Lab Report No. 05:

μP3 ALU

By:

Jiaqing Wan(16), Jiaxin Zheng(19)

Instructor: Dr. Xi Zhou

Classnumber: ZM1501

NAU ID:

5358836, 5358840

November 5, 2017

I. INTRODUCTION

A. Objectives

At the completion of this lab, we will be able to complete a VHDL ALU design from a functional specification (write VHDL, use ModelSim to simulate functional behavior, and demonstrate functional behavior on the Cyclone V FPGA board).

B. Background knowledge

Arithmetic Logic Unit: A digital electronic circuit that performs arithmetic and bitwise logic operations. It is a fundamental building block of many computing circuits including the CPU.

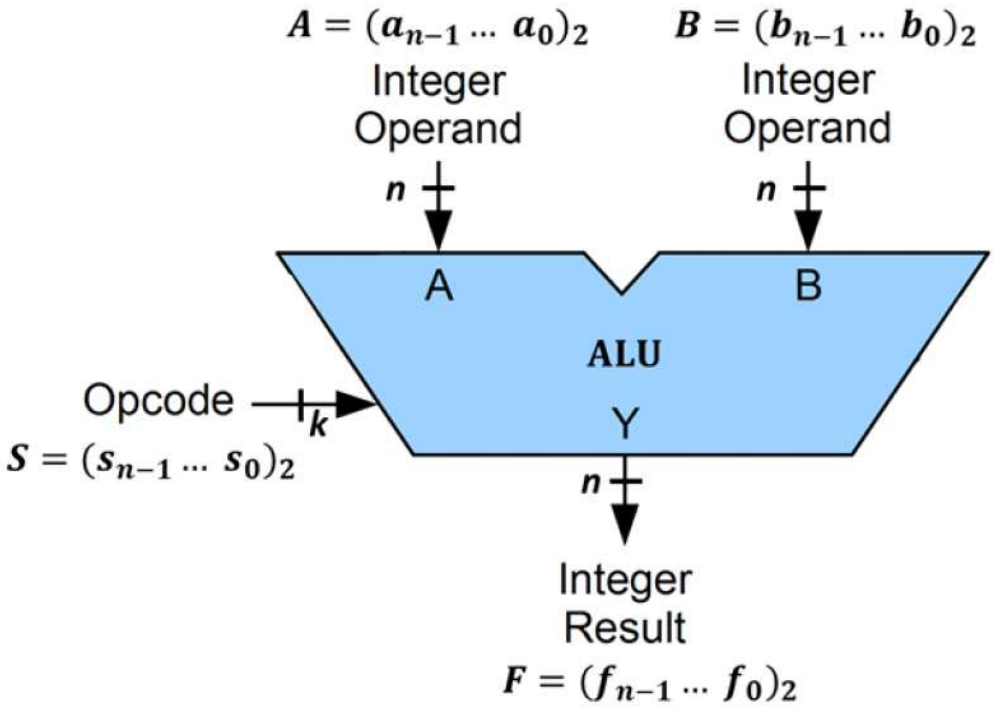

FIG. I.1 ALU

C. Design strategy

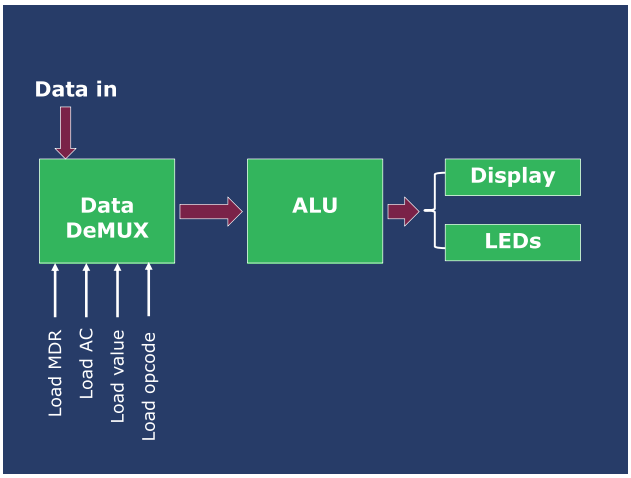

FIG. I.2 The Total Block design strategy

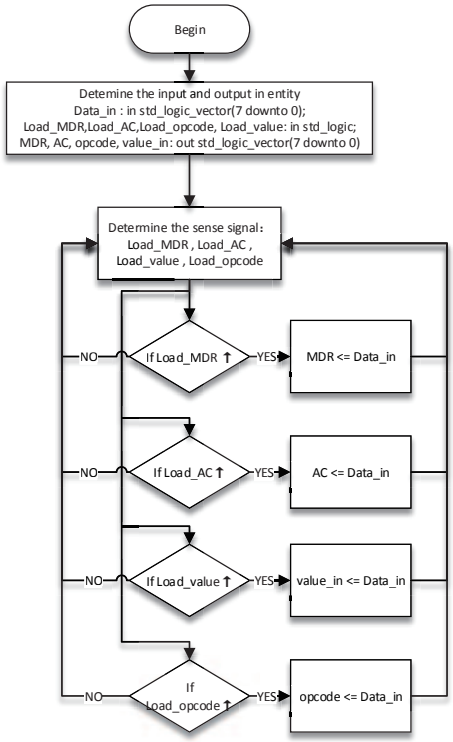

FIG. I.3 The Data DeMUX design strategy

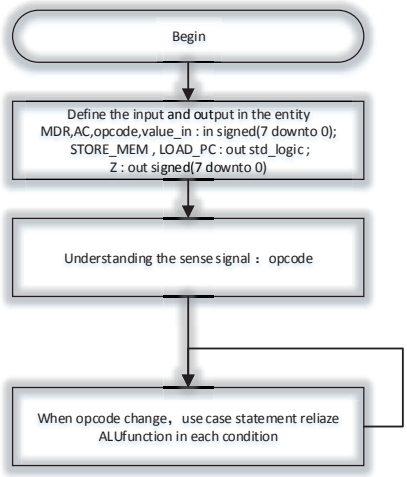

FIG. I.4 The ALU design strategy

D. Simulation strategy

Opcode, Instruction mnemonic	**Test case 1** **MDR: 00** **AC: 55** **address/value: 36**	**Test case 2** **MDR: 55** **AC: AA** **address/value: AB**	**Test case 3** **MDR: AA** **AC: 00** **address/value: 49**	**Test case 4** **MDR: FF** **AC: 9E** **address/value: C2**
