**Penn State University**

**School of Electrical Engineering and Computer Science**

**CMPEN331 – Computer Organization and Design**

**Section 002**

**Instructor: Dr. Mohamed Almekkawy**

**Lab 3**

**Produced by Hongshuo Wang**

* Verilog design code/TestBench Code (Device: XC7Z010CLG4003)

1. Lab 3

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 03/30/2019 03:49:20 PM

// Design Name:

// Module Name: Lab03

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module PCAdder(

input [7:0] currentPC,

output reg [7:0] upDatedPC);

parameter INCREMENT\_AMOUNT = 8'd4;

initial upDatedPC = 8'd100;

always @(\*)

begin

upDatedPC <= currentPC + INCREMENT\_AMOUNT;

end

endmodule

module instMem(

input [7:0] a,

output reg[31:0] do

);

reg [31:0] IM [0:255];

parameter FIRST\_LOCATION = 8'd100;

parameter SECOND\_LOCATION = 8'd104;

initial begin

IM[FIRST\_LOCATION] = 32'b10001100001000100000000000000000;

IM[SECOND\_LOCATION] = 32'b10001100001000110000000000000100;

end

always @(\*)

begin

do <= IM[a];

end

endmodule

module controlUnit(

input [31:0] do,

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

output reg [3:0] aluc

);

parameter LW = 6'b100011;

parameter R = 6'b000000;

parameter SW = 6'b101011;

parameter BEQ = 6'b000100;

parameter ADD = 6'b100000;

parameter SUB = 6'b100010;

parameter AND = 6'b100100;

parameter OR = 6'b100101;

parameter XOR = 6'b100110;

parameter SLT = 6'b101010;

wire [5:0] op, func;

assign op = do[31:26];

assign func = do[5:0];

always @(\*)

begin

case (op)

LW: begin

wreg <= 1;

m2reg <= 1;

wmem <= 0;

regrt <= 1;

aluimm <= 1;

aluc <= 4'b0010;

end

SW: begin

wreg <= 0;

wmem <= 1;

aluc <= 4'b0010;

aluimm <= 1;

end

BEQ: begin

aluimm <= 0;

wreg <= 0;

wmem <= 0;

aluc <= 4'b0110;

end

R: begin

wreg <= 1;

m2reg <= 0;

wmem <= 0;

regrt <= 1;

aluimm <= 0;

case (func)

ADD: begin

aluc <= 4'b0010;

end

SUB:begin

aluc <= 4'b0110;

end

AND: begin

aluc <= 4'b0000;

end

OR: begin

aluc <= 4'b0001;

end

SLT: begin

aluc <= 4'b0111;

end

XOR: begin

aluc <= 4'b1100;

end

endcase

end

endcase

end

endmodule

module rtrdMultiplier(

input [31:0] do,

input regrt,

output reg[4:0] outputReg

);

wire [4:0] rd,rt;

assign rd = do[15:11];

assign rt = do[20:16];

always @(\*)

begin

if (regrt == 1)

outputReg <= rt;

else

outputReg <= rd;

end

endmodule

module regfile(

input [31:0] do,

output [31:0] qa, qb

);

reg [31:0] RF [31:0];

integer i;

initial

begin

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

RF[i] <= 32'd0;

end

wire [4:0] rna, rnb;

assign rna = do[25:21];

assign rnb = do[20:16];

assign qa = RF[rna];

assign qb = RF[rnb];

endmodule

module e(

input [31:0] do,

output reg [31:0] extendedImm

);

wire [15:0] imm;

assign imm = do[15:0];

always @(\*)

begin

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

end

endmodule

module PCReg(

input [7:0] inputPC,

input clk,

output reg[7:0] outputPC

);

always @(posedge clk)

begin

outputPC <= inputPC;

end

endmodule

module IF\_IDReg(

input [31:0] inputDo,

input clk,

output reg [31:0] outputDo

);

always @(posedge clk)

begin

outputDo <= inputDo;

