

第 10 講 任務(task)

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 6 章，全華書局

2. 基本觀念[3]

2.1 任務

task <任務名稱>

<任務參數宣告>

.....
<敘述區>

endtask

例題 1: 4 bit adder with task

The screenshot shows the edaplayground online Verilog simulator interface. It features two code editors: 'testbench.v' and 'design.v'. The 'testbench.v' editor contains a testbench for a 4-bit adder, while the 'design.v' editor contains the adder's implementation. Below the editors is a log window displaying simulation results.

```
testbench.v
1 // Code your testbench here
2 // or browse Examples
3 module testbench;
4
5 reg clk, reset, cin;
6 reg [3:0] A, B;
7 wire [3:0] sum;
8 wire cout;
9
10 // 將四位元加法器模組實例化
11 four_bit_adder_with_carry dut (
12 .clk(clk),
13 .reset(reset),
14 .cin(cin),
15 .A(A),
16 .B(B),
17 .sum(sum),
18 .cout(cout)
19 );
20
21 endmodule
```

```
design.v
1 // Code your design here
2 module four_bit_adder_with_carry;
3   input wire clk;
4   input wire reset;
5   input wire cin;
6   input [3:0] A;
7   input [3:0] B;
8   output reg [3:0] sum;
9   output reg cout;
10
11
12 // 使用 task 實現加法邏輯
13 always @(posedge clk or posedge reset) begin
14   if (reset) begin
15     sum <- 4'b0000;
16     cout <- 1'b0;
17   end else begin
18     adder_task(A, B, cin, sum, cout);
19   end
20 endtask
```

```
(2024-04-14 13:06:05 UTC)iverilog -Wall design.v testbench.v && unbuffer vvp a.out
0, clk=0, reset=0, Cin=0, Cout=x, A=0000, B=0000, sum = xxxx
5, clk=1, reset=1, Cin=0, Cout=0, A=0000, B=0000, sum = 0000
10, clk=0, reset=1, Cin=0, Cout=0, A=0000, B=0000, sum = 0000
15, clk=1, reset=0, Cin=1, Cout=0, A=0010, B=0101, sum = 1000
20, clk=0, reset=0, Cin=1, Cout=0, A=0010, B=0101, sum = 1000
25, clk=1, reset=0, Cin=0, Cout=1, A=1100, B=1010, sum = 0110
30, clk=0, reset=0, Cin=0, Cout=1, A=1100, B=1010, sum = 0110
35, clk=1, reset=0, Cin=1, Cout=1, A=1111, B=0001, sum = 0001
40, clk=0, reset=0, Cin=1, Cout=1, A=1111, B=0001, sum = 0001
45, clk=1, reset=0, Cin=1, Cout=1, A=1111, B=0001, sum = 0001
```

- design.sv:

```
module four_bit_adder_with_carry(
    input wire clk,
    input wire reset,
    input wire cin,
    input [3:0] A,
    input [3:0] B,
    output reg [3:0] sum,
    output reg cout
);
```

```
// 使用 task 進行加法運算
always @(posedge clk or posedge reset) begin
    if (reset) begin
        sum <= 4'b0000;
        cout <= 1'b0;
    end else begin
        adder_task(A, B, cin, sum, cout);
    end
end
```

```
// 定義加法器 task
task adder_task;
    input [3:0] a, b;
    input cin;
    output [3:0] s;
    output co;

    assign {co, s} = a + b + cin;

endtask
```

```
endmodule
```

- testbench.sv:

```

module testbench;

reg clk, reset, cin;
reg [3:0] A, B;
wire [3:0] sum;
wire cout;

// 將四位元加法器模組實例化
four_bit_adder_with_carry dut (
    .clk(clk),
    .reset(reset),
    .cin(cin),
    .A(A),
    .B(B),
    .sum(sum),
    .cout(cout)
);

// 產生時脈信號
always begin
    #5 clk = ~clk;
end

initial
$monitor($time,", clk=%b, reset=%b, Cin=%b, Cout=%b, A=%b, B=%b, sum
= %b", clk, reset, cin, cout, A, B, sum);

// 初始化
initial begin
    clk=0;
    cin=0;
    reset=0;
    A=4'b0000;
    B=4'b0000;

// 進行測試
#5 reset=1;

```

```

#10 A=4'b0010; B=4'b0101; reset=0; cin=1; // A=2, B=5
#10 A=4'b1100; B=4'b1010; reset=0; cin=0; // A=12, B=10
#10 A=4'b1111; B=4'b0001; reset=0; cin=1; // A=15, B=1

// 結束模擬
#10 $finish;
end

endmodule

```

- 執行結果:

```

[2024-04-14 13:06:05 UTC] iverilog '-Wall' design.sv testbench.sv && unbuffer
vvp a.out

      0, clk=0, reset=0, cin=0, cout=x, A=0000, B=0000, sum = xxxx
      5, clk=1, reset=1, cin=0, cout=0, A=0000, B=0000, sum = 0000
     10, clk=0, reset=1, cin=0, cout=0, A=0000, B=0000, sum = 0000
     15, clk=1, reset=0, cin=1, cout=0, A=0010, B=0101, sum = 1000
     20, clk=0, reset=0, cin=1, cout=0, A=0010, B=0101, sum = 1000
     25, clk=1, reset=0, cin=0, cout=1, A=1100, B=1010, sum = 0110
     30, clk=0, reset=0, cin=0, cout=1, A=1100, B=1010, sum = 0110
     35, clk=1, reset=0, cin=1, cout=1, A=1111, B=0001, sum = 0001
     40, clk=0, reset=0, cin=1, cout=1, A=1111, B=0001, sum = 0001
     45, clk=1, reset=0, cin=1, cout=1, A=1111, B=0001, sum = 0001

