

數位系統設計作業

HW9

學號：01257027 姓名：林承羿

第一題

前置作業

```
1  library ieee;          Ian_TOPCS, 6 天前 • VHDL week11 tmp
2  use ieee.std_logic_unsigned.all;
3  use ieee.std_logic_1164.all;
4
5
6  entity digits is
7  port(
8      BIN:in integer range 0 to 9999;
9      num1: out integer range 0 to 9;
10     num0: out integer range 0 to 9
11 );
12
13 end digits;
14
15 architecture digits of digits is
16 begin
17     num1 <= (BIN /10);      -- 算出十位數
18     num0 <= BIN mod 10;    -- 求出個位數
19 end digits;
```

將分數求出 十位數 與 個位數

```

1  library ieee;          Ian_TOPCS, 6 天前 • VHDL week11 tmp
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity decoder_7seg is
6  PORT(
7      BCD:in std_logic_vector(3 downto 0);
8      HEX:out std_logic_vector(6 downto 0)
9  );
10 end decoder_7seg;
11
12 architecture decoder_7seg of decoder_7seg is
13 begin
14
15     HEX <= "1000000" when BCD = 0 else
16           "1111001" when BCD = 1 else
17           "0100100" when BCD = 2 else
18           "0110000" when BCD = 3 else
19           "0011001" when BCD = 4 else
20           "0010010" when BCD = 5 else
21           "0000010" when BCD = 6 else
22           "1111000" when BCD = 7 else
23           "0000000" when BCD = 8 else
24           "0010000" when BCD = 9 else
25           "1111111";
26 end decoder_7seg;

```

將十位數與各位數分別帶入七段顯示器

程式碼

```
1 library ieee;          Ian_TOPCS, 6 天前 • VHDL week11 tmp
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 use ieee.std_logic_arith.all;
5
6 entity ping_pong is
7     port(
8         clk:in std_logic;          -- 50MHz
9         reset:in std_logic;        -- 歸零
10        A:in std_logic;             -- A 發球
11        B:in std_logic;             -- B 發球
12        led:out std_logic_vector(9 downto 0);
13        HEX0, HEX1, HEX2, HEX3:out std_logic_vector(6 downto 0)
14    );
15 end ping_pong;
16
17 architecture ping_pong of ping_pong is
18
19     component decoder_7seg is
20         PORT(
21             BCD:in std_logic_vector(3 downto 0);
22             HEX:out std_logic_vector(6 downto 0)
23         );
24     end component;
25
26     component digits is
27         port(
28             BIN:in integer range 0 to 9999;
29             num1: out integer range 0 to 9;
30             num0: out integer range 0 to 9
31         );
32     end component;
33
34     type state is(s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14, s15, s16, s17, s18, s19, s20, s21, s22, s23, s24, s25, s26, s27);
35     signal present_state:state;
36     signal next_state:state;
37     signal dcnt:std_logic_vector(24 downto 0):=(others => '0');
38     signal clk2hz:std_logic;
39     signal Apoint, Bpoint:std_logic_vector(3 downto 0):="0000";
40     signal flag:std_logic:='1';
41     signal Aplus, Bplus:std_logic:='0';
42
43     type INT_array is Array (0 to 1) of integer range 0 to 9;
44     signal Anum, Bnum:INT_array;
45
46     type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
47     signal show:LOGIC_array;
48 begin
49     process(clk)
50     begin
51         if clk'event and clk = '1' then
52             if dcnt = 24999999 then
53                 dcnt <= (others => '0');
54             else
55                 dcnt <= dcnt + 1;
56             end if;
57         end if;
58     end process;
59
60     clk2hz <= dcnt(24);
```

```

61
62 process(clk2hz,reset)
63 begin
64     if reset = '0' then
65         present_state <= s0;
66     elsif rising_edge(clk2hz) then
67         present_state <= next_state;
68     end if;
69 end process;
70
71 process(Aplus,reset)
72 begin
73     if reset = '0' or present_state = s0 then
74         Apoint <= "0000";
75     elsif rising_edge(Aplus) then
76         if Apoint < "1011" then
77             Apoint <= Apoint + '1';
78         end if;
79     end if;
80 end process;
81
82 process(Bplus,reset)
83 begin
84     if reset = '0' or present_state = s0 then
85         Bpoint <= "0000";
86     elsif rising_edge(Bplus) then
87         if Bpoint < "1011" then
88             Bpoint <= Bpoint + '1';
89         end if;
90     end if;
91 end process;
92
93 process(present_state, A, B)
94 begin
95     led <= (others => '0');
96     Aplus <= '0';
97     Bplus <= '0';
98     flag <= '1';
99
100     if present_state = s0 then
101         if A = '0' then
102             next_state <= s1;
103         else
104             next_state <= s0;
105         end if;
106         flag <= '1';
107
108     elsif present_state = s1 then
109         next_state <= s2;
110         led <= "1000000000";
111
112     elsif present_state = s2 then
113         next_state <= s3;
114         led <= "0100000000";
115
116     elsif present_state = s3 then
117         next_state <= s4;
118         led <= "0010000000";
119
120     elsif present_state = s4 then
121         next_state <= s5;
122         led <= "0001000000";
123
124     elsif present_state = s5 then
125         next_state <= s6;
126         led <= "0000100000";