end

endmodule

module ID\_EXEReg(

input inputWreg, inputM2reg, inputWmem, inputAluimm, clk,

input [3:0] inputAluc,

input [4:0] inputReg,

input [31:0] inputQa, inputQb, inputImm,

output reg outputWreg, outputM2reg, outputWmem, outputAluimm,

output reg [31:0] outputQa, outputQb, outputImm,

output reg [3:0] outputAluc,

output reg [4:0] outputReg

);

always @(posedge clk)

begin

outputWreg <= inputWreg;

outputM2reg <= inputM2reg;

outputWmem <= inputWmem;

outputAluimm <= inputAluimm;

outputQa <= inputQa;

outputQb <= inputQb;

outputImm <= inputImm;

outputAluc <= inputAluc;

outputReg <= inputReg;

end

endmodule

1. Testbench

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 03/30/2019 06:28:24 PM

// Design Name:

// Module Name: testBench

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module testBench();

wire [7:0] inputPC\_tb;

wire [31:0] inputDo\_tb, outputDo\_tb, inputQa\_tb, inputQb\_tb, outputQa\_tb, outputQb\_tb, inputImm\_tb, outputImm\_tb;

wire inputWreg\_tb, inputM2reg\_tb, inputWmem\_tb, inputAluimm\_tb, regrt\_tb;

wire outputWreg\_tb, outputM2reg\_tb, outputWmem\_tb, outputAluimm\_tb;

wire [3:0] inputAluc\_tb, outputAluc\_tb;

wire [4:0] inputReg\_tb, outputReg\_tb;

reg clk\_tb;

wire [7:0] outputPC\_tb;

PCAdder PCAdder\_tb(.currentPC(outputPC\_tb), .upDatedPC(inputPC\_tb));

PCReg PCReg\_tb(.inputPC(inputPC\_tb), .clk(clk\_tb), .outputPC(outputPC\_tb));

instMem instMem\_tb(.a(outputPC\_tb), .do(inputDo\_tb));

IF\_IDReg IF\_IDReg\_tb(.inputDo(inputDo\_tb), .clk(clk\_tb), .outputDo(outputDo\_tb));

controlUnit controlUnit\_tb(.do(outputDo\_tb), .regrt(regrt\_tb), .wreg(inputWreg\_tb), .m2reg(inputM2reg\_tb), .wmem(inputWmem\_tb), .aluimm(inputAluimm\_tb), .aluc(inputAluc\_tb));

rtrdMultiplier rtrdMultiplier\_tb(.do(outputDo\_tb), .regrt(regrt\_tb), .outputReg(inputReg\_tb));

ID\_EXEReg ID\_EXEReg\_tb

(

.inputWreg(inputWreg\_tb),

.inputM2reg(inputM2reg\_tb),

.inputWmem(inputWmem\_tb),

.inputAluimm(inputAluimm\_tb),

.clk(clk\_tb),

.inputAluc(inputAluc\_tb),

.inputReg(inputReg\_tb),

.inputQa(inputQa\_tb),

.inputQb(inputQb\_tb),

.inputImm(inputImm\_tb),

.outputWreg(outputWreg\_tb),

.outputM2reg(outputM2reg\_tb),

.outputWmem(outputWmem\_tb),

.outputAluimm(outputAluimm\_tb),

.outputQa(outputQa\_tb),

.outputQb(outputQb\_tb),

.outputImm(outputImm\_tb),

.outputAluc(outputAluc\_tb),

.outputReg(outputReg\_tb)

