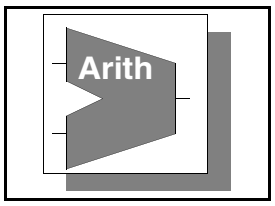
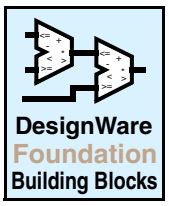




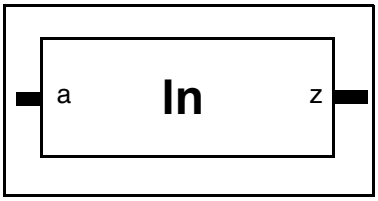
DW_ln

Natural Logarithm: ln(a)

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word width
- Supports up to 60 bits of precision



Description

DW_ln computes the natural logarithm (*ln*) of an input a in fixed-point format. The input must be in the range [1,2) (normalized input) and therefore the output is in the range [0, *ln*(2)).

The number of bits used as input and output is defined by a parameter (*op_width*). The input has 1 integer bit and *op_width-1* fractional bits. The output has *op_width* fractional bits.

The component implements the logarithm function using different algorithms depending on the number of input bits. The selection of the algorithm is done automatically to deliver the best QoR. The user can control the type of architecture to be implemented by using the parameter *arch*, as shown in Table 1-2. This parameter is effective only for values of *op_width* in the range [19,39].

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*op_width* bits	Input	Input data in the range [1,2)
z	*op_width* bits	Output	ln(a) in the range [0,ln(2))

Table 1-2 Parameter Description

Parameter	Values	Description
op_width	2 to 60 bits	Word length of a and z
arch	0 or 1 Default: 0	Implementation selection 0 - area optimized 1 - speed optimized
err_range	1 or 2 Default: 1	Error range of the result compared to the infinitely precise result 1 - 1 ulp 2 - 2 ulps

Table 1-3 Synthesis Implementations

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

Table 1-4 Simulation Model

Model	Function
DW02.DW_LN_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_ln_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_ln.v	Verilog simulation model source code

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

Another parameter, *err_range*, can be used to relax the error boundaries and get an implementation with slightly better QoR. The error does not exceed 2 ulps when *err_range* = *2*, and does not exceed 1 ulp when *err_range* = *1*. The error of 1 ulp corresponds to 2^{-op_width}.

Table 5 shows valid values for the function implemented by this component when *op_width* = *4* and *err_range* = *1* (it may not correspond to the actual component output). Since the infinite precision value of *ln(a)* is usually in between two finite precision results with *op_width* fractional bits, there are usually two valid outputs for any value of the input when *err_range* = 1 (does not apply for *a* = *1*). When *err_range* = *2*, the number of valid alternatives for the output increases to four. Also, notice that the MS bit of the input vector is expected to be 1, when it is not, the component output carries a meaningless value. The error is always less than the weight of the LS bit position of the output, in the example, less than 2^4, which corresponds to 1 ulp when *op_width* = *4*.

Table 1-5 Valid Values of ln(a) (op_width = 4, err_range=1)[a]

a(3:0)	value of 'a' (decimal)	ln(a)	z(3:0)	value of 'z' (decimal)	ln(a) (8 bits)
1.000	1.0	0.0	.0000	0.0	.00000000
1.001	1.125	0.118	.0001 or .0010	0.063 or 1.125	.00011110
1.010	1.25	0.223	.0011 or .0100	0.118 or 0.25	.00111000
1.011	1.375	0.318	.0101 or .0110	0.313 or 0.375	.01010001
1.100	1.5	0.405	.0110 or .0111	0.375 or 0.438	.01100111
1.101	1.625	0.486	.0111 or .1000	0.438 or 0.5	.01111100
1.110	1.75	0.560	.1000 or .1001	0.5 or 0.563	.10001111

Table 1-5 Valid Values of ln(a) (op_width = 4, err_range=1)[a] (Continued)

a(3:0)	value of 'a' (decimal)	ln(a)	z(3:0)	value of 'z' (decimal)	ln(a) (8 bits)
1.111	1.875	0.629	.1010 or .1011	0.625 or 0.688	.10100000

a. The values shown in the table may not correspond to the actual outputs generated by DW_ln, which may use another valid value within the error bounds defined for the implementation.

The computation of the logarithm for other ranges of the input operand can be accomplished using mathematical transformations. For example, to compute the ln of a value x in the range (0,2) the designer may use the following identity:

$$\ln(x) = \ln\left|\frac{x2^n}{2^n}\right| = \ln\left|\frac{y}{2^n}\right| = \ln(y) - n\ln(2)$$

where n corresponds to the number of zeros in MS bit positions. The value $y=x2^n$ may be obtained using a normalization unit (DW_norm).

Related Topics

- Application Specific – Data Integrity 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_ln_inst is
      generic (
        inst_op_width : INTEGER := 8;
            inst_arch : INTEGER := 0;
            inst_err_range : INTEGER := 1
        );
      port (
        inst_a : in std_logic_vector(inst_op_width-1 downto 0);
        z_inst : out std_logic_vector(inst_op_width-1 downto 0)
        );
    end DW_ln_inst;


architecture inst of DW_ln_inst is

begin

    -- Instance of DW_ln
    U1 : DW_ln
    generic map (
          op_width => inst_op_width,
                arch => inst_arch,
                err_range => inst_err_range
          )
    port map (
          a => inst_a,
          z => z_inst
          );


end inst;

-- pragma translate_off
configuration DW_ln_inst_cfg_inst of DW_ln_inst is
for inst
end for; -- inst
end DW_ln_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_ln_inst( inst_a, z_inst );

parameter inst_op_width = 8;
parameter inst_arch = 1;
parameter inst_err_range = 1;

input [inst_op_width-1 : 0] inst_a;
output [inst_op_width-1 : 0] z_inst;

    // Instance of DW_ln
    DW_ln #(inst_op_width, inst_arch, inst_err_range) U1 (
                .a(inst_a),
                .z(z_inst) );

endmodule
```

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