**Verilog第 4 次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 实现不同位加法器 | | | | |
| 学生姓名 | 邢清画 | 学号 | 2211999 | 指导老师 | 董前琨 |
| 实验地点 | 津南实验楼A区308 | | 实验时间 | 2023年11月22日 | |

1. **实验项目名称**

设计不同加法器

1. **实验目的**

本实验旨在通过构建从基础的1位、16位加法器到复杂的32位加法器的层次结构，理解数字电路中加法操作的基本概念、模块化设计的原则以及进位传递的工作机制，并深入探索超前进位技术以提高电路性能。此外，通过引入进位选择加法器，实验进一步展示了超前进位技术如何优化计算效率，提升对电路设计性能提升方法的认识和理解。

1. **必修或选修**

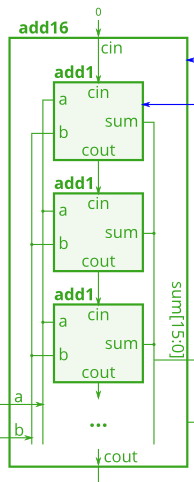
选修

1. **实验平台**

Vivado

1. **实验内容及步骤**

**实验内容：**

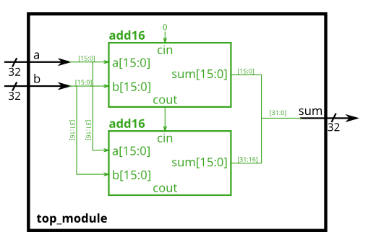
**练习题：**

1、实现一个1位加法器，可用门电路，也可用+运算符。并进行仿真测试。

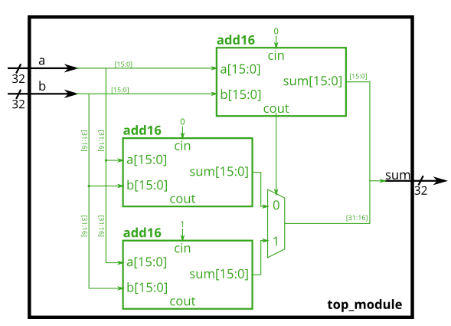
module add1 ( input a, input b, input cin, output sum, output cout );

2、在上述模块基础上，实现一个16位加法器，要求16位加法器为顶层模块，实现逻辑图如右图上。并进行仿真测试。

module add16 ( input[15:0] **a**, input[15:0] **b**, input **cin**, output[15:0] **sum**, output **cout** );

3、在上述模块基础上，实现一个32位加法器，要求32位加法器为顶层模块，实现逻辑如右图下。并进行仿真测试。

module add32( input [31:0] a, input [31:0] b, output [31:0] sum );

**作业题：**

在练习题2的基础上，完成一个特殊的加法器（进位选择加法器），该加法器实现的功能逻辑如右图。并进行仿真测试。

module cseladd32( input [31:0] a, input [31:0] b, output [31:0] sum );

此加法器已经初步具有了提前计算进位的思想，请调研超前进位思想，并在作业中总结应该如何设计一个更好的超前进位加法器。

Carry Select Adder

**实验说明：**

1、三个练习题需要每个单独建工程。需要代码复用时可以去代码里面的文本文件中把代码复制出来，避免多个工程使用同一个文件的情况。

2、写好子模块之后，另外新建一个文件作为上层模块，注意都在Disgn Source目录下，然后上层模块直接调用子模块即可，保存时会出现模块关系树。

3、三个练习题都需要独立编写testbench仿真文件，1位加法器可以一定的延迟后将模拟的输入信号取反，16位加法器和32位加法器模拟输入信号请使用$random方法。

4、在父模块调用子模块过程中，有些信号是需要用临时变量传递的，注意这些变量的类型和名称顺序，避免发生错乱。

5、每次写完一个模块，注意都需要运行综合和实现验证是否有错，避免文件和模块过多时不容易定位到错误。

**实现过程：**

**练习题：**

1. **1位加法器模块（add1）**

module add1 (input a,input b,input cin,output sum,output cout

);

assign {cout, sum} = a + b + cin;

endmodule

module tb\_add1();

reg a, b, cin;

wire sum, cout;

// 实例化1位加法器

add1 uut ( .a(a), .b(b), .cin(cin), .sum(sum), .cout(cout));

initial begin

// 初始化输入

a = 0; b = 0; cin = 0;

