1. **实验项目名称**

实现一个32位加法器

1. **实验目的**

理解数字加法器的工作原理，通过构建1位、16位和32位加法器，理解基本的数字逻辑和进位传递机制。使用子模块（例如，使用1位加法器构建16位和32位加法器），掌握如何将大型数字系统分解为可管理的小部分。学习如何使用Verilog或VHDL等硬件描述语言来设计和建模数字系统。

1. **必修或选修**

选修

1. **实验平台**

Vivado

1. **实验内容及步骤**

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，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，并进行仿真测试。

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

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

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

endmodule

module tb();

reg a,b,cin;

wire sum,cout;

add1 utt(a,b,cin,sum,cout);

initial begin

a=1'b0;

b=1'b0;

cin=1'b0;

end

always #1a=~a;

always #1b=~b;

endmodule

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

wire [14:0]c;//将a，b的每一位进行相加，前一位的cout=后一位的cin

add1 u0(a[0],b[0],cin,sum[0],c[0]);

add1 u1(a[1],b[1],c[0],sum[1],c[1]);

add1 u2(a[2],b[2],c[1],sum[2],c[2]);

add1 u3(a[3],b[3],c[2],sum[3],c[3]);

add1 u4(a[4],b[4],c[3],sum[4],c[4]);

add1 u5(a[5],b[5],c[4],sum[5],c[5]);

add1 u6(a[6],b[6],c[5],sum[6],c[6]);

add1 u7(a[7],b[7],c[6],sum[7],c[7]);

add1 u8(a[8],b[8],c[7],sum[8],c[8]);

add1 u9(a[9],b[9],c[8],sum[9],c[9]);

add1 u10(a[10],b[10],c[9],sum[10],c[10]);

add1 u11(a[11],b[11],c[10],sum[11],c[11]);

add1 u12(a[12],b[12],c[11],sum[12],c[12]);

add1 u13(a[13],b[13],c[12],sum[13],c[13]);

add1 u14(a[14],b[14],c[13],sum[14],c[14]);

add1 u15(a[15],b[15],c[14],sum[15],cout);

endmodule

module tb16( );

reg [15:0]a,b;

reg cin;

wire [15:0]sum;

wire cout;

add16 utt16(a,b,cin,sum,cout);//将a，b，cin的初始值代入函数得到sum和cout

initial begin

a=16'b0000\_0000\_0000\_0000;

b=16'b0000\_0000\_0000\_0000;

cin=1'b0;

end

//将a，b取随机十六位二进制数

always#17a=$random%17'b1\_0000\_0000\_0000\_0000;

always#17b=$random%17'b1\_0000\_0000\_0000\_0000;

endmodule

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

wire cout1;//将a，b的前十六位进行相加，得到前十六位的sum，再将a，b的后十六进行相加，其中后面输入的cin=前面输出的cout

add16 utt\_low(a[15:0],b[15:0],cin,sum[15:0],cout1);

add16 utt\_high(a[31:16],b[31:16],cout1,sum[31:16],cout);

endmodule

module tb32( );

reg [31:0]a,b;

reg cin;

wire [31:0]sum;

wire cout;

add32 utt32(a,b,cin,sum,cout);//将a，b，cin的初始值代入函数得到sum和cout

initial begin

a=32'b0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000;

b=32'b0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000;

cin=1'b0;

end

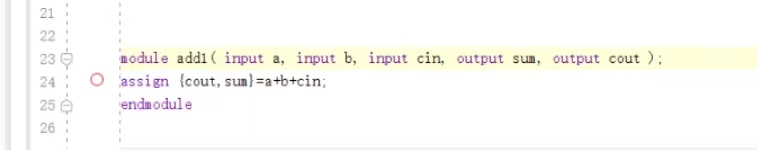
//将a，b取随机三十二位二进制数

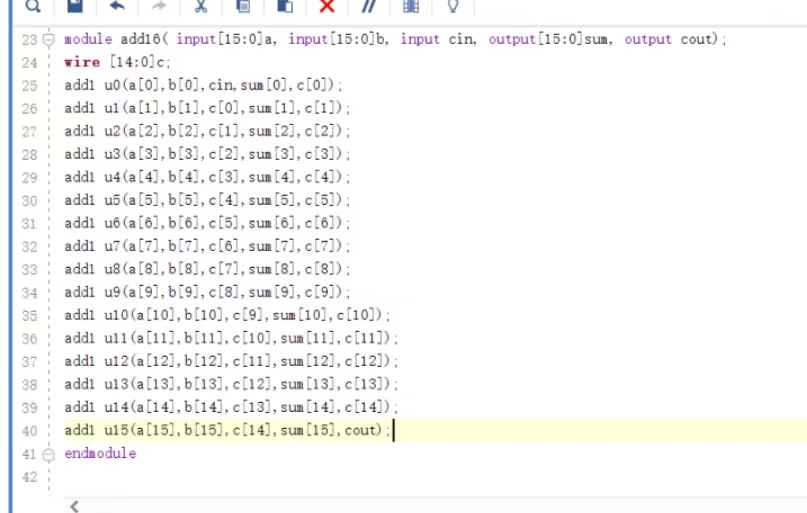
always#33a=$random%33'b1\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000;

