CMPEN 331 Lab3 Report – Yubo Jing

# Verilog code:

## IFID\_IDEXE\_Module.v

`timescale 1ns / 1ps

module IFID\_IDEXE\_Module(

input wire clock,

output wire [31:0] pc,dinstOut,eqa, eqb, eimm32,

output wire eRegWrite, eMemtoReg,eMemWrite,eALUsrc,

output wire [3:0] eALUop,

output wire [4:0] edestReg

);

wire [31:0] nextPc;

wire [31:0] instOut;

wire RegWrite, MemtoReg, MemWrite, ALUsrc,regDst;

wire [3:0] ALUop;

wire [4:0] destReg;

wire [31:0] qa,qb,imm32;

wire [5:0] op, func;

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

wire [15:0] imm;

PC PC\_tb(nextPc,clock,pc);

IM IM\_tb(pc, instOut);

Pc\_Adder Pc\_Adder\_tb(pc,nextPc);

IFID\_Pipeline\_Register IFID\_Pipeline\_Register\_tb(clock, instOut, dinstOut, op, func, rd, rs, rt, imm);

Control\_Unit Control\_Unit\_tb(op,func,RegWrite,MemtoReg,MemWrite,ALUop,ALUsrc,regDst);

regDst\_mux regDst\_mux\_tb(rt,rd,regDst,destReg);

Register\_File Register\_File\_tb(rs,rt,qa,qb);

imm\_Extender imm\_Extender\_tb(imm,imm32);

IDEXE\_Pipeline\_Register IDEXE\_Pipeline\_Register\_tb(clock,RegWrite, MemtoReg, MemWrite, ALUop, ALUsrc,destReg,qa,qb,imm32,eRegWrite,eMemtoReg,eMemWrite,eALUop,eALUsrc, edestReg, eqa, eqb,eimm32);

endmodule

## PC.v

`timescale 1ns / 1ps

module PC(

input[31:0] nextPc,

input clock,

output reg[31:0] pc

);

initial begin

pc = 100;

end

always@(posedge clock) begin

pc = nextPc;

end

endmodule

## IM.v

`timescale 1ns / 1ps

module IM(

input[31:0] pc,

output reg[31:0] instOut

);

reg [31:0] memory [0:63];

initial begin

// memory[25] = { //lw I-type

// 6'b100011, //lw op

// 5'b00010, //rs -> $v0 -> 2

// 5'b00001, //rt -> $at -> 1

// 16'b0000000000000000 //offset = 0

// };

memory[25] = 32'h8C220000;

memory[26] = 32'h8C230004;

// memory[26] = { //lw I-type

// 6'b100011, //lw op

// 5'b00011, //rs -> $v1 -> 3

// 5'b00001, //rt -> $at -> 1

// 16'b0000000000000100 //offset = 4

// };

end

always@(\*)begin

instOut = memory[pc[7:2]];

end

endmodule

## Pc\_Adder.v

`timescale 1ns / 1ps

module Pc\_Adder(

input[31:0] pc,

output reg[31:0] nextPc

);

always@(\*)begin

nextPc <= pc + 32'b100;

end

endmodule

## IFID\_Pipeline\_Register.v

`timescale 1ns / 1ps

module IFID\_Pipeline\_Register(

input clock,

input [31:0] instOut,

output reg [31:0] dinstOut,

output reg [5:0] op,

output reg [5:0] func,

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

output reg [15:0] imm

);

always@(posedge clock) begin

dinstOut <= instOut;

op = instOut[31:26];

func = instOut[5:0];

rs = instOut[25:21];

rt = instOut[20:16];

rd = instOut[15:11];

imm = instOut[15:0];

end

endmodule

## Control\_Unit.v

`timescale 1ns / 1ps

module Control\_Unit(

input[5:0] op,

input[5:0] func,

output reg RegWrite, //wreg

output reg MemtoReg, //m2reg

output reg MemWrite, //wmem

output reg[3:0] ALUop, //aluc

output reg ALUsrc, //aluimm

output reg regDst //regrt

);

always@(\*) begin

case (op)

6'b000000: // r-types

begin

case (func)

6'b100000: //ADD Instruction

begin

RegWrite <= 1;

MemtoReg <= 0;

MemWrite <= 0;

ALUop <= 4'b0010;

ALUsrc <= 0; //no sign extend

regDst <= 0; //write in rd

end

endcase

end

6'b100011: //LW

begin

RegWrite <= 1;

MemtoReg <= 1;

MemWrite <= 0;

ALUop <= 4'b0010;

ALUsrc <= 1; //sign extend

regDst <= 1; //write in rt

end

endcase

end

endmodule

## regDst\_mux.v

`timescale 1ns / 1ps

module regDst\_mux(

input[4:0] rt,

input[4:0] rd,

input regDst,

output reg[4:0] destReg

);

always@(\*)begin

if(!regDst)begin

destReg <= rd;

end

else begin

destReg <= rt;

end

end

endmodule

## Register\_File.v

`timescale 1ns / 1ps

module Register\_File(

input[4:0] rs,

input[4:0] rt,

output reg[31:0] qa,

output reg[31:0] qb

);

reg [31:0] registers [0:31];

integer i;

initial begin

for(i=0;i<32;i=i+1)begin

registers[i]=0;

end

end

always@(\*) begin

qa = registers[rs];

qb = registers[rt];

end

endmodule

## imm\_Extender.v

`timescale 1ns / 1ps

module imm\_Extender(

input[15:0] imm,

output reg[31:0] imm32

);

always@(\*) begin

imm32 = {{16{imm[15]}},imm};

end

endmodule

## IDEXE\_Pipeline\_Register.v

`timescale 1ns / 1ps

module IDEXE\_Pipeline\_Register(

input clock,

input RegWrite, //wreg

input MemtoReg, //m2reg

input MemWrite, //wmem

input[3:0] ALUop, //aluc

input ALUsrc, //aluimm

input[4:0] destReg,

input[31:0] qa,

input[31:0] qb,

input[31:0] imm32,

output reg eRegWrite,

output reg eMemtoReg,

output reg eMemWrite,

output reg[3:0] eALUop,

output reg eALUsrc,

output reg[4:0] edestReg,

output reg[31:0] eqa,

output reg[31:0] eqb,

output reg[31:0] eimm32

);

always@(posedge clock) begin

eRegWrite <= RegWrite;

eMemtoReg <= MemtoReg;

eMemWrite <= MemWrite;

eALUop <= ALUop;

eALUsrc <= ALUsrc;

edestReg <= destReg;

eqa <= qa;

eqb <= qb;

eimm32 <= imm32;

end

endmodule

# Your Verilog® Test Bench design code.

`timescale 1ns / 1ps

module testbench();

reg clock;

wire [31:0] pc,dinstOut,eqa, eqb, eimm32;

wire eRegWrite, eMemtoReg,eMemWrite,eALUsrc;

wire [3:0] eALUop;

wire [4:0] edestReg;

IFID\_IDEXE\_Module IFID\_IDEXE\_Module\_tb(clock,pc,dinstOut,eqa,eqb,eimm32,eRegWrite,eMemtoReg,eMemWrite,eALUsrc,eALUop,edestReg);

initial begin

clock = 0;

end

always#5 clock = ~clock;

endmodule

# Waveform

图形用户界面

描述已自动生成

# Schematic:

图形用户界面, 文本

描述已自动生成

# I/O Planning:

游戏的屏幕

描述已自动生成

# floor planning

图片包含 黑暗, 监控, 灯光, 屏幕

描述已自动生成