Pipelined Divider

December 30, 1998

Product Specification

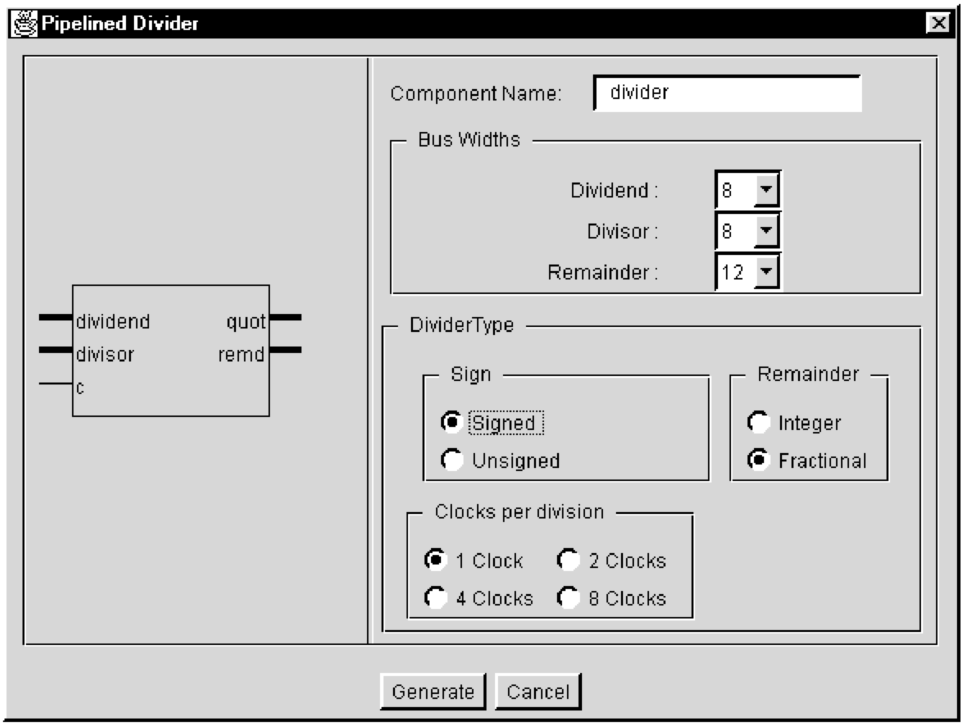
Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
E-mail: coregen@xilinx.com
URL: www.xilinx.com

Features

- Drop-in modules for the XC4000E, EX, XL, XV and Spartan families
- Divides Dividend by Divisor to give the quotient with integer or fractional remainder
- Pipelined architecture for increased throughput
- Pipeline reduction for size versus speed selections
- The Dividend value can range from 1 to 24 bits
- The Divisor value can range from 3 to 24 bits
- The Remainder value in fractional mode can range from 3 to 24 bits
- Independent Dividend, Divisor and Fractional Remainder bit widths
- Supports unsigned and 2's complement signed numbers
- Can implement 1/X function
- Fully registered inputs and outputs
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology
- Available in Xilinx CORE Generator Tool

Functional Description

This parameterized module divides an M-bit wide variable dividend by an N-bit wide variable divisor. The output consists of the quotient and either remainder or the fractional



Component Name: divider

Bus Widths
Dividend : 8
Divisor : 8
Remainder : 12

DividerType
Sign: [✓ Signed] Unsigned
Remainder: Integer [✓ Fractional]
Clocks per division: [✓ 1 Clock] 2 Clocks 4 Clocks 8 Clocks

Figure 1: Parameterization Window

result. In remainder mode, the result of the division is an M-bit wide quotient with an N-bit wide integer remainder, see Equation 1. In fractional mode, the result is an M-bit wide quotient with an F-bit wide fractional remainder, see Equation 2. It is an efficient, high speed, parallel implementation. The input data can be unsigned or signed.

$$Dividend = Quotient * Divisor + IntRmd$$

Equation 1 Dividend = quotient * divisor plus integer remainder.

$$FractRmd = \frac{IntRmd * 2^F}{Divisor}$$

Equation 2 F-bit wide Fractional Remainder

For signed mode with integer remainder, it should be noted that the sign of the quotient and remainder correspond exactly to Equation 1.

Thus

6/-4 = -1 REMD 2

whereas

-6/4 = -1 REMD –2

For signed mode with fractional remainder, the sign bit is present both in the quotient and the remainder. For example, for a four bit dividend, divisor and fractional remainder we have:

-9/4 = 9/-4 = -(2 1/4)

This corresponds to:

(1)0111 / 0100 or 1001/1100

Giving the result:

Quotient = 1110 (= -2)
Remainder = 1110 (= -1/4)

For an unsigned divider, division by zero always results in a quotient of 1's.

For example, for a 6 bit dividend:

10/0 Quotient = 011111 = 31

For the signed case the result is the maximum positive value or minus the maximum positive value depending on the sign of the dividend.

