第 5 講 資料流描述設計

1. 參考文獻:

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

[2]金門大學陳鍾誠老師電子書**:** http://ccckmit.wikidot.com/ve:main

[3]林灶生，verilog 晶片設計第 5 章，全華書局

2. 基本觀念[3]

2.1 指定 (assign)

在資料流描述方法中，連續指定適用在 **wire**、**wand**、**wor**、或 **tri** 連接線上驅動並指定一表示式之運算值。其敘述之格式為：

---

assign ＜驅動強度＞ ＜延遲時間＞ ＜設定運算式＞;

例題 1 [2]:

```
wire [8:0] sum;
wire [7:0] a, b;
wire carryin;

assign sum = a + b + carryin;
// 當 a, b, carryin 有任何一個改變時，此語句都會被重新觸發。
// x = y 左邊的 x 必須是暫存器 (reg) 型態，不可是線路(wire)型態
```

例題 2 [2]: Block 與 Nonblocking 的比較範例，

```
// Blocking assignment (=)，其執行順序不一定，因此 d2, d3, d4 最後的結果值
// 將會不固定，這是一個不好的程式
reg d1, d2, d3, d4;
always @(posedge clk) d2 = d1;
always @(posedge clk) d3 = d2;
always @(posedge clk) d4 = d3;


// 以下的程式採用 NonBlocking assignment (<=)，所有同時的事件必須全部執行
// 完一遍後才能開始執行下一個時間點的事件，因此 d2, d3, d4 分別會得到上
```

// 一輪的 d1, d2, d3。

```
reg d1, d2, d3, d4;

always @(posedge clk) d2 <= d1;

always @(posedge clk) d3 <= d2;

always @(posedge clk) d4 <= d3;
```

2.2 表示法

- 常數表示法

例題 3 [3]:

```
Module and_or(out, op, in1, in2)
    parameter op=1'b1;
    parameter length = 8;
    output [length-1 : 0] out;
    input [length-1 : 0] in1, in2;
    assign out=(op==1'b1)?  In1&in2: in1|in2;
endmodule
```

- 運算元

例題 4 [3]:

```
wire[3:0] a, b, c, d;
assign c = a + b;
assign d[0] = a[0]&b[0];
integer e, f, g;
g = e - f;
```

例題 5 [3]:

```
wire [2:0] short_bits;
wire [4:0] long_bits;
wire [7:0] byte;
assign byte = {short_bits, long_bits};
```

- 運算子[3]

表 5-1　Verilog 運算子

運算子型態	運算子	說　　明
算術運算子	+, - , * , / %	算術加減、乘、除 求餘數
邏輯運算子	! && ||	邏輯 not 邏輯 and 邏輯 or
關係運算子	>> >= < <=	大於 大於等於 小於 小於等於
等於運算子	= = != = = = != =	等於 不等於 狀況等於(case equality) 狀況不等於(case inequality)
位元邏輯 運算子	~ & | ^ ^~或~^	位元反邏輯 位元 and 位元 or 位元 xor 位元 nxor
精簡邏輯 運算子	& | ~& ~| ^ ^~ 或~^	精簡 and 精簡 or 精簡 nand 精簡 nor 精簡 xor 精簡 xnor
移位運算子	<< >>	左移 右移
條件運算子	? :	條件指定
連結運算子	{ }	連結

例題 6 [3]: 4 bits comparator (比較器)

testbench.sv

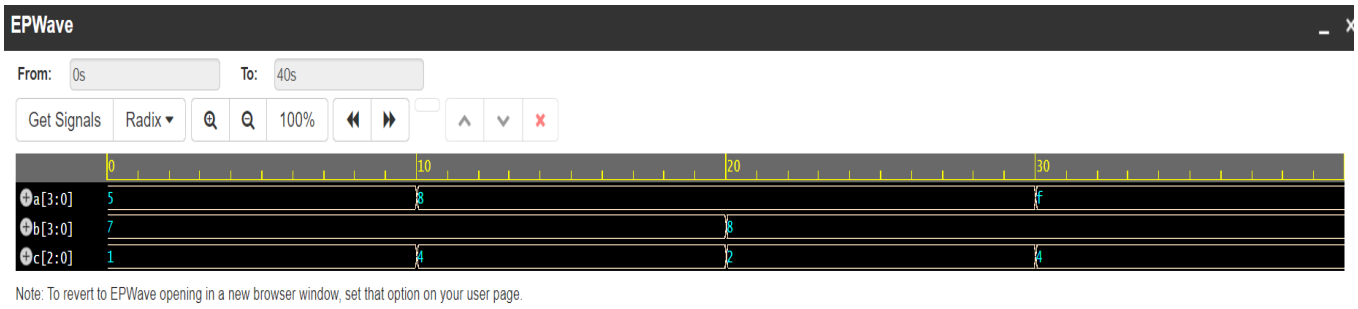
```
// Code your testbench here
// or browse Examples
module compare_tb;
  reg[3:0] a, b;
  wire[2:0] c;
  compare1 U1(a, b, c);

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

    a=4'b0101; b=4'b0111; //time = 0s

    #10  a=4'b1000; //time = 10s

    #10  b=4'b1000;//time = 20s

    #10  a=4'b1111;//time = 30s

   end

  initial #40 $finish;

  initial begin
```

design.sv

```
// Code your design here
module compare1(a, b, comout);
  input[3:0] a, b;
  output[2:0] comout;

  reg[2:0] tmp_dat;

  always @ (a or b)
    begin
      if (a > b)
        tmp_dat <= 3'b100;
      else if (a == b)
        tmp_dat <= 3'b010;
      else
        tmp_dat <= 3'b001;
    end

  assign comout = tmp_dat;

endmodule
```

Log | Share

```
[2024-03-17 12:28:43 UTC] iverilog design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
                   0    a=0101, b = 0111, c= 001
                  10    a=1000, b = 0111, c= 100
                  20    a=1000, b = 1000, c= 010
                  30    a=1111, b = 1000, c= 100
Finding VCD file...
```



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

以下為 design.sv 攔內的模組程式

```
module compare1(a, b, comout);
   input[3:0] a, b;
   output[2:0] comout;

   reg[2:0] tmp_dat;

   always @ (a or b)
      begin
         if (a > b)
```

```
            tmp_dat <= 3'b100;
        else if (a == b)
            tmp_dat <= 3'b010;
        else
            tmp_dat <= 3'b001;
    end

  assign comout = tmp_dat;

endmodule
```

以下為 testbench.sv 欄內的測試程式

```
module compare_tb;
  reg[3:0] a, b;
  wire[2:0] c;
  compare1 U1(a, b, c);

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

    a=4'b0101; b=4'b0111; //time = 0s

    #10   a=4'b1000; //time = 10s

    #10   b=4'b1000;//time = 20s

    #10   a=4'b1111;//time = 30s

   end

  initial #40 $finish;

  initial begin
    $monitor($time, "       a=%4b, b = %4b, c= %3b", a,b,c);
  end
endmodule
```

3. 作業 5-1: 4 bit adder

	B3	B2	B1	B0
	A3	A2	A1	A0
	+			Cin
Cout	S3	S2	S1	S0

以下為 design.sv 攔內的模組程式

```
module adder4(A, B, Cin, S, Cout);
   input [3:0] A,B;
   input Cin;
   output [3:0] S;
   output Cout;

   assign {Cout, S} = A + B+ Cin;

endmodule
```

(1) 請編寫測試程式，可執行上述 4 bit adder 程式，將結果貼到作業報告中。