**华中科技大学**

**实验报告**

**课程名称：微机原理与接口技术实验**

**实验项目名称： 类MIPS单周期微处理器设计**

**指导老师：罗杰**

**专业/班级：电磁1802**

**学生学号：U201813405**

**学生姓名：吴叶赛**

**成绩/等级（批阅教师）：**

**实验执行日期： 2020年9月8日**

类MIPS单周期微处理器设计

1. **实验任务、目标**

利用Verilog HDL语言，基于Xilinx FPGA nexys4实验平台，设计一个能够执行以下MIPS指令集的单周期类MIPS处理器，要求完成所有支持指令的功能仿真，验证指令执行的正确性，要求编写汇编程序将本人学号的ASCII码存入RAM的连续内存区域。

▪ 支持基本的算术逻辑运算如add，sub，and，or，slt，andi指令

▪ 支持基本的内存操作如lw，sw指令

▪ 支持基本的程序控制如beq，j指令

（由于特殊情况，此次实验仅完成Reg、ALU、ALUCtr三个模块并分别加以仿真和验算）

1. **微处理器各个模块硬件设计原理、verilog代码**
2. **Reg设计**

Reg是指令操作地主要对象，MIPS微处理器中一共32个32位寄存器。该模块输入为 RsAddr，RtAddr，WriteAddr，RegWr,WriteData,clk,reset有两项输出 RsData,RtData。其中当复位信号reset为1时所有寄存器清零，当复位信号为0且写信号有效时数据才写入writedata寄存器。（其中0号寄存器永远为0）具体代码如下:

module RegFile(

input [4:0] RsAddr,

input [4:0] RtAddr,

input [4:0] WriteAddr,

input RegWr,

input [31:0] WriteData,

input clk,

input reset,

output [31:0] RsData,

output [31:0] RtData

);

reg [31:0] regs[0:31];

assign RsData =(RsAddr==5'b0) ? 32'b0:regs[RsAddr];

assign RtData =(RtAddr==5'b0) ? 32'b0:regs[RtAddr];

integer i;

always @(negedge clk) //下降沿回写到寄存器

if(!reset&RegWr)

regs[WriteAddr]=WriteData;

else if(reset)

for(i=0;i<32;i=i+1)

regs[i]=4\*i; //寄存器的值初始化为编号乘以4

Endmodule

1. **ALU设计**

微处理器支持add,sub,and,or,slt运算指令，需要用ALU单元实现运算，同时sw,lw,beq等指令也要用到ALU单元计算地址，比较数据，因此我们设计此模块输入为in1,in2,(此处要设置为signed类型，不然无法执行slt指令），ALUctr两个输出Res,zero。对于ALUctr的控制信号设置为0b0110对应sub,0b0010对应add,0b0000对应and，0b0001对应or，0b0111对应小于设置，具体代码如下：

module ALU(

input signed [31:0] SrcA,

input signed [31:0] SrcB,

input [3:0] ALUCtr,

output reg [31:0] ALURes,

output reg zero

);

always @(SrcA or SrcB or ALUCtr) begin

case (ALUCtr)

4'b0000://and

begin

ALURes = SrcA & SrcB;

zero=0;

end

4'b0010: //add

begin

ALURes = SrcA + SrcB;

zero=0;

end

4'b0001: //or

begin

ALURes = SrcA | SrcB;

zero=0;

end

4'b0110: //sub

begin

ALURes = SrcA - SrcB;

zero=(ALURes==0)?1:0;

end

4'b0111: //slt

begin

ALURes =( SrcA < SrcB)?1:0;

zero=0;

end

default:

begin

ALURes =0;

zero=0;

end

endcase

end

Endmodule

1. **ALUCtr**

微处理器中的AlUCtr模块负责处理ALUop和func指令，并生成ALUCtr指令传递到ALU中，决定ALU采取什么运算。具体代码如下：

module ALUCtr(

input [1:0] ALUop,

input [5:0] func,

output reg [3:0] ALUCtr

);

always @(ALUop or func)

casex ({ALUop,func})

8'b00xxxxxx: ALUCtr= 4'b0010; // lw/sw, +

8'b01xxxxxx: ALUCtr <= 4'b0110; // beq, -

8'b10xx0000: ALUCtr<=4'b0010; // add

8'b10xx0010: ALUCtr<=4'b0110; // sub

8'b10xx0100: ALUCtr<=4'b0000; // and

8'b10xx0101: ALUCtr<=4'b0001; // or

8'b10xx1010: ALUCtr<=4'b0111; // slt

default: ALUCtr<=4'b0000;

endcase

Endmodule

1. **各个模块的仿真激励代码、仿真结果截图以及文字说明如何验证其正确性**

**（1）Reg仿真**

module Regsim(

output [31:0] RsData,

output [31:0] RtData

);

reg [4:0] RsAddr;

reg [4:0] RtAddr;

reg [4:0] WriteAddr;

reg RegWr;

reg [31:0] WriteData;

reg clk;

reg reset;

parameter PERIOD = 10;

always begin

clk = 1'b0;

#(PERIOD/2) clk = 1'b1;

