CMPEN 331 —— LAB5

Tianqi Liu

Section II

Source Code:

```
`timescale 1ns / 1ps


module PC(
    input clk, rst,
    input [31:0] PCaddress,
    output reg [31:0] oldPC
    );
    reg [31:0] newPC;

    always@(posedge clk) begin
        if (rst) begin
            newPC =    PCaddress;
            oldPC = newPC;
        end
        else begin
            newPC = newPC + 4;
            oldPC = newPC;
        end
    end
endmodule
```

```
module IFID(
        input clk,
        input [31:0] do,
        output reg [5:0] op,
        output reg [5:0] func,
        output reg [4:0] rd, rs, rt,
        output reg [15:0] imm
        );

        always@(posedge clk) begin
                op = do [31:26];
                func = do [5:0];
                rs = do [25:21];
                rt = do [20:16];
                rd = do [15:11];
                imm = do [15:0];
        end
endmodule

module IDEXE(
        input clk,wreg, m2reg, wmem,aluimm,
        input [3:0] aluc,
        input [31:0] qa, qb,

        input [4:0] mux,

        input [31:0] extend,

        output reg ewreg, em2reg, ewmem, ealuimm,

        output reg [3:0] ealuc,

        output reg [31:0] eqa, eqb,

        output reg [4:0] emux,

        output reg [31:0] Extender

        );


        always@(posedge clk) begin

                ewreg = wreg;

                em2reg = m2reg;

                ewmem = wmem;

                ealuimm = aluimm;

                emux = mux;

                ealuc = aluc;

                eqa = qa;

                eqb = qb;

                Extender = extend;

        end

endmodule
```

```
module InstMem(

        input [31:0] a,

        output [31:0] do

        );



        reg [31:0] Inst_memory[0:127];

        initial begin

        //initialize the first 10 words of the Data memory with the following HEX values.

        Inst_memory[100]=32'h8D090000; //lw $t1, 0($t0);

        Inst_memory[104]=32'h8D0A0004; //lw $t2, 4($t0);

        Inst_memory[108]=32'h8D0B0008; //lw $t3, 8($t0);

        Inst_memory[112]=32'h8D0C000C; //lw $t4, 12($t0);

        end



        assign do = Inst_memory[a];



endmodule



module CtrUnit(

        input [5:0] op, func,

        output reg wreg, m2reg, wmem, aluimm, regrt,
output reg [3:0] aluc
);

always@(op,func) begin
        case(op)
                6'b100011: begin
                        aluimm = 1;
                        aluc = 4'b0010;
                        wreg = 1;
                        m2reg = 1;
                        wmem = 0;
                        regrt = 1;
                end
                default: begin                      //ignor this time
                        aluc = 4'b0000;
                        aluimm = 0;
                        wreg = 0;
                        m2reg = 0;
                        wmem = 0;
                        regrt = 1;
                end
        endcase
        end

endmodule

module RegFile(
        input   we,
        input [4:0] rs, rt, wn, d,
        output reg [31:0] qa, qb
        );
        reg [31:0] registers [0:127];
        initial begin                           //initiallize register file
                registers[0] = 0;
                registers[1] = 0;
                registers[2] = 0;
                registers[3] = 0;
                registers[4] = 0;
                registers[5] = 0;
                registers[6] = 0;
                registers[7] = 0;
                registers[8] = 0;
                registers[9] = 0;
                registers[10] = 0;
```

```
            registers[11] = 0;
            registers[12] = 0;
            registers[13] = 0;
            registers[14] = 0;
            registers[15] = 0;
        end
        always@(rs, rt) begin
             qa = registers[rs];     //grab value from register file
             qb = registers[rt];
        end
endmodule


module Extender(
        input [15:0] imm,
        output reg [31:0] long
        );


        always@(imm) begin
            if(imm[15] == 1) begin
                long[31:16] = 16'hffff;
                long[15:0] = imm;
            end
```

```
                else begin
                        long[31:16] = 16'h0000;
                        long[15:0] = imm;
                end
        end
endmodule

module CtrMux(
        input regrt,
        input [4:0] rd, rt,
        output [4:0] wn
        );

  assign wn = regrt?rt:rd;
endmodule

module ALUMux(
        input [31:0] qb,
        input [31:0] imm,
        input ealuimm,
        output [31:0] b
        );
```

```
    assign b = ealuimm?imm:qb;
endmodule


module ALU(
    input [31:0] a,
    input [31:0] b,
    input [3:0] aluc,
    output reg [31:0] r
    );
    always@(a, b, aluc) begin
        case(aluc)
            4'b0010: begin
                r = a + b;
            end
            default: begin              //ignor this time
                r = 0;
            end
         endcase
    end
endmodule
```

```
module EXEMEM(
    input clk, ewreg, em2reg, ewmem,
    input [4:0] mux,
    input [31:0] qb,
    input [31:0] r,
    output reg mwreg, mm2reg, mwmem,
    output reg [4:0] mmux,
    output reg [31:0] mqb,
    output reg [31:0] mr
    );

    always@(posedge clk) begin
        mwreg = ewreg;
        mm2reg = em2reg;
        mwmem = ewmem;
        mmux = mux;
        mqb = qb;
        mr = r;
    end
endmodule

module DataMem(
input [31:0] a,

input [31:0] di,

input we,

output reg [31:0] do

);

reg [31:0] Data_memory[0:127];

initial begin

        Data_memory[0]=32'hA00000AA;

        Data_memory[1]=32'h10000011;

        Data_memory[2]=32'h20000022;

        Data_memory[3]=32'h30000033;

        Data_memory[4]=32'h40000044;

        Data_memory[5]=32'h50000055;

        Data_memory[6]=32'h60000066;

        Data_memory[7]=32'h70000077;

        Data_memory[8]=32'h80000088;

        Data_memory[9]=32'h90000099;

end

always@(a, di) begin

        if( we == 0) begin

                do = Data_memory[a];

        end
```

```
    end
endmodule


module MEMWB(
    input clk, mwreg, mm2reg,
    input [4:0] mux,
    input [31:0] r, do,
    output reg wwreg, wm2reg,
    output reg [4:0] wmux,
    output reg [31:0] wr, wdo
    );
    always@(posedge clk) begin
        wwreg = mwreg;
        wm2reg = mm2reg;
        wmux = mux;
        wr = r;
        wdo =do;
    end
endmodule
```

