EXPERIMENT NO. 3

ARITHMETIC LOGIC UNIT

AIM: To write VHDL code, simulate with test bench, synthesis, implement on PLD for Arithmetic Logic Unit.

SOFTWARE TOOL: Xilinx, FPGA Kit.

THEORY:

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU).

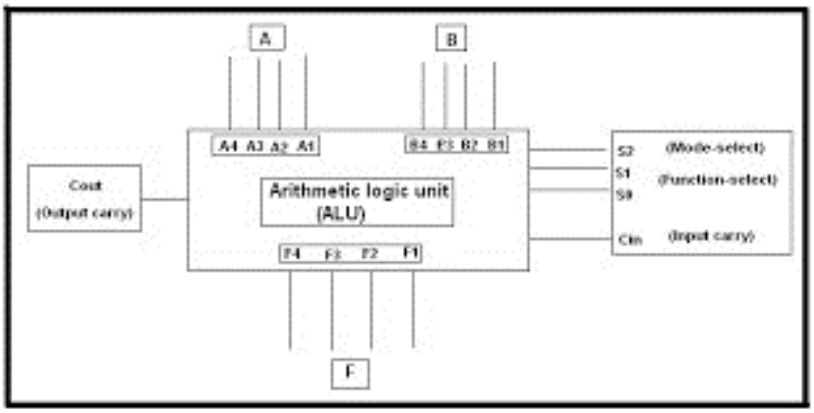
Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU. The control unit tells the ALU what operation to perform on that data and the ALU stores the result in an output register. TAn ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

All information in a computer is stored and manipulated in the form of binary numbers, i.e. 0 and 1. Transistor switches are used to manipulate binary numbers since there are only two possible states of a switch: open or closed. An open transistor, through which there is no current, represents a 0. A closed transistor, through which there is a current, represents a 1.

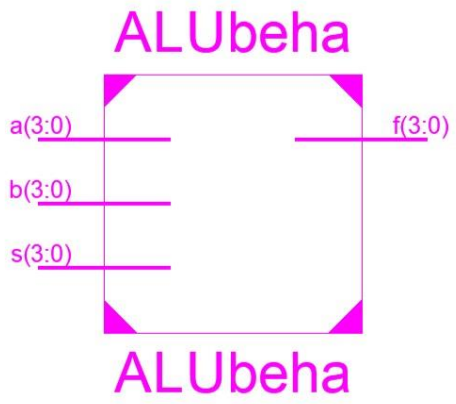
Operations can be accomplished by connecting multiple transistors. One transistor can be used to control a second one - in effect, turning the transistor switch on or off depending on the state of the second transistor. This is referred to as a gate because the arrangement can be used to allow or stop a current.

The control unit moves the data between these registers, the ALU, and memory.

DIAGRAM OF ALU:



RTL VIEW:



VHDL CODE:

```
 library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_unsigned.all;


entity alupractical3 is
    Port ( A,B : in  STD_LOGIC_VECTOR(3 DOWNTO 0);
           S : in  STD_LOGIC_VECTOR(3 DOWNTO 0);
                    M : IN STD_LOGIC;
           Y : out  STD_LOGIC_VECTOR(3 DOWNTO 0));
end alupractical3;

architecture Behavioral of alupractical3 is
```

```
begin
process(A,B,M,S)
begin
if M='0' then
case S is
when "0000"=>Y<=A+B;
when "0011"=>Y<=A-B;
when "0100"=>Y<=A+1;
when "1000"=>Y<=B+1;
when others=>null;
end case;
else
case S is
when "0000"=>Y<=A and B;
when "0011"=>Y<=A or B;
when "0100"=>Y<=A nand B;
when "1000"=>Y<= A xor A;
when others=>null;
end case;
end if;
end process;
end Behavioral;
```

Data Flow :

```
Library IEEE;

Use IEEE.STD_LOGIC_1164.ALL;

Use IEEE.STD_LOGIC_ARITH.ALL;

Use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity alu is

Port (A: in std_logic_vector (3 downto 0);

B: in std_logic_vector (3 downto 0);

sel : in std_logic_vector (2 downto 0);

out_alu : out std_logic_vector (3 downto 0));

end alu;

architecture dataflow of alu is
```

```
Begin
With sel select
Out alu <= A+B when "000",
A-B when "001",
A+1 when "010",
A-1 when "011",
A and B when "100",
A or B when "101",
A xor B when "110","
Not A when "111", I
Null when others;
End dataflow;
```

CONCLUSION: