Module Descriptions:

Not1:
Behavioral module that takes one input and “nots” it.

Or1:
Structural module that takes two inputs and “ors” them using a structural “or” gate.

And1:
Structural module that takes two inputs and “ands” them using a structural “and” gate.

Mov1:
Behavioral module that takes one input and “copies” it into the output.

FA_str:
Structural module that takes two one-bit inputs and adds them.

gen_FA:
Parameterized generator module for creating a (parameterized) number of one-bit full adders (FA_str).

Sub1:
Parameterized generator module for creating a (parameterized) number of one-bit full adders configured for subtraction.

ALU_1bit:
One-bit module that instantiates the one-bit Not1, Or1, And1, and Mov1 modules, with a behavioral mux that assigns result to the correct module output depending on the select (operation).

ALU_32bit:
Parameterized generator module for creating a (parameterized) number of one-bit ALUs, in addition to instantiating the generator module for the adder (gen_FA), and subtractor (Sub1). This includes a behavioral mux for setting result to either the output of the 32-bit ALU, the adder, or the subtractor depending on the select (operation). I chose to instantiate the adder and subtractor generator modules here so that the c_outs are able to properly feed into the c_ins of the next full adder.

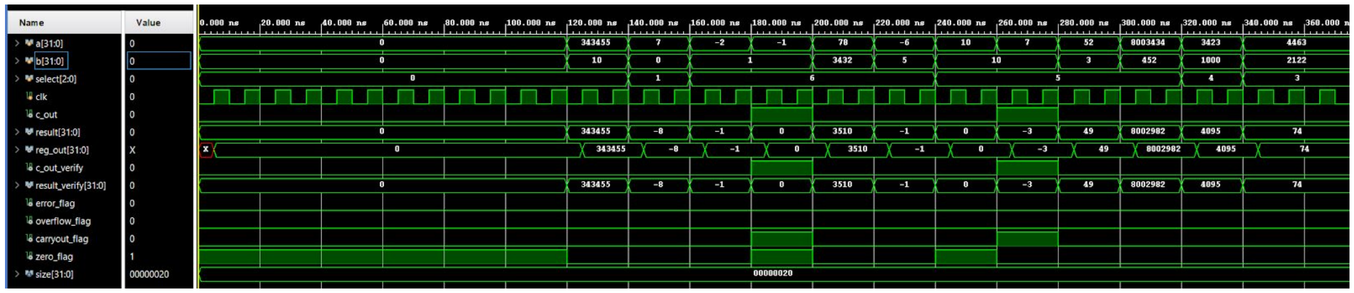
Dff:
One-bit d-flip-flop module that sets output equal to input on posedge of the clock.

Gen_dff:
Parameterized generator module for creating a (parameterized) number of one-bit d-flip-flops.

Top:
Top-level “connector” module that instantiates the d-flip-flop generator module and ALU generator module. This top module also feeds the ALU output into the input of the register, and assigns a zero_flag, overflow_flag, and carryout_flag.

tb_ALU:
(Using provided ALU testbench from Blackboard).

__**Waveform:**__



This waveform shows properly verified ALU output (result, result_verify, c_out, and c_out_verify), in addition to flags for a zero output, a carryout, an overflow, and an error. The error flag is never raised, so all outputs are correct. The output of the register (reg_out) is accurately displaying the ALU output, with a slight delay because the register is only updating on the posedge of every clock cycle.