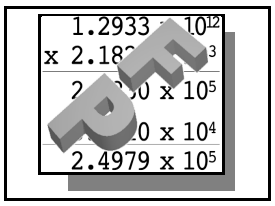
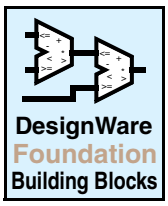




Datapath Floating-Point Overview

Introduction

The Floating Point components constitute a library of functions used to synthesize floating point computational circuits in high-end ASICs. The functions mainly deal with arithmetic operations in floating point format, format conversions, and comparisons. The main features of this library are as follows:

- The format of the floating point numbers that determines the precision of the number that it represents is parameterizable. The user can select the precision based on the number of bits in the exponent and significand (or mantissa). The parameters cover all the IEEE formats.
- Accuracy conforms to the definitions in the IEEE 754 Floating Point standard for the basic arithmetic operations. Improved accuracy is obtained with multi-operand FP components.

For a listing of the Building Block components and associated datasheets, see:

- DesignWare Building Block IP Documentation Overview

This library is an upgrade of the previous library written for the Module Compiler (MC) tool.

Formats

The DesignWare Floating-Point IP Library (DWFP) supports two basic data types: integer and floating point numbers. The length of both data types can be parameterized.

Integer Format

Integers in the floating point library may be signed or unsigned. Unsigned numbers use the standard binary notation. Signed integers use the two's complement notation. The positions of the most-significant bit (MSB) and least-significant bit (LSB) are shown Table 1-1.

Table 1-1 Integer Format Bit Positions

MSB		LSB
n-1	.....	0

Floating-Point Format

The DWFP is a binary floating point library— the radix of the number is always two. Floating point numbers are signed-magnitude numbers encoded with three unsigned integer fields: a sign bit, a biased exponent, and a fractional portion of the significand (fraction), as shown in Table 1-2.

Table 1-2 Floating Point Number Bit Positions

Sign	Biased Exponent	Fraction Bits of Significand
[e + f]	[e + f-1:f]	[f - 1:0]

As shown in Table 1-2, all the FP formats are defined by two integer values: `e` and `f`, where `e` determines the number of biased exponent bits, and `f` determines the number of fraction bits. The MSB at position (`e` + `f`) represents the sign of the floating-point number. Therefore, a floating-point representation is always `e` + `f` + 1 bits long. This definition is consistent with the IEEE Standard 754. A floating-point representation is mapped to a list of three fields: sign, biased exponent, and fraction, which is depicted in this document as (S, E, F).

The numerical value of maximum and minimum normalized numbers for a given format are defined as:

MinNorm = (S, 1, 0)

MaxNorm = (S, 2^e - 2, 2^f - 1)

Note that the exponent field always contains a biased exponent. The value of the bias is defined as (2^{e-1}-1). The value (V) of a normal FP number is calculated as:

$V = (-1)^S \times (\text{fraction} \times 2^{-f} + 1) \times 2^{E - \text{bias}}$

IEEE 754 Compatibility

The DWFP components support more formats than the IEEE 754 floating-point standard describes. For a given set of parameters, the components use FP formats that directly correspond to the standard (for example, single-precision FP format uses *f* = 23 and *e* = 8). Additionally, you can use the *ieee_compliance* parameter to control whether denormals and "not a number" (NaN) values are used.

When denormals (or subnormals) and NaNs are not enabled, denormalized numbers are considered zeros and NaNs are considered infinities. This behavior is consistent with the previous floating-point components developed for Module Compiler (MC).

When denormals and NaNs are enabled, the components are completely compatible with the IEEE standard. Note that all NaNs are treated as one type (there is no concept of quiet and signaling types). For more on how special floating point numbers are represented, see Table 1-5 on page 4.

Floating Point Format Examples

Any floating point format can be implemented in DWFP. Formats defined by IEEE Standard 754 are widely used, but a custom format can also be defined. Examples are shown in the following sections.

IEEE Standard 754 Single-Precision Floating-Point Format

The IEEE 754 single-precision format consists of 32-bit operands. The most significant bit is the sign bit, with 8 bits of exponent and 23 bits of fraction, as shown in Table 1-3.

