实验五 信号传输、指令译码的设计与调试

**一、实验目的**

1. 掌握Verilog中语句的综合应用。
2. 进一步掌握Verilog中的时序电路设计。

**二、实验内容**

1. 设计一个寄存器组，10\*32bit。
2. 连接寄存器组与RAM模块。
3. 设计一个指令译码器。
4. 满足功能上的需求，包括运算、转移、存储指令。

**三、实验要求**

1. 分析各模块的的程序结构，画出其流程图。
2. 画出模块的电路图。
3. 分析电路的仿真波形，标出关键的数值。
4. 记录设计和调试过程。

**四、实验代码及结果**

Main decoder

ALUDecoder

Alu

Register

RAM

Maindec：

module maindec(input [5:0] op,

output regwrite,regdst,alusrc,branch,memwrite,memtoreg,jump,

output [1:0] aluop);

reg [8:0]controls;

assign {

regwrite,

regdst,

alusrc,

branch,

memwrite,

memtoreg,

jump,

aluop

}=controls;

always @(\*)

begin

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

default: controls <=9'bxxxxxxxxx;

endcase

end

endmodule

Register:

module Register(input clk,

input we3,

input [4:0] ra1,ra2,wa3,

input [31:0] wd3,

output [31:0] rd1,rd2);

reg [31:0] rf[31:0];

always @ (posedge clk)

if (we3) rf[wa3] <=wd3;

assign rd1=(ra1!=0)?rf[ra1]:0;

assign rd2=(ra2!=0)?rf[ra2]:0;

endmodule

Aludec:

module aludec(input [5:0] funct,

input [1:0] aluop,

output reg [2:0] alucontrol);

always @ (\*)

case (aluop)

2'b00: alucontrol <=3'b010;//add

2'b01: alucontrol <=3'b110;//sub

default:case(funct)

6'b100000:alucontrol <=3'b010;//ad

6'b100010:alucontrol <=3'b110;//sub

6'b100100:alucontrol <=3'b000;//and

6'b100101:alucontrol <=3'b001;//or

6'b101010:alucontrol <=3'b100;

endcase

endcase

endmodule

ALU:

module alu(

ain,bin,cout,choose,zero

);

input [31:0] ain,bin;

input [2:0]choose;

output reg [31:0] cout;

output reg zero;

always@(\*)

begin

case(choose)

3'b010:

begin

cout=ain+bin;

end

3'b110:

begin

cout=ain-bin;

end

3'b000:cout=ain&bin;

3'b001:cout=ain|bin;

3'b100:

begin

if(ain<bin)cout=32'b1;

else cout=32'h00000000;

end

endcase

if(cout!=0) zero =0;

else zero=1;

end

endmodule

RAM:

module ram(

input clk,

input we,

input [31:0]addr,

input [31:0]datain,

output [31:0]dataout

);

reg [31:0]RAM[0:31];

assign dataout=RAM[addr];

always@(posedge clk)

begin

if(we) RAM[addr]=datain;

end

endmodule