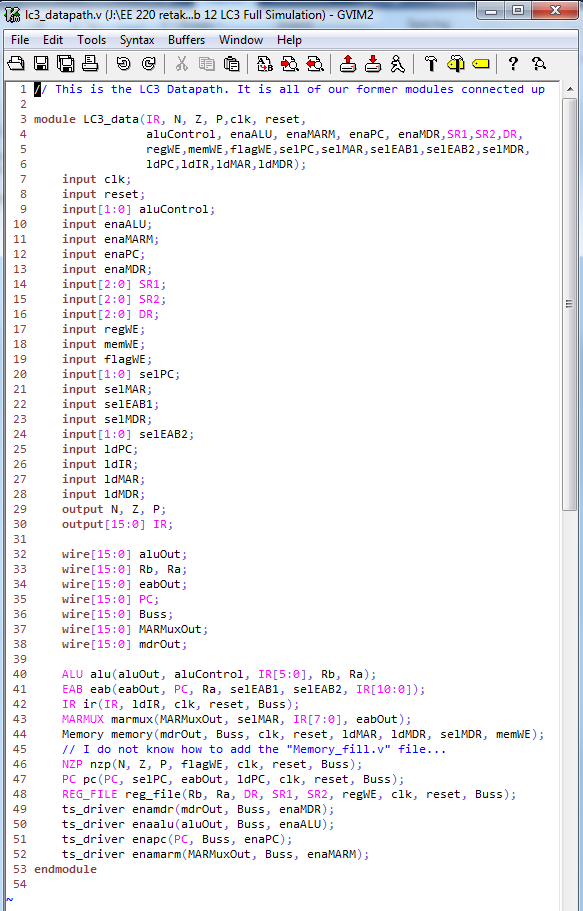
Taylor Cowley

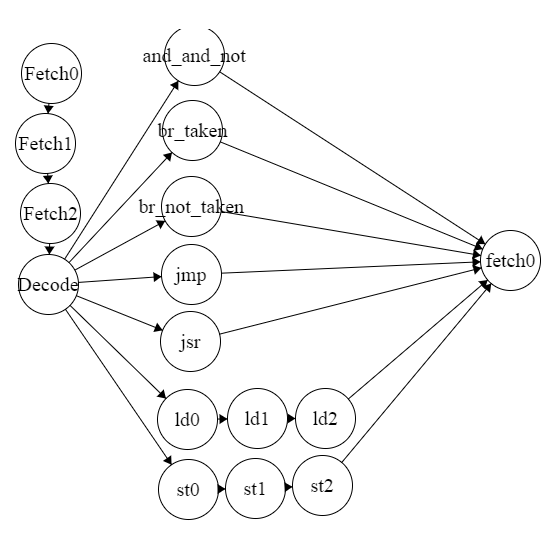
EE 220

June 8 2016

Lab 11: LC3 Control AND Lab 12: LC3 Full Simulation

**Datapath Verilog file (10pt)**



**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;

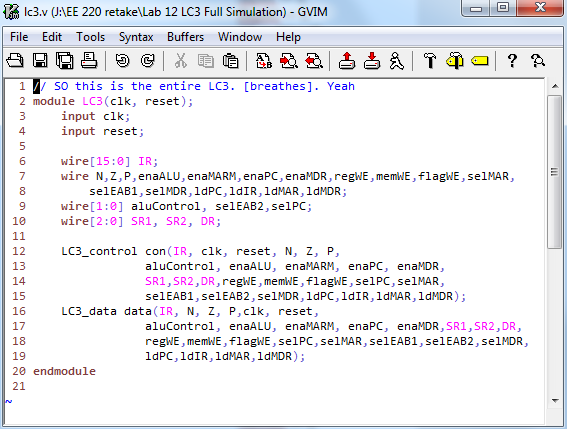
end

endcase

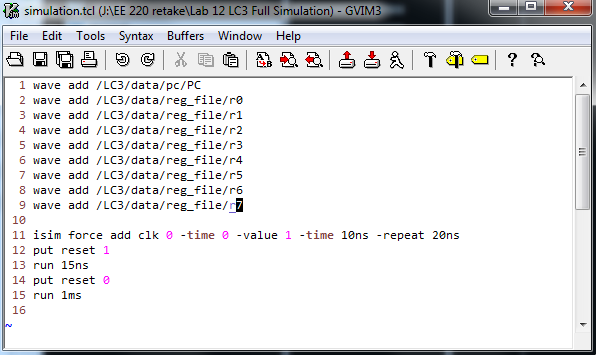
end

endmodule

**Full LC3 Verilog file (7pt)**



**Master TCL file (2pt)**



**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.**

