

Laboratory 8

Verilog 的行為模型

與

七段顯示器、開關器與按鈕器之應用



Department of Electrical Engineering
National Cheng Kung University

實驗目的

- 瞭解 Verilog 的行為模型(Behavior Modeling)
- 利用FPGA板與虛擬元實作多種硬體電路應用

使用器材

- 桌上型電腦
- Xilinx FPGA 板

Verilog 的行為模型

以下說明行為模型的基本概念

Verilog - 運作模式

Verilog的運作模式主要由兩種模型所構成

1. 結構模型 (Structural Modeling)

- 以網路連線 (netlist) 的方式描述元件和元件之間如何連接起來。

2. 行為模型 (Behavioral Modeling)

- 以抽象的方式來描述電路與測試程式(test bench)，語法更加彈性。

Verilog - 行為模型

- Verilog可以讓設計者用演算法來描述設計的功能，也就是電路的行為(behavior)。
- 因此，行為模型是用高階抽象的方式來描述電路。其較類似於C語言，而不像數位電路設計。
 - 例如: assign, case, if-else, for loop 等語法
- 利用結構化程序(Structural Procedures)來實踐行為模型
 - 在Verilog中有兩個結構化程序: initial 與 always

Verilog - initial block

- initial 區塊啟動於模擬時間零，且僅執行一次
 - 為序向區塊，每一個敘述會依照其順序執行
 - 如果在 initial 區塊中有多個敘述，則需用 **begin...end** 做群聚(grouped)
- initial 區塊無法合成實體電路，僅只用於模擬

- #1: 延遲一個模擬時間
- \$finish: 結束程式

initial
begin

多個敘述

end

initial
begin

A = 0; B = 0;

#1 A = 1; B = 0;

#1 A = 0; B = 1;

#1 A = 1; B = 1;

#1 \$finish;

end

Verilog - always block (1/2)

- always區塊是用來模組化一個持續重複工作的數位邏輯電路
 - 為序向區塊，每一個敘述會依照其順序執行
 - 敘述的左側變數必須是reg，右側則可以是wire或reg
- 當@中的事件有變化(0→1或1→0)時，將會執行always區塊

```
always @(事件1, 事件2, ...)  
begin  
  
    多個敘述  
  
end
```

```
always@(A or B or Sel)  
begin  
    if(Sel)  
        C = A;  
    else  
        C = B;  
end
```


Verilog - always block (2/2)

➤ 利用always區塊設計組合邏輯與序向邏輯時必須注意的差別

➤ 組合邏輯 (Combinational Logic)

- 無記憶功能，與時間無關
- 布林邏輯、加法器、解碼器等等
- 使用 **Blocking (=)** 符號

```
always@(A or B or SEL) // 組合邏輯用 =
begin
    if ( SEL)           (Blocking)
        C = A;
    else
        C = B;
end
```

➤ 序向邏輯 (Sequential Logic)

- 具有記憶功能，且與時間相關
- 正反器、計數器、狀態機等等
- 使用 **Non-blocking (<=)** 符號

```
always@(posedge clk or posedge reset) // 序向邏輯用 <=
begin
    if (reset)           (Non-blocking)
        dout <= 1'b0;
    else
        dout <= din;
end
```

Verilog - if else 敘述

➤ if-else 敘述

- 語法類似C語言
- 支援巢狀分支
- 例子: 解碼器

```
If (expression)  
    敘述  
else if (expression)  
    敘述  
else  
    敘述
```

```
always@(*)  
begin  
    if (din==2'b00)  
        dout = 4'b0001;  
    else if (din==2'b01)  
        dout = 4'b0010;  
    else if (din==2'b10)  
        dout = 4'b0100;  
    else  
        dout = 4'b1000;  
end
```

Verilog - case 敘述

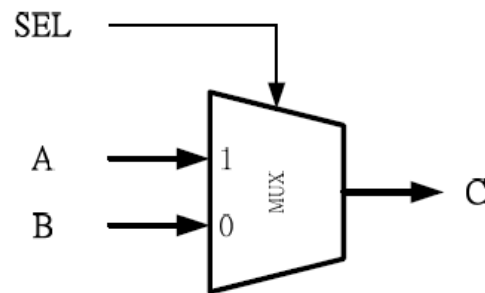
➤ case敘述

- 如果判定對象是固定個，我們可以使用case的方式來敘述會比較方便。
- 語法類似C語言，且支援巢狀分支
- 例子: 解碼器

```
case (expression)
    alternative 1 : statement 1;
    alternative 2 : statement 2;
    ...
    default: default_statement;
endcase
```

```
always@(*)
begin
    case(din)
        2'b00: dout = 4'b0001;
        2'b01: dout = 4'b0010;
        2'b10: dout = 4'b0100;
        2'b11: dout = 4'b1000;
        default: dout = 4'b0000;
    endcase
end
```

