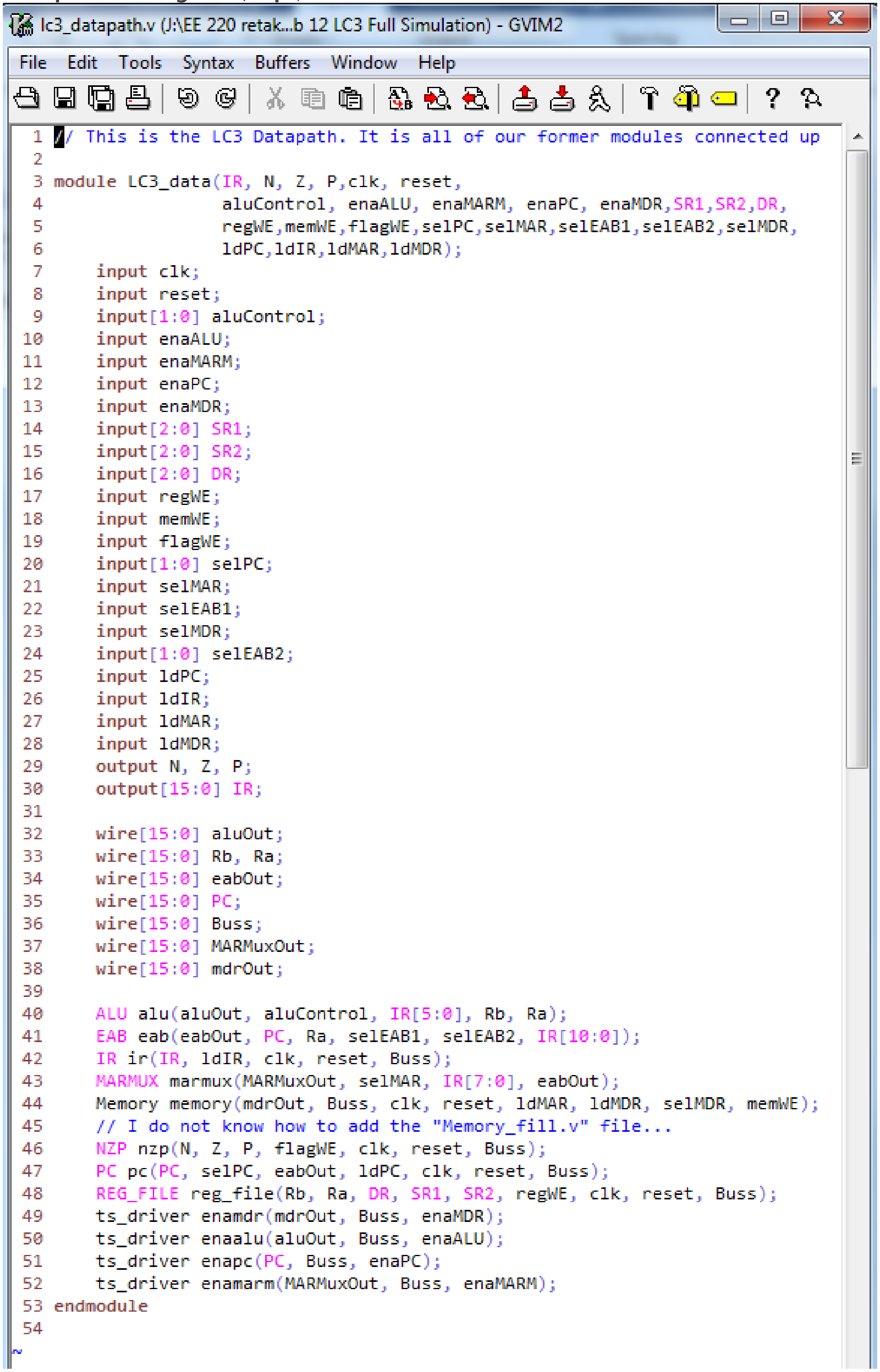
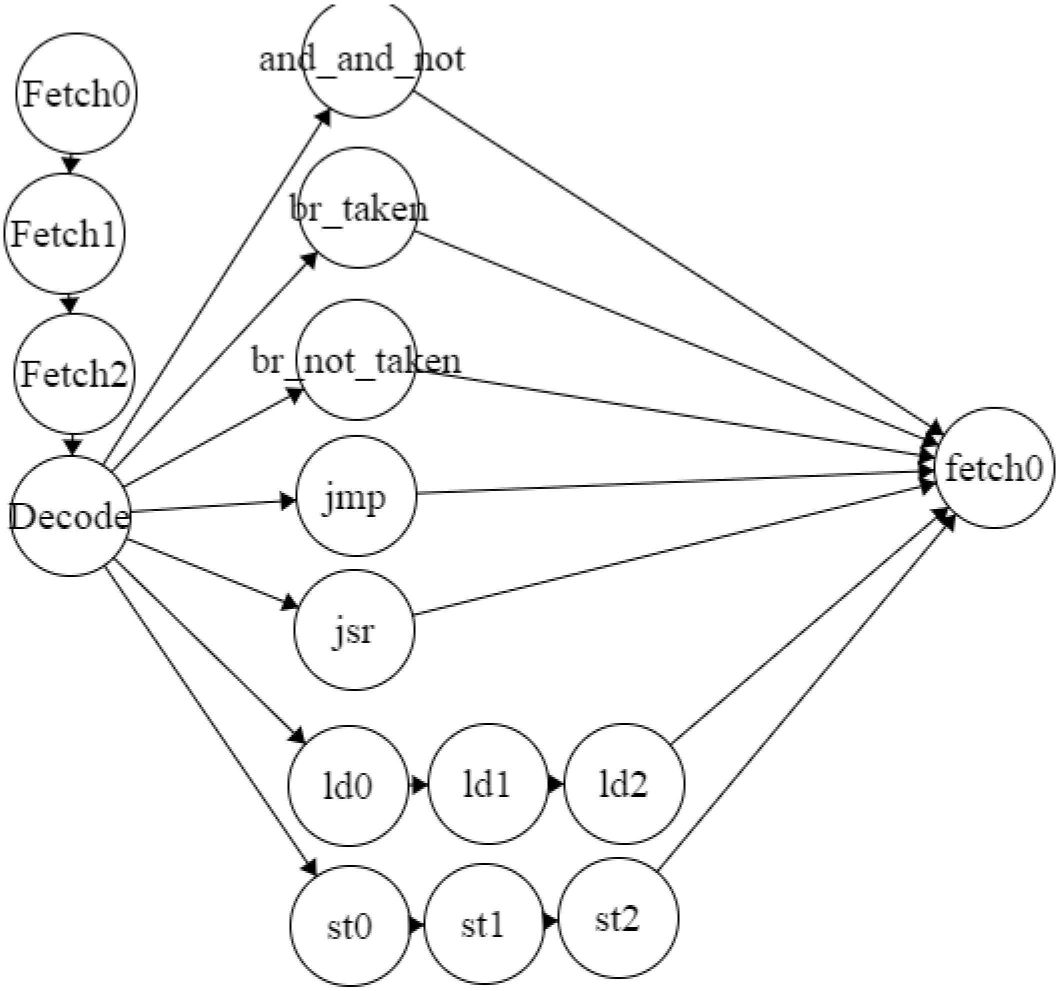
Taylor Cowley
EE 220
June 8 2016
Lab 11: LC3 Control AND Lab 12: LC3 Full Simulation

