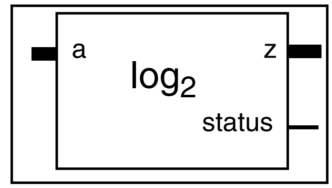
DW_fp_log2

Floating-Point Base-2 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

Description

This component computes the base-2 logarithm of a floating-point input `a`, delivering an output `z` that is also a floating-point value.

The parameter *ieee_compliance* controls the use of denormals and NaNs, as done for other FP operators in this library. 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. Independently of the value of this parameter, the floating-point format can be adjusted to match one 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	$\log_2$(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	Width of input and output data buses

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
ieee_compliance	0 or 1	When 1 the generated architecture is capable of dealing with denormals and NaNs.
extra_prec	0 to (60-sig_width) Default: 0	Internal extra precision (in bits) used in the computation of the FP output.
arch	0 to 2 Default: 2	Implementation selection. 0: Area optimized 1: Speed optimized 2: Implementation released in 2007.12

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_LOG2_CFG_SIM	Design unit name for VHDL simulation
dw/dw04/src/DW_fp_log2_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fp_log2.v	Verilog simulation model source code

Differently from other FP components, DW_fp_log2 does not have rounding mode control, given the properties of algorithms used to compute the logarithms and the goal to deliver the best possible QoR.

The component is able to 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 reason a new parameter (*extra_prec*) was introduced. When this parameter is 0, the implementation uses the minimum number of bits required to guarantee an error of 1ulp 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* the designer get smaller errors when the input is near 1, but the error bound for any other input values does not improve (continues to be 1 ulp). The use of *extra_prec* affects QoR and the designer must experiment with it to reach a good compromise between accuracy around the input value 1.0 and QoR. The designer should consider using values of *extra_prec* in the range [0,*sig_width*]. When *extra_prec* = *sig_width*, the error is bounded to 1 ulp. Values larger than *sig_width* do not improve accuracy in any way.

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.

Alternative Implementation of Floating-point Base-2 Logarithm with DW_lp_fp_multifunc

The floating-point base-2 logarithm operation can also be implemented by DW_lp_fp_multifunc component (a member of the minPower Library, licensed separately), which evaluates the value of floating-point base-2 logarithm with 1 ulp error bound. There will be 1 ulp difference between the value from DW_lp_fp_multifunc and the value from DW_fp_log2. Performance and area of the synthesis results are different between the DW_fp_log2 and base-2 logarithm implementation of the DW_lp_fp_multifunc, depending on synthesis constraints, library cells and synthesis environments. By comparing performance and area between the base-2 logarithm implementation of DW_lp_fp_multifunc and DW_fp_log2 component, the DW_lp_fp_multifunc provides more choices for the better synthesis results. Below is an example of the Verilog description for the floating-point base-2 logarithm of the DW_lp_fp_multifunc. For more detailed information, see the DW_lp_fp_multifunc datasheet.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 32) U1 (
     .A(A),
     .FUNC(16'h0020),
     .RND(3'h0),
     .Z(Z),
     .STATUS(STATUS)
);
```

For more information on the floating-point system defined for all the floating-point components in the DesignWare Library, including status flag 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.dw_foundation_comp.all;

entity DW_fp_log2_inst is
      generic (
         inst_sig_width : POSITIVE := 23;
         inst_exp_width : POSITIVE := 8;
         inst_ieee_compliance : INTEGER := 0;
         inst_extra_prec : INTEGER := 0;
         inst_arch : INTEGER := 2
        );
      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_log2_inst;


architecture inst of DW_fp_log2_inst is
begin

    -- Instance of DW_fp_log2
    U1 : DW_fp_log2
    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_log2_inst_cfg_inst of DW_fp_log2_inst is
 for inst
 end for; -- inst
end DW_fp_log2_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_log2_inst( inst_a, z_inst, status_inst );

parameter sig_width = 10;
parameter exp_width = 5;
parameter ieee_compliance = 0;
parameter extra_prec = 0;
parameter arch = 2;

input [sig_width+exp_width : 0] inst_a;
output [sig_width+exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_log2
    DW_fp_log2 #(sig_width, exp_width, ieee_compliance, extra_prec, arch)
      U1 ( .a(inst_a), .z(z_inst), .status(status_inst) );

endmodule
```

Copyright Notice and Proprietary Information

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