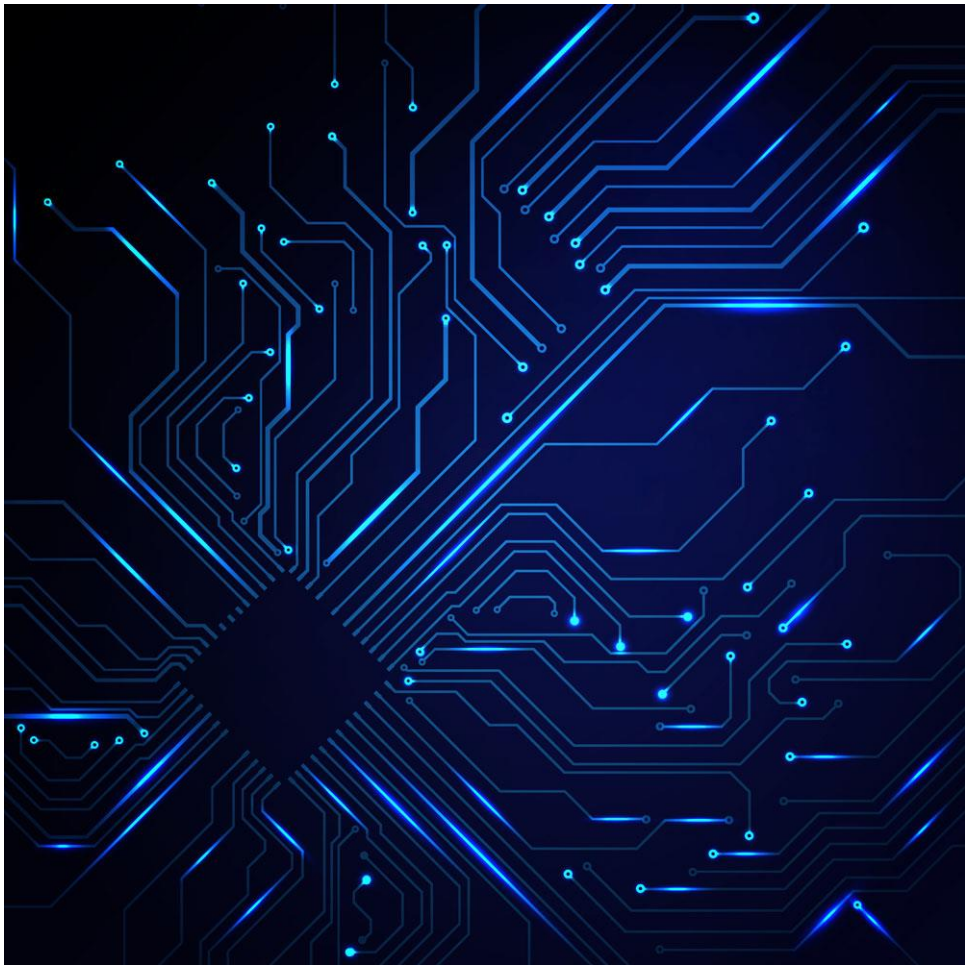


2024/XX/XX

實驗九

姓名：黃文祺　　學號：01057013

班級：資工 3A

E-mail：OOOOOOOO

注意

1. 繳交時一律轉 PDF 檔
2. 繳交期限為
   隔週三上午九點
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
   檔名請按照作業檔名格式進行填寫
   未依照格式不予批改

SPI Slave

- 實驗說明：
  1. 完成 SPI Slave 的接收與傳輸功能，並顯示其模擬圖。
  2. SPI Slave 功能說明
     - 若收到寫入指令，依序接收 32 位元 mosi 資料，依照資料格式將資料寫進 register file。
     - 若收到讀取指令，將資料從 register file 中讀出來，並依序傳至 miso 接腳。
  3. testbench.sv 會呼叫 DE0_CV.sv，請在 DE0_CV.sv 中完成程式碼撰寫。
  4. DE0_CV.sv 使用接腳：
     - CLOCK_50：50MHz 的 clk 訊號。
     - RESET_N：系統 reset，為 0 時重置系統。
     - GPIO_0[0]：傳訊號進 SPI Slave 的 mosi。
     - GPIO_0[1]：傳訊號進 SPI Slave 的 sclk。
     - GPIO_0[2]：傳訊號進 SPI Slave 的 ssn。
     - GPIO_0[3]：接收 SPI Slave 的 miso 訊號。
  5. SPI.sv 輸入:
     - clk (請將 DE0_CV 的 CLOCK_50，用 PLL 升至 100MHz)
     - mosi
     - sclk(testbench 已設定為 10MHz)
     - ssn
     - reset
  6. SPI.sv 輸出:
     - data_debug [15:0](將讀取的 register file 資料接出來)
     - miso