Datapath Verilog file (10pt)


lc3_datapath.v (J:\EE 220 retak...b 12 LC3 Full Simulation) - GVIM2
File
Edit
Tools
Syntax
Buffers
Window
Help

```verilog
// This is the LC3 Datapath. It is all of our former modules connected up

module LC3_data(IR, N, Z, P,clk, reset,
                aluControl, enaALU, enaMARM, enaPC, enaMDR,SR1,SR2,DR,
                regWE,memWE,flagWE,selPC,selMAR,selEAB1,selEAB2,selMDR,
                ldPC,ldIR,ldMAR,ldMDR);
    input clk;
    input reset;
    input[1:0] aluControl;
    input enaALU;
    input enaMARM;
    input enaPC;
    input enaMDR;
    input[2:0] SR1;
    input[2:0] SR2;
    input[2:0] DR;
    input regWE;
    input memWE;
    input flagWE;
    input[1:0] selPC;
    input selMAR;
    input selEAB1;
    input selMDR;
    input[1:0] selEAB2;
    input ldPC;
    input ldIR;
    input ldMAR;
    input ldMDR;
    output N, Z, P;
    output[15:0] IR;

    wire[15:0] aluOut;
    wire[15:0] Rb, Ra;
    wire[15:0] eabOut;
    wire[15:0] PC;
    wire[15:0] Buss;
    wire[15:0] MARMuxOut;
    wire[15:0] mdrOut;

    ALU alu(aluOut, aluControl, IR[5:0], Rb, Ra);
    EAB eab(eabOut, PC, Ra, selEAB1, selEAB2, IR[10:0]);
    IR ir(IR, ldIR, clk, reset, Buss);
    MARMUX marmux(MARMuxOut, selMAR, IR[7:0], eabOut);
    Memory memory(mdrOut, Buss, clk, reset, ldMAR, ldMDR, selMDR, memWE);
    // I do not know how to add the "Memory_fill.v" file...
    NZP nzp(N, Z, P, flagWE, clk, reset, Buss);
    PC pc(PC, selPC, eabOut, ldPC, clk, reset, Buss);
    REG_FILE reg_file(Rb, Ra, DR, SR1, SR2, regWE, clk, reset, Buss);
    ts_driver enamdr(mdrOut, Buss, enaMDR);
    ts_driver enaalu(aluOut, Buss, enaALU);
    ts_driver enapc(PC, Buss, enaPC);
    ts_driver enamarm(MARMuxOut, Buss, enaMARM);
endmodule

~
```

State Machine Diagram (6pt)



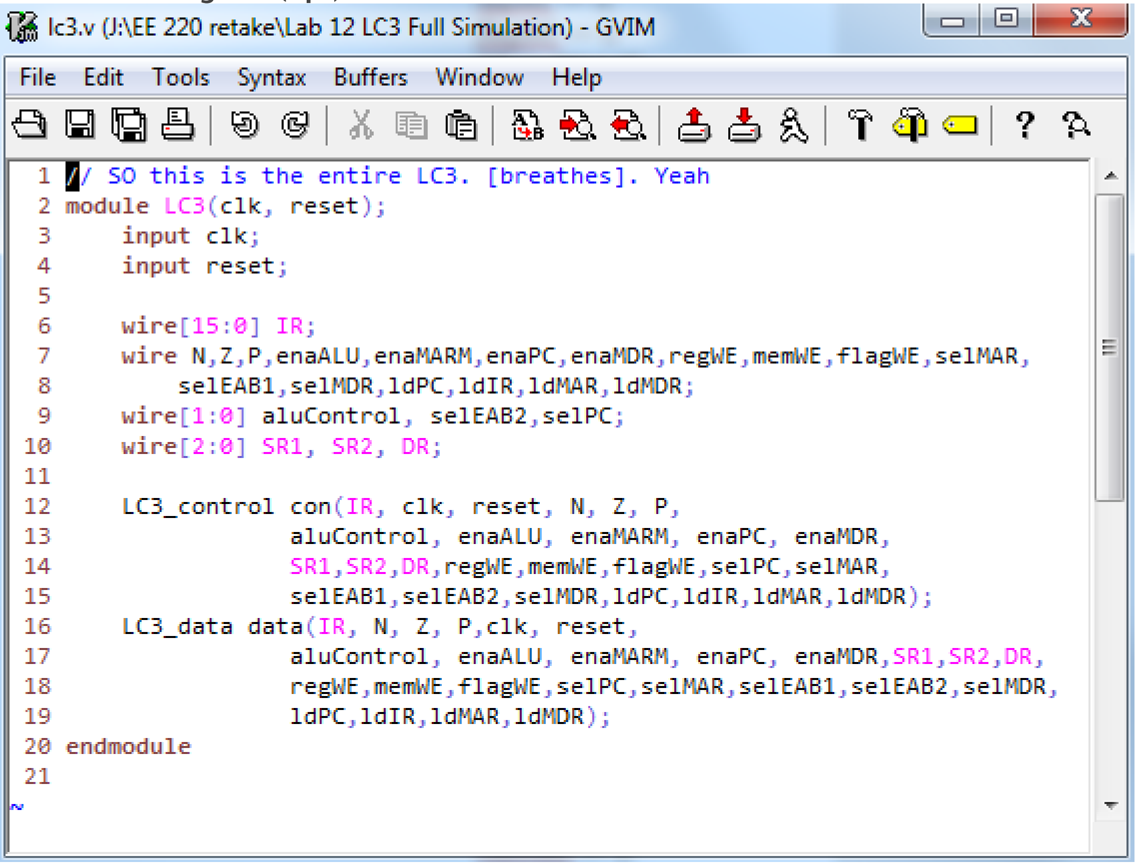
Controller Verilog file (7pt)

Sorry this is too long for screenshots

```
// This is the controller for the LC3.
// State machines with user-defined encoding. Oh goodie.
module LC3_control(IR, clk, reset, N, Z, P,
        aluControl, enaALU, enaMARM, enaPC,
        enaMDR,SR1,SR2,DR,regWE,memWE,
        flagWE,selPC,selMAR,selEAB1,selEAB2,
        selMDR,ldPC,ldIR,ldMAR,ldMDR);
        input[15:0] IR;
        input clk;              // These aren't outputs
        input reset;    // right?
        input N, Z, P;
        output reg[1:0] aluControl;
        output reg enaALU;
        output reg enaMARM;
        output reg enaPC;
        output reg enaMDR;
        output reg[2:0] SR1;
        output reg[2:0] SR2;
        output reg[2:0] DR;
        output reg regWE;
        output reg memWE;
        output reg flagWE;
        output reg[1:0] selPC;
        output reg selMAR;
        output reg selEAB1;
        output reg[1:0] selEAB2;
        output reg selMDR;
        output reg ldPC;
        output reg ldIR;
        output reg ldMAR;
        output reg ldMDR;

        // The state machine states
        parameter fetch0            = 20'd0;
        parameter fetch1            = 20'd1;
        parameter fetch2            = 20'd2;
        parameter decode            = 20'd3;
        parameter add_and_not       = 20'd4;
        parameter br_taken          = 20'd5;
        parameter br_not_taken      = 20'd6;
        parameter jmp               = 20'd7;
        parameter jsr               = 20'd8;
        parameter ld0               = 20'd11;
        parameter ld1               = 20'd12;
        parameter ld2               = 20'd13;
        parameter st0               = 20'd14;
        parameter st1               = 20'd15;
        parameter st2               = 20'd16;

        reg[19:0] current_state, next_state;
        always@(posedge clk or reset)
        begin
            if(reset)
                current_state <= fetch0;
            else
                current_state <= next_state;
        end

        always@(*)
        begin
            // Always start with the defaults
            aluControl <=0;
            enaALU <=0;
            enaMARM <=0;
            enaPC <=0;
            enaMDR <=0;
            SR1 <=0;
            SR2 <=0;
            DR <=0;
            regWE <=0;
            memWE <=0;
            flagWE <=0;
            selPC <=0;
            selMAR <=0;
            selEAB1 <=0;
            selEAB2 <=0;
            selMDR <=0;
            ldPC <=0;
            ldIR <=0;
            ldMAR <=0;
            ldMDR <=0;

            case(current_state)
                fetch0:
                begin
                    next_state <= fetch1;
                    enaPC <= 1;
                    ldMAR <= 1;
                end
```

