E85A Project (Single-Precision Floating Point Multiplier)

The multiplication of single-precision floating point number can be divided into three main parts: the sign, the exponent, and the mantissa.

The sign bit:

The sign of the output is positive (represented by 0 in the most significant bit) if the signs of the input are similar, regardless of being negative or positive. Furthermore, the sign of the output is negative (represented by 1 in the most significant bit) if the signs of the input are different. Thus, the sign bit is the output of an XOR gate with the inputs being the two sign bits of the two inputs.

The exponent:

The exponent of the output can be initially calculated as the sum of the exponents of the inputs before being adjusted based on the mantissa. However, the exponents of the inputs are biased; thus, the bias (127) must be first subtracted from the exponents of the inputs before summing the two 8-bit exponents. The bias can be subtracted by adding the complement of 127 to each exponent; however, I opted for the most straightforward way in SystemVerilog that is subtracting 127 behaviorally without the need for complements. And since that the exponent of the output should be biased as well, the bias was re-added to the calculated exponent. When the two mantissas are multiplied, there will either be one or two non-zero bits before the decimal point. In the former, the exponent needn't any adjustments; however, in the latter, the exponent need to be incremented by 1 to keep only one non-zero bit before the decimal point.

The mantissa:

First, the implicit 1 before the decimal point should be re-concatenated to the 23-bit mantissa. Then, the two adjusted 24-bit mantissas are multiplied. The product is 48-bit to accommodate for any plausible carry out. If there is carry out and the most significant bit is 1, it implies that there are two non-zero bits before the decimal point, and accordingly the exponent is adjusted as mentioned previously; in addition, the 48-bit product is logically shifted to the right 24 bits because the mantissa can only accommodate the 23 most significant bits besides the implicit 1 before the decimal point. Thus, the 24 most significant bits are shifted to occupy the least significant 24 bits, from which we select the least significant 23 bits as the mantissa of the output and omit the 1 in the 24th bit. In the case where there is no carry out and the 48^{th} bit of the product is 0, the 48-bit product is logically shifted 23 bits instead of 24 because the most significant bits in this case are the bits [46:23]. Like the former case, we select the least significant 23 bits as the mantissa of the output and omit the 1 in the 24th bit.

The final step is concatenating the sign bit, the 8-bit exponent, and the 23-bit mantissa.

The project passed all the ten test vectors successfully.