範例：多工器



```
assign    C = ( SEL ) ? A : B ;    // C要宣告成wire
```

```
always@(A or B or SEL)             // C要宣告成reg
begin
    if ( SEL )
        C = A;
    else
        C = B;
end
```

範例: T Flip-Flop

```
module tff (clk, reset, t, q, nq);  
  
    input      clk, reset, t;  
    output     q , nq;  
    reg        data;  
  
    always@( posedge clk )  
    begin  
        if (reset)  
            data <= 1'b0;  
        else if (t)  
            data <= ~data;  
    end  
    assign q=data;  
    assign nq=~data;  
  
endmodule
```

實作題

本次實作分為四個基本實作與一個挑戰實作

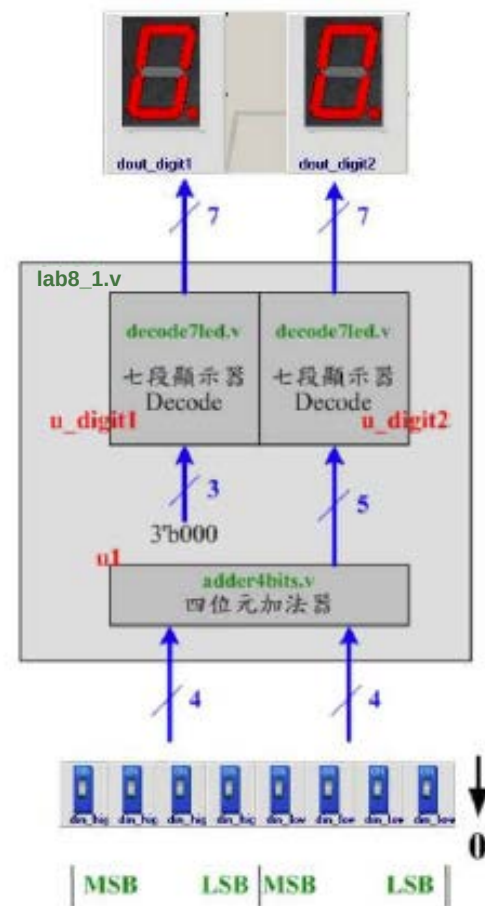
實作題(一) 七段顯示器應用 (1/3)

➤ 利用七段顯示器來顯示四位元加法器的結果

- 輸入: Switch * 8
- 輸出: 七段顯示器 * 2

➤ Switch 1~4 和 Switch 5~8 為兩個 4-bit 的輸入值

din_high 及 din_low。請利用 Verilog 的語法將輸入的兩個值相加，並將結果輸出至兩個七段顯示器。

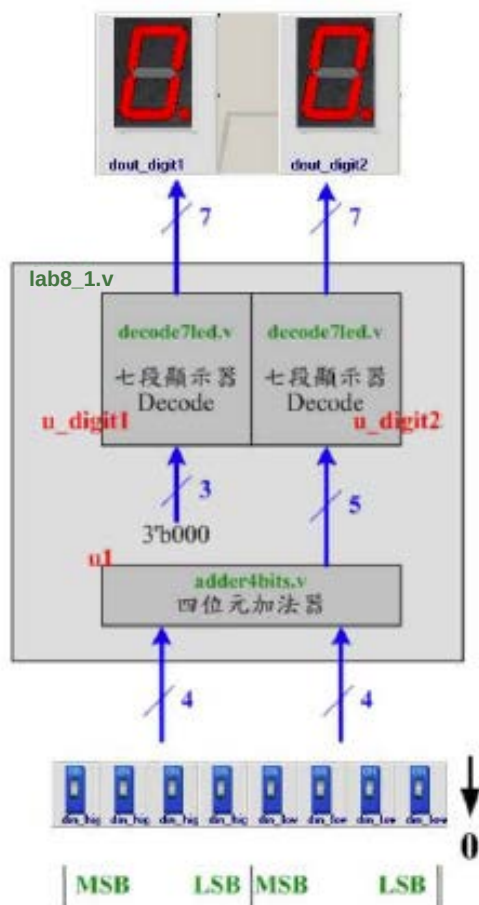


實作題(一) 七段顯示器應用 (2/3)