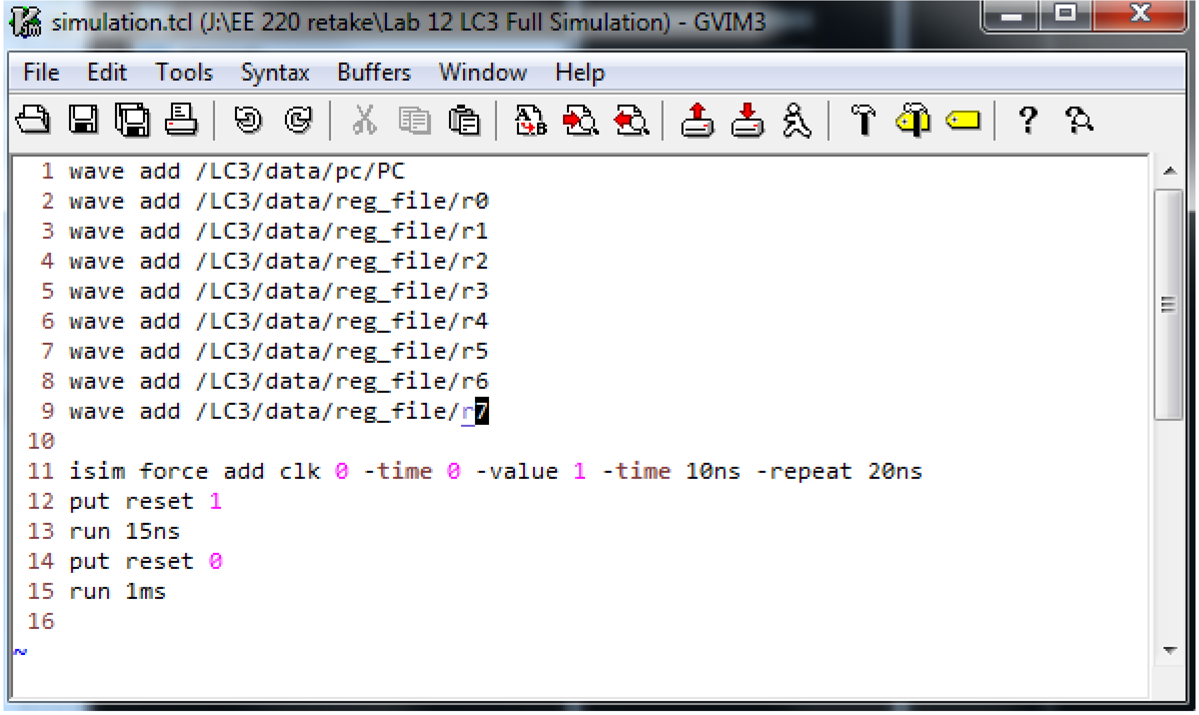
```
            fetch1:
            begin
                next_state <= fetch2;
                ldPC <= 1;
                ldMDR <= 1;
                selMDR <= 1;
                selPC <= 0;
            end
            fetch2:
            begin
                next_state <= decode;
                ldIR <= 1;
                enaMDR <= 1;
            end
            decode:
            begin
                case(IR[15:12])
                    4'b0001:
next_state <= add_and_not; // ADD
                    4'b0101:
next_state <= add_and_not; // AND
                    4'b1001:
next_state <= add_and_not; // NOT
                    4'b0000:
                    begin

    if((N*IR[11])+(Z*IR[10])+(P*IR[09]))

    next_state <= br_taken;
                        else

    next_state <= br_not_taken;
                    end
                    4'b1100:
next_state <= jmp;
                    4'b0100:
next_state <= jsr;
                    4'b0010:
next_state <= ld0;
                    4'b0011:
next_state <= st0;
                    default:
next_state <= fetch0;
                endcase
            end
            add_and_not:
            begin
                next_state <= fetch0;
                enaALU <= 1;
                regWE <= 1;
                flagWE <= 1;
                SR2 <= IR[2:0];
                SR1 <= IR[8:6]; // When
immediate, the ALU is automatic
                DR <= IR[11:9];
                case(IR[15:12])
                    4'b0001:
aluControl <= 2'b01; // ADD
                    4'b0101:
aluControl <= 2'b10; // AND
                    4'b1001:
aluControl <= 2'b11; // NOT
                endcase
            end
            br_taken:
            begin
                next_state <= fetch0;
                selPC <= 2'b01;
                selEAB1 <= 0;
                selEAB2 <= 2'b10;
                ldPC <= 1;
            end
            br_not_taken:
            begin
                next_state <= fetch0;
            end
            jmp:
            begin
                next_state <= fetch0;
                aluControl <= 2'b00;
                SR1 <= IR[8:6];
                selPC <= 2'b10;
                enaALU <= 1;
                ldPC <= 1;
            end
            jsr:
            begin
                // R7 <= pc
                // PC <= PC +
SEXT(PCoffset11);
                next_state <= fetch0;
                enaPC <= 1;
                DR <= 3'b111; // R7
                regWE <= 1;

                selEAB1 <= 0;
                selEAB2 <= 2'b11;
                selPC <= 2'b01;
                ldPC <= 1;
            end
            ld0:
            begin
                next_state <= ld1;
                selEAB1 <= 0;
                selEAB2 <= 2'b10;
                enaMARM <= 1;
                selMAR <= 0;
                ldMAR <= 1;
            end
            ld1:
            begin
                next_state <= ld2;
                selMDR <= 1;
                ldMDR <= 1;
            end
            ld2:
            begin
                next_state <= fetch0;
                enaMDR <= 1;
                regWE <= 1;
                flagWE <= 1;
                DR <= IR[11:9];
            end
            st0:
            begin
                next_state <= st1;
                selEAB1 <= 0;
                selEAB2 <= 2'b10;
                selMAR <= 0;
                enaMARM <= 1;
                ldMAR <= 1;
            end
            st1:
            begin
                next_state <= st2;
                aluControl <= 2'b00;
                enaALU <= 1;
                SR1 <= IR[11:9];
                selMDR <= 0;
                ldMDR <= 1;
            end
            st2:
            begin
                next_state <= fetch0;
                memWE <= 1;
            end
        endcase
    end
endmodule
```

Full LC3 Verilog file (7pt)



```verilog
// SO this is the entire LC3. [breathes]. Yeah
module LC3(clk, reset);
    input clk;
    input reset;

    wire[15:0] IR;
    wire N,Z,P,enaALU,enaMARM,enaPC,enaMDR,regWE,memWE,flagWE,selMAR,
        selEAB1,selMDR,ldPC,ldIR,ldMAR,ldMDR;
    wire[1:0] aluControl, selEAB2,selPC;
    wire[2:0] SR1, SR2, DR;

    LC3_control con(IR, clk, reset, N, Z, P,
                aluControl, enaALU, enaMARM, enaPC, enaMDR,
                SR1,SR2,DR,regWE,memWE,flagWE,selPC,selMAR,
                selEAB1,selEAB2,selMDR,ldPC,ldIR,ldMAR,ldMDR);
    LC3_data data(IR, N, Z, P,clk, reset,
                aluControl, enaALU, enaMARM, enaPC, enaMDR,SR1,SR2,DR,
                regWE,memWE,flagWE,selPC,selMAR,selEAB1,selEAB2,selMDR,
                ldPC,ldIR,ldMAR,ldMDR);
endmodule

~
```