```

-- 令歸零從 s0 狀態開始
 -- 否則等除頻器時脈，更新狀態
 -- 當接收到 A 得分(正緣)，更新分數
 -- 當接收到 B 得分(正緣)，更新分數
 -- 令 led 為 0 (默認狀態)
 -- 令 Aplus, Bplus 為 0 (默認狀態)
 -- 令 flag 為 1 (未進入11分顯示)
 -- 令 flag 為 1 (未進入11分顯示)


```

127
128     elsif present_state = s6 then
129         next_state <= s7;
130         led <= "0000010000";
131
132     elsif present_state = s7 then
133         next_state <= s8;
134         led <= "0000001000";
135
136     elsif present_state = s8 then
137         next_state <= s9;
138         led <= "0000000100";
139
140     elsif present_state = s9 then
141         if B = '0' then
142             next_state <= s20;
143         else
144             next_state <= s10;
145         end if;
146         led <= "0000000010";
147
148     elsif present_state = s10 then
149         if B = '0' then
150             next_state <= s11;
151         else
152             next_state <= s20;
153         end if;
154         led <= "0000000001";
155
156     elsif present_state = s20 then
157         Aplus <= '1';
158         if B = '0' then
159             next_state <= s21;
160         else
161             next_state <= s20;
162         end if;
163
164     elsif present_state = s21 then
165         next_state <= s11;
166         led <= "0000000001";
167
168     elsif present_state = s11 then
169         next_state <= s12;
170         led <= "0000000010";
171
172     elsif present_state = s12 then
173         next_state <= s13;
174         led <= "0000000100";
175
176     elsif present_state = s13 then
177         next_state <= s14;
178         led <= "0000001000";
179
180     elsif present_state = s14 then
181         next_state <= s15;
182         led <= "0000010000";
183
184     elsif present_state = s15 then
185         next_state <= s16;
186         led <= "0000100000";

