实验三：存储器RAM

姓名：姜庆彩

学号：PB15051087

* 实验内容

1.学习如何使用ISE的IP核

2.学习使用Xilinx FPGA内的RAM资源

例化一个简单双端口的RAM（32bitx64）

使用coe文件对RAM进行初始化

* 实验功能要求

综合利用三次实验的结果，完成以下功能：

从ram中0地址和1地址读取两个数， 分别赋给reg0和reg1

利用第二次实验的结果(ALU+Regfile)进行斐波拉契运算，运算结果保存在对应的寄存器

运算结果同时保存在对应的ram地址中，即ram[0]<----->reg0, ram[1]<----->reg1,ram[2]<----->reg2,……

* 实验设计要求

实现一个control模块，完成整个运算的控制。

实现一个顶层模块Top

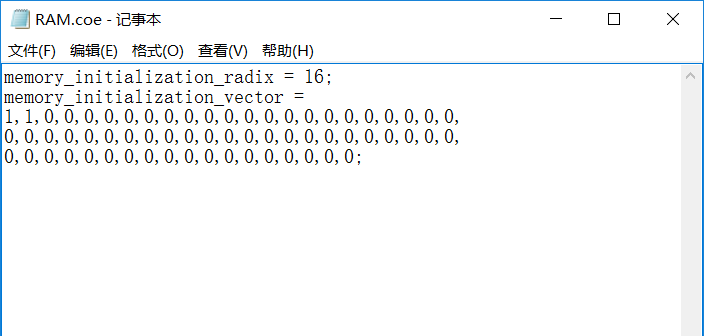
调用Ram模块

调用RegFile

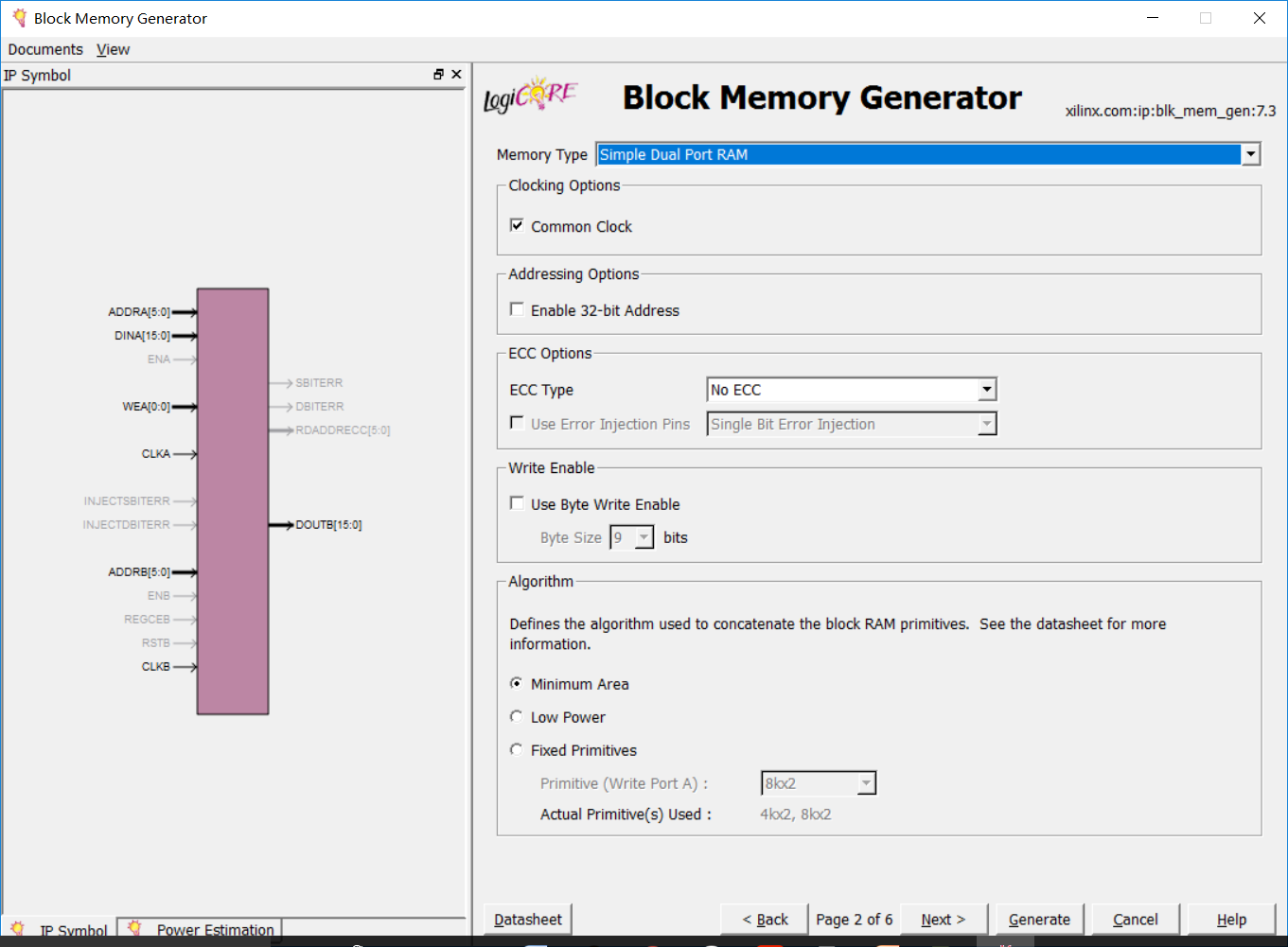
