

2024/XX/XX

實驗十

姓名：黃文祺 學號：01057013

班級：資工 3A

E-mail：OOOOOOOOO

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

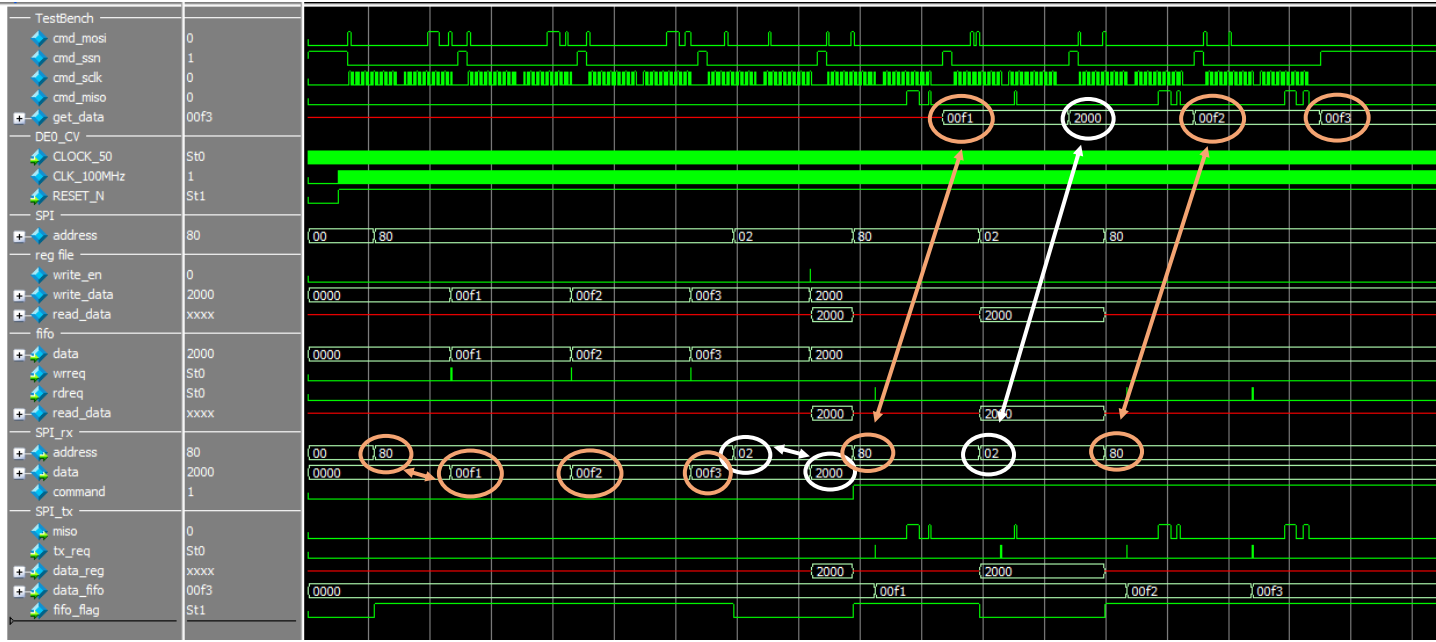
SPI Slave

■ 實驗說明：

1. 將 SPI Slave 結合 FIFO，並顯示其模擬圖。
2. SPI Slave 功能說明
 - 若位址最高位為 1，由 FIFO 讀寫資料。
 - 若位址最高位為 0，由 Register File 讀寫資料。
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：
 - miso

