**组成原理实验课程第三次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 寄存器改进 | | | 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4月19日 | |

1. **实验目的**

熟悉并掌握 MIPS 计算机中寄存器堆的原理和设计方法。

初步了解 MIPS 指令结构和源操作数/目的操作数的概念。

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**

掌握寄存器堆的工作原理；

确定寄存器堆的输入输出端口设计；

画好寄存器堆的设计框图或实验原理图；

确认寄存器堆的设计框图的正确性；

编写 verilog 代码；

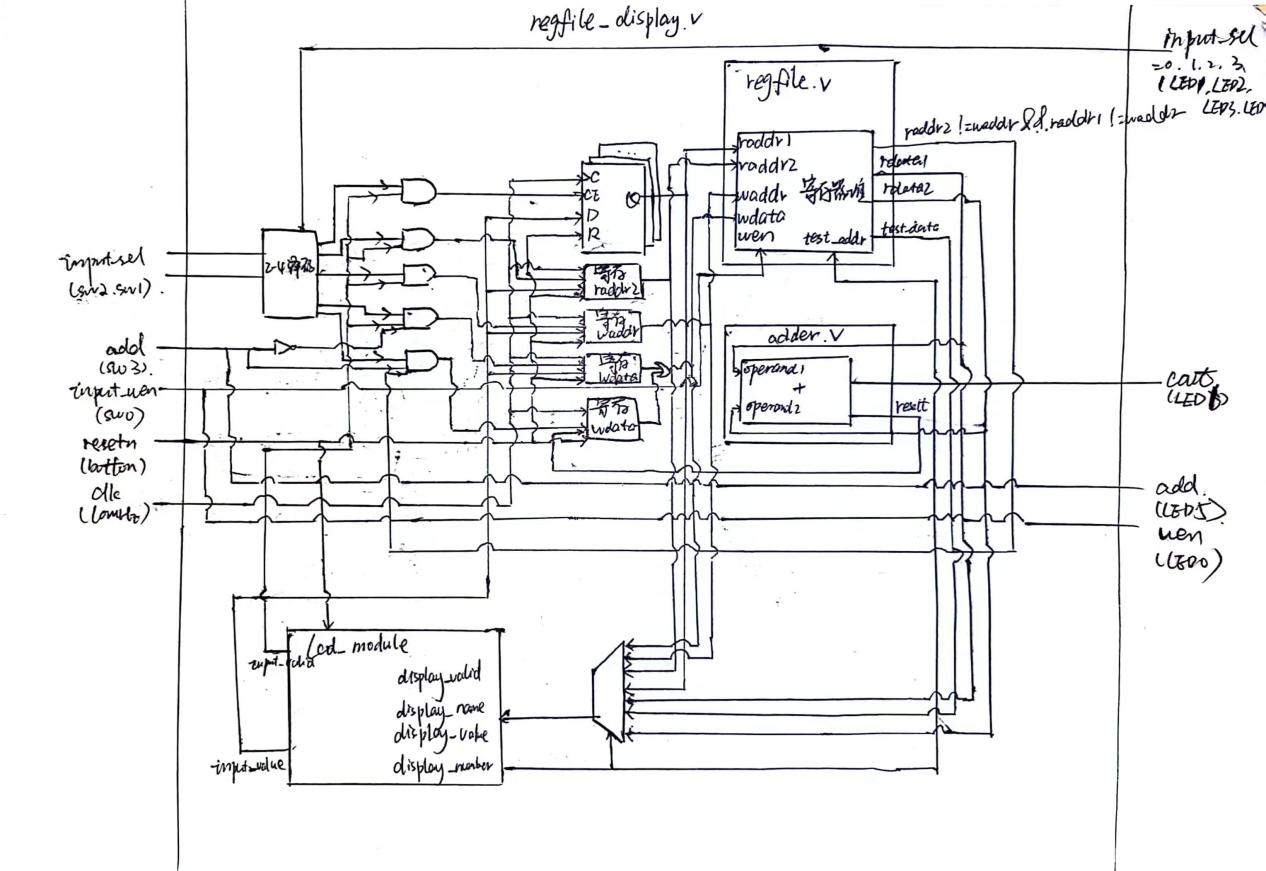
完成调用寄存器堆模块的外围模块的设计，并编写代码；

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；

完成功能改进，将寄存器堆和加法器结合，实现一个基础的寄存器数据求和操作；

实验结束后，需按照规定的格式完成实验报告的撰写。

1. **实验原理图**



输入端：第一个拨码开关wen，第二第三个input\_sel，第四个add

输出端：第一个led灯指示wen，第二到五个指示input\_sel，第六个指示add，第七个指示加法器进位cout

整体逻辑：

在wen=1将wdata赋值给waddr（若合法且不是0）指示的寄存器；

input\_sel=00，触摸板输入修改radd1的值；

input\_sel=01，触摸板输入修改radd2的值；

input\_sel=10，触摸板输入修改waddr的值；

input\_sel=11且add=0，触摸板输入修改wdata的值；

input\_sel=11且add=1且waddr!=radd1且waddr!=radd2，将加法器的结果result赋值给waddr；

rdat1显示radd1指向寄存器的值，rdat2显示radd2指向寄存器的值；

加法器两个加数分别是rdat1和rdat2，低位进位输入为0，高位进位输出用led灯标识。

1. **实验步骤**

（1）导入adder.v，后续使用，未作修改

module adder(

input [31:0] operand1,

input [31:0] operand2,

input cin,

output [31:0] result,

output cout

);

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

endmodule

（2）regfile\_display.v

module regfile\_display(

input clk,

input resetn,

input wen,

input [1:0] input\_sel,

input add, //增加一个选通加法器or写数据的拨码开关

output led\_wen,

output led\_waddr,

output led\_wdata,

output led\_raddr1,

output led\_raddr2,

output led\_add, //指示add是否为1

output led\_cout, //指示加法器是否产生进位

output lcd\_rst,

output lcd\_cs,

output lcd\_rs,

output lcd\_wr,

output lcd\_rd,

