**电子科技大学信息与软件工程学院**

**标 准 实 验 报 告**

**（实验）课程名称： 单周期CPU的 设计与实现**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名： 学 号：2014060106001 指导教师：**

**实验地点：A2 410 实验时间：周六晚19:00至22：00**

1. **实验室名称：**

**A2 410**

1. **实验项目名称：**

单周期CPU的设计与实现。

1. **实验学时：**

实验学时是8学时。

1. **实验原理：**

本实验的单周期CPU的工作原理如下图1如示。

图1 单周期CPU总体电路

Clock

4

0 1 2 3

rs

op

rt

rd

func

Wreg

aluc

pcsource

S

sa

Inst

Mem

A

Do

+

ALU

A

B

A\_data

A\_addr

B\_addr

Write

W\_addr

B\_data

Data

f

S

PC

S

e

S

aluimm

shift

Result

Data

Mem

A

Do

We

S

wmem

m2reg

+

<<

Di

imm

addr

<<

Zero

regrt

sext

jal

Control

Unit

一条指令的执行过程包括：取指令→分析指令→执行指令→保存结果（如果有的话）。对于单周期CPU来说，这些执行步骤均在一个时钟周期内完成。

1. 本实验的所设计的指令共有20条，分为4种类型：R型、I型和J型。不同的指令类型有不同的数据路径。
2. 在每个时钟周期根据当前PC的内容取出一条指令，将指令的op和func送控制器（Control Unit）译码，产生指令执行的控制码和运算码，控制指令的执行。
3. 寄存器堆、多路选择器、数据存储器及运算器等部件在相应控制码的控制下工作，产生指令的输出数据流，并将结果写入相应的存储部件。
4. 在下一个时钟周期到来时，重复2、3步骤，直到程序执行完成。
5. **实验目的**

通过本实验，让学生掌握单周期CPU的工作原理，控制器、运算器等部件设计的基本方法和技能，加深对所学知识的理解和掌握。

通过使用硬件描述语言Verilog、EDA工具软件进行软件设计与仿真，并在FPGA上实现，以培养学生的分析和设计CPU的能力。

1. **实验内容**
2. 拟定本实验的指令系统，指令应有计算类型、访问存储器类型、条件转移类型和无条件转移等，指令数应不少于20条。
3. CPU各功能模块的设计与实现；
4. CPU的封装；
5. 对所做设计的仿真测试和板级验证。
6. **实验器材（设备、元器件）：**
7. 安装了Xilinx ISE Design Suite 13.4的PC机一台
8. FPGA开发板：Anvyl Spartan6/XC6SLX45
9. 计算机与FPGA开发板通过JTAG（Joint Test Action Group）接口连接，其连接方式如图1所示。

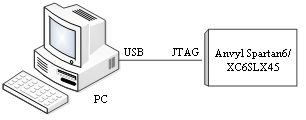


图1 PC计算机与FPGA开发板的连接

1. **实验步骤**
2. 拟定本实验所设计CPU的指令系统；

本实验所设计CPU支持的指令共有20条MIPS，包含了R型、I型和J型指令，如表1所示。

表 1 本实验所涉及的20条MIPS指令

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| R型指令 | | | | | | | |
| 指令 | [31:26] | [25:21] | [20:16] | [15:11] | [10: 6] | [5:0] | 功能 |
| Add | 000000 | rs | rt | rd | 000000 | 100000 | 寄存器加 |
| Sub | 000000 | rs | rt | rd | 000000 | 100010 | 寄存器减 |
| And | 000000 | rs | rt | rd | 000000 | 100100 | 寄存器与 |
| Or | 000000 | rs | rt | rd | 000000 | 100101 | 寄存器或 |
| Xor | 000000 | rs | rt | rd | 000000 | 100110 | 寄存器异或 |
| Sll | 000000 | 00000 | rt | rd | sa | 000000 | 左移 |
| Srl | 000000 | 00000 | rt | rd | sa | 000010 | 逻辑右移 |
| Sra | 000000 | 00000 | rt | rd | sa | 000011 | 算术右移 |
| Jr | 000000 | rs | rt | rd | 000000 | 001000 | 寄存器跳 |
| I型指令 | | | | | | | |
| Addi | 001000 | rs | rt | immediate | | | 立即数加 |
| Andi | 001100 | rs | rt | immediate | | | 立即数与 |
| Ori | 001101 | rs | rt | immediate | | | 立即数或 |
| Xori | 001110 | rs | rt | immediate | | | 立即数异或 |
| Lw | 100011 | rs | rt | offset | | | 取数据 |
| Sw | 101011 | rs | rt | offset | | | 存数据 |
| Beq | 000100 | rs | rt | offset | | | 相等转移 |
| Bne | 000101 | rs | rt | offset | | | 不等转移 |
| Lui | 001111 | 00000 | rt | immediate | | | 设置高位 |
| J型指令 | | | | | | | |
| J | 000010 | address | | | | | 跳转 |
| Jal | 000011 | address | | | | | 调用 |

