Goldschmidt Integer Divider User Guide

Jose R. Garcia

2021-09-01

Contents

1 Abstract

The Goldschmidt integer divider written in verilog. Similar to Newton-Raphson but the division step can be pipelined. This document contains and overview of the design and guidance on the usage and integration of this component.

2 Syntax and Abbreviations

Term	Definition
0b0	Binary number syntax
0x0000_0000	Hexadecimal number syntax
bit	Single binary digit (0 or 1)
BYTE	8-bits wide data unit
DWORD	32-bits wide data unit
FPGA	Field Programmable Gate Array
GCD	Goldschmidt Convergence Division
LSB	Least Significant bit
MSB	Most Significant bit
WB	Wishbone Interface

3 Design

The Goldschmidt division is an special application of the Newton-Raphson method. This iterative divider computes:

$$d(i) = d[i-1].(2 - d[i-1])$$

$$D(i) = D[i-1].(2 - d[i-1])$$

were d is the divisor; D is the dividend; i is the step. D converges toward the quotient and d converges toward 1 at a quadratic rate. For the divisor to converge to 1 it must obviously be less than 2 therefore integers greater than 2 must be multiplied by 10 to the negative powers to shift the decimal point. Consider the following example: $\frac{16}{4}$

Step	D	d	2-d
inputs	16	4	-
0	1.6	0.4	1.6
1	2.56	0.64	1.36
2	3.4816	0.8704	1.1296
3	3.93281536	0.98320384	1.01679616
4	3.99887155603702	0.999717889009254	1.00028211099075
5	3.99999968165356	0.999999920413389	1.00000007958661
6	3.99999999999997	0.999999999999994	1.00000000000001
7	4	1	1

The code implementation compares the size of the divisor against $2 * 10^n$ were n is a natural number. The result of the comparison indicates against which 10^m, were m is a negative integer, to multiply the divisor. Then the Goldschmidt division is performed until the divisor converges to degree indicated by **P_GCD_ACCURACY**. The quotient returned is the rounded up value to which the dividend converged to. Each Goldschmidt step is performed in to two half steps in order use only half the multipliers and save resources.

The remainder calculation requires an extra clock which is why the address tag is used to make the decision on whether to do the calculation or skip it. The calculation simply takes the value after the decimal point of the quotient a multiplies it by the divisor.

4 Configurable Parameters

These are the over-writable parameters.

Parameters	Default State	Description
P_GID_FACTORS_MSB	31	Dividend, divisor and results most significant bit.
P_GID_ACCURACY_LVL	12	Divisor Convergence Threshold. How close to one does it get to accept the result. These are the 32bits after the decimal point, 0.XXXXXXXX expressed as an integer. The default value represent the 999 part of a 64bit binary fractional number equal to 0.999.
P_GID_ROUND_UP_LVL	2	Number of bits to look at after the decimal point to round up.

5 Clocks and Resets

This module only possess one clock domain and a synchronous reset signal.

Signals	Initial State	Direction	Definition
i_clk	N/A	In	Input clock. All interfaces are sampled on the positive edge of this clock signal.
i_reset_sync	N/A	In	Synchronous reset. Used to reset this unit.

6 Interfaces

The divider and divisor are received through i_master_div0_read_data and i_master _div1_read_data and qualified by the i_slave_stb. The i_slave_stb signal could be managed in different ways. It can be a pulse with the width of a singe clock to operate as a pipelined Wishbone interface and the o_master_div_write_stb can be considered as a Wishbone o_ack. I can also be operated as a Wishbone standard using those same signals.

When the division concludes the o_master_div_write_stb is asserted and writes the result to the address received through i_slave_addr.

