# Experiment 4: ALU

Deepankar Sehra, Roll Number 20D070024
EE-214, WEL, IIT Bombay
September 22, 2021

## Overview of the experiment:

In two to three paragraphs, summarize

- The purpose of the experiment.

The purpose was to design an ALU which performed various functions depending on the select lines chosen.

- What you did to perform the experiment.

We were to make changes in the skeleton code to make the functions achieve their aim, which was subtraction and simply shifting the input bits by some given number and roll around the rest of the bits.

- Organization of your report and a summary of the data you will be presenting.

I will be presenting the idea I used to make sure that the ALU worked perfectly, and attach the RTL simulation, gate level simulation and scanchain results as screenshots.

## Approach to the experiment:

Describe the approach you have used to complete the assignment. You must draw a schematic/block diagram of the design (hand-drawn/software).

We were given the skeleton code for designing the ALU circuit. We had to add the necessary logic blocks that were needed to make the functions work.
For designing the rolling part of the input A, after certain value of the “tmp” given the shifting would continue but with rolling around accompanied as well.
For designing the subtraction part of A and B inputs, we had to use the logic of adder subtractor that we made in the earlier lab to get this but with one of the inputs inverted, here B. The output bits would have the subtracted 4 bits starting from the LSB and the 5th bit as carry and rest of the 8 initialized to zero.
For the third part where we had to apply the nor function to the given inputs was straight forward, just used the nor keyword between A and B, and concatenated zeros as the first four bits in output.
For the fourth part where we had to output 4 multiplied by the input A, the multiplication basically meant shifting the inputs by a value of n, where the value of n comes from 2^n = 4 (or the number with which we multiply the vector to. So I just supplied the rolling function but with the appropriate input as 2 for shifting.

## Design document and VHDL code if relevant:

You should give a brief description of whatever designs you have constructed and a sketch (architecture of main logic) of the code you have written as part of the experiment.

This is the for loop, for the subtract function. Here we basically used the concept of adder-subtractor we made for lab experiment 1, where we just used adder but with the one of the input vectors with inverted values so that may consider the subtraction that we want to perform. Here, I also declared the A and B inputs as “aa” and “bb” 4bit wide vectors.

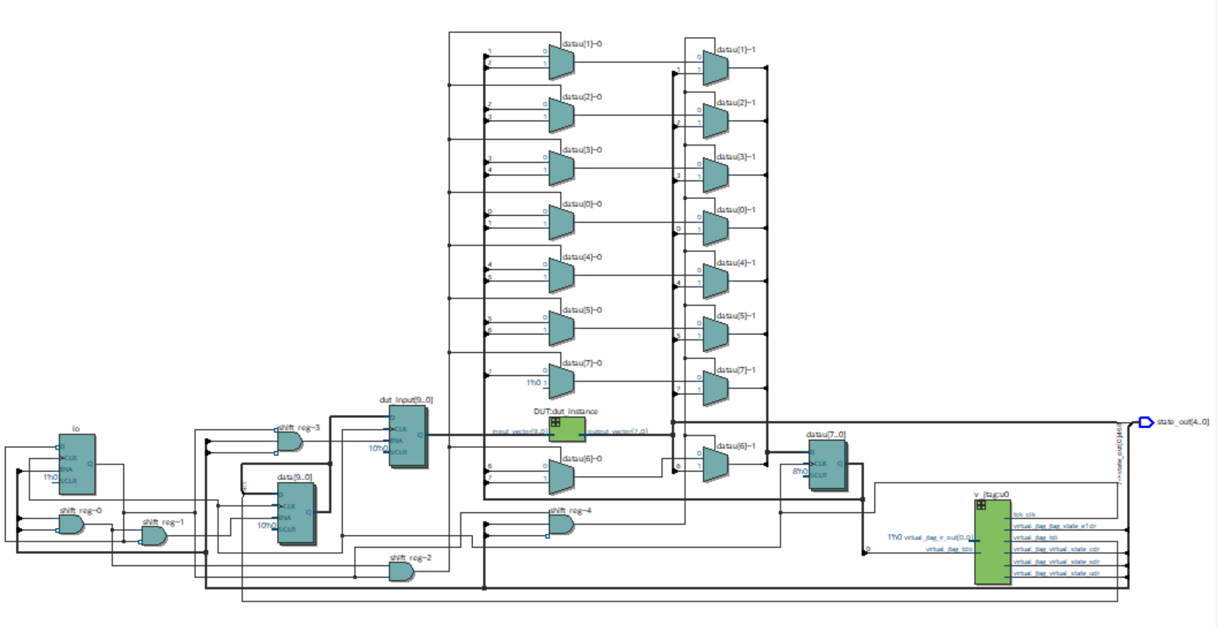
```
for i in 0 to 7 loop
            diff(i) := aa(i) xor (not bb(i)) xor carry;
            carry := (aa(i) and (not bb(i))) or ((not bb(i)) and carry) or (carry and aa(i));
end loop;
```

This block of code contains two for loops, one for calculating the exponential value of “tmp” here and the other for the amount of shifting to be done with respect to the value of “tmp”. So when the value of “i+tmp” (here, i is the for loop variable) was greater than 7, we had to roll the 4 bit input with the respect of value of “i+tmp-8”, and when “i+tmp” was less than equal to 7 we could just shift input A directly with the same value of “i+tmp”.

```
for i in 0 to 2 loop
            if (B(i)='1') then
            tmp:=tmp+2**i;
            end if;
end loop;
for i in 0 to 3 loop
            if ((i+tmp)<8) then
            shift(i+tmp) := A(i);
            elsif ((i+tmp>7)) then
            shift(i+tmp-8) := A(i);
            end if;
end loop;
```

## RTL View:

Attach screen-shot of the RTL view generated by Quartus.