Master TCL file (2pt)


simulation.tcl (J:\EE 220 retake\Lab 12 LC3 Full Simulation) - GVIM3
File
Edit
Tools
Syntax
Buffers
Window
Help

```tcl
wave add /LC3/data/pc/PC
wave add /LC3/data/reg_file/r0
wave add /LC3/data/reg_file/r1
wave add /LC3/data/reg_file/r2
wave add /LC3/data/reg_file/r3
wave add /LC3/data/reg_file/r4
wave add /LC3/data/reg_file/r5
wave add /LC3/data/reg_file/r6
wave add /LC3/data/reg_file/r7

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
put reset 1
run 15ns
put reset 0
run 1ms

~
```

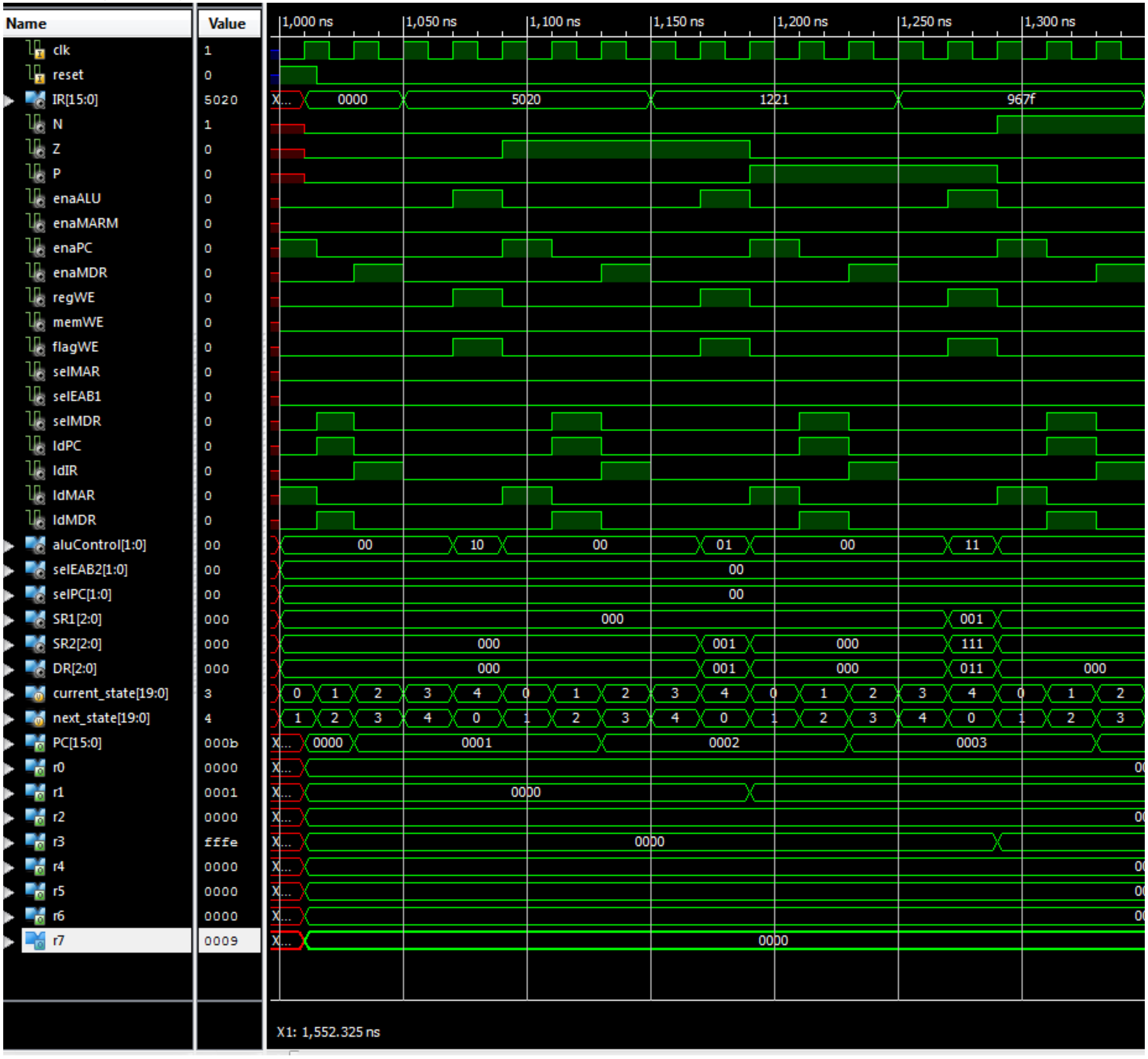
All Simulation waveforms(18pt)

See below

Anomalies (bugs, problems, and suggestions)(5pts possible)

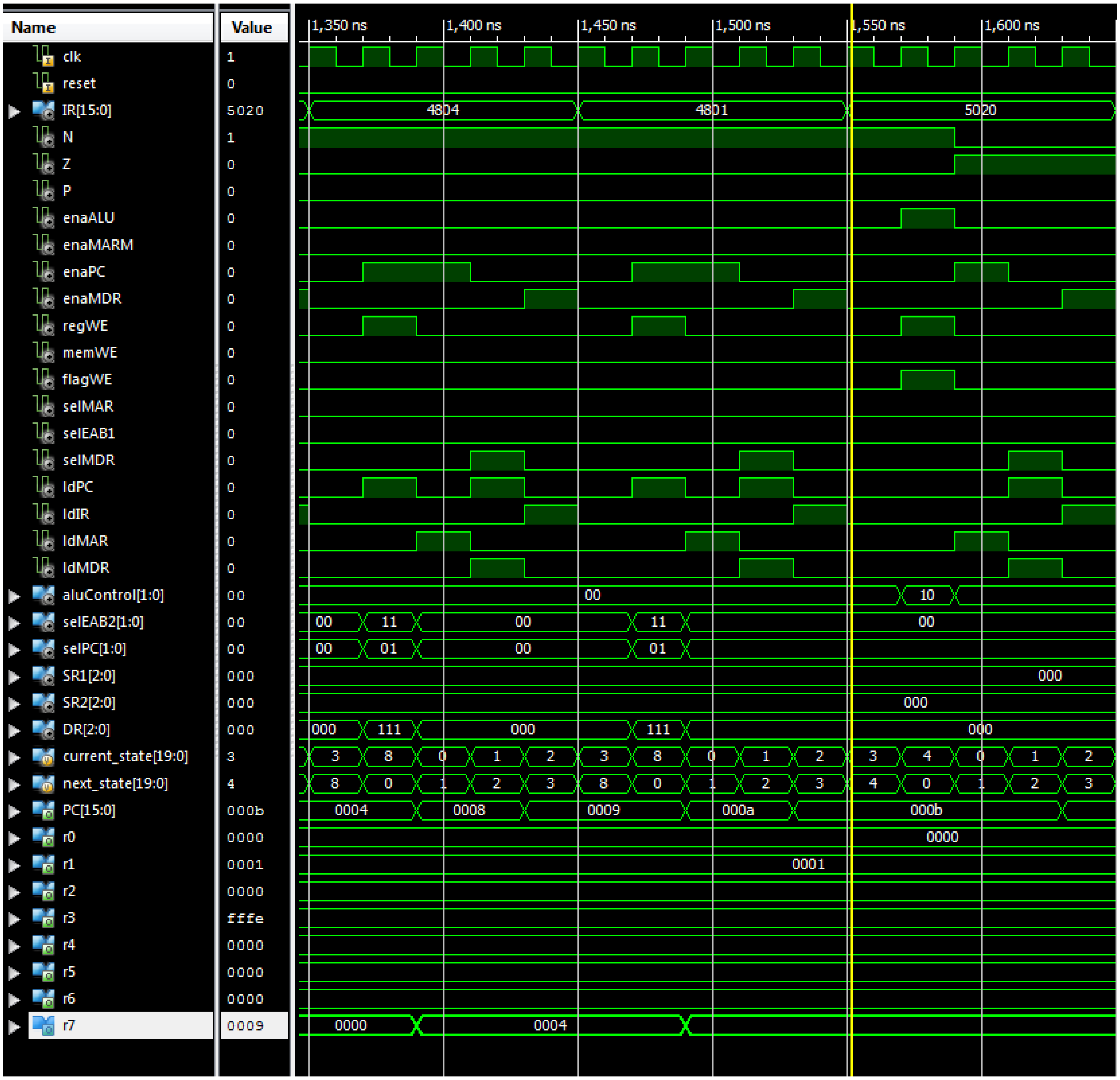
I had major issues with the state machine. Inside of my always block, I was using assign statements instead of <= statements, and it was causing infinite loops, so I had it run on the posedge clk instead of *, which made everything run a clock cycle slow. I had to change the sensitivity list to * again and use <=. Took forever to figure out…

