第 3 講 設計結構

1. 參考文獻:

[1]Online Verilog Simulator: edaplayground 教學:
https://www.youtube.com/watch?v=f9uwtAax4v0&t=249s

[2]林灶生，verilog 晶片設計第 3 章，全華書局

[3]金門大學陳鐘誠教授網站，http://ccckmit.wikidot.com/ve:module

2. 基本觀念[2]**:**

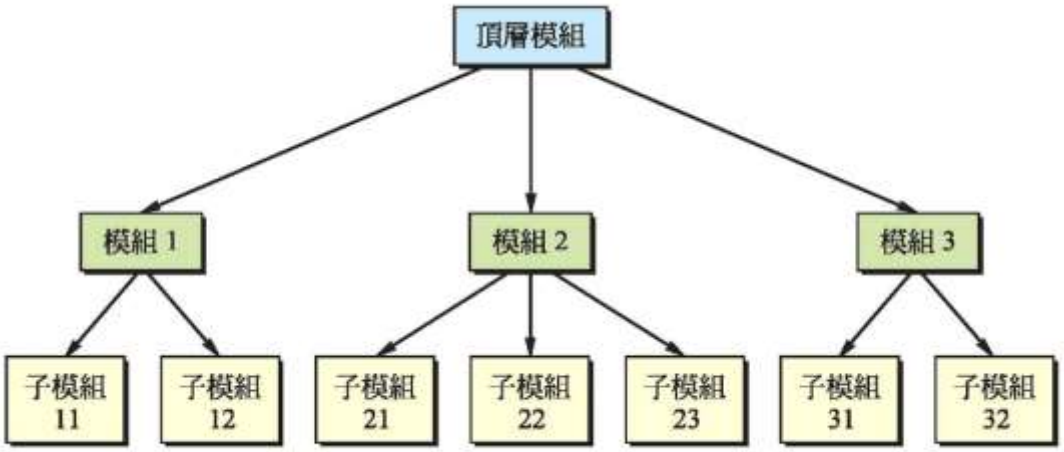


圖 3-1 由上而下設計階層圖

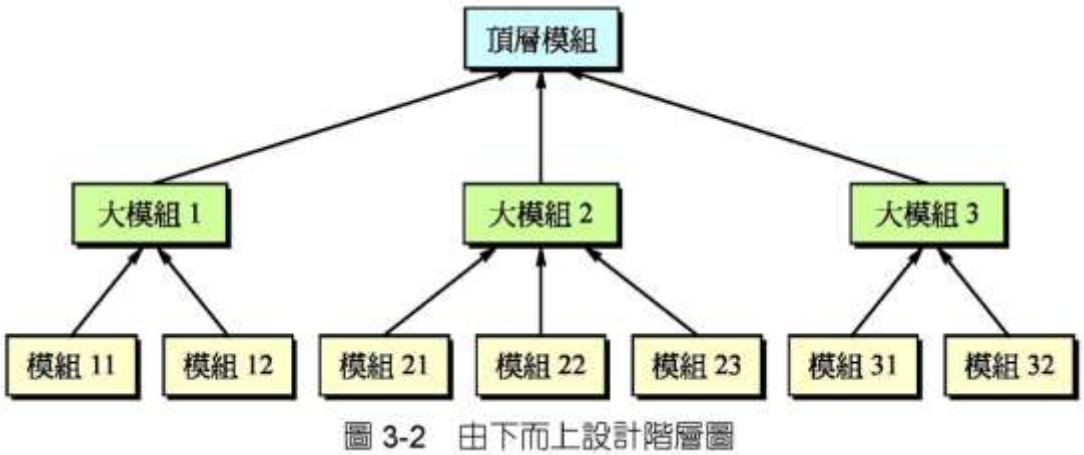


圖 3-2 由下而上設計階層圖

- 模組: 包括模組名稱(Module name)、輸出入埠定義(Inout/Output ports definition)、功能、低層元件、任務與函數等。
- 語法[3]:

```
module <module name> (<port list>);
<declares>
<module items>
endmodule
```