inout[15:0] lcd\_data\_io,

output lcd\_bl\_ctr,

inout ct\_int,

inout ct\_sda,

output ct\_scl,

output ct\_rstn

);

assign led\_wen = wen;

assign led\_raddr1 = (input\_sel==2'd0);

assign led\_raddr2 = (input\_sel==2'd1);

assign led\_waddr = (input\_sel==2'd2);

assign led\_wdata = (input\_sel==2'd3);

assign led\_add = add; //为add绑定led显示

wire [31:0] test\_data;

wire [4 :0] test\_addr;

reg [4 :0] raddr1;

reg [4 :0] raddr2;

reg [4 :0] waddr;

reg [31:0] wdata;

wire [31:0] rdata1;

wire [31:0] rdata2;

regfile rf\_module(

.clk (clk ),

.wen (wen ),

.raddr1(raddr1),

.raddr2(raddr2),

.waddr (waddr ),

.wdata (wdata ),

.rdata1(rdata1),

.rdata2(rdata2),

.test\_addr(test\_addr),

.test\_data(test\_data)

);

wire cout;

wire [31:0] result;

adder adder\_module(

.operand1(rdata1),

.operand2(rdata2),

.cin(1'b0),

.result(result),

.cout(cout)

);

assign led\_cout = cout;

//增加两个wire变量cout和result，调用加法器模块，将cout变量绑定LED灯

reg display\_valid;

reg [39:0] display\_name;

reg [31:0] display\_value;

wire [5 :0] display\_number;

wire input\_valid;

wire [31:0] input\_value;

lcd\_module lcd\_module(

.clk (clk ),

.resetn (resetn ),

.display\_valid (display\_valid ),

.display\_name (display\_name ),

.display\_value (display\_value ),

.display\_number (display\_number),

.input\_valid (input\_valid ),

.input\_value (input\_value ),

.lcd\_rst (lcd\_rst ),

.lcd\_cs (lcd\_cs ),

.lcd\_rs (lcd\_rs ),

.lcd\_wr (lcd\_wr ),

.lcd\_rd (lcd\_rd ),

.lcd\_data\_io (lcd\_data\_io ),

.lcd\_bl\_ctr (lcd\_bl\_ctr ),

.ct\_int (ct\_int ),

.ct\_sda (ct\_sda ),

.ct\_scl (ct\_scl ),

.ct\_rstn (ct\_rstn )

