

Hardware Description Language - Verilog

Outline

- Verilog 介紹
- Verilog的寫法

Verilog介紹

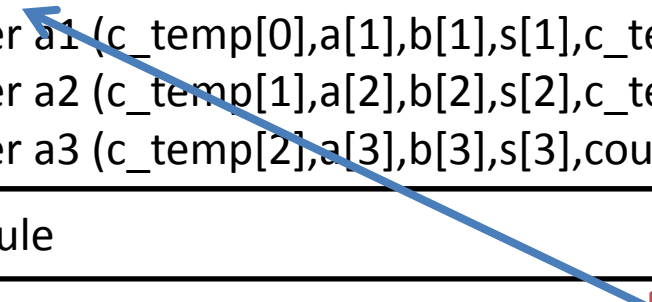
- 硬體描述語言
 - 描述” ” 硬體” 的語言
 - 先有硬體概念，再利用語言描述出來
- 兩種IEEE標準的硬體描述語言：
 - VHDL
 - Very-High-Speed Integrated Circuit **H**ardware **D**escription Language
 - Verilog
 - 容易上手與撰寫(語法類似C語言)
 - 活躍於學術界與業界

撰寫描述語言之前

- 所設計的電路所具有的功能是什麼？
- 規劃電路與外部溝通的介面(I/O)有哪些？

Verilog語言寫法

<pre>module Adder4 (cin,a,b, s,cout);</pre>	宣告模組名稱與IO
<pre> input cin; input [3:0] a,b; output [3:0] s; output cout; wire [2:0] c_temp; //中間結點</pre>	定義IO類型與中間結點
<pre> adder a0 (cin,a[0],b[0],s[0],c_temp[0]); adder a1 (c_temp[0],a[1],b[1],s[1],c_temp[1]); adder a2 (c_temp[1],a[2],b[2],s[2],c_temp[2]); adder a3 (c_temp[2],a[3],b[3],s[3],cout);</pre>	硬體電路描述
<pre>endmodule</pre>	模組結束



```
module adder (cin,a,b,s,cout);  
    input  a,b,cin;  
    output s,cout;  
  
    assign {s,cout} = cin + a + b;  
endmodule
```

常用資料型態的宣告

- 輸入: input ex: input [2:0] a;
- 輸出: output ex: output cout;
- 中間結點: wire ex: wire [1:0] c_temp;
- 暫存器的輸出值: reg ex: reg [3:0] d;
- 宣告資料寬度用中括弧 [MSB:LSB]
 - Ex: 3位元資料 bit2,bit1,bit0 → [2:0]

合法的識別字

- 命名: 開頭必須是英文字母或者底線(_), 數字是不被接受的。
- 關鍵字的組成由英文字母、數字及_組成。

correct

adder4
mux2
clk
_sub2

incorrect

2adder
mux@
Clk>
2sub2

註解與縮行

- 註解方式與C語言完全相同
 - 單行註解: //
 - 多行註解: /* 開始 */ 結束
- 縮行:程式碼必須要有漸層
 - Ex:

```
module adder (cin,a,b,s,cout);  
  input  a,b,cin;  
  output s, cout;  
  reg    s, cout;  
  
  always@(a,b,cin)  
  begin  
    {s,cout} <= cin+a+b;  
  end  
endmodule
```


常用數值表示法

- 數值可以使用以下進制表示，表示法為
<位元數>'<選用的進制><數值>。十進制33為例
 - 2進制(b) 6'b100001
 - 8進制(o) 6'o41
 - 10進制(d) 6'd33
 - 16進制(h) 6'h21

常用運算子

{ }	Concatenation		~	Bit-wise not
+, -, *, /	arithmetic		&	Bit-wise and
%	modulus			Bit-wise or
>, >=, <, <=, ==, !=	relational		^	Bit-wise XOR
!	Logical not		>>	Shift right
&&	Logical and		<<	Shift left
	Logical or			
? :	conditional			

常用運算子範例

Concatenation

```
wire a,b;  
wire [1:0]c;  
assign c = {a,b};
```

arithmetic

```
assign c = a+b;  
assign c = a-b;  
assign c = a*b;  
assign c = a/b;  
assign c = a%b;
```

Logical and bit-wise

```
a = 4'b1001;    b = 4'b0101;  
a || b = 1;     a | b = 4'b1101;  
a & b = 1;       a & b = 4'b0001;
```

conditional

```
assign f = (sel==1'b0)? a:b;  
//如果sel是0的話，f輸出a否則b
```

Shift

```
a = 4'b0001;  
assign b = a<<2; // b = 4'b0100  
//b的值為a左移2位元
```

事件控制

- 事件控制有三種類型

- 正緣觸發

```
always@(posedge clk)
begin
    Cout<=Cin;
end
/*當clk正緣時，
Cout 值等於Cin */
```

- 變動觸發

```
always@(clk)
begin
    Cout<=Cin;
end
/*當clk變動時，
Cout 值等於Cin */
```

- 負緣觸發

```
always@(negedge clk)
begin
    Cout<=Cin;
end
/*當clk負緣時，
Cout 值等於Cin */
```

Verilog常用的語法範例-1

- 4 to 1 8位元多工器-使用case

```
module mux4_1 (a1,a2,a3,a4,sel,s);
  input  [7:0] a1,a2,a3,a4;
  input  [1:0] sel;
  output [7:0] s;
  reg    [7:0] s; //因為s是在always@()中的輸出，必須宣告成reg

  always@(a1,a2,a3,a4,sel) //當a1,a2,a3,a4,sel有變動時，執行裡面程式
  begin // 當內容不只一行時，需要有begin
    case(sel) //case的()中寫判斷的變數
      2'b00: //因為下面只有s<=a1一行
        s<=a1; //所以不需要begin
      2'b01:
        s<=a2;
      2'b10:
        s<=a3;
      default: //case 的結尾一定要寫default
        s<=a4;
    endcase
  end // begin 要搭配 end
endmodule
```

Verilog常用的語法範例-2

- 4 to 1 8位元多工器 – 不使用always

```
module mux4_1 (a1,a2,a3,a4,sel,s);  
  input  [7:0] a1,a2,a3,a4;  
  input  [1:0] sel;  
  output [7:0] s;  
  
  assign s = (sel[1]==1'b0)? (sel[0]==1'b0)? a1:a2    //當sel[1]為0時  
              : (sel[0]==1'b0)? a3:a4;    //當sel[1]為1時  
  
endmodule
```

小結：

1. 只有在**always@()**中的程式片段才可以使用**case, if** 語法
2. **always@()**中的程式碼:輸出<=輸入. **ex : s<=a1**
3. 不在**always@()**中的程式碼，需要用**assign** 來指派值 且必須用 **=** 不可用**<=**
ex : assign s = a + b;

Verilog常用的語法範例-3

- 有reset功能的8位元計數器

```
module counter (clk,reset,count);
  input  clk,reset;
  input  [1:0] sel;
  output [7:0] count;
  reg     [7:0] count;

  always@(posedge clk)  //當clk正緣時判斷有無reset，無reset時count值加1
  begin
    if(reset==1'b1)
      count<= 8'h00;
    else
      count<= count + 1'b1;
    end
endmodule
```

註：

clk觸發的元件，通常只有DFF與計數器(亦是由DFF兜出)