易 灵 思

Divider Core User Guide

UG-CORE-DIVIDER-v4.4
February 2023
www.elitestek.com

efinity®

Contents

Introduction

The Divider core divides a numerator input value by a denominator input value to produce a quotient and a remainder.

Use the IP Manager to select IP, customize it, and generate files. The Divider core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an 易灵思® development board.

Features

- Supports a data width of 2 to 64 bits
- Supports signed and unsigned data representation format for both the numerator and denominator
- Supports adjustable latency and disabling Divider latency
- Supports optional asynchronous clear and clock enable ports
- Verilog HDL RTL and simulation testbench
- Includes example designs targeting the Trion® T20 BGA256 Development Board and 钛金系列 Ti60 F225 Development Board

FPGA Support

The Divider core supports all Trion® and 钛金系列 FPGAs.

Resource Utilization and Performance

Note: The resources and performance values provided are just guidance and change depending on the device resource utilization, design congestion, and user design.

钛金系列 Resource Utilization and Performance

FPGA	Mode / Width (bit) / Latency	Logic and Adders	Flip-flops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)[1]	Efinity® Version[2]
Ti60 F225 C4	Signed / 8 / 8	121	87	0	0	576	2021.2
	Signed / 32 / 32	2,027	1,122	0	0	314	
	Unsigned / 8 / 8	121	86	0	0	672	
	Unsigned / 32 / 32	2,025	1,116	0	0	320	
	Signed / 8 / 4	169	79	0	0	325	
	Signed / 32 / 16	2,530	719	0	0	50	
	Unsigned / 8 / 4	130	50	0	0	379	
	Unsigned / 32 / 16	2,604	786	0	0	50	
	Signed / 8 / 0	205	0	0	0	–	
	Signed / 32 / 0	3,407	0	0	0	–	
	Unsigned / 8 / 0	175	0	0	0	–	
	Unsigned / 32 / 0	3,310	0	0	0	–	

[1] Using default parameter settings.
[2] Using Verilog HDL.

TrionResource Utilization and Performance

FPGA	Mode / Width (bit) / Latency	Logic Utilization (LUTs)	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)[(1)]	Efinity® Version[(2)]
T20 BGA256 C4	Signed / 8 / 8	167	211	0	0	240	2021.2
	Signed / 32 / 32	1,687	1,569	0	0	165	
	Unsigned / 8 / 8	116	141	0	0	263	
	Unsigned / 32 / 32	1,616	1,459	0	0	168	
	Signed / 8 / 4	147	86	0	0	104	
	Signed / 32 / 16	2,122	557	0	0	12	
	Unsigned / 8 / 4	126	81	0	0	144	
	Unsigned / 32 / 16	2,008	620	0	0	24	
	Signed / 8 / 0	206	0	0	0	–	
	Signed / 32 / 0	3,407	0	0	0	–	
	Unsigned / 8 / 0	176	0	0	0	–	
	Unsigned / 32 / 0	3,310	0	0	0	–	

Functional Description

The Divider division result is shown in the following equation:

|Numerator| = |Quotient * Denominator| + |Remainder|

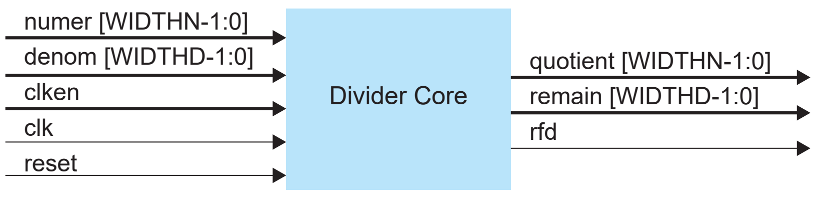
For signed mode, the remainder always stays positive, and the quotient depends on the signed bit of both numerator and denominator. Example:

- 3/2 = 1 Remainder 1
- 3/-2 = -1 Remainder 1
- -3/2 = -1 Remainder 1
- -3/-2 = 1 Remainder 1

Note: The quotient, remainder, and fractional results of division by zero are undefined. The Divider core outputs the highest possible value based on the data width for the quotient and 0 for the remainder.

