Module Description:

ALU_1bit.v

Implements a 1 bit Arithmetic Logic Unit (ALU) for basic operations like addition, subtraction, and logical operations on 1 bit operands.

ALU_1bit_tb.v

Testbench for the 1 bit ALU, used to verify the correct behavior of the ALU_1bit module.

ALU_Nbit.v

Implements an N bit ALU by cascading multiple 1-bit ALUs, allowing for multi-bit arithmetic and logic operations.

ALU_tb.v

Testbench for the N bit ALU, used to verify the ALU_Nbit module is working correctly.

ALU_top.v

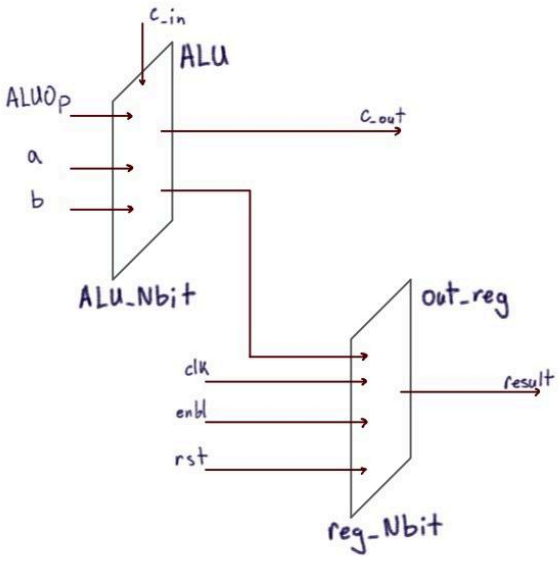
Top-level module that integrates the N bit ALU and N bit register to form a complete system for testing or application.

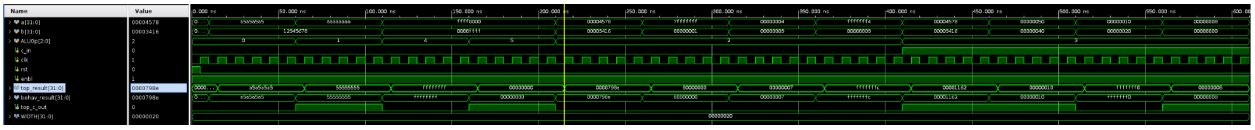
reg_Nbit.v

Implements an N-bit register that stores data for use in operations like storing intermediate or final results.

The module design hierarchy is as follows: The *ALU_top* module calls the *ALU_Nbit* and *reg_Nbit* modules. My *reg_Nbit* module does not require any additional modules to function. The *ALU_Nbit* module calls *ALU_1bit* however many times as needed depending on the parameter, but in this case, 32 times (0 to 31). Contained within each *ALU_1bit* modules are several modules describing *AND*, *OR*, and similar gate operations, due to the lab being a structural Verilog assignment.

Below, I've included a drawing of the elaborated design of lab 4.





Above is the waveform diagram produced by the ALU I designed, as well as the behavioral ALU which was included in the lab. If the image is difficult to view on the report, I have also included it as a separate PNG in my submission. You'll note that while the behavioral ALU updates instantly upon new input values, the structural ALU waits to update until the next positive clock edge, which is expected. Otherwise, the outputs of the two ALU's are identical, which is what we'd hope to see if both are working properly.