EE 309/EE 337
LC3B MICROPROCESSOR DESIGN REPORT

October 31, 2014

Group number - 20

Akhil Shetty - 12D070013

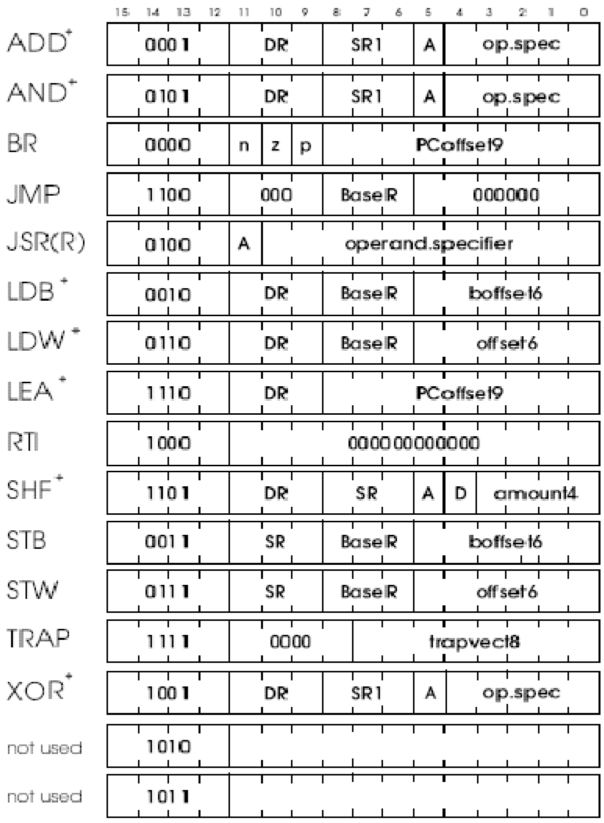
Ashish Goyal - 12D070051

Kush Motwani - 12D070018

1 Introduction

A 8-bit RISC microprocessor is implemented on FPGA. The ISA used is the LC3b ISA, an ISA which is used for academic purposes all over the world.
In the RISC design implemented, we have used multi-cycle implementation, as different instructions are inherently of varying complexity, and thus would need different amounts of time for implementation. Multi cycle implementation thus ensures that each instruction gets over in the least amount of time.
Also, the datapath in case of single cycle implementation will be more complex, as we have seen in class. Such a complex datapath would also use a lot of FPGA's resources.

2 Instructions Implemented



	15 14 13 12					
ADD$^+$	0001	DR	SR1	A	op.spec	
AND$^+$	0101	DR	SR1	A	op.spec	
BR	0000	n	z	p	PCoffset9	
JMP	1100	000	BaseR	000000		
JSR(R)	0100	A	operand.specifier			
LDB$^+$	0010	DR	BaseR	boffset6		
LDW$^+$	0110	DR	BaseR	offset6		
LEA$^+$	1110	DR	PCoffset9			
RTI	1000	000000000000				
SHF$^+$	1101	DR	SR	A	D	amount4
STB	0011	SR	BaseR	boffset6		
STW	0111	SR	BaseR	offset6		
TRAP	1111	0000	trapvect8			
XOR$^+$	1001	DR	SR1	A	op.spec	
not used	1010					
not used	1011					

Figure 1: Format for the instructions of the 16 LC3b opcodes. NOTE: + indicates instruction that modify condition codes.

