Abstract

In this project, we were assigned to create a 16-bit divide unit that uses Goldschmidt’s iteration to compute its operation, which must be accurate to 15 fractional bits. To complete this project, I first wrote the algorithm code in java, before designing the block diagram and subsequent SystemVerilog code.

Table of contents

Introduction

To first start this process, it’s important to know the algorithm that we are using to complete this entire project. We are using Goldschmidt’s iteration to use a multiplier to iteratively solve a division problem. In this project we were to work in 16 bits, 15 being fractional, so we are dividing floating point numbers [1,2).

Goldschmidt’s iteration works like this. Instead of creating a new form of architecture for division, we multiply by the inverse of the denominator.

However, if when this is taken to a formula as in the Newton-Raphan architecture you haft to multiply the final approximation for **1/d** by N again. Instead with the Goldschmidt’s method, we use our approximate values, and we multiply both top and bottom, until we reach our required accuracy.

Where K0 is our approximate value, and . We step through the iteration in the following steps:

1. Get our initial value, and store it in K0.
2. Multiply N by K0, and store the result.
3. Multiply D by K0, which is r0, and store the result.
4. Find value K1 for the next iteration by subtracting r0 from 2.
5. Repeat 1-3 until you reach your desired accuracy.

Next part to understand our operation is how we select for our initial value. To pick, we simply take the weighted average of our two endpoints, which for our project using floating point numbers, is 1 and 2.

Which when we take our values of 1 and 2,

Which is the value we are going to use for the rest of our operation. The only remaining question is how many iterations to go to reach our accuracy of 15 fractional bits. To find that out we . . .

Body

Results