// 生成输入信号的序列

#10 a = 0; b = 0; cin = 1;

#10 a = 0; b = 1; cin = 0;

#10 a = 0; b = 1; cin = 1;

#10 a = 1; b = 0; cin = 0;

#10 a = 1; b = 0; cin = 1;

#10 a = 1; b = 1; cin = 0;

#10 a = 1; b = 1; cin = 1;

#10;

end

endmodule

1. **16位加法器模块（add16），使用16个add1模块串联**

module add16 (input [15:0] a,input [15:0] b,input cin,output [15:0] sum,output cout);

wire [15:0] c; // 中间进位

// 实例化16个1位加法器

genvar i;

generate

for (i = 0; i < 16; i = i + 1) begin : adder\_loop

if (i == 0) begin

add1 u\_add1 (.a(a[i]), .b(b[i]), .cin(cin), .sum(sum[i]), .cout(c[i]));

end else begin

add1 u\_add1 (.a(a[i]), .b(b[i]), .cin(c[i-1]), .sum(sum[i]), .cout(c[i]));

end

end

endgenerate

assign cout = c[15]; // 最后一个进位为16位加法器的进位输出

endmodule

module tb\_add16();

reg [15:0] a, b;

reg cin;

wire [15:0] sum;

wire cout;

// 实例化16位加法器

add16 uut (.a(a), .b(b), .cin(cin), .sum(sum), .cout(cout));

initial begin

// 初始化输入

a = 16'h0000; b = 16'h0000; cin = 0;

// 生成随机输入信号

#10 a = $random; b = $random; cin = 0;

#10 a = $random; b = $random; cin = 1;

#10;

end

endmodule

1. **32位加法器模块（add32），使用两个add16模块**

module add32 (

input [31:0] a,

input [31:0] b,

output [31:0] sum

);

wire c; // 中间进位

// 实例化两个16位加法器

add16 lower\_add16 (.a(a[15:0]), .b(b[15:0]), .cin(0), .sum(sum[15:0]), .cout(c));

add16 upper\_add16 (.a(a[31:16]), .b(b[31:16]), .cin(c), .sum(sum[31:16]), .cout());

endmodule

module tb\_add32();

reg [31:0] a, b;

wire [31:0] sum;

// 实例化32位加法器

add32 uut (.a(a), .b(b), .sum(sum));

initial begin

// 初始化输入

a = 32'h00000000; b = 32'h00000000;

// 生成随机输入信号

#10 a = $random; b = $random;

#10 a = $random; b = $random;

#10;

end

endmodule

**作业题：**

**特殊的加法器（进位选择加法器）**

module cseladd32 ( input [31:0] a,input [31:0] b,output [31:0] sum);

wire [31:0] g; // 产生项

wire [31:0] p; // 传递项

wire [31:0] c; // 进位

assign g = a & b;// 计算产生项和传递项

assign p = a ^ b;// 逐位计算进位

assign c[0] = 0; // 第0位没有进位输入

genvar i;

generate

for (i = 1; i < 32; i = i + 1) begin

assign c[i] = g[i-1] | (p[i-1] & c[i-1]);

end

endgenerate

assign sum = p ^ c;// 计算求和

endmodule

module tb\_cseladd32();

reg [31:0] a, b;

wire [31:0] sum;

// 实例化进位选择加法器

cseladd32 uut (.a(a),.b(b),.sum(sum));

initial begin

// 初始化输入

a = 32'h00000000; b = 32'h00000000;

// 生成随机输入信号

#10 a = $random; b = $random;

