Alex Foor

Compen 331

Final Project

1. Verilog design code

`timescale 1ns / 1ps

module regfile (rna,rnb,d,wn,we,clk,clrn,qa,qb); // 32x32 regfile

input [31:0] d; // data of write port

input [4:0] rna; // reg # of read port A

input [4:0] rnb; // reg # of read port B

input [4:0] wn; // reg # of write port

input we; // write enable

input clk, clrn; // clock and reset

output [31:0] qa, qb; // read ports A and B

reg [31:0] register [1:31]; // 31 32-bit registers

assign qa = (rna == 0)? 0 : register[rna]; // read port A

assign qb = (rnb == 0)? 0 : register[rnb]; // read port B

integer i;

initial begin

register[4'd1] = 32'hA00000AA;

register[4'd2] = 32'h10000011;

register[4'd3] = 32'h20000022;

register[4'd4] = 32'h30000033;

register[4'd5] = 32'h40000044;

register[4'd6] = 32'h50000055;

register[4'd7] = 32'h60000066;

register[4'd8] = 32'h00000002;

register[4'd9] = 32'h80000088;

register[4'd10] = 32'h90000099;

end

always @(posedge clk or negedge clrn) // write port

if (!clrn)

for (i = 1; i < 32; i = i + 1)

register[i] <= 32'd0; // reset

else

if ((wn != 0) && we) // not reg[0] & enabled

register[wn] <= d; // write d to reg[wn]

endmodule

module Instmem(a,op,func,rs,rt,rd,sa, address, offset);

input [31:0] a;

reg [31:0] instructions [0:31];

output [5:0] op, func;

output [4:0] rs,rt,rd,sa;

output [25:0] address;

output [15:0] offset;

assign op = instructions[a >> 32'd2] >> 32'd26;

assign func = (instructions[a >> 32'd2] << 32'd26) >> 32'd26;

assign rs = (instructions[a >> 32'd2] << 32'd6) >> 32'd27;

assign rt = (instructions[a >> 32'd2] << 32'd11) >> 32'd27;

assign rd = (instructions[a >> 32'd2] << 32'd16) >> 32'd27;

assign sa = (instructions[a >> 32'd2] << 32'd21) >> 32'd27;

assign address = (instructions[a >> 32'd2] << 32'd6) >> 32'd6;

assign offset = (instructions[a >> 32'd2] << 32'd16) >> 32'd16;

initial begin

instructions[0] = 32'h3c010000;

instructions[1] = 32'h34240050;

instructions[2] = 32'h20050004;

instructions[3] = 32'h0c000018;

instructions[4] = 32'hac820000;

instructions[5] = 32'h8c890000;

instructions[6] = 32'h01244022;

instructions[7] = 32'h20050003;

instructions[8] = 32'h20a5ffff;

instructions[9] = 32'h34a8ffff;

instructions[10] = 32'h39085555;

instructions[11] = 32'h2009ffff;

instructions[12] = 32'h312affff;

instructions[13] = 32'h01493025;

instructions[14] = 32'h01494026;

instructions[15] = 32'h01463824;

instructions[16] = 32'h10a00001;

instructions[17] = 32'h08000008;

instructions[18] = 32'h2005ffff;

instructions[19] = 32'h000543c0;

instructions[20] = 32'h00084400;

instructions[21] = 32'h00084403;

instructions[22] = 32'h000843c2;

instructions[23] = 32'h08000017;

instructions[24] = 32'h00004020;

instructions[25] = 32'h8c890000;

instructions[26] = 32'h20840004;

instructions[27] = 32'h01094020;

instructions[28] = 32'h20a5ffff;

instructions[29] = 32'h14a0fffb;

instructions[30] = 32'h00081000;

instructions[31] = 32'h03e00008;

end

endmodule

module four\_to\_one(p4, branchaddr, regaddr, jumpaddr, pcsrc, new\_pc);

input [31:0] p4;

input [31:0] branchaddr;

input [31:0] regaddr;

input [25:0] jumpaddr;

input [1:0] pcsrc;

