For the first conditionals, we had to make sure that the signs of the numbers were the same for Y[K] = W[k] – X[k]. If this was not the case, then Y[K] = W[k] + X[k]. Therefore, the if statement checks if both values are negative OR both values are positive (including 0). Next, if the write_address is less than 256 for Z[K] = |W[k]| – |X[k]|. If write_address is greater than 256, Z[K] = (|W[k]| + |X[k]|)/2. To do this, we created the conditional if (write_address < 9'd256).

We then created both equations and had to figure out how to create the absolute value. We started off by creating two new variables. abs_read_data_a_1 which was the absolute value of read_data_a[0] and abs_read_data_a_2 which was the absolute value of read_data_a[1]. We started by creating logic [7:0] abs_read_data_a_1 and logic [7:0] abs_read_data_a_2. In order to assign the absolute values to these variables, we created an if else statement for each variable that says if the value is not positive, then the 2s compliment of the value is found and set to abs_read_data_a_1. If the value is positive, then it will set the same value in abs_read_data_a_1. The same assignment was created for abs_read_data_a_2.

For the resource usage, we were able to count 38 registers in the main file which matches the number of registers in the compilation report. These 38 registers are made up of 9 registers each for the read and write bits, 2 bits for the write enables, 3 bits for the 3 states, and 15 registers for the absolute value addresses. These were all within the FF blocks. In the whole exercise, there was a total of 99 logic elements which was what the compilation report also says. When analyzing the timing report, our critical path is the read address 0 to read address 8 of the system. This has a delay of 3.535 ns.