1. 新建工程（New Project）

启动ISE Design Suite 13.4软件，然后选择菜单File→New Project，弹出New Project Wizard对话框，在对话框中输入工程名Single\_Cycle\_CPU，并指定工作路径D:\Single\_Cycle\_CPU。

1. 基本功能器件的设计与实现
2. 多路选择器的设计与实现

* 32位2选1多路选择器的设计与实现

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称为：mux32，然后输入其实现代码：

module mux32(

input wire [31:0] a1,

input wire [31:0] a2,

input wire s,

output wire [31:0] p

);

assign p = s ? a2 : a1;

endmodule

在ISE集成开发环境中，对模块mux32进行综合（Synthesize），综合结果如图2所示。

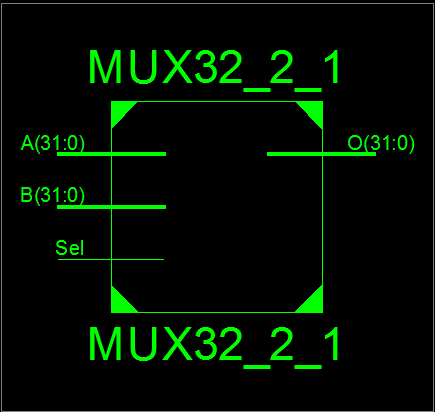


图2 模块MUX32\_2\_1的综合（Synthesize）结果

在ISE集成开发环境中，对模块MUX32\_2\_1进行仿真（Simulation）。首先输入如下测式代码：

module mux32\_test;

reg [31:0] A;

reg [31:0] B;

reg Sel;

wire [31:0] O;

// Instantiate the Unit Under Test (UUT)

mux32 uut (

.a1(A),

.a2(B),

.s(Sel),

.p(O)

);

initial begin

// Initialize Inputs

$dumpfile("mux32\_test.vcd");

$dumpvars(0,uut);

A = 0;

B = 0;

Sel = 0;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

A = 32'haaaa0000;

B = 32'hbbbb1111;

Sel = 1'b0;

#100;

Sel = 1'b1;

#100;

$stop;

end

endmodule

然后进行仿真，仿真结果如图3所示。

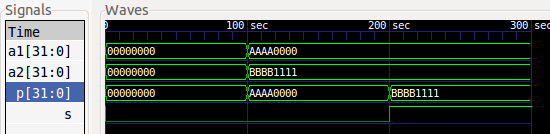


图3 模块mux32的仿真结果

* 5位2选1多路选择器的设计与实现

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称为：mux5，然后输入其实现代码：

module mux5(

input wire [4:0] a,

input wire [4:0] b,

input wire s,

output wire [4:0] y

);

assign y = s ? b : a;

endmodule

在ISE集成开发环境中，对模块mux5进行综合（Synthesize），综合结果如图5所示。

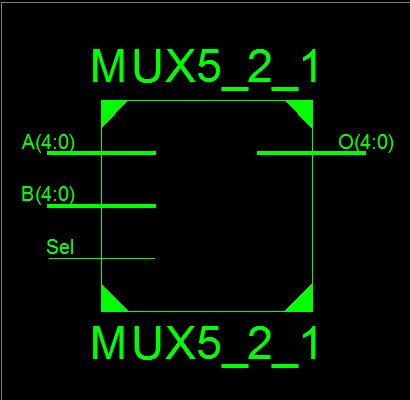


图5 模块MUX5\_2\_1的综合（Synthesize）结果

1. 移位器的设计与实现

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称为：lshift2，然后输入其实现代码：

module lshift2(

input wire [31:0] a,

output wire [31:0] b

);

assign b = a << 2;

endmodule

1. 32位加法器的设计与实现

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称为：add32，然后输入其实现代码：

module add32(

input wire [31:0] a,

input wire [31:0] b,

output wire [31:0] p

);

assign p = a + b;

endmodule

1. 运算器（ALU）设计与实现

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称为：alu，然后输入其实现代码：

module alu(

input wire [31:0] A,

input wire [31:0] B,

input wire [2:0] optr,

output wire [31:0] ret,

output wire zero

);

assign ret = (optr == 3'b000) ? A + B :