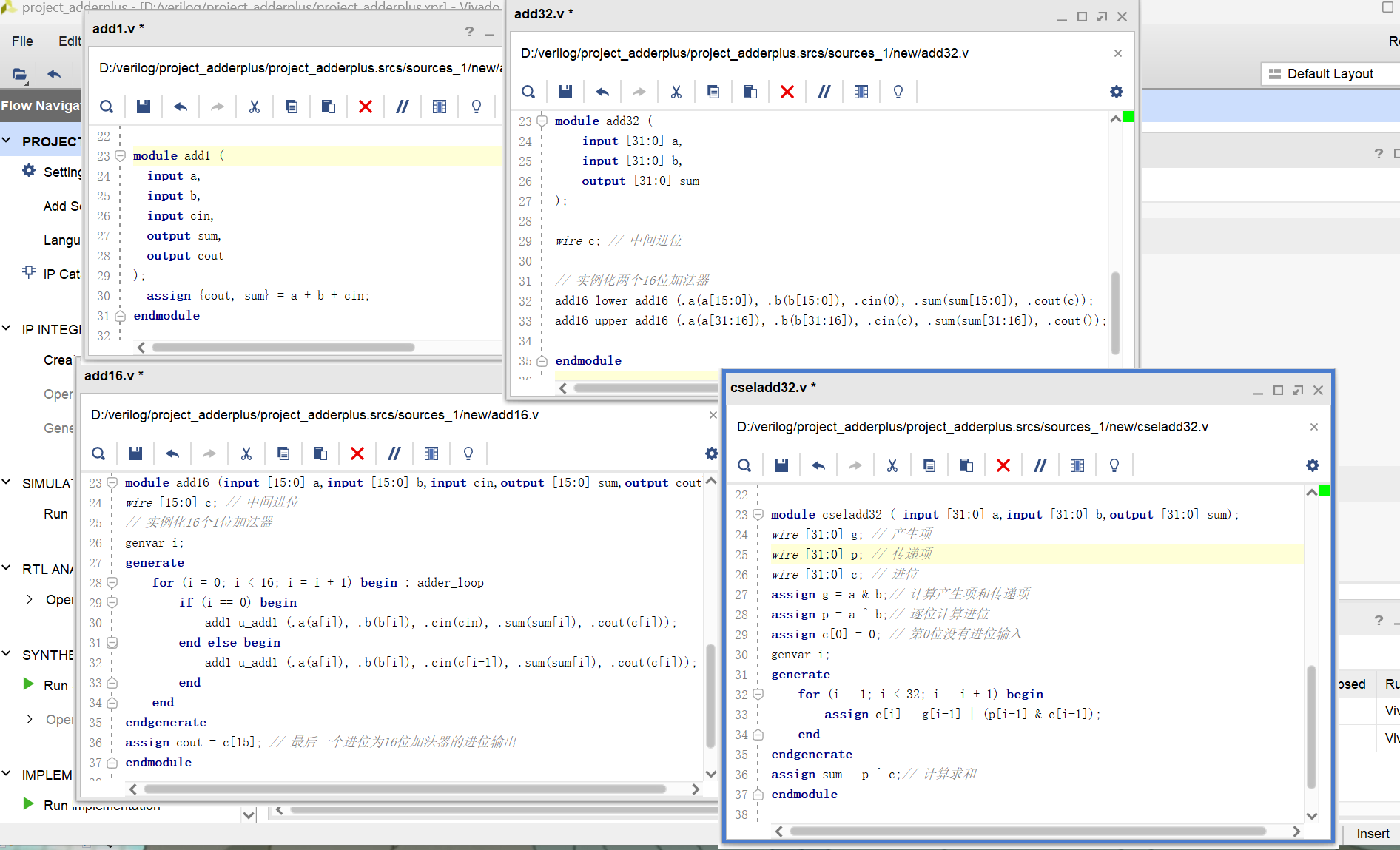
#10 a = $random; b = $random;

