Floating Point Implementation UKY EE 480

Grant Cox
University of Kentucky
grant.cox@uky.edu

Josh Carroll
University of Kentucky
Josh email

3rd
University of Kentucky
email

***Abstract*—The final component of the PinKY archetecture was to be a "mutant" 16-bit floating point module (FPU). The FPU performs conversions, multiplications, reciprocals, additions, and subtractions on floats with 1 sign bit, 8 exponent bits, and 7 mantissa bits. This is a multicycle design using a state machine in Verilog. It was simulated and tested in Icarus Verilog with GtkWave.**

I. Approach

A. FTOI

B. ITOF

C. MULF

D. RECF

E. ADDF

The addition operation is the most complex. We implemented addition of two positive or two negative numbers, but were unable to implement the addition of a positive and a negative number. However, the implementation of addition was staged as follows:

1) Check the signs of the inputs. If they are the same sign, set the output sign and set the next stage to stage 3; else, set the next state stage two and make the output sign the sign of the larger number. In addition, determine which of the inputs is the larger and smaller by the exponent. With this, shift the smaller mantissa by the difference to have the same decimal place as the larger. Add the difference to the exponent of the smaller.
2) If it is a "- + +" or "+ + -" operation, 2's complement the shifted numbers from stage 1. Be careful to shift *in* the implicit 1 in the mantissa if right shifting. Go to stage 4.
3) Add the mantissas. The result is stored in a temporary register which has an extra bit on top for checking for mantissa overflow in stage 4.
4) If there is mantissa overflow, store the top 7 bits of the result into the fraction and add 1 to the exponent. If there is not overflow, store the bottom 7 bits from the mantissa addition into the fraction. If there was a mantissa subtraction and there are leading zeros, normalize the mantissa appropriately and subtract the difference from the exponent.
5) Store the result in the output.

F. SUBF

II. Issues

FTOI, ITOF, MULF, and RECF work as expected for all positive and negative numbers. ADDF works with two positive or two negative numbers. Our issues arose when working on addition with numbers of opposite signs. There is a problem with with the twos complement operation of the mantissa. It was unclear how an overflow in the mantissa addition effected the result. Should the overflow be ignored, or should some normalization process occur after the addition? This was unclear for the *general* case. We were able to compute the result for some, but not all cases, particularly when there was mantissa overflow.

Our group ran out of time to fully implement the SUBF operation. Again, this was hindered by a lack of understanding of how the implicit 1 and potential mantissa addition overflow affected twos complement result.

Our group also failed to resolve timing issues when running the floating point module with the PinKY processsor. There was difficulty understanding how to monitor the "done" signal from the FPU. The processor's interlock trigger would be driven with x's under all implementations that we tried. Even though the FPU would finish the operation, the CPU was stuck in a loop.