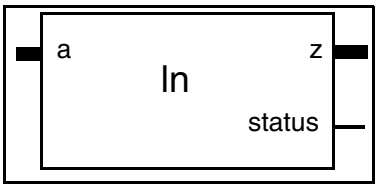
DW_fp_ln

Floating-Point Natural Logarithm

Version, STAR and Download Information: IP Directory

Features and Benefits

- The precision is controlled by parameters, and covers formats in the IEEE Standard 754
- Exponents can range from 3 to 31 bits
- Fractional part of the floating-point number can range from 2 to 60 bits
- A parameter controls the use of denormal values



Revision History

Description

This component computes the natural logarithm of a floating-point input `a`, delivering an output $z = ln(a)$ that is also a floating-point value. All the parameters available in this component are listed in Table 1-2 on page 2.

The parameter *ieee_compliance* controls the use of denormals and NaNs, as done for other FP operators in this library (see datasheet of DW_fp_log2). When *ieee_compliance* = 0, the operator takes NaN values as infinities, and denormals as zeros. When *ieee_compliance* = 1, the component accepts and generates denormalized values, handles NaN inputs, and delivers NaN outputs when appropriate.

Using *sig_width* (significand size) and *exp_width* (exponent field size) the floating-point format used for input and output operands can be adjusted to match some of the formats defined in the IEEE Standard 754.

The output `status` is an 8-bit value that carries the status flags for the FP operation, as described in the *Datapath Floating-Point Overview*.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*sig_width* + *exp_width* + 1 bits	Input	Input data
z	*sig_width* + *exp_width* + 1 bits	Output	Logarithm value ln(a)
status	8 bits	Output	■ See STATUS Flags in the *Datapath Floating-Point Overview* ■ status[7]: Divide-by-zero flag for logarithm functions

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	2 to 59 bits	Word length of fraction field of floating-point numbers `a` and `z`
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers `a` and `z`
ieee_compliance	0 or 1	Controls the use of denormals and NaNs: ■ 0: Do not use denormals or NaNs ■ 1: Use denormals and NaNs
extra_prec	0 to (59 - *sig_width*) Default: 0	Internal extra precision (in bits) used in the computation of the FP output.
arch	0 or 1 Default: 0	Implementation selection: ■ 0: Area optimized ■ 1: Speed optimized

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Implementation using the Datapath Generator technology combined with static DesignWare components	DesignWare

Table 1-4 Simulation Model

Model	Function
DW04.DW_FP_LN_CFG_SIM	Design unit name for VHDL simulation
dw/dw04/src/DW_fp_ln_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fp_ln.v	Verilog simulation model source code

The DW_fp_ln does not include a rounding mode control, given the properties of algorithms used to compute the logarithms and the goal to deliver the best possible QoR.

The component can deliver outputs with 1 ulp error for most of the FP range. The special case when the input is in the vicinity of 1.0, and therefore the output approaches 0.0, requires extra hardware just to deal with it. For this special case, consider setting the parameter *extra_prec* to minimize the extra hardware.

When *extra_prec* = 0, the implementation uses the minimum number of bits required to guarantee an error of 1 ulp for most of the calculated values. When the input value is near 1.0, the output may exhibit larger relative errors. By using non-zero values for *extra_prec,* you get smaller errors when the input is near 1, but the error bound for other input values does not improve (continues to be 1 ulp). The use of *extra_prec* affects QoR, so you should experiment with it to reach a good compromise between accuracy around the input value 1.0 and QoR. Consider using values of *extra_prec* in the range [0, *sig_width*]. When

extra_prec = *sig_width*, the error is bounded to 1 ulp for any input value. Values of *extra_prec* larger than *sig_width* do not improve accuracy.

The *arch* parameter controls implementation alternatives for this component. Different values result in different numerical behavior. You should experiment with this parameter to find out which value provides the best QoR for your design constraints and technology. Using *arch* = 0 (area optimized implementation) usually provides the best QoR for most time constraints.

For more information on the floating-point system defined for all the floating-point components in the DesignWare Library, including status bits and floating-point formats, refer to the *Datapath Floating-Point Overview*.

Related Topics

- Datapath Floating-Point Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_fp_ln_inst is
   generic (
       inst_sig_width : POSITIVE := 10;
       inst_exp_width : POSITIVE := 5;
       inst_ieee_compliance : INTEGER := 0;
       inst_extra_prec : INTEGER := 0;
       inst_arch : INTEGER := 0
            );
   port (
       inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
       z_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
       status_inst : out std_logic_vector(7 downto 0)
       );
   end DW_fp_ln_inst;

architecture inst of DW_fp_ln_inst is

begin

    -- Instance of DW_fp_ln
    U1 : DW_fp_ln
        generic map (
                sig_width => inst_sig_width,
                exp_width => inst_exp_width,
                ieee_compliance => inst_ieee_compliance,
                extra_prec => inst_extra_prec,
                arch => inst_arch
                )
        port map (
                a => inst_a,
                z => z_inst,
                status => status_inst
                );

end inst;

-- pragma translate_off
configuration DW_fp_ln_cfg_inst of DW_fp_ln_inst is
 for inst
 end for; -- inst
end DW_fp_ln_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_ln_inst( inst_a, z_inst, status_inst );

parameter inst_sig_width = 10;
parameter inst_exp_width = 5;
parameter inst_ieee_compliance = 0;
parameter inst_extra_prec = 0;
parameter inst_arch = 0;


input [inst_sig_width+inst_exp_width : 0] inst_a;
output [inst_sig_width+inst_exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_ln
    DW_fp_ln #(inst_sig_width, inst_exp_width, inst_ieee_compliance, inst_extra_prec,
inst_arch) U1 (
                        .a(inst_a),
                        .z(z_inst),
                        .status(status_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
April 2018	N-2017.09-SP5	■ For STAR 9001308455, corrected width of the extra_prec port ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043

www.synopsys.com