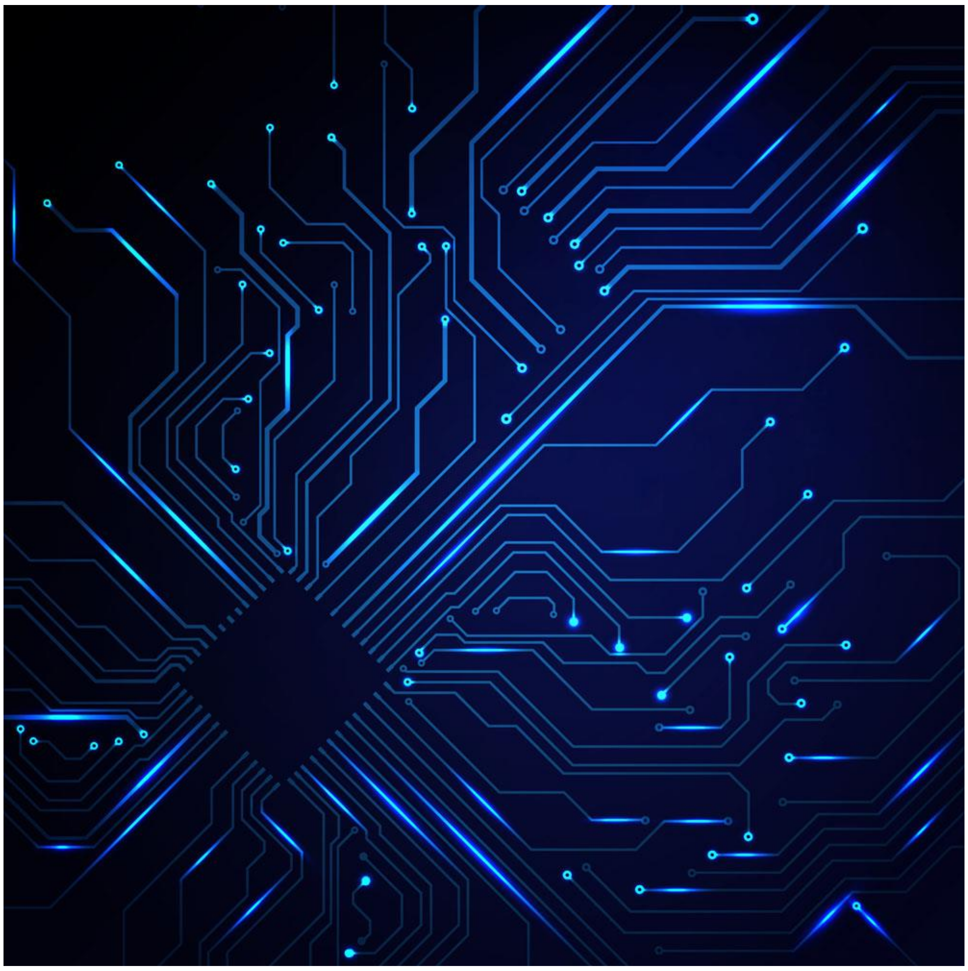


2024/XX/XX

實驗四

姓名：黃文祺　　　學號：01057013

班級：資工 3A

E-mail：OOOOOOOO

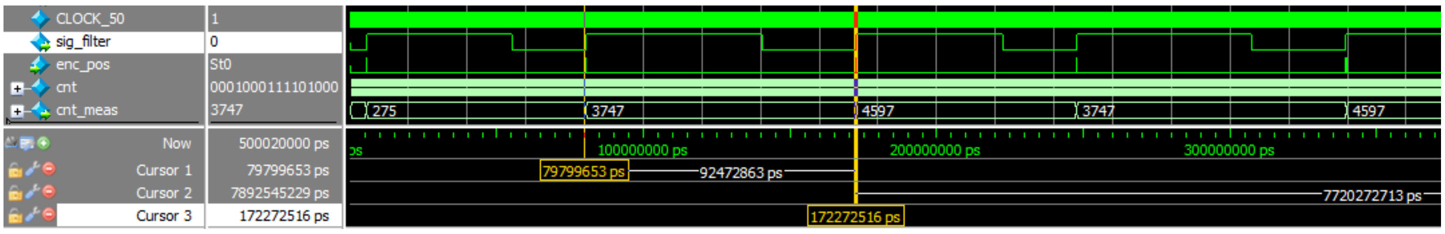
注意

1. 一律將此檔轉成 PDF 檔繳交
2. 繳交期限為
3. 隔週三上午九點
4. 一人繳交一份
5. 檔名：學號_HW?.pdf
6. 檔名請按照作業檔名格式進行填寫
7. 未依照格式不予批改

