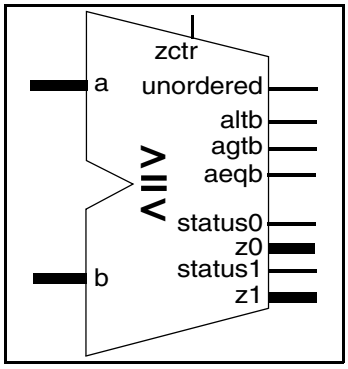
DW_fp_cmp

Floating-Point Comparator

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
- Exponents can range from 3 to 31 bits
- Significand and fractional part of the floating-point number can range from 2 to 256 bits
- Accuracy conforms to IEEE 754 Floating-point standard[1]



Description

DW_fp_cmp is a floating-point module that compares two floating-point numbers at inputs `a` and `b`. The format of these numbers is defined by the number of bits in the exponent (*exp_width*) and the number of bits in the significand (*sig_width*). Only one of the four outputs: `agtb`, `altb`, `aeqb` or `unordered` is set as a result of this operation. The input `zctr` is used to select the function that generates the `z0` and `z1` outputs. DW_fp_cmp also produces `status0` and `status1`, with information about the two `z` outputs.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*sig_width* + *exp_width* +1 bits	Input	Floating-point number
b	*sig_width* + *exp_width* +1 bits	Input	Floating-point number
altb	1 bit	Output	High when `a` is less than `b`
agtb	1 bit	Output	High when `a` is greater than `b`
aeqb	1 bit	Output	High when `a` is equal to `b`
unordered	1 bit	Output	High when one of the inputs is NaN and *ieee_compliance* = 1
z0	*sig_width* + *exp_width* +1 bits	Output	Min(`a`,`b`) when `zctr` = 0 or open; otherwise, Max(`a`,`b`)
z1	*sig_width* + *exp_width* +1 bits	Output	Max(`a`,`b`) when `zctr` = 0 or open; otherwise, Min(`a`,`b`)
status0	8 bits	Output	■ Status flags corresponding to `z0`; for details, see STATUS Flags in the *Datapath Floating-Point Overview* ■ status0[7]: This bit is operand `a` at the output (PassA)

1. For more information, see IEEE 754 Compatibility in the *Datapath Floating-Point Overview*.

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
status1	8 bits	Output	■ Status flags corresponding to `z1`; for details, see STATUS Flags in the *Datapath Floating-Point Overview* ■ status1[7]: This bit is operand `a` at the output (PassA)
zctr	1 bit	Input	Determines value passed to `z0` and `z1`

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers `a`, `b`, `z0`, and `z1`.
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers `a`, `b`, `z0`, and `z1`.
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_FP_CMP_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_fp_cmp_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fp_cmp.v	Verilog simulation model source code

The parameter *ieee_compliance* controls the functionality of this component. When the parameter is set to 0, the component is backward compatible with the DW_cmp_fp (previously available in this library). The DW_cmp_fp does not support NaNs and denormals. NaNs are considered infinities and denormals are considered zeros. Also, the unordered output is always zero in this configuration, and only one of the other outputs (`altb`, `aeqb` or `agtb`) has a value 1 for a given input.

When the `ieee_compliance` parameter is set to 1, the component is fully compliant with the IEEE 754 standard and, therefore, operates with NaNs and denormals. The unordered output is set to one when a NaN value is applied as input.

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.DWpackages.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_cmp_inst is
      generic (
        inst_sig_width : POSITIVE := 23;
        inst_exp_width : POSITIVE := 8;
        inst_ieee_compliance : INTEGER := 0
        );
    port (
    inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    inst_b : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    inst_zctr : in std_logic;
    aeqb_inst : out std_logic;
    altb_inst : out std_logic;
    agtb_inst : out std_logic;
    unordered_inst : out std_logic;
    z0_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    z1_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    status0_inst : out std_logic_vector(7 downto 0);
    status1_inst : out std_logic_vector(7 downto 0)
        );
    end DW_fp_cmp_inst;

architecture inst of DW_fp_cmp_inst is

begin
    -- Instance of DW_fp_cmp
    U1 : DW_fp_cmp
    generic map ( sig_width => inst_sig_width, exp_width => inst_exp_width,
ieee_compliance => inst_ieee_compliance )
    port map ( a => inst_a, b => inst_b, zctr => inst_zctr, aeqb => aeqb_inst, altb =>
altb_inst, agtb => agtb_inst, unordered => unordered_inst, z0 => z0_inst, z1 =>
z1_inst, status0 => status0_inst, status1 => status1_inst );

end inst;

-- pragma translate_off
```

```
configuration DW_fp_cmp_inst_cfg_inst of DW_fp_cmp_inst is
 for inst
 end for; -- inst
end DW_fp_cmp_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_cmp_inst( inst_a, inst_b, inst_zctr, aeqb_inst, altb_inst,
          agtb_inst, unordered_inst, z0_inst, z1_inst, status0_inst,
          status1_inst );

parameter sig_width = 23;
parameter exp_width = 8;
parameter ieee_compliance = 0;


input [sig_width+exp_width : 0] inst_a;
input [sig_width+exp_width : 0] inst_b;
input inst_zctr;
output aeqb_inst;
output altb_inst;
output agtb_inst;
output unordered_inst;
output [sig_width+exp_width : 0] z0_inst;
output [sig_width+exp_width : 0] z1_inst;
output [7 : 0] status0_inst;
output [7 : 0] status1_inst;

    // Instance of DW_fp_cmp
    DW_fp_cmp #(sig_width, exp_width, ieee_compliance)
      U1 ( .a(inst_a), .b(inst_b), .zctr(inst_zctr), .aeqb(aeqb_inst),
           .altb(altb_inst), .agtb(agtb_inst), .unordered(unordered_inst),
           .z0(z0_inst), .z1(z1_inst), .status0(status0_inst),
           .status1(status1_inst) );

endmodule
```

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