Chisel FP Generators

Mario Vega

January 16, 2025

1 Introduction

This library contains several Chisel-designed floating-point (FP) generators based on the IEEE-754 format. The designs support half-, single-, double-, and quad-word precision and are configurable through parameters. The FP designs included in this library are listed below.

- FP Adder
- FP Multiplier
- FP Divider
- FP Square Root
- FP Accumulator

For each design, parameters can be tuned to meet user needs, and the interface is simple.

2 Usage

The designs are implemented with Chisel 6.0.0 using SBT-based Scala. Base installation details can be found here if not already installed. You can clone the project if you want to work directly with the code.

If setting up your own project, refer to the official Chisel repository for setting up. Additionally, you may want to install the ChiselTest library for performing simulations and waveforms directly from the Chisel environment.

To import our Chisel FP generators, add the following line into the **build.sbt** file.

```
dependsOn(RootProject(uri("https://socks.lbl.gov/mvega/chisel-fp-generators")))
```

3 Interface

The interface is depicted in the diagram below.

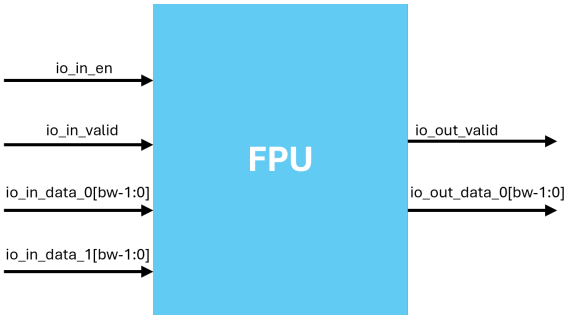

Figure 1: Interface Format

- ***io_in_en*** - Enable signal.
- ***io_in_valid*** - Input validation signal.
- ***io_in_data_x[bw-1:0]*** - Data input port, where bw is the bit width.
- ***io_out_valid*** - Output validation signal.
- ***io_out_data_x[bw-1:0]*** - Data output port.

4 Design Parameters

4.1 FP Adder/Multiplier

bw - Sets the FP precision of the unit in terms of bit width. Users can select among 16, 32, 64, and 128-bit precision.

pd - Sets the desired pipeline depth (equal to cycles) of the unit. The design supports depths of 1, 3, 7, 10, and 13 .

4.2 FP Divider/Square Root

bw - Sets the FP precision of the unit in terms of bit width. Users can select among 16, 32, 64, and 128-bit precision.

L - Sets the number of digit recurrence iterations (unrolled) for division or square root computations. The max number of iterations is equivalent to the mantissa-width of the given precision (***bw***). Setting fewer iterations will result in approximate computations and reduce resource cost.

4.3 FP Accumulator

bw - Sets the FP precision of the unit in terms of bit width. Users can select among 16, 32, 64, and 128-bit precision.

iters - Hardcodes the number of accumulations (iterations) performed per round. $iters \geq 1$

ExpExp - The forecasted largest exponent power of the final sum. For example, setting this to 10 implies we expect a final result with a magnitude of 2^{10}, which means that the final result is centered around a normalization of $1.f \times 2^{10}$.

ExpMSB - Sets the number of bits to the left of the decimal point (most significant bits) in the result register. This compensates for cases where the final result surpasses the expected magnitude (***ExpExp***). $ExpMSB \geq 1$

LSB - Sets the number of bits to the right of the decimal point (least significant bits) in the result register. Should at least be equal to the mantissa width for the given precision (***bw***), but do keep in mind that shifts during the final normalization may result in significant accuracy loss so it may be better to provide a larger width.

5 Waveform Demo

This section aims to provide insight into the interface operation of our designs.

Refer to the waveform of the FP adder below.

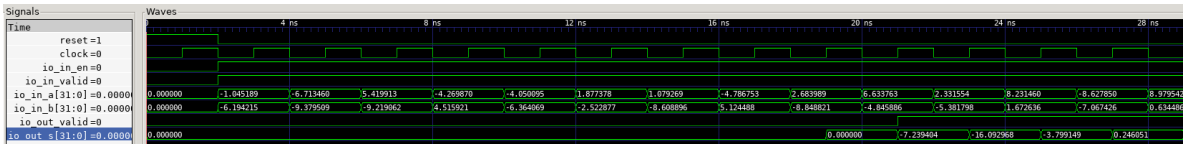

Figure 2: 32-bit FP Adder Waveform Demo

For starting an operation, set both the input valid and enable signals high and populate the data ports in the same cycle. After the first cycle, the valid signal may be kept high if additional data is to be streamed in, otherwise it can be set low if there is no more data to process. To ensure the operation is completed, the enable signal must be kept high to ensure the computation progresses to completion. Completion will be indicated by a high output valid signal along with the corresponding result on the data output port.