Table 1-3 IEEE Standard 754 Single-Precision Floating-Point Format

Sign	8-Bit Biased Exponent	23-Bit Fraction Bits of Significand
[31]	[30: 23]	[22:0]
	Zero Exponent = 0 Infinity Exponent = 255 Normal FP Exponent = integer range [1, 254] Bias = 127	Normalized Value = $(-1)^S$ x (1 + fraction * 2^{-23}) x 2^{E-127} Min Norm = $(-1)^S$ x (1.0) x 2^{-126} Max Norm = $(-1)^S$ x (2 - 2^{-23}) x 2^{127}

Customized 18-Bit Floating Point Format

A custom 18-bit floating point format has 6 bits of exponent and 11 bits of normal FP fraction. The most significant bit is reserved for the sign bit. Specific format values are shown in Table 1-4.

Table 1-4 Customized 18-Bit Floating Point

Sign	6-Bit Biased Exponent	11-Bit Fraction Bits of Significand
[17]	[16: 11]	[10: 0]
	Zero Exponent = 0 Infinity Exponent = 63 Normal FP Exponent = integer range [1, 62] Bias = 31	Normalized Value = $(-1)^S$ x (1 + fraction * 2^{-11}) x 2^{E-31} Min Norm = $(-1)^S$ x (1.0) x 2^{-30} Max Norm = $(-1)^S$ x (2 - 2^{-11}) x 2^{31}

Special Floating Point Numbers

Special values in the FP system are represented as shown in Table 1-5. Important values for exponents are:

- Bias value: $bias = 2^{e-1} - 1$
- Biased exponent for infinity: $Einf = 2^e - 1$
- Maximum biased exponent for a normal number representation: $Emax = Einf - 1$
- Minimum biased exponent for a normal number representation: $Emin = 1$

A Normal FP value is represented with a biased exponent *E* in the range [*Emin,Emax*] and any combination of bits as fraction bits of the significand.

Although the NaN generated by a DWFP component (when *ieee_compliance* = 1) has the fraction value 1, the input received by any component is considered a NaN when it has a non-zero fraction value and an exponent value equal to *Einf*.

Table 1-5 Special Floating Point Numbers

Floating Point Number	FP representation (sign, biased exponent, fraction)
+/- Zero	(0 or 1, 0, 0)
+/- Infinity	(0 or 1, Einf, 0)
NaN*	(0, Einf, 1) (when generated by a FP component) (0 or 1, Einf, ≠ 0) (as input)
Denormalized value (denormal)*	(0 or 1, 0, any bit vector)

* These special numbers are used when *ieee_compliance* is set to 1. The value of a denormalized FP is given as $(-1)^S$ x (fraction x 2^{-f}) x $2^{1 - bias}$.

Rounding

Rounding is the process by which a number, regarded as infinitely precise (unbounded exponent and infinitely precise significand), is mapped to a limited precision FP format. All the basic arithmetic functions in the Floating-Point library behave as though they first produce an intermediate result correct to infinite precision, and then round this intermediate result to fit the FP format. Special conditions apply to the more complex components and, when applicable, are described in the component datasheet.

For integers, rounding implies in generating an output that has no data to the right of the binary point. For floating-point this means that the significand is reduced to *f* bits of precision to the right of the binary point. In both cases, the long precision results are first truncated (the least-significant bits (LSBs) are discarded) to get the desired number of bits. This significand is part of one FP number that approximates the infinite precision result. There is another FP number immediately larger (in magnitude) that is obtained by incrementing the truncated significand and making adjustments to have the significand normalized. So, the rounding method defines how a decision is made to use one of these two numbers as a valid approximation for the infinite precision result. Note that the sign of the result is also considered in the rounding procedure.

The exponent values used in the representations of floating-point numbers at this phase are not bounded, which means the rounding procedure is focused on the value of the significand only. It also means that the approximated numbers may be outside the range of representable numbers, which is considered an exception and must be treated separately.

Note that the material above describes functional behavior, not necessarily hardware implementation.

Also, some components do not have rounding control. In these cases, the component uses a single method to keep the rounding error inside bounds, which is documented in the corresponding datasheet.

Rounding Modes

The rounding modes determine the conditions under which the internal long-precision result's significand is reduced to fit the precision defined for the target floating-point format. The floating-point components support dynamic selection of rounding modes, which is determined by a 3-bit input signal named `rnd`.

