# 實驗主題(Lab16)

B083022053

黃啟桓

Exercise 1:

## Exercise 1: BCD synchronous counter (1/2)

- Design and verify the BCD synchronous counter using T flip-flops with asynchronous reset and Verilog HDL
  - structural (gate-level) modeling

Exercise 2:

## Exercise 2: 4-bit binary counter with parallel load (1/2)

- Design and verify the 4-bit binary counter with parallel load using JK flip-flops with asynchronous reset and Verilog HDL
  - structural (gate-level) modeling

# 實驗內容

一、 Exercise 1:

## Exercise 1: BCD synchronous counter (1/2)

- Design and verify the BCD synchronous counter using T flip-flops with asynchronous reset and Verilog HDL
  - structural (gate-level) modeling

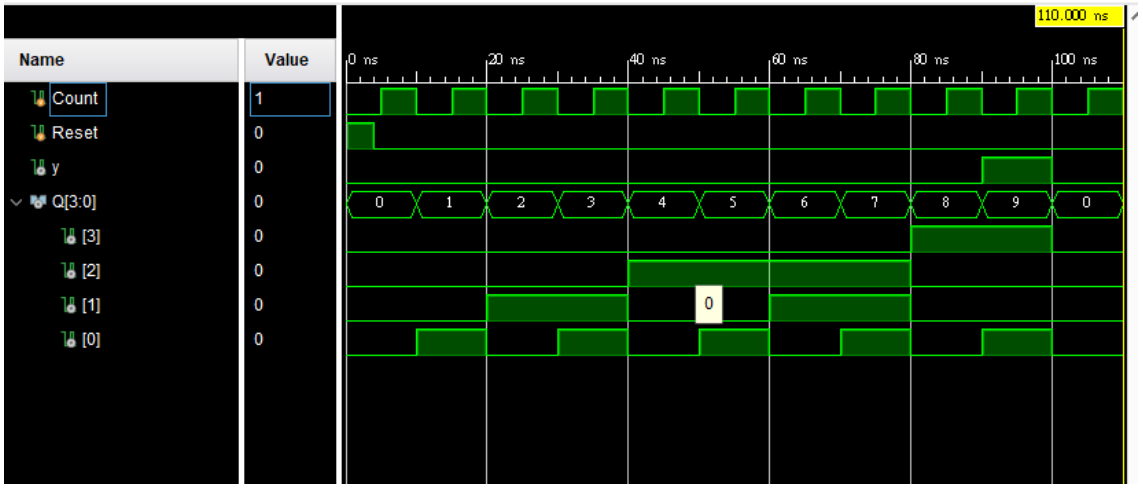
和上禮拜的用 JK flip flop 寫成的 BCD 計數器很像，只要 JK 輸入相

同就變成 T flip flop

```
3   module JK_ff (Q, clock, J, K, Reset);
4       output reg Q;
5       input J, K, clock, Reset;
6       wire Q_, K_, D_in, n1, n2,n3,n4;
7
8       not (Q_ , ~Q);
9       always @ (posedge clock, negedge Reset)
10      if (Reset == 0) Q <= 1'b0;
11      else Q <=  J&~Q|~K&Q;
12  endmodule
13
14  module T_ff (Q, clock, T, Reset);
15      output Q;
16      input T, clock, Reset;
17      JK_ff JK0(Q, clock, T, T, Reset);
18  endmodule
```

```
module BCD_synchronous_counter (Q, y,clock, Reset);
    output  [3:0]Q;
    output y;
    input clock, Reset;
    wire TQ1, TQ2, TQ4, TQ8;
    wire n1,n2;
    buf (TQ1, 1'b1);
    and (TQ2, ~Q[3], Q[0]);
    and (TQ4, Q[1], Q[0]);
    and (n1, Q[3], Q[0]);
    and (n2, Q[2], Q[1], Q[0]);
    or (TQ8, n1, n2);
    and (y, Q[3], Q[0]);

    T_ff TT0(Q[0], ~clock, TQ1, ~Reset);
    T_ff TT1(Q[1], ~clock, TQ2, ~Reset);
    T_ff TT2(Q[2], ~clock, TQ4, ~Reset);
    T_ff TT3(Q[3], ~clock, TQ8, ~Reset);
endmodule
```




