Lab 04

Designing the Controller of the CPU

Category	**Ryan Cruz** ryan.cruz25@uga.edu	**Zachary Davis** zachdav@uga.edu
Pre-lab	50	50
In-lab Module & Testbench Design	50	50
In-lab Testbench Sim. & Analysis	50	50
In-lab FPGA Synthesis & Analysis	50	50
Lab Report Writing	50	50

February 8, 2018

Contents

1 Lab Purpose

2 Implementation Details

2.1 Part 1

Prelab .

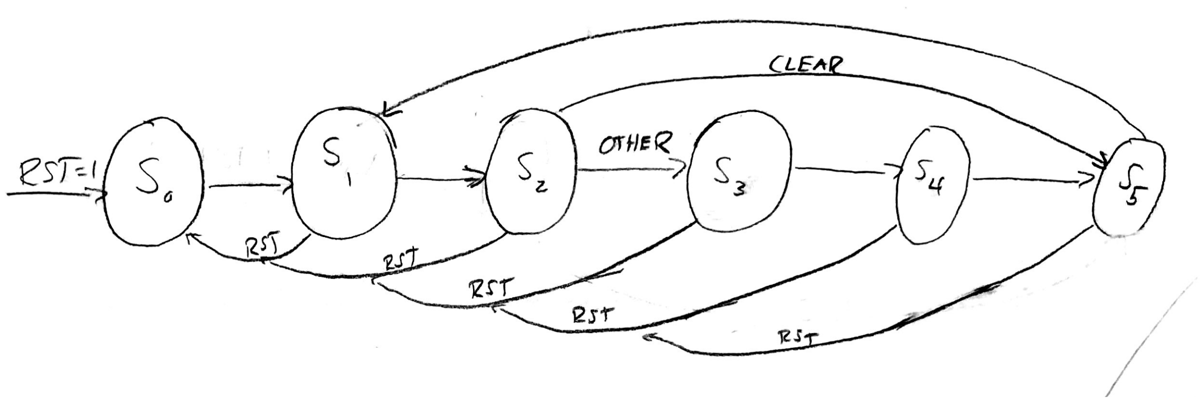
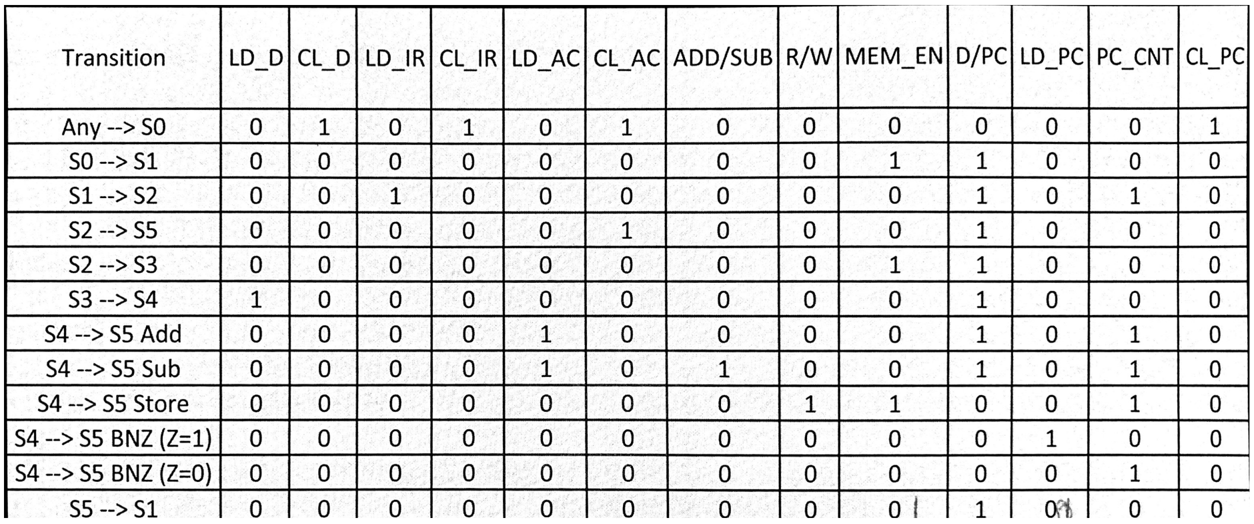


