

HB0816
Handbook
CoreFPU v2.0



a  **MICROCHIP** company



a  MICROCHIP company

Microsemi Headquarters

One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113

Outside the USA: +1 (949) 380-6100

Sales: +1 (949) 380-6136

Fax: +1 (949) 215-4996

Email: sales.support@microsemi.com

www.microsemi.com

©2020 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

1	Revision History	1
1.1	Revision 1.0	1
2	Introduction	2
2.1	Overview	2
2.2	Features	2
2.3	Core Version	2
2.4	Supported Families	2
2.5	Device Utilization and Performance	3
3	Functional Description	4
3.1	Fixed Point to Floating Point (Conversion)	5
3.2	Floating Point to Fixed Point (Conversion)	5
3.3	Floating Point Addition (Arithmetic Operation)	6
3.4	Floating Point subtraction (Arithmetic Operation)	6
3.5	Floating Point Multiplication (Arithmetic Operation)	7
4	Configuration	8
4.1	FPU Configurable Options	8
4.2	Ports	9
5	Tool Flow	10
5.1	License	10
5.2	SmartDesign	10
5.2.1	Fixed Point to Floating Point Conversion	11
5.2.2	Floating Point to Fixed Point	12
5.2.3	Floating Point Addition/Subtraction/Multiplication	13
5.3	Simulation	13
5.4	Simulation Waveforms	14
5.5	Synthesis	15
5.6	Place-and-Route	15
6	Testbench	16
7	System Integration	17

Figures

Figure 1	32-bit Frame	4
Figure 2	CoreFPU Block Diagram	5
Figure 3	SmartDesign CoreFPU Instance for Arithmetic Operation	10
Figure 4	SmartDesign CoreFPU Instance for Conversion Operation	10
Figure 5	FPU Configurator Fixed to Floating Point	11
Figure 6	FPU Configurator Float to Fix Point	12
Figure 7	FPU Configurator Float Point Subtraction	13
Figure 8	Fixed to Floating Point Conversion	14
Figure 9	Floating Point to Fixed Point Conversion	14
Figure 10	Floating Point Addition	14
Figure 11	Floating Point Subtraction	15
Figure 12	Floating Point Multiplication	15
Figure 13	CoreFPU User Testbench	16
Figure 14	Example of the CoreFPU System	17

Tables

Table 1	FPU PolarFire Unit Device Utilizations	3
Table 2	FPU RTG4 Unit Device Utilizations	3
Table 3	FPU Configurable Options	8
Table 4	Port Description	9

1 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.

1.1 Revision 1.0

Revision 1.0 is the first publication of this document. Created for CoreFPU v2.0.

2 Introduction

2.1 Overview

The Core Floating Point Unit (CoreFPU) is designed for floating-point arithmetic and conversion operations for single 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.

2.2 Features

1. Supports single precision floating numbers as per IEEE-754 standard.
2. Conversions
 - Fixed Point to Floating Point Conversion
 - Floating Point to Fixed Point Conversion
3. Arithmetic Operations
 - Floating Point Addition
 - Floating Point Subtraction
 - Floating point Multiplication
4. Provides flags for Overflow, Infinity, and Not-a-Number (NaN) for floating point numbers.
5. Fully pipelined implementation of arithmetic operations.
6. Provision to configure the core for design requirements.

2.3 Core Version

This handbook is for CoreFPU version 2.0.

2.4 Supported Families

- PolarFire SoC
- PolarFire
- RTG4

2.5 Device Utilization and Performance

Table 1 • FPU PolarFire Unit Device Utilizations

Family	FPGA Resources				Utilization		Performance	Latency
	4LUT	DFF	Total	Math Block	Device	Percentage		
Fixed Point to Floating Point								
PolarFire	290	115	405	0	MPF300T	0.14	330 MHz	2
Floating Point to Fixed Point								
PolarFire	799	117	916	0	MPF300T	0.31	257 MHz	2
Floating Point Addition								
PolarFire	937	695	1632	2	MPF300T	0.54	269 MHz	6
Floating Point Subtraction								
PolarFire	943	694	1637	2	MPF300T	0.4	250 MHz	6
Floating Point Multiplication								
PolarFire	318	793	1111	4	MPF300T	0.37	380 MHz	7

Table 2 • FPU RTG4 Unit Device Utilizations

Family	FPGA Resources				Utilization		Performance	Latency
	4LUT	DFF	Total	Math Block	Device	Percentage		
Fixed Point to Floating Point								
RTG4	301	109	410	0	RTG4150	0.38	305 MHz	2
Floating Point to Fixed Point								
RTG4	410	112	522	0	RTG4150	0.34	195 MHz	2
Floating Point Addition								
RTG4	881	795	1676	2	RTG4150	1.11	245 MHz	6
Floating Point Subtraction								
RTG4	881	688	1569	2	RTG4150	1.03	160 MHz	6
Floating Point Multiplication								
RTG4	362	858	1220	4	RTG4150	0.80	189 MHz	7

Note: Data in the preceding tables are obtained using typical synthesis settings. The design frequency (MHz) was set to 500.

3 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 number's radix point (decimal point or more commonly binary point) that can be 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 $F \times r^E$. Decimal numbers use radix of 10 ($F \times 10^E$); while binary numbers use radix of 2 ($F \times 2^E$).

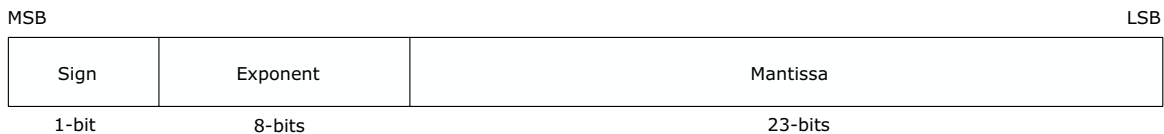
Representation of the floating point number is not unique. For example, the number 55.66 can be represented as 5.566×10^1 , 0.5566×10^2 , 0.05566×10^3 , and so on. The fractional part can be normalized. In the normalized form, there is only a single non-zero digit before the radix point. For example, decimal number 123.4567 can be normalized as 1.234567×10^2 ; binary number 1010.1011B can be normalized as $1.0101011B \times 2^3$.

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 infinite number of real numbers (even within a small range from 0.0 to 0.1). On the other hand, a n-bit binary pattern represents a finite 2^n distinct numbers. Hence, not all the real numbers can be represented. The nearest approximation will be 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)

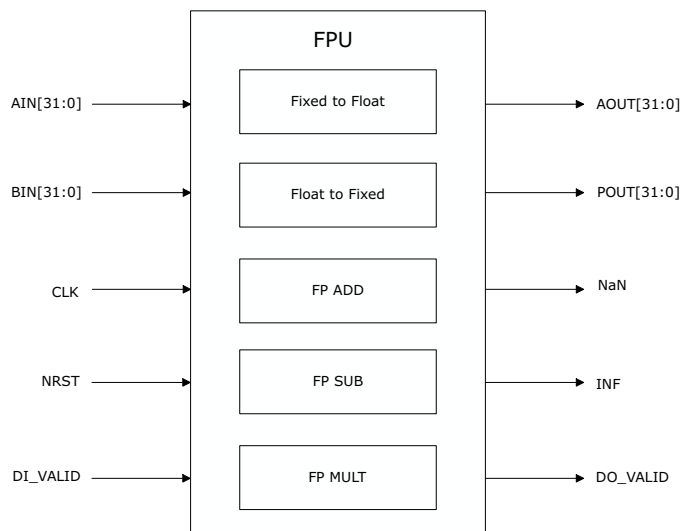
Figure 1 • 32-bit Frame



The CoreFPU is the top level integration of the two conversion modules (Fixed to Float and Float to Fixed) and three arithmetic operation (FP ADD, FP SUB, and FP MULT). User can configure any one of the operation 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 port:

Figure 2 • CoreFPU Block Diagram



3.1 Fixed Point to Floating Point (Conversion)

CoreFPU configured as **Fixed to Float** infers the Fixed point to Floating 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 width 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 precision Floating point format. The output latency for the fixed to floating-point conversion is four clock cycles.