結果如上圖

二、Exercise 2:

## Exercise 2: 4-bit binary counter with parallel load (1/2)

- Design and verify the 4-bit binary counter with parallel load using JK flip-flops with asynchronous reset and Verilog HDL

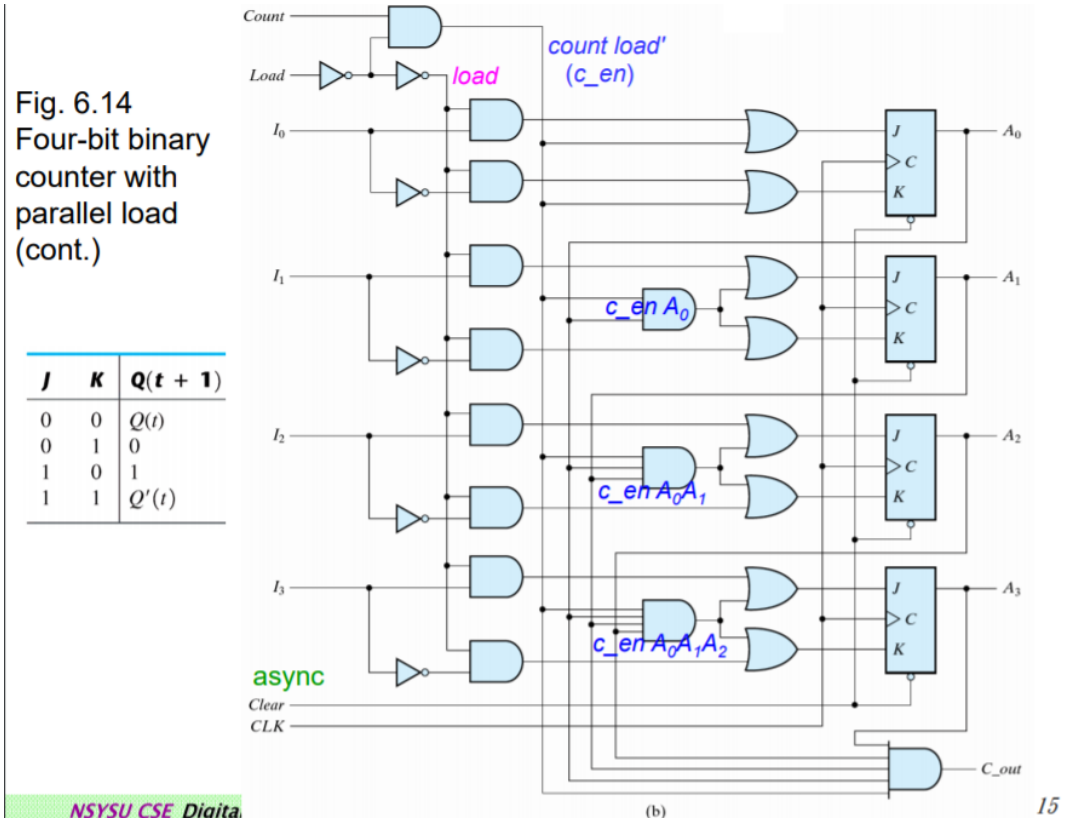



Fig. 6.14 Four-bit binary counter with parallel load (cont.)

| J | K | Q(t + 1) |
|---|---|---|
| 0 | 0 | Q(t) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Q'(t) |