其中的 <module items> 可能是以下的語句：

1. initial
2. always
3. assign
4. module 的實例

```
module A(out1, out2, in1, in2);
    output out1;
    output [3:0] out2;
    input in1;
    input [2:0] in2;
     . . .
endmodule
```

或

```
module A(output out1, output [3:0] out2, input in1, input
[2:0] in2);
    . . .
Endmodule
```

- 模組實例化[3]

A x(x1, y1, a1, b1); // 按照位置一對一連接

A (x1, y1, a1, b1); // 省略名稱

A x(**.out**1(x1), **.out2**(y1), **.in1**(a1), **.in2**(b1)); //按照名稱進行連接

例題 1: 正反器選擇電路(依據順序)

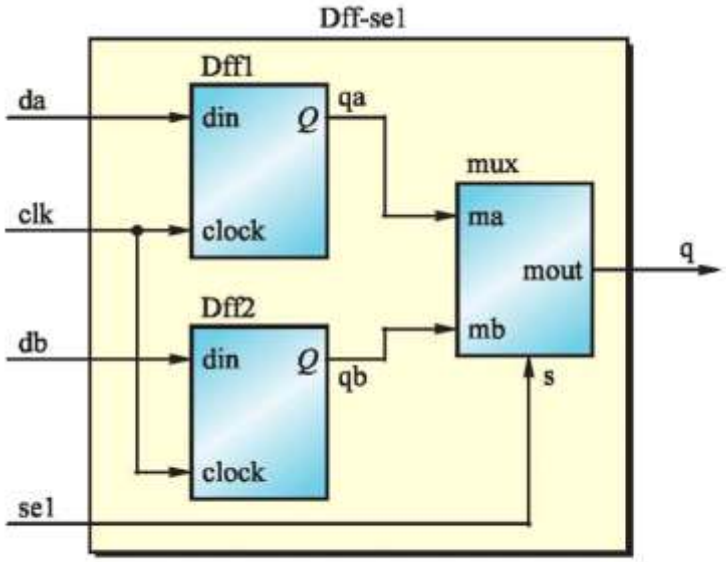
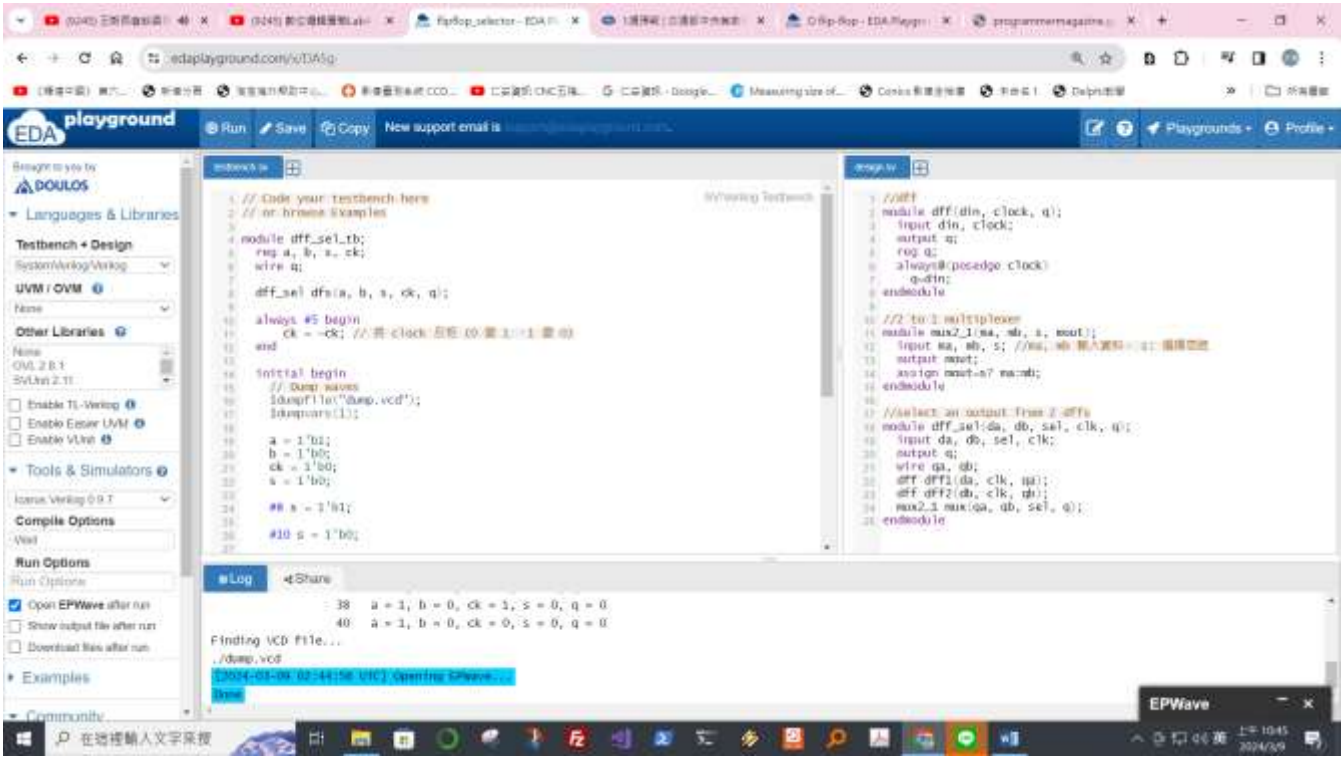
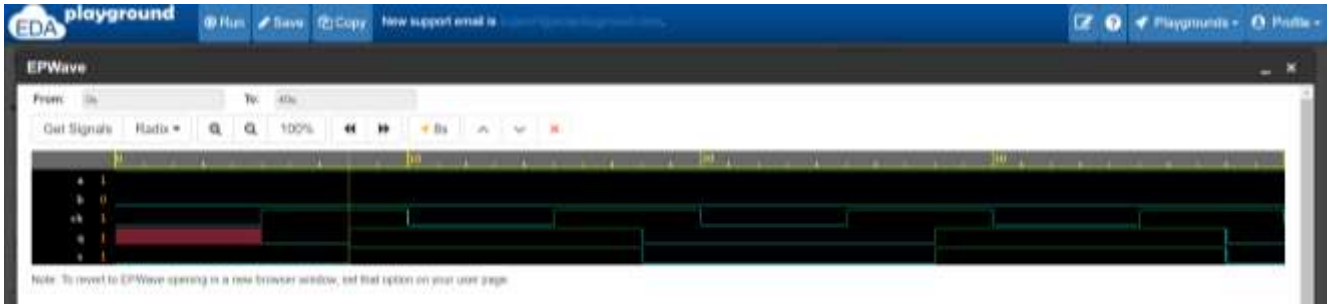

