CMPEN331--002 LAB4

Yiming Ren

[Yzr12@psu.edu](mailto:Yzr12@psu.edu)

Source file:

`timescale 1ns / 1ps

module pcadder(

input[31:0] initialPC,

input clk, rst,

output reg[31:0] PC);

reg[31:0] a;

always@(posedge clk or posedge rst) begin

if(rst) begin

a=initialPC;

PC=a;

end

else begin

a=a+4;

PC=a;

end

end

endmodule

module inst\_mem(

input[31:0] inst,

input clk,

output reg[31:0] IFID);

always@(posedge clk)begin

IFID=inst;

end

endmodule

module IFIDoutput(

input clk,

input [31:0] IFID,

output reg [5:0] op, func,

output reg [4:0] rs,rt,rd,

output reg [15:0] imm);

always@(posedge clk) begin

op=IFID[31:26];

rs=IFID[25:21];

rt=IFID[20:16];

imm=IFID[15:0];

end

endmodule

module ctrl\_unit(

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

output reg wreg, m2reg, wmem, aluimm, regrt,

output reg[3:0] aluc);

always@(op or posedge clk) begin

case(op)

6'b100011: begin

aluc=4'b0010;

aluimm=1;

wreg=1;

m2reg=1;

wmem=0;

regrt=1;

end

default: begin

end

endcase

end

endmodule

module mux01(

input[4:0] rd, rt,

input regrt,clk,

output reg[4:0] regwrt);

always@(regrt or posedge clk or rd or rt) begin

if(regrt ==0) begin

regwrt=rd;

end

else begin

regwrt=rt;

end

end

endmodule

module reg\_file(

input[31:0] register1 , register2,

input [4:0] rs, rt,

input clk,

output reg[31:0] qa, qb);

always@(posedge clk or rs or rt ) begin

qa=register1;

qb=register2;

end

endmodule

module extender(

input[15:0] imm,

input clk,

output reg[31:0] extend);

always@(posedge clk or imm)begin

extend=imm+0;

end

endmodule

module IDEXEoutput(

input clk,

input wreg, w2reg, wmem,

input[3:0] aluc,

output reg ewreg, ew2reg, ewmem,

output reg [3:0] ealuc);

always@(posedge clk) begin

ewreg = wreg;

ew2reg = w2reg;

ewmem = wmem;

ealuc=aluc;

end

endmodule

module mux02(

input ealuim, clk,

input [31:0] qb,

input [31:0] extend,

output reg[31:0] alub);

always@(posedge clk) begin

case(ealuim)

0: begin

alub=qb;

end

1: begin

alub=extend;

end

endcase

end

endmodule

module ALU(

input clk,

input[3:0] ealuc,

input[31:0] alua, alub,

output reg[31:0] alur);

always@(alub ) begin

case(ealuc)

4'b0010:begin

alur=alua+alub;

end

default: begin

end

endcase

end

endmodule

module EXEMEMreg(

input ewreg, em2reg, ewmem, clk,

input [4:0] registerwrt,

input [31:0] alur, qb,

output reg[71:0] EXEMEM);

always@(ewreg or alur or posedge clk) begin

EXEMEM[71] =ewreg;

EXEMEM[70] =em2reg;

EXEMEM[69] =ewmem;

EXEMEM[68:64] =registerwrt;

EXEMEM[63:32] =alur;

EXEMEM[31:0] = qb;

end

endmodule

module EXEMEMoutput(

input clk,

input [71:0] EXEMEM,

output reg mwreg, mw2reg);

always@(posedge clk) begin

mwreg = EXEMEM [71];

mw2reg = EXEMEM [70];

end

endmodule

module memory(

input clk,mwmem,

input [31:0] memory,

output reg[31:0] do);

always@(posedge clk) begin

case(mwmem)

0: begin

do=memory;

end

default:begin

end

endcase

end

endmodule

module MEMWBreg(

input mwreg, mm2reg, clk,

input [4:0] registerwrt,

input [31:0] alur, do,

output reg[70:0] MEMWB);

always@(mwreg or posedge clk) begin

MEMWB[70] = mwreg;

MEMWB[69] = mm2reg;

MEMWB[68:64] =registerwrt;

MEMWB[63:32] =alur;

