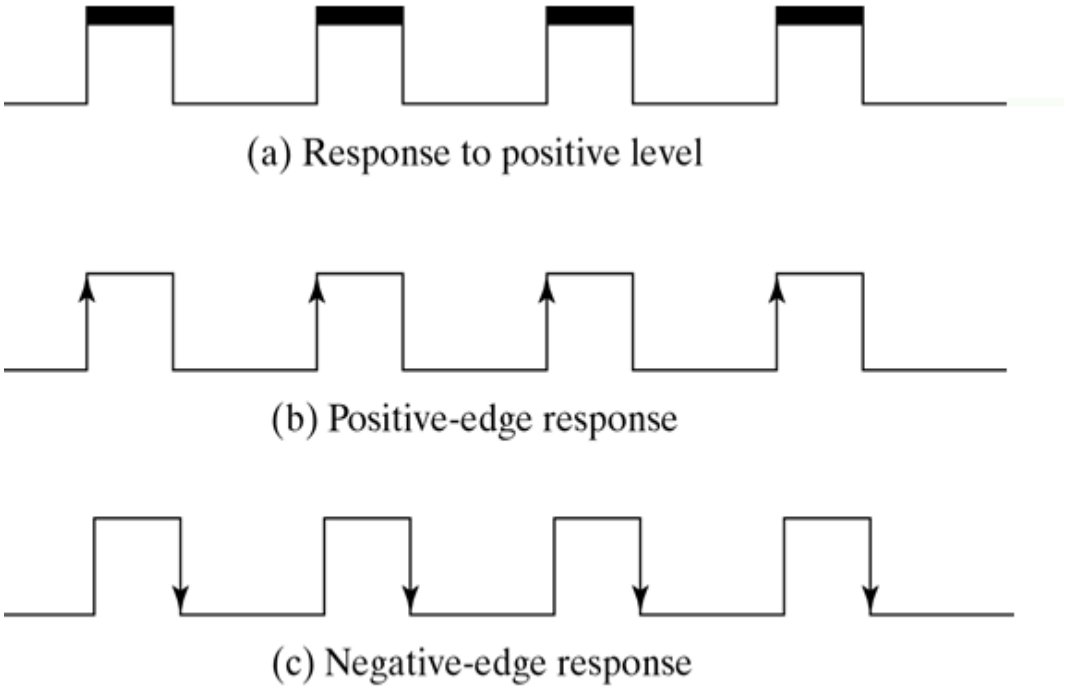


Fig. 5-10 *D*-Type Positive-Edge-Triggered Flip-Flop

Latch & Flip-Flop

The state of a latch or flip-flop is switched by a change of the control input

- Level triggered – latches
- Edge triggered – flip-flops



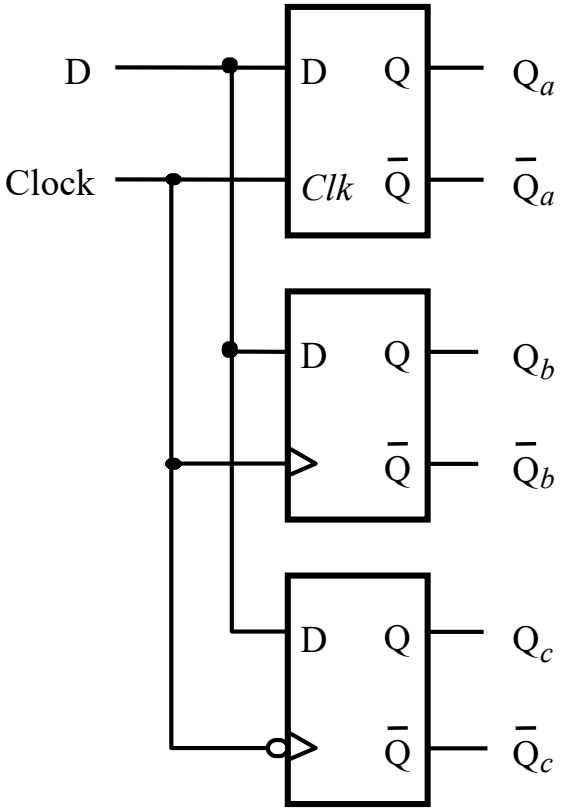
(a) Response to positive level

(b) Positive-edge response

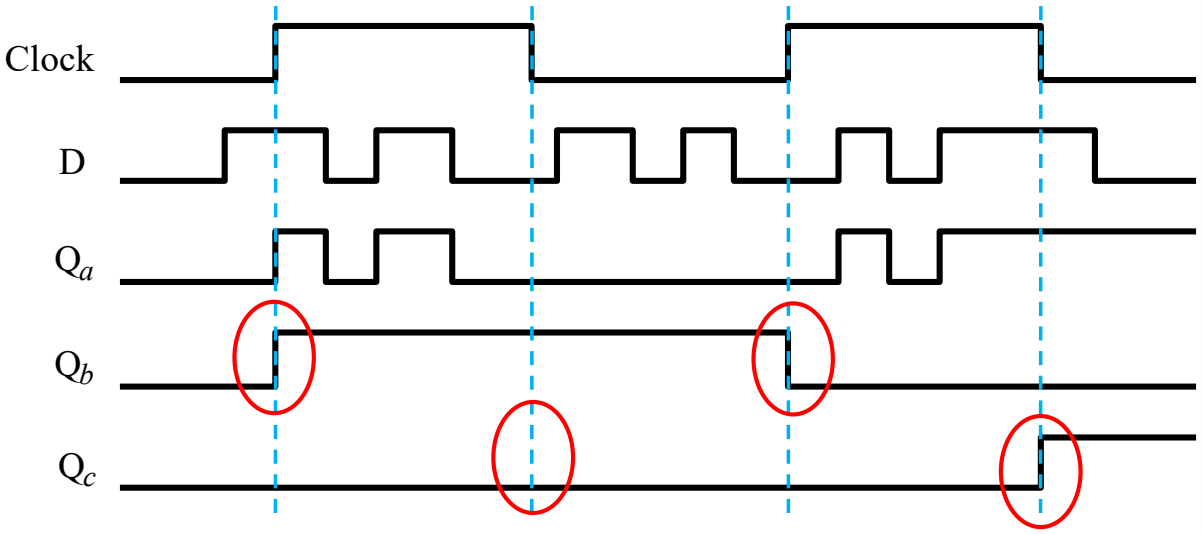
(c) Negative-edge response

Fig. 5-8 Clock Response in Latch and Flip-Flop

Waveform



(a) Circuit



(b) Timing diagram

Lexical conventions

Number format:

- Sized: default size is 32-bits decimal number

  <bit length> ' <base><value>

```
reg  d  =  4'b1010  ;          // sized constant
reg  d  =  4'd7   ;            // sized constant
```

bit length	the length of desired value in bits.
base	It can be b (binary), o (octal), d (decimal) or h (hexadecimal)
value	It is any legal number in the selected based

- Unsized: default size is 32-bits

```
reg  d  =  7;                  // unsized constant, output bit length=32
```

```
reg [11:0]a;
a <= a + 1;
```

```
10230 Verilog HDL assignment warning at LZ77_Encoder.v(358): truncated value with size 32 to match size of target (12)
10230 Verilog HDL assignment warning at LZ77_Encoder.v(380): truncated value with size 32 to match size of target (12)
10230 Verilog HDL assignment warning at LZ77_Encoder.v(402): truncated value with size 32 to match size of target (12)
10230 Verilog HDL assignment warning at LZ77_Encoder.v(424): truncated value with size 32 to match size of target (12)
10230 Verilog HDL assignment warning at LZ77_Encoder.v(446): truncated value with size 32 to match size of target (12)
10230 Verilog HDL assignment warning at LZ77_Encoder.v(468): truncated value with size 32 to match size of target (12)
10230 Verilog HDL assignment warning at LZ77_Encoder.v(490): truncated value with size 32 to match size of target (12)
```

Lexical conventions(附錄)

Logical operators	
a && b	evaluates to true if a *and* b are true
a || b	evaluates to true if a *or* b are true
!a	converts non-zero value to zero, and vice versa