3 Datapath

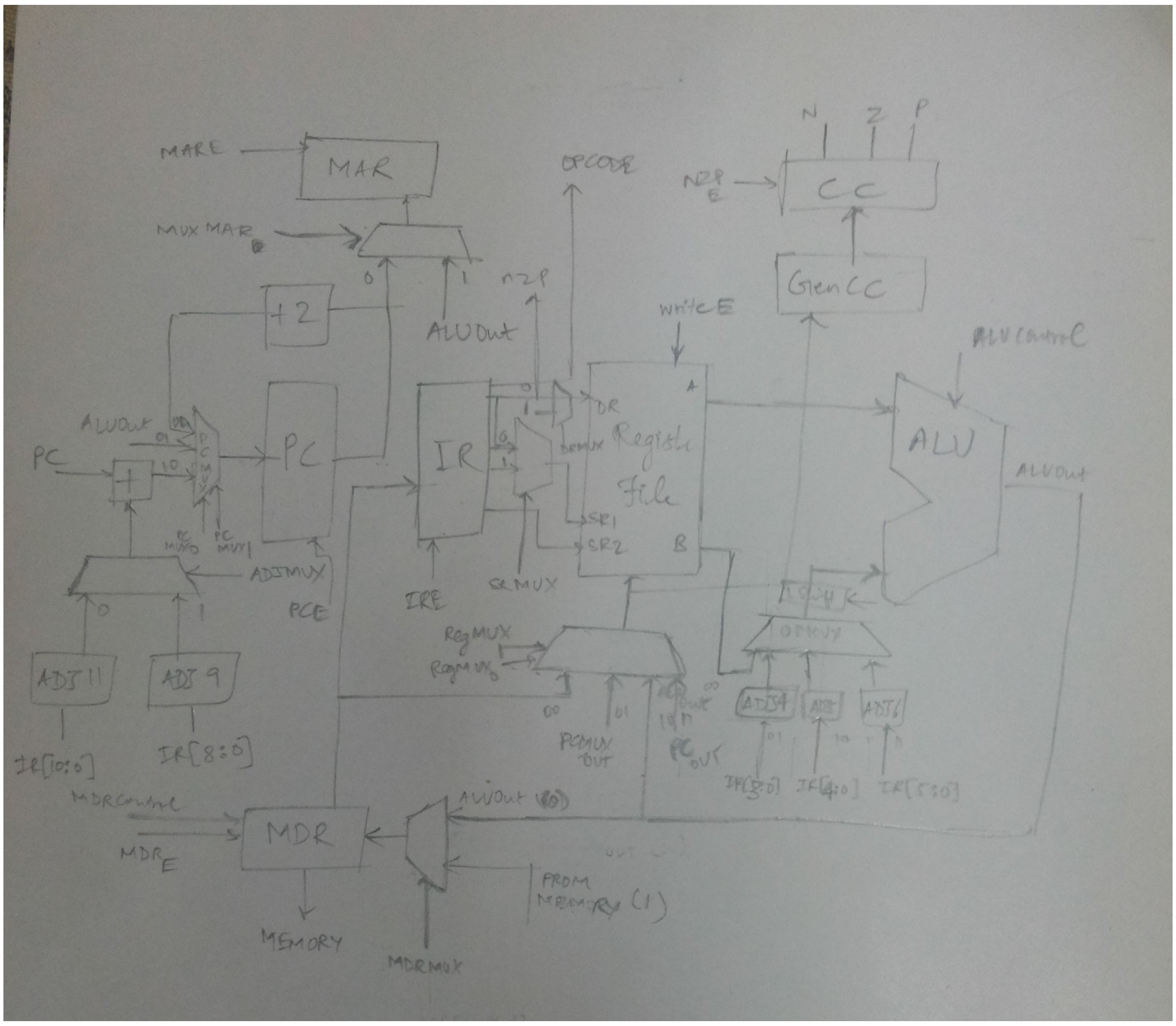


Figure 2: Datapath used.

4 FSM CONTROLLER

The FSM based controller gives the next state logic.

