Department of Computer Science and Information Engineering

National Cheng Kung University

Weekend Practice 11/6

Verilog 基本語法

- 變數: wire、reg
- 算術運算: + 、- 、* 、/ 、>> 、<<
- 位元運算子: &、|、^、~
- 邏輯運算子: ! 、&&、||
- 關係運算子: > 、>= 、< 、<= 、==
- 表示數字時可使用'b 、'o 、'd 、'h
  - 2'b10 = 2 (十進制)
  - 2'o10 = 8 (十進制)
  - 2'd10 = 10 (十進制)
  - 2'h10 = 16 (十進制)

Concatenation 語法

- 利用大括弧{}來串接多個訊號，使其作為單一訊號進行操作
- 範例一: 組合訊號線方便後續使用(依照串接的順序構成訊號)
  - wire a, b, c, d, e;
  - assign a = 1'b1;
  - assign b = 1'b0;
  - assign c = 1'b1;
  - assign d = 1'b0;

  - Ex1: assign e = {a, b, c, d}; // e = 4'b1010
  - Ex2: assign e = {a, c, b, d}; // e = 4'b1100
  - Ex3: wire [3:0] f;
    assign f = {3'd0, a}; // f = 4'b0001

Concatenation 語法

- 利用大括弧{}來串接多個訊號，使其作為單一訊號進行操作
- 範例二: 組合訊號線接收運算結果(由左至右分配bit)
  - EX1: wire carry, sum;

```
assign {carry, sum} = 1'b1 + 1'b1; // carry = 1'b1; sum = 1'b0
// 1'b1 + 1'b1 = 2'b10;
```

  - Ex2: wire overflow;

```
wire [1:0] a, b, sum;
assign {a, b} = 2'b10, 2'b01};
assign {overflow, sum} = a + b; // a+b=3'b011; overflow=1'b0, sum = 2'b11
assign {a, b} = 4'b1111; // a = 2'b11; b = 2'b11
assign {overflow, sum} = a + b; // a+b=3'b110; overflow=1'b1, sum = 2'b10
```

Lab1 – arithmetic shift right

- 右移運算分為邏輯右移和算數位移
  - 邏輯右移時最左位補0，EX: 4'b1111 >> 1 = 4'b0111
  - 算數右移時最左位補上sign bit，EX: 4'b1010 >> 1 = 1101

- 當訊號超過1 bit時，可利用bit selector選擇取用哪個bit進行操作
  - EX: wire [3:0] a;

    assign a = 4'b1101;

    // a[3] = 1; a[2] = 1; a[1] = 0; a[0] = 1
  - 可一次選擇多個bits，EX: a[2:1] = 2'b10

- 請使用concatenation搭配bit selector完成一算數右移器
  - 輸入: in(4 bits)
  - 輸出: out(4 bits)，為in算數右移1後的結果

case 語法

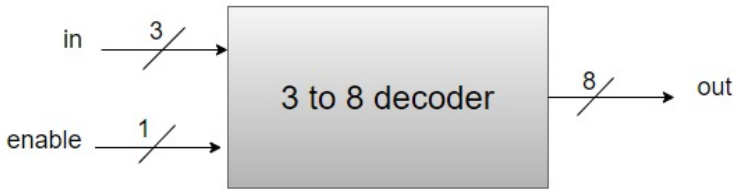
- 依照case判定訊號的情況執行不同的程式
- 在組合電路中，務必將訊號在每種情況下的值都給定，避免產生額外的記憶單元
- EX: input [1:0] in;

```
output reg [1:0] out;
always@(in) begin
    case(in)
    2'b00: out = 2'b11;
    2'b11: out = 2'b00;
    default: out = 2'b10;
    endcase
end
```