(optr == 3'b100) ? A - B :

(optr == 3'b001) ? A & B :

(optr == 3'b101) ? A | B :

(optr == 3'b010) ? A ^ B :

(optr == 3'b110) ? {B[15],16'h0} : 0;

assign zero = ~|ret;

endmodule

1. 寄储器堆（Register File）设计与实现

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称为：reg32，然后输入其实现代码：

module reg32(

input wire [4:0] rn1,

input wire [4:0] rn2,

input wire [4:0] wn,

input wire write,

input wire [31:0] wd,

input wire clk,

output wire [31:0] A,

output wire [31:0] B

);

reg [31:0] data[31:0];

assign A = rn1 == 0 ? 0 : data[rn1];

assign B = rn2 == 0 ? 0 : data[rn2];

always @ (posedge clk) begin

if(write && wn != 0) begin

data[wn] <= wd;

end

end

endmodule

在ISE集成开发环境中，对模块reg32进行综合（Synthesize），综合结果如图10所示。

r

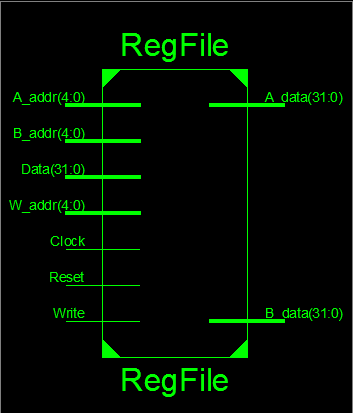


图10 模块RegFile的综合（Synthesize）结果

1. 控制器（Control Unit）设计与实现

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称分别为：ALUOp，然后输入其实现代码：

module ALUOp(

input wire [5:0] func,

input wire [1:0] ALUctr,

output wire [2:0] ALUop

);

wire temp = ALUctr[1] & ~ALUctr[0];

assign ALUop[2] = (temp & ((~func[2] & func[1]) | {func[2] & func[0]}))| ALUctr[0];

assign ALUop[1] = (temp & func[2] & func[1]) | (ALUctr[1] & ALUctr[0]);

assign ALUop[0] = (temp & func[2] & ~func[1]);

endmodule

再新建一个Verilog模块，名称为Control，输入实现代码：

module Control(

input wire [5:0] op,

output wire RegDst , RegWrite , ALUSrc ,

output wire MemWrite , MemRead , MemtoReg ,

output wire Branch ,

output wire [1:0] ALUctr

);

wire i\_Rt = ~|op;

wire i\_lw = op[5] & ~op[3];

wire i\_sw = op[5] & op[3];

wire i\_beq = op[2] & ~op[1];

wire i\_lui = op[3] & op[2];

assign RegDst = i\_Rt;

assign RegWrite = i\_Rt | i\_lw | i\_lui;

assign ALUSrc = i\_lw | i\_sw | i\_lui;

assign MemWrite = i\_sw;

assign MemRead = i\_lw;

assign MemtoReg = i\_lw;

assign Branch = i\_beq;

assign ALUctr[1] = i\_Rt | i\_lui;

assign ALUctr[0] = i\_beq | i\_lui;

endmodule

再新建一个Verilog模块，名称为Controller，输入实现代码：

module Controller(

input wire [5:0] op , func,

output wire RegDst , RegWrite , ALUSrc,

output wire MemWrite , MemRead , MemtoReg,

output wire Branch ,

output wire [2:0] ALUop

);

wire [1:0] ALUctr;

Control U0( .op(op) ,

.RegDst(RegDst) ,

.RegWrite(RegWrite) ,

.ALUSrc(ALUSrc) ,

.MemWrite(MemWrite) ,

.MemRead(MemRead) ,

.MemtoReg(MemtoReg) ,

.Branch(Branch) ,

.ALUctr(ALUctr));

