**组成原理实验课程第一次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 加法器改进 | | | 班级 | 张金老师班 |
| 学生姓名 | 姚知言 | 学号 | 2211290 | 指导老师 | 董前琨 |
| 实验地点 | 实验楼A308 | | 实验时间 | 2024年3月22日 | |

1. **实验目的**

熟悉LS-CPU-EXB-002实验箱和软件平台。

掌握利用该实验箱各项功能开发组成原理和体系结构实验的方法。

理解并掌握加法器的原理和设计。

熟悉并运用verilog语言进行电路设计。

为后续设计cpu的实验打下基础。

1. **实验内容说明**

了解软硬件平台；

掌握定点加法的工作原理；

确定定点加法的输入输出端口设计；

画好顶层模块框图；

编写verilog代码；

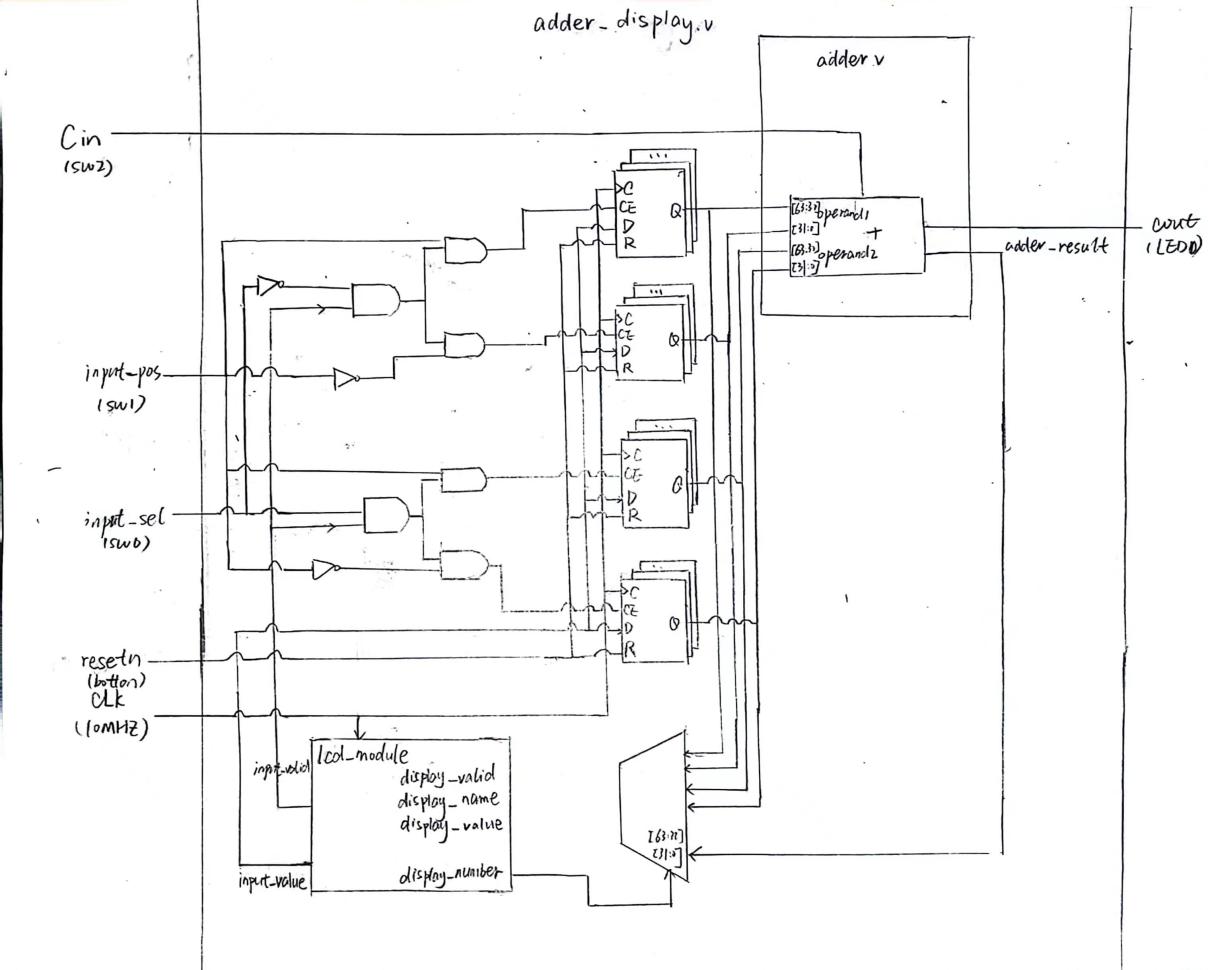
对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；

