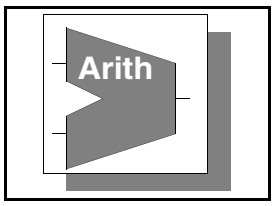
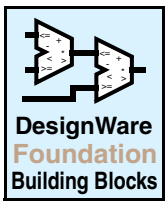




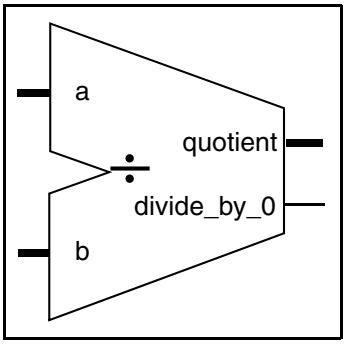
DW_div_sat

Combinational Divider with Saturation

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word lengths
- Parameterized quotient length with saturation
- Unsigned and signed (two's complement) data operation
- Inferable using a function call
- Multiple architectures for area/performance trade-off



Description

DW_div_sat is a combinational integer divider with parameterized quotient length. When the quotient length is smaller than the dividend length, the quotient is saturated in case of overflow.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*a_width* bit(s)	Input	Dividend
b	*b_width* bit(s)	Input	Divisor
quotient	*q_width* bit(s)	Output	Quotient
divide_by_0	1 bit	Output	Indicates if b equals 0

Table 1-2 Parameter Description

Parameter	Values	Description
a_width	≥ 2 Default: None	Word length of `a`
b_width	≥ 2 Default: None	Word length of `b`
q_width	2 to *a_width* Default: None	Word length of `quotient`
tc_mode	0 or 1 Default: 0	Two's- complement control 0 = unsigned 1 = signed

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
cla	Restoring carry-look-ahead divider synthesis model	DesignWare
cla2	Radix-4 restoring carry-look-ahead divider synthesis model	DesignWare
cla3	Radix-8 restoring carry-look-ahead divider synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_DIV_SAT_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_div_sat_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_div_sat.v	Verilog simulation model source code

Table 1-5 Functional Description

tc_mode	a	b	quotient
0	a (unsigned)	b (unsigned)	sat(int(a/b)) (unsigned)
1	a (two's complement)	b (two's complement)	sat(int(a/b)) (two's complement)

The parameter *tc_mode* determines whether the data of the inputs `a` and `b` and the output `quotient` are interpreted as unsigned (*tc_mode*=0) or two's-complement (*tc_mode*=1) numbers.

Regular integer division yields a quotient that has the same number of bits as the dividend (see DW_div component). DW_div_sat allows to specify a shorter `quotient` length where the upper bits are truncated and only the lower *q_width* bits are kept. In case of overflow/underflow (for example, if the division result is too large to fit in the `quotient` output) the returned `quotient` value is saturated.

The `quotient` output is defined as follows:

```
q_int[a_width-1:0] = int (a/b)
```

Unsigned (*tc_mode* = 0):

```
quotient[q_width-1:0] = 2^q_width-1                if q_int[a_width-1:0] > 2^q_width-1
                      = q_int[q_width-1:0]      else
```

Signed (*tc_mode* = 1):

```
quotient[q_width-1:0] =  2^q_width-1-1             if      q_int[a_width-1:0] >  2^q_width-1-1
                      = -2^q_width-1               else if q_int[a_width-1:0] < -2^q_width-1
                      = q_int[q_width-1:0]      else
```

Division by 0

In the case of dividing by zero, the `divide_by_0` output is set to 1 and the `quotient` is saturated to the maximum positive value if `a` is positive and to the minimum negative value if `a` is negative.

Attention	A divide-by-zero warning message is generated each time the `b` input changes to the value zero. This warning can be disabled for Verilog simulations by defining the Verilog macro, DW_SUPPRESS_WARN, either in the test bench or on the simulator command line (such as, `+define+DW_SUPPRESS_WARN+`).

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 IEEE.std_logic_arith.all;
use DWARE.DW_Foundation_arith.all;

entity DW_div_sat_func is
  generic (width : positive := 8);
  port (a            : in  std_logic_vector(2*width-1 downto 0);
        b            : in  std_logic_vector(width-1 downto 0);
        quotient_uns : out std_logic_vector(width-1 downto 0);
        quotient_tc  : out std_logic_vector(width-1 downto 0));
end DW_div_sat_func;

architecture func of DW_div_sat_func is
begin
  -- function calls for unsigned/signed quotient
  quotient_uns <= std_logic_vector(
    DWF_div_sat (unsigned(a), unsigned(b), width));
  quotient_tc  <= std_logic_vector(
    DWF_div_sat (signed(a), signed(b), width));
end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW_div_sat_func (a, b, quotient_uns, quotient_tc);

  parameter width = 8;

  input  [2*width-1 : 0] a;
  input    [width-1 : 0] b;
  output   [width-1 : 0] quotient_uns;
  output   [width-1 : 0] quotient_tc;

  // pass "a_width", "b_width", "q_width" parameters to the inference functions
  parameter a_width = 2*width;
  parameter b_width = width;
  parameter q_width = 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_div_sat_function.inc"

  // function calls for unsigned/signed quotient
  assign quotient_uns = DWF_div_sat_uns (a, b);
  assign quotient_tc  = DWF_div_sat_tc (a, b);

endmodule
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_div_sat_inst (a, b, quotient, divide_by_0);

  parameter width   = 8;
  parameter tc_mode = 0;

  input  [2*width-1 : 0] a;
  input    [width-1 : 0] b;
  output   [width-1 : 0] quotient;
  output                 divide_by_0;

  // Please add +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator
  // command line (for simulation).

  // instance of DW_div_sat
  DW_div_sat #(2*width, width, width, tc_mode)
    U1 (.a(a), .b(b),
        .quotient(quotient), .divide_by_0(divide_by_0));
endmodule
```

HDL Usage Through Component Instantiation - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_div_sat_inst is

  generic (width   : positive := 8;
           tc_mode : natural  := 0);
  port (a           : in  std_logic_vector(2*width-1 downto 0);
        b           : in  std_logic_vector(width-1 downto 0);
        quotient    : out std_logic_vector(width-1 downto 0);
        divide_by_0 : out std_logic);
end DW_div_sat_inst;

architecture inst of DW_div_sat_inst is
begin
  -- instance of DW_div_sat
  U1 : DW_div_sat
    generic map (a_width => 2*width, b_width => width,
                 q_width => width, tc_mode => tc_mode)
    port map (a => a, b => b,
              quotient => quotient, divide_by_0 => divide_by_0);
end inst;

-- pragma translate_off
configuration DW_div_sat_inst_cfg_inst of DW_div_sat_inst is
  for inst
  end for;
end DW_div_sat_inst_cfg_inst;
-- pragma translate_on
```

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