步驟

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab8_1
2. 開啟 VeriInstrumentV3，並繼續完成題目所要求的電路後
download 至 VeriLite FPGA中驗證

實作題(一) 七段顯示器應用 (3/3)



```
module adder4bits(A, B, Carry, Sum);

    input  [3:0]  A;
    input  [3:0]  B;
    output        Carry;
    output [3:0]  Sum;

    wire  [4:0]  result;

    assign result = A + B ;

    assign Carry = result[4];
    assign Sum = result[3:0];

endmodule
```

adder4bits.v

```
module SevenSeg ( din, dout );

    input  [3:0]  din;
    output [6:0]  dout;

    reg  [6:0]  dout;

    always@(din)
    begin
        case(din)
            4'b0000 : dout = 7'b0111111 ;
            4'b0001 : dout = 7'b0000110 ;
            4'b0010 : dout = 7'b1011011 ;
            4'b0011 : dout = 7'b1001111 ;
            4'b0100 : dout = 7'b1100110 ;
            4'b0101 : dout = 7'b1101101 ;
            4'b0110 : dout = 7'b1111101 ;
            4'b0111 : dout = 7'b0000111 ;
            4'b1000 : dout = 7'b1111111 ;
            4'b1001 : dout = 7'b1101111 ;
            4'b1010 : dout = 7'b1110111 ;
            4'b1011 : dout = 7'b1111100 ;
            4'b1100 : dout = 7'b0111001 ;
            4'b1101 : dout = 7'b1011110 ;
            4'b1110 : dout = 7'b1111001 ;
            4'b1111 : dout = 7'b1110001 ;
            default : dout = 7'b0000000 ;
        endcase
    end
endmodule
```

SevenSeg.v

```
module lab8_1(din_high, din_low, dout_digit1, dout_digit2,
              dout_dp1, dout_dp2, clk);

    input  clk ;
    input [3:0] din_high;
    input [3:0] din_low;

    output [6:0] dout_digit1;
    output [6:0] dout_digit2;
    output      dout_dp1;
    output      dout_dp2;

    wire  [7:0]  value;

    adder4bits u1(
        .A      (din_high),
        .B      (din_low),
        .Carry   (value[4]),
        .Sum     (value[3:0])
    );

    assign value[7:5] = 3'b000 ;

    SevenSeg  u_digit1(value[7:4], dout_digit1);
    SevenSeg  u_digit2(value[3:0], dout_digit2);

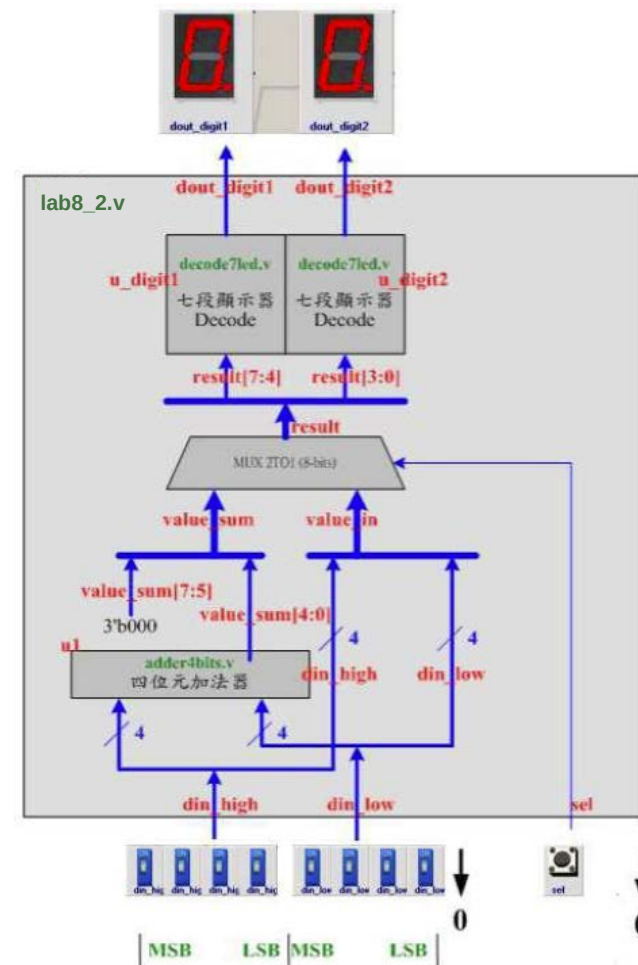
    assign dout_dp1 = 1'b1;
    assign dout_dp2 = 1'b1;

endmodule
```

