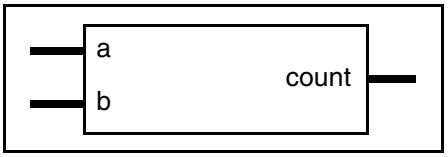
DW_lza

Leading Zeros Anticipator

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word length
- Inferable through function call



Revision History

Description

The DW_lza takes as its input the normalized and swapped unsigned values to the subtraction. The inputs are designed to have a ≥ b and of the same width, and the output will be the number of most significant zeros *anticipated* to be in the result. This anticipated value may be off by one bit less than the true required shift. The shift count generated by this component, if in error, will be one less than the actual number of shifts required by the addition.

The DW_lza is used to *anticipate* the number of bits required to shift in order to normalize the result of an addition. Conventional floating point addition requires the result to be normalized to a fixed radix point. Normalization shifts out the leading sign bits leaving a single bit adjacent to the radix point. Ordinary addition techniques wait until the final sum is obtained before determining the shift amount. In the normal floating point flow, the final determination of the shift can consume too much time, leading to a critical path. The DW_lza will do this calculation in parallel with the addition, allowing the design of higher performance adders.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*width*	Input	a input
b	*width*	Input	b input
count	ceil($\log_2$[*width*])	Output	Count of leading zeros

Table 1-2 Parameter Description

Parameter	Values	Description
width	2 ≤ *width* ≤ 256	Width of a and b inputs

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW01.DW_LZA_CFG_SIM	Design unit name for VHDL simulation
dw/dw01/src/DW_lza_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_lza.v	Verilog simulation model source code

Figure 1-1 Truth Table (width = 4, count width = 2)

a	b	count	a	b	count	a	b	count	a	b	count
0000	0000	11	0111	0110	11	1011	0010	00	1101	1011	10
0001	0000	11	0111	0111	11	1011	0011	00	1101	1100	11
0001	0001	11	1000	0000	00	1011	0100	01	1101	1101	11
0010	0000	10	1000	0001	00	1011	0101	01	1110	0000	00
0010	0001	11	1000	0010	00	1011	0110	01	1110	0001	00
0010	0010	11	1000	0011	00	1011	0111	01	1110	0010	00
0011	0000	10	1000	0100	01	1011	1000	10	1110	0011	00
0011	0001	10	1000	0101	01	1011	1001	10	1110	0100	00
0011	0010	11	1000	0110	10	1011	1010	11	1110	0101	00
0011	0011	11	1000	0111	11	1011	1011	11	1110	0110	00
0100	0000	01	1000	1000	11	1100	0000	00	1110	0111	00
0100	0001	01	1001	0000	00	1100	0001	00	1110	1000	01
0100	0010	10	1001	0001	00	1100	0010	00	1110	1001	01
0100	0011	11	1001	0010	00	1100	0011	00	1110	1010	01
0100	0100	11	1001	0011	00	1100	0100	00	1110	1011	01
0101	0000	01	1001	0100	01	1100	0101	00	1110	1100	10
0101	0001	01	1001	0101	01	1100	0110	00	1110	1101	11
0101	0010	10	1001	0110	10	1100	0111	00	1110	1110	11
0101	0011	10	1001	0111	10	1100	1000	01	1111	0000	00
0101	0100	11	1001	1000	11	1100	1001	01	1111	0001	00
0101	0101	11	1001	1001	11	1100	1010	10	1111	0010	00
0110	0000	01	1010	0000	00	1100	1011	11	1111	0011	00
0110	0001	01	1010	0001	00	1100	1100	11	1111	0100	00
0110	0010	01	1010	0010	00	1101	0000	00	1111	0101	00
0110	0011	01	1010	0011	00	1101	0001	00	1111	0110	00
0110	0100	10	1010	0100	01	1101	0010	00	1111	0111	00
0110	0101	11	1010	0101	01	1101	0011	00	1111	1000	01
0110	0110	11	1010	0110	01	1101	0100	00	1111	1001	01
0111	0000	01	1010	0111	01	1101	0101	00	1111	1010	01
0111	0001	01	1010	1000	10	1101	0110	00	1111	1011	01
0111	0010	01	1010	1001	11	1101	0111	00	1111	1100	10
0111	0011	01	1010	1010	11	1101	1000	01	1111	1101	10
0111	0100	10	1011	0000	00	1101	1001	01	1111	1110	11
0111	0101	10	1011	0001	00	1101	1010	10	1111	1111	11

Related Topics

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

HDL Usage Through Function Inferencing - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation.all;
-- If using numeric_std data types of std_logic_arith, uncomment the
-- following line:
-- use DWARE.DW_Foundation_arith.all;

entity DW_lza_func is
  generic (
    func_width : POSITIVE := 8
    );
  port (
    func_a : in std_logic_vector(func_width-1 downto 0);
    func_b : in std_logic_vector(func_width-1 downto 0);
    count_func : out std_logic_vector(bit_width(func_width)-1 downto 0)
    );
  end DW_lza_func;

architecture func of DW_lza_func is
begin
  -- Inferred function of DW_lza
  count_func <= DWF_lza(func_a,func_b);
end func;

-- pragma translate_off
configuration DW_lza_func_cfg_func of DW_lza_func is
for func
end for; -- func
end DW_lza_func_cfg_func;
-- pragma translate_on
```

HDL Usage Through Function Inferencing - Verilog

```
module DW_lza_func( func_a, func_b, count_func );

  parameter func_width = 7;

  // Passes widths to lza function with specific paramters
  parameter width       = func_width;
  parameter addr_width  = 3;    // NEEDS TO BE ceil(log2(func_width))

  // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
  // to your .synopsys_dc.setup file (for synthesis) and add
  // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
  // (for simulation).
  `include "DW_lza_function.inc"

  input [func_width-1 : 0] func_a;
  input [func_width-1 : 0] func_b;
  output [addr_width-1 : 0] count_func;

   // Function inference of DW_lz
  assign count_func = DWF_lza(func_a,func_b);

endmodule
```

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE,DW01;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DW01.DW01_components.all;

entity DW_lza_inst is
      generic (
        inst_width : NATURAL := 7
        );
      port (
        inst_a : in std_logic_vector(inst_width-1 downto 0);
        inst_b : in std_logic_vector(inst_width-1 downto 0);
        count_inst : out std_logic_vector(bit_width(inst_width)-1 downto 0)
        );
    end DW_lza_inst;


architecture inst of DW_lza_inst is

begin

    -- Instance of DW_lza
    U1 : DW_lza
    generic map ( width => inst_width )
    port map ( a => inst_a,
               b => inst_b,
           count => count_inst );


end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_lza_inst( inst_a, inst_b, count_inst );

parameter width = 7;


`define w (width)
`define bit_width_width 3

input [width-1 : 0] inst_a;
input [width-1 : 0] inst_b;
output [`bit_width_width-1 : 0] count_inst;

    // Instance of DW_lza
    DW_lza #(width)
      U1 ( .a(inst_a),
           .b(inst_b),
           .count(count_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
March 2018	N-2017.09-SP4	■ Updated example in "HDL Usage Through Function Inferencing - Verilog" on page 4 ■ 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