一、 Encoder

- ■ 實驗說明:
- ■ 系統架構程式碼與程式碼說明

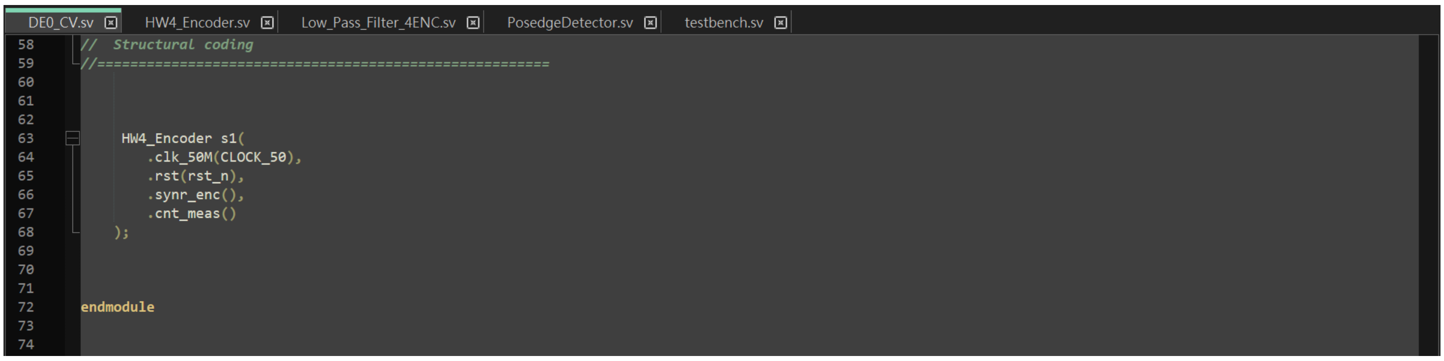
1. 以 50MHz 當作系統時脈
2. 輸入信號：enc，為輸入待測信號
3. 輸出數值：cnt_meas 為週期的計數（用 50MHz 計數）
4. enc 為有雜訊的信號，請先用 LPF 濾波後再計算



```
CLOCK_50    1
sig_filter  0
enc_pos     St0
cnt         0001000111101000
cnt_meas    3747

Now         500020000 ps
Cursor 1    79799653 ps
Cursor 2    7892545229 ps
Cursor 3    172272516 ps
```

