ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

An Efficient Implementation of Floating Point Multiplier

P.Saritha
M.Tech Student,
Department of ECE,
KITS for Women's, kodad, T.S, India.

Ms.Ch.Nirmala
Associate Professor,
Department of ECE,
KITS for Women's, kodad, T.S, India.

ABSTRACT:

A floating-point unit (FPU) colloquially is a math coprocessor, which is a part of a computer system specially designed to carry out operations on floating point numbers [1]. Typical operations that are handled by FPU are addition, subtraction, multiplication and division. The aim was to build an efficient FPU that performs basic as well as transcendental functions with reduced complexity of the logic used reduced or at least comparable time bounds as those of x87 family at similar clock speed and reduced the memory requirement as far as possible. The functions performed are handling of Floating Point data, converting data to IEEE754 format, perform any one of the following arithmetic operations like addition, subtraction, multiplication, division and shift operation and transcendental operations like square Root, sine of an angle and cosine of an angle. All the above algorithms have been clocked and evaluated under Spartan 3E Synthesis environment. All the functions are built by possible efficient algorithms with several changes incorporated at our end as far as the scope permitted. Consequently all of the unit functions are unique in certain aspects and given the right environment(in terms of higher memory or say clock speed or data width better than the FPGA Spartan 3E Synthesizing environment) these functions will tend to show comparable efficiency and speed ,and if pipelined then higher throughput.

1. INTRODUCTION:

Floating-point units (FPU) colloquially are a math coprocessor which is designed specially to carry out operations on floating point numbers [1]. Typically FPUs can handle operations like addition, subtraction, multiplication and division. FPUs can also perform various transcendental functions such as exponential or trigonometric calculations, though these are done with software library routines in most modern processors. Our FPU is basically a single precision IEEE754 compliant integrated unit. In this chapter we have basically introduced the basic concept of what an FPU is, in the section 1.2. Following the section we have given a brief introduction to the IEEE 754 standards in section 1.3. After describing the IEEE 754 standards, we have explained the motivation and objective behind this project in section 1.4. And finally the section 1.5 contains the summary of the chapter .

FLOATING POINT UNIT:

When a CPU executes a program that is calling for a floating-point (FP) operation, there are three ways by which it can carry out the operation. Firstly, it may call a floating-point unit emulator, which is a floating-point library, using a series of simple fixed-point arithmetic operations which can run on the integer ALU. These emulators can save the added hardware cost of a FPU but are significantly slow. Secondly, it may use an add-on FPUs that are entirely separate from the CPU, and are typically sold as an optional add-ons which are purchased only when they are needed to speed up math-intensive operations. Else it may use integrated FPU present in the system [2].The FPU designed by us is a single precision IEEE754 compliant integrated unit. It can handle not only basic floating point operations like addition, subtraction, multiplication and division but can also handle operations like shifting, square root determination and other transcendental functions like sine, cosine and tangential function.

2. METHODOLOGY:

Our Floating Point Unit is a single precision IEEE754 compliant integrated unit. It incorporates various basic operations like addition, subtraction,

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

multiplication, division, shifting and other transcendental functions like square root determination and trigonometric operations like sine, cosine and tangential value evaluation.In this chapter, the section 2.2 gives a brief about the literature review and the details of the related work in the field of developing an efficient FPU. Section 2.3 gives a brief description about the features implemented in our FPU like the rounding modes it handles, the operations it can carry out, the exceptions it can handle etc. After this section we have section 2.4 which describes implementation in nutshell. This section describes a brief about the algorithms implemented by us. This chapter also describes the basic algorithm of our initial FPU model in the section 2.5. And lastly, the section 2.6 gives a summary of the chapter.

