Contents [hide]

MICROCHIP CoreFPU Core Floating Point Unit

Introduction

- The Core Floating Point Unit (CoreFPU) is designed for floating-point arithmetic and conversion operations, for single and double precision floating-point numbers. CoreFPU supports fixed-point to floating-point and floating-point to fixed-point conversions and floating-point addition, subtraction, and multiplication operations. The IEEE® Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation.
- Important: CoreFPU supports calculations with normalized numbers only, and only the Verilog language is supported; VHDL is not supported.

Summary

The following table provides a summary of the CoreFPU characteristics.

Table 1. CoreFPU Characteristics

Core Version	This document applies to CoreFPU v3.0.
Supported Device Families	• PolarFire® SoC • PolarFire • RTG4™
Supported Tool Flow	Requires Libero® SoC v12.6 or later releases.
Licensing	CoreFPU is not license locked.
Installation Instructions	CoreFPU must be installed to the IP Catalog of Libero SoC automatically through the IP Catalog update function. Alternatively, CoreFPU could be manually downloaded from the catalog. Once the IP core is installed, it is configured, generated and instantiated within SmartDesign for inclusion in the project.

Device Utilization and Performance	A summary of utilization and performance information for CoreFPU is listed in Device Resource Utilization and Performance.

CoreFPU Change Log Information

This section provides a comprehensive overview of the newly incorporated features, beginning with the most recent release. For more information about the problems resolved, see the Resolved Issues section.

Version	What’s New
v3.0	Implemented additional output flags to enhance the accuracy of the IP
v2.1	Added the double precision feature
v2.0	Updated the timing waveforms
v1.0	First production release of CoreFPU

1. Features

CoreFPU has the following key features:

- Supports Single and Double Precision Floating Numbers as per IEEE-754 Standard
- Supports Conversions as listed:
    - Fixed-point to Floating-point conversion
    - Floating-point to Fixed-point conversion
- Supports Arithmetic Operations as listed:
    - Floating-point addition
    - Floating-point subtraction
    - Floating-point multiplication
- Provides the Rounding Scheme (Round to nearest even) for the Arithmetic Operations only
- Provides Flags for Overflow, Underflow, Infinity (Positive Infinity, Negative Infinity), Quiet NaN (QNaN) and Signalling NaN (SNaN) for Floating-Point Numbers.

- Supports Fully pipelined implementation of Arithmetic Operations
- Provides Provision to configure the Core for Design Requirements

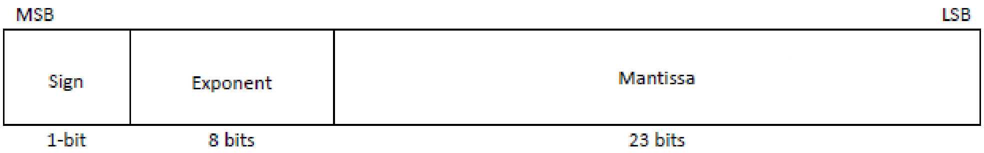
Functional Description

- The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation. The term floating-point refers to the radix point of the number (decimal point or binary point), which is placed anywhere with respect to the significant digits of the number.
  A floating-point number is typically expressed in the scientific notation, with a fraction (F), and an exponent (E) of a certain radix (r), in the form of F × r^E. Decimal numbers use radix of 10 (F × 10^E); while binary numbers use radix of 2 (F × 2^E).
- The representation of the floating-point number is not unique. For example, the number 55.66 is represented as 5.566 × 10^1, 0.5566 × 10^2, 0.05566 × 10^3, and so on. The fractional part is normalized. In the normalized form, there is only a single non-zero digit before the radix point. For example, decimal number 123.4567 is normalized as 1.234567 × 10^2; binary number 1010.1011B is normalized as 1.0101011B × 2^3.
- It is important to note that floating-point numbers suffer from loss of precision when represented with a fixed number of bits (for example, 32-bit or 64-bit). This is because there are an infinite number of real numbers (even within a small range from 0.0 to 0.1). On the other hand, an
  n- bit binary pattern represents a finite 2^n distinct numbers. Hence, not all the real numbers are represented. The nearest approximation is used instead, which results in the loss of accuracy.

The single precision floating-point number is represented as follows:

- Sign bit: 1-bit
- Exponent width: 8 bits
- Significand precision: 24 bits (23 bits are explicitly stored)

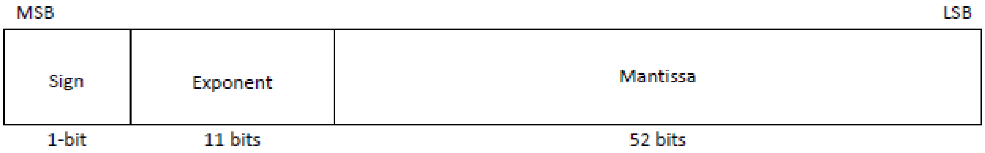
Figure 2-1. 32-bit Frame



The double precision floating-point number is represented as follows:

- Sign bit: 1-bit
- Exponent width: 11 bits
- Significand precision: 53 bits (52 bits are explicitly stored)

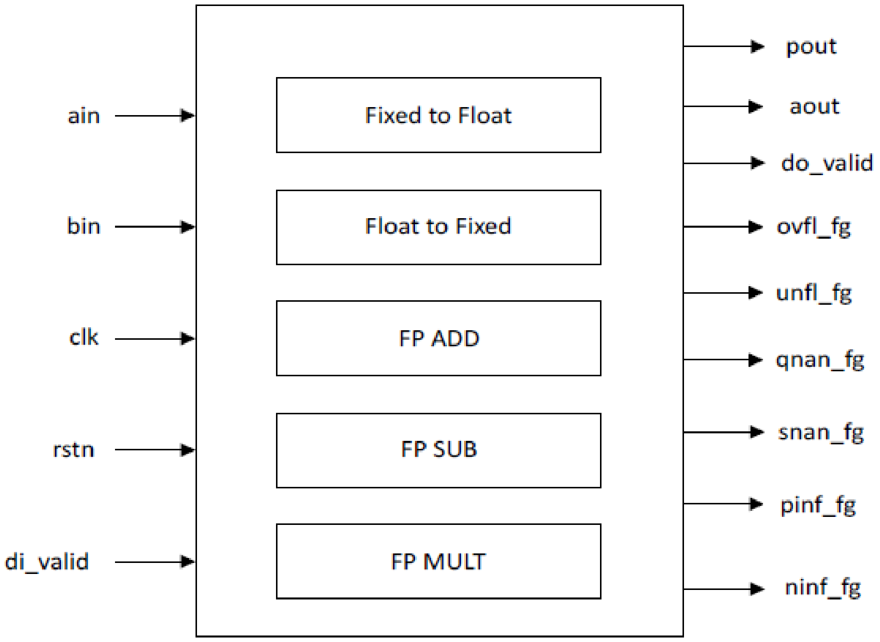
Figure 2-2. 64-bit Frame



The CoreFPU is the top-level integration of the two conversion modules (Fixed to Float point and Float to Fixed point) and three arithmetic operations (FP ADD, FP SUB, and FP MULT). The user can configure any one of the operations based on the requirement so that the resources are utilized for the selected operation.