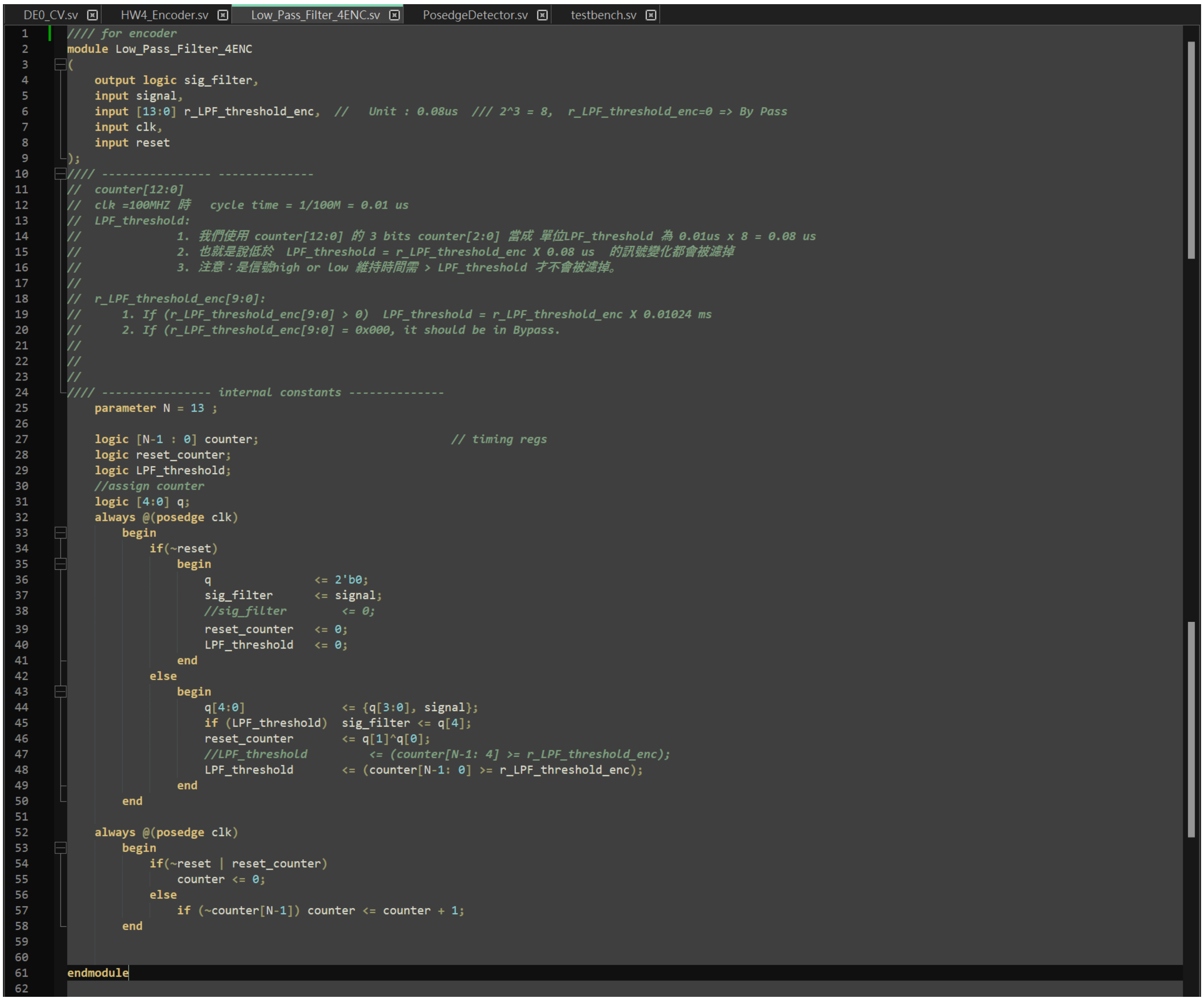
■ 系統架構程式碼與程式碼說明

DE0_CV:只有接主程式的線而已



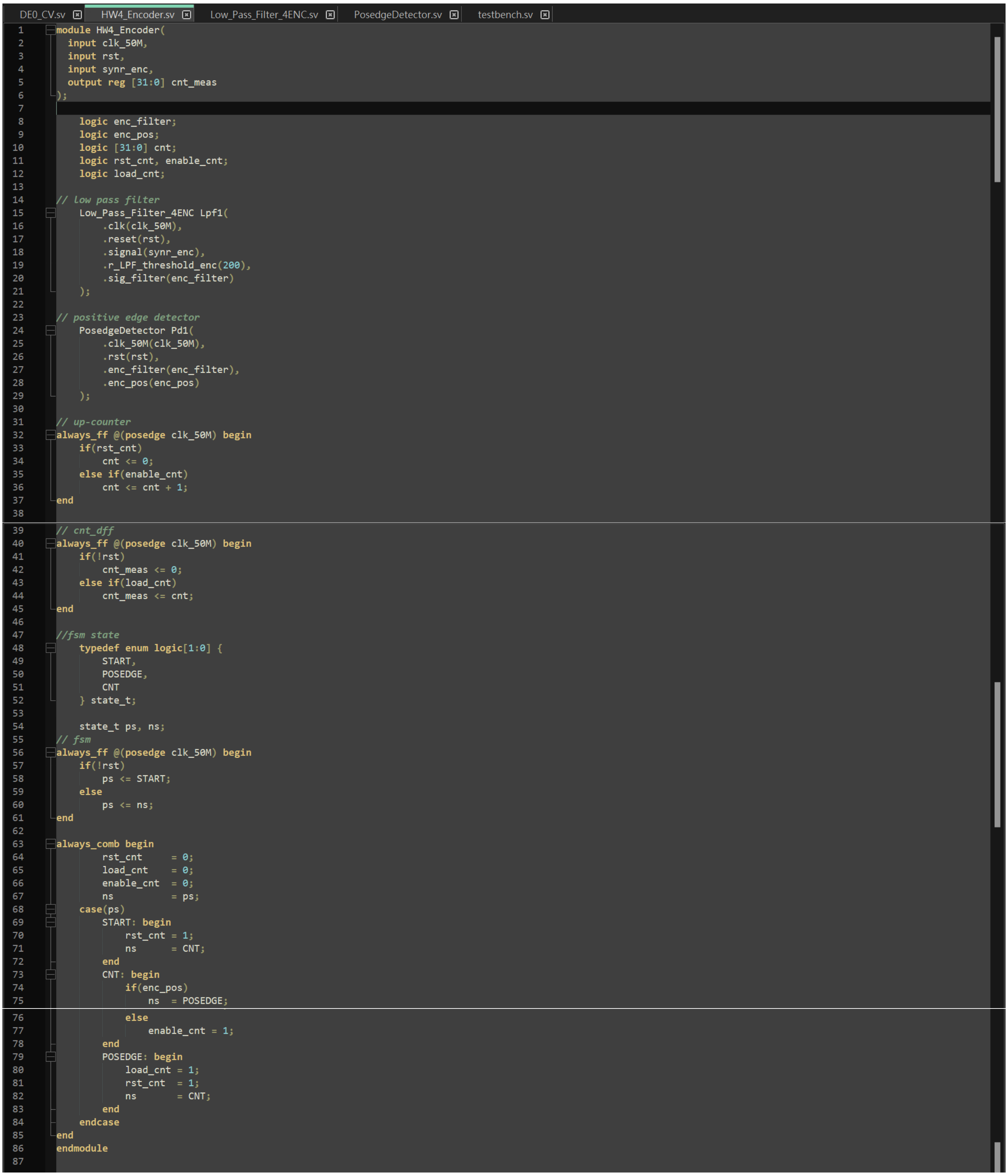
```
//  Structural coding
//=======================================================



     HW4_Encoder s1(
        .clk_50M(CLOCK_50),
        .rst(rst_n),
        .synr_enc(),
        .cnt_meas()
    );



endmodule