■ 波型圖參考

- 務必擷取到 get_data 的波型



■ 系統架構程式碼與程式碼說明

截圖請善用 win+shift+S

spi_rx.sv (這次主要只有改 Rx，所以就附就好)

```
1  module spi_rx
2  (
3      input clk,
4      input rst,
5      input mosi,
6      input ssn,
7      input sclk,
8      input tx_ack,
9
10     output logic [7:0] addr,
11     output logic [15:0] data_r,
12     output logic read_en,
13     output logic write_en,
14     output logic tx_req
15 );
16 logic rx_finish;
17
18
19
20 // sclk_pos;
21 logic sclk_pos;
22 logic sclk_d_signal;
23 logic sclk_s_signal;
24 always_ff @(posedge clk) begin
25     if(rst) begin
26         sclk_s_signal <= 1'b1;
27         sclk_d_signal <= 1'b1;
28         sclk_pos <= 1'b0;
29     end
30     else begin
31         {sclk_d_signal, sclk_s_signal} <= {sclk_s_signal, sclk};
32         sclk_pos <= ~sclk_d_signal & sclk_s_signal;
33     end
34 end
35
36
37 // ssn_neg
38 logic ssn_d_signal;
39 logic ssn_s_signal;
40 logic ssn_neg;
41
42 always_ff @(posedge clk) begin
43     if(rst) begin
44         ssn_s_signal <= 1'b1;
45         ssn_d_signal <= 1'b1;
46         ssn_neg <= 1'b0;
47     end
48     else begin
49         {ssn_d_signal, ssn_s_signal} <= {ssn_s_signal, ssn};
50         ssn_neg <= ssn_d_signal & ~ssn_s_signal;
51     end
52 end
53
54
55 //shift register
56 logic rst_shift_regs;
57 logic [15:0] shift_data;
58
59 always_ff @(posedge clk) begin
60     if(rst_shift_regs) shift_data <= 0;
61     else if(sclk_pos) shift_data <= {shift_data[14:0], mosi}; // bit_in = mosi
62 end
63
64
65 //receive data counter
66 logic rst_data_cnt;
67 logic load_data_cnt;
68 logic [15:0] rcv_data_cnt;
69
70 always_ff @(posedge clk) begin
71     if (rst_data_cnt) rcv_data_cnt <= 0;
72     else if (sclk_pos) rcv_data_cnt <= rcv_data_cnt + 1;
73 end
74
```

```

75
76
77 //reg
78 logic load_addr;
79 logic load_cmd;
80 logic load_data;
81 logic command;
82 logic [15:0] data;
83
84 always_ff @(posedge clk) begin
85     if(rst) begin
86         data <= 0;
87         addr <= 0;
88         command <= 0;
89     end
90     else if(load_addr) addr <= shift_data[7:0];
91     else if(load_cmd) command <= shift_data[7];
92     else if(load_data) data <= shift_data[15:0];
93 end
94
95
96 //register_file
97 logic [15:0] data_reg;
98 logic [15:0] reg_file [255:0];
99
100 always_ff @(posedge clk) begin
101     if(write_en) reg_file[addr] <= data;
102     if(read_en) data_reg <= reg_file[addr];
103 end
104
105
106 logic wrreq;
107 logic rdreq;
108 logic wrreq_neg;
109 logic rdreq_neg;
110 logic almost_empty;
111 logic [15:0] read_data_fifo;
112 logic [15:0] usedw;
113 logic [15:0] write_data_neg;
114
115 always_ff @(negedge clk) begin
116     if(rst) begin
117         write_data_neg <= 0;
118         wrreq_neg <= 0;
119         rdreq_neg <= 0;
120     end