Example:

Fixed-point Number

AIN(32-bit)	Integer (17-bit)	Fraction(15- bit)
0x12153524	00010010000101010	011010100100100

Floating-point Number

AOUT	Sign	Exponent	Mantissa
4610a9a9	0	10001100	00100001010100110101001

3.2 Floating Point to Fixed Point (Conversion)

CoreFPU configured as **Float to Fixed** infers the Floating point to Fixed-point conversion module. The input (AIN) to CoreFPU can be any single 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. The output latency for the floating point to fixed-point conversion is four clock cycles..

Example:

Floating-point Number

AIN	Sign	Exponent	Mantissa
41bd6783	0	10000011	01111010110011110000011

Fixed-Point Number

AOUT(32-bit)	Integer (17-bit)	Fraction (15-bit)
--------------	------------------	-------------------

0x000bd678 00000000000010111 101011001111000

3.3 Floating Point Addition (Arithmetic Operation)

CoreFPU configured as **FP ADD** infers the Float 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 precision floating point numbers. The input is valid on DI_VALID signal and output is valid on DO_VALID. The output latency for the floating-point addition is eight clock cycles. The core outputs overflow (OVFL), Not-a-Number (NaN), and Infinity (INF) Flags based on the addition operation.

Example:

Floating-point input 1

AIN	Sign	Exponent	Mantissa
4e989680	0	10011101	00110001001011010000000

Floating-point input 2

BIN	Sign	Exponent	Mantissa
4f191b40	0	10011110	00110010001101101000000

Floating-point addition output

POUT	Sign	Exponent	Mantissa
4f656680	0	10011110	11001010110011010000000

3.4 Floating Point subtraction (Arithmetic Operation)

CoreFPU configured as **FP SUB** infers the Float point addition 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 precision floating point numbers. The input is valid on DI_VALID signal and output is valid on DO_VALID. The output latency for the floating-point subtraction is eight clock cycles. The core outputs overflow (OVFL), Not-a-Number (NaN), and Infinity (INF) Flags based the subtraction operation.

Example:

Floating-point input 1

AIN	Sign	Exponent	Mantissa
ac85465f	1	01011001	00001010100011001011111

Floating-point input 2

BIN	Sign	Exponent	Mantissa
2f516779	0	01011110	10100010110011101111001

Floating-point subtraction output

POUT	Sign	Exponent	Mantissa
af5591ab	1	01011110	10101011001000110101011

3.5 Floating Point Multiplication (Arithmetic Operation)

CoreFPU configured as **FP MULT** infers the Float point addition 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 precision floating point numbers. The input is valid on DI_VALID signal and output is valid on DO_VALID. The output latency for the floating-point multiplication is eight clock cycles. The core outputs overflow (OVFL), Not-a-Number (NaN), and Infinity (INF) Flags based the multiplication operation.

Example:

Floating-point input 1

AIN	Sign	Exponent	Mantissa
1ec7a735	0	00111101	10001111010011100110101

Floating-point input 2

BIN	Sign	Exponent	Mantissa
6ecf15e8	0	11011101	10011110001010111101000

Floating-point Multiplication output

POUT	Sign	Exponent	Mantissa
4e3a7630	0	10011100	01110100111011000110000

4 Configuration

4.1 FPU Configurable Options

There are numbers 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 can be used to select appropriate values for the configurable option.

Table 3 • FPU Configurable Options

Name	Default	Description
Conversion Type	Fixed Point to Floating Point	Select the operation as required. <ul style="list-style-type: none"> • Fixed Point to Floating Point Conversion • Floating Point to Fixed Point Conversion • Floating Point Addition • Floating Point Subtraction • Floating Point Multiplication
INPUT FRACTION WIDTH ¹	15	Configures the fractional point in the Input AIN and BIN signals. Valid range is 3-1
OUTPUT FRACTION WIDTH ²	15	Configures the fractional point in the Output AOUT signals. Valid range is 23-1

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.