Dff-sel
Dff1
da
din
Q
qa
clk
clock
mux
ma
mout
q
Dff2
db
din
Q
qb
mb
s
clock
sel

圖 3-4　正反器選擇電路


EDA playground
Run
Save
Copy
Languages & Libraries
Tools & Simulators
Examples
EPWave


EDA playground
EPWave

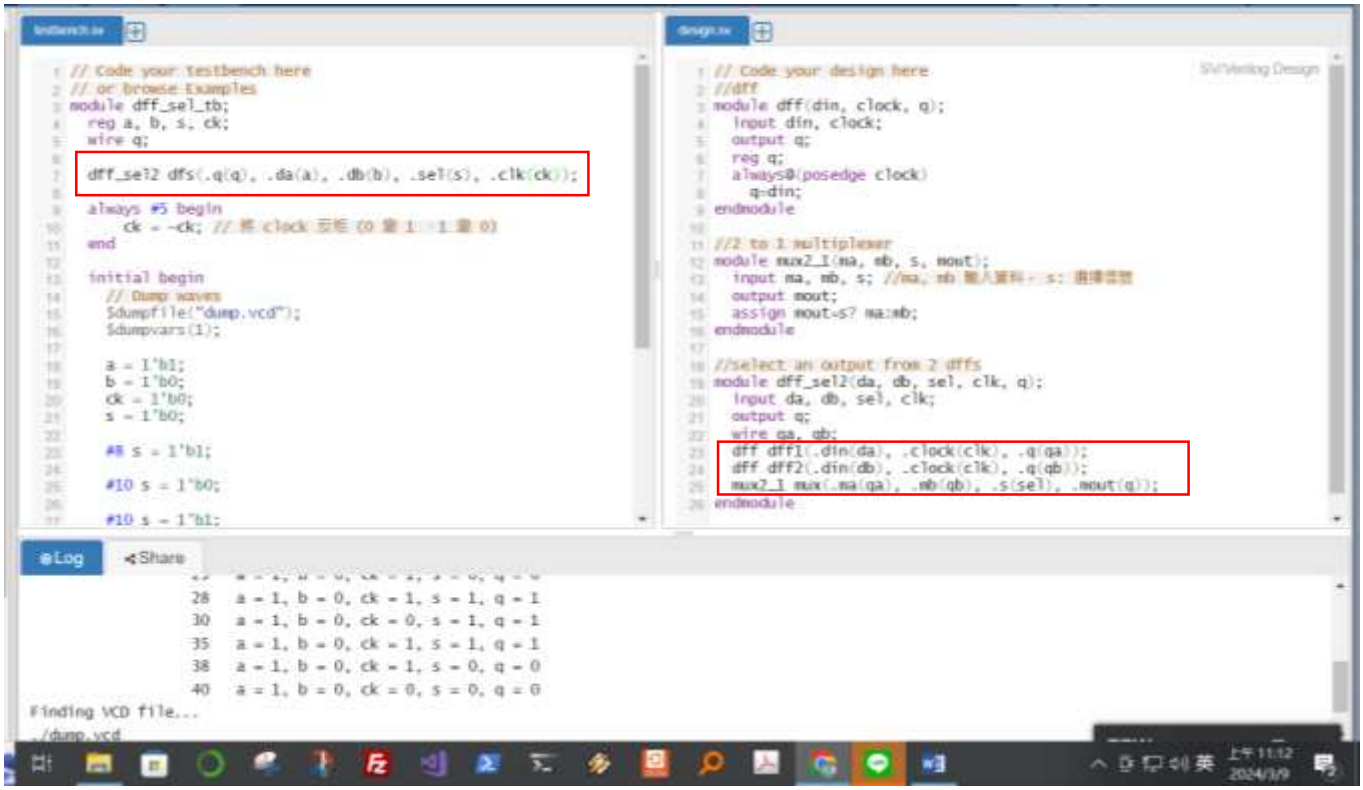
在 “design.sv”欄內的程式

```
//dff
module dff(din, clock, q);
   input din, clock;
   output q;
   reg q;
   always@(posedge clock)
      q=din;
endmodule

//2 to 1 multiplexer
module mux2_1(ma, mb, s, mout);
   input ma, mb, s; //ma, mb 輸入資料， s: 選擇信號
   output mout;
   assign mout=s? ma:mb;
endmodule

//select an output from 2 dffs
module dff_sel(da, db, sel, clk, q);
   input da, db, sel, clk;
   output q;
   wire qa, qb;
   dff dff1(da, clk, qa);
   dff dff2(db, clk, qb);
   mux2_1 mux(qa, qb, sel, q);
endmodule
```