```

Low_Pass_Filter_4ENC:和老師給的教材一樣



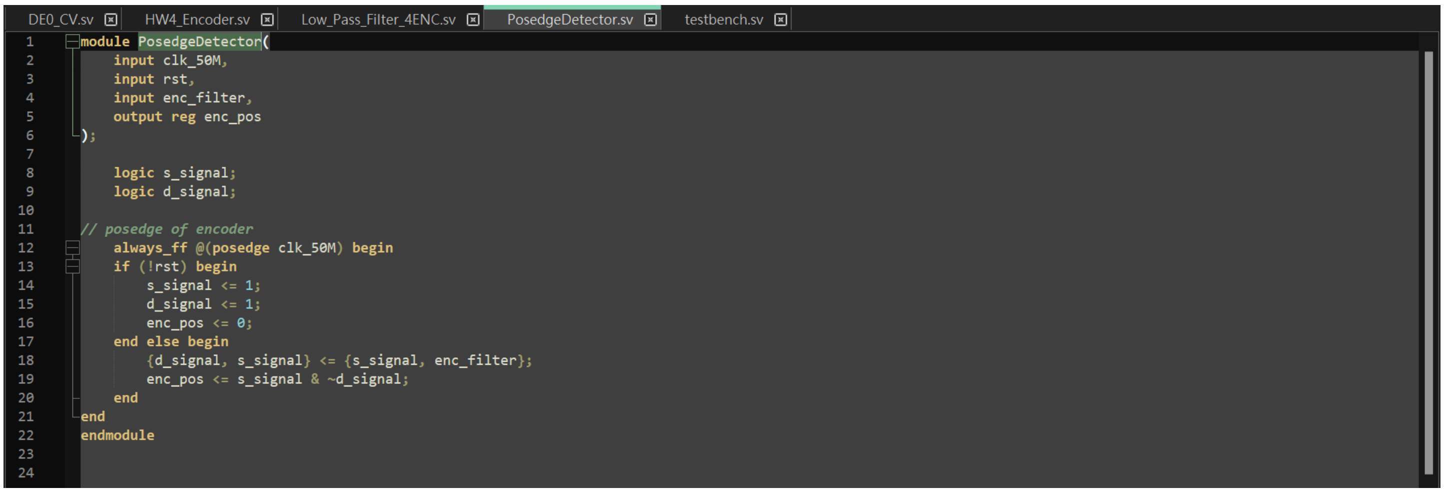
```
//// for encoder
module Low_Pass_Filter_4ENC
(
    output logic sig_filter,
    input signal,
    input [13:0] r_LPF_threshold_enc,  //   Unit : 0.08us  /// 2^3 = 8,  r_LPF_threshold_enc=0 => By Pass
    input clk,
    input reset
);
//// --------------- -------------
//  counter[12:0]
//  clk =100MHZ 時   cycle time = 1/100M = 0.01 us
//  LPF_threshold:
//            1. 我們使用 counter[12:0] 的 3 bits counter[2:0] 當成 單位LPF_threshold 為 0.01us x 8 = 0.08 us
//            2. 也就是說低於  LPF_threshold = r_LPF_threshold_enc X 0.08 us  的訊號變化都會被濾掉
//            3. 注意：是信號high or low 維持時間需 > LPF_threshold 才不會被濾掉。
//
//  r_LPF_threshold_enc[9:0]:
//      1. If (r_LPF_threshold_enc[9:0] > 0)  LPF_threshold = r_LPF_threshold_enc X 0.01024 ms
//      2. If (r_LPF_threshold_enc[9:0] = 0x000, it should be in Bypass.
//
//
//
//// --------------- internal constants -------------
    parameter N = 13 ;

    logic [N-1 : 0] counter;                            // timing regs
    logic reset_counter;
    logic LPF_threshold;
    //assign counter
    logic [4:0] q;
    always @(posedge clk)
        begin
            if(~reset)
                begin
                    q               <= 2'b0;
                    sig_filter      <= signal;
                    //sig_filter        <= 0;
                    reset_counter   <= 0;
                    LPF_threshold   <= 0;
                end
            else
                begin
                    q[4:0]              <= {q[3:0], signal};
                    if (LPF_threshold) sig_filter <= q[4];
                    reset_counter       <= q[1]^q[0];
                    //LPF_threshold         <= (counter[N-1: 4] >= r_LPF_threshold_enc);
                    LPF_threshold       <= (counter[N-1: 0] >= r_LPF_threshold_enc);
                end
        end

    always @(posedge clk)
        begin
            if(~reset | reset_counter)
                counter <= 0;
            else
                if (~counter[N-1]) counter <= counter + 1;
        end


endmodule