4.2 Ports

Table 4 • Port Description

Name	WIDTH	Type	Description
CLK	1	Input	Main System clock.
NRST	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] and BIN[31:0] is valid.
AIN	32	Input	A Input Bus (It is used for all operations).
BIN ¹	32	Input	B Input Bus (It is used for arithmetic operations only).
AOUT ²	32	Output	Output value when fixed to Floating point or Floating to fixed point conversion operations are selected.
POUT ¹	32	Output	Output value when Addition, Subtraction or Multiplication operations are selected.
DO_VALID	1	Output	Active High Signal. This signal indicates that the data present on POUT/AOUT data bus is valid.
OVFL ³	1	Output	Active high signal. This signal indicates the overflow during Floating Point operations.
NaN ³	1	Output	Active high signal. This signal indicates the Not-a-Number (NaN) during Floating Point operations.
INF ³	1	Output	Active high signal. This signal indicates the Infinity during Floating Point operations.

1. This port is available only for Floating Point Addition/Subtraction/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.

Note: AIN width (32-bit) = Integer width (32 - Fractional width) + input Fractional width (valid range 31 to 1)
 POUT width (32-bit) = Integer width (32 - Fractional width) + output Fractional width (valid range 23 to 1)

5 Tool Flow

5.1 License

CoreFPU is not license locked.

5.2 SmartDesign

CoreFPU is available for download in the Libero IP catalog through the web repository. Once it is listed in the catalog, the core can be instantiated using the SmartDesign flow. For information on using SmartDesign to configure, connect, and generate cores, refer to the [Libero SoC online help](#).

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

Figure 3 • SmartDesign CoreFPU Instance for Arithmetic Operation

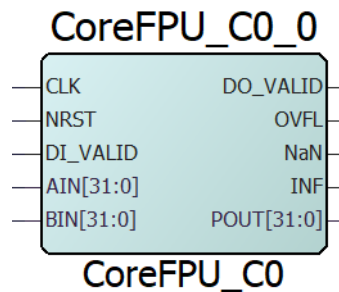
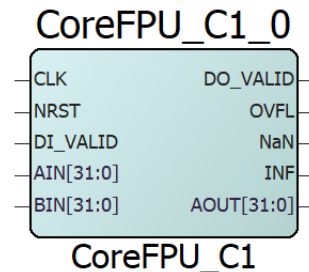


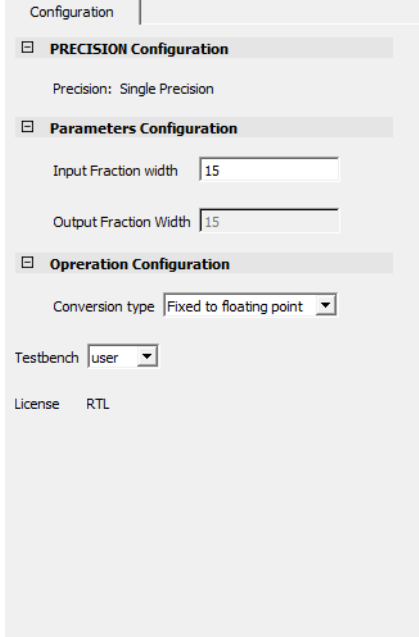
Figure 4 • SmartDesign CoreFPU Instance for Conversion Operation



5.2.1 Fixed Point to Floating Point Conversion

During this operation, the INPUT FRACTION WIDTH is configurable. The OUTPUT WIDTH is by default set to 32-bits for Single Precision Floating point.

