



Processor Design Using 32 Bit Single Precision Floating Point Unit

Mr. Anand S. Burud, Dr. Pradip C. Bhaskar

Department of Technology, Shivaji University, Kolhapur, Maharashtra, India

ABSTRACT

The floating point operations have discovered concentrated applications in the various different fields for the necessities for high precision operation because of its incredible dynamic range, high exactness and simple operation rules. High accuracy is needed for the design and research of the floating point processing units. With the expanding necessities for the floating point operations for the fast high speed data signal processing and the logical operation, the requirements for the high-speed hardware floating point arithmetic units have turned out to be increasingly requesting. The ALU is a standout amongst the most essential segments in a processor, and is ordinarily the piece of the processor that is outlined first. In this paper, a fast IEEE754-compliant 32-bit floating point arithmetic unit designed using VHDL code has been presented and all operations of addition got tested on Xilinx and verified successfully.

Keywords: Floating point unit, IEEE 754

I. INTRODUCTION

The digital arithmetic operations are very important in the design of digital processors and application-specific systems. In Digital arithmetic operations are very importance on producing DSP processor. For representing a floating point number on computer there is an IEEE (Institute of Electrical and Electronics Engineers) has produced the IEEE 754 standard for representing a floating point number on computer and for arithmetic operation like addition, subtraction, multiplication and division etc. Also there are different representation are there, but there are most widely used is IEEE 754 floating point numbers.

General form of the representing the IEEE 754 floating point number is

$$(-1)^s * M * 2^E$$

Where,

S represents the sign bit,

M represents the mantissa and

E represents the exponent

The first bit is represented as sign bit 1-bit wide and used to denote the sign of the number i.e. 0 indicate positive number and 1 represent negative number. then second bit shows the mantissa part which will shows after the decimal part of an a floating point number and the third part is of an exponent part which will shows the power of an a two. The power will be calculated as the normalizing the no. How many bit will shift to the right on that the power will be calculated.

Range of floating point number will be shows as follows

1) Single Precision

The single-precision number is 32 bit wide. The single-precision number has divided into three sub sections are sign, exponent and mantissa. The mantissa of an single precision is 23-bit can approximately represent a 7-digit decimal number, while an 8-bit exponent to an implied base of 2 provides a scale factor with a reasonable range and 1st bit is for sign bit. Thus, a total number of bits are 32 Bit is needed for single-precision number representation. To achieve a bias equal to $2^{n-1} - 1$ is added to the actual exponent in order to obtain the stored exponent. This is equals to 127 bias for 32 bit

single precision floating point unit and its exponent is eight bit. The addition of bias allows the use of an exponent in the range from -127 to $+128$, corresponding to a range of 0-255 for single precision number.

	Sign	Exponent	Mantissa	Bias
Single Precision	1(31)	8(30-23)	23(22-00)	127

2) Double Precision

The Double-precision number is 64 bit wide. The Double-precision number has also having three main fields that are sign, exponent and mantissa. The 52-bit mantissa is wide, while an 11-bit exponent to an implied base of 2 provides a scale factor with a reasonable range and 1st bit is for sign bit. Thus, a total number of bits are 64 Bit in double precision.

	Sign	Exponent	Mantissa	Bias
Double Precision	1(63)	11(62-52)	52(51-00)	1023

3) Conversion of Decimal to Floating Point Number

Process of conversion of decimal number into floating point number with an example is explained below. Suppose the decimal number considered is 36.5625. Before converting into floating format this number is converted into binary value which is 100100.1001. After the converting into binary format then now its convert into normalizing form. Normalizing form is nothing but shift the decimal point into left such as the number should be 1.001001001. The number which is after the radix point is called the mantissa which is of 23 bits and the whole number is called significand which is of 24 bits(including the hidden bit). The no is shifted how many times is counted and the value is written as 'x'. In above case there is 5 times shifting of radix point to the left. This value must be added to 127 to get the exponent value i.e. original exponent value is $127 + "x"$. Thus the exponent becomes $127 + 5 = 132$ which is 10000100. Sign bit i.e. MSB is "0" because number is +ve. Now the result is assembled into 32 bit format which is sign, exponent, mantissa: 0 10000100 001001001000000000000000.

2 Literature Review:

This standard "IEEE Standard for Floating-Point Arithmetic, ANSI/IEEE Standard 754-2008, New York:" IEEE, Inc. [1], 2008 describes interchange and arithmetic methods and formats for binary and decimal floating-point arithmetic in computer programming environments. This standard specifies exception conditions and their default handling. An implementation of a floating-point system conforming to this standard may be recognized entirely in software, entirely in hardware, or in any combination of software and hardware. The normative part of this standard, numerical results and exceptions specified for different operations in are uniquely determined by the, sequence of operations, values of the input data and destination formats, all under user control.