You can set the numerator and denominator bit widths independently. The bit width of the quotient is equal to the bit width of the numerator, `WIDTHN`, and the bit width of the remainder is equal to the width of the denominator, `WIDTHD`.

Figure 1: Divider System Block Diagram



Ports

Table 1: Divider Ports

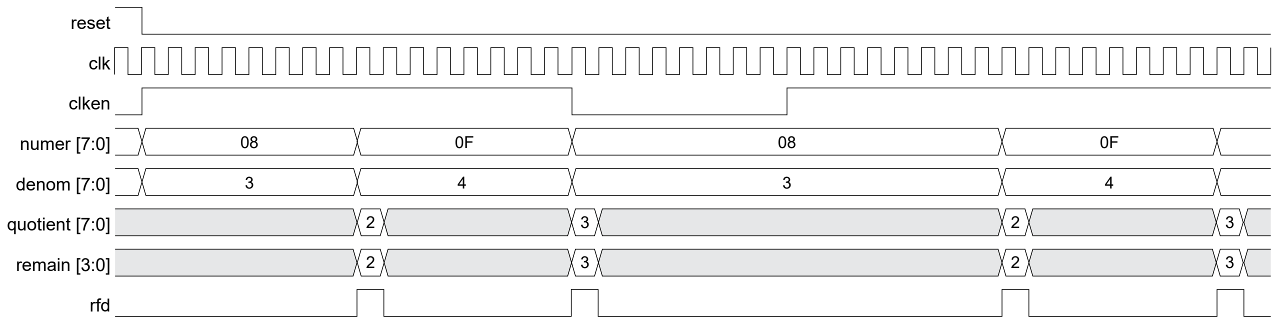
Port	Direction	Description
numer [WIDTHN-1:0]	Input	Numerator input to the Divider core for calculation.
denom [WIDTHD-1:0]	Input	Denominator input to the Divider core for calculation.
quotient [WIDTHN-1:0]	Output	Quotient output from the Divider core after calculation.
remain [WIDTHD-1:0]	Output	Remainder output from the Divider core after calculation.
rfd	Output	Ready for data signal. When Latency set to 0, this signal is constantly high. 1: Divider core calculation completed and ready for the next input. The output quotient and remainder of the previous numerator and denominator combination are updated. 0: Calculation is in progress. This port is available only when PIPELINE parameter = 0. When PIPELINE is enabled, you can issue the next numerator and denominator at any clock cycle when clken is asserted. Quotient and remainder is available after *x* clock cycles where *x* = LATENCY.
clken	Input	Clock enable. When clken is high, the Divider must wait for one clock cycle before operation continues. 1: Division operation in progress. 0: Idle.
clk	Input	System clock.
reset	Input	Asynchronous active high reset.

Divider Operations

The Divider starts taking in the numerator and denominator inputs for operation after the first clock cycle of `clken` signal assertion.

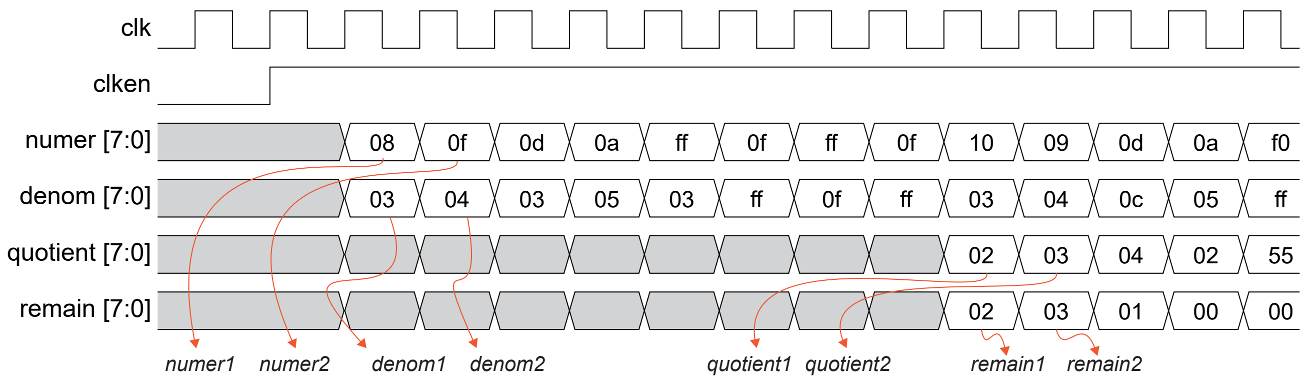
The Divider supports adjustable latency from 0 (pipeline disabled) to numerator bit width, WIDTHN (full pipeline). The following figures show the full, half, and disabled pipeline Divider operation examples.