Figure 5 • FPU Configurator Fixed to Floating Point



The screenshot displays the 'Configuration' window of the FPU Configurator. It is organized into three main sections, each with a collapse icon (a square with a minus sign) to its left:

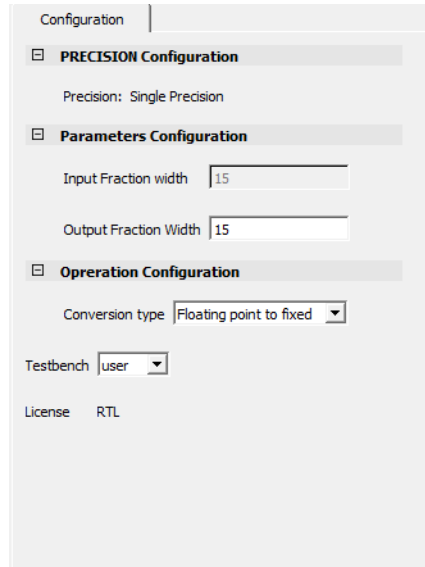
- PRECISION Configuration:** Shows 'Precision: Single Precision'.
- Parameters Configuration:** Contains two input fields: 'Input Fraction width' with the value '15' and 'Output Fraction Width' with the value '15'.
- Operation Configuration:** Features a dropdown menu for 'Conversion type' set to 'Fixed to floating point'.

Below these sections, there is a 'Testbench' dropdown menu set to 'user' and a 'License' field set to 'RTL'.

5.2.2 Floating Point to Fixed Point

During this operation, the OUTPUT FRACTIONAL WIDTH is configurable and the INPUT WIDTH is by default set to 32-bit for single precision floating point. Here $\text{OUTPUT INTEGER WIDTH} = \text{OUTPUT WIDTH} - \text{OUTPUT FRACTIONAL WIDTH}$.

Figure 6 • FPU Configurator Float to Fix Point



The screenshot shows the 'Configuration' window of the FPU Configurator. It is divided into three main sections:

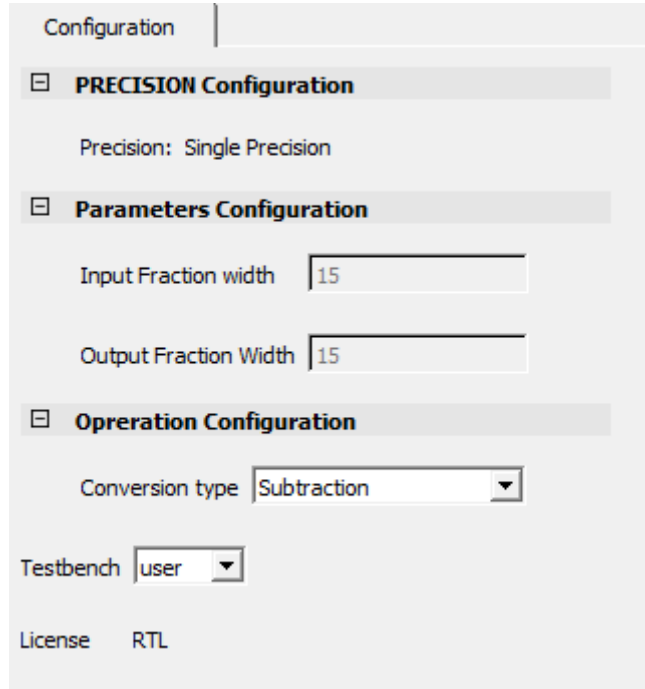
- PRECISION Configuration:** Precision is set to 'Single Precision'.
- Parameters Configuration:** 'Input Fraction width' is set to 15, and 'Output Fraction Width' is also set to 15.
- Operation Configuration:** 'Conversion type' is set to 'Floating point to fixed' (indicated by a dropdown arrow), and 'Testbench' is set to 'user' (also indicated by a dropdown arrow).

At the bottom, the 'License' is set to 'RTL'.

5.2.3 Floating Point Addition/Subtraction/Multiplication

During these operations, the INPUT and OUTPUT WIDTHS are not configurable as these are floating point arithmetic operations and the INPUT/OUTPUT WIDTH is by default set to 32-bit single precision floating point.