```

HW4_Encoder:裡面有兩個 module 和上數計數器、cnt_dff、及 FSM



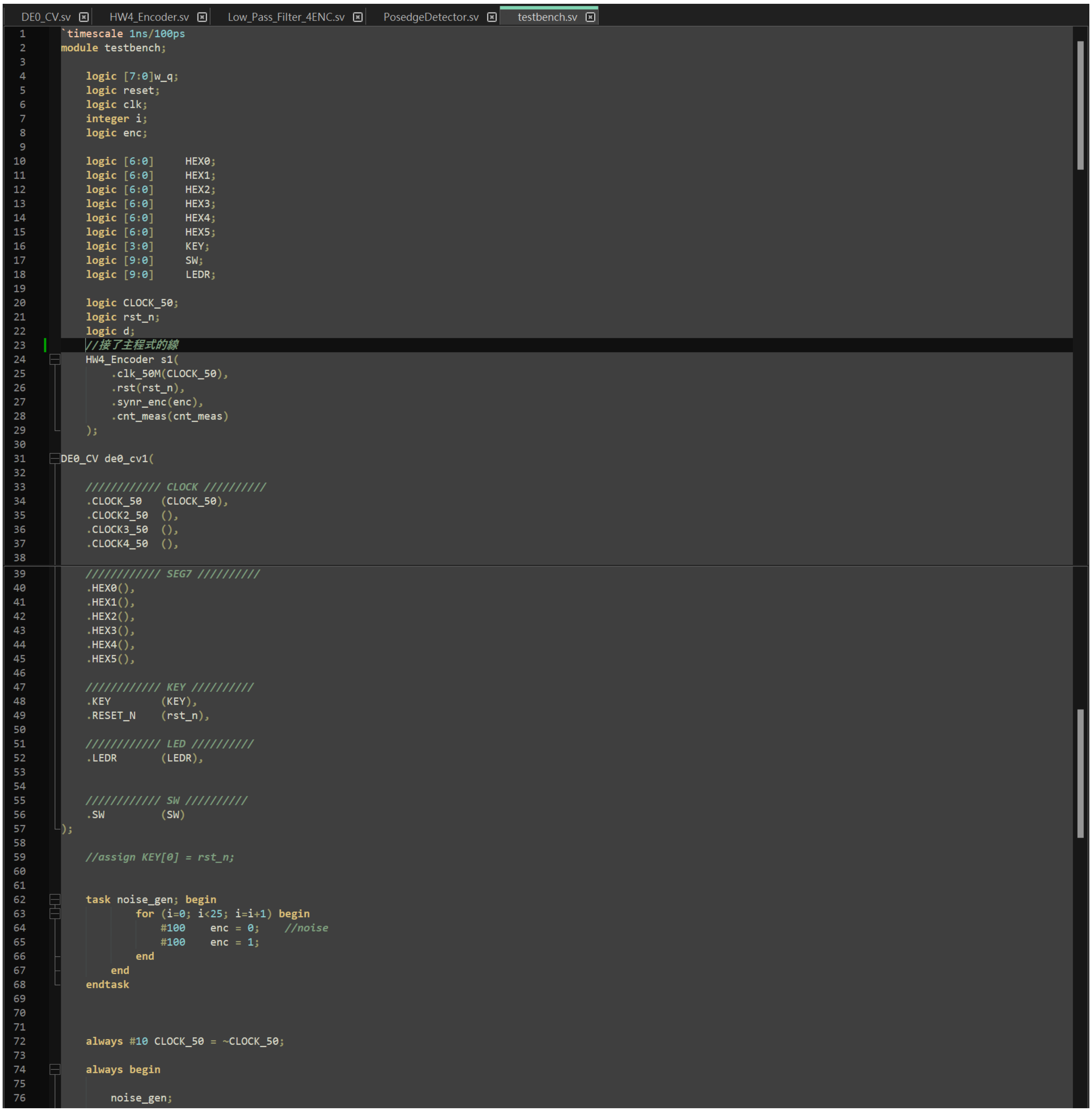
```systemverilog
module HW4_Encoder(
  input clk_50M,
  input rst,
  input synr_enc,
  output reg [31:0] cnt_meas
);

    logic enc_filter;
    logic enc_pos;
    logic [31:0] cnt;
    logic rst_cnt, enable_cnt;
    logic load_cnt;

// low pass filter
    Low_Pass_Filter_4ENC Lpf1(
        .clk(clk_50M),
        .reset(rst),
        .signal(synr_enc),
        .r_LPF_threshold_enc(200),
        .sig_filter(enc_filter)
    );

// positive edge detector
    PosedgeDetector Pd1(
        .clk_50M(clk_50M),
        .rst(rst),
        .enc_filter(enc_filter),
        .enc_pos(enc_pos)
    );

// up-counter
always_ff @(posedge clk_50M) begin
    if(rst_cnt)
        cnt <= 0;
    else if(enable_cnt)
        cnt <= cnt + 1;
end

// cnt_dff
always_ff @(posedge clk_50M) begin
    if(!rst)
        cnt_meas <= 0;
    else if(load_cnt)
        cnt_meas <= cnt;
end

//fsm state
    typedef enum logic[1:0] {
        START,
        POSEDGE,
        CNT
    } state_t;

    state_t ps, ns;
// fsm
always_ff @(posedge clk_50M) begin
    if(!rst)
        ps <= START;
    else
        ps <= ns;
end

always_comb begin
        rst_cnt     = 0;
        load_cnt    = 0;
        enable_cnt  = 0;
        ns          = ps;
    case(ps)
        START: begin
            rst_cnt = 1;
            ns      = CNT;
        end
        CNT: begin
            if(enc_pos)
                ns  = POSEDGE;
            else
                enable_cnt = 1;
        end
        POSEDGE: begin
            load_cnt = 1;
            rst_cnt  = 1;
            ns        = CNT;
        end
    endcase
end
endmodule
```

