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**

**Controller模块**

**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 Op = ((Op==0)&&(Funct==0))?6'b111111:Op;**

**assign PCSrc = Branch & Zero;**

**endmodule**

**MainDec模块**

**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;//RTYPE**

**6'b100011:Controls<=9'b101001000;//LW**

**6'b101011:Controls<=9'b001010000;//SW**

**6'b000100:Controls<=9'b000100001;//BEQ**

**6'b001000:Controls<=9'b101000000;//ADDI**

**6'b000010:Controls<=9'b000000100;//J**

**6'b111111:Controls<=9'b000000000;//nop**

**default:Controls<=9'bxxxxxxxxx;//illegal Op**

**endcase**

**endmodule**

**ALUDec模块**

**module ALUDec(**

**input [5:0] Funct,**

**input [1:0] ALUOp,**

**output reg [2:0] ALUControl**

**);**

**always@(\*)**

**case(ALUOp)**

**2'b00:ALUControl <= 3'b010;//add (for lw/sw/addi)**

**2'b01:ALUControl <= 3'b110;//sub (for beg)**

**default:case(Funct) //R-type Instruction**

**6'b100000:ALUControl <= 3'b010;//add**

**6'b100010:ALUControl <= 3'b110;//aub**

**6'b100100:ALUControl <= 3'b000;//and**

**6'b100101:ALUControl <= 3'b001;//or**

**6'b101010:ALUControl <= 3'b111;//slt**

**default:ALUControl <= 3'bxxx;//???**

**endcase**

**endcase**

**endmodule**

**仿真文件**

**module sim\_control();**

**reg [5:0] Op,Funct;**

**reg Zero;**

**wire MemtoReg,MemWrite;**

**wire PCSrc,ALUSrc;**

**wire RegDst,RegWrite;**

**wire Jump;**

**wire [2:0] ALUControl;**

**Controller a(Op,Funct,Zero,MemtoReg,MemWrite,PCSrc,ALUSrc,RegDst,RegWrite,Jump,ALUControl);**

**initial begin**

**Op=6'b000000;Funct=6'b100000;Zero=1;**

**repeat(1)#1 begin Op=6'b100011;Funct=6'b100010;end**

**repeat(1)#1 Op=6'b101011;**

**repeat(1)#1 Op=6'b000100;**

**repeat(1)#1 Op=6'b001000;**

**repeat(1)#1 Op=6'b000010;**

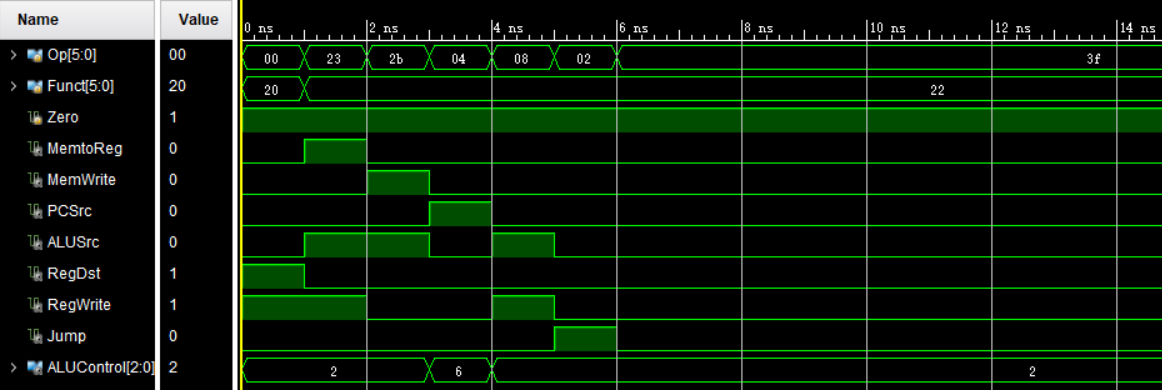