ALUOp U1(.func(func),.ALUctr(ALUctr),.ALUop(ALUop));

endmodule

在ISE集成开发环境中，对模块Controller进行综合（Synthesize），综合结果如图11所示。

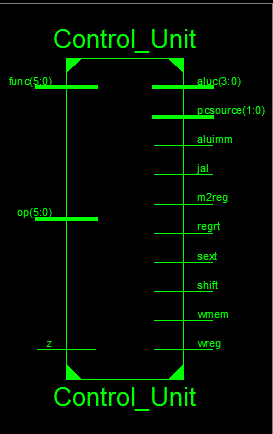


图11 模块Control\_Unit的综合（Synthesize）结果

1. CPU的封装

CPU封装的基本原理就是根据单单周期CPU的总体电路将各功能部件连接起来按照指令的数据路径连接起来。

在ISE集成开发环境中，在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，创建一个Verilog Module模块，名称为：mips\_cpu，然后输入其实现代码：

module mips\_cpu(

input wire [31:0] Inst,

input wire res,clk,

input wire [31:0] ReadData,

output wire [31:0] result,b\_data,InstAddr,

output wire MemWrite,MemRead

);

wire RegDst,RegWrite,ALUSrc,Branch,Zero,MemtoReg;

wire [2:0] ALU\_op;

wire [4:0] Wn;

wire [31:0] A,B,Ext\_Imm,ALU\_B,data;

assign b\_data = B;

fetch U0(Branch,Zero,res,clk,Ext\_Imm,InstAddr);

Controller U1 (

.op(Inst[31:26]),

.func(Inst[5:0]),

.RegDst(RegDst),

.RegWrite(RegWrite),

.ALUSrc(ALUSrc),

.MemWrite(MemWrite),

.MemRead(MemRead),

.MemtoReg(MemtoReg),

.Branch(Branch),

.ALUop(ALU\_op)

);

mux5 U2 (Inst[20:16],Inst[15:11],RegDst,Wn);

mux32 U7 (result,ReadData,MemtoReg,data);

reg32 U3 (Inst[25:21],Inst[20:16],Wn,RegWrite,data,clk,A,B);

sign\_extender U4 (Inst[15:0],Ext\_Imm);

mux32 U5(B,Ext\_Imm,ALUSrc,ALU\_B);

alu U6 (A,ALU\_B,ALU\_op,result,Zero);

endmodule

在ISE集成开发环境中，对模块mips\_cpu进行综合（Synthesize）。

1. 测试
2. 指令存储器

在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，选择Verilog Module输入，并输入Verilog文件名：inst\_mem，单击Next按钮进入端口定义对话框。其中Module Name栏输入模块名inst\_mem，单击Next进入下一步，点击Finish完成创建。实现inst\_mem的Verilog程序参考代码：

module inst\_mem(

input wire [31:0] address,

output wire [31:0] inst

);

wire [31:0] ram[0:31];

//assign ram[5'h00] = 32'h00002820;

//assign ram[5'h01] = 32'h8CB10000;

// | | | | | |

assign ram[0] = 32'b00000000000000000000100000100000;//add $1,$0,$0

assign ram[1] = 32'b10001100001000100000000000000000;//lw $2,0($1)

assign ram[2] = 32'b10001100001000110000000000000100;//lw $3,4($1)

assign ram[3] = 32'b00000000010000110001000000100000;//add $2,$2,$3

assign ram[4] = 32'b10001100001000110000000000001000;//lw $3,8($1)

assign ram[5] = 32'b00010000010000110000000000000001;//beq $2,3,1

assign ram[6] = 32'b10101100001000000000000000001100;//sw $0,12($1)

assign ram[7] = 32'b10101100001000100000000000001100;//sw $2,12($1)

assign inst = ram[address[6:2]];

endmodule

1. 内存

在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，选择Verilog Module输入，并输入Verilog文件名：data\_mem，单击Next按钮进入端口定义对话框。其中Module Name栏输入模块名data\_mem，单击Next进入下一步，点击Finish完成创建。实现data\_mem的Verilog程序参考代码：

module data\_mem(

input Clock,

output[31:0] dataout,

input [31:0] datain,

input [31:0] addr,

input write , read

);

reg [31:0] ram [0:31];

assign dataout = read ? ram[addr[6:2]] : 32'hxxxxxxxx;

always @ (posedge Clock) begin

if(write)

ram[addr] <= datain;

end

initial begin

ram[0] <= 2;

ram[1] <= 3;

ram[2] <= 5;