在 “testbench.sv”欄內的程式

```
module dff_sel_tb;
   reg a, b, s, ck;
   wire q;

   dff_sel dfs(a, b, s, ck, q);

   always #5 begin
         ck = ~ck; // 每 5 秒將 clock 反相 (0 變 1 、1 變 0)
   end

   initial begin
      // Dump waves
      $dumpfile("dump.vcd");
      $dumpvars(1);

      a = 1'b1;
      b = 1'b0;
      ck = 1'b0;
      s = 1'b0;

      #8    s = 1'b1;    //延遲 8 秒, time = 8

      #10    s = 1'b0;    //再延遲 10 秒, time = 18

      #10    s = 1'b1;    //再延遲 10 秒, time = 28

      #10    s = 1'b0;    //再延遲 10 秒, time = 38

   end

   initial    #40    $finish;    //time = 40 結束

   initial begin
      $monitor($time, "      a = %b, b = %b, ck = %b, s = %b, q = %b", a,b,ck,s,q); //監視(示)
   end
endmodule
```

例題 2: 正反器選擇電路(依接腳名稱)




```
module dff_sel2(da, db, sel, clk, q);
    input da, db, sel, clk;
    output q;
    wire qa, qb;
    dff dff1(.din(da), .clock(clk), .q(qa));
    dff dff2(.din(db), .clock(clk), .q(qb));
    mux2_1 mux(.ma(qa), .mb(qb), .s(sel), .mout(q));
endmodule


module dff_sel_tb;
    reg a, b, s, ck;
    wire q;

    dff_sel2 dfs(.q(q), .da(a), .db(b), .sel(s), .clk(ck));
```

其餘程式同例題 1.

3. 資料型態

- 數值集合:

在 Verilog 中數值集合包含了以下四種基本數值：

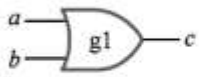
0 ：代表邏輯零電位，或條件式中之“假”值。

1 ：表示邏輯高電位，或條件式中之“真”值。

x ：表示未知的邏輯值。

z ：表示高阻抗狀態。

- 連接線(wire)[3]




```
reg   a , b;      //宣告 a,b 爲 reg 以保持上一狀態值
wire c;
or g1(c, a, b)    //將 OR 閘 g1 輸出連至 c，輸入連至 a 及 b
```

宣告一條線路 == > wire w1;

一次宣告很多條線路 == > wire w, x, y, z;

宣告一整個排線（例如匯流排) == > wire [31:0] bus;

一次宣告很多組排線 == > wire [31:0] bus [0:3];

- 暫存器(Registers)[3]: 穩定儲存位元的型態，稱為 reg (暫存器)

```
reg w;              // 宣告一位元的暫存器變數 w
reg x, y, z;        // 宣告三個一位元的暫存器變數 x, y, z
reg [31:0] r1;      // 宣告 32 位元的暫存器 r1
reg [31:0] R [0:15];   // 宣告 16 個 32 位元的暫存器群組 R[0..15]
```

- 基本邏輯閘[3]: 邏輯閘有 and, nand, or, nor, xor, xnor, not 等元件

```
module fulladder (input a, b, c_in, output sum, c_out);
        wire s1, c1, c2;
        xor g1(s1, a, b);
        xor g2(sum, s1, c_in);
        and g3(c1, a,b);
        and g4(c2, s1, c_in) ;
        or g5(c_out, c2, c1) ;
endmodule
```

- 三態(Tri-state): 輸出 “1”, “0”, “z”

```
module triBuffer (dbus, enable, value1);
    inout[3:0] dbus ;                    //Tri-state bus dbus
    input enable;
    input[3:0] value1;
    assign dbus = (enable ==1) ? value1 : 4'bz;
endmodule    //triBuffer
```

4. 作業 3-1: 參考以下的 module dflipflop 程式， 請寫一個 testbench 顯示 輸出入接腳 q, qbar, clk, rst, d 訊號圖

```
module dflipflop(q,qbar,clk,rst,d);
	output reg q;
	output qbar;
	input clk, rst;
	input d;

	assign qbar = ~q;

	always @(posedge clk)
	begin
		if (rst)
			q <= 0;
		else
			q <= d;
	end
endmodule
```

(1) 請編寫 testbench 程式、執行結果與圖形貼到作業報告中。