PosedgeDetector:和投影片中的一樣

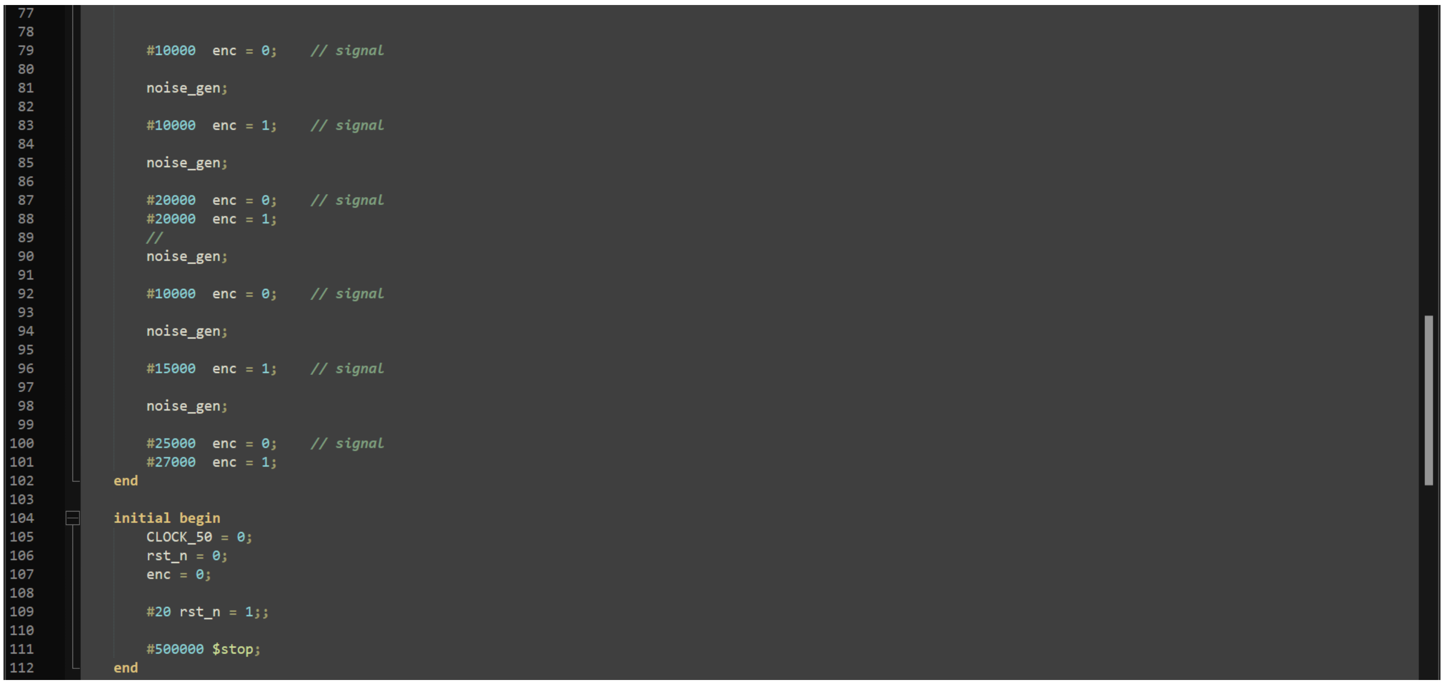


```
module PosedgeDetector(
    input clk_50M,
    input rst,
    input enc_filter,
    output reg enc_pos
);

    logic s_signal;
    logic d_signal;

// posedge of encoder
    always_ff @(posedge clk_50M) begin
    if (!rst) begin
        s_signal <= 1;
        d_signal <= 1;
        enc_pos <= 0;
    end else begin
        {d_signal, s_signal} <= {s_signal, enc_filter};
        enc_pos <= s_signal & ~d_signal;
    end
end
endmodule
```