);

regfile regfile\_tb(.do(outputDo\_tb), .qa(inputQa\_tb), .qb(inputQb\_tb));

e e\_tb(.do(outputDo\_tb), .extendedImm(inputImm\_tb));

initial begin

clk\_tb <= 1;

end

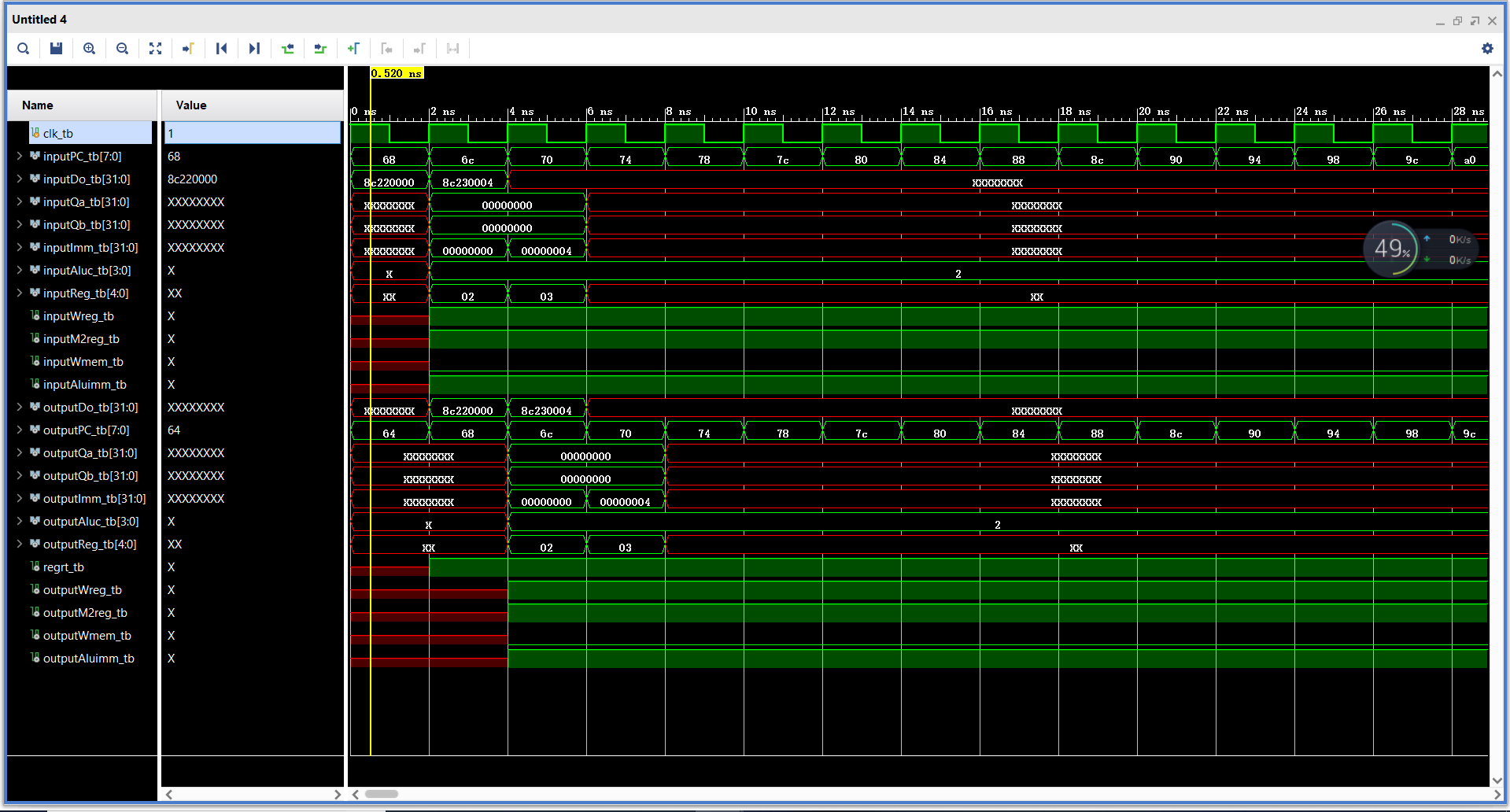
always begin

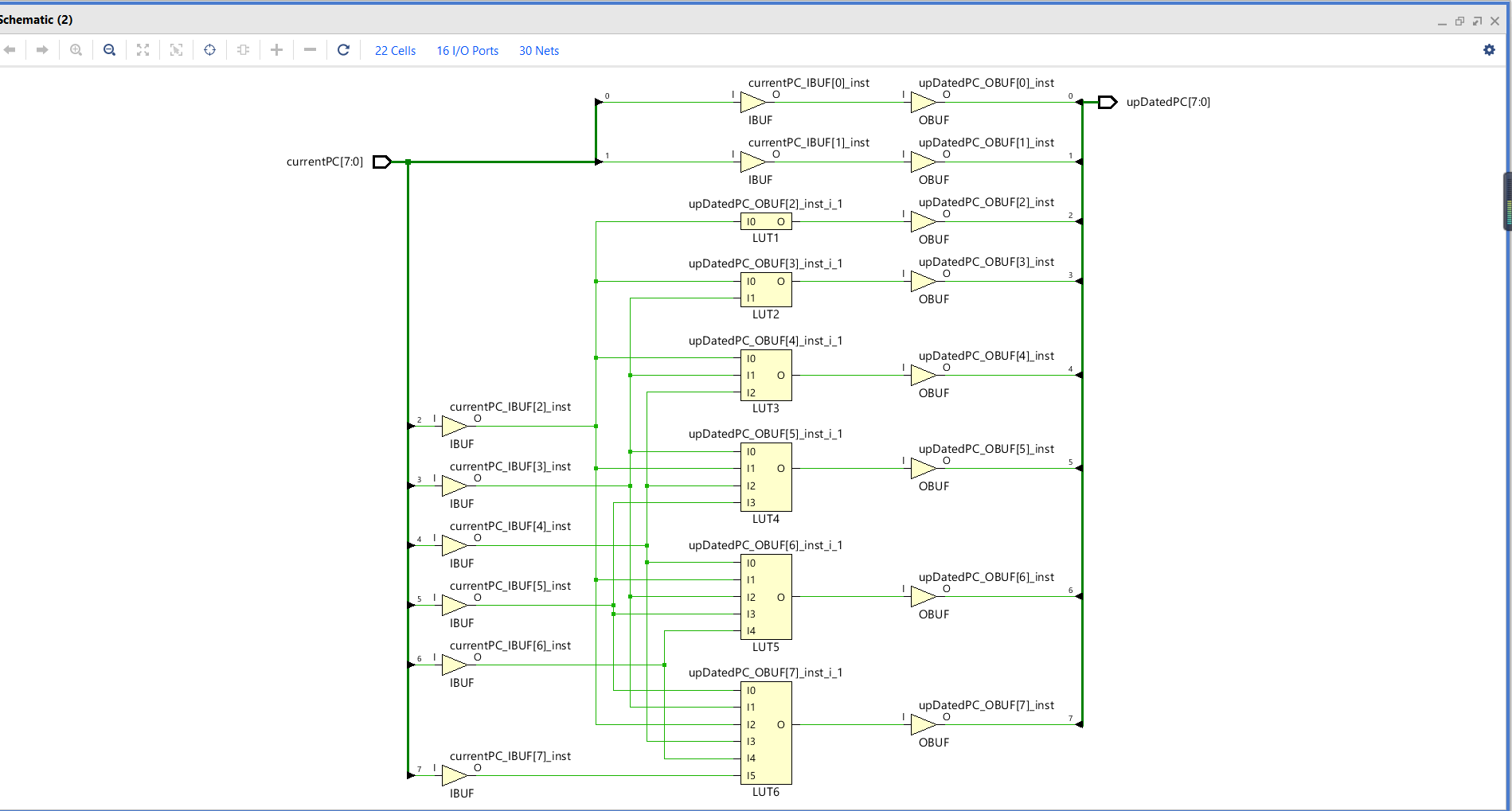
#1 clk\_tb <= ~clk\_tb;

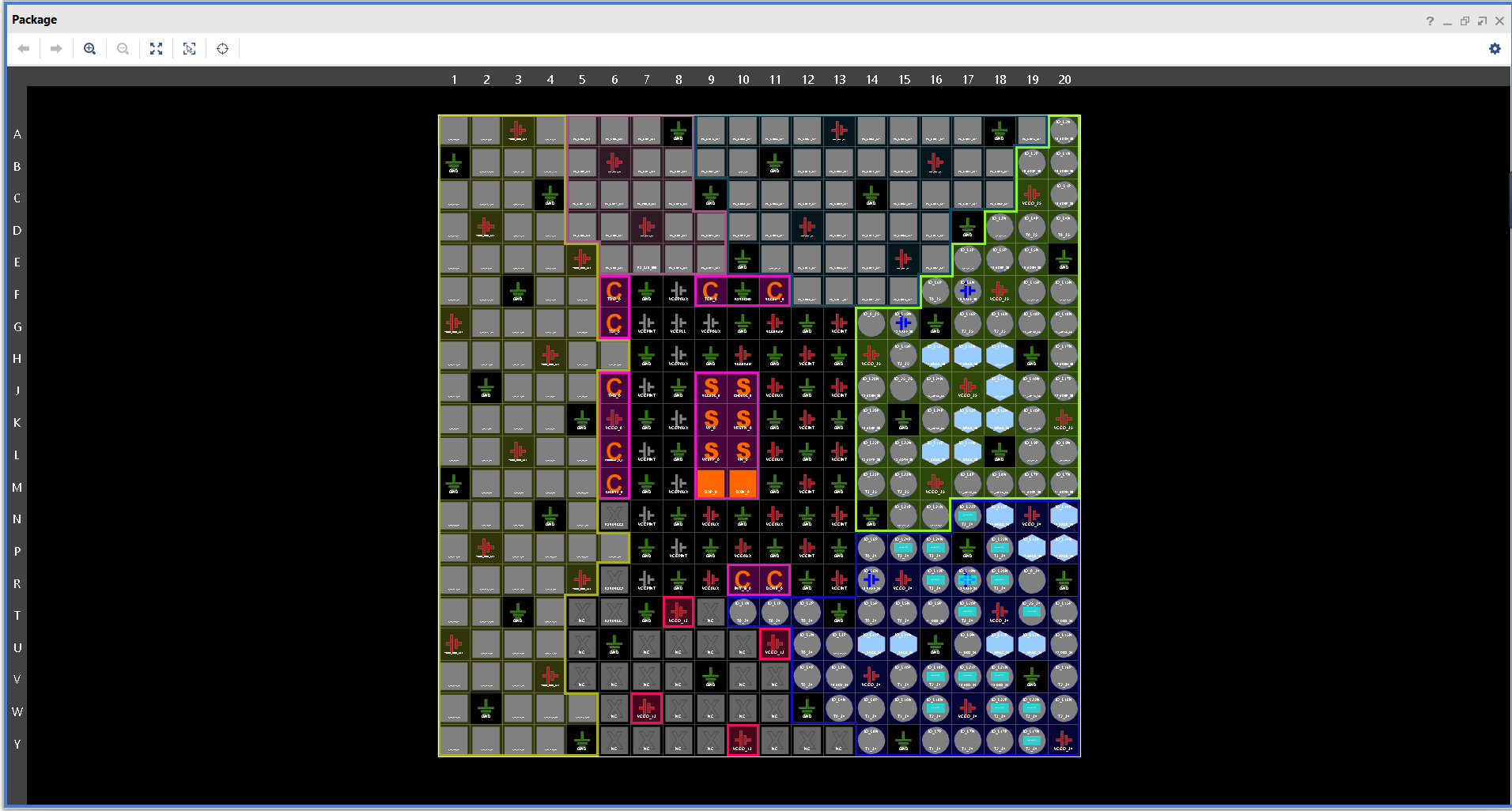
end

endmodule

* Waveform



* The design schematics
* I/O Planning



* Floor planning