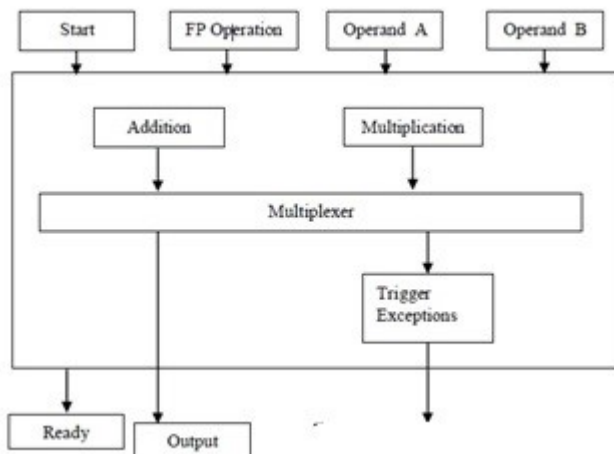
Itagi Mahi P.a* and S. S. Kerurb et al [2] In this paper a pipelined ALU is proposed simulating five arithmetic operations namely addition, subtraction, multiplication, division and square root in the HDL environment. Simulation Results is also obtained in IP Core Generator supported by Xilinx. Synthesis is carried out on the Xilinx 13.2 platform and ISim is used for the simulation process.

Jain, Jenil, and Rahul Agrawal et al. [3] this paper presents design of high speed floating point unit using reversible logic. In recent nanotechnology, Programmable reversible logic design is trending as a prospective logic design style for implementation and quantum computing with low impact on circuit heat generation. There are various reversible implementations of logical and arithmetic units have been proposed in the existing research, but very few reversible floating-point designs has been designed. Floating-point operations are used very frequently in nearly all computing disciplines. The analysis of proposed reversible circuit can be done in terms of quantum cost, garbage outputs, constant inputs, power consumption, speed and area.

Gopal, Lenin, Mohd Mahayadin et al. [4] in the paper, eight arithmetic and four logical operations has been presented. In the proposed design 1, Peres Full Adder Gate (PFAG) is used in reversible ALU design and HNG gate is used as an adder logic circuit in the proposed ALU design 2. Both proposed designs are analyzed and compared in terms of number of gates count, garbage output, quantum cost and propagation delay. The simulation results show that the proposed reversible ALU design 2 outperforms the proposed reversible ALU design 1 and conventional ALU design.

3 Proposed Work:

The block diagram of the designed arithmetic unit is shown in the following figure. It supports three arithmetic operations: addition, subtraction and multiplication. It has 4 inputs: a start bit, two 32-bit operand in IEEE format, one 3-bit operation code ; and has outputs: 32-bit output in IEEE format, one ready bit, eight exceptions. All arithmetic operations have been carried out in four separate modules one for addition and subtraction and one each for multiplication and division as shown in figure. In this unit one can operation is operated as per input is given and the same op-code selects the output from that particular module and connects it to the final output of the unit. Ready bit will be high when result will be available to be taken from the output. Particular exception signal will be high whenever that type of exception will occur. Result precision output is obtained.



FPU BASIC ARCHITECTURE

Floating point addition algorithm

1. Finding the difference of exponents
2. Shifting the fraction of the operand A (smaller one) to the right fraction A
3. Incrementing the exponent A by '1' to equalize it to the larger one.
4. Adding the two fraction parts
5. Normalizing the result if necessary
6. Writing the result

Floating point subtraction algorithm

1. Finding the difference of exponents

2. Shifting the fraction of the operand A (smaller one) to the right fraction A
3. Incrementing the exponent A by '1' to equalize it to the larger one.
4. Subtracting the two fraction parts
5. Normalizing the result if necessary
6. Writing the result

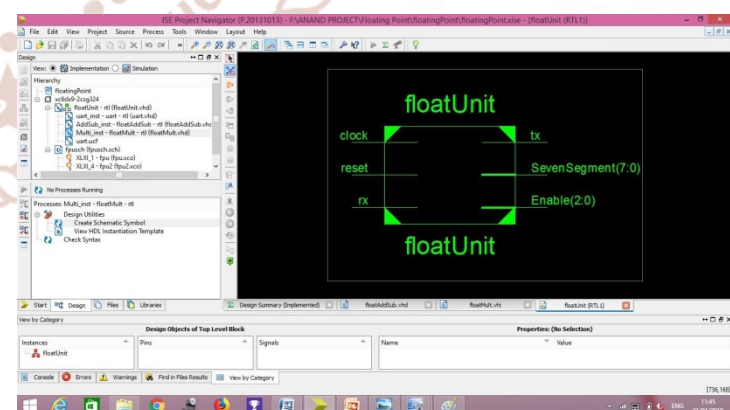
Floating point multiplication algorithm

1. Add exponents
2. Multiply fractions
3. If product is 0, adjust for proper 0
4. Normalize product fraction
5. Check for exponent overflow or underflow
6. Round product fraction

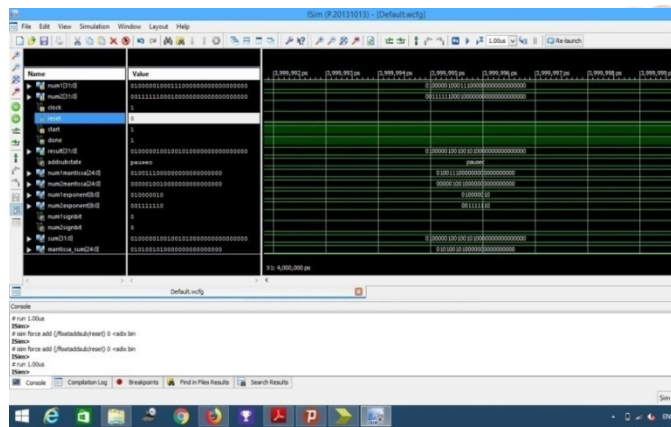
The design has been simulated and synthesized on Xilinx 14.7 ISE Design Suite. The no. of device utilization with reference to it [6,7] reduced. In our system Number of Slice LUTs utilization is 3% and Number used as logic utilization is 3%.

Results:

Top level view Schematic:



Device Utilization Summary:



Number of Slice Registers	235
Number of Slice LUTs	186
Number used as Logic	146
Number used as Memory	0
Number of LUT Flip Flop pairs used	201
Number with an unused Flip Flop	23
Number with an unused LUT	15
Number of fully used LUT-FF pairs	163
Number of IOs	15
Number of bonded IOBs	15

Single precision floating point multiplier is designed and implemented using xilinx in this paper. The designed multiplier conforms to IEEE 754 single precision floating point standard. In this implementation exceptions (like invalid, inexact, infinity, etc) are considered. The designed is verified using fpv_test test bench. Hence hardware requirement is reduced, thereby reducing power consumption and delay.

Arithmetic logical unit plays important role in Digital signal processing applications. ALU is one of the most important components in a processor, and is typically the part of the processor that is designed first. In future by using this ALU we can implement processor for various applications.

Page: 201

- 1) Sneha S.jumle and M. V. Vyawhare Development of an FPGA based high speed single precision floating point multiplier (International Journal of Electronic and Electrical Engineering.)
- 2) Sayali A. Bawankar¹, Prof. G. D. Korde² Review on 32 bit single precision Floating point unit (FPU) Based on IEEE 754 Standard using VHDL (IRJET)
- 3) Tasnim Ferdous Design, Synthesis and FPGA-based Implementation of a 32-bit Digital Signal Processor International Journal Of Scientific & Engineering Research, Volume 3, Issue 7, July-2012
- 4) Itagi Mahi P.a* and S. S. Kerurb Design and Simulation of Floating Point Pipelined ALU Using HDL and IP Core Generator International Journal of Current Engineering and Technology
- 5) Prabhjot Kaur Rajiv Ranjan R a m inder Preet Pal Singh Onkar Singh Double Precision Floating Point Arithmetic Unit Implementation- A Review (IJERT)
- 6) Amana Yadav Ila Chaudhary Design of 32-bit Floating Point Unit for Advanced Processors. ijera
- 7) Itagi Mahi P.a* and S. S. Kerurb Design and Simulation of Floating Point Pipelined ALU Using HDL and IP Core Generator International Journal of Current Engineering and Technology
- 8) Kahan, William. "IEEE standard 754 for binary floating-point arithmetic."Lecture Notes on the Status of IEEE 754.94720-1776 (1996): 11.
- 9) Ykuntam, Yamini Devi, MV Nageswara Rao, and G. R. Locharla. "Design of 32-bit Carry Select Adder with Reduced Area." International Journal of Computer Applications 75.2 (2013).
- 10) Quinnell, Eric, Earl E. Swartzlander Jr, and Carl Lemonds. "Floating-point fused multiply-add architectures." Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on. IEEE, 2007
- 11) F. Conti, D. Rossi, A. Pullini, I. Loi, and L. Benini, "Energy-efficient vision on the PULP platform for ultra-low power parallel computing," in Signal Processing Systems (SiPS), 2014 IEEE Workshop on, Oct 2014.
- 12) V. Camus, J. Schlachter, and C. Enz, "Energy-efficient inexact speculative adder with high performance and accuracy control," in Circuits and Systems (ISCAS), 2015 IEEE International Symposium on, May 2015.
- 13) Naresh Grover, M.K.Soni Design of FPGA based 32-bit Floating Point Arithmetic Unit and verification of its VHDL code using MATLAB I.J. Information Engineering and Electronic Business, 2014, 1, 1-14 Published Online February 2014 in MECS
- 14) Syed M. Qasim, Ahmed A. Telba and Abdulhameed Y. AlMazrooFPGA Design and Implementation of Matrix Multiplier Architectures for Image and Signal Processing Applications IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010.