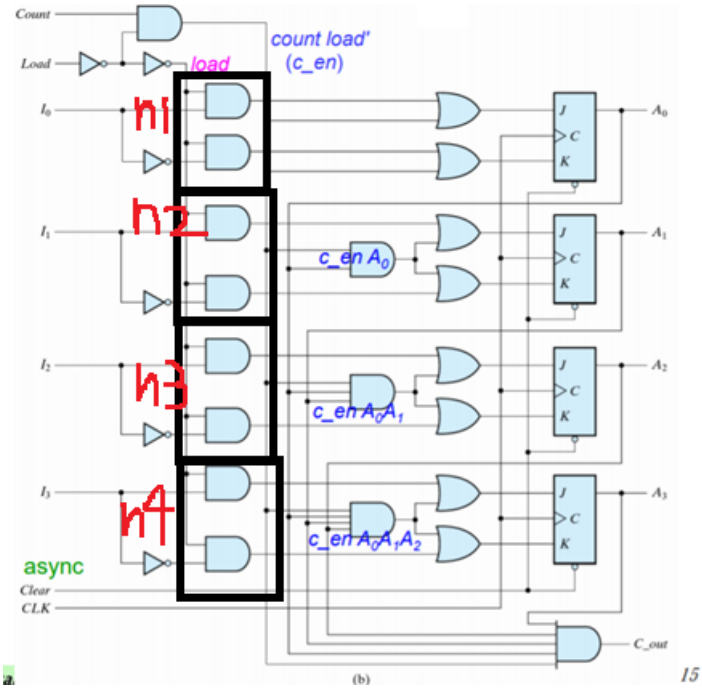
↓ ↓ ↓ JK flip flop ↓ ↓ ↓

```
module JK_ff (Q, clock, J, K, Reset);
    output reg Q;
    input J, K, clock, Reset;
    wire Q_, K_, D_in, n1, n2,n3,n4;
    not (Q_ , ~Q);
    always @ (posedge clock, negedge Reset)
    if (Reset == 0) Q <= 1'b0;
    else Q <=  J&~Q|~K&Q;
endmodule
```

```
module Four_bit_binary_counter (output [3:0]A,output C_out,input clock, Clear, Load, Count, input [3:0]Data_in);
    wire n1a,n1b,n2a,n2b,n3a,n3b,n4a,n4b,n5a,n5b,n6a,n6b,n7a,n7b,n8a,n8b;
    wire c_en0,c_en1,c_en2,c_en3,_Load,_I0,_I1,_I2,_I3;
    wire J0,K0,J1,K1,J2,K2,J3,K3;
    not (_Load, Load);
    not (_I0, Data_in[0]);
    not (_I1, Data_in[1]);
    not (_I2, Data_in[2]);
    not (_I3, Data_in[3]);

    and (n1a, Load,Data_in[0]);
    and (n1b, Load,_I0);
    and (n2a, Load,Data_in[1]);
    and (n2b, Load,_I1);
    and (n3a, Load,Data_in[2]);
    and (n3b, Load,_I2);
    and (n4a, Load,Data_in[3]);
    and (n4b, Load,_I3);
```