MEMWB[31:0] = do;

end

endmodule

Testbench

`timescale 1ns / 1ps

module testbench();

reg clk,rst;

reg[31:0] initialPC=100;

reg [31:0] instruction [112:100];

reg [31:0] register [31:0];

reg[31:0] memory [9:0];

initial begin

register[0] =0 ;

register[1] =0 ;

register[2] =0 ;

register[3] =0 ;

register[4] =0 ;

register[5] =0 ;

instruction[100] = 32'b10001100001000100000000000000000;

instruction[104] = 32'b10001100001000110000000000000100;

instruction[108] = 32'b10001100001001000000000000001000;

instruction[112] = 32'b10001100001001010000000000001100;

memory[0] = 32'hA00000AA;

memory[1] = 32'h10000011;

memory[2] = 32'h20000022;

memory[3] = 32'h30000033;

memory[4] = 32'h40000044;

memory[5] = 32'h50000055;

memory[6] = 32'h60000066;

memory[7] = 32'h70000077;

memory[8] = 32'h80000088;

memory[9] = 32'h90000099;

clk=0;

rst=1;

#1 rst=0;

#100 clk=1;

#100 clk=0;

#100 clk=1;

#100 clk=0;

#100 clk=1;

#100 clk=0;

#100 clk=1;

#100 clk=0;

end

wire wreg, m2reg, aluimm, wmem, regrt, ewreg, em2reg, ewmem, mwreg, mm2reg;

wire[3:0] aluc, ealuc;

wire[4:0] registerwrt, rs, rt, rd;

wire[5:0] op, func;

wire[15:0] imm;

wire[31:0] IFID, pc, qa, qb, extend, alub, alur, do;

wire[70:0] MEMWB;

wire[71:0] EXEMEM;

pcadder adder(initialPC, clk, rst, pc);

inst\_mem im( instruction[pc], clk, IFID);

IFIDoutput IFIDout( clk, IFID, op, func, rs, rt, rd, imm);

ctrl\_unit CU(op, func, clk, wreg, m2reg, wmem, aluimm, regrt, aluc);

mux01 m1(rd, rt, regrt, clk, registerwrt);

reg\_file RF( register[rs], register[rt], clk, rs, rt, qa, qb);

extender ext( imm, clk, extend);

IDEXEoutput IDEXEout(clk, wreg, m2reg, wmem ,aluc, ewreg, em2reg, ewmem, ealuc);

mux02 m2(aluimm, clk, qb, extend, alub);

ALU alu(clk, ealuc, qa, alub, alur);

EXEMEMreg EXEMEMregister( ewreg, em2reg, ewmem, clk, registerwrt, alur, qb, EXEMEM);

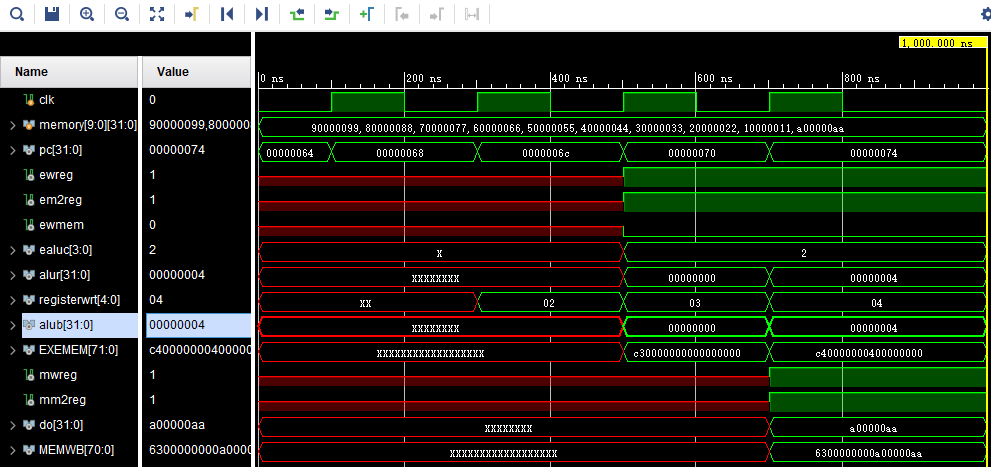
EXEMEMoutput EXEMEMout(clk, EXEMEM, mwreg, mm2reg);

memory data\_memory( clk, EXEMEM[69], memory[qa], do);