Table 1-6 describes the rounding modes in terms of the near floating-point values *F1* and *F2* (*F1* < *F2*) of an infinitely precise F.

Table 1-6 Rounding Modes

rnd	Rounding Mode	Rounding Mode Alias	Description
000	IEEE round to nearest (even)	even	Round to the nearest FP number. If *F1* and *F2* are equally near, choose the one with the even significand (the one with LSB = 0).
001	IEEE round to zero	zero	Use *F1* if (F ≥ 0) or *F2* if (F < 0).
010	IEEE round to positive infinity	+ ∞	Round to *F2*.
011	IEEE round to negative infinity	- ∞	Round to *F1*.
100	round to nearest up	up	Round to the nearest FP number. If *F1* and *F2* are equally near, then use *F2* if (F ≥ 0) or *F1* if (F < 0).
101	round away from zero	away	Use *F2* if (F ≥ 0) or *F1* if (F < 0).
110	Reserved		
111	Reserved		

In all rounding modes, if the LSBs being discarded from the infinitely precise significand of F are zeros, then *F1* or *F2* matches F exactly and there is no rounding. When the LSBs being discarded are non-zero, the infinitely precise significand lies strictly between two representable significands, and the rounded number is, therefore, inexact. The rounded floating-point number (*F_rnd*) is decided to be *F1* or *F2*, based on the rounding mode and the value/sign of F. It may be the case that the exponent of *F_rnd* is too large (overflow) or too small (underflow) to be represented as a FP value in the given format. Those cases are called exceptions and are described in "Exceptions" on page 6.

Independent of the value of parameter *ieee_compliance*, all internal calculations of FP components handle denormalized values, and therefore F may be a denormalized FP value. When *ieee_compliance* = 1, *F1* and *F2* can be identified for a denormalized value F, and the rules listed in Table 1-6 are used to generate the final output. When *ieee_compliance* = 0 and F falls between MinNorm and zero (the only possible FP outputs around denormalized values in this case) then the rules defined in Table 1-7 are applied. The absolute value of the largest denormalized number (MaxDenorm = $2^{1-bias} \times (1 - 2^{-f})$) is used in Table 1-7.

Table 1-7 Rounding Denormalized Values when *ieee_compliance* = 0

rnd	Range of F	Rounded Result
0, 4	(MaxDenorm + MinNorm) / 2 ≤ |F| < MinNorm	+MinNorm if F > 0 -MinNorm if F < 0
	|F| < (MaxDenorm + MinNorm) / 2	+0 if F > 0 -0 if F < 0
1	F < 0	-0
	F > 0	+0

Table 1-7 Rounding Denormalized Values when *ieee_compliance* = 0

rnd	Range of F	Rounded Result
2	F < 0	-0
	F > 0	+MinNorm
3	F < 0	-MinNorm
	F > 0	+0
5	F < 0	-MinNorm
	F > 0	+MinNorm

For some algorithms used to implement FP operations, it is not possible to control the rounding direction. In these cases, the FP component may deliver either *F1* or *F2* as the result of a given computation. The actual value delivered depends on the inputs. This is called *faithful rounding* in the documentation.

Exceptions

An exception occurs when rounding yields a number that is out-of- bounds:

- For integers, out-of-bounds means that the rounded significand exceeds the range of the output integer.
- For floating-point formats, out-of-bounds means that the rounded floating-point value has a biased exponent that is outside the range of normal floating-point values for the given format.
  - Overflow happens when the absolute value of the rounded floating-point value is greater than MaxNorm.
  - Underflow happens when the absolute value of the rounded floating-point is less than MinNorm and it is not an exact zero.