Shift operators	
a >> n a << n	logical shift right (left) n bit
a >>> n a <<< n	arithmetic shift right (left) n bit

Arithmetic operators	
a + b	a plus b
a - b	a minus b
a * b	a multiplied by b
a / b	a divided by b
a % b	a modulo b
a ** b	a to the power of b

Relational operators	
a < b	a less than b
a > b	a greater than b
a <= b	a less than or equal to b
a >= b	a greater than or equal to b

Bitwise operators	
a & b	a *and* b
a | b	a *or* b

Equality operators	
a === b	a equal to b, including x and z
a !== b	a not equal to b, including x and z
a == b	a equal to b, result can be unknown
a != b	a not equal to b, result can be unknown

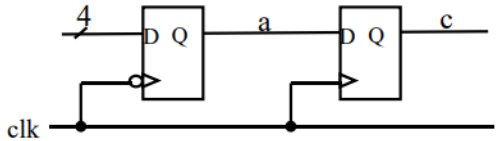
請同學自行參考 ~

Data assignment

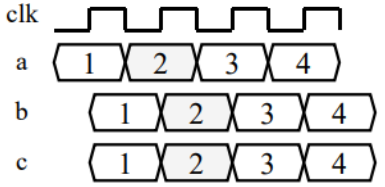
- **Blocking assignment**
  - Evaluated and assigned in a single step
  - Execution flow within the procedure is blocked until the assignment is completed

```
reg b, c;
always@(posedge clk)
begin
   b = a;
   c = b;
end
```

Wrong



register b has been combined and equivalent to c in the data behavior.



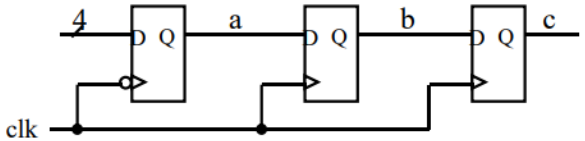
- **Non-Blocking assignment**
  - Evaluated and assigned in two steps
    - The right-hand side is evaluated immediately
    - The assignment to the left hand side is postponed until other evaluations in the current time step are completed

SWAP

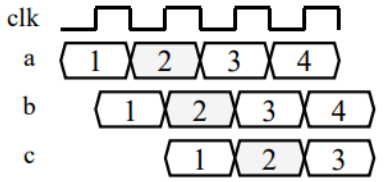
```
always @( posedge clk )
  begin
     A <= B;
     B <= A;
  end
```

```
reg b, c;
always@(posedge clk)
begin
   b <= a;
   c <= b;
end
```

Correct



Data can be passed sequentially from register b to c



Data Shift!!

Conditional Statement

```
if (<expression>)
    <statement>
else if (<expression>)
    <statement>
else if (<expression>)
    <statement>
else (<expression>)
    <statement>
```

