

## Key Design Features

- Synthesizable, technology independent VHDL Core
- 32-bit floating-point input
- Signed fixed-point or integer output
- Configurable word and fraction width up to 32 integer bits and 23 fraction bits
- IEEE 754 compliant
- High-speed fully pipelined architecture
- 2 clock-cycle latency

## Applications

- Floating-point pipelines and arithmetic units
- Floating-point processors
- Interfacing between floating-point and fixed-point number systems

## Pin-out Description

| Pin name             | I/O | Description  | Active state |
|----------------------|-----|--|--------------|
| clk                  | in  | Synchronous clock                                      | rising edge  |
| en                   | in  | Clock enable   | high         |
| ieee_in [31:0]       | in  | Floating-point input in IEEE 754 format                | data         |
| fixed_out [dw - 1:0] | out | Signed fixed-point or integer output in [dw fw] format | data         |
| ofl                  | out | Overflow flag  | high         |
| inf                  | out | Infinity flag  | high         |
| den                  | out | Denormalized number flag                               | high         |
| nan                  | out | NaN flag   | high         |

## Generic Parameters

| Generic name | Description                | Type    | Valid range                          |
|--------------|----------------------------|---------|--------------------------------------|
| dw           | Fixed-point word width     | integer | $2 \leq dw \leq 32$                  |
| fw           | Fixed-point fraction width | integer | $0 \leq fw \leq 23$<br>( $fw < dw$ ) |

## Block Diagram

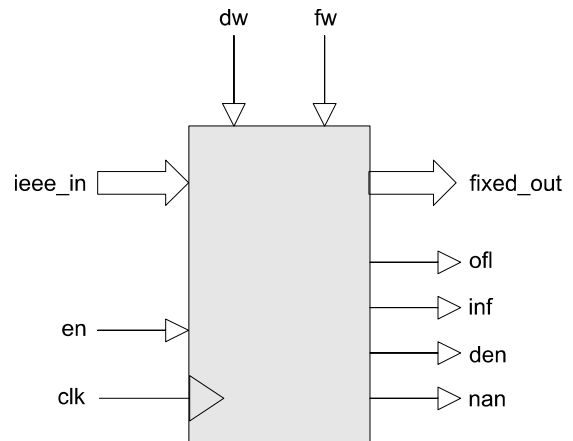
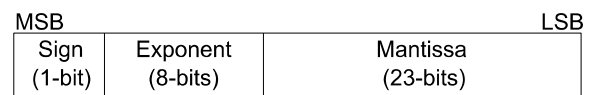


Figure 1: 32-bit Floating-point to Fixed-point converter

## General Description

IEEE\_TO\_FIXED (Figure 1) is a high-speed fully pipelined conversion unit that accepts a 32-bit floating-point number as input and generates a fixed-point representation at the output. The input number is based on the IEEE 754 standard with the bits arranged in the following format:



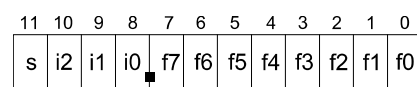
The real number representation of the floating-point number may be calculated as:

$$Value = -1(S) * 2^{(E-127)} * 1.M$$

The fixed-point format is configured using the generic parameters *dw* and *fw*. The value *dw* specifies the width of the output word and *fw* specifies the number of fraction bits.

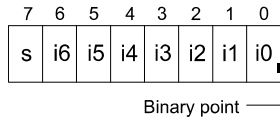
The output may be specified as either a signed fixed-point number or a signed integer. If a signed integer is preferred, then *fw* must be set to 0. In all cases *dw* must be at least 2 bits and *fw* must be less than *dw*.

As an example, to generate a 12-bit fixed-point output with 8 fraction bits the generic parameters must be set to: *dw* = 12, *fw* = 8. In this example the output word would be arranged as follows:



Binary point —

Alternatively, consider the case where the output format is an 8-bit signed integer. Setting  $dw = 8$  and  $fw = 0$ , the output word would be:



In addition, there are some special cases that need to be observed. In the case of a positive floating-point input being larger than the maximum representable fixed-point number then the output will saturate to the maximum positive fixed-point number. Likewise, the output will saturate to the maximum allowable negative number if the a negative floating-point input is too large. In both instances of positive and negative overflow, the signal *off* will be asserted with the output value.

If the input floating-point value is a denormalized number, then the fixed-point output will be 0 and the *den* flag will be asserted high. If the input is a NaN then the fixed-point output will be set to 0 and the *nan* flag asserted. Finally, the case of the input being positive or negative Infinity is treated in the same manner as a positive or negative overflow, with additional *inf* flag also being asserted.

All values are sampled on the rising clock-edge of *clk* when *en* is high. The function has a 2 clock-cycle latency.