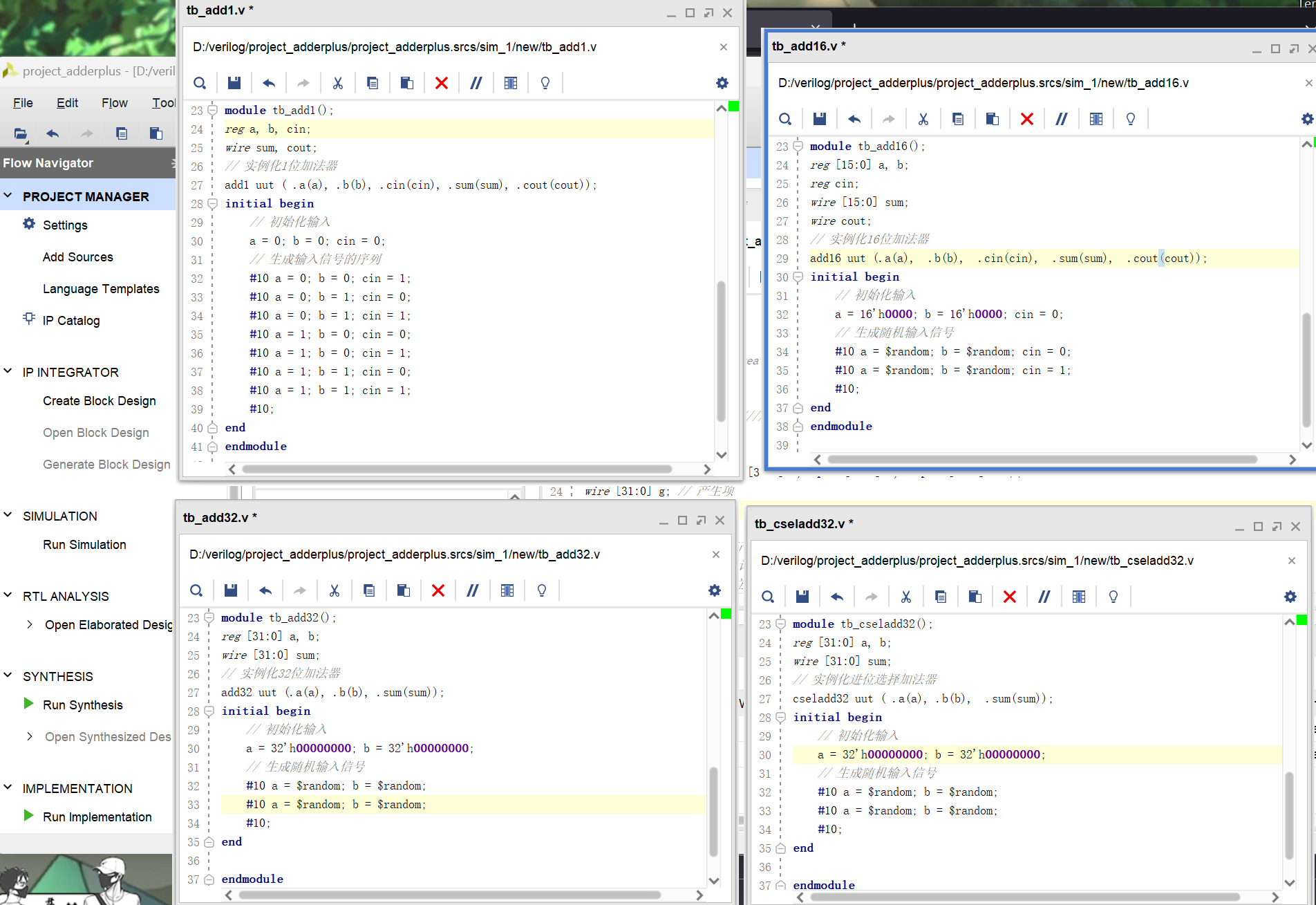
#10;