**repeat(1)#1 Op=6'b111111;**

**$finish;**

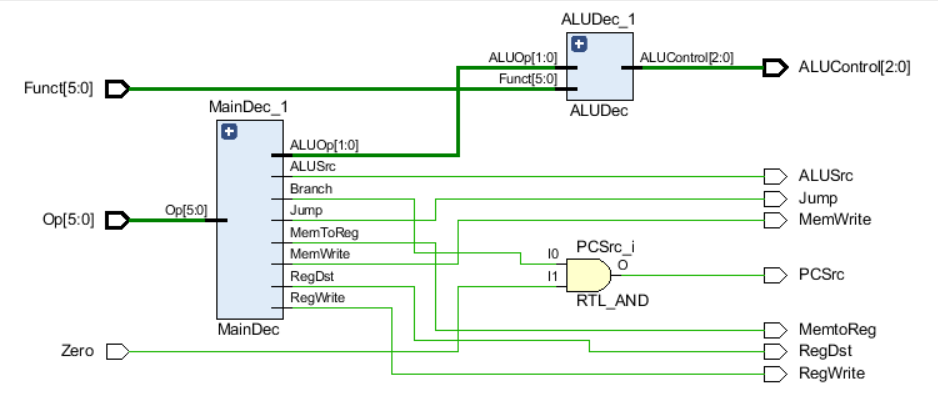
**end**

**endmodule**

**仿真图像：**



**电路图：**



**分析：**

**Controller 模块**

Controller 模块是顶层模块，整合了两个子模块，即 MainDec 和 ALUDec。它接收操作码 (Op)、功能码 (Funct) 和零标志 (Zero) 作为输入，并生成用于处理器的各种控制信号。以下是简要分析：

输入:

Op, Funct: 6位的操作码和功能码。

Zero: 用于条件分支判断的信号。

输出:

MemtoReg, MemWrite: 存储器到寄存器写入和写使能信号。

PCSrc, ALUSrc: 程序计数器选择和 ALU 选择信号。

RegDst, RegWrite: 寄存器目的寄存器选择和写使能信号。

Jump: 跳转信号。

ALUControl: ALU 控制信号。

中间信号:

ALUOp: 由 MainDec 模块输出，用于 ALU 控制。

Branch: 由 MainDec 模块输出，用于分支判断。

引用模块：

MainDec：负责根据操作码解码主控制信号。

ALUDec：负责根据功能码解码 ALU 控制信号。

**MainDec 模块**

MainDec 模块负责解码操作码并生成各种主控制信号。

输入：

Op：6 位操作码。

输出：

MemToReg：存储器到寄存器写入控制信号。

MemWrite：存储器写使能控制信号。

Branch：分支控制信号。

ALUSrc：ALU 源选择控制信号。

RegDst：寄存器目的寄存器选择控制信号。

RegWrite：寄存器写使能控制信号。

Jump：跳转控制信号。

ALUOp：2 位 ALU 操作控制信号。

内部寄存器：

Controls：9 位寄存器，存储控制信号。

行为：

根据输入的操作码对应的操作，将相应的控制信号分配给 Controls 寄存器。

**ALUDec 模块：**

ALUDec 模块根据 ALU 操作码和功能码解码 ALU 控制信号。

输入：

Funct：6 位功能码。

ALUOp：2 位 ALU 操作码。

输出：

ALUControl：3 位 ALU 控制信号。

行为：根据 ALUOp 和 Funct 的组合，解码 ALU 控制信号。

**仿真 (sim\_control) 模块**：

输入：

Op、Funct、Zero：输入给 Controller 模块的信号。

输出：

从 Controller 模块输出的各种信号。

行为：

初始化输入信号。

在仿真中改变输入信号，以测试不同的场景。

在执行各种指令后，仿真将结束。

**五、调试和心得体会**

我在代码中采用了模块化设计，将功能划分为多个模块。这使得代码更易于理解、调试和维护。每个模块都有清晰的输入和输出，提高了代码的可读性和可维护性。仿真是验证硬件描述语言代码功能的重要步骤。通过在仿真中模拟各种输入情况，我可以验证代码是否按预期工作。在仿真中调试有助于查找和解决潜在的问题。对于每个模块，我会编写针对该模块的单元测试，以确保其独立工作。这有助于定位问题的根本原因，并提高系统整体的可靠性。通过查看仿真波形图，我弄明白了信号的变化和模块之间的时序关系。时序图是检查时序问题的有力工具