lab8_1.v

實作題(二) 多工器應用 (1/3)

- 延續上個實作，新增多工器來顯示輸入數值
 - 輸入: Switch * 8, Button * 1
 - 輸出: 七段顯示器 * 2
- 接續lab8_1 的內容中，加入一個 2 對 1 的多工器，讓PushBottom 來選擇輸出到雙七段顯示器的值是輸入的值($\text{din_high} \rightarrow \text{digit1}$ ， $\text{din_low} \rightarrow \text{digit2}$) 或是加總後的值。(PushButton 壓下去時值為0，此時雙七段顯示器顯示加總後的值)

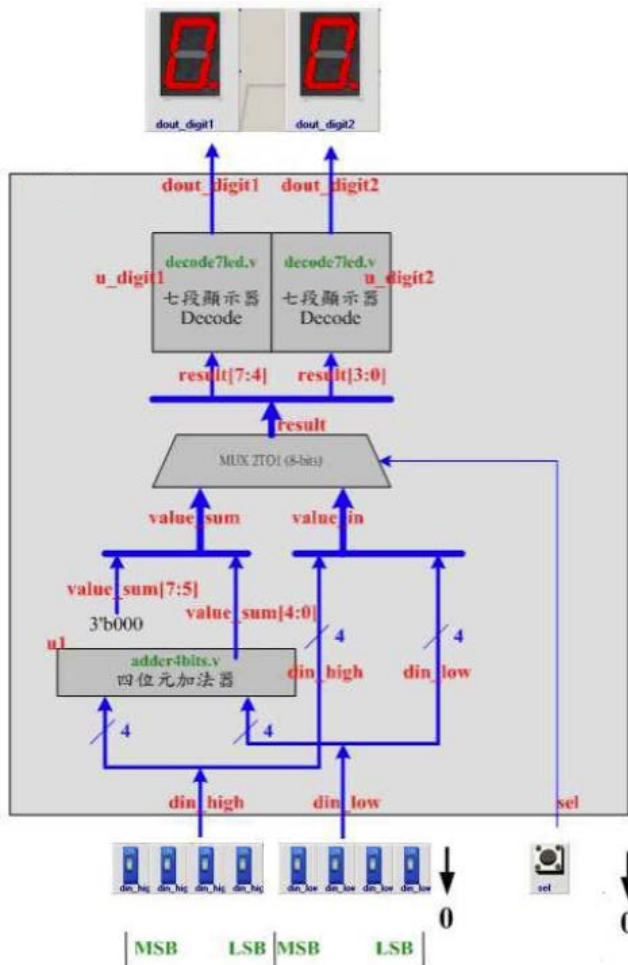


實作題(二) 多工器應用 (2/3)

步驟

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab8_2
2. 開啟 VeriInstrumentV3，並繼續完成題目所要求的電路後
download 至 VeriLite FPGA中驗證

實作題(二) 多工器應用 (3/3)



```

module lab8_2(sel, din_high, din_low, dout_digit1, dout_digit2,
              dout_dp1, dout_dp2, clk );

  input      sel, clk ;
  input [3:0] din_high;
  input [3:0] din_low;

  output [6:0] dout_digit1;
  output [6:0] dout_digit2;
  output      dout_dp1;
  output      dout_dp2;

  wire [7:0] value_in;
  wire [7:0] value_sum;
  wire [7:0] result;

  adder4bits u1(
    .A      (din_high),
    .B      (din_low),
    .Carry  (value_sum[4]),
    .Sum    (value_sum[3:0])
  );

  assign value_sum[7:5] = 3'b000;

  assign value_in = {din_high, din_low};
  assign result = (sel == 1'b0) ? value_sum : value_in;

  SevenSeg u_digit1(result[7:4], dout_digit1);
  SevenSeg u_digit2(result[3:0], dout_digit2);

  assign dout_dp1 = 1'b1;
  assign dout_dp2 = 1'b1;

endmodule