Lab2 -- decoder to DE0-CV

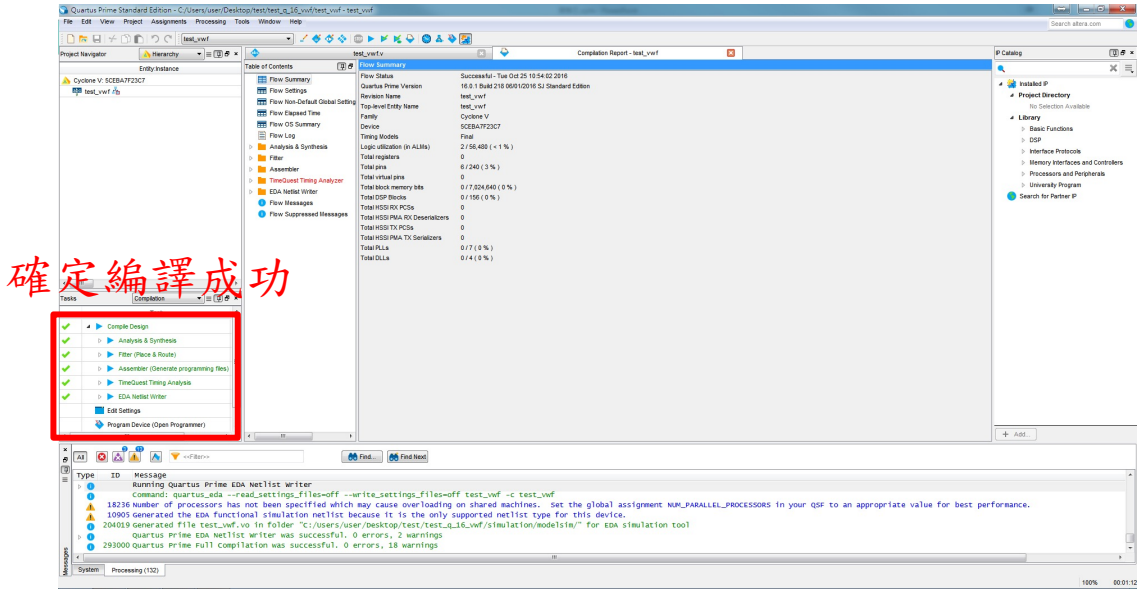
- 請設計一3對8解碼器(3 to 8 decoder)
- 解碼器可以將n個輸入訊號轉換成2^n位元輸出訊號，假設有m個輸入與n個輸出，則稱為m對n解碼器
- Hint: 可使用behavior description之case語法實作

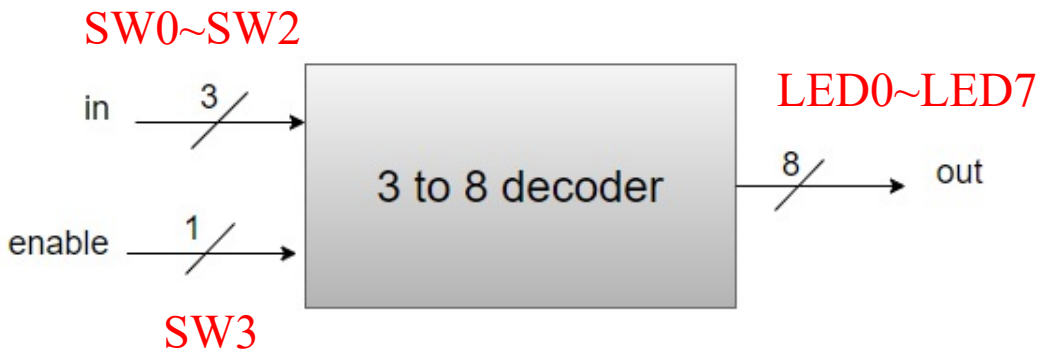
輸入 (input)				輸出 (output)							
enable	in[2]	in[1]	in[0]	out[7]	out[6]	out[5]	out[4]	out[3]	out[2]	out[1]	out[0]
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0



Lab2 -- decoder to DE0-CV

- 完成verilog電路設計後，需先確認其在Quartus可順利編譯，再將其燒錄至DE0-CV開發板進行驗證
- 使用Switch(SW0~SW2)控制input訊號，Switch(SW3)控制enable訊號，使用LED(LED0~LED7)表示output





Appendix

- 為了幫助同學驗證程式是否正確，moodle上有提供testbench
- 驗證方式:
  - Step 1: 前往verilog線上模擬網站: https://www.jdoodle.com/execute-verilog-online/
  - Step 2: 將moodle上的testbench code整段貼上
  - Step 3: 修改adder_subtractor/encoder module
  - Step 4: 按下Execute按鈕，確認模擬結果是否正確

Appendix

- Lab 1 模擬結果

Result

CPU Time: 0.00 sec(s), Memory: 7120 kilobyte(s)

```
in = 1010; The result of arithmetically shift right = 1101
```

Appendix

- Lab 2 模擬結果

Result

CPU Time: 0.00 sec(s), Memory: 7168 kilobyte(s)

```
in = 000, enable = 0; out = 00000000
in = 001, enable = 0; out = 00000000
in = 010, enable = 0; out = 00000000
in = 011, enable = 0; out = 00000000
in = 100, enable = 0; out = 00000000
in = 101, enable = 0; out = 00000000
in = 110, enable = 0; out = 00000000
in = 111, enable = 0; out = 00000000
in = 000, enable = 1; out = 00000001
in = 001, enable = 1; out = 00000010
in = 010, enable = 1; out = 00000100
in = 011, enable = 1; out = 00001000
in = 100, enable = 1; out = 00010000
in = 101, enable = 1; out = 00100000
in = 110, enable = 1; out = 01000000
in = 111, enable = 1; out = 10000000
```