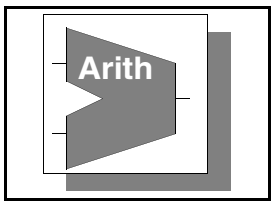
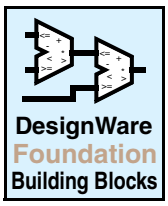




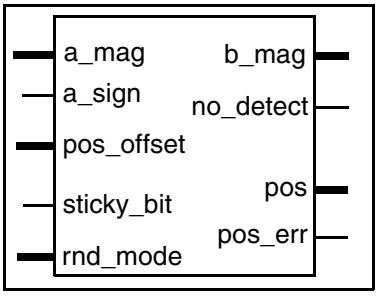
DW_norm_rnd

Normalization and rounding

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word lengths
- Parameterized search window
- Exponent calculation useful for floating-point numbers
- Implements the most used rounding modes
- Provides status indications for exceptional cases
- DesignWare datapath generator is employed for better timing and area



Description

DW_norm_rnd is a general-purpose normalization and rounding module for a value represented in the sign-and-magnitude number system (*a_sign*, *a_mag*). The value represented in this system corresponds to $(-1)^{a\_sign} a\_mag$. The magnitude of the number is a positive fixed-point value in the format $a = (a_0.a_1\ a_2\ a_3\ a_4\ a_5\ ...a_{a\_width-1})$, where a_i represents a bit. This means that *a_mag* has 1 integer bit and (*a_width*-1) fractional bits.

The normalization process consists in generating an output in the range $1 \leq b\_mag < 2$ (the output bit-vector has a 1 in the MS bit position) when there is a 1 bit in the search window provided as a parameter.

The rounding is done using a method controlled by one of the component inputs. Output *pos* carries information about the position of binary point (exponent), and it is affected by the normalization shifts performed on the main input (*a_mag*). A list of pins for this component is provided in Table 1.

The number of bit positions in *a_mag* shifted during initial normalization and post-rounding normalization (*n* bit positions) is reflected in the value of *pos*. This output corresponds to (*pos_offset* + n) when parameter *exp_ctr* = 0, or to (*pos_offset* - n) when *exp_ctr* = 1. When the result of this operation does not fit in the bit-vector used for *pos* or it is negative, the output *pos_err* is set to 1.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a_mag	*a_width* bit(s)	Input	Input data
a_sign	1 bit	Input	0 = positive 1 = negative
pos_offset	*exp_width* bit(s)	Input	Offset value for the position of the binary point

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
sticky_bit	1 bit	Input	Indicates the presence of non-zero bits in fractional bit positions beyond bit $a\_mag_{a\_width-1}$
rnd_mode	3 bits	Input	Rounding mode 000 – Round to nearest even 001 – Round towards zero 010 – Round to plus infinity 011 – Round to minus infinity 100 – Round up 101 – Round away from zero
no_detect	1 bit	Output	Result of MS 1 bit search in the search window. 0 = bit found 1 = bit not found
pos_err	1 bit	Output	Value provided at output pos cannot fit in an *exp_width* - bit vector or *pos* is negative
b_mag	*b_width* bit(s)	Output	Normalized and rounded output data
pos	*exp_width* bit(s)	Output	pos_offset combined with the number of bit positions that input `a_mag` was shifted (*n*): *exp_ctr* = 0 → pos_offset+*n* *exp_ctr* = 1 → pos_offset-*n*

Table 1-2 Parameter Description

Parameter	Values	Description
a_width	≥ 2 Default: 16	Word length of `a_mag`
srch_wind	2 to *a_width* Default: 4	Search window for the MS 1 bit (from left to right, or from bit 0 to *a_width - 1*)
exp_width	≥ 1 Default: 4	Word length of *pos_offset* and *pos*
b_width	2 to *a_width* Default: 10	Word length of *b_mag*
exp_ctr	0 or 1 Default: 0	Controls computation of the binary point position (`pos` output)

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW01.DW_NORM_RND_CFG_SIM	Design unit name for VHDL simulation
dw/dw01/src/DW_norm__rnd_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_norm_rnd.v	Verilog simulation model source code

The input *pos_offset* is provided by the user to indicate that pre-shifting of the input operand was already done to adjust it to the required input format. For example, if the actual fixed-point value is (101.0011), the input for the DW_norm_rnd component could be *a_mag* = (0.01010011) and *pos_offset* = (0100) for the parameters *a_width* = 9 and *exp_width* = 4. Observe that the designer can use other input combinations as needed.