LITERATURE REVIEW:

When a CPU is executing a program that calls for a FP operation, a separate FPU is called to carry out

Module Name	Functionality
Cnvrt_2_integral_form	Converts 32 bit integral and 32 fractional part into single novel integral representation
cnvrt_2_ieee	Converts 32 bit binary to its equivalent IEEE-754 format
pre_normalization	Adjusts the operands by performing the necessary shifts before an add or subtract operation
add	Performs addition
sub	Performs subtraction
post_normalization	Normalizes the result of add/sub operation to its IEEE754 form
multiplication	Performs pre-normalization and multiplication of the operands intended to be multiplied and finally post-normalization of the result
Division	Performs pre-normalization and division of the operands intended to be divided, determines the remainder and finally post-normalization of the result
Squareroot determination	Evaluates the square root of the first operand op1_ieee
Shifting	Performs the shifting of the operand to the specified bit in specified direction
Cordic	Performs the trigonometric evaluation

Table 2.1: Modules and its Functionalities

the operation. So, the efficiency of the FPU is of great importance. Though, not many have had great achievements in this field, but the work by the following two are appreciable.

Open Floating Point Unit :

This was the open source project done by Rudolf Usselmann [6]. His FPU described a single precision floating point unit which could perform add, subtract, multiply, divide, and conversion between FP number and integer. It consists of two pre-normalization units that can adjust the mantissa as well as the exponents of the given numbers, one for addition/subtraction and the other for multiplication/division operations. It also has a shared post normalization unit that normalizes the fraction part. The final result after post-normalization is directed to a valid result which is in accordance to single precision FP format. The main drawback of this model was that most of the codes were written in MATLAB and due to this it is non-synthesizable.

GRFPU:

This high Performance IEEE754 FPU was designed at Gaisler Research for the improvement of FP operations of a LEON based systems [7]. It supports both single precision and double precision operands. It implements all FP operations defined by the IEEE754 standard in hardware. All operations are dealt with the exception of denormalized numbers which are flushed to zero and supports all rounding modes. This advanced design combines low latency and high throughput. The most common operations such as addition, subtraction and multiplication are fully pipelined which has throughput of one CC and a latency of three CC. More complex divide and square root operation takes between 1 to 24 CC to complete and execute in parallel with other FP operations. It can also perform operations like converse and compliment. It supports all SPARC V8 FP instructions. The main drawback of this model is that it is very expensive and complex to implement practically.

3. IMPLEMENTATION:

This document describes a single precision floating point unit. The floating point unit is fully IEEE 754 compliant. The design implemented here incorporates the following modules. Both the module name and its functionality have been specified in the table 2.1 in sequence of the manner they appear in the attached code.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

As our FPU works with floating point numbers, the operations, intermediate calculations and output are conventionally in the same floating point structure. But this invariably increases the complexity of calculation and the number of adjustments required at each level to obtain the correct result. Our proposal is to convert the floating point number into a simple yet quite precise integral representation and perform the calculations on the same, followed by the final conversion of the output into its expected floating point result format.The floating point data is inputted in two parts.

The first part is a 32 bit binary value of the integer part of the floating point operand and other is a 32 bit binary value of fractional part of the floating point operand. This is done because Verilog cannot deal with floating point numbers. So we need to consolidate the two parts (integral and fractional) of the operand into a single 32 bit effective operand. This is done by the following algorithm explained

4. DISCUSSION:

As our FPU is IEEE754 compliant, the next step is to convert the input (here the effective operand into the IEEE specified format.IEEE754 single precision can be encoded into 32 bits using 1 bit for the sign bit (the most significant i.e. 31st bit), next eight bits are used for the exponent part and finally rest 23 bits are used for the mantissa part.

S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF
` 31 30 23 22 0

However, it uses an implicit bit, so the significant part becomes 24 bits, even though it usually is encoded using 23 bits.This conversion can be done using the below algorithm of figure 2.2:

Step1: Sign bit of the binary number becomes the sign bit (31st bit) of the IEEE equivalent.

Step 2: 30th bit to 8th bit of the binary number becomes the mantissa part of the IEEE equivalent.

Step 3: The exponent part is calculated by subtracting the position of the 1st one obtained in the algorithm described in section 2.2.1.

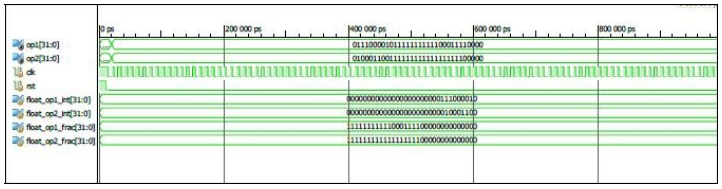
Step 4: A bias of 127 is added to the above exponent value.

Pre-normalization is the process of equalizing the exponents of the operands and accordingly adjusting the entire IEEE754 expression of the inputs to produce correct results maintaining the IEEE754 standard throughout all calculation steps inclusive of the intermediate calculations and their outputs Subtraction can be interpreted as addition of a positive and a negative number. So using the same algorithm as that of addition, we can complete the subtraction operation by taking complement of the negative number and adding 1 to the complement. This is same as taking the 2"s complement of the negative number. Doing this we interpreted the negative number as positive and carry the addition operation.

5. Experimental Results:

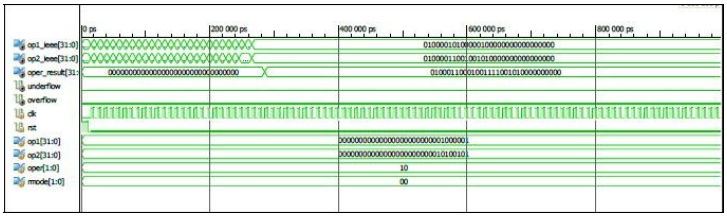
The code was simulated in Xilinx 13.3. We have given some of the screen shots of the simulations that were obtained as a result of simulation in Xilinx software.

Float to Integer Conversion simulation result



ADD simulation result

Multiplication simulation result



6.CONCLUSIONS AND FUTURE WORK:

This paper presents an implementation of a floating point multiplier that supports the IEEE 754-2008 binary interchange format; the multiplier doesn't implement rounding and just presents the significand

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

multiplication result as is (48 bits); thisgives better precision if the whole 48 bits are utilized in another unit; i.e. a floating point adder to form a MAC unit. The design has three pipelining stages and after implementation on a Xilinx Virtex5 FPGA it achieves 301 MFLOPs

7. ACKNOWLEDGMENTS:

I am P.Saritha and would like to thank the publishers, researchers for making their resources material available. I am greatly thankful to Assistant Prof: Miss.Ch.Nirmala for their guidance. We also thank the college authorities, PG coordinator and Principal for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

8.REFERENCES:

[1] IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.

[2] B. Fagin and C. Renard, “Field Programmable Gate Arrays and FloatingPoint Arithmetic,” IEEE Transactions on VLSI, vol. 2, no. 3, pp. 365–367, 1994.

[3] N. Shirazi, A. Walters, and P. Athanas, “Quantitative Analysis ofFloating Point Arithmetic on FPGA Based Custom ComputingMachines,” Proceedings of the IEEE Symposium on FPGAs for CustomComputing Machines (FCCM’95), pp.155–162, 1995.

[4] L. Louca, T. A. Cook, and W. H. Johnson, “Implementation of IEEESingle Precision Floating Point Addition and Multiplication on FPGAs,”Proceedings of 83 the IEEE Symposium on FPGAs for CustomComputing Machines (FCCM’96), pp. 107–116, 1996.

[5] A. Jaenicke and W. Luk, “Parameterized Floating-PointArithmetic on FPGAs”, Proc. of IEEE ICASSP, 2001, vol. 2, pp.897-900.

[6] B. Lee and N. Burgess, “Parameterisable Floating-point Operations onFPGA,” Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems, and Computers, 2002.

[7] “DesignChecker User Guide”, HDL Designer Series 2010.2a, MentorGraphics, 2010

[8] “PrecisionR Synthesis User’s Manual”, Precision RTL plus 2010aupdate 2, Mentor Graphics, 2010
.

[9] Patterson, D. & Hennessy, J. (2005), Computer Organization andDesign: The Hardware/software Interface , Morgan Kaufmann .

[10] John G. Proakis and Dimitris G. Manolakis (1996), “Digital SignalProcessing: Principles,. Algorithms and Applications”, Third Edition.Figure 9. Floating point multiplier with pipelined stagesSep.

Author’s Details:

Ms.P.Saritha. MTech student, in M.Tech Student, Dept of ECE in KITS for women’s,kodad, T.S, India

Ms.Ch.Nirmala working as a Assistant at ECE in KITS for women’s,kodad, T.S, IndiaJNTUH Hyderabad. she has 3 years of UG/PG Teaching Experience.