实验4 简单计算机系统-系统设计B-代码

电 25 吴晨聪 2022010311

# 1. cpuD.v

module cpuD(memData,clk,rst\_n,s,zeroout);

wire [7:0]scrA;

wire [7:0]scrB;

wire [1:0]Rs;

wire [1:0]Rt;

wire [1:0]Rd;

wire [3:0]Op;

input [7:0]memData;

wire selscrB;

wire redges;

wire memtoreg;

wire [7:0]imm;

wire regwrite;

wire [2:0]alucs;

input clk;

input rst\_n;

wire flagwrite;

wire [1:0]nd;

wire [7:0]di;

wire [7:0]q2;

wire carry\_out;

wire carry\_in;

wire zeroin;

wire wren;

output [7:0]s;

output zeroout;

always@(posedge clk or negedge rst\_n) begin

if(!rst\_n) begin

end

else begin

end

end

mux2 muxscrB(

.d0(q2),

.d1(imm),

.sel(selscrB),

.y(scrB)

);

mux1 muxnd(

.d0(Rt),

.d1(Rd),

.sel(redges),

.y(nd)

);

mux2 muxdi(

.d0(s),

.d1(memData),

.sel(memtoreg),

.y(di)

);

regfile regfile(

.clk(clk),

.rst\_n(rst\_n),

.n1(Rs),

.n2(Rt),

.nd(nd),

.di(di),

.reg\_we(regwrite),

.q1(scrA),

.q2(q2)

);

alu alu(

.data\_a(scrA),

.data\_b(scrB),

.s(s),

.zero(zeroin),

.cs(alucs),

.carry\_in(carry\_in),

.carry\_out(carry\_out)

);

flag flag(

.clk(clk),

.rst\_n(rst\_n),

.zeroin(zeroin),

.flagwrite(flagwrite),

.flagin(carry\_out),

.flagout(carry\_in),

.zeroout(zeroout)

);

ROM ROM(

.clk(clk),

.rst\_n(rst\_n),

.Op(Op),

.Rs(Rs),

.Rt(Rt),

.Rd(Rd),

.imm(imm)

);

controller controller(

.Op(Op),

.alucs(alucs),

.flagwrite(flagwrite),

.regwrite(regwrite),

.selscrB(selscrB),

.redges(redges),

.memtoreg(memtoreg),

.wren(wren)

);

endmodule

# 2. cpuD\_tb.v

` timescale 1ns/1ps

module cpuD\_tb;

reg clk;

reg rst\_n;

reg [7:0]memData;

wire [7:0]s;

wire zeroout;

initial begin

memData=0;

clk=1;

rst\_n=0;

#20 rst\_n=1;

end

always #10 clk = ~clk;

cpuD cpuD(

.memData(memData),

.clk(clk),

.rst\_n(rst\_n),

.s(s),

.zeroout(zeroout)

);

endmodule

# 3. cpuE.v

module cpuE(aclr,clk,rst\_n,s,zeroout);

input aclr;

wire [7:0]scrA;

wire [7:0]scrB;

wire [1:0]Rs;

wire [1:0]Rt;

wire [1:0]Rd;

wire [3:0]Op;

wire [7:0]memData;

wire selscrB;

wire redges;

wire memtoreg;

wire [7:0]imm;

wire regwrite;

wire [2:0]alucs;

input clk;

input rst\_n;

wire flagwrite;

wire [1:0]nd;

wire [7:0]di;

wire [7:0]q2;

wire carry\_out;

wire carry\_in;

wire zeroin;

wire wren;

output [7:0]s;

output zeroout;

always@(posedge clk or negedge rst\_n) begin

if(!rst\_n) begin

end

else begin

end

end

mux2 muxscrB(

.d0(q2),

.d1(imm),

.sel(selscrB),

.y(scrB)

);

mux1 muxnd(

.d0(Rt),

.d1(Rd),

.sel(redges),

.y(nd)

);

mux2 muxdi(

.d0(s),

.d1(memData),

.sel(memtoreg),

.y(di)

);

regfile regfile(

.clk(clk),

.rst\_n(rst\_n),

.n1(Rs),

.n2(Rt),

.nd(nd),

.di(di),

.reg\_we(regwrite),

.q1(scrA),

.q2(q2)

);

alu alu(

.data\_a(scrA),

.data\_b(scrB),

.s(s),

.zero(zeroin),

.cs(alucs),

.carry\_in(carry\_in),

.carry\_out(carry\_out)

);

flag flag(

.clk(clk),

.rst\_n(rst\_n),

.zeroin(zeroin),

.flagwrite(flagwrite),

.flagin(carry\_out),

.flagout(carry\_in),

.zeroout(zeroout)

);

ROM ROM(

.clk(clk),

.rst\_n(rst\_n),

.Op(Op),

.Rs(Rs),

.Rt(Rt),

.Rd(Rd),

.imm(imm)

);

controller controller(

.Op(Op),

.alucs(alucs),

.flagwrite(flagwrite),

.regwrite(regwrite),

.selscrB(selscrB),

.redges(redges),

.memtoreg(memtoreg),

.wren(wren)

);

cpuram cpu\_ram\_inst(

.aclr(aclr),

.wren(wren),

.address(s),

.data(q2),

.clock(clk),

.q(memData)

);

endmodule

# 4. cpuE\_tb.v

`timescale 1ns/1ps

module cpuE\_tb;

reg clk;

reg rst\_n;

reg aclr;

wire [7:0]s;

wire zeroout;

initial begin

aclr=0;

clk=1;

rst\_n=0;

#20 rst\_n=1;

end

always #10 clk = ~clk;

cpuE cpuE(

.aclr(aclr),

.clk(clk),

.rst\_n(rst\_n),

.s(s),

.zeroout(zeroout)

);

endmodule