Get ready for 8 pictures of waveforms.

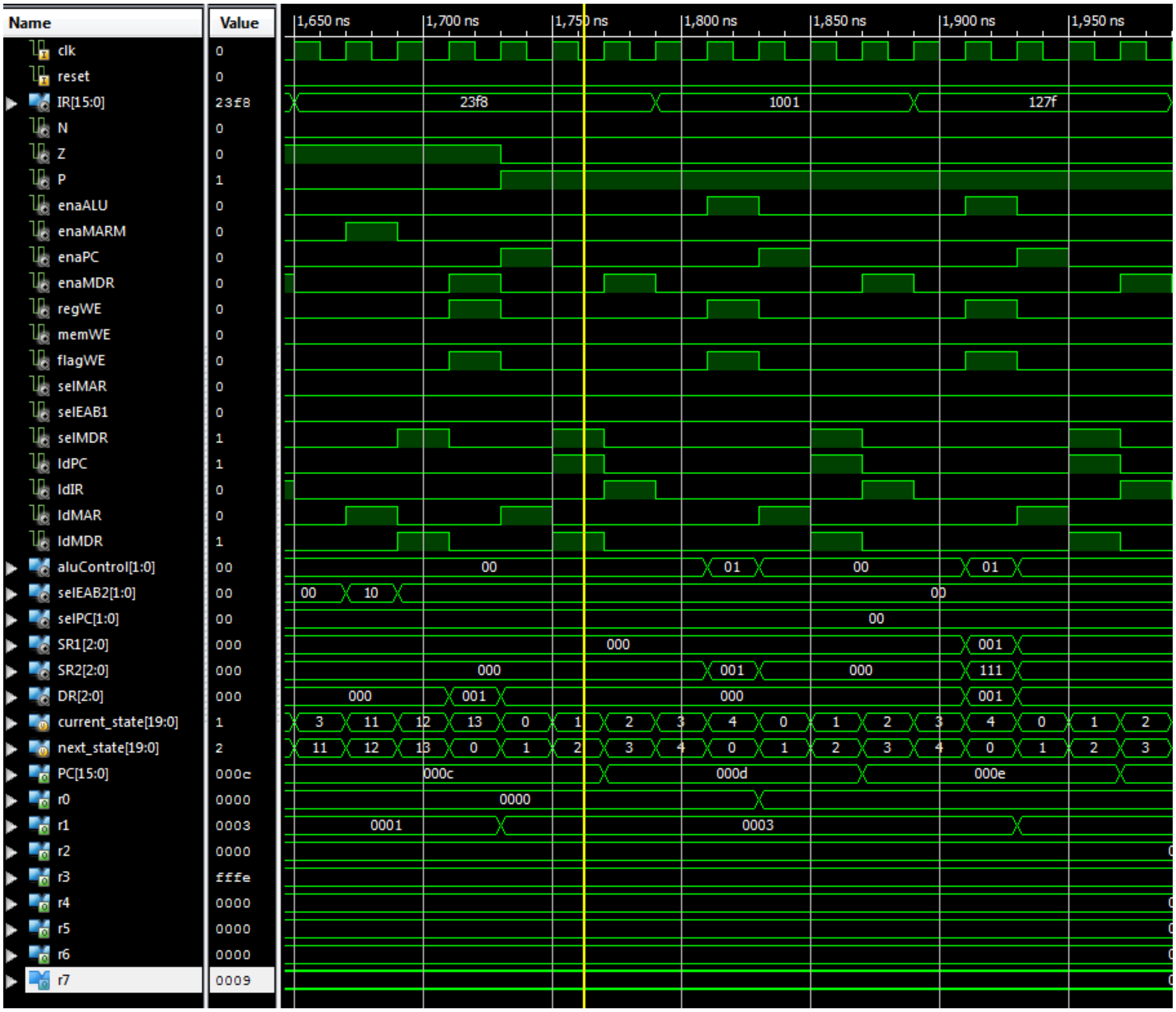


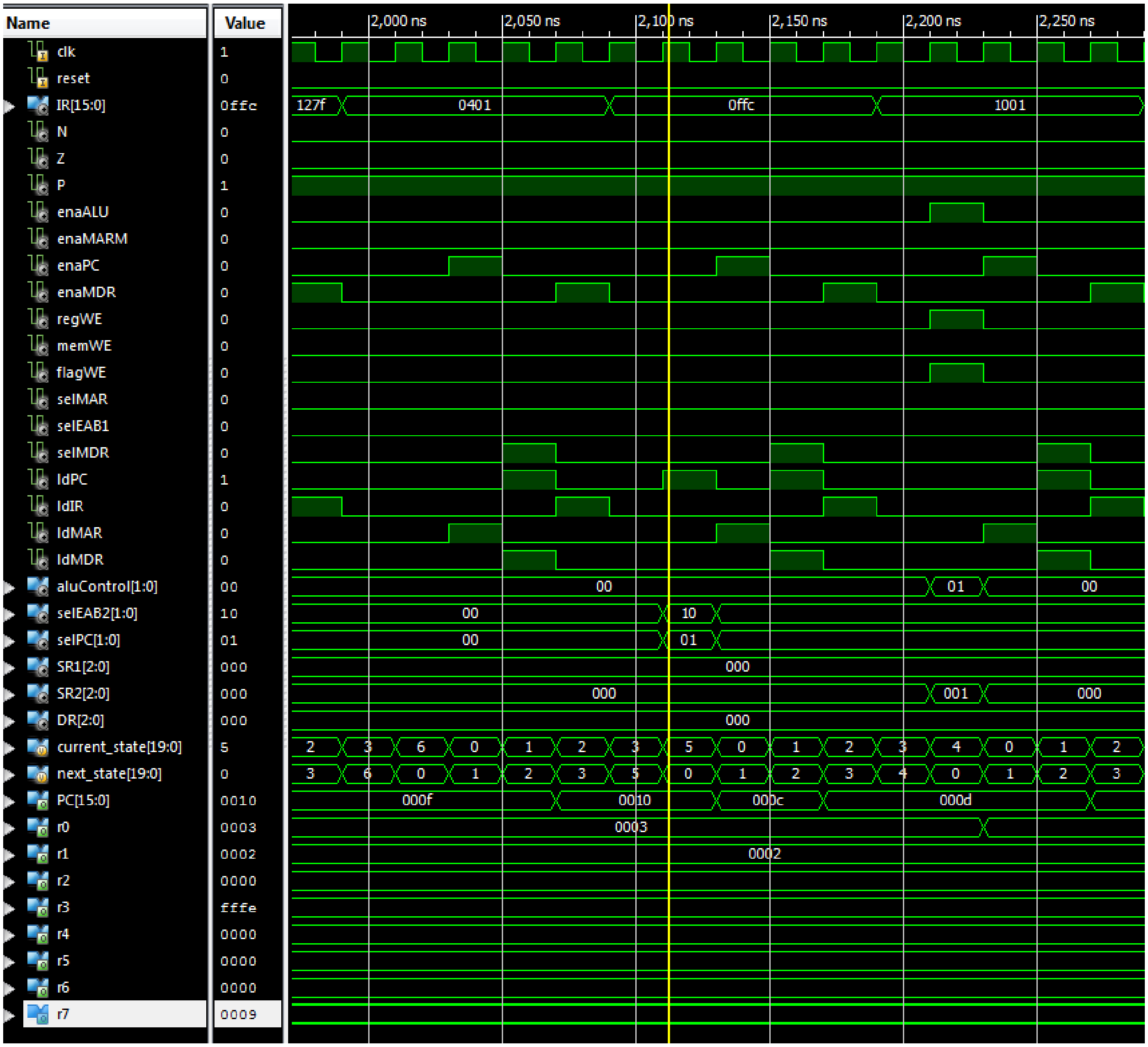
Name	Value
clk	1
reset	0
IR[15:0]	5020
N	1
Z	0
P	0
enaALU	0
enaMARM	0
enaPC	0
enaMDR	0
regWE	0
memWE	0
flagWE	0
selMAR	0
selEAB1	0
selMDR	0
ldPC	0
ldIR	0
ldMAR	0
ldMDR	0
aluControl[1:0]	00
selEAB2[1:0]	00
selPC[1:0]	00
SR1[2:0]	000
SR2[2:0]	000
DR[2:0]	000
current_state[19:0]	3
next_state[19:0]	4
PC[15:0]	000b
r0	0000
r1	0001
r2	0000
r3	fffe
r4	0000
r5	0000
r6	0000
r7	0009