6.1 WB4 Slave

Signals	Initial State	Dimension	Direction	Definition
i_wb4_slave_stb	N/A	1-bit	Input	Valid data strobe and start indicator.
i_wb4_slave_data	N/A	[(P_GID_FACTORS_MSB*2)+1:0]	Input	Divisor and Dividend.
i_wb4_slave_tgd	N/A	[1:0]	Input	Indicates the calculation to perform. bit[1] 0=quotient, 1=remainder; bit[0] 0=signed, 1=unsigned
o_wb4_slave_stall	N/A	1-bit	Output	Stall, not ready when set to 1.

6.2 WB4 Master

Signals	Initial State	Dimension	Direction	Definition
o_wb4_master_stb	N/A	1-bit	Input	Valid data strobe and start indicator.
o_wb4_master_data	N/A	[P_GID_FACTORS_MSB:0]	Output	Operation result.
i_wb4_master_stall	N/A	1-bit	Input	Stall, not ready when set to 1.

7 Test Bench and Simulation

Test Bench and Simulation

7.1 Test Bench

The test bench is written in uvm-python (UVM), it requires cocotb. A Wishbone Slave verification agent stimulates the DUT by sending factors for the DUT to provide the division result. A Wishbone Master verification agent is used as a monitor to capture the result. A predictor sees the Wishbone Slave transactions and consequently generates Wishbone Master transactions to compare against the response generated by the DUT. Figure 1 provides a low detailed description of the test bench components and connections

