Envision ALU Report

Kruti Deepan Panda
Project Mentor
krutideepanpanda.191ec126@nitk.edu.in

Vineeth Narayan
Project Mentor
vineethnarayan.191ec260@nitk.edu.in

Harsh Nahata
harshnahata.201ee123@nitk.edu.in

Akheel Muhammed
akheelmuhammed.201it206@nitk.edu.in

Bote Sharavani Rajesh
boteshravani.201ee115@nitk.edu.in

Rishee K
risheek.201ee244@nitk.edu.in

***Abstract*—A project to design a simple ALU that can perform some of the most basic arithmetic and logical operations.**

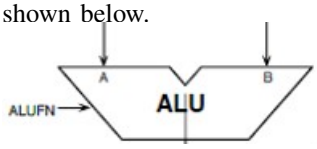
***Index Terms*—Envision, NITK, IEEE, computer architecture, ALU**

I. Introduction

Microprocessors are some of the most widely used components in circuits. They are present in laptops, desktops, mobiles, cars, and many more devices. Every microprocessor has an essential element called the ALU. The ALU performs the basic arithmetic and logical operations in a circuit.
The aim of this project is to design and simulate a 32 bit ALU, using a hardware description language (Verilog). The ALU consists of 9 different operations, based on operation code (OP code) the ALU decides which operation is to be done among various 9 operations. Basically, it is a combinational circuit that takes 32-bit data as input and gives 32-bit output by performing specified arithmetic and logical operations.

II. Designing the ALU

ALU is a combinational circuit taking two 32-bit data words A and B as the input, and producing a 32-bit output Y by performing the specified arithmetic or logical operation on the A and B inputs. The operation to be executed by the ALU is specified by a 6-bit control input. The ALU also has a Z flag and an N flag as output. The Z flag signifies whether or not the output is zero. The N flag signifies whether or not the output is negative. The logical symbol of the ALU is shown below.



The ALU consists of 9 different operations .The operations and their Boolean expressions are listed in following table -

No.	Operation	Boolean expression
1.	Full adder	t[32:0] = A + B + Cin sum = t[31:0] Cout = t[32]
2.	Full subtractor	t[32:0] = A – B – Bin diff = t[31:0] Bout = t[32]
3.	Logical left shift	Y =A << B
4.	Logical right shift	Y = A >> B
5.	Arithmetic right shift	Y = A>>>B
6.	Equality	Y = A == B
7.	Inequality	Y = A != B
8.	Less than or equal to	Y = A <= B
9.	Greater than	Y = A > B

The following are the design of the nine components of the ALU namely -

• **FULL ADDER:** we have assigned the total as the sum of the two numbers and a carry the 33rd bit will be assigned to the carry bit and the first 32 bits as the output.

• **FULL SUBTRACTER:** we have assigned the total as the difference between the two numbers and the borrow in. The borrow bit is assigned under an if condition the input 'a' is less than 'b' then the borrow bit is 1 bit or else 0.

• **EQUAL TO:** we have used the logical operator("==") to check equal to which gives 1 if true or else 0.

• **NOT EQUAL TO:** we have used the logical operator("!=") to check not equal to which gives 1 if true or else 0

• **LESSER THAN OR EQUAL TO:** we have used the logical operator less than or equal to ("¡=")to check the two inputs and return 1 if true or else 0.

• **GREATER THAN OR:** we have used the logical operator greater than("¿") to check the two inputs and return

1 if true or else 0.

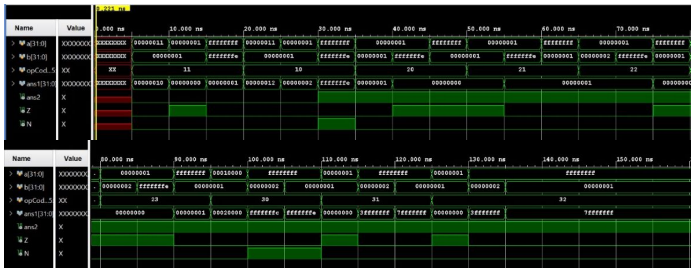
• **LOGICAL RIGHT SHIFT:** we have used the logical operator logical right shift(“¿¿”) to shift the input.

• **LOGICAL LEFT SHIFT:** we have used the logical operator logical left shift (“¡¡”)to shift the input.

• **ARITHMETIC RIGHT SHIFT:** we have used the operator arithmetic right shift(“ ¿¿¿”)to shift the input.

III. Simulation

The ALU design is simulated using the ALU test bench. The test bench consists of several different inputs for two variables and a case selection input to determine which operation is to be executed by the ALU. The test bench then simulates these inputs on the ALU design module and generates outputs in the form of a wave. The simulation has been generated using the “Xilinx Vivado” platform.
There are 9 different operations in the ALU simulated using the test bench starting from subtraction, addition, equality check, not equal to check, less than or equal to check, greater than check, left shift, right shift, and arithmetic right shift.
The case selection input which determines the operation to be executed is represented by the variable opCode in the waveform. The value of opCode for each operation (in hexadecimal) is as follows -



Subtraction -0x11
Addition -0x10
equality check -0x20
not equal to check -0x21
less than or equal to check -0x22
greater than check -0x23
left shift -0x30
right shift -0x31
arithmetic right shift -0x32

These operations take 2 inputs represented by ‘a’ and ‘b’.The output generated by them is then stored in the register variable ‘ ans1’ The N and Z outputs are flags that show whether the output ans1 is negative or not and zero or not respectively.

IV. Results

The ALU design supported nine operations – Addition, Subtraction, Equality check between 2 numbers, not equal to check between 2 numbers, ’greater than and less than ’ or ’equal to’ check between 2 numbers, logical left shift, logical right shift, and lastly arithmetic right shift. The design was simulated using the test bench and after a successful run produced the following outputs.

V. Conclusion

The 32 bit ALU was successfully designed and was simulated accordingly. It was tested on various conditions in the testbench and simulated the correct outputs. Therefore, we succeeded in programming a 32 bit ALU which performs the desired tasks. The future scope of this project can be the planning and development of a CU. The CU can be integrated with this ALU which in turn helps us understand the overall working.

References

[1] https://youtube.com/playlist?list=PLBlnK6fEyqRjMH3mWf6kwqiTbT798eAOm
[2] https://youtu.be/B7uAD11uRMc
[3] https://youtu.be/yHLXpg_-LOg
[4] https://youtu.be/tmgOkwX8YtA
[5] basics of digital electronics
[6] Basic of sequential circuits
[7] Basics of Verilog