EXPERIMENT6 - Arithmetic Logic Unit(ALU)

DEVESH SONI
Roll Number: 21D070025

September 9, 2022

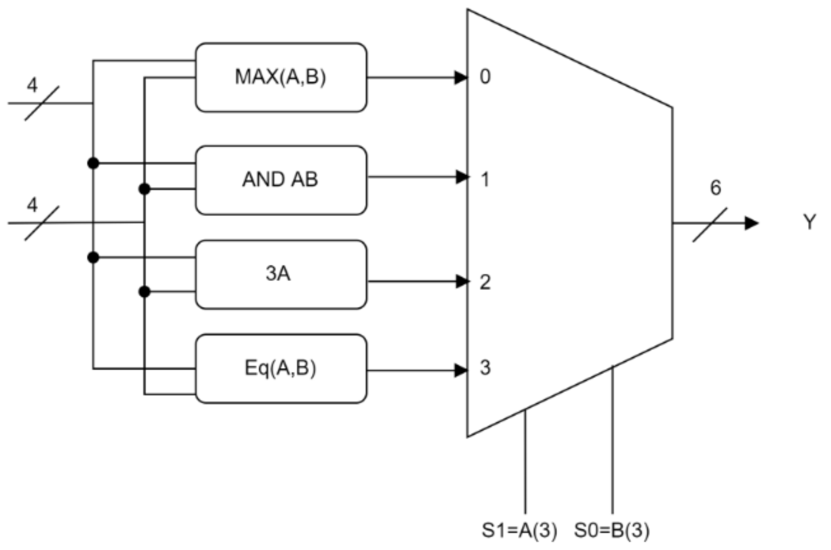
1 Overview of the Experiment

- In this experiment designed an ALU, which can perform and select 4 functions as per the inputs.Moreover the main aim of this experiment is to get familiarized with **Behavioral-Dataflow modelling** for writing VHDL description
- We also tested the correctness of our ALU design using scanchain.
- In this report I've provided the overview, approach and observations of the experiment. To summarize -
  - VHDL code for the ALU (and the changes made in gates.vhdl, testbench.vhdl and DUT.vhdl files).
  - RTL Viewer and RTL Stimulation
  - Creating SVF file, and dumping this SVF file onto the board and testing using scanchain.

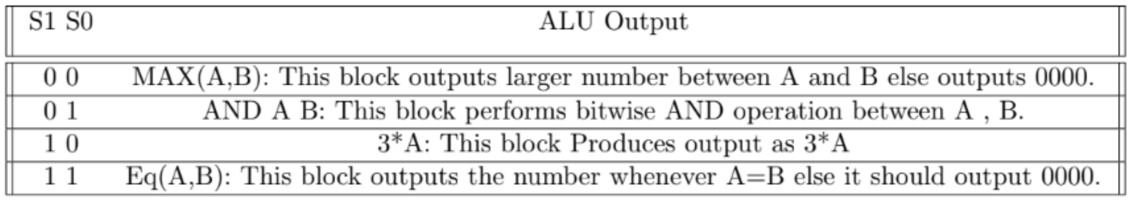
2 Approach to the Assignment

2.1 ALU

- We made the ALU by the following method :-
  Block diagram :-



A and B are 4 bit inputs , and MSB of bth the inputs also acts a selection inputs for the MUX.
given below is the table for various selection inputs and outputs:



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.