完成调用定点加法模块的外围模块的设计，并编写代码；

对代码进行综合布局布线下载到实验箱里FPGA板上，进行上板验证；

撰写实验报告。

1. **实验原理图**



输入端：如图所示，在resetn，input\_valid满足输入条件的情况下，通过控制input\_sel和input\_pos分别控制向哪一个加数输入，输入到高位还是低位。当input\_sel为0的时候，向加数1输入，否则向加数2输入。当input\_pos为0的时候，向低32位输入，否则向高32位输入。

输出端：1-6号屏幕分别输出加数1的高32位，低32位，加数2的高32位，低32位，和的高32位，低32位。

1. **实验步骤**

（1）adder.v

module adder(

input [63:0] operand1,

input [63:0] operand2,

input cin,

output [63:0] result,

output cout

);

assign {cout,result} = operand1 + operand2 + cin;

endmodule

//主要对位宽进行了调整。

（2）testbench.v

.......

reg [63:0] operand1;

reg [63:0] operand2;

reg cin;

wire [63:0] result;

wire cout;

//对位宽进行了调整

......

always #10 operand1[63:32] = $random;

always #10 operand2[63:32] = $random;

always #10 operand1[31:0] = $random;

always #10 operand2[31:0] = $random;

//由于random只能生成32位的结果，需要对前32位和后32位分别生成随机数

......

（3）adder\_display.v

......

input clk,

input resetn,

input input\_sel, //0:输入为加数1(add\_operand1);1:输入为加数2(add\_operand2)

input input\_pos,//0:输入为低位([31:0]);1:输入为高位([63:32])

input sw\_cin,

output led\_cout,

//增加了判断输入高低位的拨码开关

......

reg [63:0] operand1;

reg [63:0] operand2;

reg cin;

wire [63:0] result;

wire cout;

//调整了位宽

......

always @(posedge clk)

begin

if (!resetn)

begin

adder\_operand1 <= 64'd0;//调整了位宽

end

else if (input\_valid && !input\_sel)

begin

if(!input\_pos)//选择传值到高位还是低位

begin

adder\_operand1[31:0] <= input\_value;

end

else

begin

adder\_operand1[63:32] <= input\_value;

end

end

end

always @(posedge clk)

begin

if (!resetn)

begin

adder\_operand2 <= 64'd0;//调整了位宽

end

else if (input\_valid && input\_sel)

begin

if(!input\_pos)//选择传值到高位还是低位

begin

adder\_operand2[31:0] <= input\_value;

end

else

begin

adder\_operand2[63:32]<= input\_value;

end

end

end

always @(posedge clk)

begin

case(display\_number)

6'd1 :

begin

display\_valid <= 1'b1;

display\_name <= "ADD1H";

display\_value <= adder\_operand1[63:32];

end

6'd2 :

begin

display\_valid <= 1'b1;

display\_name <= "ADD1L";

display\_value <= adder\_operand1[31:0];

end

6'd3 :

begin

display\_valid <= 1'b1;

display\_name <= "ADD2H";

display\_value <= adder\_operand2[63:32];

end

6'd4 :

begin

display\_valid <= 1'b1;

display\_name <= "ADD2L";

display\_value <= adder\_operand2[31:0];

end

6'd5 :

begin

display\_valid <= 1'b1;

display\_name <= "RESUH";

display\_value <= adder\_result[63:32];

end

6'd6 :

begin

display\_valid <= 1'b1;

display\_name <= "RESUL";

display\_value <= adder\_result[31:0];

end //选通32位高低位输出和高低位操作数

default :

begin

display\_valid <= 1'b0;

display\_name <= 40'd0;

display\_value <= 64'd0;//这个当时改了，但是现在感觉反而应该保持32位，只是做实验的时候没发现也没有造成什么问题，就暂且保留在这里

end

endcase

end

endmodule

（4）mycons.xdc

set\_property PACKAGE\_PIN AC19 [get\_ports clk]

set\_property PACKAGE\_PIN H7 [get\_ports led\_cout]

set\_property PACKAGE\_PIN Y3 [get\_ports resetn]

set\_property PACKAGE\_PIN AC21 [get\_ports input\_sel]

set\_property PACKAGE\_PIN AD24 [get\_ports input\_pos]//为新增的input pos分配开关

set\_property PACKAGE\_PIN AC22 [get\_ports sw\_cin]//将cin的开关调整到第三个

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports led\_cout]

set\_property IOSTANDARD LVCMOS33 [get\_ports resetn]

set\_property IOSTANDARD LVCMOS33 [get\_ports input\_sel]