output reg [31:0] new\_pc;

always @(\*) begin

if (pcsrc == 2'b00)

new\_pc <= p4;

else if(pcsrc == 2'b01)

new\_pc <= p4 + (branchaddr<<2);

else if(pcsrc == 2'b10)

new\_pc <= regaddr;

else

new\_pc <= ((p4>>5'd28)<<5'd28) + (jumpaddr<<2'b10);

end

endmodule

module program\_counter

(

input clk,

input rst,

input [31:0] new\_pc,

output reg [31:0] pc,

output [31:0] p4

);

assign p4 = pc + 5'd4;

initial begin

pc <= 0;

end

always @(posedge clk or posedge rst)

begin

if (rst)

pc <= 0;

else

pc <= new\_pc;

end

endmodule

module ALU(input [31:0] a, input [31:0] b, input [3:0] aluc,output reg [31:0] r, output reg z);

always @(\*)

begin

if(aluc == 0) begin

r = a + b;

z = 0;

end

if(aluc == 4'b0001) begin

r = a & b;

z = 0;

end

if(aluc == 4'b0100) begin

r= a - b;

z = 0;

end

if(aluc == 4'b0101) begin

r = a | b;

z = 0;

end

if(aluc == 4'b0010) begin

r = a ^ b;

z = (a == b)? 1 : 0;

end

if(aluc == 4'b0110) begin

r = a << 6'd16;

z = 0;

end

if(aluc == 4'b0011) begin

r = b << a;

z = 0;

end

if(aluc == 4'b0111) begin

r = b >> a;

z = 0;

end

if(aluc == 4'b1111) begin

r = $signed(b) >> a;

z = 0;

end

end

endmodule

module aluamux(qa,sa,shift,alua);

input [31:0] qa;

input [4:0] sa;

input shift;

output [31:0] alua;

assign alua = (shift == 0)? qa : sa;

endmodule

module alubmux(qa,sa,shift,alua);

input [31:0] qa;

input [31:0] sa;

input shift;

output [31:0] alua;

assign alua = (shift == 0)? qa : sa;

endmodule

module wnmux(qa,sa,regrt,wn);

input [4:0] qa;

input [4:0] sa;

input regrt;

output [4:0] wn;

assign wn = (regrt == 1)? qa : sa;

endmodule

module extend(im, sext, e);

input [15:0] im;

input sext;

output [31:0] e;

assign e = (sext == 0)? im : (im > 16'h0fff)? (32'hffff0000 + im) : im;

endmodule

module datafile(a,di,we,do, clk);

input [31:0] a,di;

input we;

output [31:0] do;

input clk;

reg [31:0] data [0:31];

assign do = data[a >> 4'd2];

always @(posedge clk) // write

if (we)begin

data[a >> 4'd2 ] = di;

end

endmodule

module control\_unit(

input[5:0] op,func, input z,

output reg mtoreg,

output reg [1:0] pcsrc,

output reg wmem,

output reg [3:0] aluc,

output reg shift,

aluimm,

regrt,

jal,

sext,

wreg);

always @(\*)

begin

wmem = 0;

if (op == 0) begin //r type

regrt = 0;

jal = 0;

aluimm = 0;

mtoreg = 0;

if(func == 5'd8)begin //jr instruction

pcsrc = 2'b10;

wreg = 0;

end

else begin

pcsrc <= 2'b00;

wreg <= 1;

if(func == 6'd32)//add

aluc <= 0;

else if(func == 6'd34)//sub

aluc = 4'b0100;

else if(func == 6'd36)//and

aluc = 4'b0001;

else if(func == 6'd37)//or

aluc = 4'b0101;

else if(func == 6'd38)//xor

aluc = 4'b0010;

else if(func == 0)//sll

aluc = 4'b0011;

else if(func == 6'd2)//srl

aluc = 4'b0111;

else //sra

aluc = 4'b1111;

end

end

else if (op >= 6'b001000 && op <= 6'b001111) //i type

begin

pcsrc = 0;

wreg = 1;

regrt = 1;

jal = 0;