Figure 7 • FPU Configurator Float Point Subtraction



The screenshot shows the 'Configuration' window of the FPU Configurator. It is divided into three main sections:

- PRECISION Configuration:** Precision: Single Precision
- Parameters Configuration:** Input Fraction width: 15, Output Fraction Width: 15
- Operation Configuration:** Conversion type: Subtraction (selected in a dropdown menu)

At the bottom, there is a 'Testbench' dropdown menu set to 'user' and a 'License' field set to 'RTL'.

5.3 Simulation

To run simulations, select the user testbench in the core configuration window. After generating the CoreFPU, the pre-synthesis test-bench hardware description language (HDL) files are installed in Libero.

5.4 Simulation Waveforms

Figure 8 • Fixed to Floating Point Conversion

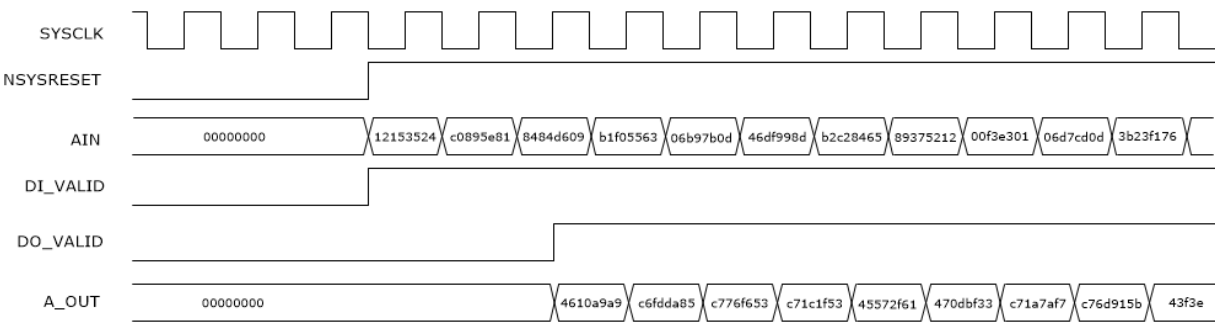


Figure 9 • Floating Point to Fixed Point Conversion

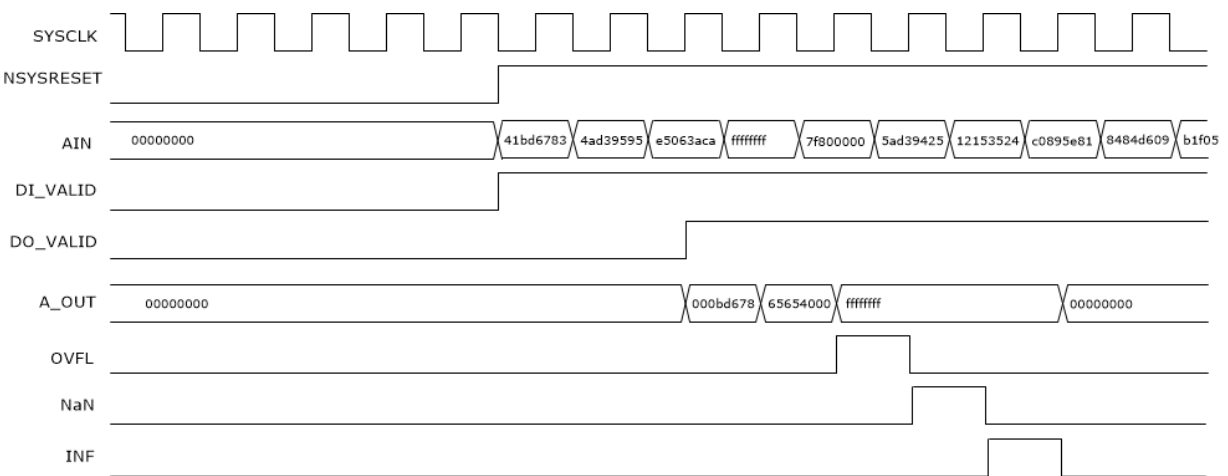


Figure 10 • Floating Point Addition

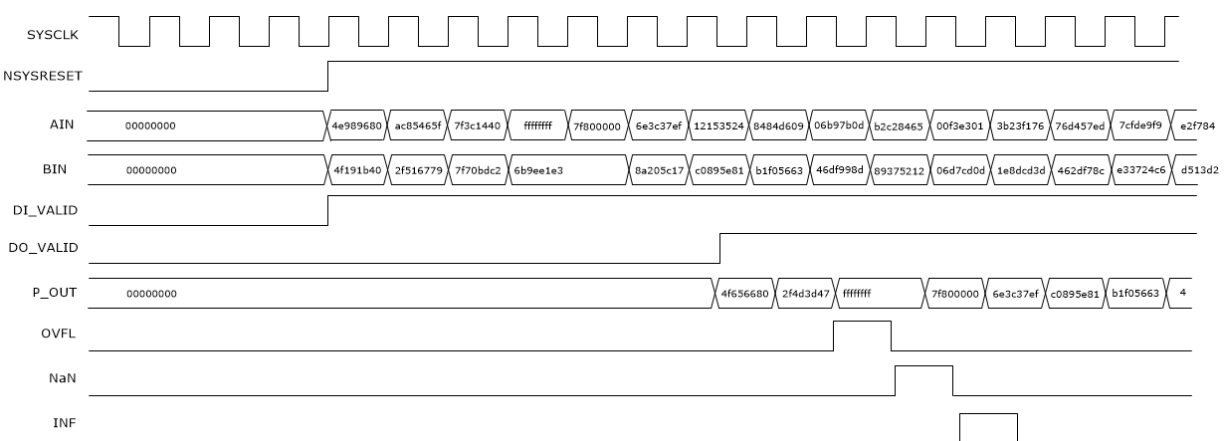


Figure 11 • Floating Point Subtraction

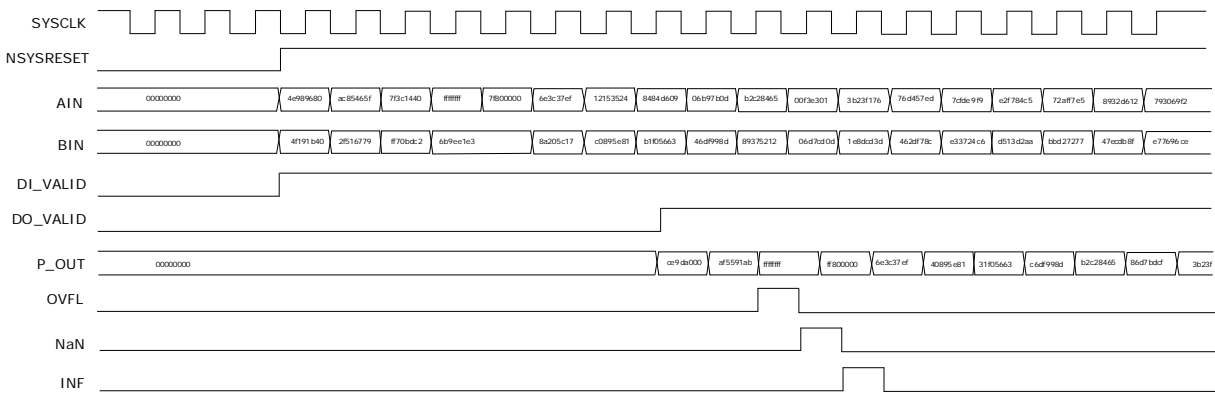
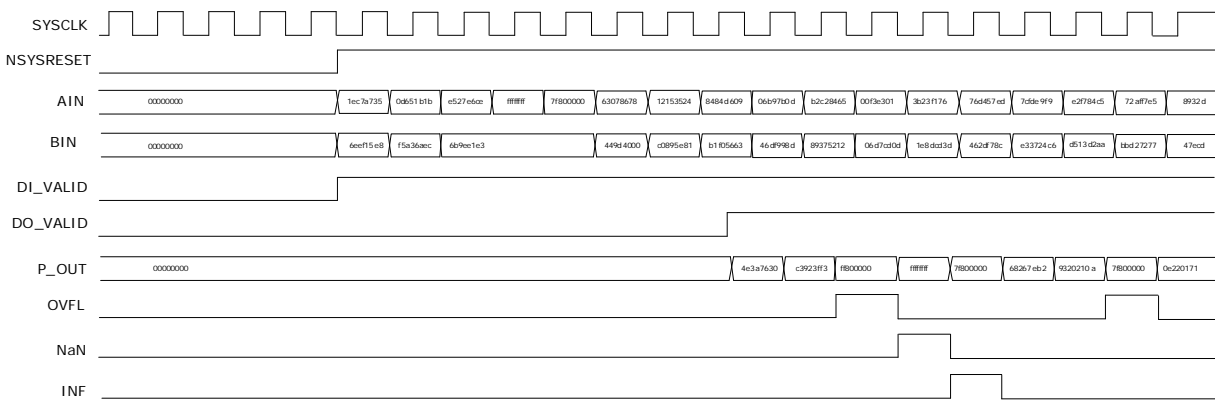


