Datapath – Floating Point Overview

(Module Compiler Only)

ntroduction

The Floating Point components comprise 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 comparison functions. 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.
- The parameter range for exponents is from 3 to 31 bits.
- The parameter range for the significand or the fractional part of the floating point number is from 2 bits to 256 bits.
- The parameter range for integers is from 3 to 512 bits.
- 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.

Download instructions for the Floating Point components can be found at the following web address:

http://www.synopsys.com/products/designware/dwest The following list identifies the Floating Point IP:

IP	Description
DW_add_fp	Floating Point Adder - (Module Compiler Only)
DW_cmp_fp	Floating Point Comparator - (Module Compiler Only)
DW_div_fp	Floating Point Divider - (Module Compiler Only)
DW_flt2i_fp	Floating Point to Integer Converter - (Module Compiler Only)
DW_i2flt_fp	Integer to Floating Point Converter - (Module Compiler Only)
DW_mult_fp	Floating Point Multiplier - (Module Compiler Only)

Although a DesignWare license is needed to access the components of this library; again it must be noted that the floating point library does not work with Design Compiler. This library is written specifically for the Module Compiler (MC) tool, and all the Floating Point functions and their supporting functions in the library are written in Module Compiler language (MCL). These functions are not a part of the Module Compiler code; therefore, the library is completely external to MC. The MC startup files contain the information about the library and its location; and this information is used by MC to load the floating point extensions along with other libraries at startup.

The current version of the Floating-Point IP Library has the following operations implemented:

- Floating Point Arithmetic Operations
  - DW_add_fp() – floating point addition
  - DW_mult_fp() – floating point multiplication
- Floating Point Format Conversions
  - DW_i2flt_fp() – conversion from integer to floating point
  - DW_flt2i_fp() – conversion from floating point to integer
- Floating Point Comparison Operations
  - DW_cmp_fp() – compares two floating point numbers

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. The unsigned numbers use the standard binary notation. Signed integers use the two's complement notation. The bit positions are shown Table 1.

Table 1: Integer Format Bit Positions

msb		lsb
n-1	....	0

Floating-Point Format

Floating point numbers are signed-magnitude numbers encoded with three unsigned integer fields: a sign bit, a biased exponent, and a fraction as shown in Table 2.

Table 2: Floating Point Number Bit Positions

Sign	Exponent	Fraction
(e + f)	(e + f -1).........f	(f-1)..........................0

The IEEE 754 floating point format contains two integer parameters: `e` and `f`. The parameter `e` determines the number of biased exponent bits, and `f` determines the number of normalized fraction bits. The msb (`e+f`) signifies the sign of the floating-point number. Therefore, an IEEE 754 floating-point format based number is always `e+f+1` bits long.

The DWFP is a binary floating point library - the radix of the number is always two. The maximum and minimum representable numbers in this format are according to the definitions in Table 5 on page 5.

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

MinNorm = (S, ZeroExponent + 1, 0)

MaxNorm = (S, InfinityExponent–1, 2^{f-1})

Note that the exponent field always contains a biased exponent. The signed magnitude number is expressed as:

$$V = (-1)^{\text{sign}} \times \text{Fraction} \times 2^{\text{exponent}}$$

IEEE 754 Compatibility

The floating point components are not fully compliant with the IEEE 754 floating point standard. Table 3 shows the number interpretations that do meet the IEEE 754 standard:

Table 3: IEEE 754 Number Interpretation Compatibility

IEEE Number Interpretations	Supported by Floating Point Components
NaN (not a number)	No
Infinity	Yes
Normalized number	Yes
Denormalized number	No
Zero	Yes

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 precision can also be defined. Examples are shown on the following pages.

IEEE Standard 754 Single-Precision Floating Point Format

The IEEE 754 single-precision format consists of 32 bits operand. the most significant bit is the sign bit, with eight bits of exponent and 23 bits of normalized fraction, as shown in Table 4.

Table 4: IEEE Standard 754 Single-Precision Floating Point Format

Sign	8-Bit Based Exponent	23-Bit Normalized Fraction
[31]	[30 : 23]	[22:0]

Zero Exponent = 0
Infinity Exponent = 255
Normalized Exponent = integer range [1, 254]
Bias = 127

Normalized Value = $(-1)^S\ 2^{E-127}\ (1.F_{22}......F_1F_0)$
Min Norm = $(-1)^S\ 2^{-126}\ (1.0)$
Max Norm = $(-1)^S\ 2^{127}\ (2 - 2^{-23})$

In practice, the exponent may be parameterized from 3 to 31 bits, and fractions may be tuned from 2 to 256 bits

Customized 18 Bit Floating Point Format

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

Table 5: Values of Representable Floating Point Numbers

Floating Point Type	Definition	Numerical Value
ZeroExponent	0	$(-1)^S 0$
InfinityExponent	2^e-1	(-1)S ∞
NormalizedExponent	ZeroExponent<E<InfinityExponent	$(-1)^S\ 2^{E\text{-bias}}\ (1.F_{f-1}.....F_1F_0\ )$

Table 6: Customized 18-Bit Floating Point

Sign	6-Bit Biased Exponent	11-Bit Normalized Fraction
[17]	[16 : 11]	[10 : 0]

Zero Exponent = 0
Infinity Exponent = 63
Normalized Exponent = integer range [1, 62]
Bias = 31

Normalized Value = $(-1)^S\ 2^{E-31}\ (1.F_{10}......F_1F_0)$
Min Norm = $(-1)^S\ 2^{-30}\ (1.0)$
Max Norm = $(-1)^S\ 2^{-31}\ (1.1111.......11)$

Rounding

Rounding is the process by which a number regarded as infinitely precise is mapped into the destination's format. All functions in the Floating-Point Extensions behave as though they first produced an intermediate result correct to infinite precision, and then rounded this intermediate result to fit into the destination's format.

Significand Rounding (rounding with an unbounded exponent)

The first step in rounding is to round the significand of the infinitely precise result according to the constraints of the significand of the destination's format. For integers this means that the rounded significand has no data to the right of the binary point. For floating-point this means that the significand has value one to the left of the binary point (so called normalized; the number 0 is a special case which will be discussed later), and at most f bits of precision to the right of the binary point. Based on these constraints, data to the right of some bit position (bit 0 for integer, bit -f for floating-point) will be discarded, and the remaining significand will be conditionally incremented at the new LSB position. If the destination format is floating-point, this process of significand rounding may require normalization of the significand before and after rounding and adjustment of the biased exponent. This significand rounding just described is said to assume an unbounded exponent in the case of floating-point, and infinite precision to the left of the binary point in the case of integer.

It is worth noting that significand rounding operates on an infinitely precise result and generates a signed-magnitude result. This is the case even when the destination format is integer. The conversion to two's complement form is done after significand rounding. It is also worth noting that the foregoing description documents the functional behavior, not necessarily the hardware implementation.

Rounding Modes

The rounding modes determine the conditions under which the significand is incremented. The Floating-Point Extensions support dynamically programmable rounding modes. The current rounding mode is encoded on a three bit input signal named `RND`. Table 7 describes the supported rounding modes and how they are encoded.

Table 7: Rounding Modes

RND	Rounding Mode	Rounding Mode Alias	Description
000	IEEE round to nearest (even)	even	Round to the nearest representable significand. If the two significands are equally near, choose the even significand (the one with LSB=0).
001	IEEE round to zero	zero	Never increment the significand.
010	IEEE round to positive infinity	+ ∞	Increment the significand only if the number being rounded is positive.
011	IEEE round to negative infinity	- ∞	Increment the significand only if the number being rounded is negative.
100	round to nearest zero	up	Round to the nearest representable significand. If the two significands are equally near, then increment the significand.
101	round away from zero	away	Increment the significand.
110	Reserved		
111	Reserved		

In all rounding modes, if the significand data being discarded is 0 then the remaining significand will not be incremented. In order to increment, the data being discarded must be non-zero. In all cases there are two choices, the non-incremented significand, and the incremented significand. These two significands are referred to as representable significands. When the data being discarded is non-zero, the infinitely precise significand lies strictly between two representable significands. Rounding must decide which representable significand is appropriate. The following descriptions assume that the data being discarded is non-zero.

Special Rounding

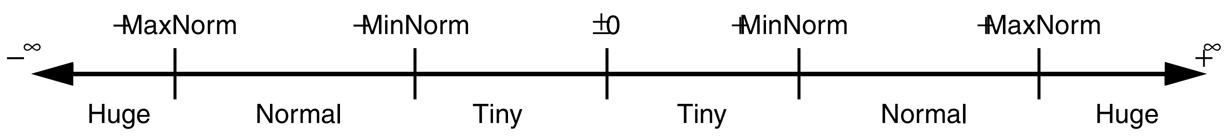
After the significand has been rounded, the last step of rounding is to handle the cases where the number still does not fit into the destination's format. Loosely stated, these are the overflow and underflow conditions. For integers this means that the rounded significand exceeds the range of the output two's complement integer. For floating-point this means that the significand rounding adjusted biased exponent exceeds the range of normalized exponents for the given floating-point format. More simply stated, `E ∉ NormalizedExponent`. These conditions correspond to the status flags HugeInt, Huge, and Tiny. These flags are formally defined later in the status flags section. The delivered results are described here.

- For integers, if the magnitude of the rounded significand 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 outputs, if Huge or Tiny are detected, the output is determined based on the rounding mode and the sign of the infinitely precise result according to Table 8.

Table 8: Special Rounding Modes

		Rounding Mode					
Condition	**Sign**	**even**	**zero**	**+ ∞**	**– ∞**	**up**	**away**
Huge	+	+ ∞	+MaxNorm	+ ∞	+MaxNorm	+ ∞	+ ∞
	–	–∞	–MaxNorm	–MaxNorm	–∞	–∞	–∞
Tiny	+	+ 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 8.



STATUS Flags

Every IP in the library has an optional output port called `STATUS`. The function of the floating point library generate a number of flags that depend on the result. The high bit in the indicated Bit position in Table 9 flags the condition detailed in the table.

Table 9: Status Flags

Bit	Flag	Description
0	Zero	Integer or floating point out is zero.
1	Infinity	Floating point output is infinity.
2	Invalid	Floating point operation is not valid. (0x$^{\infty}$, $^{\infty - \infty}$)
3	Tiny	Non zero floating point output after rounding has a magnitude less than the minimum normalized number.
4	Huge	Finite floating point result after rounding 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 twos compliment integer with the same sign.
7	PassA/Divide by Zero	In a DW_cmp_fp operation, this flag indicates that it is operand A at the output. In a DW_div_fp function this flag indicates Divide-by-Zero operation.

Special Operations

In floating point arithmetic there are certain situations when the result can be obtained through normal computations. Consider the case when 0 and ∞ are at the input, the result is determined automatically. There are some cases when the result is not very obvious and the details are as follows:

- 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 $-\infty$ the output is -0. In all other rounding mode the output is +0.This is special because the result cannot be normalized since all the bits of the significand are all zero.
- Comparison of zeros - the sign of zeros is not used for comparison, i.e. +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 with output being $\pm\infty$ depending on the rounding mode.
- Product of Zero and Infinity - for operation $\pm(0 \times \infty)$ the invalid flag is asserted and the output is $\pm\infty$ depending on the rounding mode.