00 NOP	××,0,0	××,0,0	××,0,0	××,0,0
01 LOAD	00,0,0	55,0,0	AA,0,0	FF,0,0
02 LOADI	36,0,0	AB,0,0	49,0,0	C2,0,0
03 STORE	××,1,0	××,1,0	××,1,0	××,1,0
04 CLR	00,0,0	00,0,0	00,0,0	00,0,0
05 ADD	55,0,0	FF,0,0	AA,0,0	9D,0,0
06 ADDI	8B,0,0,	55,0,0	49,0,0	60,0,0
07 SUBT	55,0,0	55,0,0	56,0,0	9F,0,0
08 SUBTI	1F,0,0	FF,0,0	87,0,0	DC,0,0
09 NEG	00,0,0	AB,0,0	56,0,0	01,0,0
0A NOT	FF,0,0	AA,0,0	55,0,0	00,0,0
0B AND	00,0,0	00,0,0	00,0,0	9E,0,0
0C OR	55,0,0	FF,0,0	AA,0,0	FF,0,0
0D XOR	55,0,0	FF,0,0	AA,0,0	61,0,0
0E SHL	40,0,0	50,0,0	00,0,0	78,0,0
0F SHR	01,0,0	15,0,0	00.0.0	27,0,0
10 JUMP	××,0,1	××,0,1	××,0,1	××,0,1
11 JNEG	××,0,0	××,0,1	××,0,0	××,0,1
12 JPOSZ	××,0,1	××,0,0	××,0,1	××,0,0
13 JZERO	××,0,0	××,0,0	××,0,1	××,0,0
14 JNZER	××,0,1	××,0,1	××,0,0	××,0,1

E. Demonstration strategy

TABLE I The ALU Set

Input List	Hardware	Output List	Hardware
Load_opcode	KEY0	Z[7..0]	HEX[1..0]
Load_value	KEY1	AC[7..0]	HEX[3..2]
Load_AC	KEY2	MDR[7..0]	HEX[5..4]
Load_MDR	KEY3	value[7..0]	LEDR[7..0]
Data_in[7..0]	SW[7..0]	LOAD_PC	LEDR[8]
test	SW[8]	STORE_MEM	LEDR[9]
blank	SW[9]		

F. Contribution of each team member

Jiaxin Zheng: Write code.