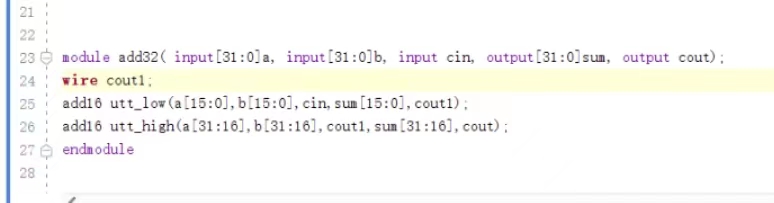
always#33b=$random%33'b1\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000;

endmodule

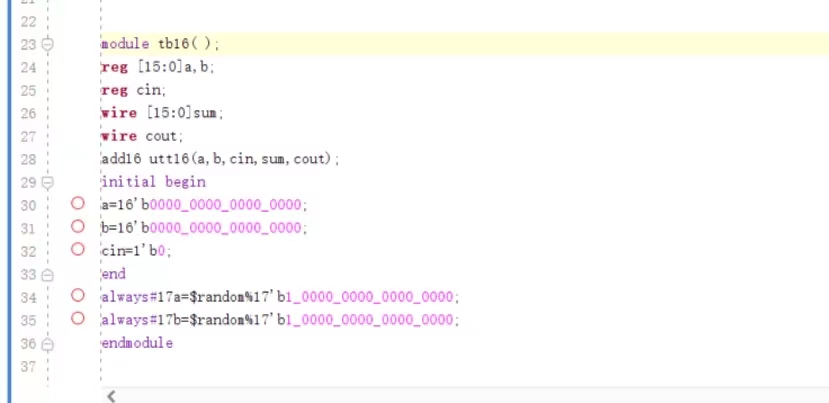
1. **关键问题讨论**

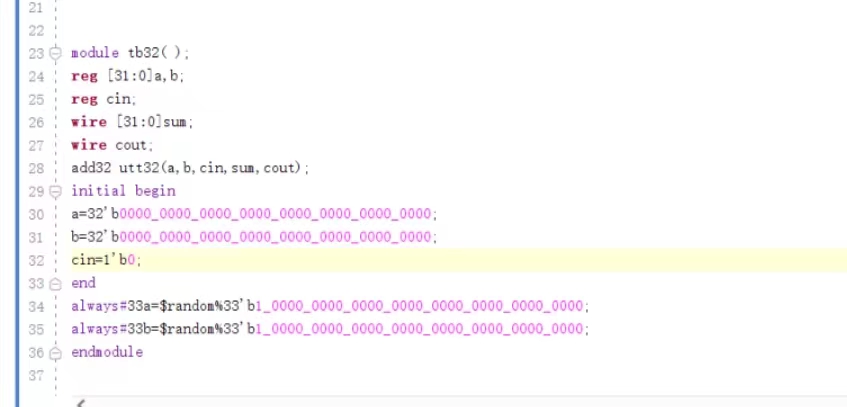
****

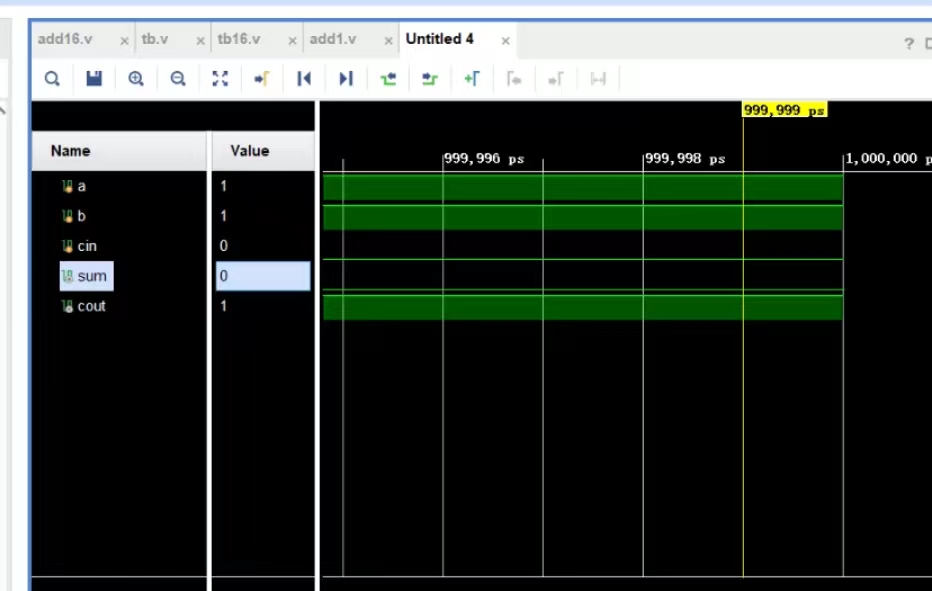
****

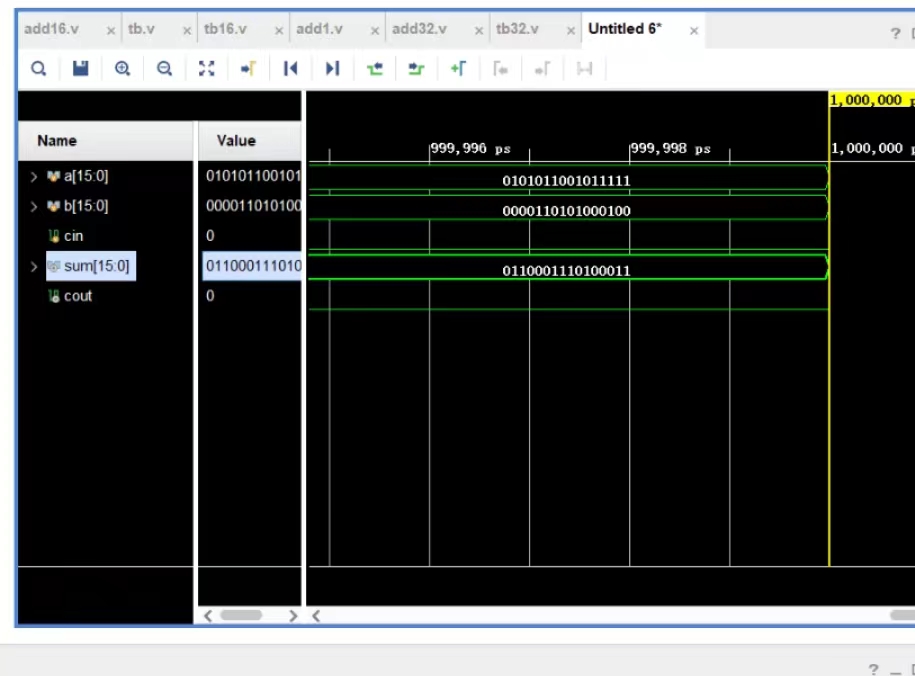
****

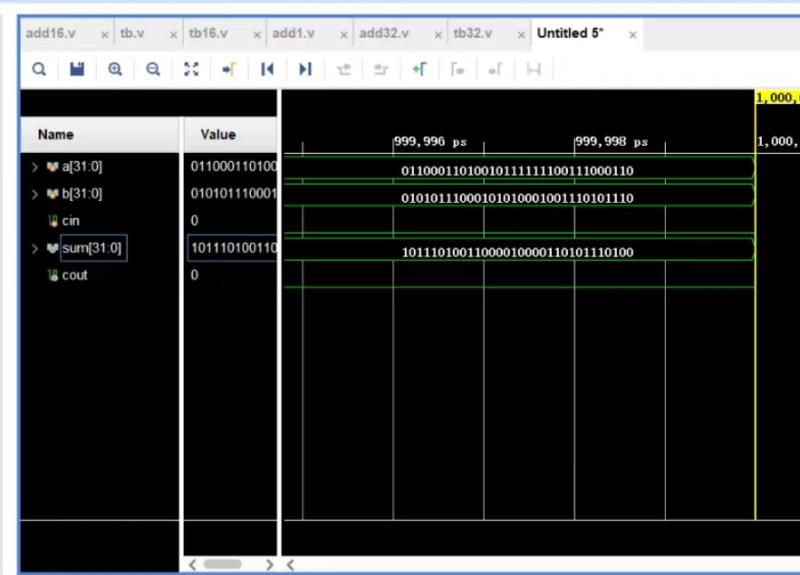
****

****

****

****

****

****

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是正确的

在三十二位加法器中，a=01100011010010111111100111000110，b=01010111000101010001001110101110,a+b的理论值为10111010011000010000110101110100，经检验，sun=10111010011000010000110101110100，cout=0是正确的

**七、总结**

通过本次实验，我对Verilog语言有了更深入的理解，特别是在定义模块、声明端口以及编写内部逻辑方面。通过亲自动手编写和仿真1位、16位和32位加法器，我掌握了数字电路中的基本运算方法，如加法、移位和比较，还了解到它们是如何在硬件描述语言中实现的。这些操作对于构建更复杂的数字系统至关重要。

在Verilog中实现基本的数字运算和逻辑操作的过程中，我加强了对逻辑运算的理解，这不仅提高了我的逻辑思维能力，而且增强了我的数字电路设计能力。实验中，我遇到并克服了多种挑战，例如模块间的信号连接问题，这些经历对我解决问题的能力有了显著提升。