Figure 2: 8 Bit Width with 8 Clock Latency (Pipeline Disabled) Waveform



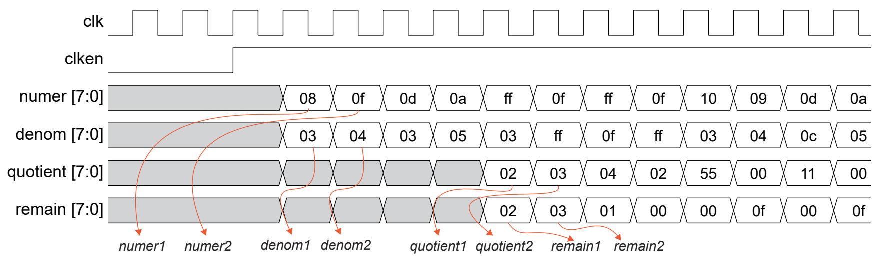
In pipeline mode, latency value from 1 to WIDTHN, the Divider goes into idle when the `clken` is low, and resumes operation 1 clock cycle after the `clken` is high. The following figure shows a full pipeline divider operation where the latency is equal to the bit width.

Figure 3: 8 Bit Width with 8 Clock Latency Waveform



Similar to the full pipeline operation but the Divider only needs 4 cycle for each output. The following figure shows a half pipeline divider operation where the latency is half of the bit width.

Figure 4: 8 Bit Width with 4 Clock Latency Waveform



IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate 易灵思® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an 易灵思 development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.

Note: Not all 易灵思 IP cores include an example design or a testbench.

Generating a Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose an IP core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.

   Note: You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the IP core's user guide or on-line help.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an 易灵思® development board and/or testbench. For SoCs, you can also optionally generate embedded software example code. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.

   Note: You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:

- **<module name>_define.vh**—Contains the customized parameters.
- **<module name>_tmpl.v**—Verilog HDL instantiation template.
- **<module name>_tmpl.vhd**—VHDL instantiation template.
- **<module name>.v**—IP source code.
- **settings.json**—Configuration file.
- **<kit name>_devkit**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains generated RTL and testbench files.

Note: Refer to the IP Manager chapter of the Efinity® Software User Guide for more information about the Efinity® IP Manager.

Customizing the Divider

The core has parameters so you can customize its function. You set the parameters in the General tab of the core's IP Configuration window.

Table 2: Divider Core Parameters

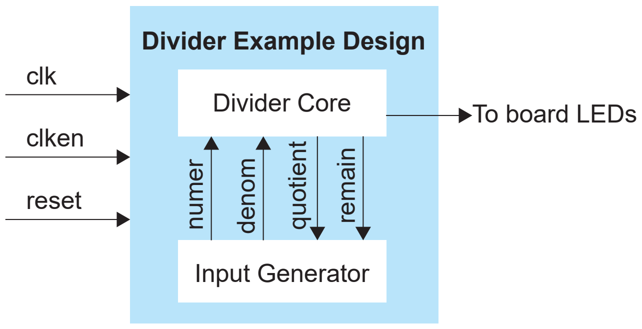
Parameters	Options	Description
Numerator Sign	SIGNED, UNSIGNED	Sign representation of numerator input. When set to SIGNED, the core interprets the numerator input as signed two's complement. Default: UNSIGNED
Numerator and Quotient Data Width	2 - 64	Defines the numerator and quotient data width. Default: 8
Denominator and Remainder Data Width	2 - 64	Defines the denominator and remainder data width. The value must not be greater than Numerator and Quotient Data Width Default: 8
Denominator Sign	SIGNED, UNSIGNED	Sign representation of denominator input. When set to SIGNED, the core interprets the denominator input as signed two's complement. Default: UNSIGNED
Enable Pipeline	Enable, Disable	Enables or disables the Divider pipeline. When disabled, rfd signal is used. You must wait for the rfd signal to be asserted before inserting the next numerator and denominator. Default: Enable
Latency	0 - WIDTHN	Defines the latency. Set this to a higher value to achieve a higher f_{MAX}. When the core latency is not a concern, you can set this to the maximum value to achieve the best f_{MAX}.

