实验五 数据通路与控制单元设计

**一、实验目的**

1 掌握Verilog语言和Vivado、Logisim开发平台的使用；

2 掌握数据通路与控制单元的设计和测试方法。

**二、实验内容**

1 数据通路的设计；

2 控制单元的设计。

**三、实验要求**

1 掌握Vivado或Logisim开发工具的使用，掌握以上电路的设计和测试方法；

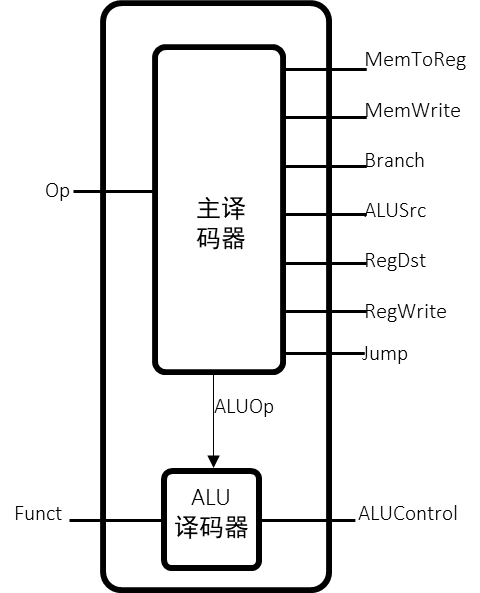
2 记录设计和调试过程（Verilog代码/电路图/表达式/真值表，Vivado仿真结果，Logisim验证结果等）；