121     else begin
122         write_data_neg <= data;
123         wrreq_neg <= wrreq;
124         rdreq_neg <= rdreq;
125     end
126 end
127
128
129 fifo fifo_1
130 (
131     .clock(clk),
132     .data(write_data_neg),
133     .rdreq(rdreq_neg),
134     .sclr(rst),
135     .wrreq(wrreq_neg),
136     .almost_empty(almost_empty),
137     .empty(empty),
138     .full(full),
139     .q(read_data_fifo),
140     .usedw(usedw)
141 );
142
143 assign data_r = addr[7] ? read_data_fifo : data_reg;
144
145 typedef enum {
146     INIT,
147     START_SPI_RX,
148     RECEIVE_ADDRESS,
149     DUMMY,
150     CHECK_COMMAND,
151     TX_REQ,
152     FINISH,
153     RECEIVE_DATA,
154     WRITE
155 } state_t;
156
157 state_t ps, ns;
158
159 always_ff @(posedge clk)
160     if(rst) ps <= INIT;
161     else ps <= ns;
162

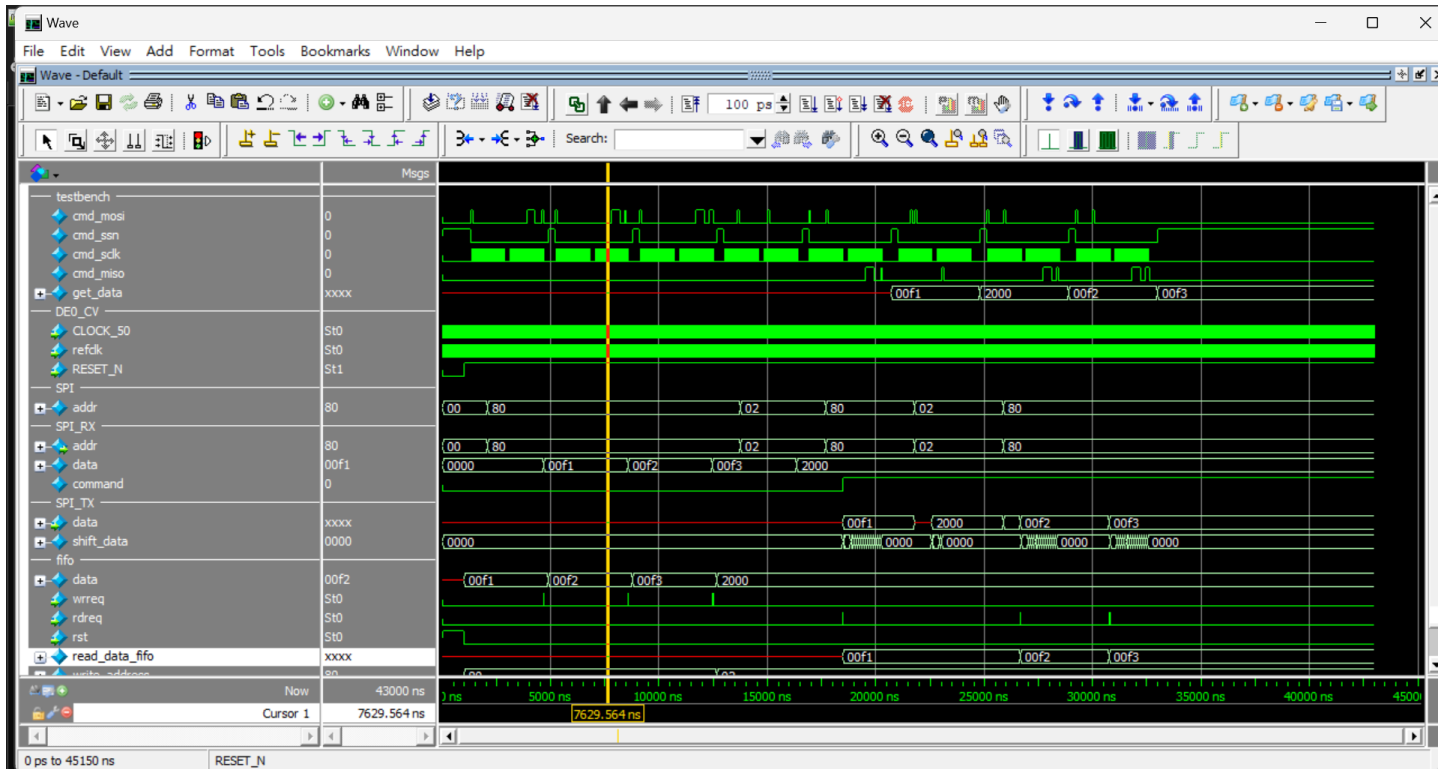
```

```

163 always_comb begin
164     rst_data_cnt = 0;
165     rst_shift_regs = 0;
166     load_data_cnt = 0;
167
168     load_addr = 0;
169     load_cmd = 0;
170     load_data = 0;
171
172     write_en = 0;
173     rx_finish = 0;
174
175     read_en = 0;
176     tx_req = 0;
177
178     wrreq = 0;
179     rdreq = 0;
180
181     ns = ps;
182
183     case(ps)
184     INIT: begin
185         ns = START_SPI_RX;
186     end
187
188     START_SPI_RX: begin
189         if(ssn_neg) begin
190             rst_data_cnt = 1;
191             rst_shift_regs = 1;
192             ns = RECEIVE_ADDRESS;
193         end
194     end
195
196     RECEIVE_DATA: begin
197         if(rcv_data_cnt >= 16) begin
198             load_data = 1;
199             rst_data_cnt = 1;
200             ns = WRITE;
201         end
202         load_data_cnt = 1;
203     end
204
205     WRITE: begin
206         if(addr[7]) wrreq = 1;
207         else write_en = 1;
208
209         ns = FINISH;
210     end
211
212     FINISH: begin
213         rx_finish = 1;
214         ns = INIT;
215     end
216
217     endcase
218 end
219
220 endmodule
221
222 RECEIVE_ADDRESS: begin
223     if(rcv_data_cnt >= 8) begin
224         load_addr = 1;
225         ns = DUMMY;
226     end
227     load_data_cnt = 1;
228 end
229
230 DUMMY: begin
231     if(rcv_data_cnt >= 16) begin
232         load_cmd = 1;
233         rst_data_cnt = 1;
234         ns = CHECK_COMMAND;
235     end
236     load_data_cnt = 1;
237 end
238
239 CHECK_COMMAND: begin
240     if(command) begin
241         if(addr[7]) rdreq = 1;
242         else read_en = 1;
243
244         ns = TX_REQ;
245     end
246
247     else ns = RECEIVE_DATA;
248 end
249
250 TX_REQ: begin
251     tx_req = 1;
252     ns = FINISH;
253 end
254
255 end
256

```

■ 模擬結果與結果說明：



■ 結論與心得：

這次作業一開始覺得應該只是個模組而已，應該不會太難，但沒想到在做的時候有些理解錯誤，以為要調整 register file，所以弄得亂七八糟，好險每次都會把作業備份，重新理解後再接線就沒甚麼問題了！