EE 337 / EE 309 - Microprocessor Design Report

OJAS APOORVA KANHERE
12D070002

PRAVEEN AGRAWAL
12D020030

KOTWAL ALANKAR SHASHIKANT
12D070010

October 31, 2014

1 Introduction

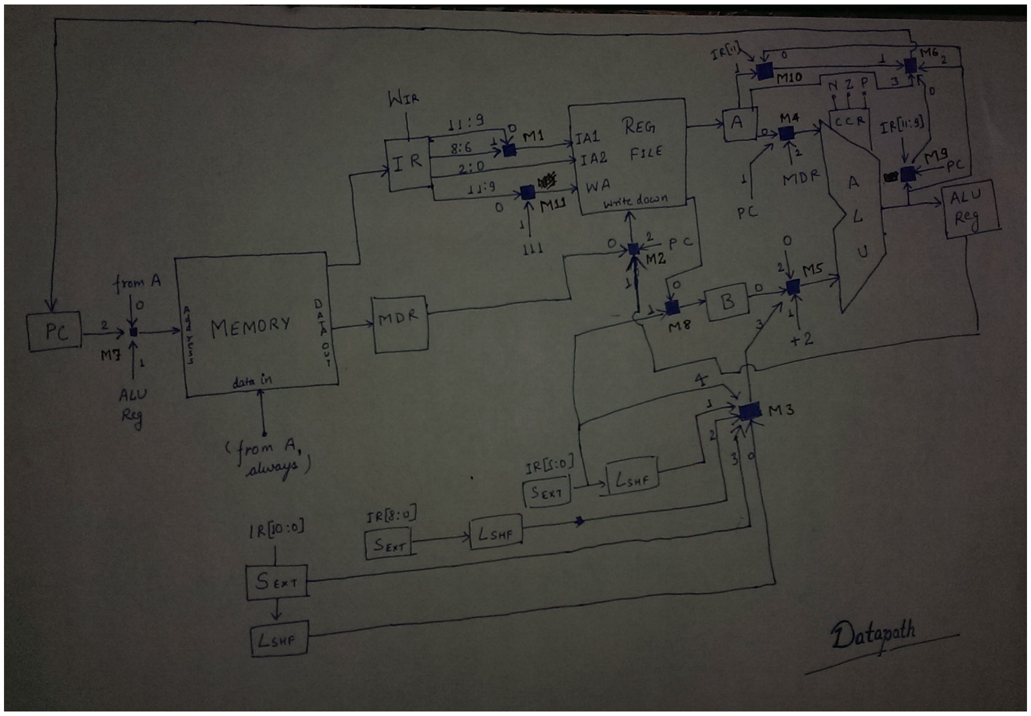
The aim of this exercise was to design and implement an 8-bit RISC processor on an FPGA. The ISA used was the LC-3b instruction set. You may find the instruction set here.

In RISC design, we have to choose between a multi-cycle implementation and a single-cycle implementation. We chose to implement a multi-cycle data-path because all instructions do not take an equal amount of time, and the multi-cycle implementation makes sure that each instruction gets over in the minimum amount of time it requires. Also the single-cycle data-path is more complicated as compared to the multi-cycle data-path, and we save significant resources on the FPGA by doing a multi-cycle implementation.

You may find all the code for our project here

2 Data-path

The data-path we designed is summarised in the diagram below.