X1: 1,552.325 ns

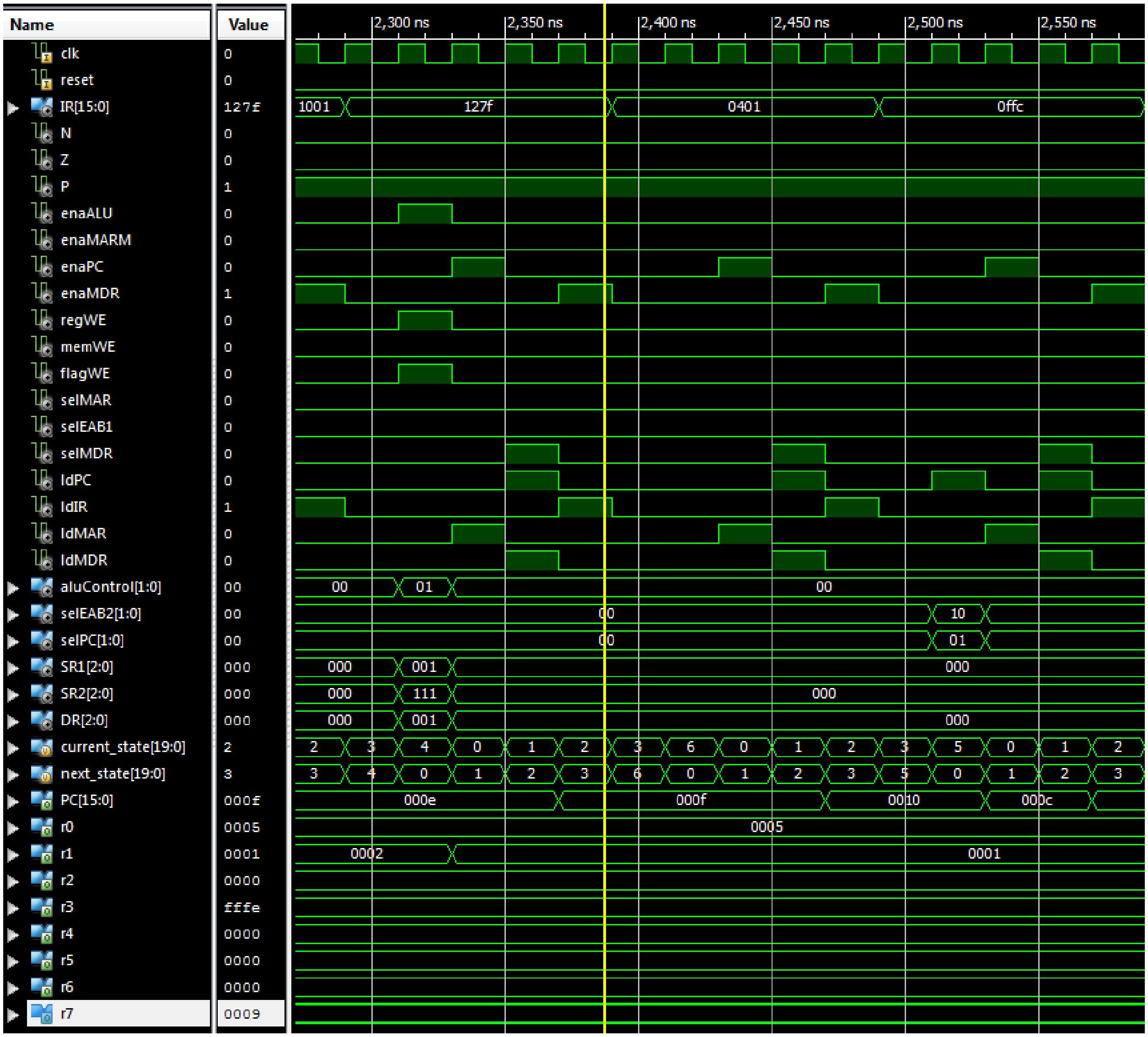


Name	Value
clk	1
reset	0
IR[15:0]	5020
N	1
Z	0
P	0
enaALU	0
enaMARM	0
enaPC	0
enaMDR	0
regWE	0
memWE	0
flagWE	0
selMAR	0
selEAB1	0
selMDR	0
ldPC	0
ldIR	0
ldMAR	0
ldMDR	0
aluControl[1:0]	00
selEAB2[1:0]	00
selPC[1:0]	00
SR1[2:0]	000
SR2[2:0]	000
DR[2:0]	000
current_state[19:0]	3
next_state[19:0]	4
PC[15:0]	000b
r0	0000
r1	0001
r2	0000
r3	fffe
r4	0000
r5	0000
r6	0000
r7	0009

