Lab 5 - A Nvidia TensorFloat-32 FPU

EE4449 - HCMUT

Objective and Overview

Design and implement a 19-bit floating point unit (FPU) using Nvidia's TensorFloat-32 format. It should perform basic arithmetic operations on 19-bit floating point operands, rounding results to the nearest value.

https://blogs.nvidia.com/blog/tensorfloat-32-precision-format/

This is a **group** project, to be done on your Altera DE10 board.

Note: Do **NOT** use FPU in IP CATALOG for this lab.

Schedule and Scoring

If you have not uploaded anything by the drop-dead date, we will assume you are no longer in the course. Why? Because the syllabus says you must attempt every project. Not uploading anything and not showing up to explain what you've done is not attempting the project — see the syllabus for details.

A Note about Collaboration

Project 5 is to be accomplished in groups. All work must be from your own team members.

Hints from others can be of great help, both to the hinter and the hintee.
Thus, discussions and hints about the assignment are encouraged. However, the project must be coded and written up in groups (you may not show, nor view, any source code from other students' groups). We may use automated tools to detect copying.

Use Emails/LMS to ask questions or come visit us (203B3) during office hours.
We want to see you succeed, but you have to ask for help.

Project Overview

This project is **Simulation** only, however, your synthesis result on Quartus for the DE10 Kit still affects your credit, so code should be synthesizable to resources/timing.

Functional Specifications

The FPU must implement:

- Addition
- Subtraction
- Multiplication
- Division

For any pair of 19-bit Nvidia TensorFloat-32 formatted inputs, it should perform the specified operation, round the result to the nearest value, and output a 19-bit TensorFloat-32 formatted result.

Interface (`fpu_top.sv`):

- Two 19-bit input ports for operands
- One 19-bit output port for result

Input signals to specify:

- Operation (add, subtract, multiply, divide)
- Output signals for exception flags

Implementation Guidelines:

- Decode inputs into sign, exponent, mantissa
- Align and normalize mantissas
- Perform arithmetic operation on mantissas
- Normalize result mantissa
- Round mantissa to nearest value by:
  - Truncating LSB if 0
  - Adding 1 and truncating if LSB is 1
- Check for exceptions
- Encode result back into Nvidia TensorFloat-32 format
- Testing
- Verify with testbenches, some assertions

Suggested Algorithms

You might consider using algorithms like:

- Modified Booth's Algorithm, Karatsuba, Wallace Tree, Dadda Multiplier, Toom-Cook, ... for multiplication
- Radix-n, CBEEA, SRT Division, Newton-Raphson, Goldschmidt's Algorithm, ... for division
- Kogge-Stone, Brent-Kung, Carry Propagate Adder (CPA), Carry Look-Ahead Adder (CLA), Sklansky Adder, Han-Carlson Adder, ... for addition and subtraction

For Credit

Project 5 is entirely simulation. You should write a careful testbench for it.

Some Other Things You Should Learn

- Nvidia TensorFloat-32 format
- Overview of sign, exponent, mantissa
- Special values (infinity, NaN, denormals)
- Normalization and denormalization of floating point values
- Handling underflow and overflow conditions
- Different rounding modes and how they work
- Exception handling
  - Invalid operation
  - Divide by zero
  - Inexact rounding

- Tradeoffs between floating point precision and range
- Using fixed-point vs floating-point representations
- Testing floating point designs
- Generating directed tests
- Constrained random test cases
- Verification strategies
- Floating point optimizations
    - Pipelining
    - Parallelization
    - Hardware optimizations like FMA
- Applications of floating point units
    - Scientific computing
    - 3D graphics
    - Machine learning

How To Turn In Your Solution

This semester we will be using LMS, simply submit the zip file with your reports and codes.

Demos and Late Penalty

We will have demo times outside of class times on or near the due date. Since we will demo from the files in your zip, it is possible that you'll demo on a following day.

Define Late: Lateness is determined by the file dates of your submission on LMS.