3 Controller

The controller implemented has two parts: logic driving current control signals and next state logic. The code for the current control signal values is as follows:

```
always@(*) begin
  case(StateID)
    1: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b111;
      Mux4 = 2'b01;
      Mux5 = 2'b01;
      Mux6 = 2'b10;
      Mux7 = 2'b10;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = ~|IR;
      wir = 1'b0;
      lccr = 1'b1;
      aluop = 2'b00;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    2: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b111;
      Mux4 = 2'b11;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b11;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b0;
      wb = 1'b0;
      lalu = 1'b1;
    end

    3: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b111;
      Mux4 = 2'b00;
      Mux5 = 2'b00;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b0;
      aluop = {IR[15], IR[14]};
      alushop = {IR[5], IR[4]};
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b0;
    end

    4: begin
      Mux1 = 1'b1;
      Mux2 = 2'b01;
      Mux3 = 3'b111;
      Mux4 = 2'b11;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b0;

      wrf = 1'b0;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b11;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    5: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b010;
      Mux4 = 2'b01;
      Mux5 = 2'b11;
      Mux6 = 2'b00;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b0;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b00;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    6: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b111;
      Mux4 = 2'b11;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b0;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b11;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    7: begin
      Mux1 = 1'b1;
      Mux2 = 2'b10;
      Mux3 = 3'b111;
      Mux4 = 2'b11;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b0;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b11;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    8: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b000;
      Mux4 = 2'b01;
      Mux5 = 2'b11;
      Mux6 = 2'b01;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b0;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b00;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    9: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b100;
      Mux4 = 2'b00;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b1;
```

```
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b00;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b0;
    end

    10: begin
      Mux1 = 1'b0;
      Mux2 = 2'b11;
      Mux3 = 3'b111;
      Mux4 = 2'b11;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b11;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b0;
      wb = 1'b1;
      lalu = 1'b1;
    end

    11: begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b111;
      Mux4 = 2'b11;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b01;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b11;
      alushop = 2'b11;
      wmem = 1'b0;
```

```
        wa = 1'b1;
        wb = 1'b1;
        lalu = 1'b1;
      end

      12:begin
        Mux1 = 1'b1;
        Mux2 = 2'b11;
        Mux3 = 3'b001;
        Mux4 = 2'b00;
        Mux5 = 2'b11;
        Mux6 = 2'b11;
        Mux7 = 2'b11;
        Mux11 = 1'b1;

        wrf = 1'b1;
        wpc = 1'b1;
        wir = 1'b1;
        lccr = 1'b1;
        aluop = 2'b00;
        alushop = 2'b11;
        wmem = 1'b1;
        wa = 1'b1;
        wb = 1'b1;
        lalu = 1'b0;
      end

      13:begin
        Mux1 = 1'b1;
        Mux2 = 2'b11;
        Mux3 = 3'b111;
        Mux4 = 2'b11;
        Mux5 = 2'b11;
        Mux6 = 2'b11;
        Mux7 = 2'b01;
        Mux11 = 1'b1;

        wrf = 1'b1;
        wpc = 1'b1;
        wir = 1'b1;
        lccr = 1'b1;
        aluop = 2'b11;
        alushop = 2'b11;
        wmem = 1'b1;
        wa = 1'b1;
        wb = 1'b1;
        lalu = 1'b1;
      end
```

```
    14:begin
      Mux1 = 1'b1;
      Mux2 = 2'b00;
      Mux3 = 3'b111;
      Mux4 = 2'b10;
      Mux5 = 2'b10;
      Mux6 = 2'b11;
      Mux7 = 2'b01;
      Mux11 = 1'b0;

      wrf = 1'b0;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b0;
      aluop = 2'b00;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    15:begin
      Mux1 = 1'b1;
      Mux2 = 2'b11;
      Mux3 = 3'b010;
      Mux4 = 2'b01;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b1;

      wrf = 1'b1;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b0;
      aluop = 2'b00;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b0;
    end

    16:begin
      Mux1 = 1'b1;
      Mux2 = 2'b01;
      Mux3 = 3'b111;
      Mux4 = 2'b11;
      Mux5 = 2'b11;
      Mux6 = 2'b11;
      Mux7 = 2'b11;
      Mux11 = 1'b0;

      wrf = 1'b0;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b11;
      alushop = 2'b11;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end

    default: begin
      Mux1 = 1'b0;
      Mux2 = 2'b00;
      Mux3 = 3'b000;
      Mux4 = 2'b00;
      Mux5 = 2'b00;
      Mux6 = 2'b00;
      Mux7 = 2'b00;
      Mux11 = 1'b0;

      wrf = 1'b1;
      wpc = 1'b1;
      wir = 1'b1;
      lccr = 1'b1;
      aluop = 2'b00;
      alushop = 2'b00;
      wmem = 1'b1;
      wa = 1'b1;
      wb = 1'b1;
      lalu = 1'b1;
    end
  endcase
end
```

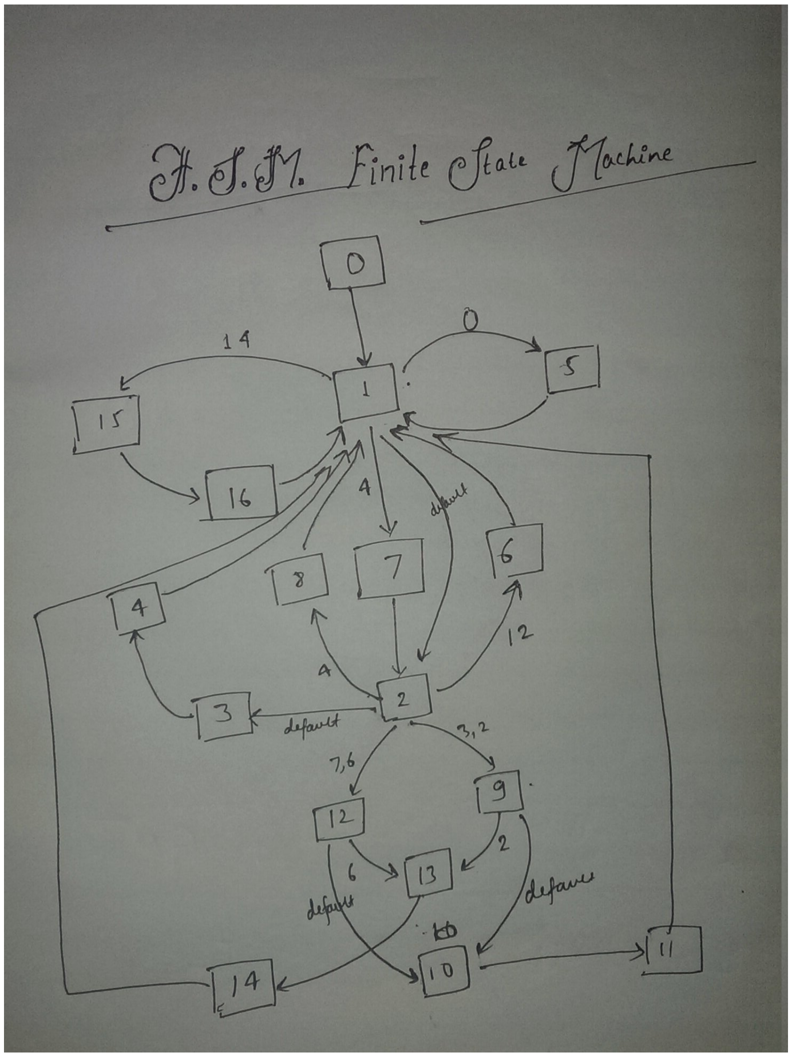
where, the control signals mean the following:

No	Signal Name	Purpose
1	Mux1	Mux1 Control Signal
2	Mux2	Mux2 Control Signal
3	Mux3	Mux3 Control Signal
4	Mux4	Mux4 Control Signal
5	Mux5	Mux5 Control Signal
6	Mux6	Mux6 Control Signal
7	Mux7	Mux7 Control Signal
8	Mux11	Mux11 Control Signal
9	wrf	Write to Register File
10	wpc	Write to Program Counter
11	wir	Write to Instruction Register
12	lccr	Write to Condition Code Register
13	aluop	ALU Operation
14	alushop	Shift Operation
15	wmem	Write to Memory
16	wa	Write to Temporary Register A
17	wb	Write to Temporary Register B
18	lalu	Load ALU Register

Mux8-Mux10 receive their inputs directly via combinational logic from IR. The logic is:

```
Mux8  = IR[5];
Mux9  = ((IR[11]&&N) ||  (IR[10]&&Z) ||(IR[9]&&P));
Mux10 = IR[11];
```

The next state logic is summarised in the following diagram:



For each instruction the state sequence is shown below:

ALU Operation (Add, And, XOR, Not, Shifts): 1 → 2 → 3 → 4

Branch: 1 → 5

Call Subroutine: 1 → 7 → 2 → 8

Load Byte: 1 → 2 → 9 → 13 → 14

Load Word: 1 → 2 → 12 → 13 → 14

Load Immediate Address: 1 → 15 → 16

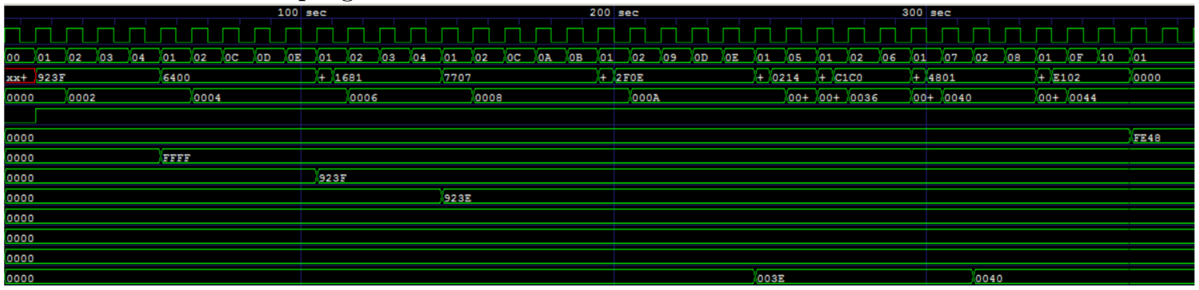
Store Byte: 1 → 2 → 9 → 10 → 11
Store Word: 1 → 2 → 12 → 10 → 11

4 Simulation

We simulated the implementation using a test code that contained all the necessary functionality. In assembly language as per the LC-3b document, the code is as follows:

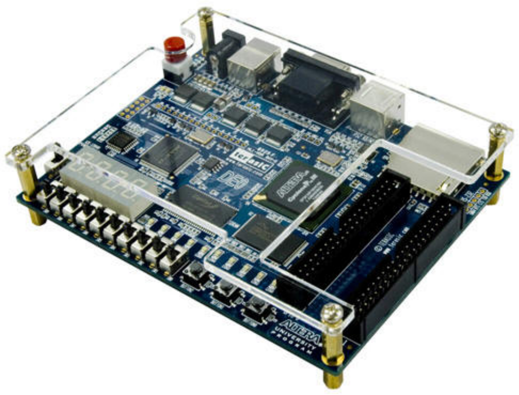
```
NOT R1,R0
LDW R2,R0,#0
ADD R3,R2,R1
STW R3,R4,#7
LDB R7,R4,#14
BRp label

ORG 0x34
label:
JMP R7;
JSR label1

ORG 0x66
label1:
```

The simulation was carried out using GTKWave and iVerilog. The result for the above program were as shown below.



5 Testing

The implementation was tested on an Altera DE0-nano board having a Cyclone-IV FPGA.



The signal-tap tool was used to get states of the elements inside the FPGA at various points in time. For convenience in debugging we gave clock using a push-button on-board. The output was well as expected.