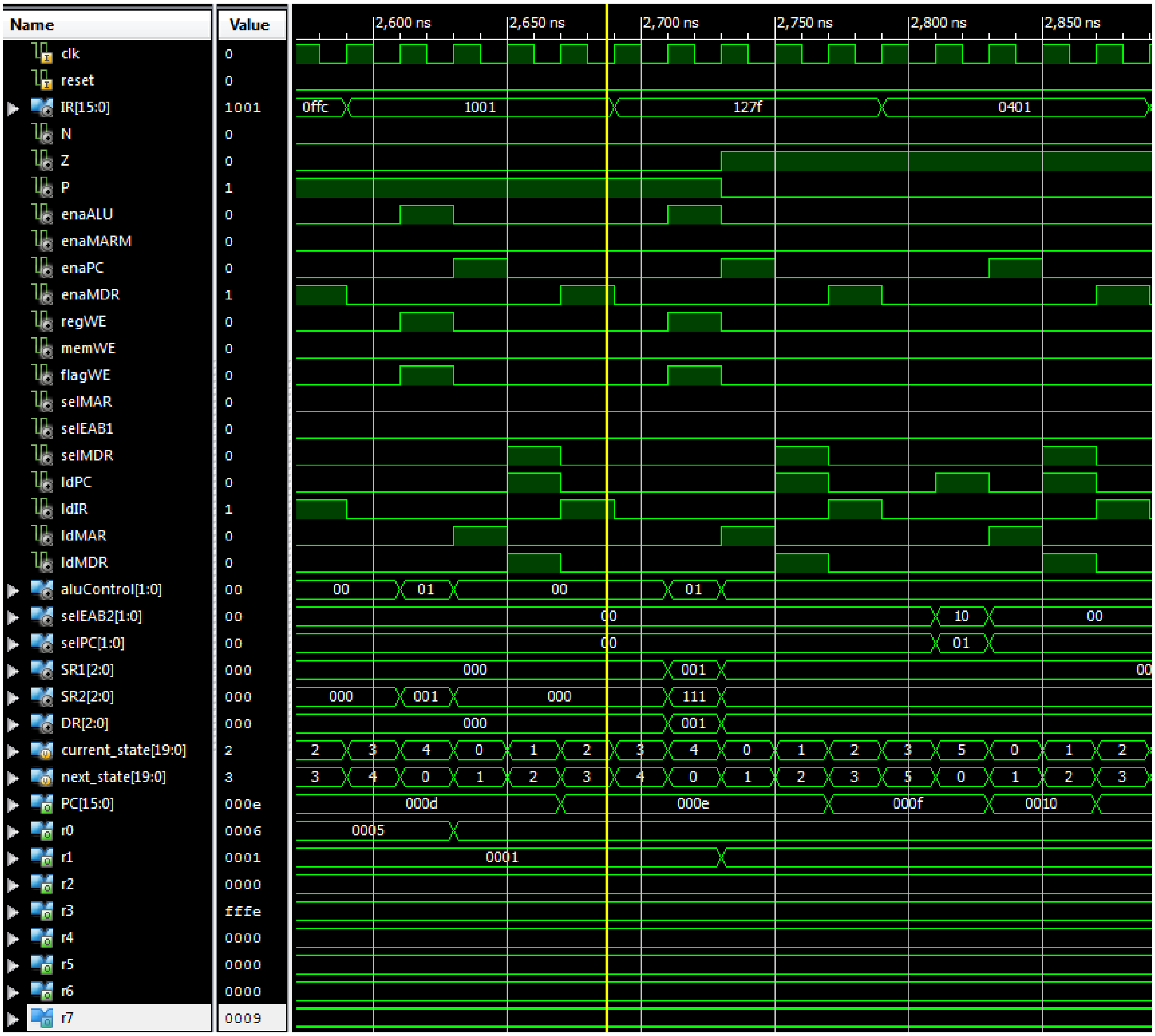

Name
Value
1,650 ns
1,700 ns
1,750 ns
1,800 ns
1,850 ns
1,900 ns
1,950 ns
clk 0
reset 0
IR[15:0] 23f8
23f8
1001
127f
N 0
Z 0
P 1
enaALU 0
enaMARM 0
enaPC 0
enaMDR 0
regWE 0
memWE 0
flagWE 0
selMAR 0
selEAB1 0
selMDR 1
ldPC 1
ldIR 0
ldMAR 0
ldMDR 1
aluControl[1:0] 00
00
01
00
01
selEAB2[1:0] 00
00
10
00
selPC[1:0] 00
00
SR1[2:0] 000
000
001
SR2[2:0] 000
000
001
000
111
DR[2:0] 000
000
001
000
001
current_state[19:0] 1
3
11
12
13
0
1
2
3
4
0
1
2
3
4
0
1
2
next_state[19:0] 2
11
12
13
0
1
2
3
4
0
1
2
3
4
0
1
2
3
PC[15:0] 000c
000c
000d
000e
r0 0000
0000
r1 0003
0001
0003
r2 0000
r3 fffe
r4 0000
r5 0000
r6 0000
r7 0009



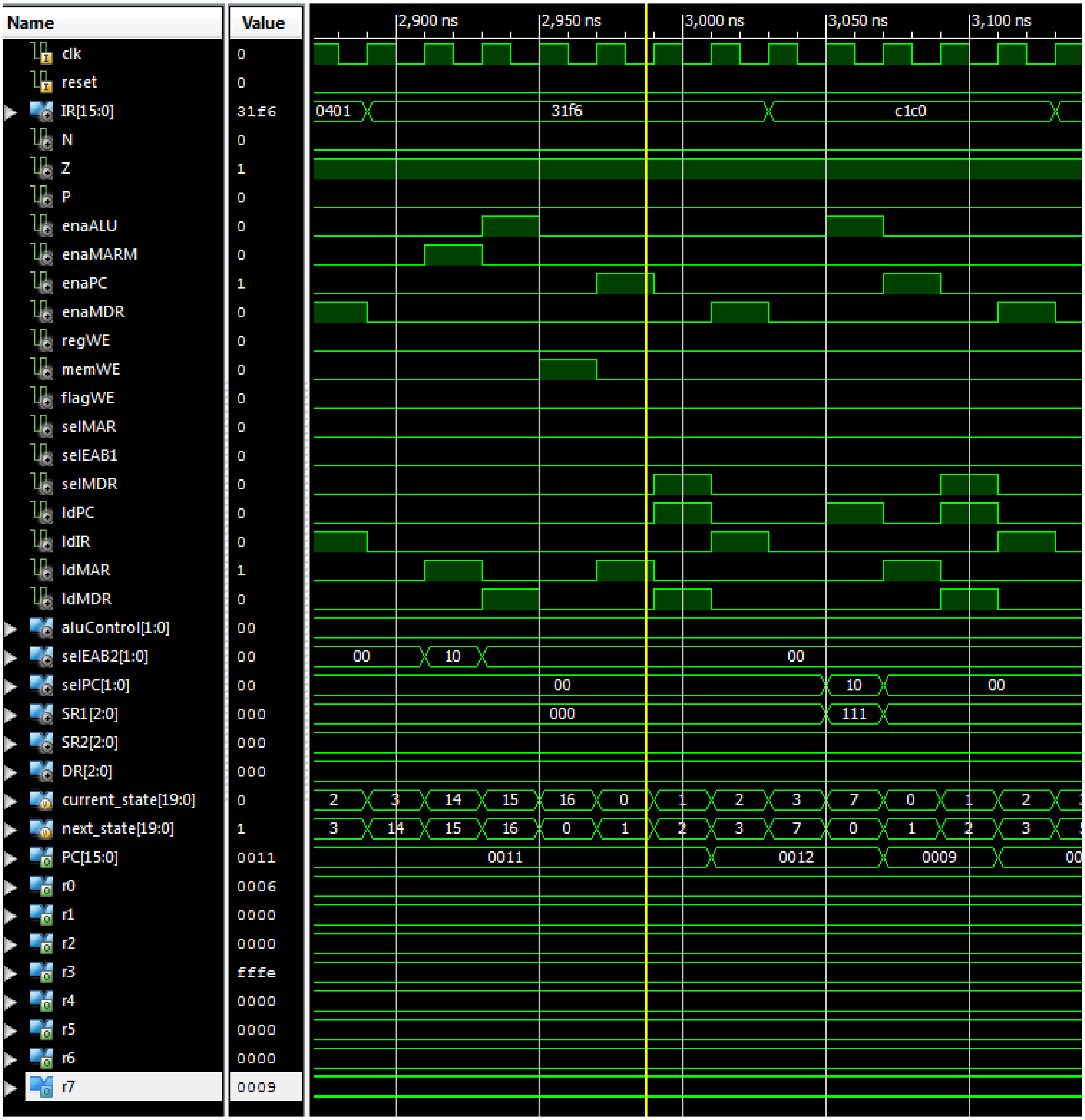
Name	Value
clk	1
reset	0
IR[15:0]	0ffc
N	0
Z	0
P	1
enaALU	0
enaMARM	0
enaPC	0
enaMDR	0
regWE	0
memWE	0
flagWE	0
selMAR	0
selEAB1	0
selMDR	0
ldPC	1
ldIR	0
ldMAR	0
ldMDR	0
aluControl[1:0]	00
selEAB2[1:0]	10
selPC[1:0]	01
SR1[2:0]	000
SR2[2:0]	000
DR[2:0]	000
current_state[19:0]	5
next_state[19:0]	0
PC[15:0]	0010
r0	0003
r1	0002
r2	0000
r3	fffe
r4	0000
r5	0000
r6	0000
r7	0009