);

assign test\_addr = display\_number-5'd7;

always @(posedge clk)

begin

if (!resetn)

begin

raddr1 <= 5'd0;

end

else if (input\_valid && input\_sel==2'd0)

begin

raddr1 <= input\_value[4:0];

end

end

always @(posedge clk)

begin

if (!resetn)

begin

raddr2 <= 5'd0;

end

else if (input\_valid && input\_sel==2'd1)

begin

raddr2 <= input\_value[4:0];

end

end

always @(posedge clk)

begin

if (!resetn)

begin

waddr <= 5'd0;

end

else if (input\_valid && input\_sel==2'd2)

begin

waddr <= input\_value[4:0];

end

end

//当input\_sel为2'b11时，表示输入数为写数据，即wdata

//若add为1，为了避免反复累加的情况，当写地址不等于读地址时，将加法器结果写入wdata中

//若add为0，执行原本将触摸屏幕输入数据写入wdata的逻辑

always @(posedge clk)

begin

if (!resetn)

begin

wdata <= 32'd0;

end

else if (input\_sel==2'd3)

begin

if(add)

begin

if(waddr!=raddr1 && waddr!=raddr2)

begin

wdata <= result;

end

end

else if(input\_valid)

begin

wdata <= input\_value;

end

end

end

......//后续输出显示屏显示部分未进行更改

（4）mycons.xdc

//为新增的输入输出端绑定引脚

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

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

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

set\_property PACKAGE\_PIN D5 [get\_ports led\_waddr]

set\_property PACKAGE\_PIN A3 [get\_ports led\_wdata]

set\_property PACKAGE\_PIN A5 [get\_ports led\_raddr1]

set\_property PACKAGE\_PIN A4 [get\_ports led\_raddr2]

set\_property PACKAGE\_PIN F7 [get\_ports led\_add]

set\_property PACKAGE\_PIN G8 [get\_ports led\_cout]

set\_property PACKAGE\_PIN AC21 [get\_ports wen]

set\_property PACKAGE\_PIN AD24 [get\_ports input\_sel[1]]

set\_property PACKAGE\_PIN AC22 [get\_ports input\_sel[0]]

set\_property PACKAGE\_PIN AC23 [get\_ports add]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports resetn]

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

set\_property IOSTANDARD LVCMOS33 [get\_ports led\_raddr1]

set\_property IOSTANDARD LVCMOS33 [get\_ports led\_raddr2]

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

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

set\_property IOSTANDARD LVCMOS33 [get\_ports wen]

set\_property IOSTANDARD LVCMOS33 [get\_ports input\_sel[1]]

set\_property IOSTANDARD LVCMOS33 [get\_ports input\_sel[0]]

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

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

set\_property IOSTANDARD LVCMOS33 [get\_ports add]