3 分析Vivado仿真波形/Logism验证结果，注重输入输出之间的对应关系。

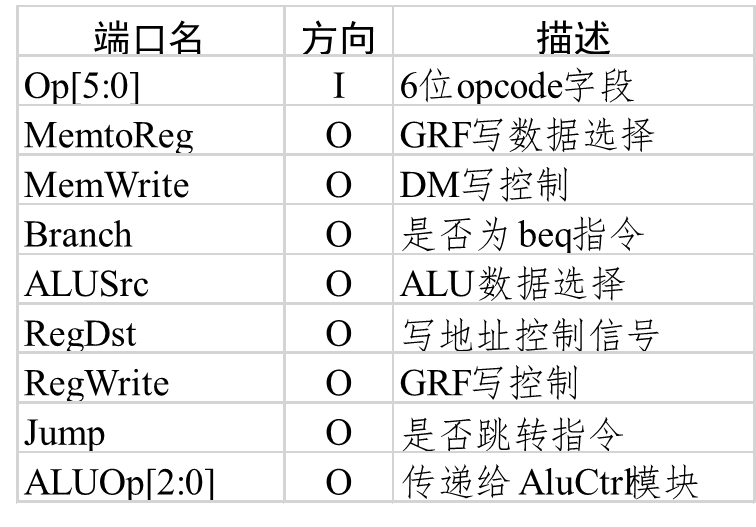
**四、实验过程及分析**

1 控制单元设计

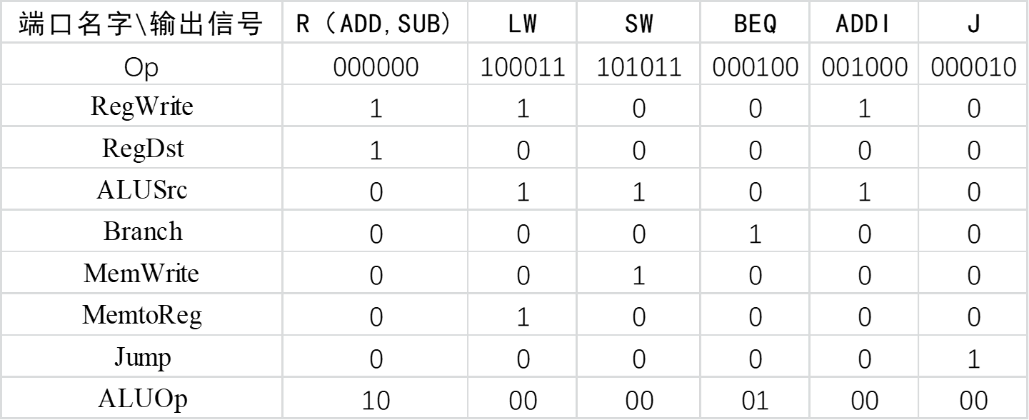
一个MIPS处理器的的控制单元逻辑框图如下：



主译码器各端口说明如下：



功能是根据6位输入操作码Op，在相应的数据通路上输出控制信号传输到实现对应功能的器件，实现控制功能。译码表如下：



主译码器设计代码：

module MainDec(

Op,MemToReg,MemWrite,Branch,ALUSrc,RegDst,RegWrite,Jump,ALUOp

);

input [5:0] Op;

output MemToReg,MemWrite;

output Branch,ALUSrc;

output RegDst,RegWrite;

output Jump;

output [1:0] ALUOp;

reg [8:0] Controls;

assign {RegWrite,RegDst,ALUSrc,Branch,MemWrite,MemToReg,Jump,ALUOp}=Controls;

always @(\*)

case (Op)

6'b000000:Controls<=9'b110000010;

6'b100011:Controls<=9'b101001000;

6'b101011:Controls<=9'b001010000;

6'b000100:Controls<=9'b000100001;

6'b001000:Controls<=9'b101000000;

6'b000010:Controls<=9'b000000100;

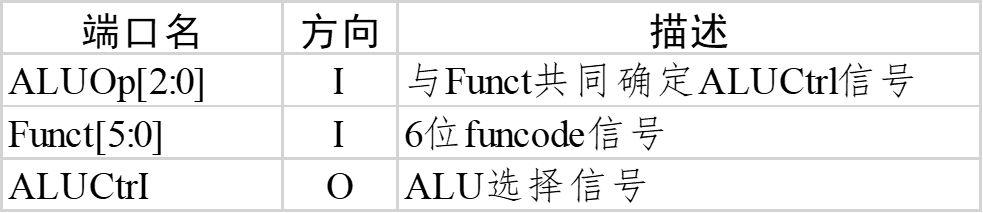
default :Controls<=9'bxxxxxxxxx;