Figure 12 • Floating Point Multiplication



5.5 Synthesis

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

5.6 Place-and-Route

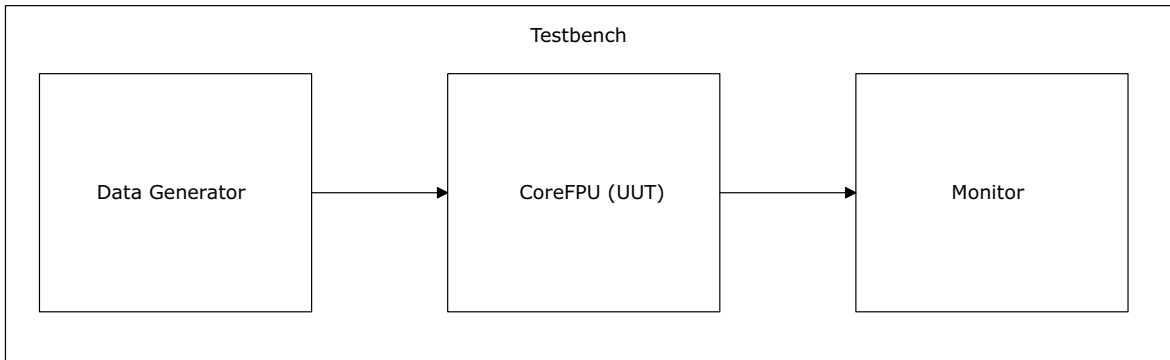
After the design is synthesized, run the place-and-route from the tools. CoreFPU requires no special place-and-route settings.

6 Testbench

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

A simplified block diagram of the user testbench is shown in [Figure 13](#). The user testbench instantiates the configured CoreFPU design (UUT), and includes behavioral test data generator, necessary clock and reset signals.

Figure 13 • CoreFPU User Testbench



Note: The user has to monitor the output signals in ModelSim simulator, refer to: [Simulation](#) section.

7 System Integration

The following figure shows an example of using the core. In this example, design UART is used as the communication channel between design and host PC. Both AIN and BIN (each of 32-bit width) are the inputs to the design from UART. After the CoreFPU receives the DI_valid signal, coreFPU computes the result, after computing the result, the DO_valid signal is high and stores the result AOUT/POUT data in output buffer. Same procedure is applicable for conversion and arithmetic operations. For conversion operations only input AIN is sufficient. Whereas, for arithmetic operations AIN and BIN both inputs are required. Output AOUT is enabled for conversion operations and POUT port is enabled for arithmetic operations.

Figure 14 • Example of the CoreFPU System