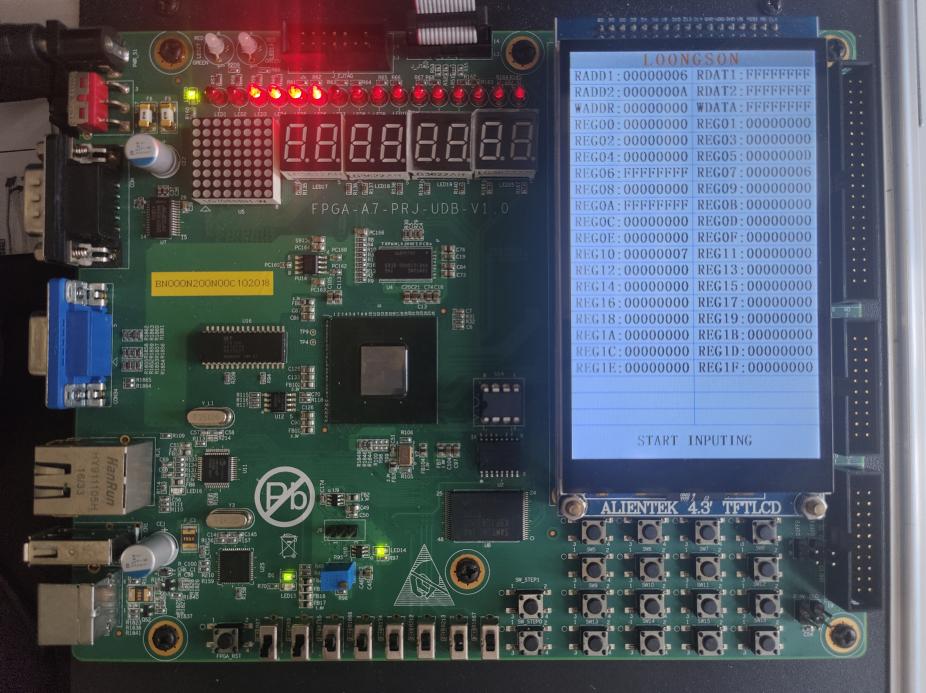
......//后续显示屏接口略去

1. **实验结果分析**

（1）整体寄存器功能验证与说明及cout LED灯检查

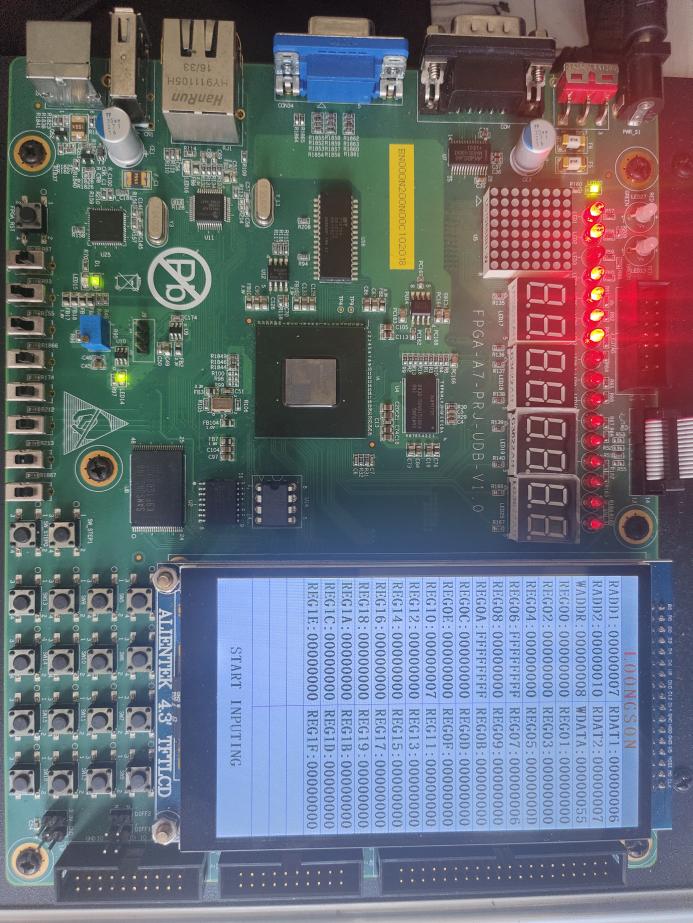
input\_sel=00，触摸屏修改radd1的数值；input\_sel=01，触摸屏修改radd2的数值；input\_sel=10，触摸屏修改waddr的数值；input\_sel=11，add=0触摸屏修改wdata的数值。

wen=1时，wdata会被写入waddr指向的寄存器中。但图中wen=1，reg00=0，验证了0号寄存器恒为0的性质。同时，因为0xFFFFFFFF与0xFFFFFFFF之和会产生进位，此时cout进位灯不亮（可与下一小节对比），说明加法器在正常工作。