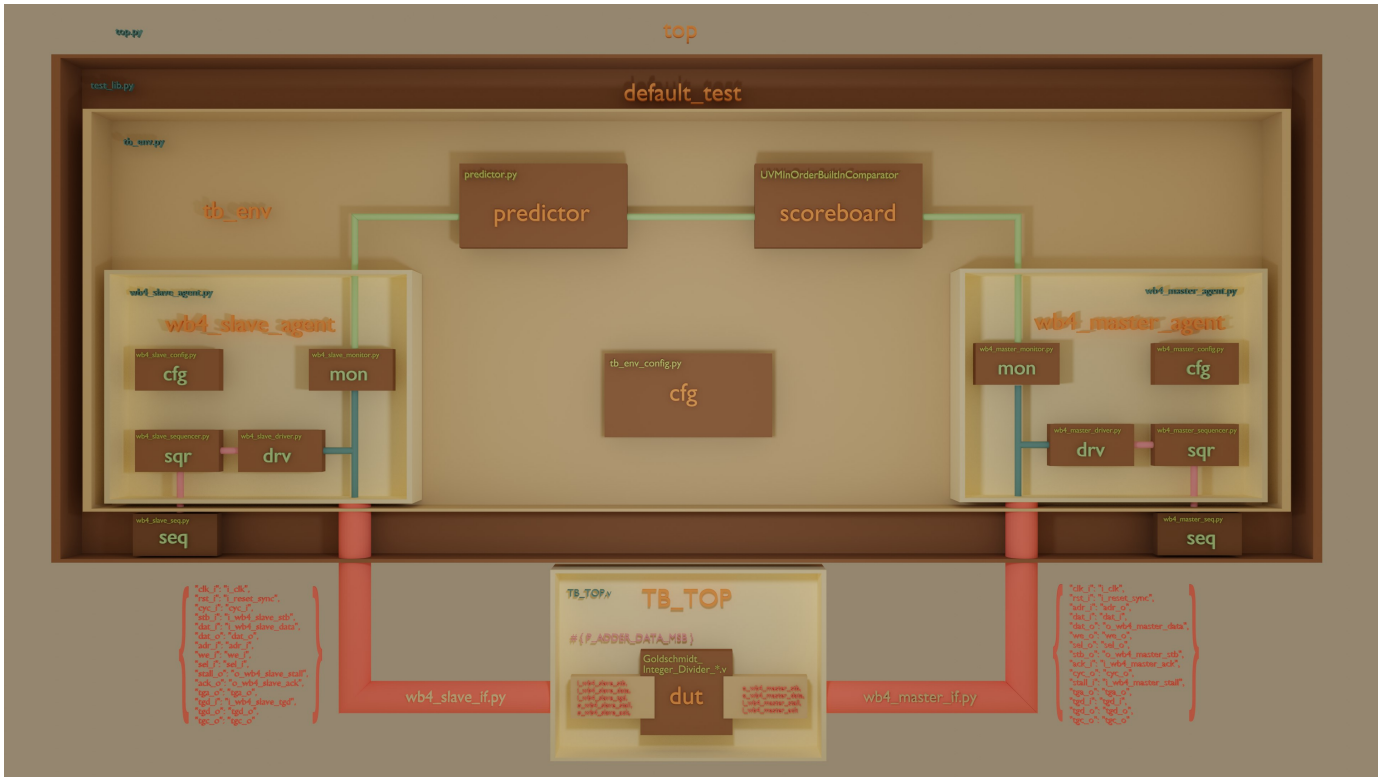

Figure 1: Test Bench Block Diagram

7.2 Simulation

7.2.1 Prerequisites

Verilator(version 4.106)
cocotb
cocotb-coverage
uvm-python
gtkwave(optional)

To install a simulator follow the instructions in in the verilator website. For setting cocotb and uvm-python:

```
sudo apt install python3-pip
pip install cocotb
pip install cocotb-coverage
git clone https://github.com/tpoikela/uvm-python.git
cd uvm-python
python -m pip install --user .
```

7.2.2 Execution

From the `/Goldschmidt_Integer_Divider/sim/` directory run the command

```
make
```

By default the two clocks per step implementation is used for the simulation. To view the wave form open the file **wave2.gtkw** with **gtkwave**. If the one clock per step is selected for simulation use **wave1.gtkw** for the proper set of signals to be loaded into the wave viewer.

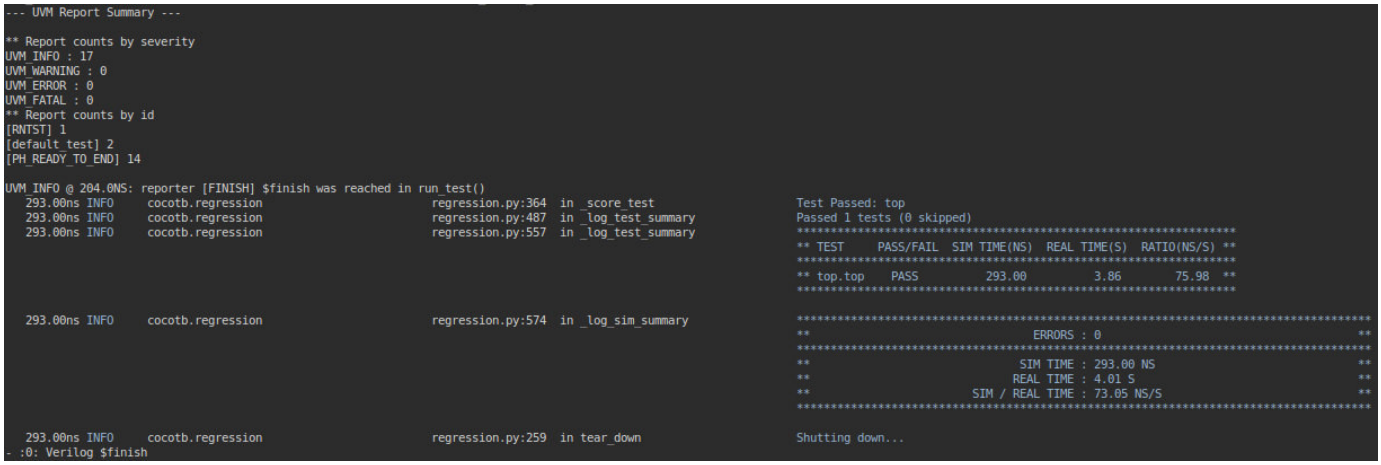

Figure 2: Sim successfully completed.