# DUT Input/Output Format:

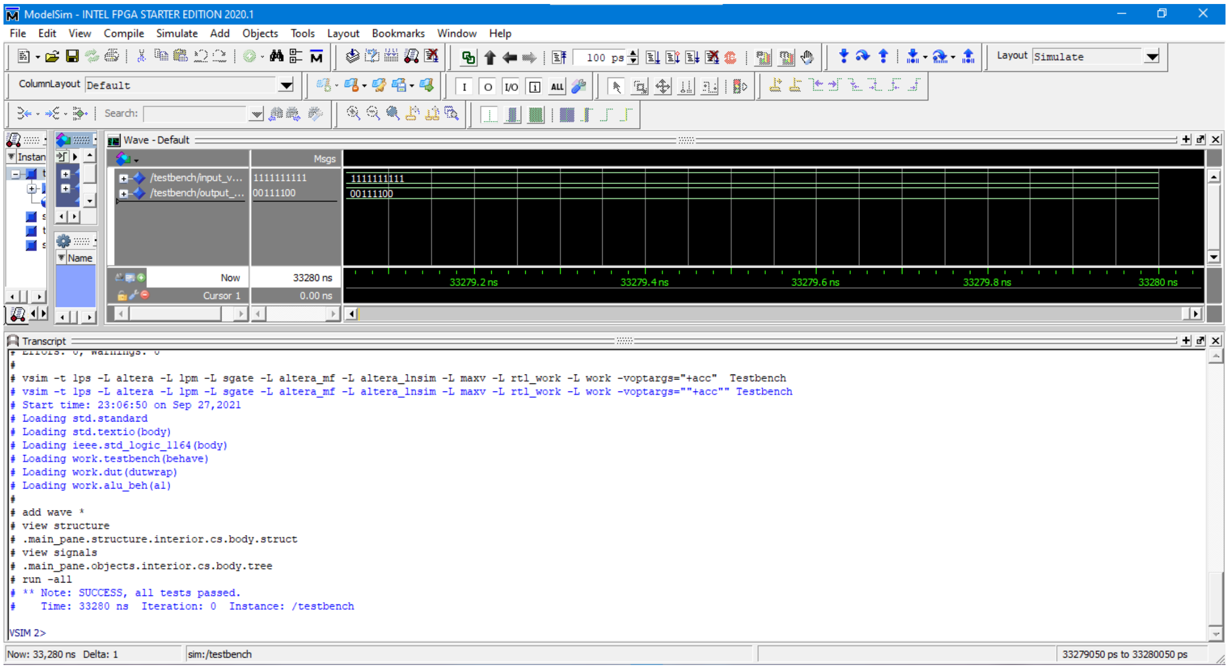
Mention the format (LSB/MSB of input and output) and few test cases from trace-file.

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

```
0000000000 00000000 11111111
0000000001 00000000
0000000010 00000000
0000000011 00000000
0000000100 00000000
0000000101 00000000
0000000110 00000000
0000000111 00000000
0000001000 00000000
```

# RTL Simulation:

Attach the clearly visible screen-shot of RTL simulation waveforms.



| | Msgs | |
|---|---|---|
| /testbench/input_v... | 1111111111 | 1111111111 |
| /testbench/output_... | 00111100 | 00111100 |
| Now | 33280 ns | |
| Cursor 1 | 0.00 ns | |