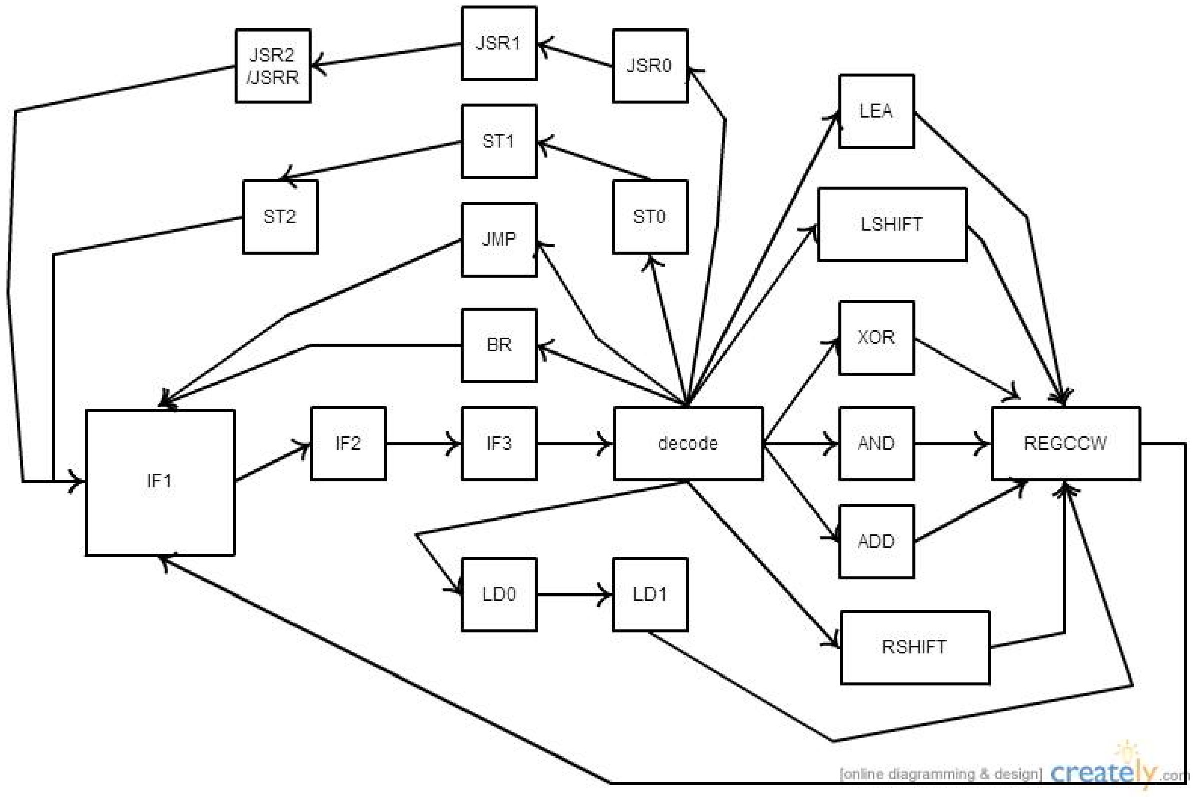


Figure 3: FSM Controller.

5 Control Signals

The control signals drive the different elements of the datapath. These control signals depend on the state of the FSM controller.
The code for the control path can be seen. In the code, for each state, it can be seen which control signals are active, for each state.
The control signals used are:

No	Signal Name	Purpose
1	marE	Enabling MAR
2	marMUX	Choose between PC and ALUout
3	pcMUX	Choose between PC+2,ALUout, and IR offset
4	pcE	Enable PC
5	adjMUX	Choose between IR9 and IR11
6	regMUX	Choose between ALUout, PCout, and MDRout for input to regfile.
7	mdrCon	Differentiate between 8 bit and 16 bit.
8	mdrMUX	Choose between ALUcon and Memory
9	mdrE	Enabling MDR
10	opMUX	Choose second ALU input
11	aluCon	ALU control
12	srMUX	Choose for input to SR1
13	drMUX	Choose for input to DR
14	regWriteE	Enabling writing in register
15	nzpE	Set condition code
16	irE	Enable IR
17	opLSHF	For shifted left-input to ALU

6 Simulations on MODELSIM

To test the datapath and the control path, test-benches code were written, which were executed in MODELSIM. The codes can be found in the code directory. To test the entire design, some instructions were hard coded in the memory, like a 'hex-file', and the code was then run on MODELSIM. The code ran as expected. The hardcoded instructions and data are:

```
assign q[40] = 8'b11110000;
assign q[41] = 8'b00000000;
assign q[42] = 8'b00000000;
assign q[43] = 8'b00001111;
assign q[1] = 8'b01100001; //ldw 40
assign q[0] = 8'b10010100;
assign q[3] = 8'b01100011; //ldw 42
assign q[2] = 8'b10010101;
assign q[5] = 8'b00010100; //add r2 = r1+r0
assign q[4] = 8'b00000001;
//assign q[7] = 8'b01110101;
//assign q[6] = 8'b10010110;
//assign q[9] = 8'b01100001;
//assign q[8] = 8'b10010110;
assign q[7] = 8'b00000010;
assign q[6] = 8'b00000011; //branch n
assign q[15] = 8'b01110101; //stw
assign q[14] = 8'b10010111;
assign q[17] = 8'b11000001;
assign q[16] = 8'b10000000;
```

The MODELSIM simulation result can be seen on the next page.

