Homework 06: Floating Point Conversion

Due on May 21 23:59

Objective

Floating point format is commonly used to represents a wide dynamic range of numeric values. The key concept is that the radix point is not fixed to a specific position. In this assignment, we are going to design a simple converter from a fixed-point number to a floating-point one in Verilog, using the finite-state machine technique

Problem Description and Design Specification

Assume that you are going to multiply two positive 8-bit integers. Instead of using a 16-bit integer to represent the result, a way-too-simple floating-point format is used. The conversion from the integer to floating-point number can be performed as follows:

1. The I/O signals of this design are as follows:
   a. RST_N: the reset signal
   b. CLK: the clock signal
   c. A, B: the two 8-bit input integers
   d. START: the control flag to indicate the two input integers are valid
   e. DONE: the control flag to indicate the output is valid
   f. Y: the fraction part of the floating-point result
   g. E: the exponent part of the floating-point result
2. Perform the integer multiplication of two positive 8-bit integers, A and B. The result is a 16-bit integer.
3. If the most significant bit (MSB) of the result is zero, we say there is a leading zero. In this case, perform the left shifting of the result until there is no leading zero.
4. Truncate the most significant eight bits of the result when there is no leading zero. This eight-bit value represents Y ($Y = 1.f$), where the MSB is always 1, and f is a 7-bit fraction value. In the meanwhile, you need to calculate the exponent E such that $Y \times 2^E$ is the resultant floating-point number. (Hint: assign a proper initial value to E, and adjust it for every left-shit.)
5. If the multiplication result is zero, the output should be 0×2^0. That is, Y=0 and E=0.

In this homework assignment, you are going to design this multiplier of two 8-bit positive integers in Verilog.

1. Refer to hw06-2.pdf for the block diagram, finite state machine, and timing diagrams of

examples. Finish the design as FPMUL module in `hw03.v`. Note that its I/O signals are declared in the ANSI-C style, which is more elegant sometimes. E is declared as a 5-bit signed integer.

2. Complete the test stimulus, `hw03_t.v`, with at least 15 different pairs of integers. Explain what kind of combinations for each pair of inputs, and the reason you choose them. Does hw06-2.pdf cover all kinds of possible combinations/categories for this problem?
3. In addition, take a look at the task and $monitor in `hw03_t.v`, to get familiar with their usage. Add necessary signals to observe in $monitor to help the debugging.
4. Verify your own design and show the waveforms of your test patterns and their output responses in nWave. You have to show at least CLK, RST_N, START, A, B, DONE, Y, E, and `state`, in the waveforms. Add any additional signals if they will help the debugging.
5. Write a report to summarize all the discussions.
   NOTE:
   No score if you simply cut-and-paste waveforms or draw some diagram without proper discussion.

Note

1. **Raise the discussion for any questions when in doubt.**
2. **Submit the source code and the electrical report based on TA's instructions.**