```

Done

例題 2: function 與 task 比較

```

testbench.sv
SV/Verilog Testbench
1 // Code your testbench here
2 // or browse Examples
3 module test2;
4
5 reg [3:0] A, B;
6 reg [3:0] result;
7
8 // 定義 task
9 task automatic adder_task;
10    input [3:0] a, b;
11    output [3:0] s;
12
13 begin
14    s = a + b;
15 end
16 endtask
17
18 // 在 Initial 區塊中使用 task
19 initial begin
20    A = 4'b0010;
21    B = 4'b0110;
22
23    // 呼叫 task 並將結果賦值給 result
24    adder_task(A, B, result);
25
26    // 展示結果
27    $display("Result from task: %b", result);
28
29 end
30
31 endmodule

design.sv
SV/Verilog
1 // Code your design here
2 module test;
3
4 reg [3:0] A, B;
5 reg [3:0] result;
6
7 // 定義 function
8 function [3:0] adder_function;
9    input [3:0] a, b;
10   begin
11      adder_function = a + b;
12   end
13 endfunction
14
15 // 在 initial 區塊中使用 function
16 initial begin
17    A = 4'b0010;
18    B = 4'b0110;
19
20    // 呼叫 function 並將結果賦值給 result
21    result = adder_function(A, B);
22
23    // 展示結果
24    $display("Result from function: %b", result);
25
26 end
27
28 endmodule

```

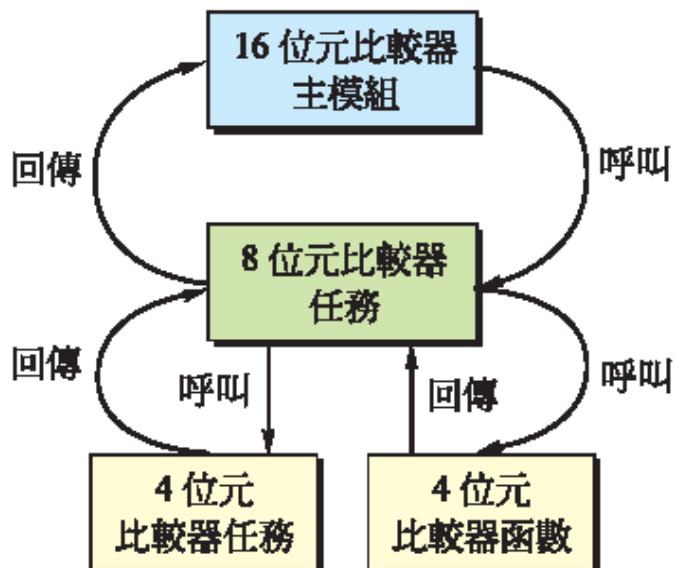
eLog Share

[2024-04-14 11:32:48 UTC] iverilog '-Wall' design.sv testbench.sv && unbuffer vvp a.out

Result from function: 1000
Result from task: 1000

Done

3. 作業 10-1: 16 位元比較器



```

module comp16_t_tf(A, B, Yout);
    input [15:0] A, B;
    output [2:0] Yout;

    reg [7:0] A8bit, B8bit;
    reg [2:0] Yout;

    always @ (A or B)
        begin
            A8bit = A[15:8];
            B8bit = B[15:8];
            compare8(A8bit, B8bit, Yout);
            if(Yout == 3'b010)
                begin
                    A8bit = A[7:0];
                    B8bit = B[7:0];
                    compare8(A8bit, B8bit, Yout);
                end
        end

    task compare8; //Task for a 8-bit comparator
        input [7:0] I_A, I_B;
        output [2:0] S;
        //S[2]=1--> Great than, S[1]=1-->Equal, S[0]=1-->Less
        than
        reg [3:0] A4bit, B4bit;
        begin
            A4bit = I_A[7:4];
            B4bit = I_B[7:4];
            compare4_T(A4bit, B4bit, S);
            if(S==3'b010) //I_A[7:4] = I_B[7:4]
                begin
                    A4bit = I_A[3:0];
                    B4bit = I_B[3:0];
                    S = compare4_F(A4bit, B4bit);
                end
        end
    endtask

```

```

task compare4_T; //Task for a 4-bit comparator
    input [3:0] I_A, I_B;
    output [2:0] S;
    //S[2]=1--> Great than, S[1]=1-->Equal, S[0]=1-->Less
than
begin
    if(I_A > I_B)
        S=3'b100;
    else if (I_A == I_B)
        S=3'b010;
    else
        S=3'b001;
end
endtask

function [2:0] compare4_F;
    input [3:0] I_A, I_B;
begin
    if(I_A > I_B)
        compare4_F = 3'b100;
    else if(I_A == I_B)
        compare4_F = 3'b010;
    else
        compare4_F = 3'b001;
end
endfunction

endmodule

```

(1) 請設計 **testbench** 程式，執行後將結果貼到作業報告中。