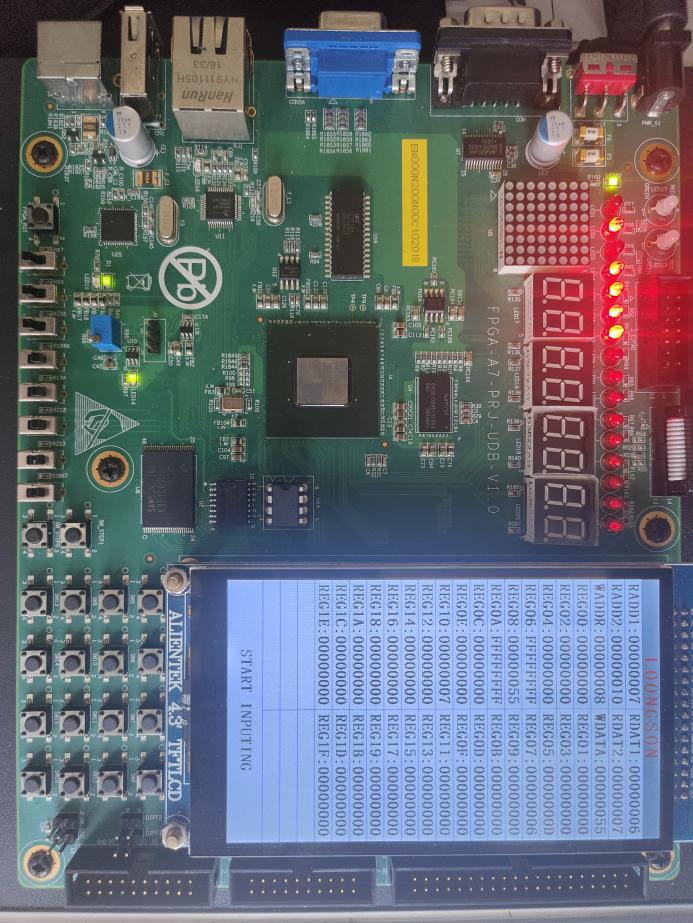


（2）add=0，input\_sel=11时正常赋值功能验证

如图，此时add=0，input\_sel=11，向触摸板传入数值0x55，可以看到wdata成功被赋值，由于wen=0，数值并没有被存入reg08。注意此时cout进位灯亮，没有产生进位，说明加法器在正常工作。