Figure 1: FSM for the controller of the CPU



Transition	LD_D	CL_D	LD_IR	CL_IR	LD_AC	CL_AC	ADD/SUB	R/W	MEM_EN	D/PC	LD_PC	PC_CNT	CL_PC
Any --> S0	0	1	0	1	0	1	0	0	0	0	0	0	1
S0 --> S1	0	0	0	0	0	0	0	0	1	1	0	0	0
S1 --> S2	0	0	1	0	0	0	0	0	0	1	0	1	0
S2 --> S5	0	0	0	0	0	1	0	0	0	1	0	0	0
S2 --> S3	0	0	0	0	0	0	0	0	1	1	0	0	0
S3 --> S4	1	0	0	0	0	0	0	0	0	1	0	0	0
S4 --> S5 Add	0	0	0	0	1	0	0	0	0	1	0	1	0
S4 --> S5 Sub	0	0	0	0	1	0	1	0	0	1	0	1	0
S4 --> S5 Store	0	0	0	0	0	0	0	1	1	0	0	1	0
S4 --> S5 BNZ (Z=1)	0	0	0	0	0	0	0	0	0	0	1	0	0
S4 --> S5 BNZ (Z=0)	0	0	0	0	0	0	0	0	0	0	0	1	0
S5 --> S1	0	0	0	0	0	0	0	0	0 1	1	0 1	0	0

