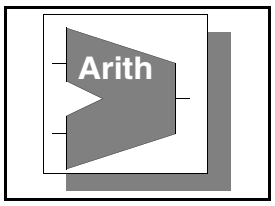
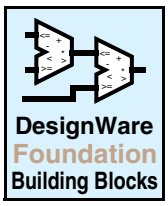




DW_exp2

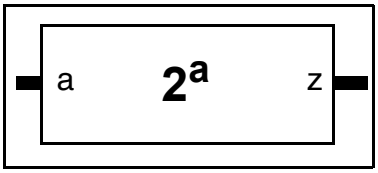
Base 2 Exponential (2^a)

Version, STAR and Download Information: IP Directory

Features and Benefits

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

Revision History



Description

The DW_exp2 component takes an input a in fixed-point format and computes the function 2^a. The input must be in the range (0,1) and therefore the output is in the range (1,2) (normalized).

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

The component implements the exponential function based on different algorithms for different ranges of precision. The selection of the algorithm is done automatically to deliver the best QoR.

Table 1-1 Pin Description

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

Table 1-2 Parameter Description

Parameter	Values	Description
op_width	2 to 60[a] Default: 8	Word length of a and z
arch	0 or 2 Default: 2	Implementation selection 0: Area optimized 1: Speed optimized 2: 2007.12 implementation
err_range	1 or 2 Default: 1	Error range of the result compared to the infinitely precise result 1: 1 ulp 2: 2 ulps Note: error range is 1 ulp when *arch* = 2

a. The synthesis model fully supports this range, as does the Verilog simulation model in VCS, but the VHDL simulation model (in all simulators) and the Verilog simulation model in non-VCS simulators are limited to a range of 2 - 38.

Table 1-3 Synthesis Implementations

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

Table 1-4 Simulation Model

Model	Function
DW02.DW_EXP2_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_exp2_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_exp2.v	Verilog simulation model source code

The truth table for the function implemented by this component when *op_width* = 4, *arch* = 2 and *err_range* = 1 is shown in Table 1-5. The values provided for eight bits of precision were truncated from a longer-precision value. The MS bit of the output vector is always 1. Also, observe that the error is always less than the weight of the LS bit position of the output; in the example, less than 2^{-3}.

Table 1-5 Truth Table (*op_width* = 4, *arch* = 2, *err_range* = 1)

a(3:0)	a value (decimal)	z(3:0)	z value (decimal)	2^a (8 bits)
.0000	0.0	1.000	1.000	1.0000000
.0001	0.063	1.000	1.000	1.0000101
.0010	0.125	1.000	1.000	1.0001011
.0011	0.188	1.001	1.125	1.0010001
.0100	0.250	1.001	1.125	1.0011000
.0101	0.313	1.001	1.125	1.0011110
.0110	0.375	1.010	1.250	1.0100101
.0111	0.438	1.010	1.250	1.0101101
.1000	0.500	1.011	1.375	1.0110101
.1001	0.563	1.011	1.375	1.0111101
.1010	0.625	1.100	1.500	1.1000101
.1011	0.688	1.100	1.500	1.1001110
.1100	0.750	1.101	1.325	1.1010111
.1101	0.813	1.110	1.750	1.1100000

Table 1-5 Truth Table (*op_width* = 4, *arch* = 2, *err_range* = 1) (Continued)

a(3:0)	a value (decimal)	z(3:0)	z value (decimal)	2^a (8 bits)
.1110	0.875	1.110	1.750	1.1101010
.1111	0.938	1.111	1.875	1.1110101

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

The *arch* parameter controls implementation alternatives for this component. Different values result in different numerical behavior. You should experiment with the *arch* 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.

To compute the exponential of a negative value in two's complement, and avoid the use of a reciprocal or division operation, the designer can use a combination of simple transformations. Assume a negative input x, with $|x|<1$. We compute 2^x as:

$$2^x = \frac{2^x * 2}{2} = \frac{2^{x+1}}{2}$$

where x is in the proper range for DW_exp2. The value $a=(1+x)$ is easily obtained from x by discarding the sign bit. The output of DW_exp2(a) is then shifted to the right to obtain the final output. For example: $x = -3/8 = 1.1010$, $a=0.1010$, $2^a = 1.100$, and thus $2^x = 0.1100=0.75$. The value 2^x computed with 8 bits of precision would be 0.11000101.

Alternative implementation of base-2 exponential with DW_lp_multifunc

The base-2 exponential operation can also be implemented by the DW_lp_multifunc component (a member of the minPower Library, licensed separately), which evaluates the value of base-2 exponential with 1 ulp error bound. There will be 1 ulp difference between the value from DW_lp_multifunc and the value from DW_exp2. Performance and area of the synthesis results are different between the DW_exp2 and the base-2 exponential implementation of the DW_lp_multifunc, depending on synthesis constraints, library cells and synthesis environments. By comparing performance and area between these alternatives, the DW_lp_multifunc provides more choices for better synthesis results. Below is an example of the Verilog description for the base-2 exponential of the DW_lp_multifunc. For more, see DW_lp_multifunc.

```
DW_lp_multifunc #(op_width, 64) U1 (
    .A(A),
    .FUNC(16'h0040),
    .Z(Z),
    .STATUS(STATUS)
);
```

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_exp2_inst is
      generic (
            inst_op_width : INTEGER := 8;
            inst_arch : INTEGER := 2;
            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_exp2_inst;


architecture inst of DW_exp2_inst is

begin

    -- Instance of DW_exp2
    U1 : DW_exp2
        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_exp2_inst_cfg_inst of DW_exp2_inst is
for inst
end for; -- inst
end DW_exp2_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_exp2_inst( inst_a, z_inst );

parameter inst_op_width = 8;
parameter inst_arch = 2;
parameter inst_err_range = 1;


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

    // Instance of DW_exp2
    DW_exp2 #(inst_op_width, inst_arch, inst_err_range) U1 (
                        .a(inst_a),
                        .z(z_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
July 2018	O-2018.06-SP1	■ For STAR 9001366624, in Table 1-2 on page 1, clarified the range of *op_width* for the VHDL simulation model (in all simulators) and the Verilog simulation model for non-VCS simulators. ■ 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