## Functional Timing

Figure 2 demonstrates the conversion of two floating-point numbers. The first number is 0xC0FE6666 and the second is 0x40F80000 which represent the real numbers -7.95 and 7.75 respectively. The generic parameters have been set to:  $dw = 12$ ,  $fw = 8$ . The results are available two clock cycles later.

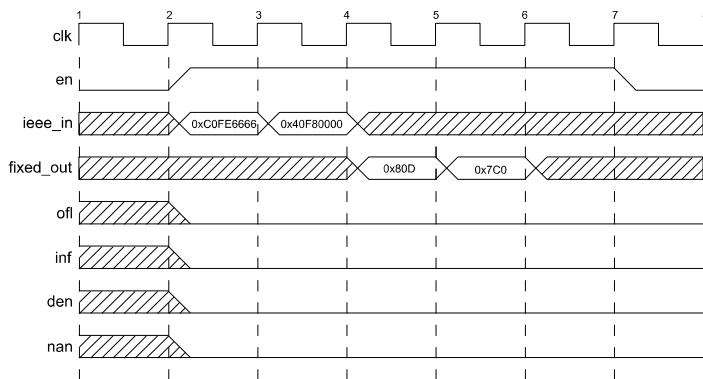


Figure 2: Floating-point to [12 8] fixed-point conversion

Figure 3 shows a sequence of floating-point exceptions that cause the various flags to be exercised. The first value of 0xC101999A represents the number -8.1. The largest negative number allowed in [12 8] format is -8 so this causes the overflow flag to be asserted. The next three values represent a denormalized number, +Infinity and NaN.

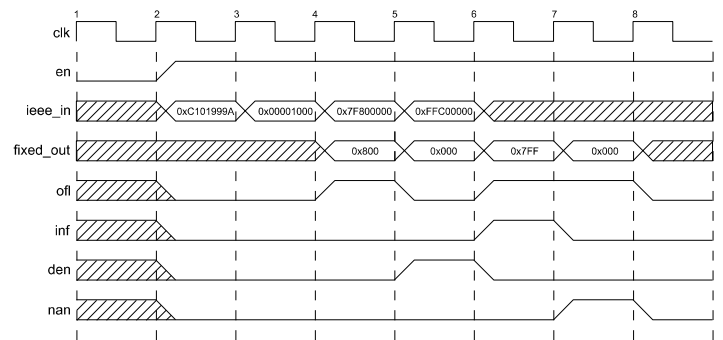


Figure 3: Floating-point to [12 8] fixed-point with flags

## Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

| Source file             | Description          |
|-------------------------|----------------------|
| ieee_to_fixed.vhd       | Top-level component  |
| ieee_to_fixed_bench.vhd | Top-level test bench |

## Functional Testing

An example VHDL testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. ieee\_to\_fixed.vhd
2. ieee\_to\_fixed\_bench.vhd

The VHDL testbench instantiates the top-level component and the user may modify the generic parameters *dw* and *fw* as required

The simulation must be run for at least 2 ms during which time the 'ieee\_to\_fixed' component will receive an input stimulus of randomized floating-point numbers.

The simulation generates two text files called: *ieee\_to\_fixed\_in.txt* and *ieee\_to\_fixed\_out.txt*. These files respectively contain the input and output values captured during the test.

## Synthesis

The source file 'ieee\_to\_fixed.vhd' is the only file required for synthesis. There are no sub-modules in the design.

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx Virtex 5 and the Altera Stratix III series of FPGA devices. The lowest and highest speed grade devices have been chosen in both cases for comparison. Trial synthesis results are shown with the generic parameters set to:  $dw = 32$ ,  $fw = 23$ .

Resource usage is specified after Place and Route.

*VIRTEX 5*

| <b>Resource type</b>         | <b>Quantity used</b> |
|------------------------------|----------------------|
| Slice register               | 78                   |
| Slice LUT                    | 267                  |
| Block RAM                    | 0                    |
| DSP48                        | 0                    |
| Clock frequency (worst case) | 250 MHz              |
| Clock frequency (best case)  | 300 MHz              |

*STRATIX III*

| <b>Resource type</b>         | <b>Quantity used</b> |
|------------------------------|----------------------|
| Register                     | 78                   |
| ALUT                         | 373                  |
| Block Memory bit             | 0                    |
| DSP block 18                 | 0                    |
| Clock frequency (worse case) | 250 MHz              |
| Clock frequency (best case)  | 360 MHz              |

**Revision History**

| <b>Revision</b> | <b>Change description</b>  | <b>Date</b> |
|-----------------|--|-------------|
| 1.0             | Initial revision   | 30/04/2008  |
| 1.1             | Updated synthesis results in line with minor source code changes | 21/09/2011  |
|                 |  |             |

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[Zipcores:](#)

[SKU14](#)