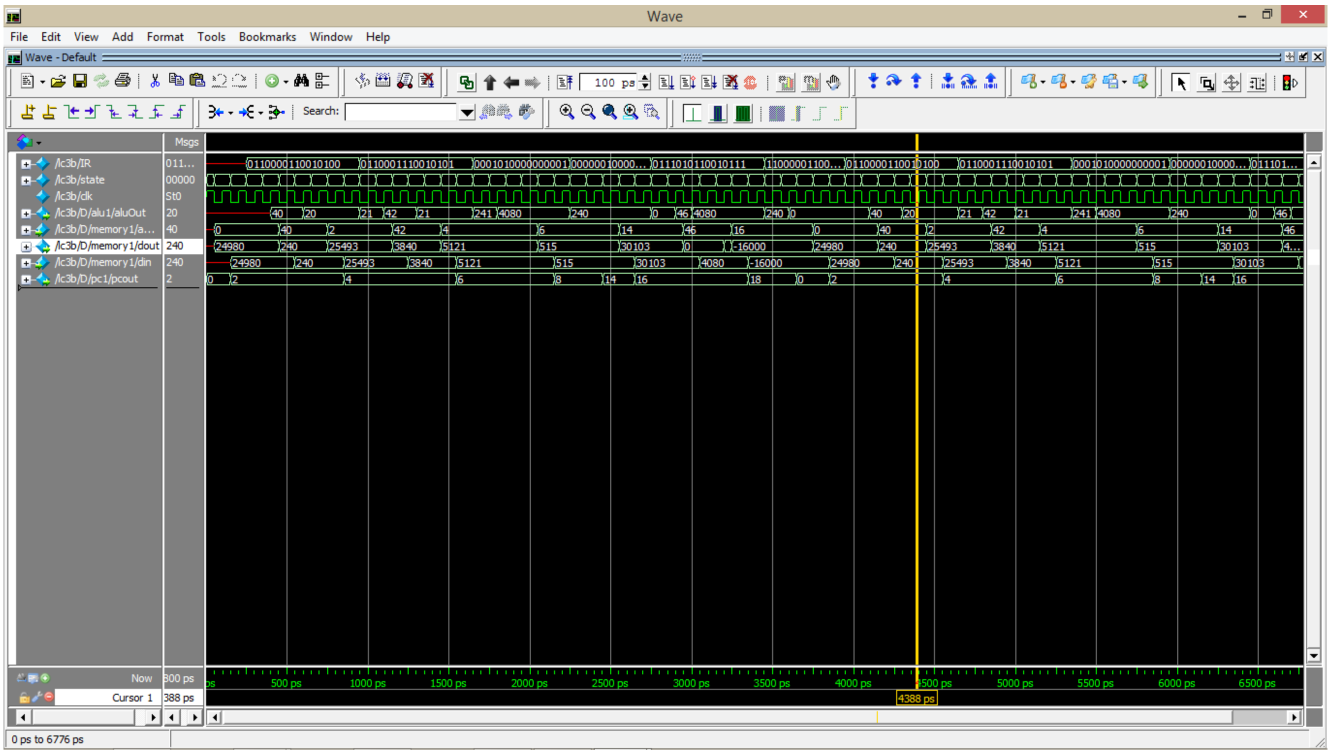


Figure 4: Simulation Waveforms.

7 Testing on FPGA

The code was burned on an Altera DE0-nano board, having a Cyclone-IV FPGA. As asked in the assignment, the signal-tap tool was used to get the states of the various signals, which were either the input, or the output of any of the modules. The clock was given using a pushbutton, so that the progress of the program can be observed properly. The output was as in MODELSIM, as expected.
The code directory for the entire QUARTUS project, which includes all the Verilog files, can be downloaded from this link:- http://home.iitb.ac.in/ goyal.ag/resources/lc3bfinal.zip