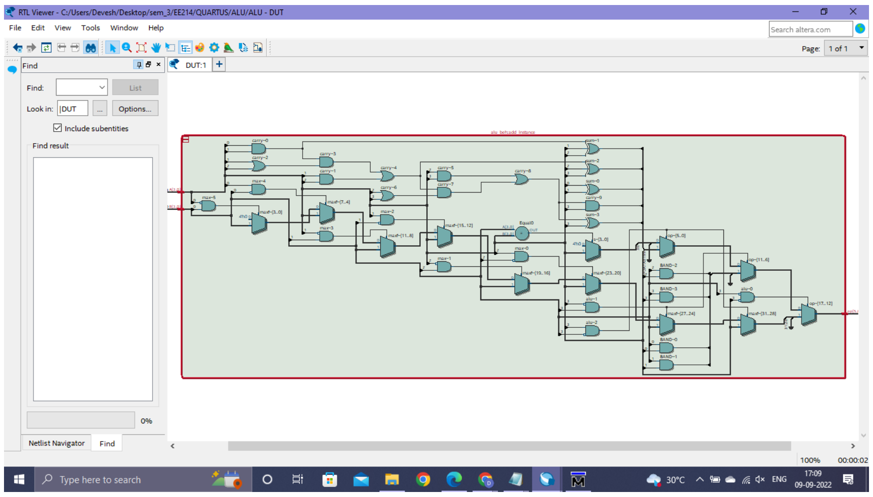
- ALU.vhdl
  Imported the required libraries and wrote the following code:-

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.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 (3 downto 0);
B: in std_logic_vector (3 downto 0))
return std_logic_vector is
variable sum: std_logic_vector (3 downto 0);
variable carry: std_logic_vector (3 downto 0);
begin
yay: for i in 0 to (operand_width-1) loop
if i = 0 then
sum (i) := A(i) xor B (i) xor ’0’;
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) or B(i)));
end if;
end loop yay;
return carry(3) & sum;
end add;




function BITWISEAND(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 BAND : std_logic_vector(5 downto 0):= (others=>’0’);

begin
wine: for i in 0 to (operand_width - 1) loop

BAND(i) := A(i) and B(i);



end loop wine;
```

```
BAND(4):= '0';
BAND(5):= '0';
return BAND;
end BITWISEAND;



function max(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 maxf : std_logic_vector(operand_width*2-3 downto 0) ;
begin
if A(3) > B(3) then
maxf := "00" & A;
elsif A(3) < B(3) then
maxf := "00" & B;
elsif A(2) > B(2) then
maxf := "00" & A;
elsif A(2) < B(2) then
maxf := "00" & B;
elsif A(1) > B(1) then
maxf := "00" & A;
elsif A(1) < B(1) then
maxf := "00" & B;
elsif A(0) > B(0) then
maxf := "00" & A;
elsif A(0) < B(0) then
maxf := "00" & B;
else
maxf := "000000";
end if;
return maxf;
end max;

function mul3(A: in std_logic_vector(3 downto 0))
return std_logic_vector is
variable s: std_logic_vector (5 downto 0):= (others=>'0');
variable y: std_logic_vector (3 downto 0);
variable z: std_logic_vector (4 downto 0);
begin
y:= ('0',A(3),A(2),A(1));
z:= add(A,y);
s:=  z & A(0);
return s;


return s;
end mul3;



function eq(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 s: std_logic_vector (5 downto 0):= (others=>'0');
begin
if A = B then
s := "00"&A;
else
s := "000000";
```

```
end if;
return s;
end eq;

begin


alu : process( A, B)

begin
-- complete VHDL code for various outputs of ALU based on select lines
-- Hint: use if/else statement
--
-- sub function usage :
-- signal_name <= sub(A,B)
-- variable_name := sub(A,B)
--
-- concatenate operator usage:
-- "0000"&A