The following figure shows the top level CoreFPU block diagram with ports.

Figure 2-3. CoreFPU Ports Block Diagram



The following table lists the width of the Input and Output ports. Table 2-1. Input and Output Port Width

Signal	Single Precision Width	Double Precision Width
ain	[31:0]	[63:0]
bin	[31:0]	[63:0]
aout	[31:0]	[63:0]
pout	[31:0]	[63:0]

Fixed-Point to Floating-Point (Conversion)

CoreFPU configured as fixed to floating-point infers the fixed-point to floating-point conversion module. The input (ain) to CoreFPU is any fixed-point number containing the integer and fractional bits. The CoreFPU configurator has the options to select the input integer and fraction widths. The input is valid on di_valid signal and output is valid on do_valid. The output (aout) of the fixed to float operation is in single or double precision floating-point format.

Example for fixed-point to floating-point conversion operation is listed in the following table.

Table 2-2. Example for Fixed-Point to Floating-Point Conversion

Fixed-Point Number			Floating-Point Number			
ain	Integer	Fraction	aout	Sign	Exponent	Mantissa
0x12153524 (32-bit)	00010010000101010	011010100100100	0x4610a9a9	0	10001100	00100001010100110101001

0x00000000008CCC (64-bit)	0000000000000000000000000000000000000000000000000	000110011001100	0x3FF199999999999A	0	01111111111	0001100110011001100110011001100110011001100110011010

Floating-Point to Fixed-Point (Conversion)

CoreFPU configured as floating to fixed-point infers the floating-point to fixed-point conversion module. The input (ain) to CoreFPU is any single or double precision floating-point number and produces an output (aout) in fixed-point format containing integer and fractional bits. The input is valid on di_valid signal and output is valid on do_valid. The CoreFPU configurator has the options to select the output integer and fraction widths. Example for floating-point to fixed-point conversion operation is listed in the following table.

Table 2-3. Example for Floating-Point to Fixed-Point Conversion

Floating-Point Number				Fixed-Point Number		
ain	Sign	Exponent	Mantissa	aout	Integer	Fraction
0x41bd6783 (32-bit)	0	10000011	01111010110011110000011	0x000bd678	0000000000010111	1010110011111000

0x4002094c447c30d3 (64-bit)	0	10000000000	0010000010010100110001000100011111000011000011010011	0x0000000000012095	0000000000000000000000000000000000000000000000000010	010000010010101

Floating-Point Addition (Arithmetic Operation)

CoreFPU configured as FP ADD infers the floating-point addition module. It adds the two floating-point numbers (ain and bin) and provides the output (pout) in floating-point format. The input and output are single or double precision floating-point numbers. The input is valid on di_valid signal and output is valid on do_valid. The core produce ovfl_fg (Overflow), qnan_fg (Quiet Not a Number), snan_fg (Signalling Not a Number), pinf_fg(Positive Infinity), and ninf_fg (Negative Infinity) flags based on the addition operation.

Examples for floating-point addition operation are listed in the following tables.

Table 2-4. Example for Floating-Point Addition Operation (32-bit)

Floating-Point Value	Sign	Exponent	Mantissa
Floating-point input 1 ain (0x4e989680)	0	10011101	00110001001011010000000
Floating-point input 2 bin (0x4f191b40)	0	10011110	00110010001101101000000
Floating-point addition output pout (0x4f656680)	0	10011110	11001010110011010000000

Table 2-5. Example for Floating-Point Addition Operation (64-bit)

Floating-Point Value	Sign	Exponent	Mantissa
Floating-point input 1 ain (0x3ff4106ee30caa32)	0	01111111111	0100000100000110111011100011000011001010101000110010
Floating-point input 2 bin (0x40020b2a78798e61)	0	10000000000	0010000010110010101001111000011110011000111001100001
Floating-point addition output pout (0x400c1361e9ffe37a)	0	10000000000	1100000100110110000111101001111111111110001101111010

Floating-Point Subtraction (Arithmetic Operation)

CoreFPU configured as FP SUB infers the floating-point subtraction module. It subtracts the two floating-point numbers (ain and bin) and provides the output (pout) in floating-point format. The input and output are single or double precision floating-point numbers. The input is valid on di_valid signal and output is valid on do_valid. The core produce ovfl_fg (Overflow), unfl_fg (underflow), qnan_fg (Quiet Not a Number), snan_fg (Signalling Not a Number), pinf_fg (Positive Infinity), and ninf_fg (Negative Infinity) flags based on the subtraction operation.

Examples for floating-point subtraction operation are listed in the following tables.

Table 2-6. Example for Floating-Point Subtraction Operation (32-bit)

Floating-Point Value	Sign	Exponent	Mantissa
Floating-point input 1 ain (0xac85465f)	1	01011001	00001010100011001011111

Floating-point input 2 bin (0x2f516779)	0	01011110	10100010110011101111001
Floating-point subtraction output pout (0xaf5591ac)	1	01011110	10101011001000110101011

Floating-Point Value	Sign	Exponent	Mantissa
Floating-point input 1 ain (0x405569764adff823)	0	10000000101	0101011010010111011001001010110111111111100000100011
Floating-point input 2 bin (0x4057d04e78dee3fc)	0	10000000101	0111110100000100111001111000110111101110001111111100
Floating-point subtraction output pout (0xc02336c16ff75ec8)	1	10000000010	0011001101101100000101101111111101110101111011001000

Floating-Point Multiplication (Arithmetic Operation)

CoreFPU configured as FP MULT infers the floating-point multiplication module. It multiplies the two floating-point numbers (ain and bin) and provides the output (pout) in floating-point format. The input and output are single or double precision floating-point numbers. The input is valid on di_valid signal and output is valid on do_valid. The core produce ovfl_fg (Overflow), unfl_fg (Underflow), qnan_fg (Quiet Not a Number), snan_fg (Signalling Not a Number), pinf_fg (Positive Infinity), and ninf_fg (Negative Infinity) flags based on the multiplication operation.

Examples for floating-point multiplication operation are listed in the following tables.

Table 2-8. Example for Floating-Point Multiplication Operation (32-bit)

Floating-Point Value	Sign	Exponent	Mantissa
Floating-point input 1 ain (0x1ec7a735)	0	00111101	10001111010011100110101
Floating-point input 2 bin (0x6ecf15e8)	0	11011101	10011110001010111101000
Floating-point Multiplication output pout (0x4e21814a)	0	10011100	01000011000000101001010

Floating-Point Value	Sign	Exponent	Mantissa
Floating-point input 1 ain (0x40c1f5a9930be0df)	0	10000001100	0001111101011010100110010011000010111110000011011111
Floating-point input 2 bin (0x400a0866c962b501)	0	10000000000	1010000010000110011011001001011000101011010100000001
Floating-point multiplication output pout (0x40dd38a1c3e2cae9)	0	10000001101	1101001110001010000111000011111000101100101011101001

Truth Table for Addition and Subtraction

The following truth tables list the values for addition and subtraction operation. Table 2-10. Truth Table for Addition

Data A	Data B	Sign Bit	Result	Overflow	Underflow	SNaN	QNaN	PINF	NINF
QNaN/SNaN	x	0	POSQNaN	0	0	0	1	0	0
x	QNaN/SNaN	0	POSQNaN	0	0	0	1	0	0
zero	zero	0	POSZERO	0	0	0	0	0	0
zero	posfinite(y)	0	posfinite(y)	0	0	0	0	0	0
zero	negfinite(y)	1	negfinite(y)	0	0	0	0	0	0
zero	posinfinite	0	posinfinite	0	0	0	0	1	0
zero	neginfinite	1	neginfinite	0	0	0	0	0	1
posfinite(y)	zero	0	posfinite(y)	0	0	0	0	0	0
posfinite	posinfinite	0	posinfinite	0	0	0	0	1	0

Table 2-10. Truth Table for Addition (continued)

Data A	Data B	Sign Bit	Result	Overflow	Underflow	SNaN	QNaN	PINF	NINF
posfinite	neginfinite	1	neginfinite	0	0	0	0	0	1

negfinite(y)	zero	1	negfinite(y)	0	0	0	0	0	0
negfinite	posinfinite	0	posinfinite	0	0	0	0	1	0
negfinite	neginfinite	1	neginfinite	0	0	0	0	0	1
posinfinite	zero	0	posinfinite	0	0	0	0	1	0
posinfinite	posfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	negfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	posinfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	neginfinite	0	POSQNaN	0	0	0	1	0	0
neginfinite	zero	1	neginfinite	0	0	0	0	0	1
neginfinite	posfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	negfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	posinfinite	0	POSQNaN	0	0	0	1	0	0
neginfinite	neginfinite	1	neginfinite	0	0	0	0	0	1

posfinite	posfinite	0	posfinite	0	0	0	0	0	0
posfinite	posfinite	0	posinfinite	0	0	0	0	1	0
posfinite	posfinite	0/1	QNaN	0	0	0	1	0	0
posfinite	posfinite	0/1	SNaN	0	0	1	0	0	0
posfinite	posfinite	0	POSSNaN	1	0	1	0	0	0
posfinite	negfinite	0	posfinite	0	0	0	0	0	0
posfinite	negfinite	1	negfinite	0	0	0	0	0	0
posfinite	negfinite	0	POSSNaN	0	1	1	0	0	0
negfinite	posfinite	0	posfinite	0	0	0	0	0	0
negfinite	posfinite	1	negfinite	0	0	0	0	0	0
negfinite	posfinite	0	POSSNaN	0	1	1	0	0	0
negfinite	negfinite	1	negfinite	0	0	0	0	0	0
negfinite	negfinite	1	neginfinite	0	0	0	0	0	1
negfinite	negfinite	0/1	QNaN	0	0	0	1	0	0
negfinite	negfinite	0/1	SNaN	0	0	1	0	0	0
negfinite	negfinite	0	POSSNaN	1	0	1	0	0	0

Data A	Data B	Sign Bit	Result	Overflow	Underflow	SNaN	QNaN	PINF	NINF
QNaN/SNaN	x	0	POSQNaN	0	0	0	1	0	0
x	QNaN/SNaN	0	POSQNaN	0	0	0	1	0	0
zero	zero	0	POSZERO	0	0	0	0	0	0
zero	posfinite(y)	1	negfinite(y)	0	0	0	0	0	0
zero	negfinite(y)	0	posfinite(y)	0	0	0	0	0	0
zero	posinfinite	1	neginfinite	0	0	0	0	0	1
zero	neginfinite	0	posinfinite	0	0	0	0	1	0
posfinite(y)	zero	0	posfinite(y)	0	0	0	0	0	0
posfinite	posinfinite	1	neginfinite	0	0	0	0	0	1
posfinite	neginfinite	0	posinfinite	0	0	0	0	1	0
negfinite(y)	zero	1	negfinite(y)	0	0	0	0	0	0
negfinite	posinfinite	1	neginfinite	0	0	0	0	0	1

Table 2-11. Truth Table for Subtraction (continued)

Data A	Data B	Sign Bit	Result	Overflow	Underflow	SNaN	QNaN	PINF	NINF
negfinite	neginfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	zero	0	posinfinite	0	0	0	0	1	0
posinfinite	posfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	negfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	posinfinite	0	POSQNaN	0	0	0	1	0	0
posinfinite	neginfinite	0	posinfinite	0	0	0	0	1	0
neginfinite	zero	1	neginfinite	0	0	0	0	0	1
neginfinite	posfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	negfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	posinfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	neginfinite	0	POSQNaN	0	0	0	1	0	0

posfinite	posfinite	0	posfinite	0	0	0	0	0	0
posfinite	posfinite	1	negfinite	0	0	0	0	0	0
posfinite	posfinite	0	POSSNaN	0	1	1	0	0	0
posfinite	negfinite	0	posfinite	0	0	0	0	0	0
posfinite	negfinite	0	posinfinite	0	0	0	0	1	0
posfinite	negfinite	0/1	QNaN	0	0	0	1	0	0
posfinite	negfinite	0/1	SNaN	0	0	1	0	0	0
posfinite	negfinite	0	POSSNaN	1	0	1	0	0	0
negfinite	posfinite	1	negfinite	0	0	0	0	0	0
negfinite	posfinite	1	neginfinite	0	0	0	0	0	1
negfinite	posfinite	0/1	QNaN	0	0	0	1	0	0
negfinite	posfinite	0/1	SNaN	0	0	1	0	0	0
negfinite	posfinite	0	POSSNaN	1	0	1	0	0	0
negfinite	negfinite	0	posfinite	0	0	0	0	0	0
negfinite	negfinite	1	negfinite	0	0	0	0	0	0
negfinite	negfinite	0	POSSNaN	0	1	1	0	0	0

Important:

- They in the preceding tables denotes any number.
- The in the preceding tables denotes a don’t care condition.

Truth Table for Multiplication

The following truth table lists the values for multiplication operation.

Table 2-12. Truth Table for Multiplication

Data A	Data B	Sign Bit	Result	Overflow	Underflow	SNaN	QNaN	PINF	NINF
QNaN/SNaN	x	0	POSQNaN	0	0	0	1	0	0
x	QNaN/SNaN	0	POSQNaN	0	0	0	1	0	0
zero	zero	0	POSZERO	0	0	0	0	0	0
zero	posfinite	0	POSZERO	0	0	0	0	0	0
zero	negfinite	0	POSZERO	0	0	0	0	0	0
zero	posinfinite	0	POSQNaN	0	0	0	1	0	0
zero	neginfinite	0	POSQNaN	0	0	0	1	0	0

Table 2-12. Truth Table for Multiplication (continued)

Data A	Data B	Sign Bit	Result	Overflow	Underflow	SNaN	QNaN	PINF	NINF

posfinite	zero	0	POSZERO	0	0	0	0	0	0
posfinite	posinfinite	0	posinfinite	0	0	0	0	1	0
posfinite	neginfinite	1	neginfinite	0	0	0	0	0	1
negfinite	zero	0	POSZERO	0	0	0	0	0	0
negfinite	posinfinite	1	neginfinite	0	0	0	0	0	1
negfinite	neginfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	zero	0	POSQNaN	0	0	0	1	0	0
posinfinite	posfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	negfinite	1	neginfinite	0	0	0	0	0	1
posinfinite	posinfinite	0	posinfinite	0	0	0	0	1	0
posinfinite	neginfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	zero	0	POSQNaN	0	0	0	1	0	0

neginfinite	posfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	negfinite	0	posinfinite	0	0	0	0	1	0
neginfinite	posinfinite	1	neginfinite	0	0	0	0	0	1
neginfinite	neginfinite	0	posinfinite	0	0	0	0	1	0
posfinite	posfinite	0	posfinite	0	0	0	0	0	0
posfinite	posfinite	0	posinfinite	0	0	0	0	1	0
posfinite	posfinite	0	POSQNaN	0	0	0	1	0	0
posfinite	posfinite	0	POSSNaN	0	0	1	0	0	0
posfinite	posfinite	0	POSSNaN	1	0	1	0	0	0
posfinite	posfinite	0	POSSNaN	0	1	1	0	0	0
posfinite	negfinite	1	negfinite	0	0	0	0	0	0
posfinite	negfinite	1	neginfinite	0	0	0	0	0	1
posfinite	negfinite	0	POSQNaN	0	0	0	1	0	0

posfinite	negfinite	0	POSSNaN	0	0	1	0	0	0
posfinite	negfinite	0	POSSNaN	1	0	1	0	0	0
posfinite	negfinite	0	POSSNaN	0	1	1	0	0	0
negfinite	posfinite	1	negfinite	0	0	0	0	0	0
negfinite	posfinite	1	neginfinite	0	0	0	0	0	1
negfinite	posfinite	0	POSQNaN	0	0	0	1	0	0
negfinite	posfinite	0	POSSNaN	0	0	1	0	0	0
negfinite	posfinite	0	POSSNaN	1	0	1	0	0	0
negfinite	posfinite	0	POSSNaN	0	1	1	0	0	0
negfinite	negfinite	0	posfinite	0	0	0	0	0	0
negfinite	negfinite	0	posinfinite	0	0	0	0	1	0
negfinite	negfinite	0	POSQNaN	0	0	0	1	0	0
negfinite	negfinite	0	POSQNaN	0	0	1	0	0	0

negfinite	negfinite	0	POSQNaN	1	0	1	0	0	0
negfinite	negfinite	0	POSQNaN	0	1	1	0	0	0

Important:

Sign Bit ‘0’ defines positive output and ‘1’ defines negative output.
The x in the preceding table denotes don’t care condition.

CoreFPU Parameters and Interface Signals

This section discusses the parameters in the CoreFPU Configurator settings and I/O signals.

Configuration GUI Parameters

There are number of configurable options that apply to the FPU unit as shown in the following table. If a configuration other than default is required, configuration dialog box is used to select appropriate values for the configurable option.

Table 3-1. CoreFPU Configuration GUI Parameters

Parameter Name	Default	Description
Precision	Single	Select the operation as required: Single Precision Double Precision

Conversion Type	Fixed-point to Floating-point conversion	Select the operation as required: • Fixed-point to Floating-point conversion • Floating-point to Fixed-point conversion • Floating-point addition • Floating-point subtraction • Floating-point multiplication
Input Fraction Width1	15	Configures the fractional point in the Input ain and bin signals Valid range is 31–1
Output Fraction Width2	15	Configures the fractional point in the Output aout signals Valid range is 51–1

Important:

1. This parameter is configurable only during fixed-point to floating-point conversion.
2. This parameter is configurable only during floating-point to fixed-point conversion.

Input and Output Signals (Ask a Question)

The following table lists the input and output port signals of CoreFPU.

Table 3-2. Port Description

Signal Name	Width	Type	Description
clk	1	Input	Main system clock

rstn	1	Input	Active-low asynchronous reset
di_valid	1	Input	Active-high input valid This signal indicates that the data present on ain[31:0], ain[63:0] and bin[31:0], bin[63:0] is valid.
ain	32/64	Input	A Input Bus (It is used for all operations)
bin[1]	32/64	Input	B Input Bus (It is used for arithmetic operations only)
aout[2]	32/64	Output	Output value when fixed to floating-point or floating to fixed-point conversion operations are selected.
pout[1]	32/64	Output	Output value when addition, subtraction, or multiplication operations are selected.

Table 3-2. Port Description (continued)

Signal Name	Width	Type	Description
do_valid	1	Output	Active-high signal This signal indicates that the data present on pout/aout data bus is valid.

ovfl_fg[3]	1	Output	Active-high signal This signal indicates the overflow during floating-point operations.
unfl_fg	1	Output	Active-high signal This Signal indicates the underflow during floating point operations.
qnan_fg[3]	1	Output	Active-high signal This signal indicates the Quiet Not a Number (QNaN) during floating-point operations.
snan_fg	1	Output	Active-high signal This signal indicates the Signalling Not-a-Number (SNaN) during floating point operations.
pinf_fg[3]	1	Output	Active-high signal This signal indicates the positive infinity during floating-point operations.
ninf_fg	1	Output	Active-high signal This signal indicates the negative infinity during floating-point operations.

Important:

1. This port is available only for floating-point addition, subtraction, or multiplication operations.
2. This port is available only for fixed-point to floating-point and floating-point to fixed-

point conversion operations.

3. This port is available for floating-point to fixed-point, floating-point addition, floating-point subtraction, and floating-point multiplication.

Implementation of CoreFPU in Libero Design Suite

This section describes the implementation of CoreFPU in the Libero Design Suite.

SmartDesign

CoreFPU is available for download in the Libero IP catalog through the web repository. Once it is listed in the catalog, the core is instantiated using the SmartDesign flow. For information on using SmartDesign to configure, connect, and generate cores, see Libero SoC online help.

After configuring and generating the core instance, the basic functionality is simulated using the testbench supplied with the CoreFPU. The testbench parameters automatically adjust to the CoreFPU configuration. The CoreFPU is instantiated as a component of a larger design.

Figure 4-1. SmartDesign CoreFPU Instance for Arithmetic Operations

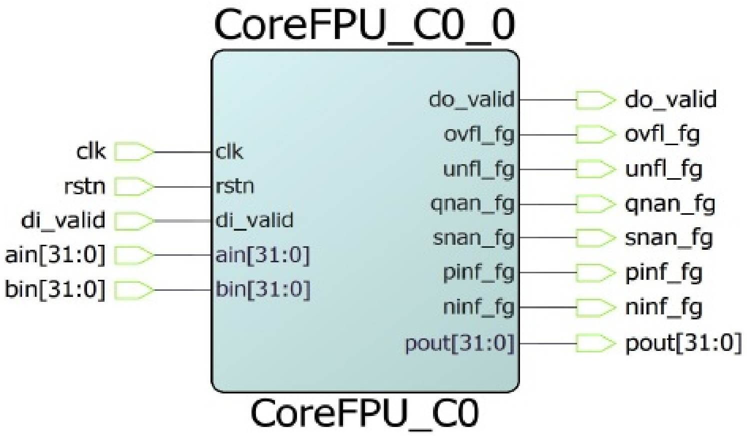
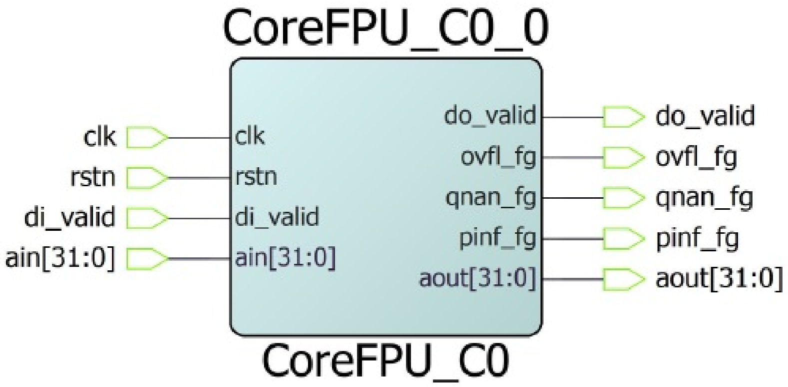


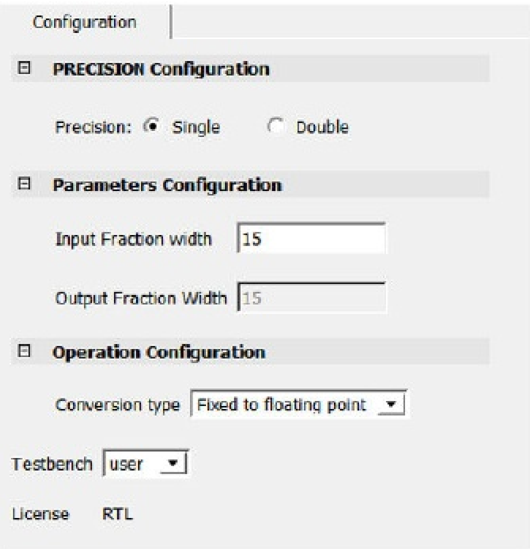
Figure 4-2. SmartDesign CoreFPU Instance for Conversion Operation



Fixed-Point to Floating-Point Conversion

During fixed-point to floating-point conversion, the Input Fraction Width is configurable. The Output Width is set to 32-bit for single precision and 64-bit for double precision floating-point by default.

To convert from fixed-point to floating-point, select Fixed to floating point Conversion type, as shown in the following figure.



Configuration

⊟ PRECISION Configuration

Precision: ● Single ○ Double

⊟ Parameters Configuration

Input Fraction width 15

Output Fraction Width 15

⊟ Operation Configuration

Conversion type Fixed to floating point ▾

Testbench user ▾

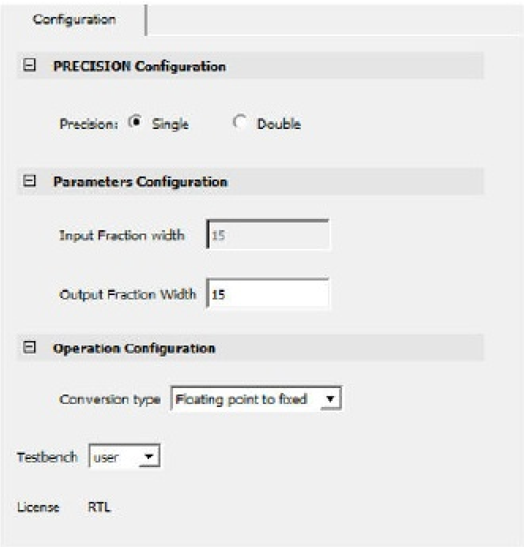
License RTL

Floating-Point to Fixed-Point

During floating-point to fixed-point conversion, the Output Fractional Width is configurable, and the Input Width is set to 32-bit for single precision and 64-bit for double precision floating-point by default.

To convert from floating-point to fixed-point, select Floating point to fixed Conversion type, as shown in the following figure.

Figure 4-4. CoreFPU Configurator for Floating Point to Fixed



Configuration

⊟ PRECISION Configuration

Precision: ● Single ○ Double

⊟ Parameters Configuration

Input Fraction width 15

Output Fraction Width 15

⊟ Operation Configuration

Conversion type Floating point to fixed

Testbench user

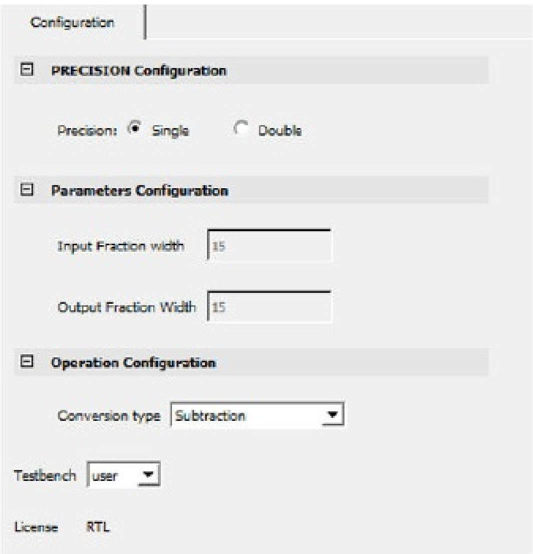
License RTL

Floating-Point Addition/Subtraction/Multiplication

During floating-point addition, subtraction, and multiplication operation, the Input Fraction Width and Output Fraction Width are not configurable as these are floating-point arithmetic operations, and the Input/Output Width is set to 32-bit single precision and 64-bit for double precision floating-point by default.

The following figure shows the CoreFPU configurator for floating point subtraction operation.

Figure 4-5. CoreFPU Configurator for Floating Point Subtraction



Configuration

⊟ PRECISION Configuration

Precision: ● Single ○ Double

⊟ Parameters Configuration

Input Fraction width 15

Output Fraction Width 15

⊟ Operation Configuration

Conversion type Subtraction

Testbench user

License RTL

Simulation (Ask a Question)

To run simulations, in the core configuration window, select User Testbench. After generating the CoreFPU, the pre-synthesis testbench Hardware Description Language (HDL) files are installed in Libero.

Simulation Waveforms (Ask a Question)

This section discusses the simulation waveforms for CoreFPU.

The following figures show the waveform of fixed-point to floating-point conversion for both 32-bit and 64-bit.

Figure 4-6. Fixed-Point to Floating-Point Conversion (32-bit)

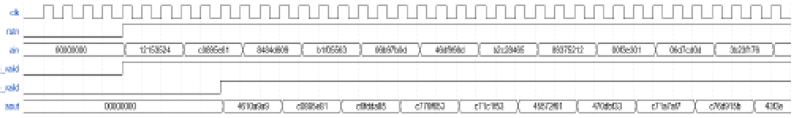


Figure 4-7. Fixed-Point to Floating-Point Conversion (64-bit)

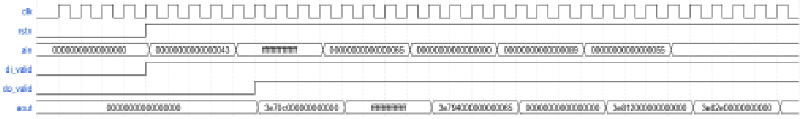


Figure 4-8. Floating-Point to Fixed-Point Conversion (32-bit)

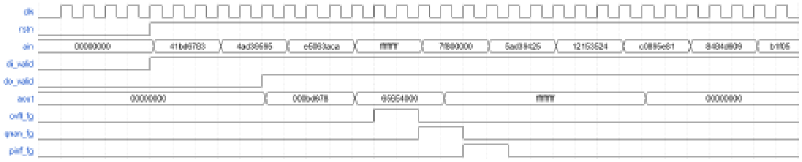
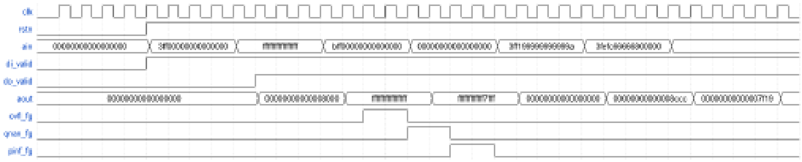


Figure 4-9. Floating-Point to Fixed-Point Conversion (64-bit)



The following figures show the waveform of floating-point addition operation for both 32-bit and 64-bit.

Figure 4-10. Floating-Point Addition (32-bit)

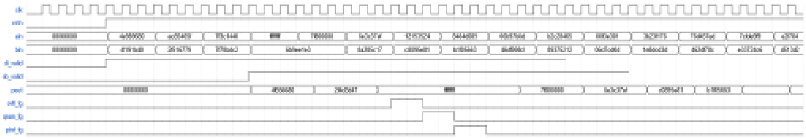


Figure 4-11. Floating-Point Addition (64-bit)

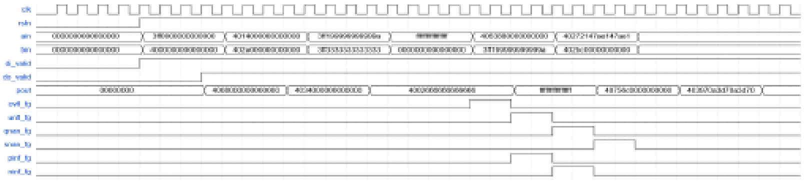


Figure 4-12. Floating-Point Subtraction (32-bit)

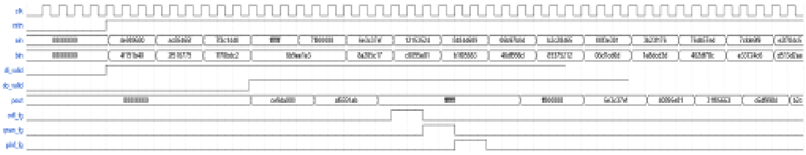


Figure 4-13. Floating-Point Subtraction (64-bit)

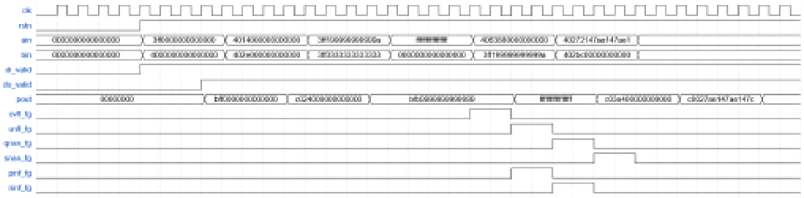


Figure 4-14. Floating-Point Multiplication (32-bit)

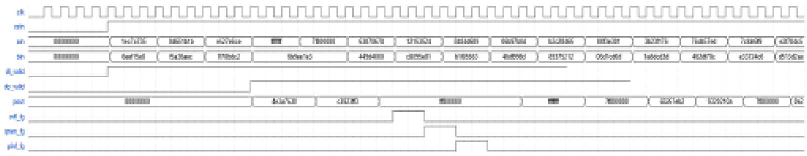
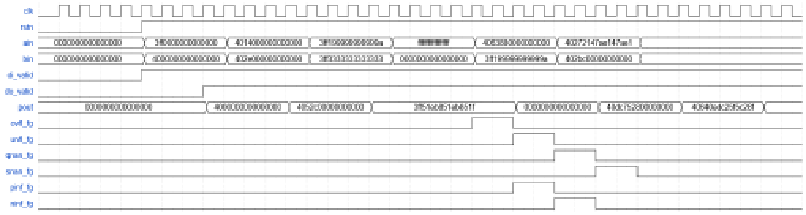


Figure 4-15. Floating-Point Multiplication (64-bit)

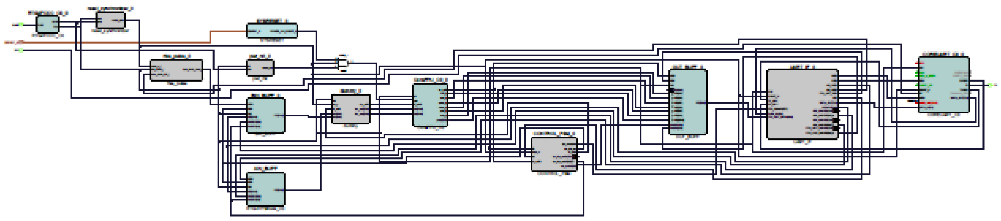


System Integration

The following figure shows an example of using the core. In this example, the design UART is used as a communication channel between the design and the host PC. The signals ain and bin (each of 32-bit or 64-bit width) are the inputs to the design from UART. After the CoreFPU receives the di_valid signal, it computes the result. After computing the result, the do_valid signal goes high and stores the result (aout/pout data) in the output buffer. This same procedure is applicable for conversion and arithmetic operations. For conversion operations, only input ain is sufficient whereas for arithmetic operations, both ain and bin inputs are required. Output aout is enabled for conversion operations and pout port is enabled for arithmetic operations.

Figure 4-16. Example of the CoreFPU System

Figure 4-16. Example of the CoreFPU System



1. Synthesis (Ask a Question)

   To run synthesis on the CoreFPU, set the design root to the IP component instance and from the Libero design flow pane, run the Synthesis tool.

   Place and Route (Ask a Question)

   After the design is synthesized, run the Place-and-Route tool. CoreFPU requires no special placeand- route settings.

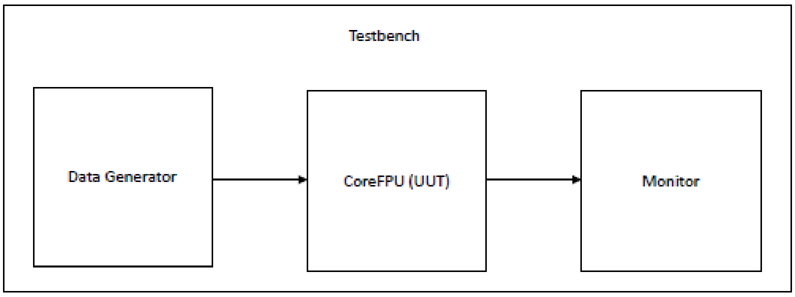
2. User Testbench (Ask a Question)

   A user testbench is provided with the CoreFPU IP release. Using this testbench, you can verify functional behavior of CoreFPU.

A simplified block diagram of the user testbench is shown in the following figure. The user testbench instantiates the Configured CoreFPU design (UUT), and includes behavioral test data generator, necessary clock, and reset signals.

Figure 4-17. CoreFPU User Testbench

Figure 4-17. CoreFPU User Testbench



Important: You have to monitor the output signals in ModelSim simulator, see Simulation section.

Additional References (Ask a Question)

This section provides a list for additional information.

For updates and additional information about the software, devices, and hardware, visit the

Intellectual Property pages on the Microchip FPGAs and PLDs website.

1. Known Issues and Workarounds (Ask a Question)

   There are no known issues and workarounds for CoreFPU v3.0.
2. Discontinued Features and Devices (Ask a Question)

   There are no discontinued features and devices with this IP release.

Glossary

The following are the list of terms and definitions used in the document.

Table 6-1. Terms and Definitions

Term	Definition
FPU	Floating Point Unit
FP ADD	Floating-Point Addition
FP SUB	Floating-Point Subtraction
FP MULT	Floating-Point Multiplication

Resolved Issues

The following table lists all the resolved issues for the various CoreFPU releases.

Table 7-1. Resolved Issues

Release	Description
3.0	The following is the list of all resolved issues in the v3.0 release: Case Number: 01420387 and 01422128 Added the rounding scheme logic (round to the nearest even number).
2.1	The following is the list of all resolved issues in the v2.1 release: The design encounters issues due to the presence of duplicate modules when multiple cores are instantiated. Renaming the CoreFPU IP instance results in an “Undefined module” error.
1.0	Initial Release

Device Resource Utilization and Performance

The CoreFPU macro is implemented in the families listed in the following table.

Table 8-1. FPU PolarFire Unit Device Utilization for 32-Bit

FPGA Resources					Utilization			
Family	4LUT	DFF	Total	Math Block	Device	Percentage	Performance	Latency
Fixed-Point to Floating-Point								
PolarFire®	260	104	364	0	MPF300T	0.12	310 MHz	3
Floating-Point to Fixed-Point								
PolarFire	591	102	693	0	MPF300T	0.23	160 MHz	3
Floating-Point Addition								
PolarFire	1575	1551	3126	0	MPF300T	1.06	340 MHz	16
Floating-Point Subtraction								
PolarFire	1561	1549	3110	0	MPF300T	1.04	345 MHz	16
Floating-Point Multiplication								
PolarFire	465	847	1312	4	MPF300T	0.44	385 MHz	14

FPGA Resources					Utilization			
Family	4LUT	DFF	Total	Math Block	Device	Percentage	Performance	Latency
Fixed-Point to Floating-Point								

RTG4™	264	104	368	0	RT4G150	0.24	160 MHz	3
Floating-Point to Fixed-Point								
RTG4	439	112	551	0	RT4G150	0.36	105 MHz	3
Floating-Point Addition								
RTG4	1733	1551	3284	0	RT4G150	1.16	195 MHz	16
Floating-Point Subtraction								
RTG4	1729	1549	3258	0	RT4G150	1.16	190 MHz	16
Floating-Point Multiplication								
RTG4	468	847	1315	4	RT4G150	0.87	175 MHz	14

FPGA Resources					Utilization			
Family	4LUT	DFF	Total	Math Block	Device	Percentage	Performance	Latency
Fixed-Point to Floating-Point								
PolarFire®	638	201	849	0	MPF300T	0.28	305 MHz	3
Floating-Point to Fixed-Point								
PolarFire	2442	203	2645	0	MPF300T	0.89	110 MHz	3

Floating-Point Addition								
PolarFire	5144	4028	9172	0	MPF300T	3.06	240 MHz	16
Floating-Point Subtraction								
PolarFire	5153	4026	9179	0	MPF300T	3.06	250 MHz	16
Floating-Point Multiplication								
PolarFire	1161	3818	4979	16	MPF300T	1.66	340 MHz	27

FPGA Resources					Utilization			
Family	4LUT	DFF	Total	Math Block	Device	Percentage	Performance	Latency
Fixed-Point to Floating-Point								
RTG4™	621	201	822	0	RT4G150	0.54	140 MHz	3
Floating-Point to Fixed-Point								
RTG4	1114	203	1215	0	RT4G150	0.86	75 MHz	3
Floating-Point Addition								
RTG4	4941	4028	8969	0	RT4G150	5.9	140 MHz	16
Floating-Point Subtraction								

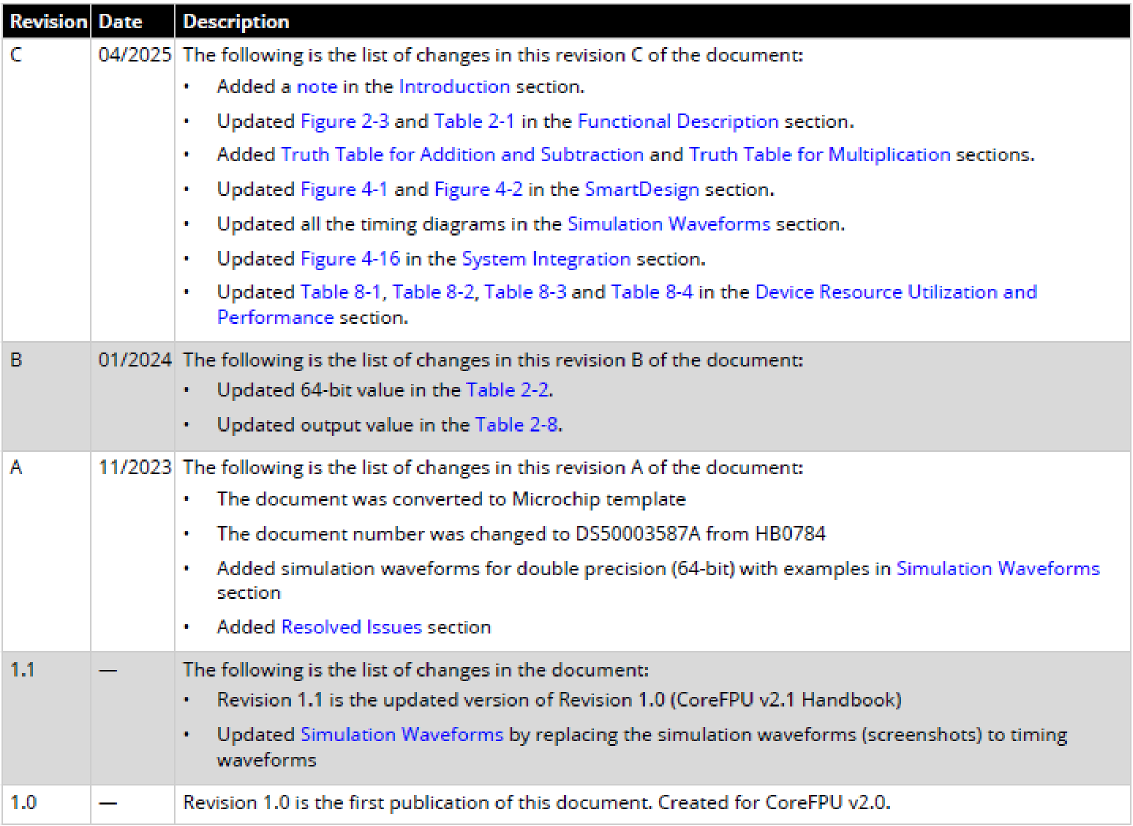
RTG4	5190	4026	9216	0	RT4G150	6.07	130 MHz	16
Floating-Point Multiplication								
RTG4	1165	3818	4983	16	RT4G150	3.28	170 MHz	27

Important: To increase the frequency, select Enable retiming option in synthesis setting.

Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Table 9-1. Revision History



Revision	Date	Description
C	04/2025	The following is the list of changes in this revision C of the document: • Added a note in the Introduction section. • Updated Figure 2-3 and Table 2-1 in the Functional Description section. • Added Truth Table for Addition and Subtraction and Truth Table for Multiplication sections. • Updated Figure 4-1 and Figure 4-2 in the SmartDesign section. • Updated all the timing diagrams in the Simulation Waveforms section. • Updated Figure 4-16 in the System Integration section. • Updated Table 8-1, Table 8-2, Table 8-3 and Table 8-4 in the Device Resource Utilization and Performance section.
B	01/2024	The following is the list of changes in this revision B of the document: • Updated 64-bit value in the Table 2-2. • Updated output value in the Table 2-8.
A	11/2023	The following is the list of changes in this revision A of the document: • The document was converted to Microchip template • The document number was changed to DS50003587A from HB0784 • Added simulation waveforms for double precision (64-bit) with examples in Simulation Waveforms section • Added Resolved Issues section
1.1	—	The following is the list of changes in the document: • Revision 1.1 is the updated version of Revision 1.0 (CoreFPU v2.1 Handbook) • Updated Simulation Waveforms by replacing the simulation waveforms (screenshots) to timing waveforms
1.0	—	Revision 1.0 is the first publication of this document. Created for CoreFPU v2.0.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call 800.262.1060
- From the rest of the world, call 650.318.4460
- Fax, from anywhere in the world, 650.318.8044

Microchip Information

Trademarks

ISBN: 979-8-3371-0947-3

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF

NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

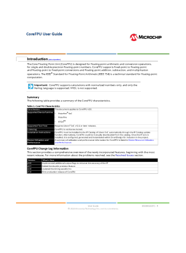
Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Documents / Resources

	MICROCHIP CoreFPU Core Floating Point Unit [pdf] User Guide v3.0, v2.1, v2.0, v1.0, CoreFPU Core Floating Point Unit, Core Floating P oint Unit, Floating Point Unit, Point Unit

References

- User Manual

MICROCHIP

Core Floating Point Unit, CoreFPU Core Floating Point Unit, Floating Point Unit, MICROCHIP, Point Unit, V1.0, v2.0, V2.1, V3.0

Leave a comment

Your email address will not be published. Required fields are marked *

Comment *

Name

Email

Website

□ Save my name, email, and website in this browser for the next time I comment.

Post Comment

Search:

e.g. whirlpool wrf535swhz

Search

Manuals+ | Upload | Deep Search | Privacy Policy | @manuals.plus | YouTube

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.