Testbench:除了接上主程式的線外，其他都沒變動



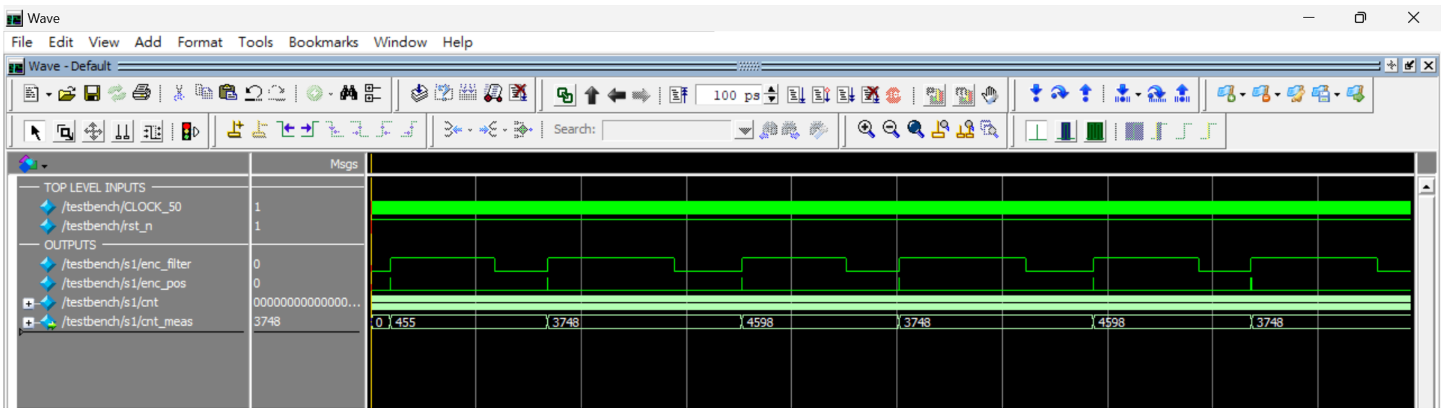
```
`timescale 1ns/100ps
module testbench;

    logic [7:0]w_q;
    logic reset;
    logic clk;
    integer i;
    logic enc;

    logic [6:0]     HEX0;
    logic [6:0]     HEX1;
    logic [6:0]     HEX2;
    logic [6:0]     HEX3;
    logic [6:0]     HEX4;
    logic [6:0]     HEX5;
    logic [3:0]     KEY;
    logic [9:0]     SW;
    logic [9:0]     LEDR;

    logic CLOCK_50;
    logic rst_n;
    logic d;
    //接了主程式的線
    HW4_Encoder s1(
        .clk_50M(CLOCK_50),
        .rst(rst_n),
        .synr_enc(enc),
        .cnt_meas(cnt_meas)
    );

DE0_CV de0_cv1(

    /////////// CLOCK /////////
    .CLOCK_50   (CLOCK_50),
    .CLOCK2_50  (),
    .CLOCK3_50  (),
    .CLOCK4_50  (),

    /////////// SEG7 /////////
    .HEX0(),
    .HEX1(),
    .HEX2(),
    .HEX3(),
    .HEX4(),
    .HEX5(),

    /////////// KEY /////////
    .KEY        (KEY),
    .RESET_N    (rst_n),

    /////////// LED /////////
    .LEDR       (LEDR),


    /////////// SW /////////
    .SW         (SW)
);

    //assign KEY[0] = rst_n;


    task noise_gen; begin
            for (i=0; i<25; i=i+1) begin
                #100    enc = 0;    //noise
                #100    enc = 1;
            end
        end
    endtask



    always #10 CLOCK_50 = ~CLOCK_50;

    always begin

        noise_gen;
```



```
        #10000  enc = 0;    // signal

        noise_gen;

        #10000  enc = 1;    // signal

        noise_gen;

        #20000  enc = 0;    // signal
        #20000  enc = 1;
        //
        noise_gen;

        #10000  enc = 0;    // signal

        noise_gen;

        #15000  enc = 1;    // signal

        noise_gen;

        #25000  enc = 0;    // signal
        #27000  enc = 1;
    end

    initial begin
        CLOCK_50 = 0;
        rst_n = 0;
        enc = 0;

        #20 rst_n = 1;;

        #500000 $stop;
    end
```

```

endmodule
```

模擬結果與結果說明：


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
TOP LEVEL INPUTS
/testbench/CLOCK_50 1
/testbench/rst_n 1
OUTPUTS
/testbench/s1/enc_filter 0
/testbench/s1/enc_pos 0
/testbench/s1/cnt 00000000000000...
/testbench/s1/cnt_meas 3748
0
455
3748
4598
3748
4598
3748

■ 結論與心得：

本來以為 demo 完成之後，作業概念就差不多沒問題了，但沒想到在接線上卻出了些小誤會，拉拉扯扯的檢查好久才處理好(›´ω`‹)......

不過好在最後有在 deadline 前完成! (๑•̀ㅂ•́)و✧