Jiaqing Wan: Do simulation and write report.

II. LAB DETAILS

A. Instructor questions in the lab instruction

1. In the left box below, list the instructions that produce a data result on the Z bus of the ALU. In the right box, list all of the remaining instructions.

Instructions producing results on Z bus:	Instructions where Z bus does not matter:
01,02,04,05,06,07, 08 09,0A,0B,0C,0D,0E,0F	03,10,11,12,13,14

2.There are some instructions that do not require a specific output from the ALU.You are free to choose the value of the ALU output (the Z bus) in these cases. What is it and why did you select it?

A: opcode = 01;(LOAD address);

opcode = 02; (lLOADI value);

because we can determine it use input, it only load function .

3. What are the input and output signals of the ALU entity and how many bits are they?

Input	Output
MDR 8bit	STORE_MEN 1bit
AC 8bit	LOAD_PC 1bit
opcode 8bit	Z 8bit
value_in 8bit	

4.Please see section "Simulation strategy".

B. VHDL code

Data_DeMUX.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity Data_DeMUX is
   port(
   Data_in : in std_logic_vector(7 downto 0);
   Load_MDR,Load_AC,Load_opcode, Load_value: in std_logic;
   MDR, AC, opcode, value_in: out std_logic_vector(7 downto 0)
   );
end entity Data_DeMUX;

   architecture Data_DeMUXbhv of Data_DeMUX is
   begin
      ------------------------------------------------------
      ------------------------------------------------------
      process(Load_MDR , Load_AC , Load_value , Load_opcode)
      begin
         ---------------------------------------------
         if Load_MDR'event AND Load_MDR = '1'          then
            MDR <= Data_in;                    --Load MDR
         end if;
         ---------------------------------------------
         if Load_AC'event AND Load_AC = '1'            then
            AC <= Data_in;                     --Load AC
         end if;
         ---------------------------------------------
         if Load_value'event AND Load_value = '1'      then
            value_in <= Data_in;               --Load value
         end if;
         ---------------------------------------------
         if Load_opcode'event AND Load_opcode = '1'    then
```

```
            opcode <= Data_in;                    --Load opcode
        end if;
        ------------------------------------------
    end process;
    ---------------------------------------------------
    ---------------------------------------------------
end architecture Data_DeMUXbhv;
```

ALU.vhd

```
-- alu.vhd
-- Implements the instruction set
-- for the uP3 microprocessor in Lab 4
--
-- Author: Jiaxin Zheng,Jiaqing Wan, NAU/CUPT EE
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library altera_mf;
use altera_mf.altera_mf_components.all;

entity alu is
  port(
     MDR,AC,opcode,value_in : in signed(7 downto 0);
     STORE_MEM , LOAD_PC : out std_logic ;
     Z : out signed(7 downto 0)
  );
end entity alu;

  architecture alubhv of alu is
  begin

     process(opcode)
     begin
      STORE_MEM <= '0';
      LOAD_PC <= '0';
         case opcode is
            ------------------------------------------
            --Instruction Groups       --opcode 00;
            --No operation
            ------------------------------------------
               when x"00" =>
                  null;                           --Nop
            ------------------------------------------
            --Instruction Groups       --opcode 01~04;
            --Load and Store Instruction
            ------------------------------------------
               when X"01" =>
                  Z <=MDR;                        --LOAD address
            ------------------------------------------
               when x"02" =>
                  Z <= value_in;                  --LOADI value
            ------------------------------------------
               when x"03" =>
                  STORE_MEM <= '1';               --STORE address
            ------------------------------------------
               when x"04" =>
                  Z <= "00000000";                --CLEAR
```