For example, for a 6 bit dividend:

6/0 Quotient = 011111 = 31
-6/0 Quotient = 100001 = -31

The remainder for division by zero for the integer remainder case is always equal to the dividend.

The design is highly pipelined. The amount of pipelining can be reduced to decrease the area of the design at the expense of throughput. In the fully-pipelined mode the design supports one division per clock cycle after an initial latency. The design also supports the options of 2,4 and 8 clock cycles per division after an initial latency, see Table 2.

The dividend and divisor bit widths can be set independently. The bit width of the quotient is equal to the bit width of the dividend. For fractional output, the remainder bit width is also independent of the dividend and divisor. The core will handle data ranges of 3 to 24 bits for the dividend, divisor and fractional output.

The divider can be used to implement the 1/X function, i.e. the reciprocal of the variable X. To do this, the dividend bit width is set to 1 for unsigned or 2 for signed data and fractional mode is selected. The dividend input is tied high within the user's design.

The divider core provides one of the basic math building blocks often encountered in general purpose processing and DSP. This core is targeted at those applications where the divisor is known to be a true variable changing every input sample. The high speed implementation of the divider will ease the design process, both in the DSP and microprocessor environment.

Pinout

The schematic symbol with signal names are shown in Figure 2 and described in Table 1.

CORE Generator Parameters

The CORE Generator parameterization window for this macro is shown in Figure 1. The parameters are as follows:

Component Name: Enter a name for the component.

Dividend: Select an input bit width from the pull-down menu for the variable width. The valid range is 1 to 24.

Divisor: Select an input bit width from the pull-down menu for the variable width. The valid range is 3 to 24.

Remainder: Select from integer remainder or fractional remainder.

Sign: Set the sign of the input and output data to signed or unsigned.

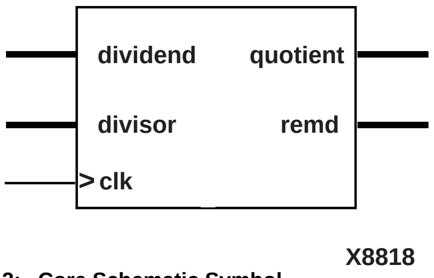


Figure 2: Core Schematic Symbol

Fractional: Select an output bit width from the pull-down menu for the variable width. The valid range is 1 to 24. This option can only be modified if the Fractional check box (see below) is selected.

Clocks per Division: Select the number of clock cycles per division.

Latency

The total latency (number of clocks required to get the first output) is a function of the bit width of the dividend. If fractional output is required the latency is also a function of the fractional bit width.

In general:

Latency = m for integer remainder dividers

Latency = m + f for fractional remainder dividers

Table 2 gives a list of the latency for some divider selections.

Table 1: Core Signal Pinout

Signal	Signal Direction	Description
Dividend[m:0]	Input	Dividend –Parallel Data In
Divisor[n:0]	Input	Divisor–Parallel Data In
clk	Input	CLOCK – with the exception of asynchronous control inputs (where applicable), control and data inputs are captured, and new output data formed on rising clock transitions.
Quotient[m:0]	Output	Quotient-Parallel data out
Remd[n:0] Remd[f:0]	Output	Remainder-Integer data bit width N Fractional data bit width F Parallel data out

Core Resource Utilization

Table 2 shows the resource utilization for some available bit widths.

Ordering Information

This macro comes free with the Xilinx CORE Generator. For additional information, contact your local Xilinx sales representative or e-mail requests to coregen@xilinx.com.

Table 2: Latency based on Divider Parameters

Signed	Fractional	Clks/div	Latency
False	False	1	M+3
False	False	>1	M+4
False	True	1	M+4
False	True	>1	M+DC+4
True	False	1	M+F+3
True	False	>1	M+F+4
True	True	1	M+F+3
True	True	>1	M+F+DC+3

M=dividend width, F=remainder width, DC=no. of clocks per division.

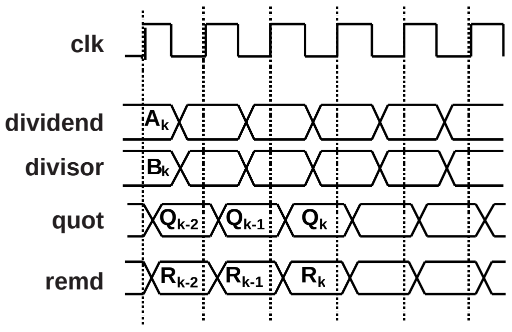


Figure 3: Two Clock Cycle Latency on Output Data

Parameter File Information

Parameter	Type	Notes
Component_Name	String	
Dividend_Width	Integer	3-24 Integer Remainder, 1-24 Fractional Remainder
Divisor_Width	Integer	3-24
Fractional_Width	Integer	3-24
Signed_B	Boolean	True/False
Fractional_B	Boolean	True/False
Divclk_Sel	Integer	1,2,4 or 8