实验五 指令译码器设计与调试

**一、实验目的**

1. 学习并实现指令译码相关内容。

**二、实验内容**

1. 设计一个指令译码器。
2. 译码器支持的指令集为：{addu, subu, ori, lw, sw, beq, lui, nop}。
3. nop机器码为0x00000000， 即空指令，不进行任何有效行为（修改寄存器等）。
4. 设计ALU译码器。
5. addu,subu可以不支持溢出。

**三、实验要求**

1. 通过真值表验证译码模块。
2. 记录设计和调试过程。

**四、实验过程及结果**

Vivado实现：

代码:

module MainDec(

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

module ALUDec(

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

module Controller(

input [5:0] Op, Funct,

input Zero,

output MemToReg, MemWrite,

output PCSrc, ALUSrc,

output RegDst, RegWrite,

output 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

module sim\_controller();

reg [5:0] Op, Funct;

reg Zero;

wire MemToReg, MemWrite;

wire PCSrc, ALUSrc;

wire RegDst, RegWrite;

wire Jump;

wire [2:0] ALUControl;

initial begin

fork

#0 Zero <= 0;

#0 Op <= 6'b000000;

#0 Funct <= 6'b100000;

#10 Funct <= 6'b100010;

#20 Funct <= 6'b100100;

#30 Funct <= 6'b100101;

#40 Funct <= 6'b101010;

#50 Op <= 6'b100011;

#60 Op <= 6'b101011;

#70 Op <= 6'b000100;

#80 Op <= 6'b001000;

#90 Op <= 6'b000010;

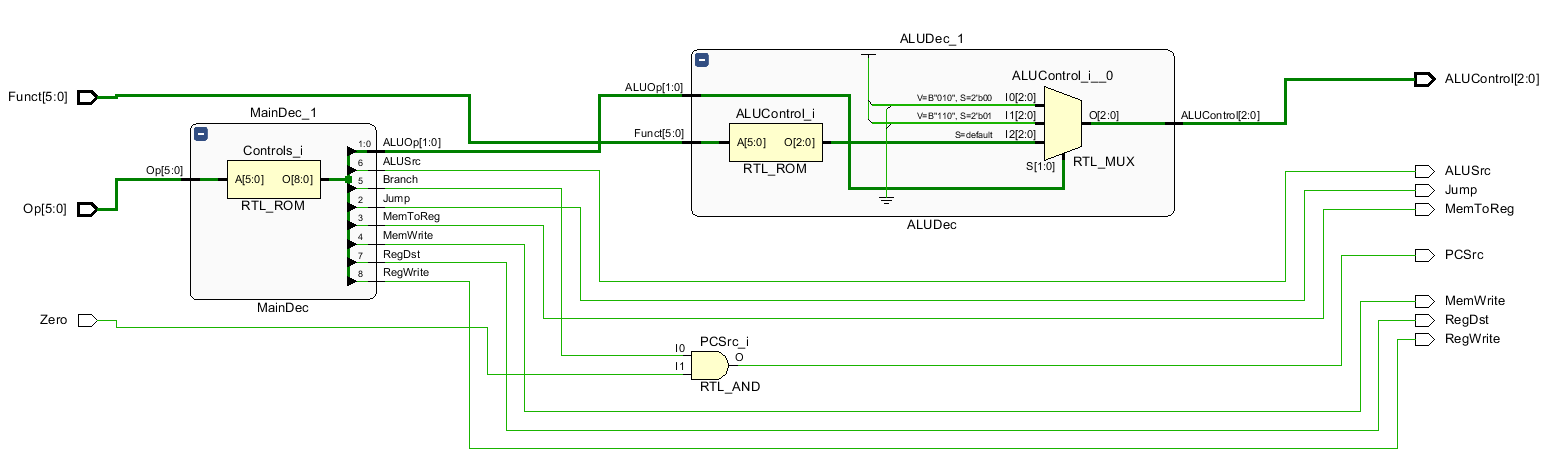
join

end

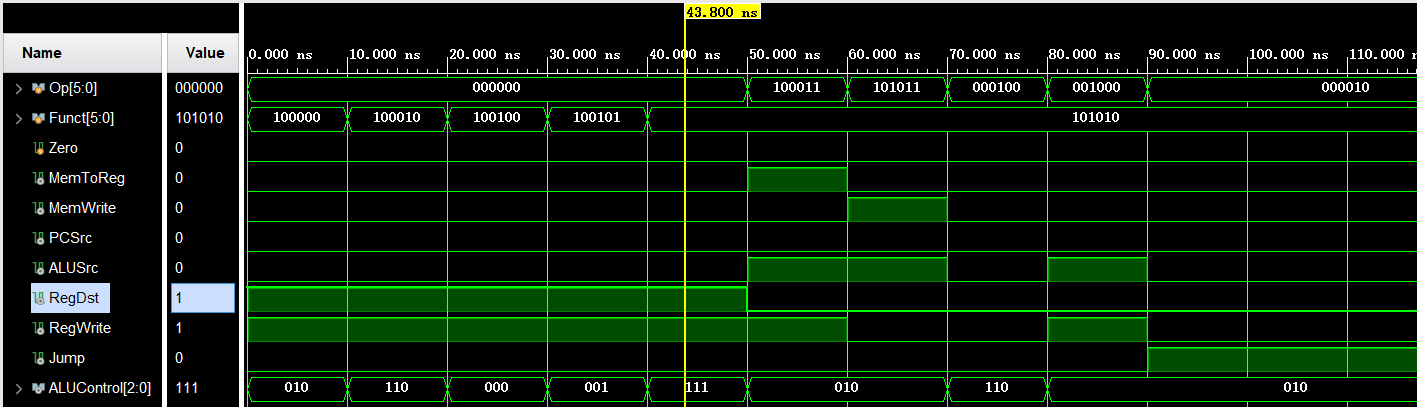
Controller Controller\_1(Op, Funct, Zero, MemToReg, MemWrite, PCSrc, ALUSrc, RegDst, RegWrite, Jump, ALUControl);

Endmodule

电路图：



仿真波形：



0-50ns，Op=000000，RegWrite=1，RegDst=1，执行R型指令。

Funct=100000时，ALUControl=010，执行add；

Funct=100010时，ALUControl=110，执行sub；

Funct=100100时，ALUControl=000，执行and；

Funct=100101时，ALUControl=001，执行or；

Funct=101010时，ALUControl=111，执行slt。

50-60ns，Op=100011，RegWrite=1，ALUSrc=1，MemToReg=1，执行lw。

60-70ns，Op=101011，ALUSrc=1，MemWrite=1，执行sw。

70-80ns，Op=000100，执行beq。

80-90ns，Op=001000，RegWrite=1，ALUSrc=1，执行addi。

90ns后，Op=000010，Jump=1，执行J型指令。

**五、调试和心得体会**

通过本次实验，我了解了指令译码的实现原理以及MIPS体系结构的部分指令，同时也学到了CPU执行指令的基本过程，为下次实验设计CPU做好了准备。