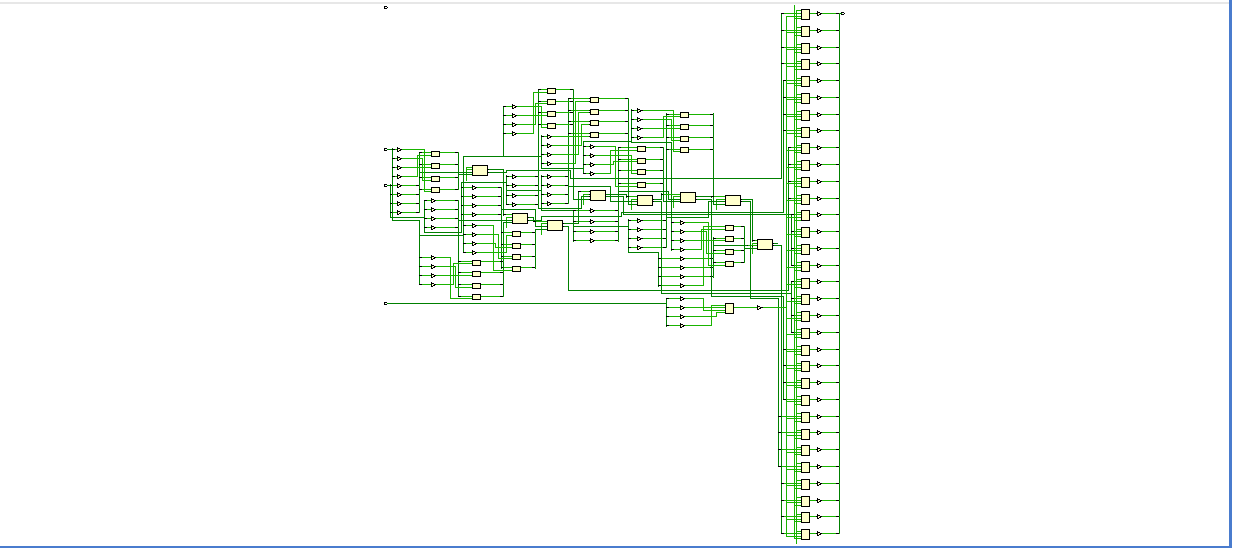
MEMWBreg MEMWBregister( mwreg, mm2reg, clk, EXEMEM[68:64], alur, do, MEMWB);

Endmodule

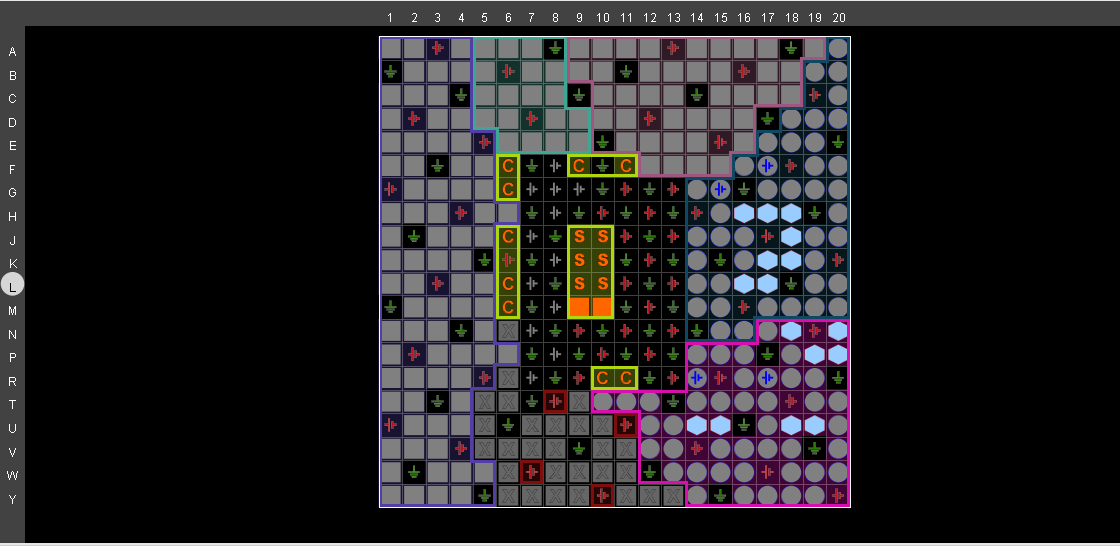
Waveform



Schematic



I/O planning

  
Floor planning