Divider Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

Important: 易灵思 tested the example design generated with the default parameter options only.

Figure 5: Example Design Block Diagram



The example design targets the Trion® T20 BGA256 Development Board and 钛金系列 Ti60 F225 Development Board. This design sends predetermined numerator and denominator combinations to the Divider core and displays the pass/fail results on the LEDs.

Note: Additional to the design with default parameters, 易灵思® also tested the core example design with signed 4 bit parameter.

Trion® T20 BGA256 Development Board

Example design input and output for the Trion® T20 BGA256 Development Board:

- *LED D4 to D3*—divider output display:
  - LED D3 turned on: Operation passed
  - LED D4 turned on: Operation failed
- *DIPswitch SW3.2*—`clken` signal to turn the Divider core on or off.
- *Pushbutton SW4*—Divider core reset.

钛金系列 Ti60 F225 Development Board

In the unsigned and signed mode, the LEDs blink from D16 blue, D16 green, D16 red, and D17 blue continuously.

Example design input and output for the 钛金系列 Ti60 F225 Development Board:

- *LED D16 to D17*—divider output display:
  - LED D16 turned on (blue): Operation passed
  - LED D17 turned on (blue): Operation failed
- *DIPswitch SW2.2*—`clken` signal to turn the Divider core on or off.
- *Pushbutton SW5*—Divider core reset.

Table 3: Trion® Example Design Implementation

FPGA	Mode / Width (bit) / Latency	Logic Utilization (LUTs)	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)[3]	Efinity® Version[4]
T20 BGA256 C4	Unsigned / 8 / 8	349	332	0	0	50	2021.2

Table 4: 钛金系列 Example Design Implementation

FPGA	Mode / Width (bit) / Latency (bit)	Logic and Adders	Flip-flops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)[3]	Efinity® Version[4]
Ti60 F225 C4	Unsigned / 8 / 8	331	209	0	0	180	2021.2

Divider Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.

Note: You must include all **.v** files generated in the **/testbench** directory in your simulation.

易灵思 provides a simulation script for you to run the testbench quickly using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` in a terminal application. You must have Modelsim installed in your computer to use this script.

The testbench compares the divider output to a Verilog division operation results. You can set the numerator and denominator values in the **divider_demo.v** file. After running the simulation successfully, the test prints the following message:

```
100500 ns---PASSED---
```

When the operation failed, the test prints the following message:

```
100500 ns---FAILED---
```

[3] Using default parameter settings.
[4] Using Verilog HDL.

Revision History

Table 5: Revision History

Date	Version	Description
February 2023	4.4	Added note about the resource and performance values in the resource and utilization table are for guidance only.
April 2022	4.3	Improved Latency parameter description. (DOC-796)
March 2022	4.2	Updated latency parameter description. (DOC-755)
January 2022	4.1	Updated rfq port and Enable Pipeline parameter descriptions. Updated testbench. Updated operation waveforms. Updated Resource Utilization and Performance tables.
December 2021	4.0	Updated simulation testbench. Added new IP Manager parameters. Updated example design. Added Divider operation waveforms.
October 2021	3.1	Added note to state that the f_{MAX} in Resource Utilization and Performance, and Example Design Implementation tables were based on default parameter settings. Updated design example target board to production 钛金系列 Ti60 F225 Development Board and updated Resource Utilization and Performance, and Example Design Implementation tables. (DOC-553)
June 2021	3.0	Added note about including all **.v** generated in testbench folder is required for simulation. Updated resource utilization and performance table. Updated example design output and implementation table. Added support for 钛金系列 FPGAs and example design for 钛金系列 Ti60 F225 Development Board. Updated for Efinity v2021.1.
December 2020	2.0	Updated user guide for 易灵思® IP Manager which includes added IP Manager topics, updated parameters, and user guide structure.
May 2020	1.0	Initial release.