调用ALU完成加法运算

调用control模块，完成运算控制

* 我的实现过程



首先写一个coe文件



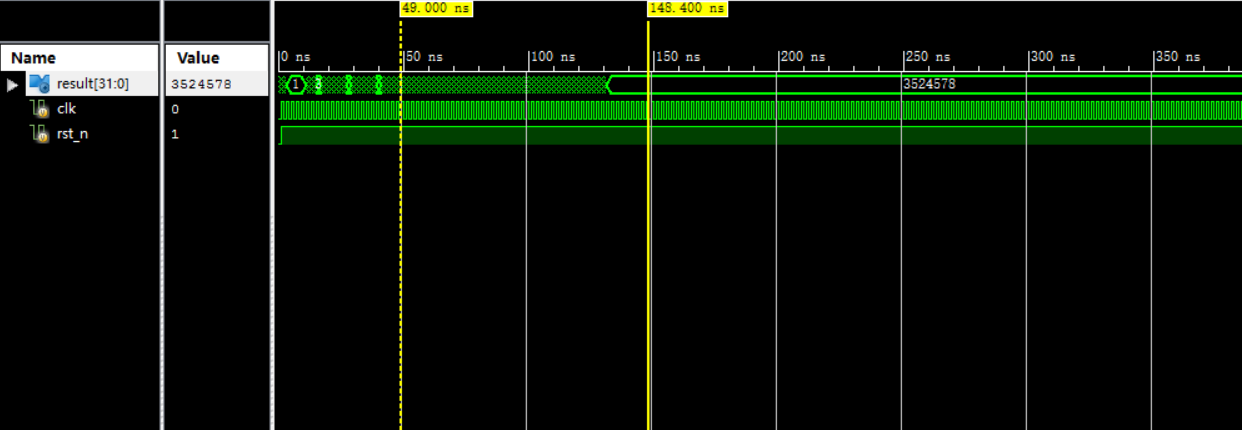
接着，例化一个ram

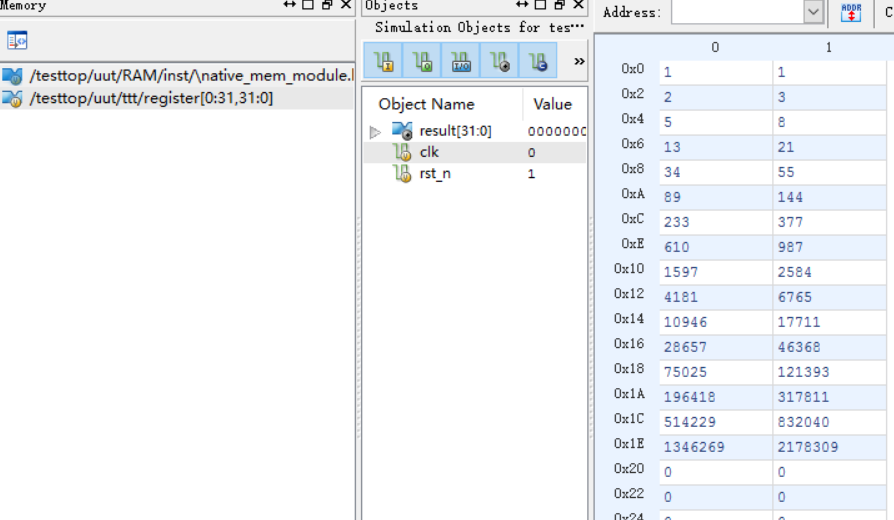
然后把代码部分完成，调试，得到正确的仿真结果。

* 代码部分

这次的代码部分中的alu模块与REG\_FILE模块与上次相同，control模块中则添加了一个always块，用来判断din是由ram给出还是有alu运算给出（前两次由ram读出，后面的由alu给出）

* 实验结果





* 源代码：

module alu(

input signed [31:0] alu\_a,

input signed [31:0] alu\_b,

input [4:0] alu\_op,

output reg [31:0] alu\_out

);

always@(\*)

begin

case(alu\_op)

1: alu\_out <= alu\_a + alu\_b;

2: alu\_out <= alu\_a - alu\_b;

3: alu\_out <= alu\_a & alu\_b;

4: alu\_out <= alu\_a | alu\_b;

5: alu\_out <= alu\_a ^ alu\_b;

6: alu\_out <= ~(alu\_a | alu\_b);

default:alu\_out <= alu\_out;

endcase

end

endmodule

module REG\_FILE(

input clk,input rst\_n,input [4:0] r1\_addr,input [4:0] r2\_addr,input [4:0] r3\_addr,input [31:0] r3\_din,input r3\_wr,output reg [31:0] r1\_dout,output reg [31:0] r2\_dout );

reg [31:0] register[0:31];

integer k,i;

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

begin

register[0]=32'b0;

register[1]=32'b0;

for(k=2;k<32;k=k+1)

register[k]=32'b0;

end

else if(r3\_wr)

begin

i=r3\_addr;

register[i]=r3\_din;

i=r1\_addr;

r1\_dout=register[i];

i=r2\_addr;

r2\_dout=register[i];

end

else

begin

i=r1\_addr;

r1\_dout=register[i];

i=r2\_addr;

r2\_dout=register[i];

end

end

endmodule

module control(

input clk,

input rst\_n,

input [31:0] doutb,

input [31:0] aluin,

output reg [31:0] din,

output reg wea,

output reg [5:0] addra,

output reg [5:0] addrb,

output reg [4:0] r1\_addr,

output reg [4:0] r2\_addr,

output reg [4:0] r3\_addr,

output reg r3\_wr

);

reg state;

reg [2:0] read;

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

begin

read=2'b00;

addrb=6'b00;

end

else if(read<=2'b10)

begin

read=read+1'b1;

addrb=addrb+1'b1;

din=doutb;

end

else

din=aluin;

end

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

begin

state=1'b0;

wea=1'b0;

addra=5'b01;

r1\_addr=5'b00;

r2\_addr=5'b01;

r3\_addr=5'b00;

r3\_wr=1;

end

else

begin

if(state==1'b0&&read>2'b01)

begin

r3\_wr=1;

state=1'b1;

end

else if(state==1'b1 &&r3\_addr<5'b101111)

begin

r3\_wr=0;

wea=0;

r3\_addr=r3\_addr+1;

state=1'b0;

if(r3\_addr>=5'b10)

begin

r1\_addr=r1\_addr+5'b1;

r2\_addr=r2\_addr+5'b1;

addra=addra+5'b1;

wea=1;

end

end

else

begin

wea=0;

r3\_wr=0;

end

end

end

endmodule

module top(

input clk,

input rst\_n,

output [31:0] result

);

assign result=r3\_din;

wire [4:0] alu\_op;

wire [31:0] r1\_dout;

wire [31:0] r2\_dout;

wire [31:0] r3\_din;

wire [4:0] r1\_addr;

wire [4:0] r2\_addr;

wire [4:0] r3\_addr;

wire r3\_wr;

wire wea;

wire [31:0] doutb;

wire [5:0] addra;

wire [5:0] addrb;

wire [31:0] alu\_in;

assign alu\_op=5'h01;

ram RAM(

.clka(clk),

.wea(wea),

.addra(addra),

.dina(r3\_din),

.clkb(clk),

.addrb(addrb),

.doutb(doutb)

);

control uut(

.clk(clk),

.rst\_n(rst\_n),

.doutb(doutb),

.aluin(alu\_in),

.din(r3\_din),

.wea(wea),

.addra(addra),

.addrb(addrb),

.r1\_addr(r1\_addr),

.r2\_addr(r2\_addr),

.r3\_addr(r3\_addr),

.r3\_wr(r3\_wr)

);

alu utt(

.alu\_a(r1\_dout),

.alu\_b(r2\_dout),

.alu\_op(alu\_op),

.alu\_out(alu\_in)

);

REG\_FILE ttt(

.clk(clk),

.rst\_n(rst\_n),

.r1\_addr(r1\_addr),

.r2\_addr(r2\_addr),

.r3\_addr(r3\_addr),

.r3\_din(r3\_din),

.r3\_wr(r3\_wr),

.r1\_dout(r1\_dout),

.r2\_dout(r2\_dout)

);

endmodule

module testtop;

// Inputs

reg clk;

reg rst\_n;

// Outputs

wire [31:0] result;

// Instantiate the Unit Under Test (UUT)

top uut (

.clk(clk),

.rst\_n(rst\_n),

.result(result)

);

initial begin

// Initialize Inputs

clk =1;

rst\_n = 0;

// Wait 100 ns for global reset to finish

#1;

rst\_n= 1;

// Add stimulus here

end

always

begin

clk=~clk;

#1;

end

endmodule