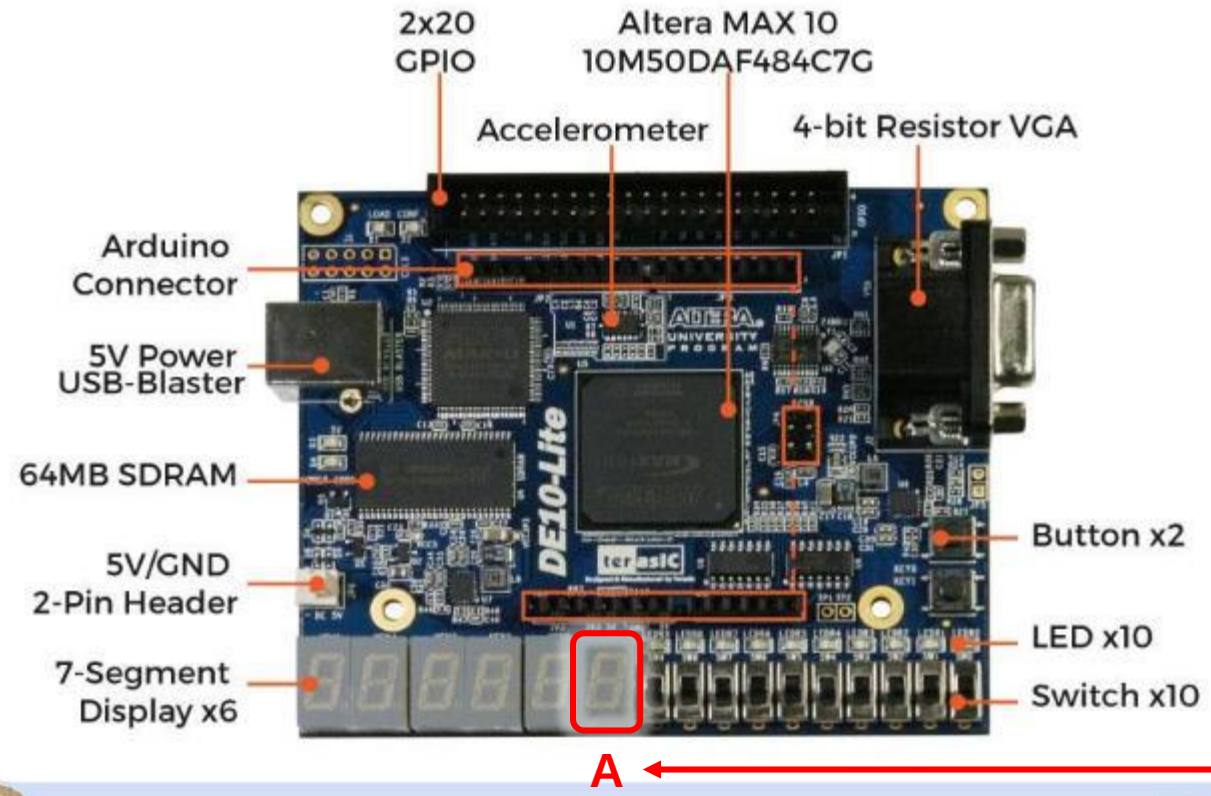
```
if(cond[3])
    out=0
else if(!cond[3] & cond[2])
    out=1
else if(!cond[3] & !cond[2] & cond[1])
    out=2
else if(!cond[3] & !cond[2] & !cond[1] & cond[0])
    out=3
else if(!cond[3] & !cond[2] & !cond[1] & !cond[0])
    out=4
```