endcase

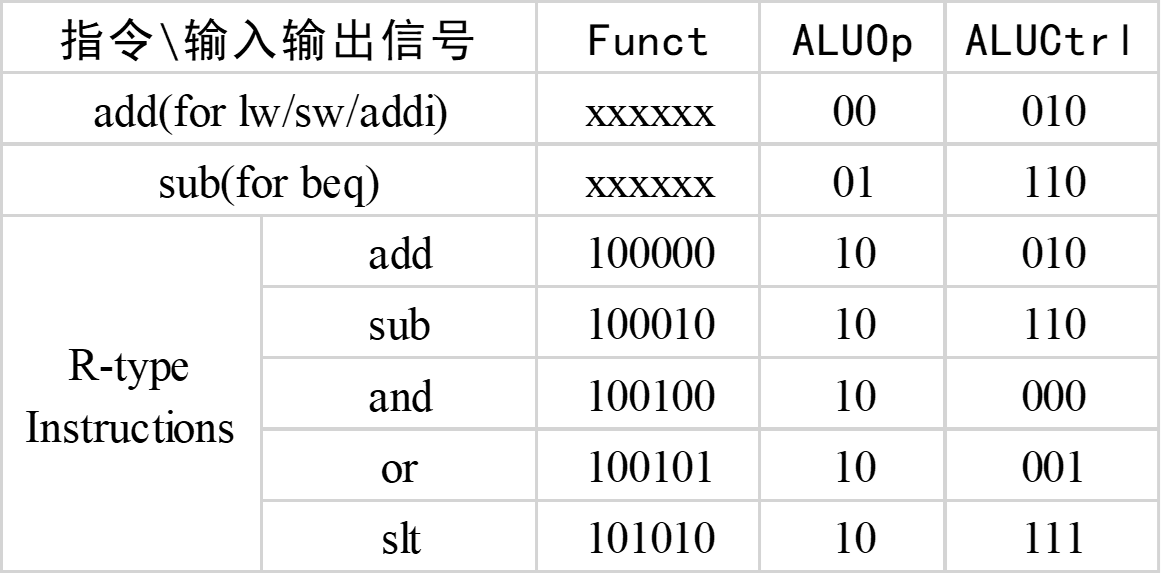
endmodule

根据译码表用case语句描述输入Op信号和输出控制信号Controls的对应关系。这里将各路输出控制信号合并描述位一个9位的信号。输入信号非法时，输出高阻态。

主译码器输出的2位ALU操作码ALUOp又作为ALU译码器的一个输入，和输入的ALU函数信号Funct共同决定输出的ALU控制信号ALUCtrl。ALU译码器各端口说明如下：



译码输出信号控制算数逻辑单元实现操作码对应的运算。译码表如下：



ALU译码器设计代码：

module ALUDec(

Funct,ALUOp,ALUControl

);

input [5:0] Funct;

input [1:0] ALUOp;

output reg [2:0] ALUControl;

always @(\*)

case(ALUOp)

2'b00:ALUControl<=3'b010;

2'b01:ALUControl<=3'b110;

default:case(Funct)

6'b100000:ALUControl<=3'b010;

6'b100010:ALUControl<=3'b110;

6'b100100:ALUControl<=3'b000;

6'b100101:ALUControl<=3'b001;

6'b101010:ALUControl<=3'b111;

default: ALUControl<=3'bxxx;

endcase

endcase

endmodule

同样是用case语句描述由ALUOp和Funct决定的输出ALUControl信号。ALUOp=00时，操作码只可能需要加运算；ALUOp=01时，只可能需要减运算。而ALUOp=10时，根据操作函数Funct决定要进行的运算。如果输入信号非法，同样输出高阻态。

将两个译码模块整合得到控制单元：

module Controller(

Op,Funct,Zero,MemToReg,MemWrite,PCSrc,ALUSrc,RegDst,RegWrite,Jump,ALUControl

);

input [5:0] Op,Funct;

input Zero;

output MemToReg,MemWrite;

output PCSrc,ALUSrc,RegDst,RegWrite,Jump;

output [2:0] ALUControl;

wire [1:0] ALUOp;

wire Branch;

MainDec MainDec\_1(Op,MemToReg,MemWrite,Branch,ALUSrc,RegDst,RegWrite,Jump,ALUOp);

ALUDec ALUDec\_1(Funct,ALUOp,ALUControl);

assign PCSrc=Branch&Zero;

endmodule

仿真测试代码：

module sim\_controller(

);

reg[5:0] Op,Funct;

reg Zero,clk1,clk2;

wire MemToReg,MemWrite,PCSrc,ALUSrc,RegDst,RegWrite,Jump;

wire[2:0] ALUControl;

reg[5:0] nextOp,nextFunct;

initial begin

Zero=0;

Op=6'b000000;

Funct=6'b100000;

clk1=0;

clk2=0;

end

initial begin

forever #30 clk1=~clk1;

end

initial begin

forever #5 clk2=~clk2;

end

always @(posedge clk1) begin

case(Op)

6'b000000: begin nextOp=6'b100011; Op=nextOp;end

6'b100011: begin nextOp=6'b101011; Op=nextOp;end

6'b101011: begin nextOp=6'b000100; Op=nextOp;end

6'b000100: begin nextOp=6'b001000; Op=nextOp;end

6'b001000: begin nextOp=6'b000010; Op=nextOp;end

6'b000010: begin nextOp=6'b000000; Op=nextOp;end

default: begin nextOp=6'b000000; Op=nextOp;end

endcase

end

always @(posedge clk2) begin

case(Funct)