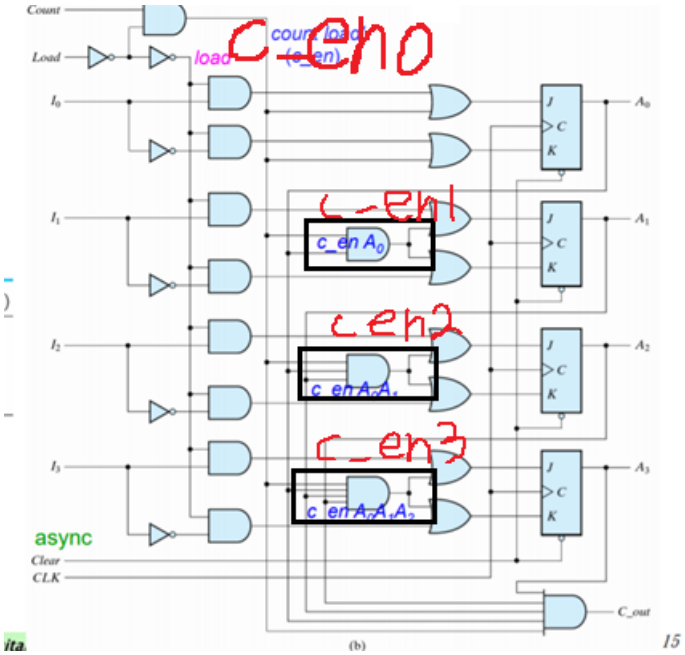
↓ ↓ ↓n1a、n1b、n2a、n2b.....的意思↓ ↓ ↓



```
    and (c_en0, Count, _Load);
    and (c_en1, c_en0, A[0]);
    and (c_en2, c_en0, A[0], A[1]);
    and (c_en3, c_en0, A[0], A[1], A[2]);

    or (J0, n1a, c_en0);
    or (K0, n1b, c_en0);
    or (J1, n2a, c_en1);
    or (K1, n2b, c_en1);
    or (J2, n3a, c_en2);
    or (K2, n3b, c_en2);
    or (J3, n4a, c_en3);
    or (K3, n4b, c_en3);

    JK_ff JK0(A[0], clock, J0, K0, ~Clear);
    JK_ff JK1(A[1], clock, J1, K1, ~Clear);
    JK_ff JK2(A[2], clock, J2, K2, ~Clear);
    JK_ff JK3(A[3], clock, J3, K3, ~Clear);
    and (C_out, c_en0, A[0], A[1], A[2], A[3]);
endmodule
```



# 全部就是長這樣

```
module JK_ff (Q, clock, J, K, Reset);
    output reg Q;
    input J, K, clock, Reset;
    wire Q_, K_, D_in, n1, n2,n3,n4;
    not (Q_ , ~Q);
    always @ (posedge clock, negedge Reset)
    if (Reset == 0) Q <= 1'b0;
    else Q <=  J&~Q|~K&Q;
endmodule
module Four_bit_binary_counter (output [3:0]A,output C_out,input clock, Clear, Load, Count, input [3:0]Data_in);
    wire n1a,n1b,n2a,n2b,n3a,n3b,n4a,n4b,n5a,n5b,n6a,n6b,n7a,n7b,n8a,n8b;
    wire c_en0,c_en1,c_en2,c_en3,_Load,_I0,_I1,_I2,_I3;
    wire J0,K0,J1,K1,J2,K2,J3,K3;
    not (_Load, Load);
    not (_I0, Data_in[0]);
    not (_I1, Data_in[1]);
    not (_I2, Data_in[2]);
    not (_I3, Data_in[3]);

    and (n1a, Load,Data_in[0]);
    and (n1b, Load,_I0);
    and (n2a, Load,Data_in[1]);
    and (n2b, Load,_I1);
    and (n3a, Load,Data_in[2]);
    and (n3b, Load,_I2);
    and (n4a, Load,Data_in[3]);
    and (n4b, Load,_I3);

    and (c_en0, Count, _Load);
    and (c_en1, c_en0, A[0]);
    and (c_en2, c_en0, A[0], A[1]);
    and (c_en3, c_en0, A[0], A[1], A[2]);

    or (J0, n1a, c_en0);
    or (K0, n1b, c_en0);
    or (J1, n2a, c_en1);
    or (K1, n2b, c_en1);
    or (J2, n3a, c_en2);
    or (K2, n3b, c_en2);
    or (J3, n4a, c_en3);
    or (K3, n4b, c_en3);

    JK_ff JK0(A[0], clock, J0, K0, ~Clear);
    JK_ff JK1(A[1], clock, J1, K1, ~Clear);
    JK_ff JK2(A[2], clock, J2, K2, ~Clear);
    JK_ff JK3(A[3], clock, J3, K3, ~Clear);
    and (C_out, c_en0, A[0], A[1], A[2], A[3]);
endmodule
```

# 實驗心得

1. Structural level modeling 中邏輯閘使用函數表示
2. Structural level modeling 可以很直觀的轉換成 Schematic
3. Structural level modeling 需要宣告邏輯閘之間的線路
4. 所有的 modeling 都需要注意該邏輯閘是否有交換律或結合律