# 16-bit Absolute Value: Implementor's Notes

Brian Carlson
brian.carlson1@uky.edu
UKY ACM Chapter
Lexington, Kentucky

## ABSTRACT

This project was used to refresh basic Verilog usage through implementing a 16 bit absolute value combinatorial circuit, including writing tests for evaluating the correctness of those solutions.

**ACM Reference Format:**
Brian Carlson. 2019. 16-bit Absolute Value: Implementor's Notes. In *Proceedings of EE480 F19: Advanced Computer Architecture (EE480 F19).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 GENERAL APPROACH

For this project, a synthesizable 2's complement 16-bit Absolute Value was constructed using combinatorial elements and no Verilog word level operators. There were many different ways to perform an absolute value, with hints given about an efficient bit twiddling solution. For the sake of code reuse later, a more straightforward approach was used. The absolute value, given by *abs*, could be found using a MUX that selects between the original value and the 2's complement of the value with the select bit being the 15th sign bit.

For the 2's complement circuit, a n-bit adder, given by the module *n_bit_adder*, was constructed from 1-bit adders given by the module *one_bit_adder*. The input was negated bitwise and 1 was added.

For the MUX from one select bit, an n-bit, two port input, 1-bit select MUX, given by *n_mux_one_select* was constructed from 1-bit two port, 1-bit select MUXs, given by *mux*.

For testing, a reference implementation of the absolute value function was built using Verilog's word-level operators, given by $ref abs$. This could be compactly represented by $b = ((a < 0)? - a : a)$ in Verilog. This acted as an oracle for comparing values against *abs*.

A module named *testbench* was created that iterated through all possible n-bit values and compared the output to of $ref abs$ and *abs*, *display*ing the times they were different and keeping count of the failed cases, summarizing the number of failed cases.

## 2 ISSUES

Inside of *abs*, I had been using the 0th bit to represent the sign bit instead of the 15th bit. This was causing roughly half of my answers to be correct and the other half to be incorrect. This was a problem that I didn't find until later in the debugging process.

While hunting for that bug, I created testbenches, *test_adder* and *test_mux* for *one_bit_adder* and *mux*, while better organizing the code for those.

*EE480 F19, January - May, 2019, Lexington, KY*
2019.