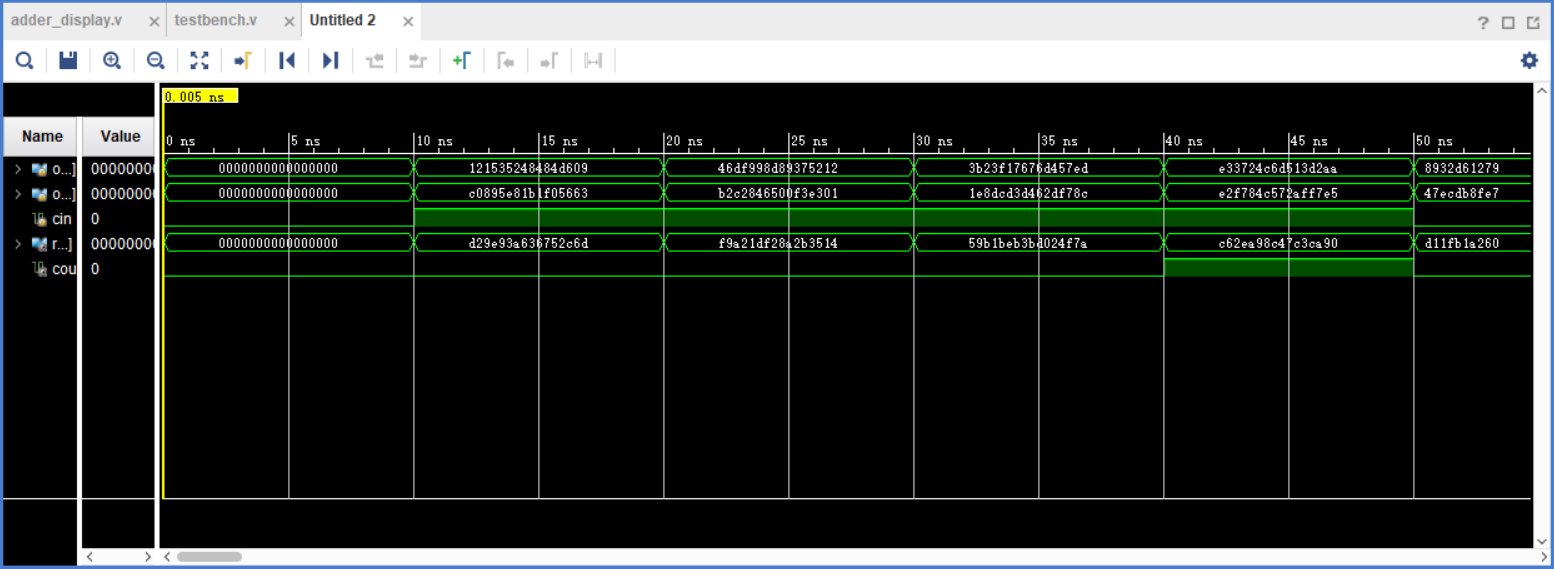
set\_property IOSTANDARD LVCMOS33 [get\_ports input\_pos]

set\_property IOSTANDARD LVCMOS33 [get\_ports sw\_cin]

......

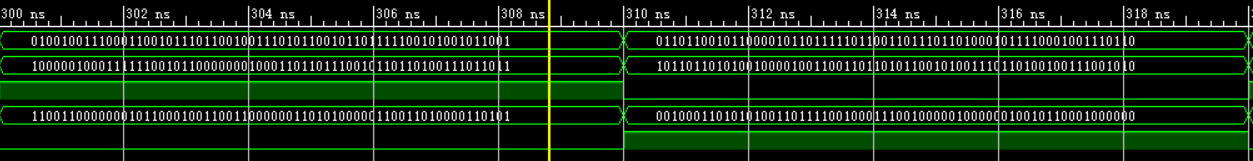
1. **实验结果分析**

（1）仿真结果



仿真结果总览。可以看到前两行为两个64位的操作数输入operand1和operand2，第三行为1位二进制数末位进位输入cin，第四行为一个64位操作数输出result，第五行位一个1位二进制向上进位输出cout。

可以看到，在前10ns中，五个操作数的结果都为0，符合初始设置。自此之后，每过10ns，前三行的输入进行一次随机数的生成，并同时计算出后两行的输出。总体上来看，结果基本符合预期结果。



