CMPEN 331

Sec 001

LAB 5

Kaile Ying

Datapath.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 10:00:13

// Design Name:

// Module Name: Datapath

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module Datapath(

input clk,

output reg [31:0] pc,

output reg [31:0]dinstOut,

output reg ewreg,

output reg em2reg,

output reg ewmem,

output reg [3:0] ealuc,

output reg ealuimm,

output reg [4:0] edestReg,

output reg [31:0] eqa,

output reg [31:0] eqb,

output reg [31:0] eimm32,

output reg mwreg,

output reg mm2reg,

output reg mwmem,

output reg [4:0] mdestReg,

output reg [31:0] mr,

output reg [31:0] mqb,

output reg wwreg,

output reg wm2reg,

output reg [4:0] wdestReg,

output reg [31:0] wr,

output reg [31:0] wdo

);

wire [31:0] nextPc;

wire [31:0] pc;

PC Pc(nextPc, clk, pc);

wire [31:0]instOut;

InstructionMemory mem(pc, instOut);

PcAdder adder(pc, nextPc);

wire [31:0] dinstOut;

IFIDPipelineReg ifid(instOut, clk, dinstOut);

wire [5:0] op = dinstOut[31:26];

wire [5:0] func = dinstOut[5:0];

wire wreg;

wire m2reg;

wire wmem;

wire [3:0] aluc;

wire aluimm;

wire regrt;

ControlUnit cu(op, func, wreg, m2reg, wmem, aluc, aluimm, regrt);

wire [4:0] rt = dinstOut[20:16];

wire [4:0] rd = dinstOut[15:11];

wire [4:0] destReg;

RegrtMultiplexer rm(rt, rd, regrt, destReg);

wire [4:0] rs = dinstOut[25:21];

wire [31:0] qa;

wire [31:0] qb;

wire [31:0] wbData;

RegisterFile rf(rs, rt, wdestReg, wbData, wwreg, clk, qa, qb);

wire [15:0] imm = dinstOut[15:0];

wire [31:0] imm32;

ImmediateExtender ie(imm, imm32);

wire ewreg;

wire em2reg;

wire ewmem;

wire [3:0] ealuc;

wire ealuimm;

wire [4:0] edestReg;

wire [31:0] eqa;

wire [31:0] eqb;

wire [31:0] eimm32;

IDEXEPipelineReg idexe(wreg, m2reg, wmem, aluc, aluimm, destReg, qa, qb, imm32, clk, ewreg, em2reg, ewmem, ealuc, ealuimm, edestReg, eqa, eqb, eimm32);

wire [31:0] b;

aluMultiplexer alumult(eqb, eimm32, ealuimm, b);

wire [31:0] r;

alu alu(eqa, b, ealuc, r);

wire mwreg;

wire mm2reg;

wire mwmem;

wire mdestReg;

wire [4:0] mdestReg;

wire [31:0] mr;

wire [31:0] mqb;

exememPipelineReg exemem(ewreg, em2reg, ewmem, edestReg, r, eqb, clk, mwreg, mm2reg, mwmem, mdestReg, mr, mqb);

wire [31:0] mdo;

dataMemory datamemory(mr, mqb, memem, clk, mdo);

wire wwreg;

wire wm2reg;

wire [4:0] wdestReg;

wire [31:0] wr;

wire [31:0] wdo;

memwbPipelineReg memwb(mwreg, mm2reg, mdestReg, mr, mdo, clk, wwreg, wm2reg, wdestReg, wr, wdo);

WbMux wbmux(wr,wdo, wm2reg, wbData);

Endmodule

PC.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 07:19:52

// Design Name:

// Module Name: PC

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

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

InstructionMemory.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 07:57:30

// Design Name:

// Module Name: InstructionMemory

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module InstructionMemory(

input [31:0] pc,

output reg [31:0] instOut

);

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

initial begin

memory[25] = {6'b100011, 5'b00001, 5'b00010, 16'd0};

memory[26] = {6'b100011, 5'b00001, 5'b00011, 14'd4};

memory[27] = {6'b100011, 5'b00001, 5'b00100, 16'd8};

memory[28] = {6'b100011, 5'b00001, 5'b00101, 14'd12};

memory[29] = {6'b000000, 5'b00010, 5'b00110, 5'b00110, 5'b00000, 6'b100000};

end

always @(\*) begin

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

end

endmodule

PcAdder.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 07:19:52

// Design Name:

// Module Name: PcAdder

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module PcAdder(

input [31:0] pc,

output reg [31:0] nextPc

);

initial begin

nextPc = 100;

end

always @(\*) begin

nextPc = pc + 4;

end

endmodule

IFIDPipelineReg.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 07:56:50

// Design Name:

// Module Name: IFIDPipelineReg

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module IFIDPipelineReg(

input [31:0] instOut,

input clock,

output reg [31:0] dinstOut

);

always @(posedge clock) begin

dinstOut = instOut;

end

endmodule

ControlUnit.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 07:58:40

// Design Name:

// Module Name: ControlUnit

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module ControlUnit(

input [5:0] op,

input [5:0] func,

output reg wreg,

output reg m2reg,

output reg wmem,

output reg [3:0] aluc,

output reg aluimm,

output reg regrt

);

always @(\*) begin

case (op)

6'b000000: //r-type

begin

case (func)

6'b100000: //add

begin

wreg = 1;

m2reg = 0;

wmem = 0;

aluc = 4'b0010;

aluimm = 0;

regrt = 0;

end

endcase

end

6'b100011: //LW

begin

wreg = 1;

m2reg = 1;

wmem = 0;

aluc = 4'b0010;

aluimm = 1;

regrt = 1;

end

endcase

end

endmodule

RegrtMultiplexer.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 08:49:41

// Design Name:

// Module Name: RegrtMultiplexer

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module RegrtMultiplexer(

input [4:0] rt,

input [4:0] rd,

input regrt,

output reg [4:0] destReg

);

always @(\*) begin

if(regrt == 0)

begin

destReg <= rd;

end

else

begin

destReg <= rt;

end

end

endmodule

RegisterFile.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 07:19:52

// Design Name:

// Module Name: RegisterFile

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module RegisterFile(

input [4:0] rs,

input [4:0] rt,

input [4:0] wdestReg,

input [31:0] wbData,

input wwreg,

input clock,

output reg [31:0] qa,

output reg [31:0] qb

);

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

integer i;

initial begin

for(i = 0; i <= 31; i= i + 1)

begin

registers[i] = 0;

end

end

always @(negedge clock) begin

if (wwreg == 1)

begin

registers[wdestReg] = wbData;

end

end

always @(\*) begin

qa <= registers[rs];

qb <= registers[rt];

end

endmodule

ImmediateExtender.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 09:47:09

// Design Name:

// Module Name: ImmediateExtender

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module ImmediateExtender(

input [15:0] imm,

output reg [31:0] imm32

);

always @(\*) begin

imm32 <= {{16{imm[15]}},imm[15:0]};

end

endmodule

IDEXEPipelineReg.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/05 09:52:12

// Design Name:

// Module Name: IDEXEPipelineReg

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module IDEXEPipelineReg(

input wreg,

input m2reg,

input wmem,

input [3:0] aluc,

input aluimm,

input [4:0] destReg,

input [31:0] qa,

input [31:0] qb,

input [31:0] imm32,

input clock,

output reg ewreg,

output reg em2reg,

output reg ewmem,

output reg [3:0] ealuc,

output reg ealuimm,

output reg [4:0] edestReg,

output reg [31:0] eqa,

output reg [31:0] eqb,

output reg [31:0] eimm32

);

always @(posedge clock) begin

ewreg = wreg;

em2reg = wmem;

ewmem = wmem;

ealuc = aluc;

ealuimm = aluimm;

edestReg = destReg;

eqa = qa;

eqb = qb;

eimm32 = imm32;

end

endmodule

aluMultiplexer.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/19 09:49:04

// Design Name:

// Module Name: aluMultiplexer

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module aluMultiplexer(

input [31:0] eqb,

input [31:0] eimm32,

input ealuimm,

output reg [31:0] b

);

always @(\*) begin

if(ealuimm == 0)

begin

b = eqb;

end

else

begin

b = eimm32;

end

end

endmodule

alu.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/19 09:52:22

// Design Name:

// Module Name: alu

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module alu(

input [31:0] eqa,

input [31:0] b,

input [3:0] ealuc,

output reg [31:0] r

);

always @(\*) begin

if (ealuc == 4'b0010)

begin

r = eqa + b;

end

end

endmodule

exememPipelineReg.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/19 10:01:04

// Design Name:

// Module Name: exememPipelineReg

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module exememPipelineReg(

input ewreg,

input em2reg,

input ewmem,

input [4:0] edestReg,

input [31:0] r,

input [31:0] eqb,

input clock,

output reg mwreg,

output reg mm2reg,

output reg mwmem,

output reg [4:0] mdestReg,

output reg [31:0] mr,

output reg [31:0] mqb

);

always @(posedge clock) begin

mwreg = ewreg;

mm2reg = em2reg;

mwmem = ewmem;

mdestReg = edestReg;

mr = r;

mqb = eqb;

end

endmodule

dataMemory.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/19 10:05:57

// Design Name:

// Module Name: dataMemory

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module dataMemory(

input [31:0] mr,

input [31:0] mqb,

input mwmem,

input clock,

output reg [31:0] mdo

);

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

initial begin

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;

end

always @(negedge clock) begin

if(mwmem ==1)

begin

memory[mr[7:2]] = mqb;

end

mdo = memory[mr[7:2]];

end

endmodule

memwbPipelineReg.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/19 11:09:22

// Design Name:

// Module Name: memwbPipelineReg

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module memwbPipelineReg(

input mwreg,

input mm2reg,

input [4:0] mdestReg,

input [31:0] mr,

input [31:0] mdo,

input clock,

output reg wwreg,

output reg wm2reg,

output reg [4:0] wdestReg,

output reg [31:0] wr,

output reg [31:0] wdo

);

always @(posedge clock) begin

wwreg = mwreg;

wm2reg = mm2reg;

wdestReg <= mdestReg;

wr = mr;

wdo = mdo;

end

endmodule

WbMux.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2022/07/31 10:43:26

// Design Name:

// Module Name: WbMux

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module WbMux(

input [31:0] wr,

input [31:0] wdo,

input wm2reg,

output reg [31:0] wbData

);

always @(\*) begin

if (wm2reg == 0)

begin

wbData = wr;

end

else

begin

wbData = wdo;

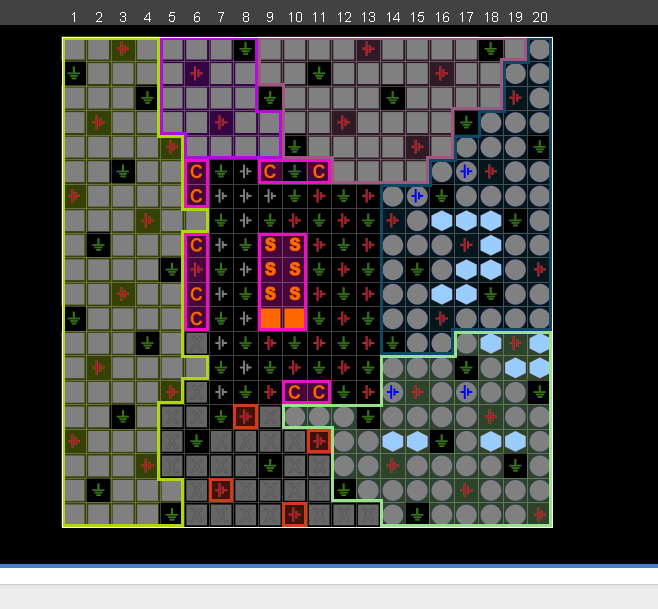
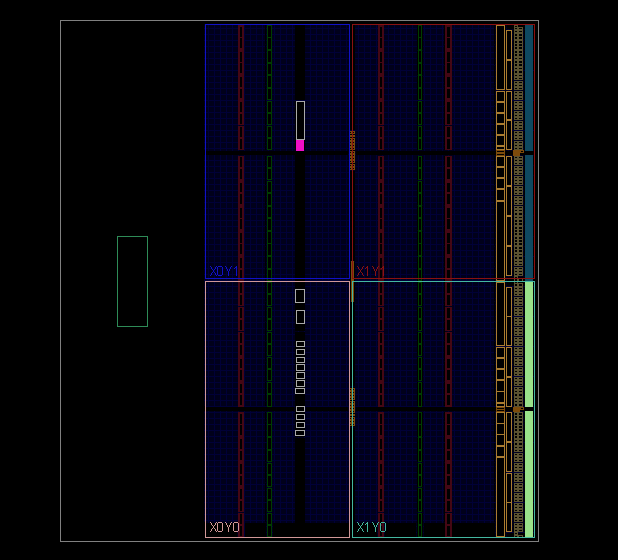
end

end

endmodule

日程表

描述已自动生成电视游戏的萤幕截图

描述已自动生成图示

中度可信度描述已自动生成