Experiment 1: ALU
A submission Report

Hardik Panchal
Roll Number 200070054
EE-214, WEL, IIT Bombay
September 27, 2021

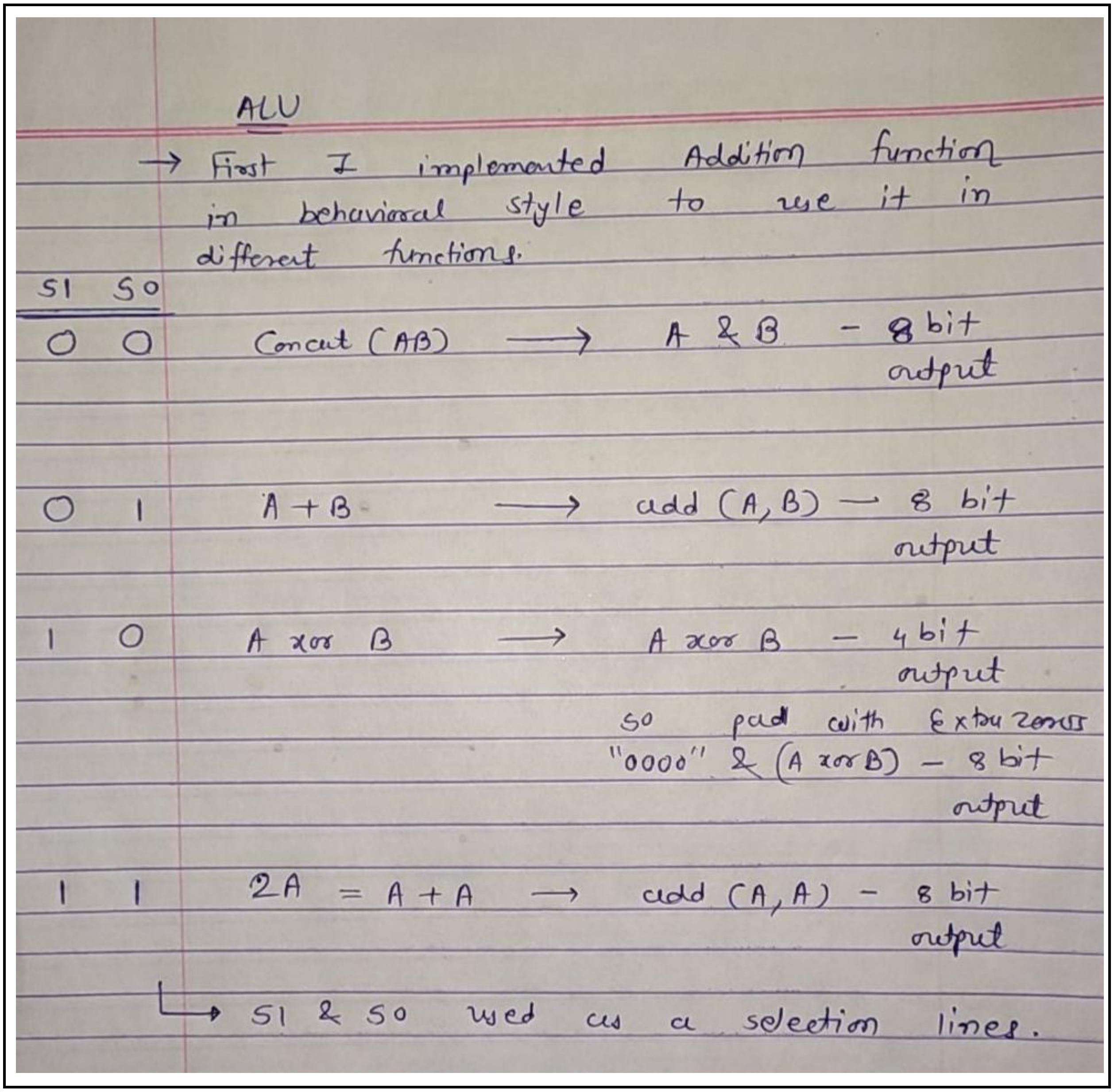
Overview of the experiment:

In the experiment we have to describe an ALU using behavioral modelling. The ALU has to perform 4 different operations based on the input S1 and S0.
First function was to concatenate two 4-bit inputs and make one 8-bit output.
Second function was to add two 4-bit inputs and make one 8-bit output.
Third function was to do a bitwise XOR operation of two 4-bit inputs and report 8-bit output by concatenating extra zeros in front of output.
Last function was to output the double of 4-bit input as 8-bit output.

I will be presenting here the design and code for this and also screenshots of successful RTL and Gate Level simulations.

Approach to the experiment:

I firstly described the addition function in behavioral style and used that for accomplishing first and last function.
I used the in built xor operator to do bitwise xor of two inputs to implement third function.
I used the in built & operator to concatenate two inputs to implement fourth function and also, I used this operator to make my other 4-bit output into 8-bit output via concatenating extra zeros in front of them.



ALU

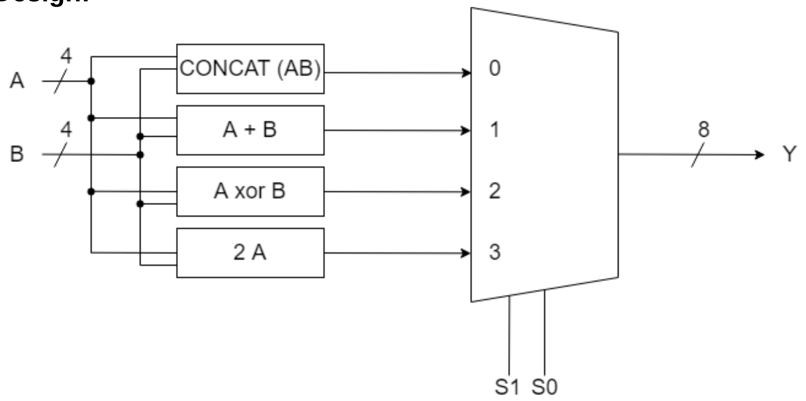
→ First I implemented Addition function in behavioral style to use it in different functions.

S1	S0			
0	0	Concat (AB)	→	A & B - 8 bit output
0	1	A + B	→	add (A, B) — 8 bit output
1	0	A xor B	→	A xor B — 4 bit output so pad with Extra zeros "0000" & (A xor B) — 8 bit output
1	1	2A = A + A	→	add (A, A) - 8 bit output

↳ S1 & S0 used as a selection lines.

Design document and VHDL code if relevant:

Design:

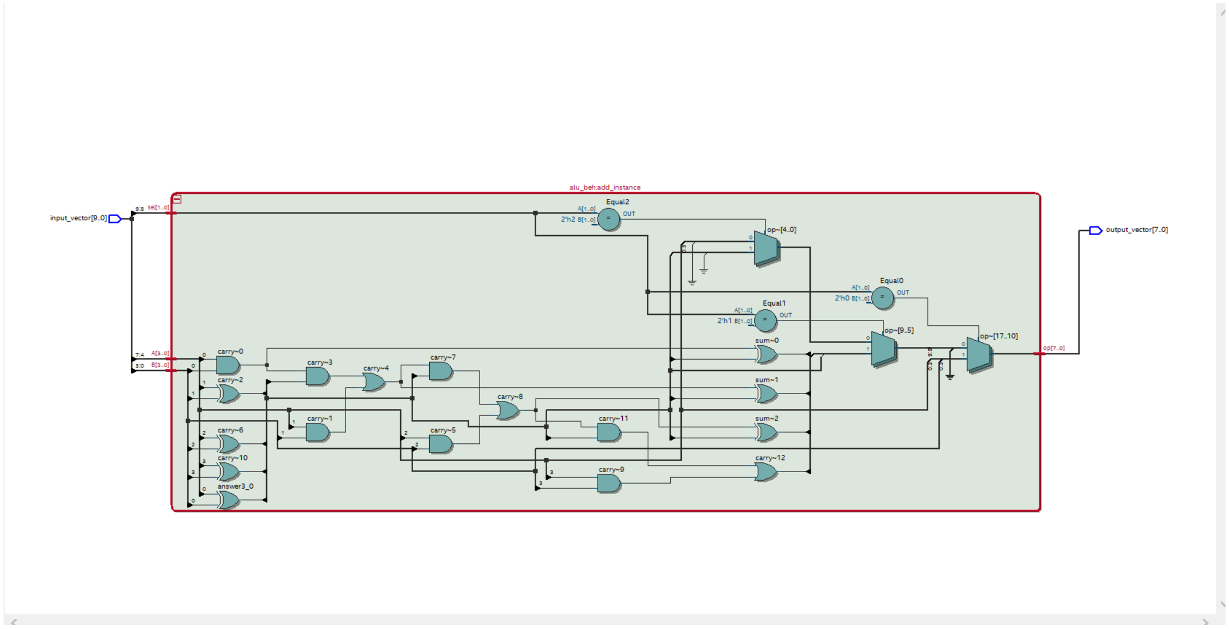


Code of an architecture:

```
architecture a1 of alu_beh is

   function add(A: in std_logic_vector((operand_width*2)-1 downto 0); B: in
std_logic_vector((operand_width*2)-1 downto 0))
      return std_logic_vector is

                                       variable sum : std_logic_vector(7 downto 0) := (others => '0');
                                       variable carry : std_logic_vector(8 downto 0) := (others => '0');


      begin

                             carry(0) := '0';
                                       for i in 0 to 7 loop
                                                       sum(i) := A(i) xor B(i) xor carry(i);
                                                       carry(i+1) := (A(i) and B(i)) or (carry(i) and (A(i) xor
B(i)));
                                       end loop;
         return sum;
   end add;


begin
alu : process( A, B, sel )

          variable answer1 : std_logic_vector(7 downto 0);

          variable answer2 : std_logic_vector(7 downto 0);
          variable Anew : std_logic_vector(7 downto 0);
          variable Bnew : std_logic_vector(7 downto 0);

          variable answer3 : std_logic_vector(7 downto 0);
          variable answer3_0 : std_logic_vector(3 downto 0);

          variable answer4 : std_logic_vector(7 downto 0);
          variable answer4_0 : std_logic_vector(7 downto 0);
```

```
begin


        if (sel = "00") then

                        answer1 := A&B;
                        op <= answer1;

        elsif (sel = "01") then

                        Anew := "0000" & A;
                        Bnew := "0000" & B;

                        answer2 := add(Anew,Bnew);
                        op <= answer2;

        elsif (sel = "10") then

                        answer3_0 := A xor B;
                        answer3 := "0000" & answer3_0;
                        op <= answer3;

        else

                        answer4_0 := "0000" & A;
                        answer4 := add(answer4_0, answer4_0);

                        op <= answer4;


        end if;

end process ; -- alu
end a1 ; -- a1
```

RTL View:



DUT Input/Output Format:

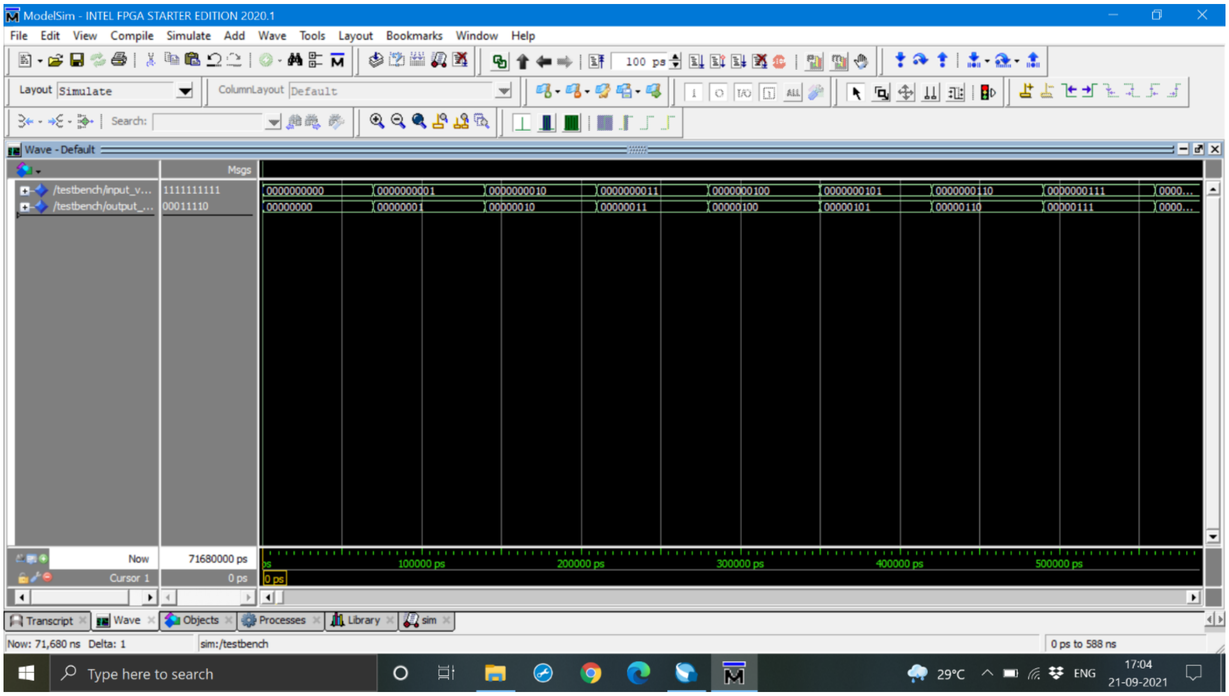
Input: S1 S0 A3 A2 A1 A0 B3 B2 B1 B0 **LSB** = B0, **MSB** = S1
Output: Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0 **LSB** = Y0, **MSB** = Y7