Figure 2: Control Signals

LD_D = S_3

CL_D = RST

LD-IR = S_1

CL-IR = RST

LD_AC = $S_4 \cdot (ADD + SUB)$

CL_AC = RST + $S_2 \cdot$ CLEAR

ADDORSUB = $S_4 \cdot SUB$

R|W = $S_4 \cdot STORE$

D|PC = $[RST + S_4 \cdot (BNZ + STORE)]'$

LD_PC = $S_4 \cdot BNZ \cdot ZERO$

PC-CNT = $S_1 + S_4 \cdot [(ADD + SUB) + BNZ \cdot ZERO' + STORE]$

CL-PC = RST

MEM_ENC = $S_1 + S_3 + S_4 \cdot STORE$

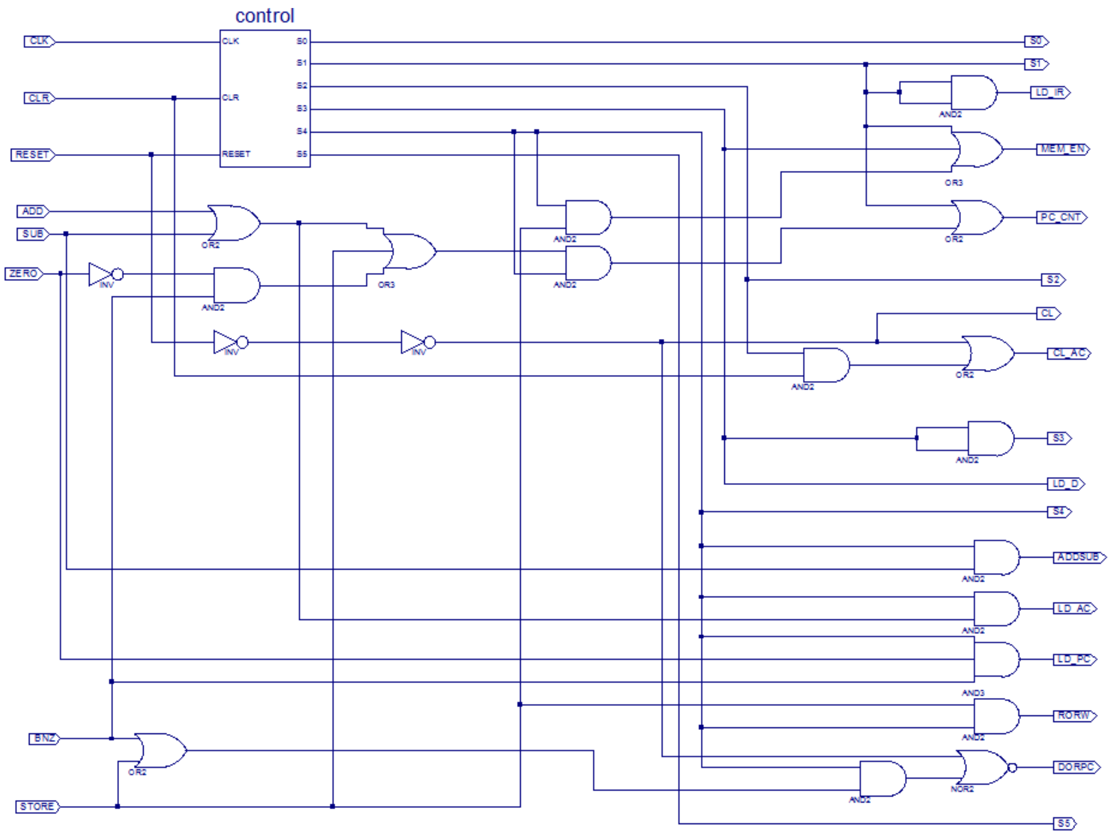
Figure 3: Boolean Equations for figure 2. control signals

```
//control.v
‘timescale 1s/1s

module control(CLK,CLR,RESET,S0,S1,S2,S3,S4,S5);


        input CLK;
        input CLR,RESET;
        output S0,S1,S2,S3,S4,S5;

        reg         S0,next_S0,
                        S1,next_S1,
                        S2,next_S2,
                        S3,next_S3,
                        S4,next_S4,
                        S5,next_S5;

        reg         STATE0,next_STATE0,
                        STATE1,next_STATE1,
                        STATE2,next_STATE2,
                        STATE3, next_STATE3,
                        STATE4,next_STATE4,
                        STATE5,next_STATE5;
```

```
always @(posedge CLK)
        begin
                STATE0 = next_STATE0;
                STATE1 = next_STATE1;
                STATE2 = next_STATE2;
                STATE3 = next_STATE3;
                STATE4 = next_STATE4;
                STATE5 = next_STATE5;
                S0 = next_S0;
                S1 = next_S1;
                S2 = next_S2;
                S3 = next_S3;
                S4 = next_S4;
                S5 = next_S5;
        end

always @ (CLR or RESET or STATE0 or STATE1 or STATE2 or STATE3
 or STATE4 or STATE5)
        begin
                if ( RESET )
                        next_STATE0=1;
                else
                        next_STATE0=0;

                if ( ~RESET & STATE0 | ~RESET & STATE5 )
                        next_STATE1=1;
                else
                        next_STATE1=0;

                if ( ~RESET & STATE1 )
                        next_STATE2=1;
                else
                        next_STATE2=0;

                if ( ~RESET & ~CLR & STATE2 )
                        next_STATE3=1;
                else
                        next_STATE3=0;

                if ( ~RESET & STATE3 )
                        next_STATE4=1;
                else
                        next_STATE4=0;

                if ( ~RESET & CLR & STATE2 | ~RESET & STATE4 )
                        next_STATE5=1;
                        else
                                next_STATE5=0;

                        if ( RESET )
                                next_S0=1;
                        else
                                next_S0=0;

                        if ( ~RESET & STATE0 | ~RESET & STATE5 )
                                next_S1=1;
                        else
                                next_S1=0;

                        if ( ~RESET & STATE1 )
                                next_S2=1;
                        else
                                next_S2=0;

                        if ( ~RESET & ~CLR & STATE2 )
                                next_S3=1;
                        else
                                next_S3=0;

                        if ( ~RESET & STATE3 )
                                next_S4=1;
                        else
                                next_S4=0;

                        if ( ~RESET & CLR & STATE2 | ~RESET & STATE4 )
                                next_S5=1;
                        else
                                next_S5=0;
        end
endmodule
```