切换为二进制形式查看，任取两例较为有代表性的模拟进行分析：首先，在300-310ns中operand1的结果为：

0100100111000110010111011001001110101100101101111100101001011001

，operand2的结果为：

1000001000111111001011000000010001101101110010110110100111011011

，cin为1。

从而，result结果为：

1100110000000101100010011001100000011010100000110011010000110101

，cout为0。

符合operand1+operand2+cin={cout，result}。

为验证cout的正常运行，另取310-320ns中结果，operand1的结果为：

0110110010110000101101111101100110111011010001011110001001110110

，operand2的结果为：

1011011010100100001001100110110101100101001110110100100111001010

，cin为0。

从而，result结果为：

0010001101010100110111100100011100100000100000010010110001000000

，cout为1。

同样符合operand1+operand2+cin={cout，result}。

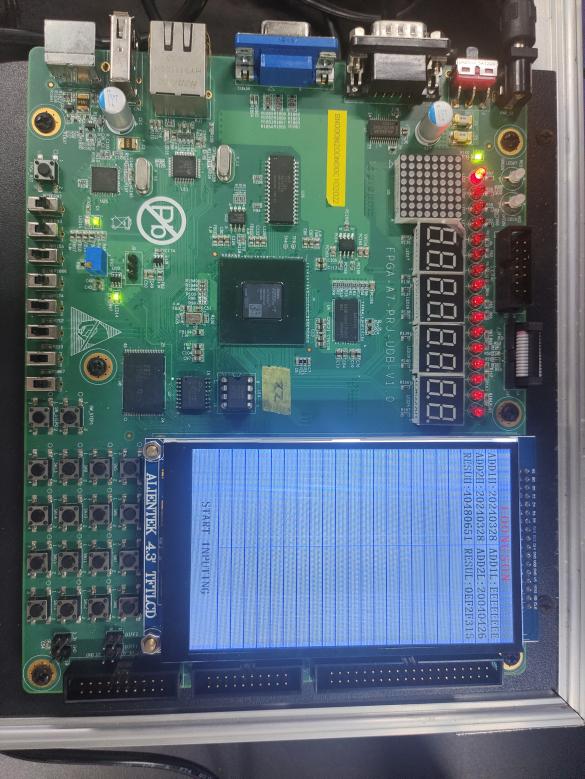
从而得出结论，即验证了正确性。

（2）实验箱运行结果

输入输出同上，operand1为ADDIH（高32位），ADD1L（低32位），operand2为ADD2H（高32位），ADD2L（低32位），result为RESUH（高32位），RESUL(低32位），cin为第三个开关，cout为第一个LED灯。

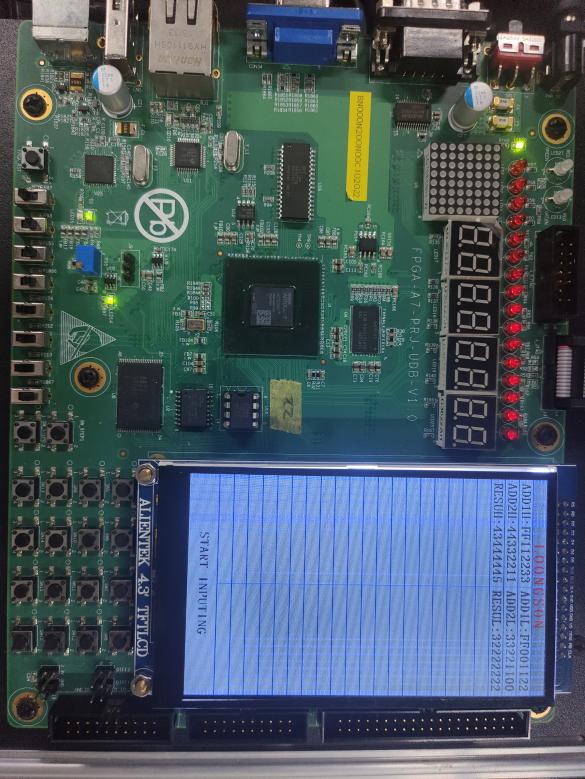
测试一：如下图所示，设置operand1的值为20240328EEEEEEEE，operand2的值为2024032820040426，cin为1。

此时显示输出结果为404806510EF2F315，cout为0，符合operand1+operand2+cin={cout，result}，验证了正确性。



测试二：如下图所示，设置operand1的值为FF112233FF001122，operand2的值为4433221133221100，cin为0。

此时显示输出结果为4344444532222222，cout为1，符合operand1+operand2+cin={cout，result},验证了正确性。



1. **总结感想**

在本次实验中，我初次接触计算机组成原理实验，熟悉了vivado的操作流程和文件结构，通过用模拟数据simulation仿真和在实验箱中实际操作，验证设计正确性的方法。

同时，第一次接触verilog语言，学习了很多verilog语言的架构和语法，并最终将其应用在加法器的改进实验中，通过多文件的修改最终达到将32位加法器改装为64位加法器的效果。

接触了实验箱的结构和约束文件的使用，对硬件体系结构有了更深的认识。

这次实验也为后续实验的操作流程和期末的最终实验打下了基础。