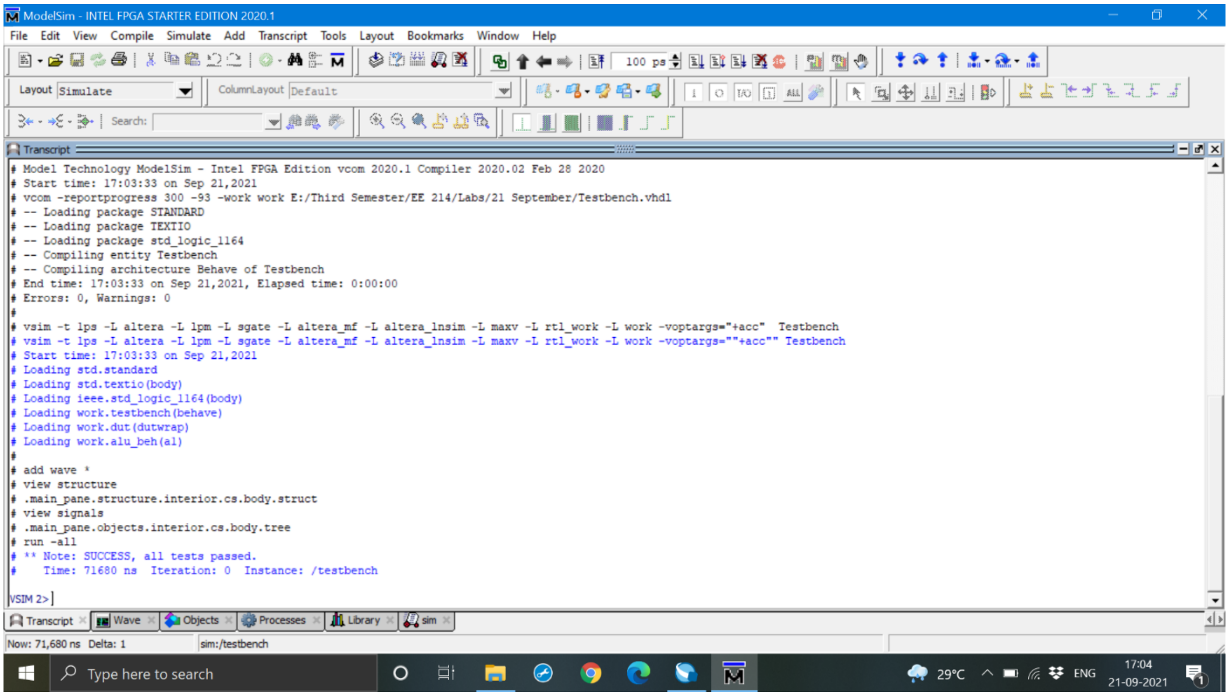
Some Test Cases from TRACEFILE.txt

Format: S1 S0 A3 A2 A1 A0 B3 B2 B1 B0 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0

```
0101010110 00001011 11111111
0101010111 00001100 11111111
0101011000 00001101 11111111
0101011001 00001110 11111111
0101011010 00001111 11111111
0101011011 00010000 11111111
0101011100 00010001 11111111
0101011101 00010010 11111111
0101011110 00010011 11111111
0101011111 00010100 11111111
0101100000 00000110 11111111
0101100001 00000111 11111111
0101100010 00001000 11111111
0101100011 00001001 11111111
0101100100 00001010 11111111
0101100101 00001011 11111111
0101100110 00001100 11111111
0101100111 00001101 11111111
```

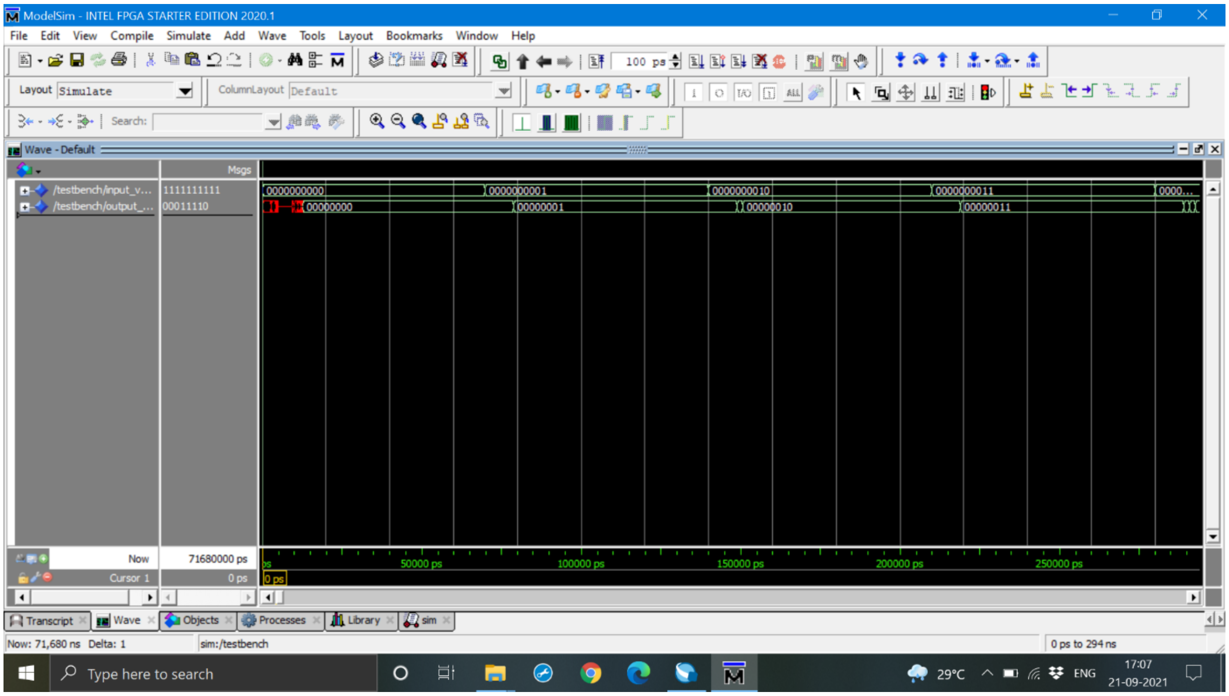
RTL Simulation:

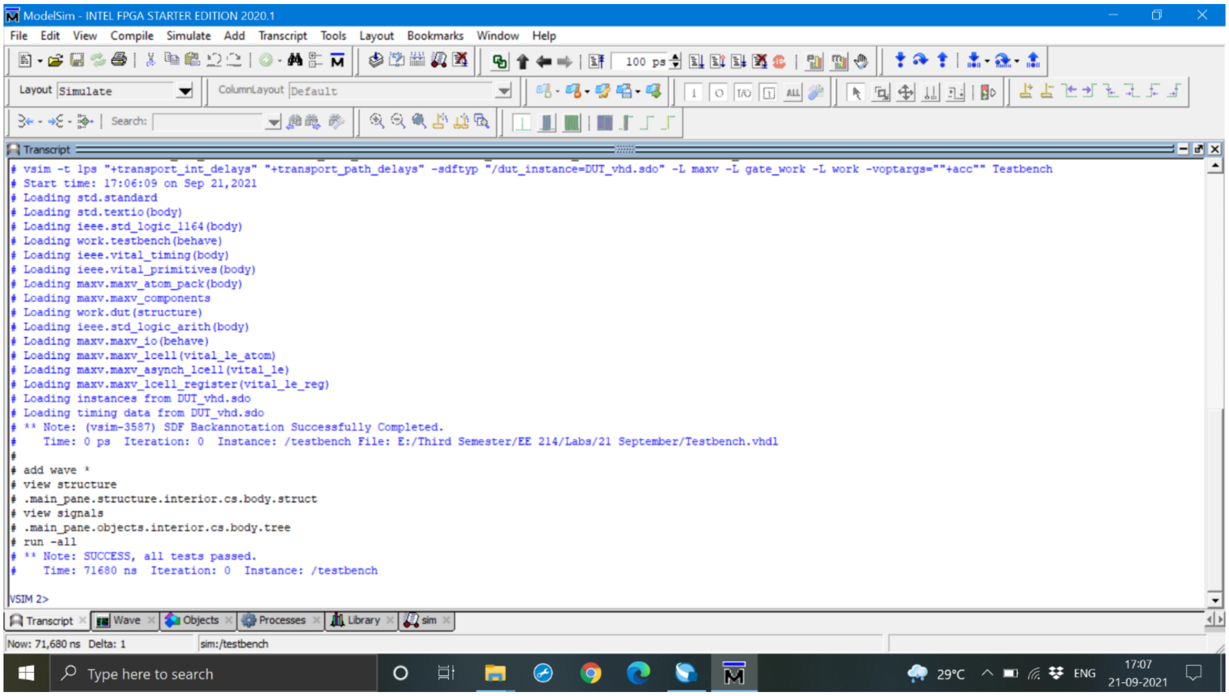



ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
Layout Simulate
ColumnLayout Default
Search:
Transcript

```
# Model Technology ModelSim - Intel FPGA Edition vcom 2020.1 Compiler 2020.02 Feb 28 2020
# Start time: 17:03:33 on Sep 21,2021
# vcom -reportprogress 300 -93 -work work E:/Third Semester/EE 214/Labs/21 September/Testbench.vhdl
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Compiling entity Testbench
# -- Compiling architecture Behave of Testbench
# End time: 17:03:33 on Sep 21,2021, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs="+acc"  Testbench
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs=""+acc"" Testbench
# Start time: 17:03:33 on Sep 21,2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading work.dut(dutwrap)
# Loading work.alu_beh(al)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 71680 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Gate-level Simulation:

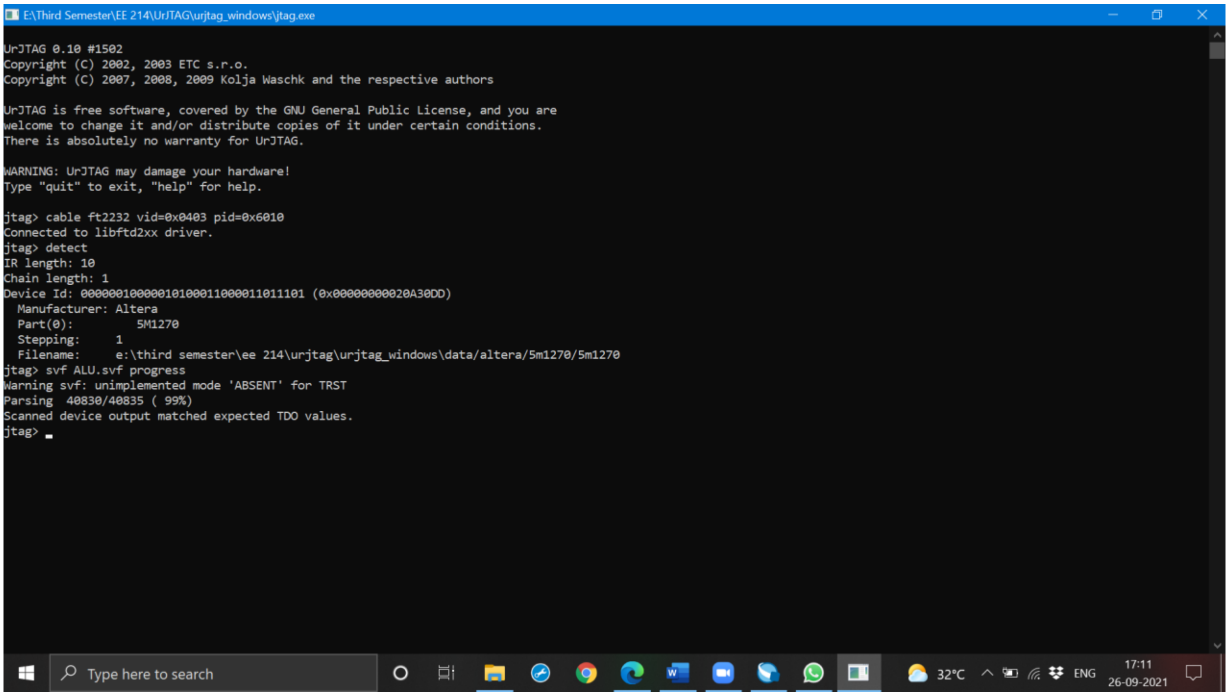