2.2 Part 2

2.3 Part 3

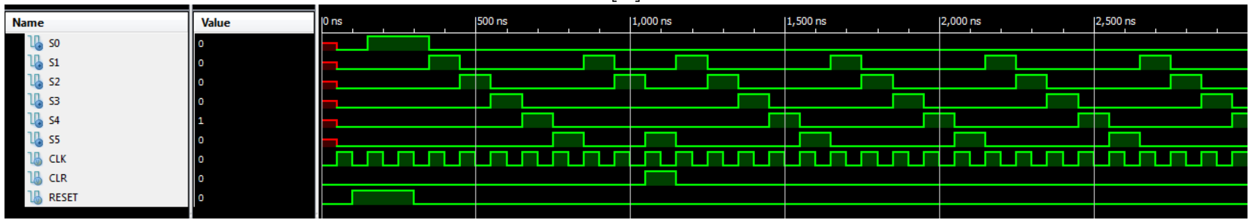


```
//controller_tb.v
‘timescale 1ns/1ps

module controller_tbw_tb_0;


        reg ADD = 1’b0;
        reg BNZ = 1’b0;
        reg CLK = 1’b0;
        reg CLR = 1’b0;
        reg RESET = 1’b0;
        reg STORE = 1’b0;
        reg SUB = 1’b0;
        reg ZERO = 1’b0;

        wire ADDSUB;
        wire CL;
        wire CL_AC;
        wire DORPC;
wire LD_AC;
wire LD_D;
wire LD_IR;
wire LD_PC;
wire MEM_EN;
wire PC_CNT;
wire RORW;
wire S0;
wire S1;
wire S2;
wire S3;
wire S4;
wire S5;

initial // Clock process for CLK
        begin
                forever
                begin
                        CLK = 1'b0;
                        #100;
                        CLK = 1'b1;
                        #100;
                end
        end

controller_sch UUT(
        .ADD(ADD),
        .BNZ(BNZ),
        .CLK(CLK),
        .CLR(CLR),
        .RESET(RESET),
        .STORE(STORE),
        .SUB(SUB),
        .ZERO(ZERO),
        .ADDSUB(ADDSUB),
        .CL(CL),
        .CL_AC(CL_AC),
        .DORPC(DORPC),
        .LD_AC(LD_AC),
        .LD_D(LD_D),
        .LD_IR(LD_IR),
        .LD_PC(LD_PC),
        .MEM_EN(MEM_EN),
        .PC_CNT(PC_CNT),
        .RORW(RORW),
        .S0(S0),
        .S1(S1),
        .S2(S2),
        .S3(S3),
        .S4(S4),
        .S5(S5));

initial
        begin
                // ------------- Current Time: 85ns
                #85;
                RESET = 1'b1;
                // -------------------------------------

                // ------------- Current Time: 285ns
                #200;
                RESET = 1'b0;
                // -------------------------------------

                // ------------- Current Time: 485ns
                #200;
                CLR = 1'b1;
                // -------------------------------------

                // ------------- Current Time: 1085ns
                #600;
                ADD = 1'b1;
                CLR = 1'b0;
                // -------------------------------------

                // ------------- Current Time: 2085ns
                #1000;
                ADD = 1'b0;
                SUB = 1'b1;
                // -------------------------------------

                // ------------- Current Time: 3085ns
                #1000;
                STORE = 1'b1;
                SUB = 1'b0;
                // -------------------------------------

                // ------------- Current Time: 4085ns
                #1000;
                BNZ = 1'b1;
                STORE = 1'b0;
                // -------------------------------------

                // ------------- Current Time: 5085ns
```

```
                        #1000;
                        ZERO = 1'b1;
                        // ------------------------------------

                        // ------------- Current Time: 5885ns
                        #800;
                        ZERO = 1'b0;
                        // ------------------------------------

                        // ------------- Current Time: 6085ns
                        #200;
                        BNZ = 1'b0;
                        // ------------------------------------
                end
endmodule
```

3 Experimental Results

[h]



Name	Value
S0	0
S1	0
S2	0
S3	0
S4	1
S5	0
CLK	0
CLR	0
RESET	0

Figure 4:

[h]

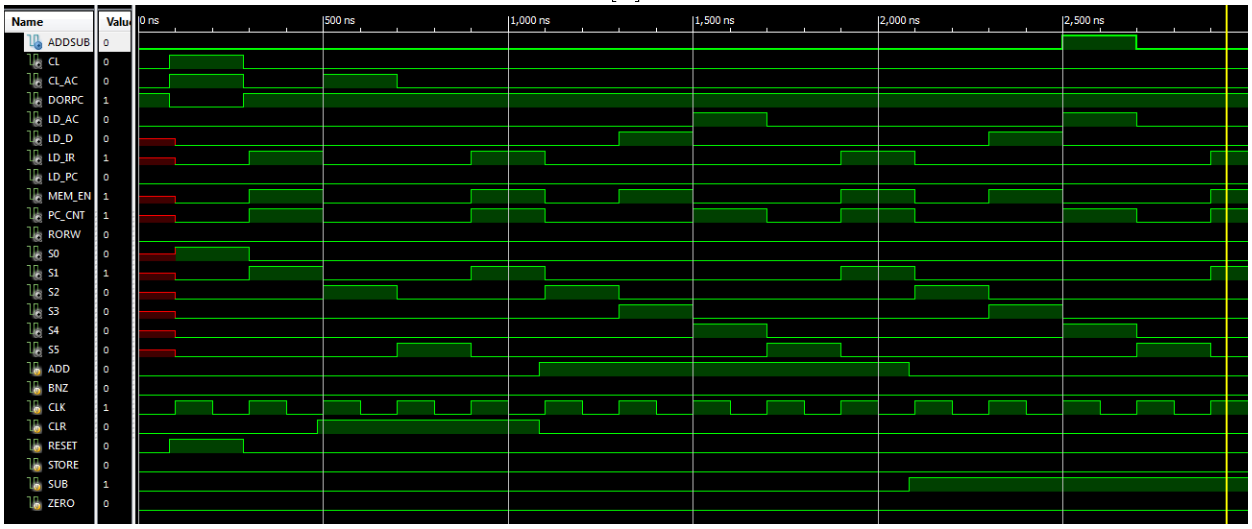


Figure 5:

4 Significance

5 Comments/Suggestions