  More precisely, these conditions can be defined as a function of the biased exponent of *F_rnd* (call it e_rnd), as follows:
  - Overflow: e_rnd > Emax
  - Underflow: e_rnd < Emin = 1

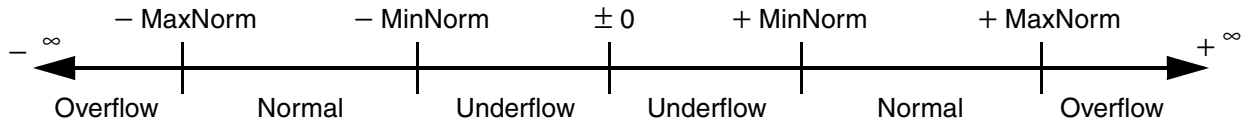
These exceptions are indicated in the status flags as HugeInt, Huge, and Tiny (for more about status flags, see "Status Flags" on page 7). When an exception occurs, the FP components respond as follows:

- For integers, if the rounded result exceeds the largest representable integer with the same sign (HugeInt), the largest representable integer with the correct sign will be the output, regardless of the rounding mode.
- For floating-point values, when not using denormal numbers, if overflow or underflow is detected, the output is determined based on the rounding mode and the sign of the infinitely precise result, as described in Table 1-8. When denormalized values are used, the underflow exception does not apply and the output is represented by a denormalized value that loses precision as it approaches zero.

Table 1-8 Exception Handling After Rounding with Unbounded Exponent

		Rounding Mode					
Condition	**Sign**	**even**	**zero**	**+ ∞**	**– ∞**	**up**	**away**
Overflow	+	+ ∞	+ MaxNorm	+ ∞	+ MaxNorm	+ ∞	+ ∞
	–	– ∞	– MaxNorm	– MaxNorm	– ∞	– ∞	– ∞
Underflow (when not using denormals)	+	+ 0	+ 0	+ MinNorm	+ 0	+ 0	+ MinNorm
	–	– 0	– 0	– 0	– MinNorm	– 0	– MinNorm

The number line below shows the key representable floating-point numbers across the top, and the various ranges of results across the bottom. This illustration provides a useful view of the representable numbers and an intuitive basis for Table 1-8.



Status Flags

Every FP component in the library has an output port called `status`. The component generates a number of flags that depend on the result. Each bit in the `status` output indicates a different condition as detailed in Table 1-9.

Table 1-9 Status Flags

Bit	Flag	Description
0	Zero	Integer or floating point output is zero.
1	Infinity	Floating point output is infinity.
2	Invalid[a]	Floating point operation is not valid. It is also set to 1 when one of the inputs is NaN (`ieee_compliance` = 1)
3	Tiny	Rounded floating-point number with unbounded exponent has a magnitude less than the minimum normalized number, and it is not an *exact zero*[b].
4	Huge	Rounded floating-point number with unbounded exponent has a magnitude greater than the maximum normalized number.
5	Inexact	Integer or floating point output is not equal to the infinitely precise result.
6	HugeInt	Integer result after rounding has a magnitude greater than the largest representable two's complement integer with the same sign.
7	CompSpecific	This flag has its meaning specified for some DW components (see datasheets). When not described in a component datasheet, this flag is not being used and has a value of 0.

a. Invalid operations that set the INVALID status flag are defined in the IEEE Standard 754-2008, such as:
   - Any operation on a NaN input (ieee_compliance = 1)
   - Addition of infinities with opposite signs, or subtraction of infinities with same sign.
   - Multiplication of zero and infinity.
   - Division 0/0 and infinity/infinity.
   - Square root of a negative FP value.
b. When a floating point value, calculated with infinite precision (without rounding), is a zero, then the value is an “exact zero.”

Special Operations

There are some cases when the result of an FP operation is not obvious or is considered invalid:

- Exact zero differences — when the operation is effective subtraction and the result is exactly zero, the sign of the result is determined by the rounding mode. If the rounding mode is *RND* = 3, the output is -0. In all other rounding modes, the output is +0.
- Comparison of zeros — the sign of zeros is not used for comparison (+0 = -0).
- Integer overflows — if HugeInt flag is set, the largest representable integer with correct sign will be output. For positive numbers the output will be 2^{n-1} -1. And for negative numbers the output will be -2^{n-1}.
- Subtraction of infinities — in this case the Invalid flag is asserted and the output is set to NaN (when *ieee_compliance* = 1) or infinity (when *ieee_compliance* = 0).
- Product of Zero and Infinity — the Invalid flag is asserted and the output is set to NaN (when *ieee_compliance* = 1) or infinity (when *ieee_compliance* = 0).
- Complex FP operations, such as dot-products, may assert the Invalid flag based on internal invalid operations. For example, the internal products generated by the component are infinities with opposite signs, forcing the rule related to the subtraction of infinities to be applied. More information about these situations are given in the individual component datasheets.

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