end

endmodule

1. 仿真测试

将CPU、指令存储器和数据存储器连接起来。在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择New Source命令，选择Verilog Module输入，并输入Verilog文件名：Mainboard，单击Next按钮进入端口定义对话框。其中Module Name栏输入模块名Main\_board，单击Next进入下一步，点击Finish完成创建。实现Mainboard的Verilog程序参考代码：

module mainboard(

input wire clk,

input wire res,

output wire [31:0] inst,

output wire [31:0] pc,

output wire [31:0] result,

output wire [31:0] memout

);

wire [31:0] result;

wire [31:0] b\_data;

wire MemWrite;

wire MemRead;

mips\_cpu uut (

.Inst(inst),

.res(res),

.clk(clk),

.ReadData(memout),

.result(result),

.b\_data(b\_data),

.InstAddr(pc),

.MemWrite(MemWrite),

.MemRead(MemRead)

);

data\_mem U1 (

.Clock(clk),

.dataout(memout),

.datain(b\_data),

.addr(result),

.write(MemWrite),

.read(MemRead)

);

inst\_mem U2 (

.address(pc),

.inst(inst)

);

Endmodule

其连接如图13所示。

向工程中添加测试模块。综合通过后，在工程管理区将View设置为Simulation，在任意位置单击鼠标右键，并在弹出的菜单中选择New Source，在类型中选择Verilog Test Fixture，输入测试文件名：mainboard\_test，单击下一步。这时所有工程中的模块名都会显示出来，选择要进行测试的模块：mainboard。点击Next ,再单击Finish按钮，ISE会在源代码编辑区自动生成测试模块的代码。我们看到，ISE已经自动生成了基本的信号并对被测模块做了例化。

在Mainboard\_tb添加如下测试代码：

module mainboard\_test;

reg clk;

reg res;

wire [31:0] inst;

wire [31:0] pc;

wire [31:0] result;

wire [31:0] memout;

mainboard uut(

.clk(clk),

.res(res),

.inst(inst),

.pc(pc),

.result(result),

.memout(memout)

);

initial begin

$dumpfile("mainboard.vcd");

$dumpvars(0,uut);

// Initialize Inputs

res = 1;

clk = 0;

// Wait 100 ns for global reset to finish

#10;

res = 0;

#10;

res = 1;

// Add stimulus here

#1000;

$dumpflush;

$stop;

end

always begin

clk <= ~clk;

#50;

end

endmodule

完成测试文件编辑后，确认工程管理区中View选项设置为Simulation，并且选中的测试模块是Mainboard\_tb，这时在过程管理区会显示与仿真有关的进程。右键单击其中的Simulate Behavioral Model项，选择弹出菜单中的Process Properties项，会弹出属性设置对话框，将其中Simulation Run Time设置为1000ns。

在过程管理区双击Simulate Behavioral Model，ISE将启动ISE Simulator，可以得到仿真结果，如图14所示。

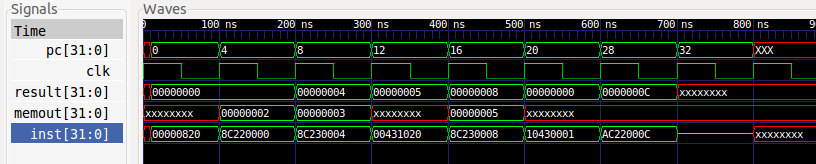


图14 CPU仿真测试结果

1. **实验数据及结果分析：**

在一个时钟周期内所设计的CPU能够完成一条指令的执行，指令执行结果与预期的结果是一致的：即内存的0号单元和1号单元的和是等于2号单元的，当PC到达20的时候会直接跳转到28,即将加法得到的结果存到4号单元中。

1. **实验结论：**

在一个时钟周期完成指令的所有执行步骤，可以使CPU的设计简化，但由于没有考虑不同部件完成时间上的差异，故CPU各部件的利用率不高。

1. **总结及心得体会：**
   1. 更加熟悉了CPU内部的具体实现，组成，以及各个模块的作用和实现方法，进一步加深了对计算机组成原理的理解。
   2. 进一步熟悉了Verilog硬件语言的使用以及工程编写以及仿真的方法。
2. **对本实验过程及方法、手段的改进建议：**
   1. 本实验实现的CPU结构比较简单，实现的指令条数也比较少，可以适当增加要实现的指令条数。
   2. 可以考虑将某些寄存器映射到板子的LED上，从而实现编写一个流水灯。

**报告评分：**

**指导教师签字：**