Faculty of Electrical & Electronics Engineering
BEE3233 Electronics System Design

Laboratory 2: Arithmetic Logic Unit (ALU)

Mapping CO, PO, Domain, KI : CO2,PO3,P5,CTPS5
CO2: Construct logic circuit using HDL.[PO3, P5, CTPS3] CO3: Design a digital system using combinational & sequential (medium scale integrated logic) MSI component. PO3: Identify, formulate and provide effective solution to engineering problem P5: Complex Overt Response CTPS3: Ability to get ideas and find alternative solutions

Learning Outcomes:

a) Design a 4 bit full adder
b) Design a 4 bit multiplier
c) Develop an ALU using multiplexer
d) Use testbench to verify the ALU designs

General Xilinx Tips

1. SAVE EARLY AND OFTEN (in your own memory device!!)

   Xilinx is notorious for crashing at the most inopportune times. Do yourself a favor and save.

2. At the end of a lab session (or any work session), archive your project using the Xilinx utility (this will ensure you save everything), and save this zip archive on your ENIAC drive or on a flash drive. Do NOT assume files will remain on the lab computers or that “your” computer will be available at a later time.

3. Make sure all components are connected. Loose wires are a frequent cause of problems.

4. Try your hand at debugging first before calling me ☺. You will learn a lot by struggling through problems that seem hard at first.

5. Read all instructions carefully before starting the lab. Often, there will be a little detail that ends up being very important.

6. Make sure you test all important cases, particularly edge/corner cases. You can be sure that your TA will test these as part of the demo.

Background of Arithmetic Logic Unit

An Arithmetic and Logic Unit (ALU) is a combinational circuit that performs logical and arithmetic operations on a pair of n-bit operands (in our case, A[3:0] and B[3:0]). Unless otherwise stated, you can assume that the inputs A and B are signed two's complement numbers when they are presented to the input of the ALU. The operations performed by an ALU are controlled by a set of operation-select inputs. In this lab you will design a 4-bit ALU with 4 operation-select inputs, S[3:0]. Logical operations take place on the bits that comprise a value (known as bitwise operations), while arithmetic operations treat inputs and outputs as two's complement integers. Errors must be detected by the ALU, specifically when A is equal to zero; if any occur, enable the Error signal. If an addition results in overflow or a multiplication results in a value that cannot be shortened to 4 bits, enable the Overflow output. The 16 functions performed by the ALU are specified in Table 1.

Table 1: ALU Arithmetic and Logic Function

Logical Function				
S_3	S_2	S_1	S_0	Output (3:0)
0	0	0	0	A
0	0	0	1	A **AND** B
0	0	1	0	A **OR** B
0	0	1	1	A **XOR** B
0	1	0	0	**NOT** A
0	1	0	1	**NOT** (A **AND** B)
0	1	1	0	**NOT** (A **OR** B)
0	1	1	1	**NOT** (A **XOR** B)
Arithmetic Function				
1	0	0	0	A+1 (increment)
1	0	0	1	A-1 (decrement)
1	0	1	0	A (transfer)
1	0	1	1	A+B
1	1	0	0	A+B+1 (carry in)
1	1	0	1	A*B (multiply)
1	1	1	0	A+ **NOT** B + 1 (subtraction)
1	1	1	1	0 (reset)

Section A – ALU Entity

1. Create the ALU entity. The port data for the intended module is as below.

Input:

____
____
____
____
____

Output:

____
____
____

Section B – Creating the macros for the ALU

Since this is a mini-design lab, you are not required you to use schematics or VERILOG explicitly. Instead, for the most part, you may choose either method depending on what you find easier.

1. You may use schematic entry for any and all modules
3. In general, you may reuse modules from previous labs. However, note that your adder (miniProject 1) will not work here as they are unsigned and only operate on single-bit input.

macro1 – 4 bit full adder
macro2 – 4 bit multiplier

Section C – Designing the ALU

Use a case statement within a clocked process to describe the functionality for the ALU as shown in Table 1.

Section D – Verifying the ALU Design

Perform a syntax check, create a test bench and simulate the design

Section E – Implement the ALU on FPGA Spartan Development Board and Verify Your Design

For Implementation steps, please refer to the reference.

NAME:-
ID:-

Evaluation Lab 2 (7%)

Print this and present it to me when you demonstrate your work.

Requirement	Points
Complete simulation (1st lab sessions)	/2
Complete implementation (2nd lab sessions)	/2
Verilog syntax code for the ALU	/3
Accurate simulation of the ALU	/3
Demonstrate the ALU on FPGA Development Board	/3
Display the output of ALU on LCD display	/4
Accurate description of post-implementation report (Q4,Q5,Q6)	/3
Total	/20

1. ALU. A) schematic diagram B) syntax code

2. Write the VERILOG syntax code for the ALU – case statement

3. Sketch / snapshot the simulation results of the ALU

4. Record the FPGA resources consumed by your ALU design

FPGA Resources	Utilization	Total Available	Percentage Utilization
Slice LUTs (LookUp Tables)			
Bounded IOBs			

5. What is the maximum estimated frequency at which this design in your ALU can run?

   a. What is the corresponding minimum clock period?

   b. Maximum Frequency: ________ MHz Minimum Clock Period: ______ ns

6. Record the path reported under "Pad to Pad Report" – Place and Route Report.

Source Pad	**Destination Pad**	**Delay**

* Please hand in the design summary report of your ALU design along with this Lab 2 – Evaluation form.