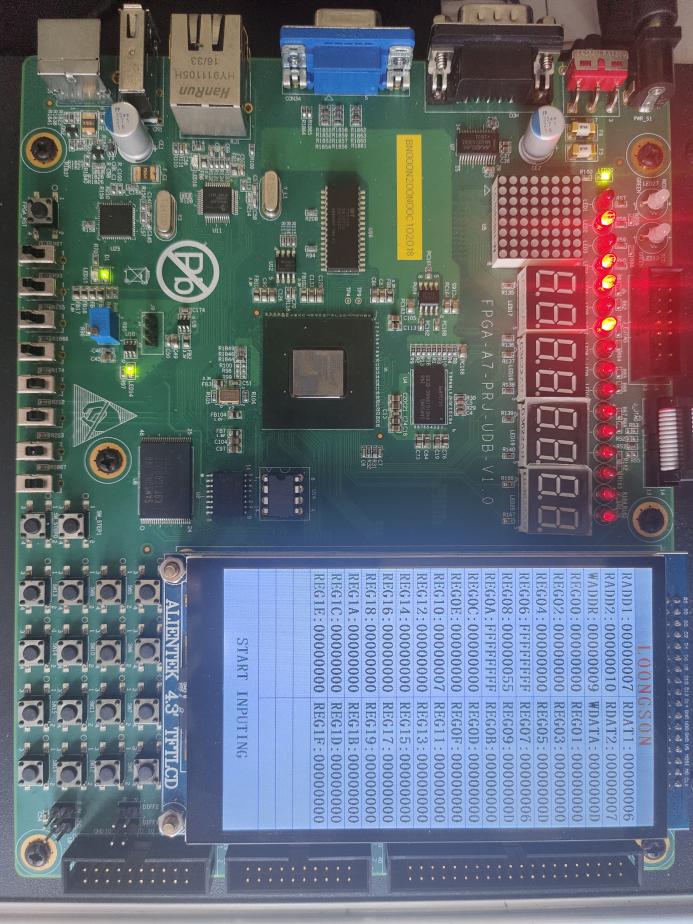


此时设置wen=1，完成对reg08的赋值。



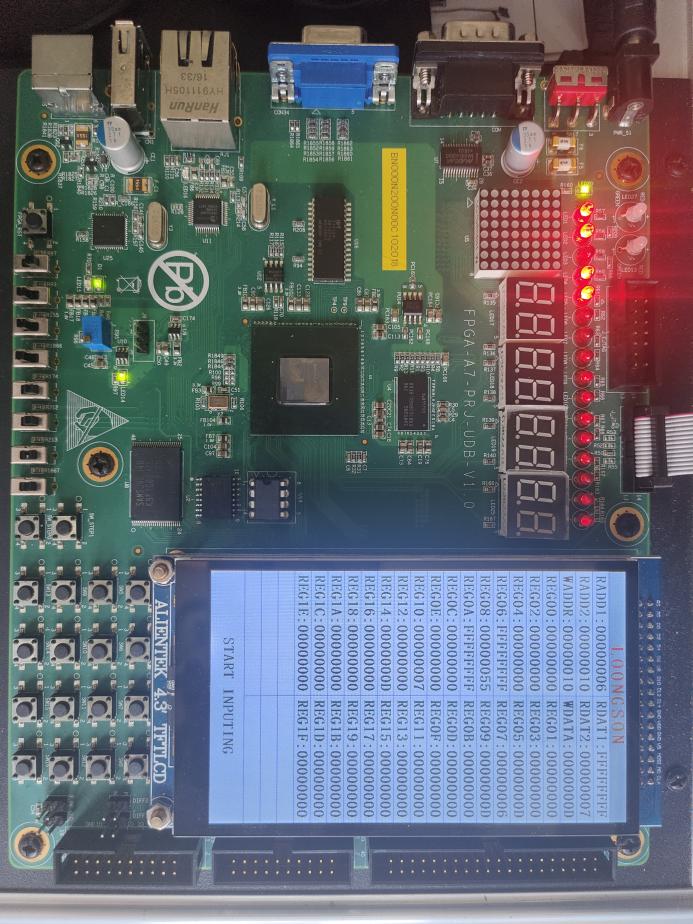
（3）add=1，input\_sel=11加法器功能验证

选择add=1，input\_sel=11，在一般情况下（三个寄存器各不相同），wdata被更新为rdat1+rdat2，如图，即0x6+0x7=0xD。由于wen=1，被赋值到reg09中。在此情况下再在触摸屏输入不再能更改wdata的数值。