```
-------------------------------------------
--Instruction Groups       --opcode 05~09;
--Arithmetic Instruction
-------------------------------------------
    when x"05" =>
       Z <= (AC + MDR);                --ADD address
-------------------------------------------
    when x"06" =>
       Z <= AC+value_in;               --ADDI value
-------------------------------------------
    when x"07" =>
       Z <= AC-MDR;                    --SUBT address
-------------------------------------------
    when x"08" =>
       Z <= AC-value_in;               --SUBTI value
-------------------------------------------
    when x"09" =>
       Z <= 0-MDR;                     --NEG address
-------------------------------------------
--Instruction Groups       --opcode 0A~0F;
--Logic Instruction
-------------------------------------------
    when x"0A" =>
       Z <= NOT MDR;                   --NOT address
-------------------------------------------
    when x"0B" =>
       Z <= AC AND MDR;                --AND address
-------------------------------------------
    when x"0C" =>
       Z <= AC OR MDR;                 --OR address
-------------------------------------------
    when x"0D" =>
       Z <= AC XOR MDR;                --XOR address
-------------------------------------------
    when x"0E" =>
       Z <= AC sll to_integer(unsigned(value_in(2 downto 0)));
                                       --SHL value
-------------------------------------------
    when x"0F" =>
       Z <= AC  srl to_integer(unsigned(value_in(2 downto 0)));
                                       --SHR value
-------------------------------------------
--Instruction Groups       --opcode 10~14;
--Control Flow (Jump) Instruction
-------------------------------------------
    when x"10" =>
       LOAD_PC <= '1';                 --JUMP address
-------------------------------------------
    when x"11" =>
       if AC < 0 then
       LOAD_PC<='1';
          else
             LOAD_PC<='0';
       end if;                         --JNEG address
-------------------------------------------
    when x"12" =>
       if  signed(AC) >= 0 then
          LOAD_PC <= '1';
          else
```

```
                    LOAD_PC <= '0';
                end if;                                --JPOSZ address
            ------------------------------------------------
                when x"13" =>
                    IF AC = 0 then
                        LOAD_PC <= '1';
                        else
                            LOAD_PC <= '0';
                    end IF;                            --JZERO address
            ------------------------------------------------
                when x"14" =>
                    IF AC  /= 0 then
                        LOAD_PC <= '1';
                        else
                            LOAD_PC <= '0';
                    end IF;                            --JNZER address
            ------------------------------------------------
            --Others condition
            ------------------------------------------------
                when others => null;
            ------------------------------------------------
        end case;
    end process;
end alubhv;
```

C. Block diagram

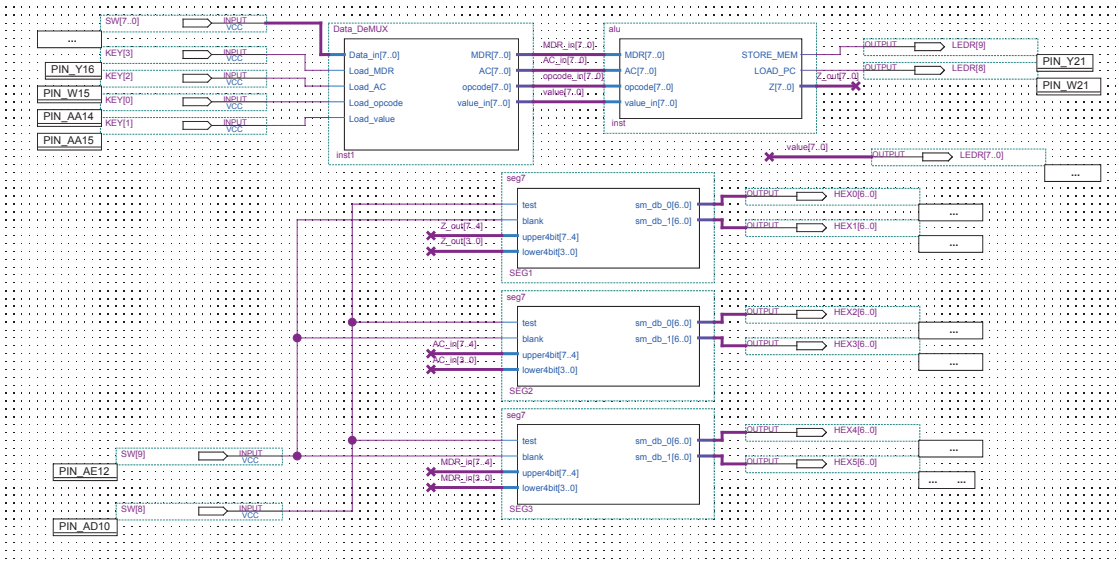

FIG. II.1 alu_tb Block diagram

D. ModelSim force file

Data_DeMUX.txt:

```
#Load Data to MDR
force Load_MDR      1
force Load_AC       0
force Load_value    0
force Load_opcode   0
force Data_in       10000000
run    100ns

#Load Data to AC
force Load_MDR      0
force Load_AC       1
force Load_value    0
force Load_opcode   0
force Data_in       11000000
run    100ns

#Load Data to value
force Load_MDR      0
force Load_AC       0
force Load_value    1
force Load_opcode   0
force Data_in       11100000
run    100ns

#Load Data to opcode
force Load_MDR      0
force Load_AC       0
force Load_value    0
force Load_opcode   1
force Data_in       11110000
run    100ns
```

alu_tb.txt:

```
#initial all input
force MDR    00000000
force AC     00000000
force opcode    00000000
force value_in     00000000
run 100ns

#test opcode X"01" --load address
force MDR    00001111
force opcode    00000001
run 100ns

#test opcode X"05" --ADD address
force AC     00001111
force MDR    11110000
force opcode    00000101
run 100ns

#test opcode X"0A" --NOT MDR
force MDR    00000000
```

```
force opcode   00001010
run 100ns

#text opcode X"10" --assert Load_PC
force opcode   00010000
run 100ns

#test opcode X"04" --CLR
force opcode   00000100
run 100ns
```

E. Simulation results and detailed explanation

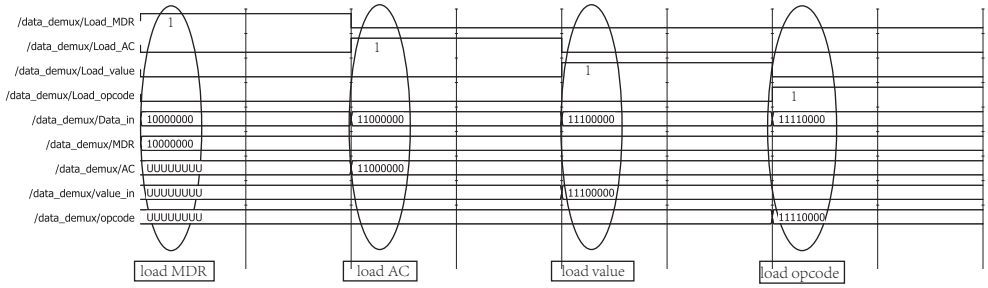

FIG. II.2 DeMUX Wave

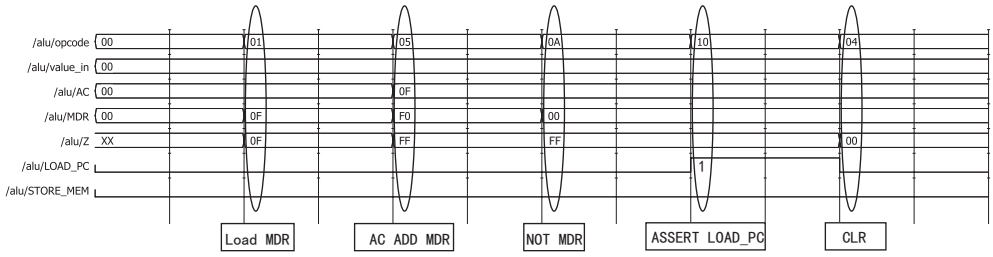

FIG. II.3 alu Wave

F. Board demonstration result

The board demonstration result is following as the table at section "Simulation strategy". In others case the result is the same logic we need.

III. CONCLUSION

The lab is successful.

I realize all function we should achieve. The most difficulty things is the Instructor questions.

I refer to manual and ask classmates that I solved it.