

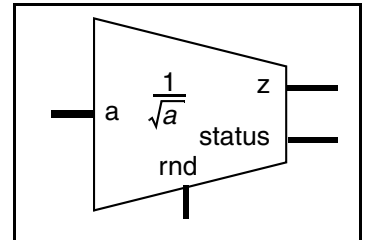
DW_fp_invsqrt

Floating-Point Reciprocal of Square Root

Version, STAR and Download Information: [IP Directory](#)

Features and Benefits

- The floating-point format is controlled by parameters, and covers formats in the IEEE standard
- Exponents can range from 3 to 31 bits
- Fractional part of the floating-point number can range from 2 to 253 bits
- A parameter controls the use of denormal values
- Provides a variety of rounding modes
- Accuracy conforms to IEEE 754 Floating-point standard¹
- DesignWare datapath generator is employed for better timing and area



Description

DW_fp_invsqrt is a component that works with floating-point values to compute the reciprocal of the square-root of a floating-point input *a*, to produce a floating-point result $1/\sqrt{a}$. The output of this component has accuracy consistent with the IEEE Standard 754.

The input *rnd* defines a 3-bit rounding mode value (see [Rounding Modes](#) in the *Datapath Floating-Point Overview*).

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	<i>sig_width</i> + <i>exp_width</i> + 1 bits	Input	FP Input data
rnd	3 bits	Input	Rounding mode
z	<i>sig_width</i> + <i>exp_width</i> + 1 bits	Output	FP output data
status	8 bits	Output	See STATUS Flags in the <i>Datapath Floating-Point Overview</i>

Table 1-2 Parameter Description

Parameter	Values	Description
<i>sig_width</i>	2 to 253 bits	Word length of fraction field of floating-point numbers <i>a</i> and <i>z</i>
<i>exp_width</i>	3 to 31 bits	Word length of biased exponent of floating-point numbers <i>a</i> and <i>z</i>

1. For more information, see [IEEE 754 Compatibility](#) in the *Datapath Floating-point Overview*.

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
ieee_compliance	0 or 1	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_FP_INVSQRT_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_fp_invsqrt_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fp_invsqrt.v	Verilog simulation model source code

Table 1-5 Functional Description

A	status ^a	z ^b
a (FP)	*	$z = 1/\sqrt{a}$

a. The details of status flags, if used, can be found in Table 9 of the Datapath - Floating-Point Overview document

b. The actual output value is defined by the rounding mode

The parameter `ieee_compliance` controls the functionality of this component. When the parameter is set to 0, the component considers denormal values as zeros and NaNs as infinity. When the `ieee_compliance` parameter is set to 1, the component is fully compliant with the IEEE 754 standard and therefore operates with denormals and NaNs.

Alternative Implementation of Floating-point Reciprocal Square Root with DW_lp_fp_multifunc

The floating-point reciprocal square root 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 reciprocal square root 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_invsqrt`. Performance and area of the synthesis results are different between the `DW_fp_invsqrt` and reciprocal square root implementation of the `DW_lp_fp_multifunc`, depending on synthesis constraints, library cells and synthesis environments. By comparing performance and area between the reciprocal square root implementation of `DW_lp_fp_multifunc` and `DW_fp_invsqrt` 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 reciprocal square root 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, 4) U1 (  
    .A(A),  
    .FUNC(16'h0004),  
    .RND(3'h0),  
    .Z(Z),  
    .STATUS(STATUS)  
);
```

For more information on the floating-point system defined for all the DesignWare Floating-point components, including `status` flag bits, floating-point formats and compatibility with IEEE standard, 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_invsqrt_inst is
    generic (
        inst_sig_width : POSITIVE := 23;
        inst_exp_width  : POSITIVE := 8;
        inst_ieee_compliance : INTEGER := 0
    );
    port (
        inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        inst_rnd : in std_logic_vector(2 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_invsqrt_inst;

architecture inst of DW_fp_invsqrt_inst is

begin

    -- Instance of DW_fp_invsqrt
    U1 : DW_fp_invsqrt
        generic map ( sig_width => inst_sig_width, exp_width => inst_exp_width,
            ieee_compliance => inst_ieee_compliance )
        port map ( a => inst_a, rnd => inst_rnd, z => z_inst, status => status_inst );

end inst;

-- pragma translate_off
configuration DW_fp_invsqrt_cfg_inst of DW_fp_invsqrt_inst is
for inst
end for; -- inst
end DW_fp_invsqrt_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_invsqrt_inst( inst_a, inst_rnd, z_inst, status_inst );

parameter sig_width = 23;
parameter exp_width = 8;
parameter ieee_compliance = 0;

input [sig_width+exp_width : 0] inst_a;
input [2 : 0] inst_rnd;
output [sig_width+exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_invsqrt
    DW_fp_invsqrt #(sig_width, exp_width, ieee_compliance)
        U1 ( .a(inst_a), .rnd(inst_rnd), .z(z_inst), .status(status_inst) );

endmodule
```

Copyright Notice and Proprietary Information

© 2018 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

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