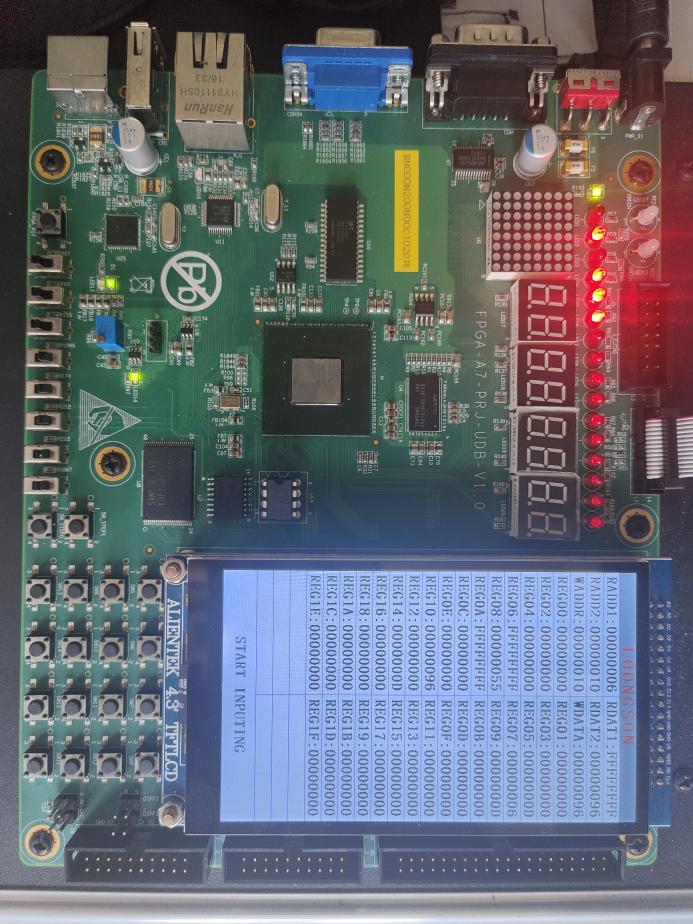


（4）特殊情况说明

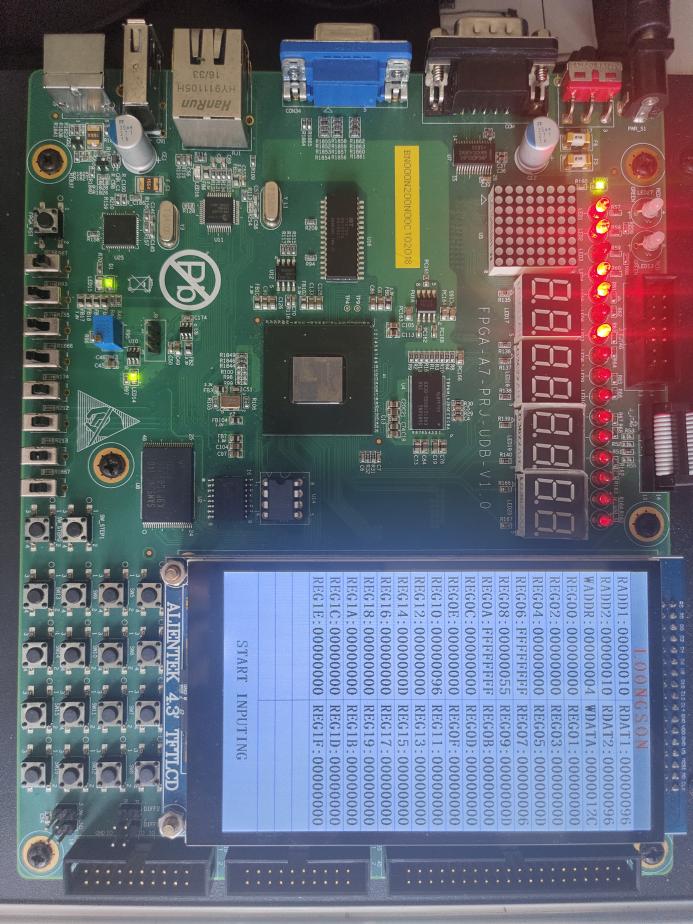
为防止反复累加赋值的情况，当waddr与任意raddr相等的时候，不再将加法器结果赋值给wdata。如图所示，radd2=waddr，input\_sel=11，add=1，wdata!=rdat1+rdat2，还是先前的结果。



该功能不会影响到add=0的时候wdata的正常赋值。



同样，在radd1=radd2的时候，也可以正常执行加法器。



1. **总结感想**

在本次实验中，我学习了寄存器的组成结构和形成形式，对verilog的语句也有了更深入的认识。通过本次修改regfile\_display.v模块导入adder.v模块的过程，认识了接口设置和模块调用的方式。

通过对实验箱的操作，更深刻的了解了寄存器的工作原理，与理论课所学知识得以结合。通过改装过程，将组合逻辑和时序逻辑结合起来，进一步了解了计算机的工作原理及方式。