

FLOATING POINT MULTIPLIER IP

Introduction:

Floating Point numbers represented in IEEE 754 format are used in most of the DSP Processors. Floating point arithmetic is useful in applications where a large dynamic range is required or in rapid prototyping applications where the required number range has not been thoroughly investigated.

The Floating Point Multiplier IP helps designers to perform floating point Multiplication on FPGA represented in IEEE 754 single precision floating point format.

Functional Description:

A Floating point multiplier is the most common element in most digital applications such as digital filters, digital signal processors, data processors and control units.

The present Floating Point Multiplier IP has three blocks sign calculator, exponent calculator, mantissa calculator, which works parallel and a normalization unit.

The Multiplier is pipelined, so the first result appears after the latency period and then the result can be obtained after every clock cycle.

Features:

- Available for wide range of FPGA families
- Pipelined
- Compliance with IEEE 754 standard
- Single precision real floating point support
- Supports Overflow, Underflow, Invalid operation flags
- Simple Interface
- Optimized for Speed and Latency
- Fully Synchronous design with single clock
- Compatible, Flexible and integrable with other modules

Description:

Figure 1 shows the Schematic Symbol of Floating Point Multiplier. It takes two IEEE 754 format single precision floating point numbers and produces the multiplied output. It also supports the features like underflow, overflow and invalid operations.

Figure 2 shows the implementation of Floating point Multiplier unit. The Unit consists of two stages, multiplication calculation and Normalization.

The first stage consists of the following three blocks which work in parallel.

Sign Calculator: The Output Sign is the exor of two sign bit inputs.

Exponent Calculator: The input exponents are added and the bias is removed to produce the exponent of Output.

Mantissa Calculator: Output Mantissa is calculated by multiplying the mantissa's of two inputs.

Second stage performs Normalization of the Output obtained from the first stage.

Normalization Block: The normalization is the last and most complicated part. This block is implemented in three pipelined stages.

This block first calculates how much amount the mantissa needs to be left shifted. The mantissa is processed in parallel in a number of modules, each looking at four bits of the mantissa.

The first module looks at first four bits of the mantissa and outputs the amount to be shifted assuming a one was found on these four bits. The second module operates on the next four bits of the mantissa treating first four bits are zero and outputs the amount to be shifted left.

Symbolic Diagram:

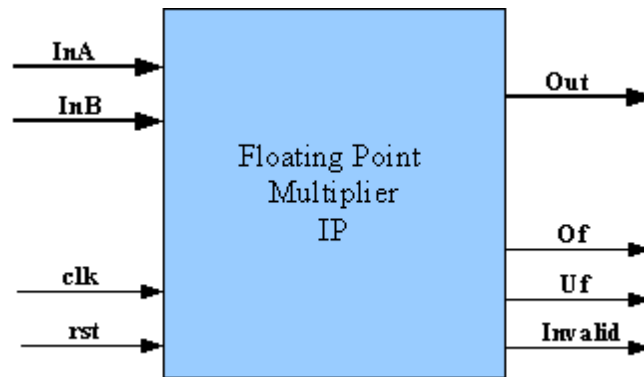


Figure1 : Schematic Block Diagram

Implementation:

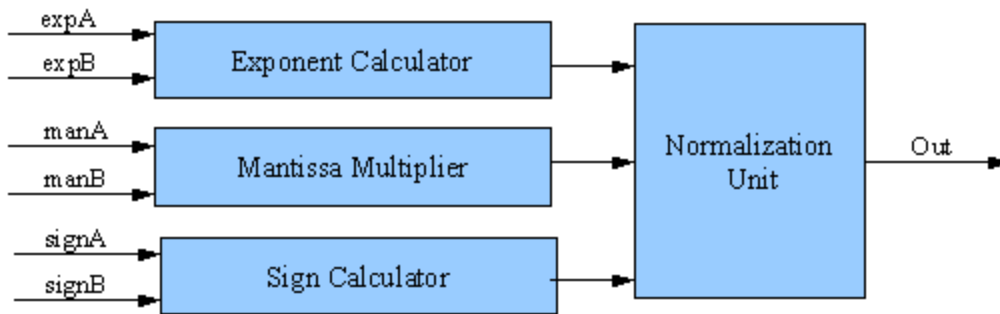


Figure2 : Implementation

Signal Table :

Signal	Direction	Data Width	Description
InA	IN	32	First Input
InB	IN	32	Second Input
clk	IN	1	Global Clock
rst	IN	1	Active low Reset
Out	OUT	32	Multiplier Output
Of	OUT	1	Overflow Flag
Uf	OUT	1	Underflow Flag
Invalid	OUT	1	Invalid Output Flag

Table 1: Signal Definition Table

This process is repeated for the remaining bits of mantissa. Signals are generated if the four bits of the mantissa are zero. Depending on the signal values the amount of shift is selected. This selection is implemented in three multiplexer stages.

Depending on the two leading bits of final mantissa, the final mantissa is shifted left by previously calculated shift amount or shifted right. The final exponent is also corrected accordingly.

Performance:

Device	Slice Count	Frequency (MHz)
Virtex-4 (XC4VLX25-FF676)	848	199.2
Virtex-5 (XC5VLX50-FF324)	872	207.04

Table 2: Performance table.

Verification:

The Floating point Multiplier IP has been verified with the following approaches:

- Exhaustive Functional/Timing simulation
- Results are compared with the C-code generated results and Behavioral model results
- Emulation on Xilinx FPGA board

Deliverables:

- Verilog Behavioral, RTL source code
- Test Benches
- C- code for generation of test vectors
- Detailed user documentation