■ 波型圖參考


TestBench
clk 1
reset 0
cmd_mosi 0
cmd_sclk 0
cmd_ssn 1
cmd_miso 0
write_address ff
data ffff
read_address 00
get_data 0000
DE0_CV
CLK_100MHz 0
RESET_N St1
mosi 0
sclk 0
ssn 1
miso 0
SPI
clk St0
reset St0
mosi St0
sclk St0
ssn St1
miso 0
tx_req 0
address 00
write_data ffff
read_data 0000
data_debug 0000
00 0000
a5 aa55
5a 5678
ff ffff
ff ffff
5a 5678
a5 aa55
00 0000
00 0000
a5 aa55
5a 5678
ff ffff
5a 5678
a5 aa55
00 0000
ffff 5678 aa55 0000
ffff 5678 aa55 0000

■ 系統架構程式碼與程式碼說明

截圖請善用 win+shift+S

DE0_CV.sv:

C:\Users\user\Desktop\DE0_CV_SPI\design\DE0_CV.sv - Notepad++

檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M) 執行(R) 外掛(P) 視窗(W) ?

SPI.sv | SPI_rx.sv | SPI_tx.sv | DE0_CV.sv

```
64      //=======================================================
65
66
67      assign mosi             =   GPIO_0[0];
68      assign sclk             =   GPIO_0[1];
69      assign ssn              =   GPIO_0[2];
70      assign GPIO_0[3]        =   miso;
71      assign LEDR         =   data_debug;
72
73      /*請使用PLL IP
74          輸入：CLOCK_50
75          輸出：CLK_100MHz
76      */
77      pll pll(
78          .refclk(CLOCK_50),
79          .rst(~RESET_N),
80          .outclk_0(CLK_100MHz),
81          .locked()
82      );
83
84      SPI SPI(
85          //------output----------------------------------
86          .miso               (miso),
87          .data_debug         (data_debug),
88
89          //------input-----------------------------------
90          .mosi               (mosi),
91          .sclk               (sclk),
92          .ssn                (ssn),
93          .clk                (CLK_100MHz),
94          .reset              (~RESET_N)
95  );
96
97
98  endmodule
```

SPI.sv:

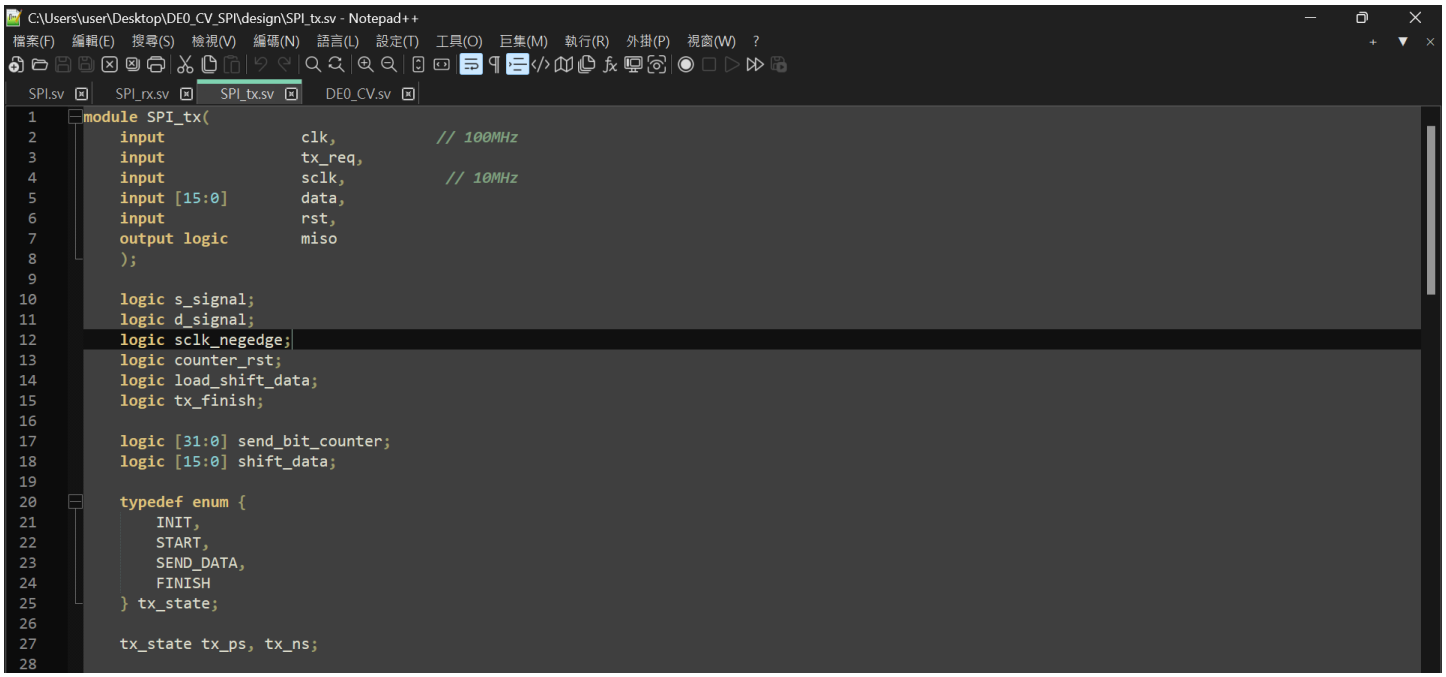
C:\Users\user\Desktop\DE0_CV_SPI\design\SPI.sv - Notepad++

檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M) 執行(R) 外掛(P) 視窗(W) ?

SPI.sv | SPI_rx.sv | SPI_tx.sv | DE0_CV.sv

```
16
17      );
18
19      SPI_rx rx1(
20      //------output----------------------------------
21          .address(),             //8 bits address
22          .data(),    //16 bits data
23          .read_data(data_debug),     //16 bits data
24          .read_en(),
25          .write_en(),
26          .tx_req(tx_req),
27
28      //------input-----------------------------------
29          .mosi(mosi),
30          .sclk(sclk),
31          .ssn(ssn),
32          .clk(clk),
33          .rst(reset)
34      );
35
36
37
38      SPI_tx tx1(
39          .clk(clk),
40          .tx_req(tx_req),
41          .sclk(sclk),
42          .data(data_debug),
43          .rst(reset) ,
44          .miso(miso)
45
46      );
47  endmodule
```

SPI_tx.sv:

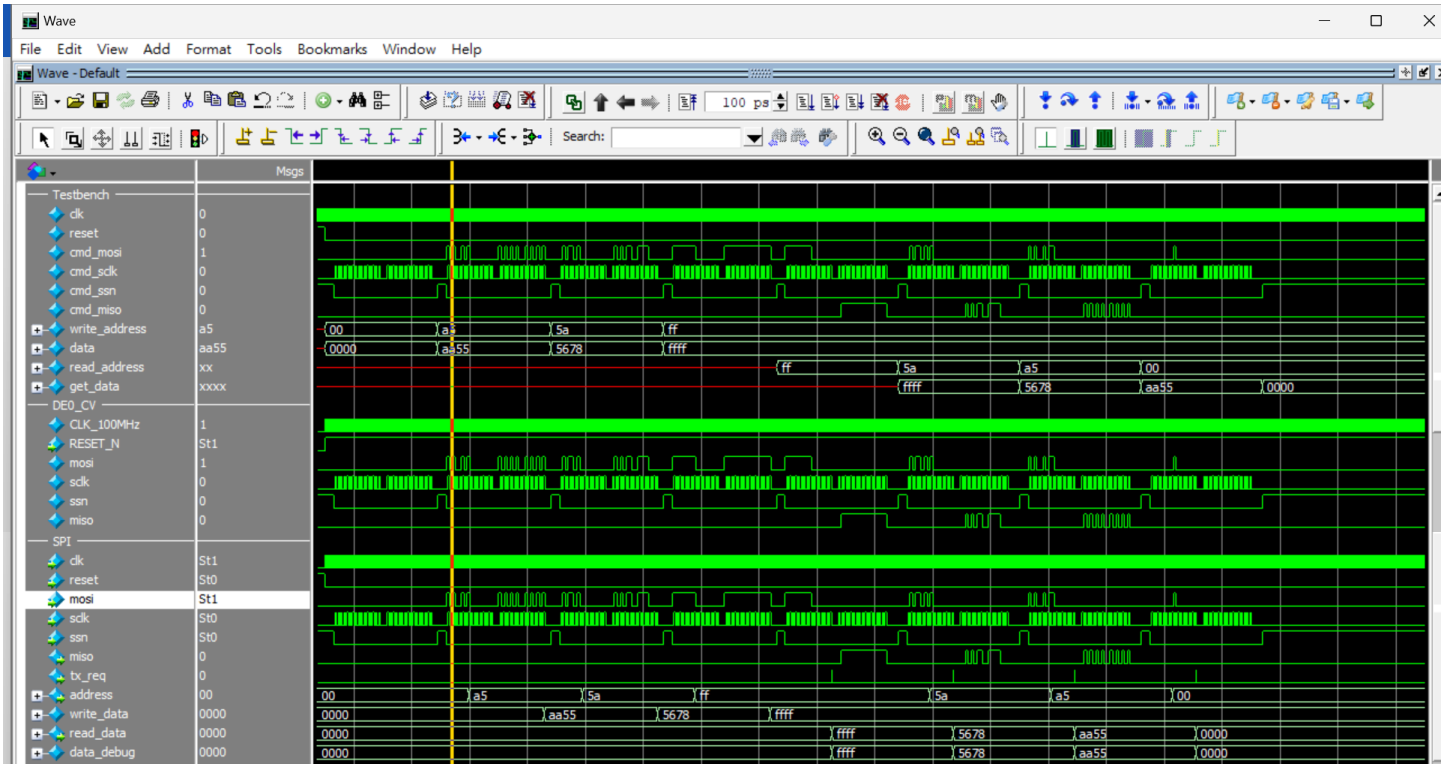



```
29  // sclk negedge detector
30  always_ff @(posedge clk or posedge rst) begin
31      if(rst) begin
32          s_signal        <= 1;
33          d_signal        <= 1;
34          sclk_negedge    <= 0;
35      end
36      else begin
37          {d_signal, s_signal} <= {s_signal, sclk};
38          sclk_negedge <= ~s_signal & d_signal;
39      end
40  end
41
42  // send data counter
43  always_ff @(posedge clk) begin
44      if(rst| counter_rst) begin
45          send_bit_counter[31:0] <= 0;
46      end
47      else if(sclk_negedge) begin
48          send_bit_counter <= send_bit_counter + 1;
49      end
50  end
51
```

```
52    //  shift register
53    always_ff @(posedge clk or posedge rst) begin
