

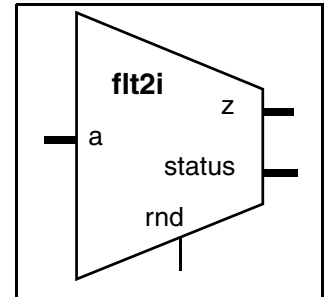
DW_fp_ft2i

Floating-Point to Integer Converter

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

Features and Benefits

- The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- Accuracy conforms to IEEE 754 Floating-point standard¹
- DesignWare datapath generator is employed for better timing and area



Description

DW_fp_ft2i is a floating-point to integer converter that takes a floating-point number, *a*, to produce an integer number, *z*. The output *z* is always a signed two's complement integer. The input *rnd* is 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	Floating-point number
rnd	3 bits	Input	Rounding mode
z	<i>isize</i> bits	Output	Two's complement integer number
status	8 bits	Output	See STATUS Flags in the <i>Datapath Floating-Point Overview</i>

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	2 to 253 Default: 23	Word length of fraction field of floating-point number <i>a</i>
exp_width	3 to 31 Default: 8	Word length of biased exponent of floating-point number <i>a</i>
isize	3 to 512 Default: 32	Word length of converted integer number <i>z</i>
ieee_compliance	0 or 1 Default: 0	When 1, it recognizes NaN, Inf and Denormal inputs.

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

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Fast Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_FP_FLT2I_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_fpflt2i_sim.vhd	VHDL Simulation Model Source Code
dw/dw02/sim_ver/DW_fpflt2i.v	Verilog Simulation Model Source Code

Table 1-5 Function Example 1 (*sig_width* = 10, *exp_width* = 5, *isize* = 16, *ieee_compliance* = 0)

Description	a (FP format)	rnd	status	z (Integer Number)
Zero	0000_0000_0000_0000	any	0000_0001	0000_0000_0000_0000
Denormal	0000_0000_0000_0001	any	0000_0001	0000_0000_0000_0000
Infinity	0111_1100_0000_0000	any	0110_0000	0111_1111_1111_1111
	1111_1100_0000_0000	any	0110_0000	1000_0000_0000_0000
NaN	0111_1100_0000_0001	any	0110_0000	0111_1111_1111_1111
	1111_1100_0000_0001	any	0110_0000	1000_0000_0000_0000
Normal Number	0110_0011_1111_1111	0	0010_0000	0000_0100_0000_0000
	0110_0011_1111_1111	1	0010_0000	0000_0011_1111_1111
	0110_0011_1111_1111	2	0010_0000	0000_0100_0000_0000
	0110_0011_1111_1111	3	0010_0000	0000_0011_1111_1111
	0110_0011_1111_1111	4	0010_0000	0000_0100_0000_0000
	0110_0011_1111_1111	5	0010_0000	0000_0100_0000_0000
	1110_0011_1111_1111	0	0010_0000	1111_1100_0000_0000

Table 1-6 Function Example 2 (*sig_width* = 10, *exp_width* = 5, *isize* = 16, *ieee_compliance* = 1)^a

Description	a (FP Format)	rnd	status	z (Integer Number)
Zero	0000_0000_0000_0000	any	0000_0001	0000_0000_0000_0000

Table 1-6 Function Example 2 (*sig_width* = 10, *exp_width* = 5, *isize* = 16, *ieee_compliance* = 1)^a (Continued)

Description	a (FP Format)	rnd	status	z (Integer Number)
Denormal	0000_0000_0000_0001	0, 1, 3, 4	0010_1001	0000_0000_0000_0000
	0000_0000_0000_0001	2, 5	0010_0000	0000_0000_0000_0001
	1000_0000_0000_0001	0, 1, 2, 4	0010_1001	0000_0000_0000_0000
	1000_0000_0000_0001	3, 5	0010_0000	1111_1111_1111_1111
Infinity	0111_1100_0000_0000	any	0110_0000	0111_1111_1111_1111
	1111_1100_0000_0000	any	0110_0000	1000_0000_0000_0000
NaN	0111_1100_0000_0001	any	0110_0000	0111_1111_1111_1111
	1111_1100_0000_0001	any	0110_0000	1000_0000_0000_0000
Normal Number	0110_0011_1111_1111	0	0010_0000	0000_0100_0000_0000
	0110_0011_1111_1111	1	0010_0000	0000_0011_1111_1111
	0110_0011_1111_1111	2	0010_0000	0000_0100_0000_0000
	0110_0011_1111_1111	3	0010_0000	0110_0011_1111_1111
	0110_0011_1111_1111	4	0010_0000	0000_0100_0000_0000
	0110_0011_1111_1111	5	0010_0000	0110_0011_1111_1111
	1110_0011_1111_1111	0	0010_0000	1111_1100_0000_0000

a. Although the output is an integer number, the tiny bit is set when denormal input is rounded to 0.

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;
-- If using numeric types from std_logic_arith package,
-- comment the preceding line and uncomment the following line:
-- use DWARE.DW_Foundation_comp_arith.all;

entity DW_fp_flt2i_inst is
  generic (
    inst_sig_width : POSITIVE := 23;
    inst_exp_width : POSITIVE := 8;
    inst_isize      : INTEGER := 32
  );
  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_isize-1 downto 0);
    status_inst  : out std_logic_vector(7 downto 0)
  );
end DW_fp_flt2i_inst;

architecture inst of DW_fp_flt2i_inst is

begin

  -- Instance of DW_fp_flt2i
  U1 : DW_fp_flt2i
  generic map (
    sig_width => inst_sig_width,
    exp_width => inst_exp_width,
    isize => inst_isize
  )
  port map (
    a => inst_a,
    rnd => inst_rnd,
    z => z_inst,
    status => status_inst
  );

end inst;

-- pragma translate_off
configuration DW_fp_flt2i_inst_cfg_inst of DW_fp_flt2i_inst is
  for inst
  end for;
```

```
end DW_fp_ft2i_inst_cfg_inst;  
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_flt2i_inst( inst_a, inst_rnd, z_inst, status_inst );

parameter inst_sig_width = 23;
parameter inst_exp_width = 8;
parameter inst_isize = 32;
parameter inst_ieee_compliance = 0;

input [inst_sig_width+inst_exp_width : 0] inst_a;
input [2 : 0] inst_rnd;
output [inst_isize-1 : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_flt2i
    DW_fp_flt2i #(inst_sig_width, inst_exp_width, inst_isize, inst_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

