



VLSI Design & Consultancy

DATASHEET

**Floating Point Multiplier Core
Version 1.0**

1 Overview

The Think-Silicon *Floating Point Multiplier* is a standard IEEE 754 compliant, single precision floating point multiplier. The units support proper round to nearest and Denormalised numbers.

2 Floating Point Numbers

The format of the IEEE 754 single precision numbers is shown in Table 2-1:

Table 2-1 IEEE 754 single precision floating point numbers format

BITS	NAME	DESCRIPTION
31	sign	This is the sign bit. 0 signifies a positive number and a 1 a negative number
30:23	exponent	This number defines the power of 2 that is multiplied by the fractional part. It also defines if the number is normal.
22:0	fraction	These are fractional bits of the number.

The number is defined as follows:

$$number = -1^{sign} \times 2^{(127 - exponent)} \times 1.(fraction)$$

There are also special bit patterns that define +ve and -ve ∞ (infinity) as well as positive and negative ∅ (zero). Those patterns as are also supported and handled according to the standard.

3 Features

- IEEE-754 single precision floating point compliance
- Denormalized numbers support
- Rounding to the nearest

4 Architecture

4.1 Block Diagram/Port Port Diagram

The generated *fp_mult* module Block Diagram is shown in Figure 4-1. The IP module has two input ports for the multiplier and the multiplicand and one output port for the product. The IP module is fully combinatorial.

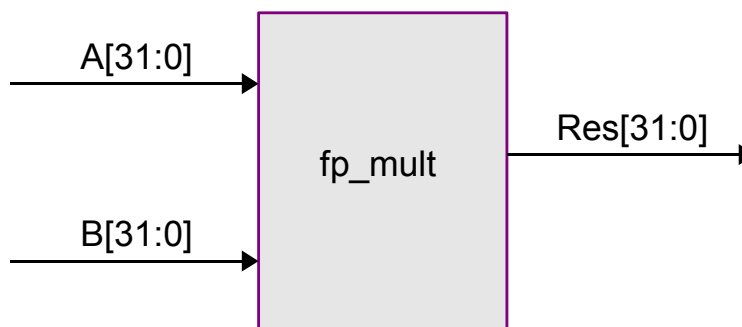


Figure 4-1 fpmult Block Diagram/Port Diagram

The 32-bit *A* and *B* input ports comply the IEEE-754 single precision floating point numbers standards. The

output port. *Res*, has the result of the multiplication and it is also encoded to comply IEEE-754 compliant single precision standard.

4.2 Port Interface

The ports of *fp_mult* module are listed in Table 4-1.

Table 4-1 *fp_mult* Port Interface

PORT	TYPE	DESCRIPTION
A[31:0]	Input	The multiplier
B[31:0]	Input	The multiplicand
RES	Output	The product of multiplication

5 Interfacing fp_mult

The design is fully combinatorial, thus should you require to pipeline the unit, you are advised to do so at the synthesis level, using a balanced registers technique that will provide better results for your target technology.

6 Generator Usage

The *Floating Point Multiplier* generator employs a graphical web user interface (GUI) for configuring and generating the *fp_mult* module. In order to use the GUI you must sign-in Think Silicon Ltd web site. If already registered, click on *Sign-in* link in the upper, right side of the web page. Otherwise click on the *Register* link first and follow the instructions. The generator GUI page is shown in Figure 6-1. Press the *Generate* button in order to generate the *fp_mult* module.

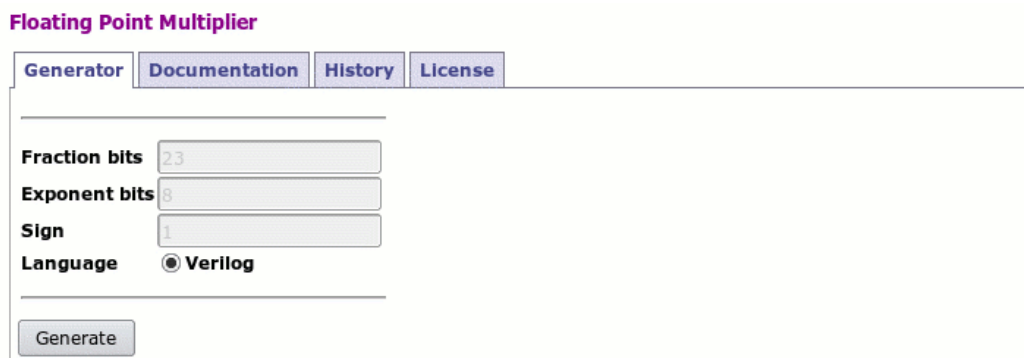


Figure 6-1 The *Floating Point Multiplier* GUI page

7 Deliverables

The package generated with *Floating Point Multiplier* consists of the present document and source code files in Verilog™¹ HDL and C language. The files are listed in Table 7-1.

Table 7-1 *Floating Point Multiplier* Deliverables

FILE	DESCRIPTION
./src/fp_mult.v	Floating Point Multiplier <i>fp_mult</i> module
./src/tb_vectors.v	Floating Point Multiplier testbench
./src/gen_vectors.c	Floating Point Multiplier Test Vectors generator

¹ Verilog is a trademark of Cadence Design Automation. (<http://www.cadence.com>)

FILE	DESCRIPTION
./src/Makefile	Makefile with simulation commands
./parameters.txt	Floating Point Multiplier configuration parameters
./doc/TSi_fp_multiplier.pdf	The present document

8 Verification

The Verification Flow Diagram of the *fp_mult* core is shown in Figure 8-1.

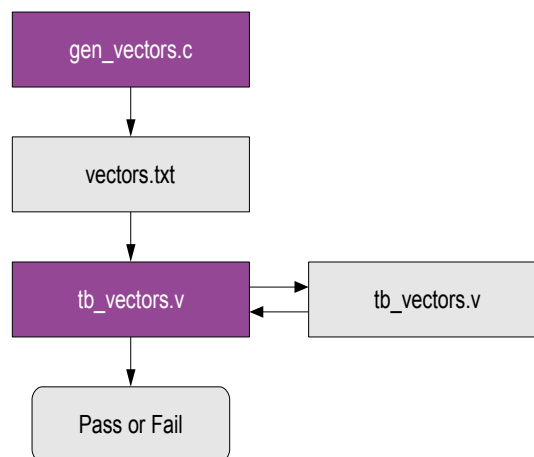


Figure 8-1 Verification Flow Diagram of the *fp_mult* core

The testbench is based on gcc and the Icarus Verilog simulator and we assume these are installed in your system. In order to generate test vectors and run the verification tests for the *fp_mult* core, enter the commands shown in Figure 7-2 in a command line shell.

```

> cd ./fp_mult_32
> make
  
```

Figure 8-2 Verification commands for *fp_mult* core

This should automatically compile the test vector generator and start the simulator. The simulator finishes with a pass or fail. The test vector generator and the netlist are written in standard C and Verilog^{TM 2}, so other compilers and Simulators, such as NCVerilog^{TM 3} and MTI Modelsim^{TM 4} can be used.

2 Verilog is a trademark of Cadence Design Automation, UK (<http://www.cadence.com>)
 3 NCVerilog is a trademark of Cadence Design Automation, UK (<http://www.cadence.com>)
 4 MTI Modelsim is a trademark of Mentor Graphics (<http://www.mentor.com>)

Contact:

Think Silicon Ltd
Suite B12
Patras Science Park
Rion Achaïas 26504
Greece

web: <http://www.think-silicon.com>
email: info@think-silicon.com
Tel: +30 2610 911543