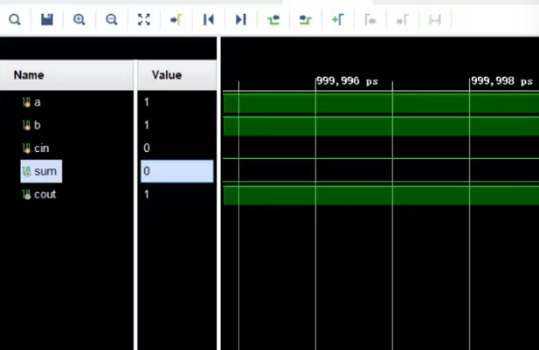
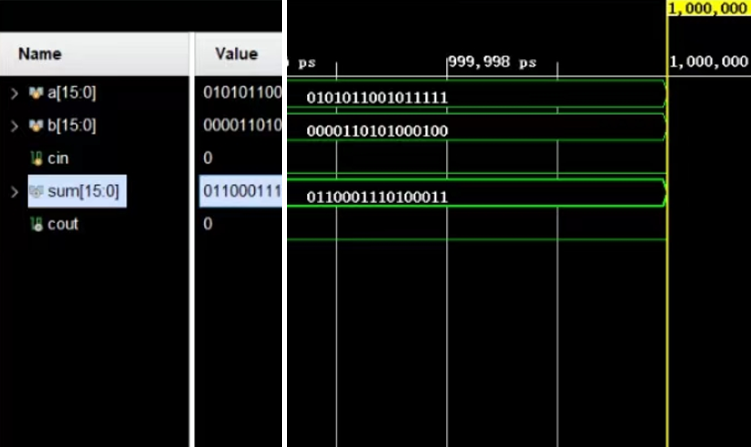
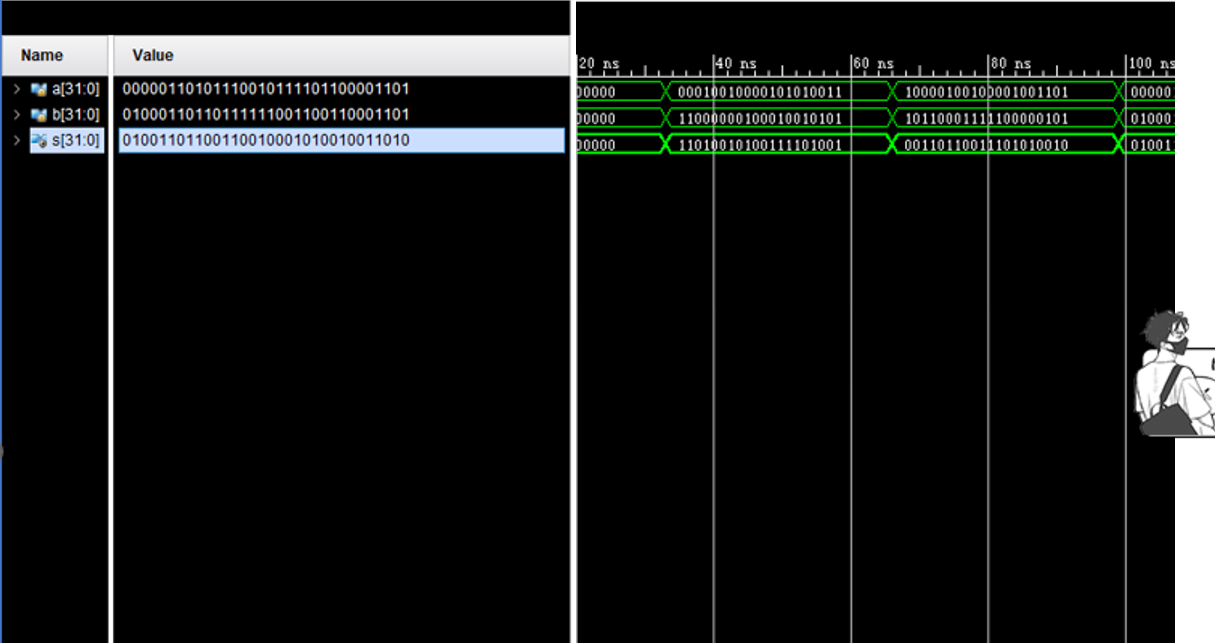
end

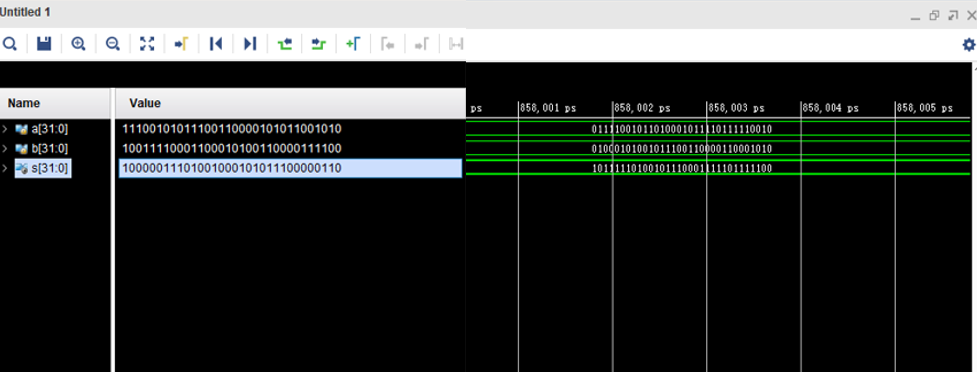
endmodule

1. **关键问题讨论**





**练习**

**作业**

如上波形图所示：

在十六位加法器中，a=0101011001011111，b=0000110101000100

a+b的理论值为0110001110100011

经检验，sum=0110001110100011，cout=0正确

1. **总结**

更加深入理解Verilog硬件描述语言在设计和验证数字电路中的应用。从实现基本的1位加法器开始，到构建更复杂的16位和32位加法器，再到探索具有提前计算进位功能的进位选择加法器，不仅练习了逐步抽象和模块化设计的方法，而且通过编写和运行测试平台，加强了对数字电路仿真测试的认识。此外，对编写可重用代码、进行综合和时序分析以及使用仿真工具来验证设计的正确性有了更深的了解。