54        if(rst) begin
55            miso            <= 0;
56            shift_data[15:0] <= 0;
57        end
58        else if(load_shift_data) begin
59            shift_data <= data;
60        end
61        else if(sclk_negedge) begin
62            miso        <=  shift_data[15];
63            shift_data  <=  {shift_data[14:0],1'b0};
64        end
65    end
66
67    //fsm
68    always_ff@ (posedge clk) begin
69        if(rst) begin
70            tx_ps <= INIT;
71        end
72        else begin
73            tx_ps <= tx_ns;
74        end
75    end
76
```

```
77    always_comb begin
78        counter_rst     = 0;
79        load_shift_data = 0;
80        tx_finish       = 0;
81        tx_ns           = tx_ps;
82        case(tx_ps)
83            INIT: begin
84                if(tx_req) begin
85                    tx_ns = START;
86                end
87            end
88            START: begin
89                counter_rst     = 1;
90                load_shift_data = 1;
91                tx_ns           = SEND_DATA;
92            end
93            SEND_DATA: begin
94                if(send_bit_counter >= 16) begin
95                    tx_ns = FINISH;
96                end
97            end
98            FINISH: begin
99                tx_finish = 1;
100               tx_ns     = INIT;
101           end
102       endcase
103   end
104   endmodule
```

■ 模擬結果與結果說明：



■ 結論與心得：

這次作業把整個 SPI 都完成了! 但 testbench 是助教提供的，一開始還不太了解，把整個程式碼拿去餵 GPT，再看看不會的寫法，也才比較了解 read 和 write 的 task 原來是這樣寫，而這次的 tx 雖然沒有那麼難，但因為方塊圖跟流程圖比較沒有那麼詳細，也研究了一會才慢慢摸索出來，另外就是這次比較有收穫的是接線的部分，一開始在課堂上的時候還因為接線問題，一直跑不出正確的波型圖，後來跟同學討論才發現問題，真是麻煩別人了@@