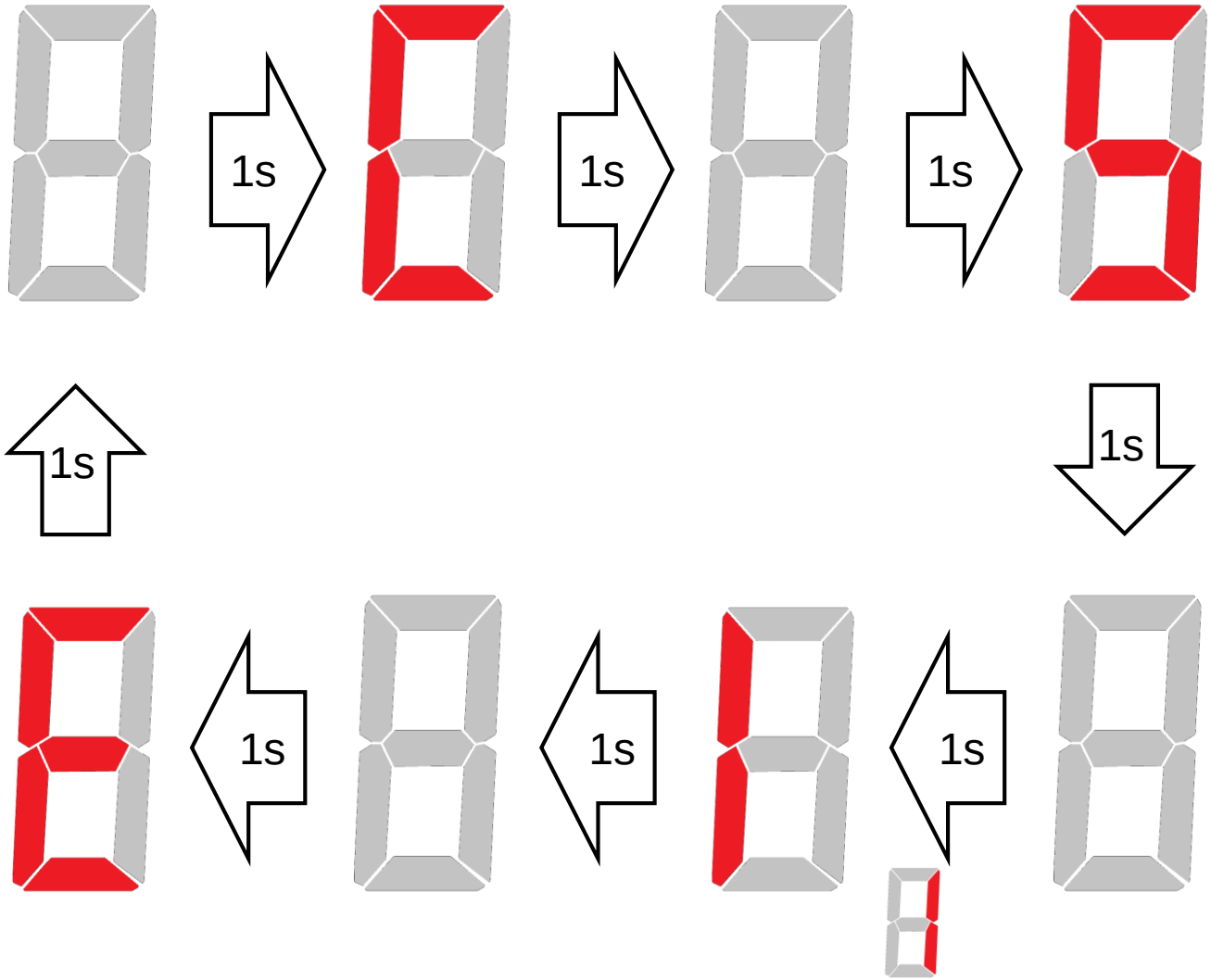
```
case (<expression>)
    altarnative1: <statement>
    altarnative2: <statement>
    altarnative3: <statement>
    ……
    ……
    default: <default statement>
endcase
```

```
case (cond.)
    4'b1???:
        out=0
    4'b01??:
        out=1
    4'b001?:
        out=2
    4'b0001:
        out=3
    4'b0000:
        out=4
endcase
```

實 驗 項 目

- 使用內建50MHz時脈練習顯示「閃爍」CSIE 於A，時間間隔為1s亮一下

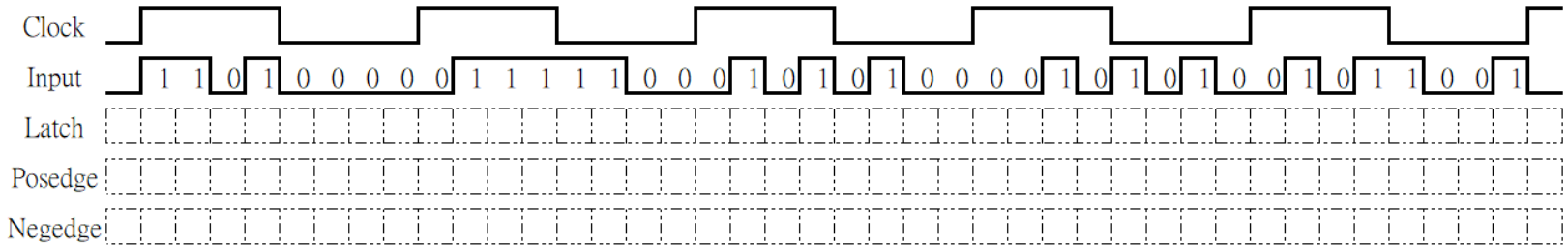



1s
1s
1s
1s
1s
1s
1s
1s

REPORT OF LAB 5 (4%)

The report should include …

1. 完成下表
2. 在conditional statement中，若把if的條件式交換順序，答案還會一樣嗎？為什麼？
3. 心得與討論
4. 題目難度: 超難/難/普/易/超易



- File type: pdf
- File name: Lab5_(Number of team)_report
- Deadline: 2022/04/27 23:59