Experiment 6: Arithmetic Logic Unit

Debasish Panda 21D070021

September 25, 2022

1 Overview of the experiment

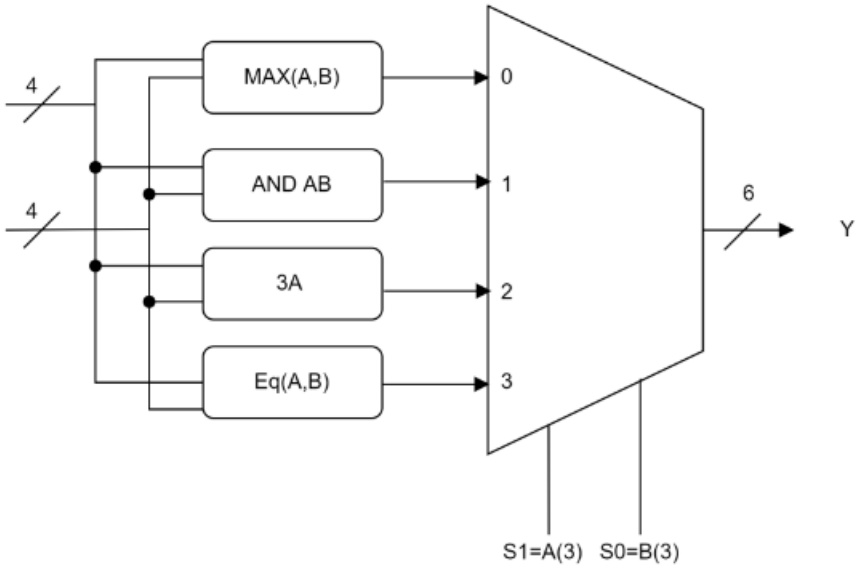
The purpose of the experiment is to describe an ALU(Arithmetic Logic Unit) using fundamental logic operations and 2-to-1 MUXes in VHDL using Behavioral-Dataflow modelling. Following a successful RTL simulation, we have to demonstrate the same using Scanchain on Xenon board.
In order to perform the experiment, we need to design four separate digital logic based blocks, whose output will be governed by two select inputs as follows:

S1	S0	ALU Output
0	0	MAX(A,B): This block outputs larger number between A and B else outputs 0000.
0	1	AND A B: This block performs bitwise AND operation between A , B.
1	0	3*A: This block Produces output as 3*A
1	1	Eq(A,B): This block outputs the number whenever A=B else it should output 0000.

The appropriate code for Behavioral-Dataflow modelling of these components is written using VHDL on Quartus Prime software. Next, we have to demonstrate the ALU practically by using the Xen10 FPGA board and Scanchain. In order to run the VHDL code on the board, we also need to determine the input and output pins correctly as per the specifications of the circuit board.
In this report first we discuss the setup required for the experiment and the approach used by me to solve the assignment. Attached alongwith are the pen-and-paper diagram and picture showing the simulation output of the design.

2 Experimental setup

First we need to describe the ALU using logic operations and MUXes. Note that we will need to use behavioral modelling in order to execute the required operations on the input, so we do not need to design any hardware design as such.



Structure of ALU

Now that we have the required circuit, we need to assign various component instances, architecture and signals as per VHDL syntax. The required code for the same is discussed in the next section.

2.1 Design Documentation and VHDL Code

The VHDL code for designing the entity-architecture pairs associated with the digital circuit is as follows:

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

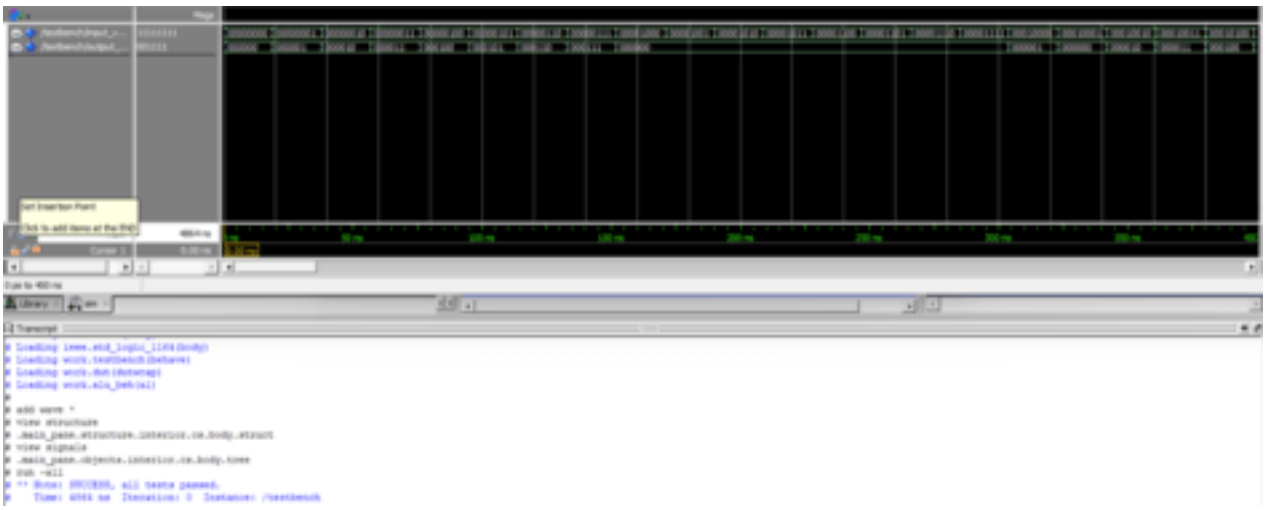
```
entity alu_beh is
 generic(
   operand_width : integer:=4);
 port (
   A: in std_logic_vector(operand_width-1 downto 0);
   B: in std_logic_vector(operand_width-1 downto 0);
   op: out std_logic_vector(5 downto 0)) ;
end alu_beh;

architecture a1 of alu_beh is

 function add(A: in std_logic_vector(operand_width-1 downto 0);
  B: in std_logic_vector(operand_width-1 downto 0))
    return std_logic_vector is

    variable sum : std_logic_vector(operand_width-1 downto 0) := (others => '0');
 variable carry : std_logic_vector(operand_width-1 downto 0) := (others => '0');

   begin
      for i in 0 to ((operand_width)-1) loop
if i=0 then
sum(i) :=  A(i) xor B(i);
carry(i) := A(i) and B(i) ;
else
   sum(i) :=  A(i) xor B(i) xor carry(i-1);
carry(i) := (A(i) and B(i)) or (carry(i-1) and (A(i) xor B(i))) ;
end if;
end loop;

    return carry(3) & sum;
 end add;

begin
  alu : process(A, B)
  variable dummy : std_logic_vector(operand_width downto 0);
  variable sum_1 : std_logic_vector(operand_width downto 0);
  variable carry_1: std_logic;
```

```
  variable carry_2: std_logic;
  variable temp: std_logic_vector(operand_width-1 downto 0);
  begin


  if A(3)='1' and B(3)='1' then
   temp:=A xor B;
  if temp="0000" then
op<= "00"&A;
  else
op<="000000";
  end if;

     elsif A(3)='0' and B(3)='0' then
     if A > B then
    op<= "00"&A;
     elsif B > A then
    op<= "00"&B;
  else
    op<= "000000";
     end if;

  elsif A(3)='0' and B(3)='1' then
op<= "00"&(A and B);


  else
    dummy := add(A, A);
 sum_1 := add(dummy(3 downto 0),A);
 carry_1 := dummy(4) xor sum_1(4);
 carry_2 := dummy(4) and sum_1(4);
       op <= carry_2 & carry_1 & sum_1(3 downto 0);
  end if;


end process ;
end a1 ;
```

A new addition to this lab's VHDL code is the *std_logic_vector* class, which is used to easily define a large number of similar circuit elements by representing them in the form of an array.

3 Observations

The experiment was conducted successfully, and the expected output was obtained through the RTL Simulation. The Scanchain also gave the desired result, with all test cases passing successfully. The expected output of the Universal Shifter is shown in the form of the final RTL Simulation result of the design.



RTL Simulation of ALU

Following this, we have to do the pin planning correctly and convert our VHDL file to a .svf file, which is executable by the Xen10 board and is also needed for running the Scanchain program.

4 References

[1] EE214 Github page

[2] Overleaf LaTeX tutorial for beginners

[3] Introduction to Xen10 Board