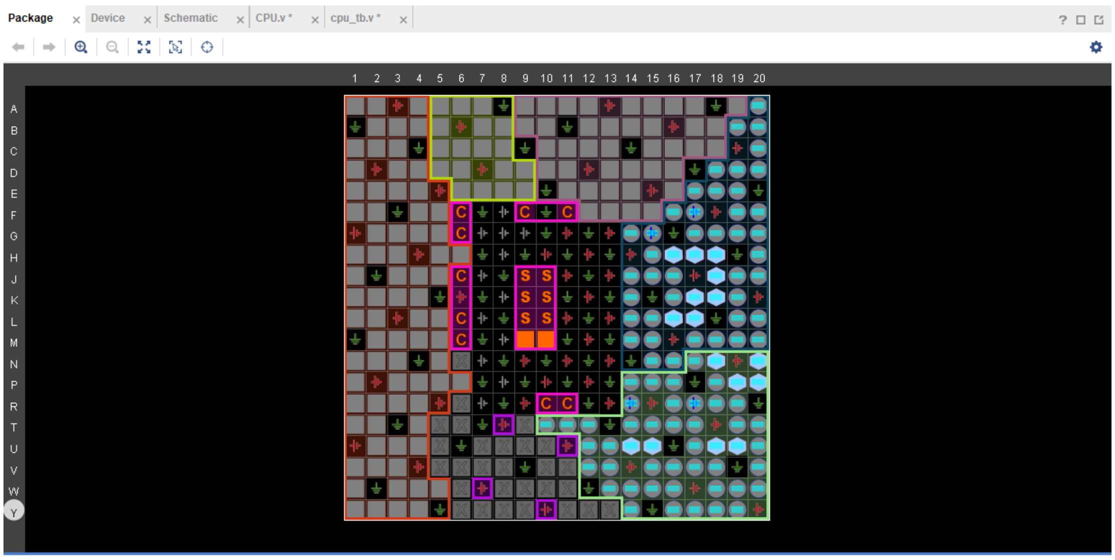
Test Bench:

```
`timescale 1ns / 1ps


module cpu_tb();
    reg clk, rst;


    wire [31:0] pcnow;
    wire [31:0] dowire;
    wire [5:0] opwire;
    wire [5:0] funcwire;
    wire [4:0] rdwire, rtwire, rswire;
    wire [15:0] immwire;
    wire wregwire, m2regwire, wmemwire, aluimmwire;
    wire ewregwire, em2regwire, ewmemwire, ealuimmwire;
    wire mwregwire, mm2regwire, mwmemwire;
    wire wwregwire, wm2regwire;
    wire [3:0] alucwire, ealucwire;
    wire regrtwire;
    wire [31:0] longwire;
    reg wewire;
    wire [4:0] muxwire, dwire, emuxwire, mmuxwire, wmuxwire;
    reg [31:0] PCAddress;
```

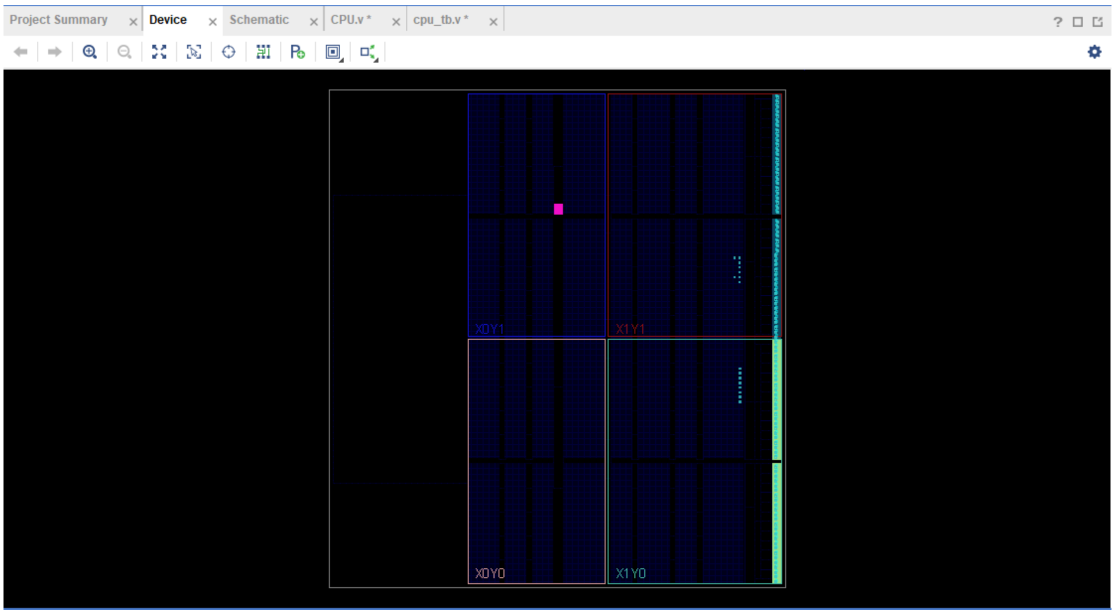
```
wire [31:0] qawire, qbwire, mqbwire;

wire [4:0] wnwire;

wire [31:0] eqbwire, eqawire, Extenderwire;

wire [31:0] bwire, rwire, mrwire;

wire [31:0] mdowire;

wire [31:0] wrwire, wdowire;




PC pc(clk, rst, PCAddress, pcnow);

InstMem instmem(pcnow, dowire);

IFID ifid (clk, dowire, opwire, funcwire, rdwire,

             rswire, rtwire, immwire);

CtrUnit ctrunit(opwire, funcwire,

            wregwire, m2regwire, wmemwire,

            aluimmwire, regrtwire, alucwire);

RegFile regfile(wewire, rswire, rtwire, wnwire,

            dwire, qawire, qbwire);

CtrMux ctrmux(regrtwire, rdwire, rtwire,muxwire);

Extender extender(immwire, longwire);

IDEXE idexe(clk, wregwire, m2regwire,

            wmemwire, aluimmwire, alucwire, qawire, qbwire, muxwire,
            longwire, ewregwire, em2regwire, ewmemwire, ealuimmwire,

            ealucwire, eqawire, eqbwire, emuxwire, Extenderwire);

    ALUMux alumux(eqbwire, Extenderwire,ealuimmwire, bwire);

    ALU alu(eqawire, bwire, ealucwire, rwire);

    EXEMEM exemem(clk, ewregwire, em2regwire, ewmemwire, emuxwire,

            eqbwire,rwire, mwregwire, mm2regwire, mwmemwire, mmuxwire,

            mqbwire, mrwire);

    DataMem datamem(mrwire, mqbwire, mwmemwire, mdowire);

    MEMWB memwb(clk, mwregwire, mm2regwire, mmuxwire, mrwire, mdowire,

            wwregwire, wm2regwire, wmuxwire, wrwire, wdowire);


    initial

    begin

        clk = 1;

        PCAddress = 100;

        rst = 1;

        #1 rst = 0;

    end


    always

       #100 clk = ! clk;

endmodule
```

Device: XC7Z010--1CLG400C

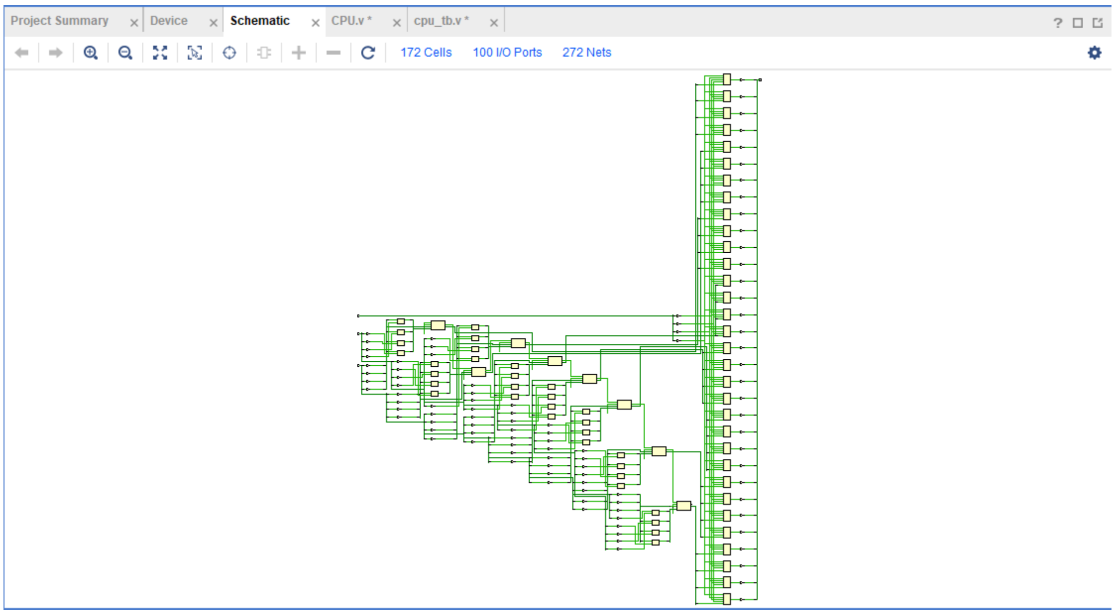
I/O Planning:



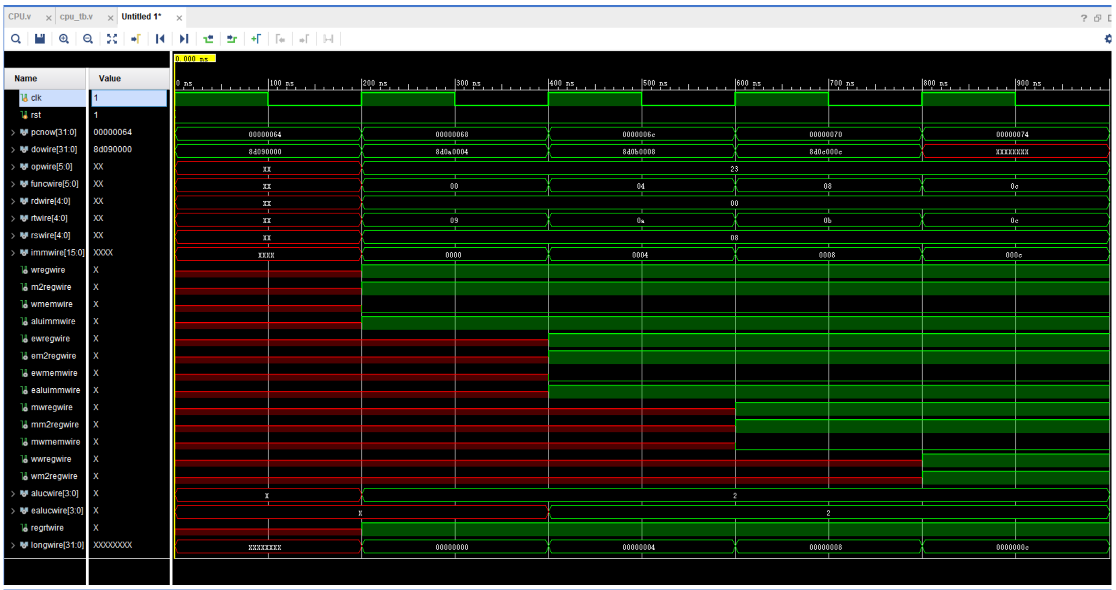
Floor Planning:

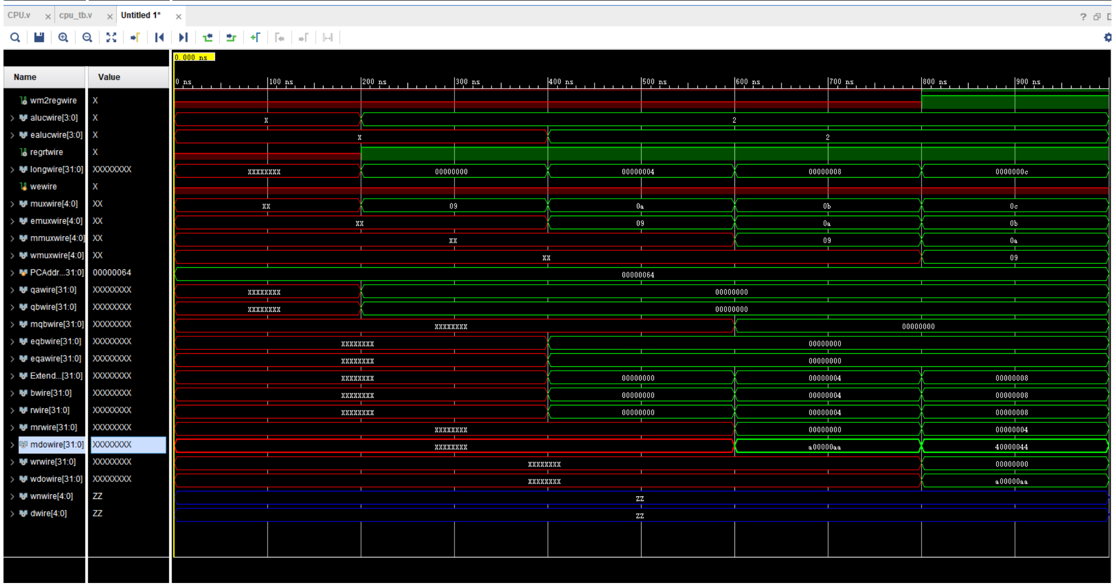


Schematic:


Project Summary
Device
Schematic
CPU.v *
cpu_tb.v *
172 Cells
100 I/O Ports
272 Nets

Wave Form:





Wnwire and dwire are unused.