The *srch_wind* parameter reduces the complexity of this component when it is known that the MS 1 bit resides in a limited group of MS bit positions. When there is no 1 bit in the search window, the input *a_mag* is shifted to the left by (*srch_wind* - 1) bit positions and *no_detect* is set to 1.

Once the input is normalized, a rounding method is applied to it according to the *rnd_mode* input. Rounding consists in truncating the normalized input vector (*a_norm*) to *b_width* bits and adding 1 or 0 (value called *rnd* in the following equations) to the least-significant bit position of this truncated normal vector (fractional bit $L = a\_norm_{b\_width - 1}$). In order to make a decision about the value of *rnd*, the rounding procedure makes use of two extra bits called round bit (*R*) and sticky bit (*T*). These bits are also obtained from the normalized vector *a_norm*, and for this reason the user must make sure that enough significant bits are provided to the normalization and rounding unit to allow proper rounding. The round bit corresponds to $a\_norm_{b\_width}$, when *b_width* < *a_width*; otherwise it has value 0. The sticky bit is a combination of input *sticky_bit* and any other bits beyond $a\_norm_{b\_width}$ (value after normalization). The *sticky_bit* input indicates that there are other non-zero bits beyond the fractional bit position *a_width* - 1 in the main input *a_mag*.

Once the bits *R* and *T* are determined, the value of *rnd* used for rounding is generated as:

- RNE: *rnd* = *R* and (*L* or *T*)
- Rzero: *rnd* = 0 (only truncated value)
- Rpos: *rnd* = not *a_sign* and (*R* or *T*)
- Rneg: rnd = *a_sign* and (*R* or *T*)
- Rup: *rnd* = *R*
- Raway: *rnd* = *R* + *T*

The component detects the cases that require post-rounding normalization and adjusts the *pos* output accordingly. For these cases, when *exp_ctr* = 0, the exponent value at output *pos* is decremented by 1, and when *exp_ctr* = 1, the exponent value is incremented by 1.

Related Topics

- Datapath– Arithmetic Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

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

entity DW_norm_rnd_inst is
      generic (
        inst_a_width : POSITIVE := 16;
        inst_srch_wind : POSITIVE := 4;
        inst_exp_width : POSITIVE := 4;
        inst_b_width : POSITIVE := 10
        );
      port (
        inst_a_mag : in std_logic_vector(inst_a_width-1 downto 0);
        inst_pos_offset : in std_logic_vector(inst_exp_width-1 downto 0);
        inst_sticky_bit : in std_logic;
        inst_a_sign : in std_logic;
        inst_rnd_mode : in std_logic_vector(2 downto 0);
        pos_err_inst : out std_logic;
        no_detect_inst : out std_logic;
        b_inst : out std_logic_vector(inst_b_width-1 downto 0);
        pos_inst : out std_logic_vector(inst_exp_width-1 downto 0)
        );
    end DW_norm_rnd_inst;


architecture inst of DW_norm_rnd_inst is

begin

    -- Instance of DW_norm_rnd
    U1 : DW_norm_rnd
    generic map ( a_width => inst_a_width, srch_wind => inst_srch_wind, exp_width =>
inst_exp_width, b_width => inst_b_width )
    port map ( a_mag => inst_a_mag, pos_offset => inst_pos_offset, sticky_bit =>
inst_sticky_bit, a_sign => inst_a_sign, rnd_mode => inst_rnd_mode, pos_err =>
pos_err_inst, no_detect => no_detect_inst, b => b_inst, pos => pos_inst );


end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_norm_rnd_inst( inst_a_mag, inst_pos_offset, inst_sticky_bit, inst_a_sign,
inst_rnd_mode,
          pos_err_inst, no_detect_inst, b_inst, pos_inst );

parameter a_width = 16;
parameter srch_wind = 4;
parameter exp_width = 4;
parameter b_width = 10;
parameter exp_ctr = 0;


input [a_width-1 : 0] inst_a_mag;
input [exp_width-1 : 0] inst_pos_offset;
input inst_sticky_bit;
input inst_a_sign;
input [2 : 0] inst_rnd_mode;
output pos_err_inst;
output no_detect_inst;
output [b_width-1 : 0] b_inst;
output [exp_width-1 : 0] pos_inst;

    // Instance of DW_norm_rnd
    DW_norm_rnd #(a_width, srch_wind, exp_width, b_width, exp_ctr)
      U1 ( .a_mag(inst_a_mag), .pos_offset(inst_pos_offset),
.sticky_bit(inst_sticky_bit), .a_sign(inst_a_sign), .rnd_mode(inst_rnd_mode),
.pos_err(pos_err_inst), .no_detect(no_detect_inst), .b(b_inst), .pos(pos_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