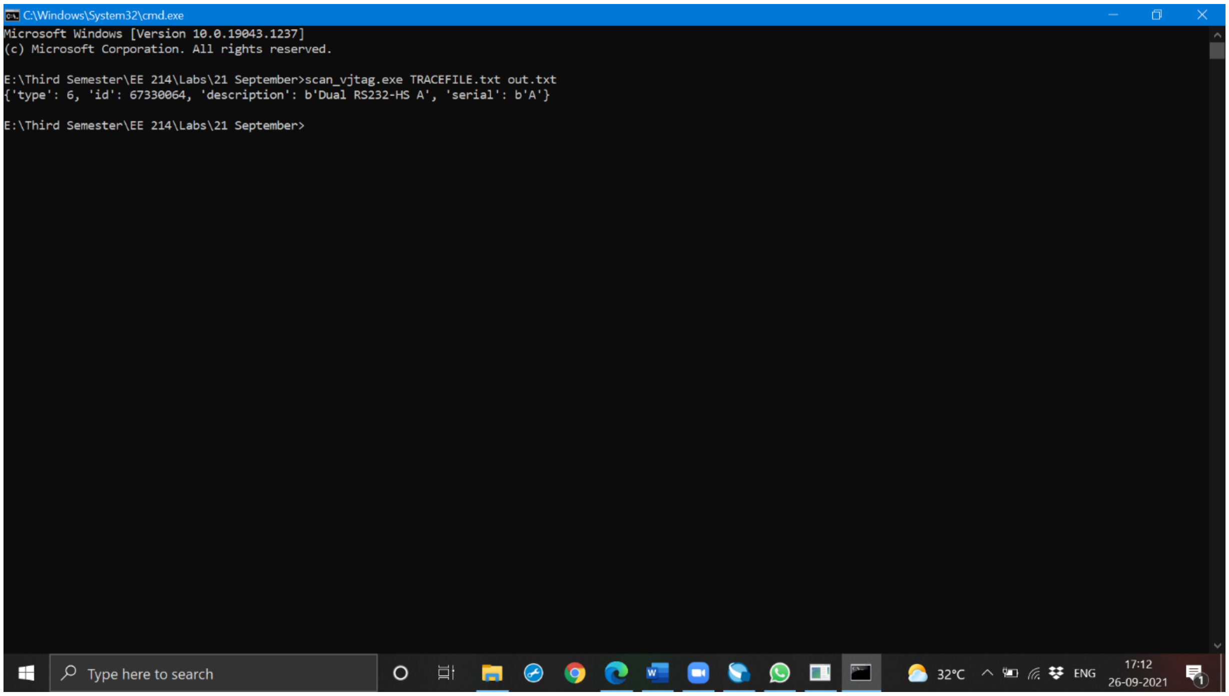
```
# vsim -t 1ps "+transport_int_delays" "+transport_path_delays" -sdftyp "/dut_instance=DUT_vhd.sdo" -L maxv -L gate_work -L work -voptargs=""+acc"" Testbench
# Start time: 17:06:09 on Sep 21,2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading ieee.vital_timing(body)
# Loading ieee.vital_primitives(body)
# Loading maxv.maxv_atom_pack(body)
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behave)
# Loading maxv.maxv_lcell(vital_le_atom)
# Loading maxv.maxv_asynch_lcell(vital_le)
# Loading maxv.maxv_lcell_register(vital_le_reg)
# Loading instances from DUT_vhd.sdo
# Loading timing data from DUT_vhd.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testbench File: E:/Third Semester/EE 214/Labs/21 September/Testbench.vhdl
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 71680 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Krypton board:

We have used scanchain for this experiment. So out.txt has output which I got using scanchain.



```
UrJTAG 0.10 #1502
Copyright (C) 2002, 2003 ETC s.r.o.
Copyright (C) 2007, 2008, 2009 Kolja Waschk and the respective authors

UrJTAG is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
There is absolutely no warranty for UrJTAG.

WARNING: UrJTAG may damage your hardware!
Type "quit" to exit, "help" for help.

jtag> cable ft2232 vid=0x0403 pid=0x6010
Connected to libftd2xx driver.
jtag> detect
IR length: 10
Chain length: 1
Device Id: 00000010000010100011000011011101 (0x00000000020A30DD)
  Manufacturer: Altera
  Part(0):         5M1270
  Stepping:     1
  Filename:     e:\third semester\ee 214\urjtag\urjtag_windows\data/altera/5m1270/5m1270
jtag> svf ALU.svf progress
Warning svf: unimplemented mode 'ABSENT' for TRST
Parsing  40830/40835 ( 99%)
Scanned device output matched expected TDO values.
jtag> _
```



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

E:\Third Semester\EE 214\Labs\21 September>scan_vjtag.exe TRACEFILE.txt out.txt
{'type': 6, 'id': 67330064, 'description': b'Dual RS232-HS A', 'serial': b'A'}

E:\Third Semester\EE 214\Labs\21 September>
```

Some outputs from out.txt

```
0111011011 00011000 Success
0111011100 00011001 Success
0111011101 00011010 Success
0111011110 00011011 Success
0111011111 00011100 Success
0111100000 00001110 Success
0111100001 00001111 Success
0111100010 00010000 Success
0111100011 00010001 Success
0111100100 00010010 Success
0111100101 00010011 Success
0111100110 00010100 Success
0111100111 00010101 Success
0111101000 00010110 Success
0111101001 00010111 Success
0111101010 00011000 Success
0111101011 00011001 Success
0111101100 00011010 Success
0111101101 00011011 Success
0111101110 00011100 Success
```

Observations:

The main observation and learning from this experiment were to how to implement the functions using behavioral style.
We can see that if S1='0' and S2='0' then our design will concatenate two 4-bit inputs A and B and report it as 8-bit output.
if S1='0' and S2='1' then our design will add two 4-bit inputs A and B and report it as 8-bit output.
if S1='1' and S2='0' then our design will do bitwise xor of two 4-bit inputs A and B and report it as 8-bit output.
if S1='1' and S2='1' then our design will add two 4-bit inputs A and A and report it as 8-bit output 2A.

References:

My main reference was our course webpage it contained many useful things such as a sample code and many other specifications. This I used as my reference.