```

-- B 提早揮，A 得分

-- B 漏接，A 得分

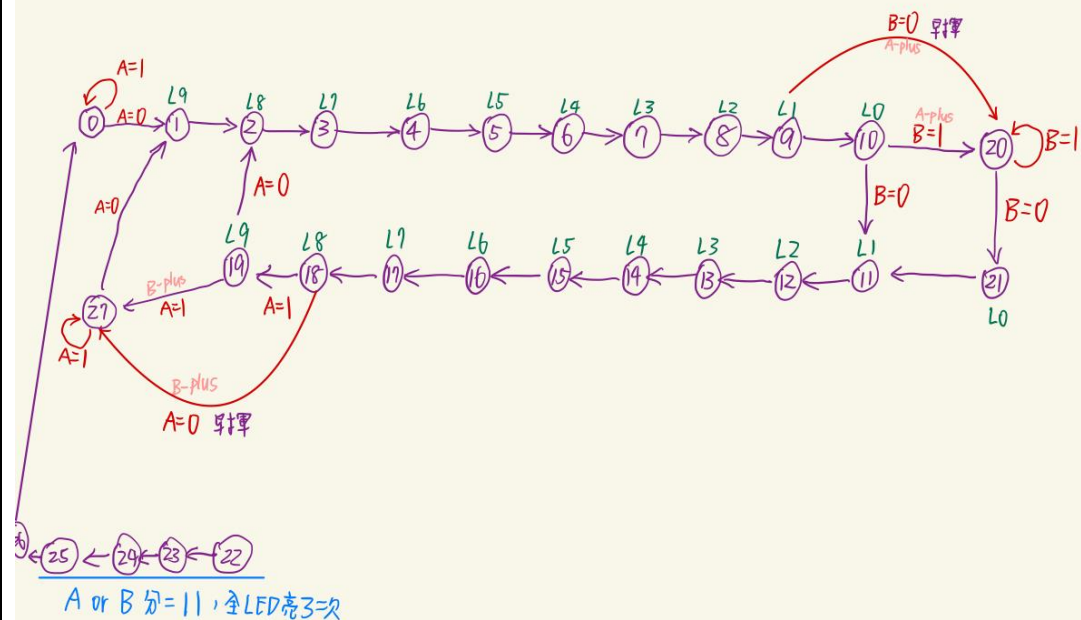
-- A 得分

```

187
188     elsif present_state = s16 then
189         next_state <= s17;
190         led <= "0001000000";
191
192     elsif present_state = s17 then
193         next_state <= s18;
194         led <= "0010000000";
195
196     elsif present_state = s18 then
197         if A = '0' then
198             next_state <= s27;
199             -- A 提早揮，B 得分
200         else
201             next_state <= s19;
202         end if;
203         led <= "0100000000";
204
205     elsif present_state = s19 then
206         if A = '0' then
207             next_state <= s2;
208             -- A 漏接，B 得分
209         else
210             next_state <= s27;
211         end if;
212         led <= "1000000000";
213
214     elsif present_state = s27 then
215         Bplus <= '1';
216         -- B 得分
217         if A = '0' then
218             next_state <= s1;
219         else
220             next_state <= s27;
221         end if;
222
223     elsif present_state = s22 then
224         -- 11分 (進入最後顯示)
225         next_state <= s23;
226         led <= (others => '1');
227         flag <= '0';
228         -- 令 flag 為 0 (更改默認 flag，使判斷不會再進入 s22)
229
230     elsif present_state = s23 then
231         next_state <= s24;
232         led <= (others => '0');
233         flag <= '0';
234         -- 令 flag 為 0 (更改默認 flag，使判斷不會再進入 s22)
235
236     elsif present_state = s24 then
237         next_state <= s25;
238         flag <= '0';
239         led <= (others => '1');
240         -- 令 flag 為 0 (更改默認 flag，使判斷不會再進入 s22)
241
242     elsif present_state = s25 then
243         next_state <= s26;
244         flag <= '0';
245         led <= (others => '0');
246         -- 令 flag 為 0 (更改默認 flag，使判斷不會再進入 s22)
247
248     elsif present_state = s26 then
249         next_state <= s0;
250         -- 重新開始 (回到 s0)
251         flag <= '0';
252         -- 令 flag 為 0 (更改默認 flag，使判斷不會再進入 s22)
253         led <= (others => '1');
254
255     end if;
256     if(((Apoint = "1011") or (Bpoint = "1011")) and (flag = '1'))then
257         -- 兩方中有一方得到 11 分，且未進入顯示狀態
258         next_state <= s22;
259         flag <= '0';
260         -- 令 flag 為 0 (更改默認 flag，使判斷不會再進入 s22)
261     end if;
262 end process;
263
264 U0:digits port map(conv_integer(Apoint), Anum(1), Anum(0));
265 -- 得分數字
266 U1:digits port map(conv_integer(Bpoint), Bnum(1), Bnum(0));
267 -- 顯示(int => std_logic_vector)
268 show(0) <= conv_std_logic_vector(Bnum(0), 4);
269 show(1) <= conv_std_logic_vector(Bnum(1), 4);
270 show(2) <= conv_std_logic_vector(Anum(0), 4);
271 show(3) <= conv_std_logic_vector(Anum(1), 4);
272
273 HEX0_part:decoder_7seg port map(show(0), HEX0);
274 -- 顯示(轉成七段顯示器)
275 HEX1_part:decoder_7seg port map(show(1), HEX1);
276 HEX2_part:decoder_7seg port map(show(2), HEX2);
277 HEX3_part:decoder_7seg port map(show(3), HEX3);
278
279 end ping_pong;

```

- ①若 A(B) lose, 則 B(A)得分, A(B)發球
- ②按下 reset, A發球
- ③得11分者勝, 10顆LED閃3次(加分)



心得

經過此次的實驗，我更了解何謂狀態圖的規劃，在每一次的狀態切換，都依然在自己預期的範圍內。在這一次作業中，我體悟最深的是何時要做分數的更改，如果在狀態 9、10 都做加分，可能因為偵測時根本還沒按，分數也會有所錯誤，故應該是在確定到某狀態後才加，除了保證結果的正確外，也讓自己觀念更加清楚。