if (A(3) ='0' and B(3) ='0') then
op <= max(A,B);
elsif (A(3) ='0' and B(3) ='1') then
op <= BITWISEAND(A,B);
elsif (A(3) ='1' and B(3) ='0') then
op <= mul3(A);
else
op <= eq(A,B);
end if;
end process ;
end a1 ;
```

- No other changes were required in the gates.vhdl. Modified number of inputs and outputs to 8 and 6 respectively in testbench.vhdl. Changed component name and instance name in DUT file and changed the number of input and output to 7 down to 0 and 5 down to 0 respectively.
  The final output is 6 bit output and all the functions have 4 bit inputs thus to make ouput 6 bits we have to concatenate 2 extra zeroes.
  Uplaoded the already provided scanchain files and set the top level entity.
- Verified the correctness the code from the RTL viewer design :-



netlist Viewer with scanchain

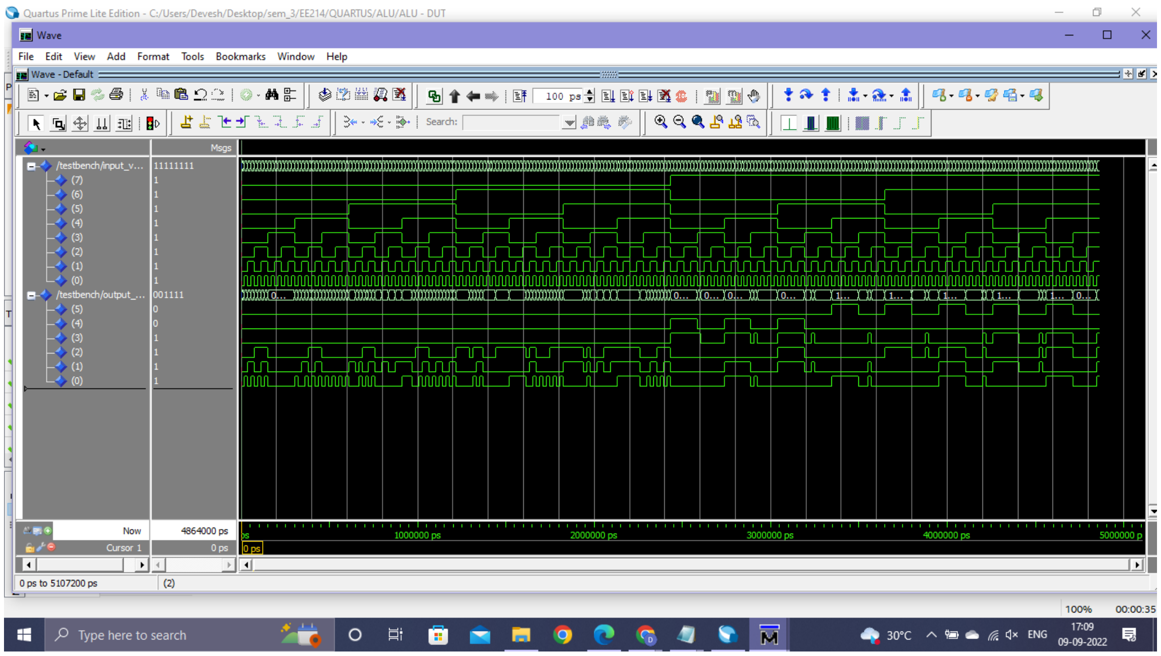
2.2 Testing using Scanchain

After compilation is complete create a .svf file in tools in programmer then new file of format SVF. A new .svf file will be created. After that connect the board to the laptop through USB cabel provided, open the jtag.exe application write the required code and then verify the working of ALU through scanchain. To do so :-

1. Install Python, PIP, Python Packages
2. Files Required for Scan Chain : DUT wrapped design file and other dependent design files with working RTL simulation, Tracefile,TopLevel.vhd, Folder vjatg, scanvjtag.exe file, scanvjtag.py, vjatg.vhd, vjtag.qip
3. Set the TopLevel.vhd as the top priority.
4. After this do a full compilation of the design, generate SVF file again, dump the SVF file on the Xenon Board, Close the Urjtag Terminal
5. Run the following command- scanvjtag.exe TRACEFILE.txt out.txt. out.txt is a file that is generated by the code check out.txt to find whether your design has successfully passed all test cases or not.

3 Observations

3.1 RTL Stimulation of ALU



RTL Stimulation