#(PERIOD/2);

end

initial

begin

reset=1;

RsAddr=5'h0;

RtAddr=5'h0;

#15

reset=0;

#30

RegWr=1;

WriteAddr=5'h03;

WriteData=32'h1803;

#20

RsAddr=5'h03;

RtAddr=5'h03;

end

RegFile U0(RsAddr,RtAddr,WriteAddr,RegWr,WriteData,clk,reset,RsData,RtData);

Endmodul



图一 Reg仿真结果

由上图可知，在写入信号输入前，读取信号和结果都为0，对应的寄存器也是初始值。而写入信号输入后寄存器的值发生改变，读取信号和值也改变。与预计结果一致。

**（2）ALU仿真**

module alusim(

output [31:0] ALURes,

output zero

);

reg [31:0] SrcA;

reg [31:0] SrcB;

reg [3:0] ALUCtr;

ALU U0(SrcA,SrcB,ALUCtr,ALURes,zero);

initial

begin

SrcA=32'hffff0000;

SrcB=32'h00ffff00;

ALUCtr=4'h2;//add

#10

ALUCtr=4'h6;//sub

#10

ALUCtr=4'h0;//and

#10

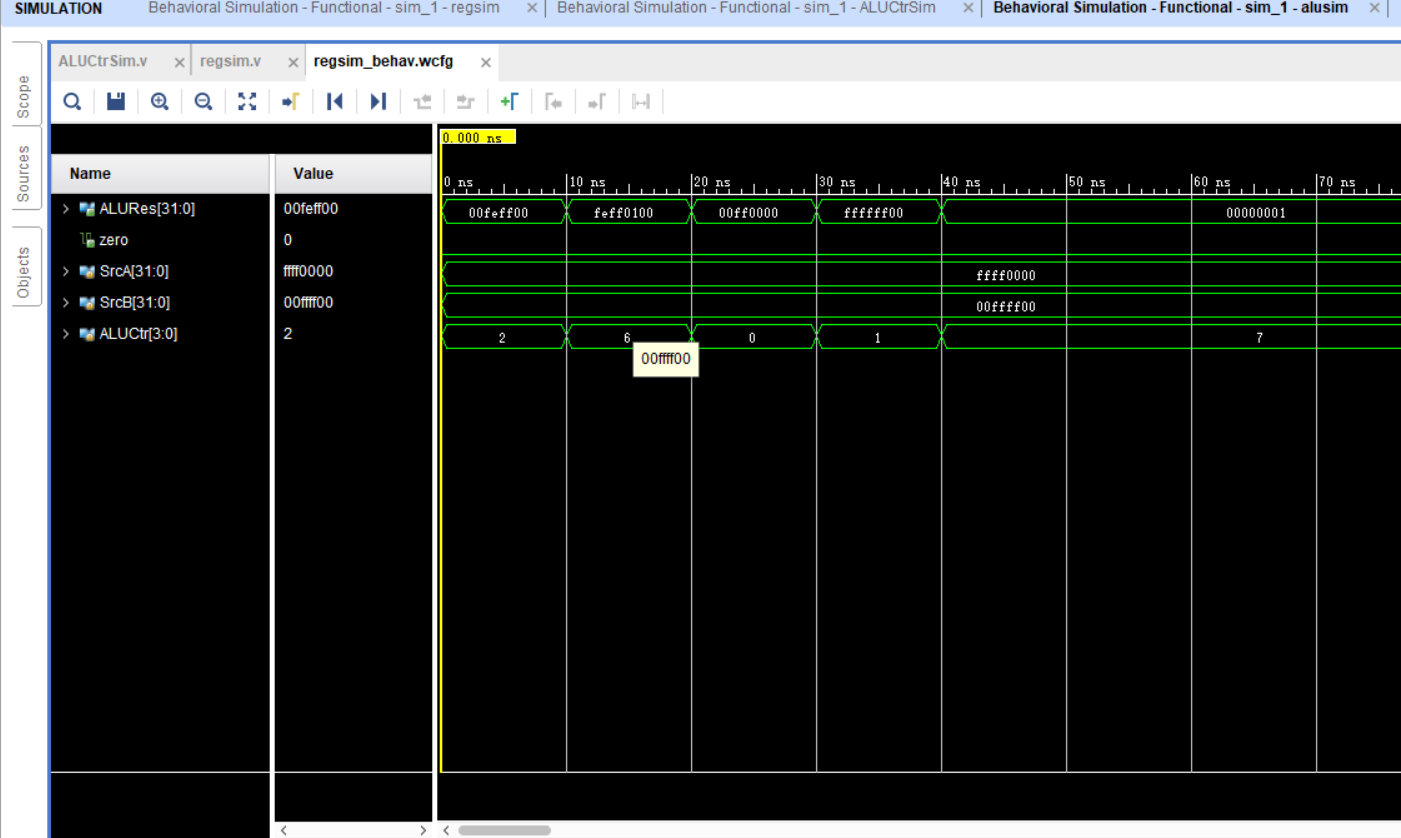
ALUCtr=4'h1;//or

#10

ALUCtr=4'h7;//slt

end

Endmodule



图二 ALU仿真结果

如上图所示，对于不同的ALUCtr信号，ALU可以输出正确的指令。

1. **ALUCtr仿真**

module ALUCtrSim(

output [3:0] ALUCtr

);

reg [1:0] ALUop;

reg [5:0] func;