```

lab8_2.v

實作題(三) case應用 (1/3)

- 利用always區塊與case語法設計編碼器
 - 輸入: Switch * 8
 - 輸出: 七段顯示器 * 1
- 在七段顯示器之上顯示 Switch1~8 中的哪個 Bit 被設立為 1，如果同時有兩個 Bit 以上的 Switch 被設定為 1，則七段顯示器顯示 0。

實作題(三) case應用 (2/3)

步驟

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab8_3
2. 開啟 VeriInstrumentV3，並繼續完成題目所要求的電路後
download 至 VeriLite FPGA中驗證

實作題(三) case應用 (3/3)

```
module lab8_3(din, dout_digit , clk);  
    input    clk    ;  
    input [7:0]  din;  
    output [6:0]  dout_digit;  
  
    reg [3:0]  value;  
  
    always@(din)  
    begin  
        case(din)  
            8'b10000000:  value = 4'b0001;  
            8'b01000000:  value = 4'b0010;  
            8'b00100000:  value = 4'b0011;  
            8'b00010000:  value = 4'b0100;  
            8'b00001000:  value = 4'b0101;  
            8'b00000100:  value = 4'b0110;  
            8'b00000010:  value = 4'b0111;  
            8'b00000001:  value = 4'b1000;  
            default:      value = 4'b0000;  
        endcase  
    end  
  
    SevenSeg u_digit(value, dout_digit);  
  
endmodule
```

lab8_3.v

實作題(四) 比較器應用 (1/3)

➤比較兩個數字的大小並且透過七段顯示器顯示結果

- 輸入: Switch * 8
- 輸出: 七段顯示器 * 2

➤ Switch1~4 和 Switch5~8 為兩個 4Bits 的輸入值 din_high 及 din_low。

請將輸入的兩個輸入輸出至雙七顯示器(din_high → digit1，din_low → digit2)，
但是判斷輸入的兩個值哪一個大，則那一邊的七段顯示器小數點(Decimal Point)
會亮，如果兩值相同，則兩邊都不亮。

實作題(四) 比較器應用 (2/3)

步驟

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab8_4
2. 開啟 VeriInstrumentV3，並繼續完成題目所要求的電路後
download 至 VeriLite

實作題(四) 比較器應用 (3/3)

```
module lab8_4(din_high, din_low, dout_digit1, dout_digit2,  
              dout_dp1, dout_dp2, clk );  
  
    input      clk      ;  
    input [3:0] din_high;  
    input [3:0] din_low;  
  
    output [6:0] dout_digit1;  
    output [6:0] dout_digit2;  
    output      dout_dp1;  
    output      dout_dp2;  
  
    SevenSeg u_digit1(din_high, dout_digit1);  
    SevenSeg u_digit2(din_low, dout_digit2);  
  
    assign dout_dp1 = (din_high > din_low) ? 1'b1 : 1'b0 ;  
    assign dout_dp2 = (din_high < din_low) ? 1'b1 : 1'b0 ;  
  
endmodule
```

lab8_4.v

挑戰題(一) 排序顯示器

➤ 依大小順序將兩個數入數值顯示在七段顯示器上

- 輸入: Switch * 8
- 輸出: 七段顯示器 * 2

➤ Switch1~4 和 Switch5~8 為兩個 4Bits 的輸入值 `din_high` 及 `din_low`。

請將輸入的兩個輸入輸出至雙七顯示器，但是 `digit1` 兩個值較大的值，`digit2`顯示較小的值。

由`dp1`顯示輸入的兩個值是否有交換。

(假設 `din_high` → `digit1`，`din_low` → `digit2`) 而`dp2`顯示兩個值是否相同。

Verilog 參考資料

- Verilog 硬體描述語言(Verilog HDL)第二版
原著:Samir Palnitkar
原出版社: Prentice Hall
編譯: 黃英叡, 黃稚存, 張銓淵, 江文啟
全華科技圖書
- Verilog HDL: A Guide to Digital Design and Synthesis," 2nd ed. by Samir Palnitkar, Prentice Hall, 2003. ISBN: 0-13-044911-3
- <http://www.asic-world.com/verilog>