6'b100000:begin nextFunct=6'b100010; Funct=nextFunct;end

6'b100010:begin nextFunct=6'b100100; Funct=nextFunct;end

6'b100100:begin nextFunct=6'b100101; Funct=nextFunct;end

6'b100101:begin nextFunct=6'b101010; Funct=nextFunct;end

6'b101010:begin nextFunct=6'b100000; Funct=nextFunct;end

default:begin nextFunct=6'b100000; Funct=nextFunct;end

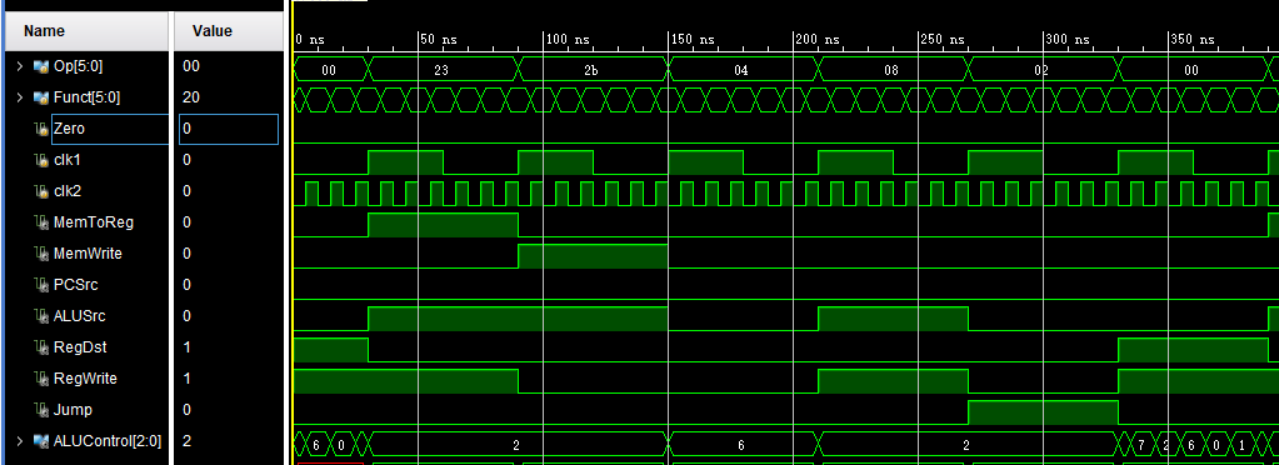
endcase

end

Controller test(Op,Funct,Zero,MemToReg,MemWrite,PCSrc,ALUSrc,RegDst,RegWrite,Jump,ALUControl);

Endmodule

时钟信号clk1周期为60ns，clk2为10ns，输入的Op每个clk1上升沿循环变化（有效输入信号，次序和设计代码中一致），Funct在每个clk2上升沿也循环变化（有效输入信号，次序和设计代码中一致）。由于设计中有效的Funct信号有6种，因此每个Op操作码都能测试到所有的Funct。观察到仿真波形如下：



可以看出：MemToReg只有在Op=23H=100011B时为1；MenWrite只有在Op=2bH=101011B时为1；ALUSrc在Op=23H，2bH，08H，即100011B，101011B，001000B时为1；RegDst只有在Op=00H=000000B时为1；RegWrite只有在Op=00H，23H即000000B，100011B时为1；Jump只在Op=02H=000010B时为1，和主译码器的译码表相符。

对于输出的ALUControl信号，对照主译码器译码表可以看出，只有操作码Op=000000B时，可以进行Funct定义的多种运算；对于Op=000100B时，只能进行减法运算；其他有效操作码都只会进行加法运算。波形图中对于ALUControl的输出波形对应关系符合设计期望。

**五、调试和心得体会**

1 对于控制单元的所有功能要一一进行仿真测试，模拟所有可能存在的有效输入信号，观察输入输出对应关系是否符合设计期望。本次实验仿真测试代码中用到两个不同周期的时钟信号，就好像分别用两个时钟信号去驱动主译码器和ALU译码器，使得它们的状态变化频率不同，而时钟信号的周期满足一定的倍数关系，这类似于程序设计中for循环的嵌套，可以很好的遍历所有的可能，确保设计无误、测试没有遗漏。