mtoreg = 0;

shift = 0;

aluimm = 1;

if(op == 6'b001000) begin //addi

aluc = 0;

sext = 1;

end

if(op == 6'b001100) begin //andi

aluc = 4'b0001;

sext = 0;

end

if(op == 6'b001101) begin //ori

aluc = 4'b0101;

sext = 0;

end

if(op == 6'b001110) begin //xori

aluc = 4'b0010;

sext = 0;

end

if(op == 6'b001111) begin //lui

aluc = 4'b0110;

sext = 0;

end

end

else if(op == 6'b100011) begin //lw instruction

wreg = 1;

regrt = 1;

jal = 0;

mtoreg = 1;

shift = 0;

aluimm = 1;

sext = 1;

aluc = 0;

pcsrc = 0;

end

else if(op == 6'b101011) begin //sw instruction

wmem = 1;

wreg = 0;

shift = 0;

aluimm = 1;

sext = 1;

aluc = 0;

pcsrc = 0;

end

else if (op == 6'b000100) begin //beq

wreg = 0;

shift = 0;

aluimm = 0;

sext = 1;

aluc = 4'b0010;

if(z) begin

pcsrc = 2'd1;

end

else begin

pcsrc = 0;

end

end

else if(op == 6'b000101) begin //bne

wreg = 0;

shift = 0;

aluimm = 0;

sext = 1;

aluc = 4'b0010;

if(z) begin

pcsrc = 0;

end

else begin

pcsrc = 2'd1;

end

end

else if(op == 6'b000010) begin //j instruction

wreg = 0;

pcsrc = 2'b11;

end

else if(op == 6'b000011) begin //jal instruction

wreg = 1;

jal = 1;

pcsrc = 2'b11;

end

else if(op == 0) begin //jr

wreg = 0;

pcsrc = 2'b10;

end

end

endmodule;

1. Test bench code

`timescale 1ns / 1ps

module testbench;

wire [4:0] rna,rnb,wn,sa,rt;

reg clk,clrn;

wire z;

wire [1:0] pcsrc;

wire [3:0] aluc;

wire [31:0] qa,qb,pc,p4,new\_pc,d,alua,alub,r,do;

wire [5:0] op, func;

wire [25:0] jumpaddr;

wire [31:0] regaddr, immext;

wire [15:0] imm;

wire mtoreg,wmem,shift,aluimm,regrt,jal,sext,wreg;

regfile rf (rna,rnb,(jal==1)? p4: d,(jal==1)? 31: wn,wreg,clk,clrn,qa,qb);

control\_unit cu(op,func,z,mtoreg,pcsrc,wmem,aluc,shift,aluimm,regrt,jal,sext,wreg);

Instmem im(pc,op,func,rna,rnb,rt,sa, jumpaddr, imm);

ALU al(alua,alub,aluc,r,z);

aluamux mx(qa,sa,shift,alua);

alubmux mx2(qb,immext,aluimm,alub);

wnmux wmux(rnb,rt,regrt,wn);

alubmux mx4(r,do,mtoreg,d);

program\_counter progc(clk,0, new\_pc,pc,p4);

four\_to\_one pc\_mux(p4,immext,qa,jumpaddr,pcsrc,new\_pc);

extend ext(imm,sext,immext);

datafile data(r,qb,wmem,do, clk);

initial begin

clk = 0;

clrn = 1;

end

always #10 clk = !clk;