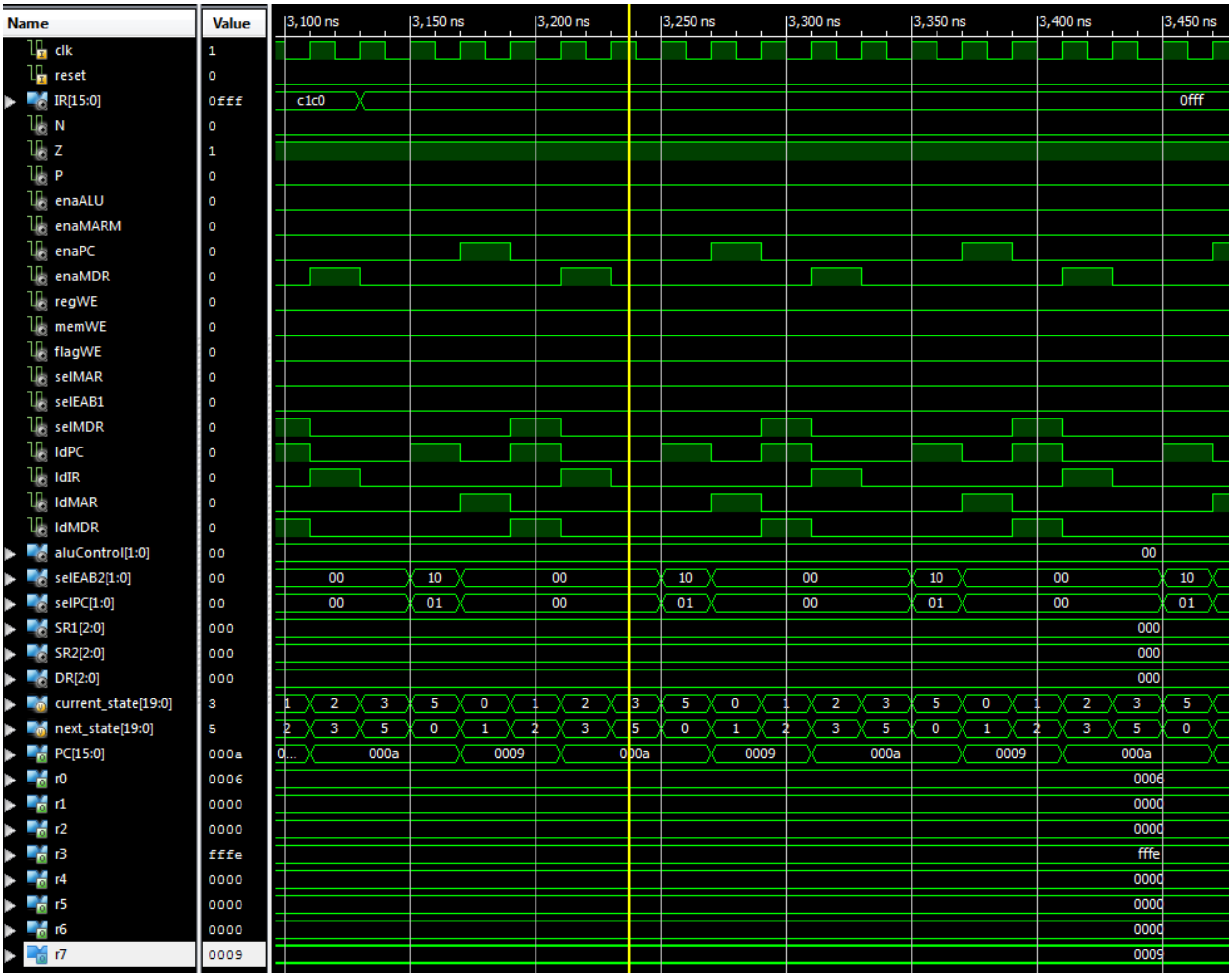
Name	Value
clk	0
reset	0
IR[15:0]	127f
N	0
Z	0
P	1
enaALU	0
enaMARM	0
enaPC	0
enaMDR	1
regWE	0
memWE	0
flagWE	0
selMAR	0
selEAB1	0
selMDR	0
ldPC	0
ldIR	1
ldMAR	0
ldMDR	0
aluControl[1:0]	00
selEAB2[1:0]	00
selPC[1:0]	00
SR1[2:0]	000
SR2[2:0]	000
DR[2:0]	000
current_state[19:0]	2
next_state[19:0]	3
PC[15:0]	000f
r0	0005
r1	0001
r2	0000
r3	fffe
r4	0000
r5	0000
r6	0000
r7	0009



Name	Value
clk	0
reset	0
IR[15:0]	1001
N	0
Z	0
P	1
enaALU	0
enaMARM	0
enaPC	0
enaMDR	1
regWE	0
memWE	0
flagWE	0
selMAR	0
selEAB1	0
selMDR	0
ldPC	0
ldIR	1
ldMAR	0
ldMDR	0
aluControl[1:0]	00
selEAB2[1:0]	00
selPC[1:0]	00
SR1[2:0]	000
SR2[2:0]	000
DR[2:0]	000
current_state[19:0]	2
next_state[19:0]	3
PC[15:0]	000e
r0	0006
r1	0001
r2	0000
r3	fffe
r4	0000
r5	0000
r6	0000
r7	0009



Name	Value
clk	0
reset	0
IR[15:0]	31f6
N	0
Z	1
P	0
enaALU	0
enaMARM	0
enaPC	1
enaMDR	0
regWE	0
memWE	0
flagWE	0
selMAR	0
selEAB1	0
selMDR	0
ldPC	0
ldIR	0
ldMAR	1
ldMDR	0
aluControl[1:0]	00
selEAB2[1:0]	00
selPC[1:0]	00
SR1[2:0]	000
SR2[2:0]	000
DR[2:0]	000
current_state[19:0]	0
next_state[19:0]	1
PC[15:0]	0011
r0	0006
r1	0000
r2	0000
r3	fffe
r4	0000
r5	0000
r6	0000
r7	0009



Name	Value
clk	1
reset	0
IR[15:0]	0fff
N	0
Z	1
P	0
enaALU	0
enaMARM	0
enaPC	0
enaMDR	0
regWE	0
memWE	0
flagWE	0
selMAR	0
selEAB1	0
selMDR	0
ldPC	0
ldIR	0
ldMAR	0
ldMDR	0
aluControl[1:0]	00
selEAB2[1:0]	00
selPC[1:0]	00
SR1[2:0]	000
SR2[2:0]	000
DR[2:0]	000
current_state[19:0]	3
next_state[19:0]	5
PC[15:0]	000a
r0	0006
r1	0000
r2	0000
r3	fffe
r4	0000
r5	0000
r6	0000
r7	0009