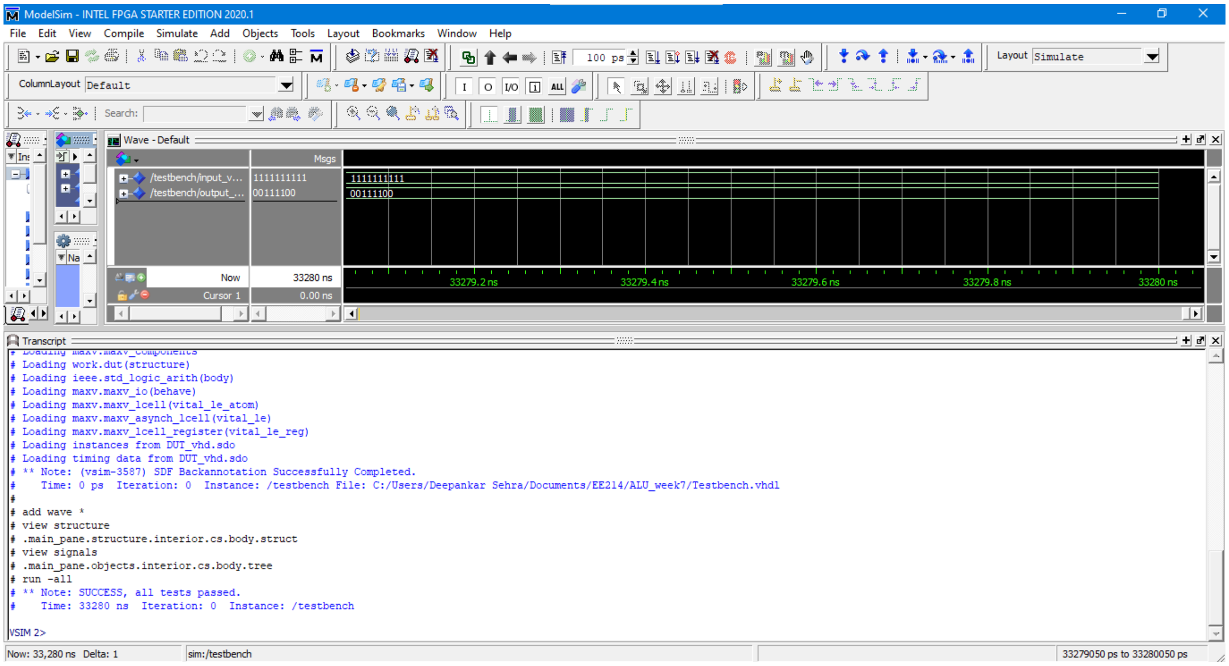
```
# Errors: 0, Warnings: 0
#
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs="+acc"  Testbench
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs=""+acc"" Testbench
# Start time: 23:06:50 on Sep 27,2021
# 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: 33280 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Now: 33,280 ns  Delta: 1    sim:/testbench    33279050 ps to 33280050 ps

# Gate-level Simulation:

Attach the clearly visible screen-shot of Gate-level Simulation.



ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Objects Tools Layout Bookmarks Window Help
Layout Simulate
ColumnLayout Default
Search:
Wave - Default
Msgs
33279.2 ns
33279.4 ns
33279.6 ns
33279.8 ns
33280 ns
Transcript

```
/testbench/input_v...   1111111111
/testbench/output_...   00111100
Now                     33280 ns
Cursor 1                0.00 ns
```

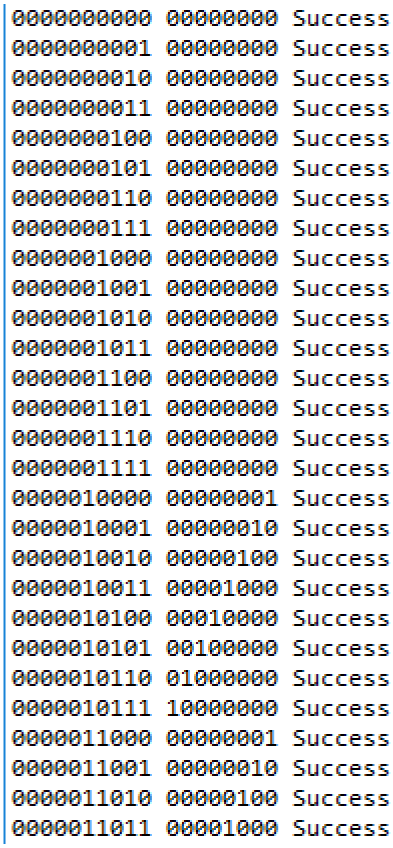
```
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behave)
# Loading maxv.maxv_lcell(vital_le_atom)
# Loading maxv.maxv_asynch_lcell(vital_le)
# Loading maxv.maxv_lcell_register(vital_le_reg)
# Loading instances from DUT_vhd.sdo
# Loading timing data from DUT_vhd.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testbench File: C:/Users/Deepankar Sehra/Documents/EE214/ALU_week7/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: 33280 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

```
Now: 33,280 ns  Delta: 1    sim:/testbench    33279050 ps to 33280050 ps
```

## Krypton board*:

Map the logic circuit to the Krypton board and attach the images of the pin assignment and output observed on the board (switches/LEDs).
Screenshots of some of the scanchain results:



```
0000000000 00000000 Success
0000000001 00000000 Success
0000000010 00000000 Success
0000000011 00000000 Success
0000000100 00000000 Success
0000000101 00000000 Success
0000000110 00000000 Success
0000000111 00000000 Success
0000001000 00000000 Success
0000001001 00000000 Success
0000001010 00000000 Success
0000001011 00000000 Success
0000001100 00000000 Success
0000001101 00000000 Success
0000001110 00000000 Success
0000001111 00000000 Success
0000010000 00000001 Success
0000010001 00000010 Success
0000010010 00000100 Success
0000010011 00001000 Success
0000010100 00010000 Success
0000010101 00100000 Success
0000010110 01000000 Success
0000010111 10000000 Success
0000011000 00000001 Success
0000011001 00000010 Success
0000011010 00000100 Success
0000011011 00001000 Success
```

## Observations*:

You must summarize your observations, either in words, using figures and/or tables.

## References:

You may include the references if any.