endmodule

1. Waveforms

A screenshot of a computer screen

Description automatically generated with low confidence

A picture containing screenshot, electronics, circuit, synthesizer

Description automatically generated

A picture containing screenshot, electronics, circuit

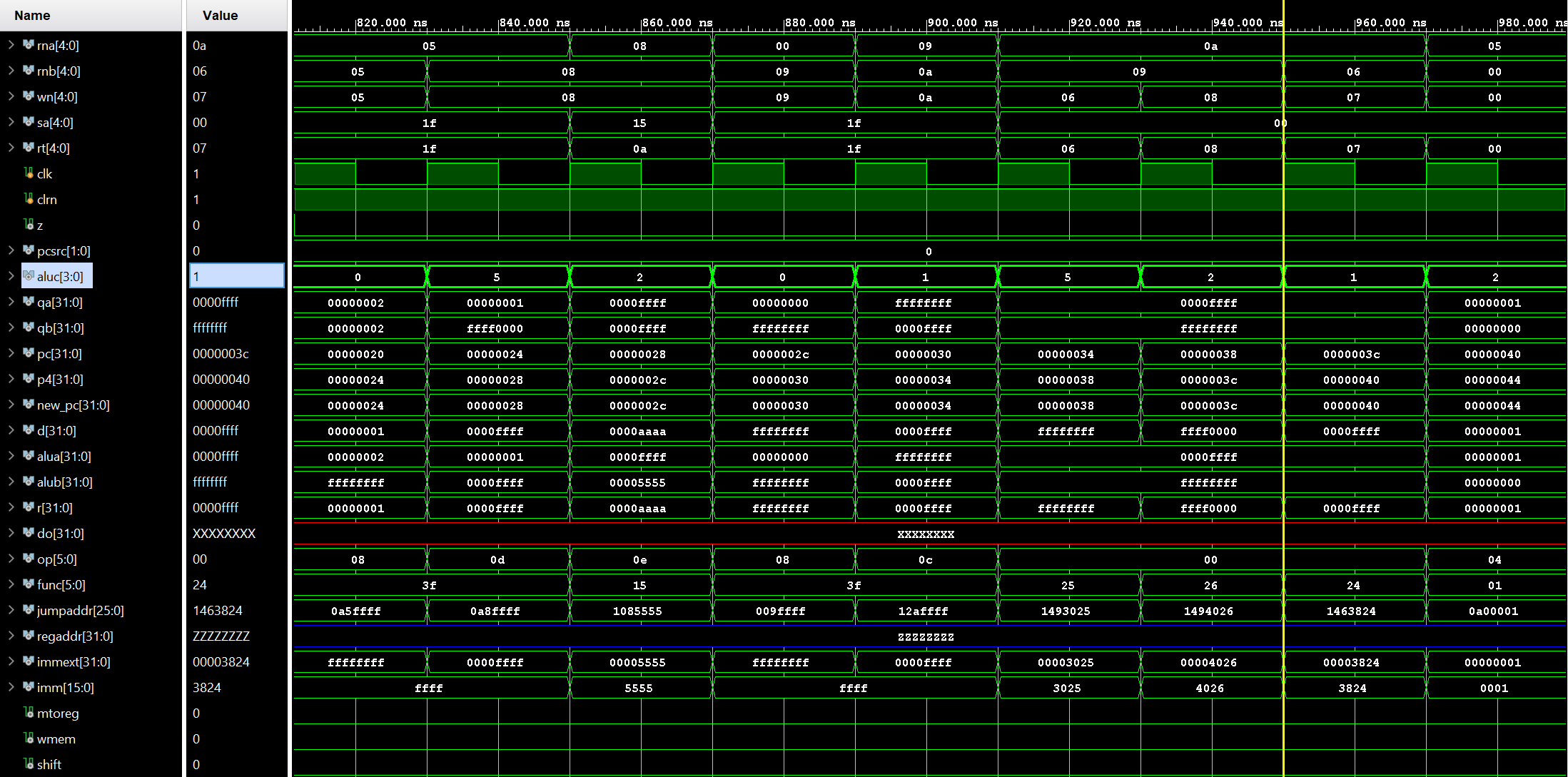
Description automatically generated

A picture containing screenshot, electronics, circuit, synthesizer

Description automatically generated

A picture containing electronics, screenshot, electronic engineering, circuit

Description automatically generated



A picture containing screenshot, circuit

Description automatically generated

A picture containing screenshot, electronics, circuit

Description automatically generated

1. Schematic

A picture containing text, diagram, plan, line

Description automatically generated

1. i/o planning

A screenshot of a game

Description automatically generated

1. floor planning

A screen shot of a computer

Description automatically generated with medium confidence