ALUCtr U0(ALUop,func,ALUCtr);

initial

begin

ALUop=00;//l/s

#20

ALUop=01; //beq

#20

{ALUop,func}=8'b10100000; //add

#20

{ALUop,func}=8'b10100010; //sub

#20

{ALUop,func}=8'b10100100;//and

#20

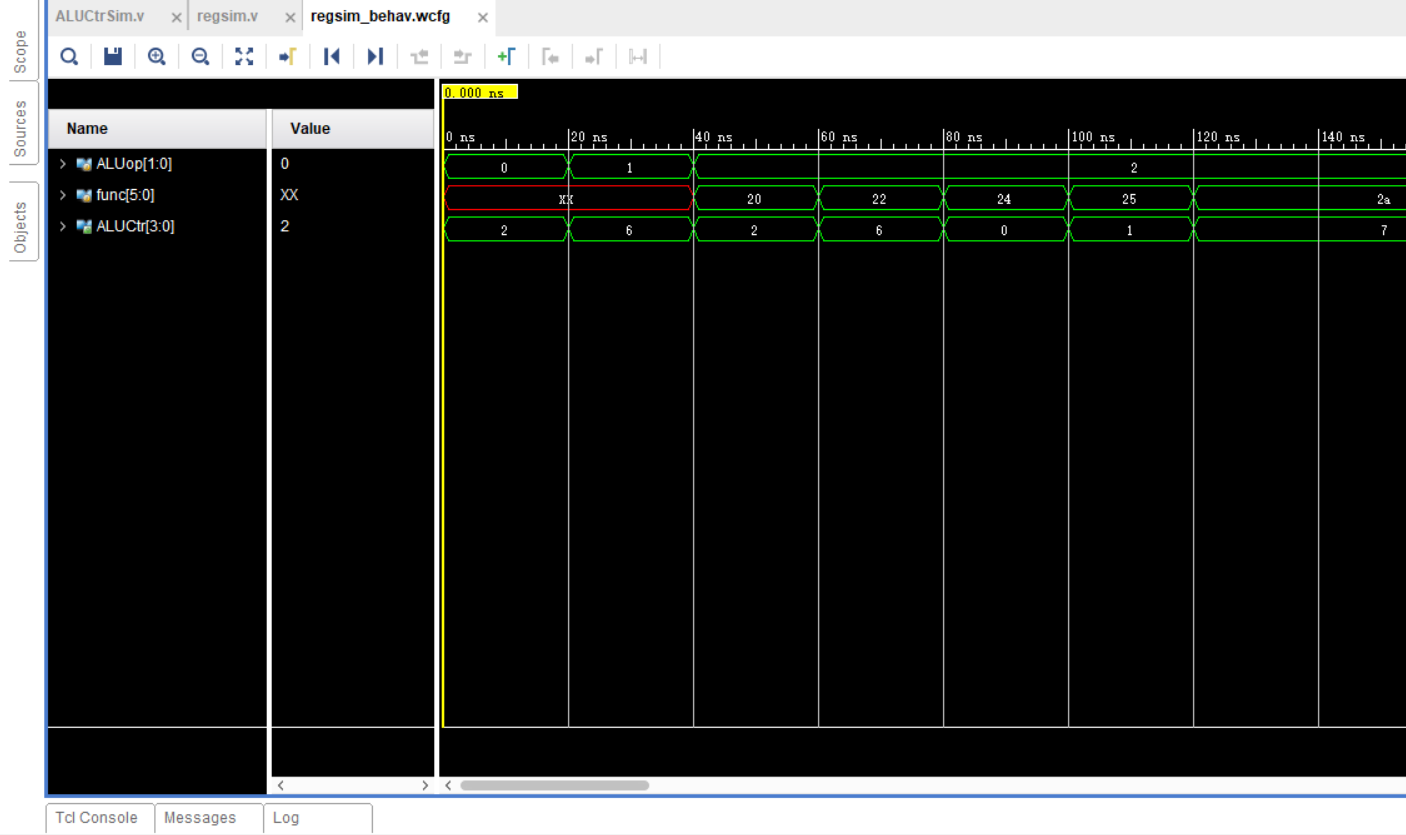
{ALUop,func}=8'b10100101;//or

#20

{ALUop,func}=8'b10101010;//slt

end

Endmodule



图三 ALUCtr仿真

如上图所知，根据不同的ALUop，ALUCtr可以输出正确的ALUCtr指令给ALU。

1. **心得体会**

经过本次实验，我学会了如何使用vivado。了解用Verilog 硬件描述语言实现了简单Mips微处理器的实现和仿真。学会了Verilog 的程序编写和对仿真波形图的观察分析以及通过仿真图来进行代码的调试。对MIPS单周期微处理器有了更深的理解。