International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 5
ISSN: 2454-4248
141– 144

# Enhancing Precision in Cloud Computing: Implementation of a Novel Floating point Format on FPGA

Pranali Kalamkar
PG Student, Dept. of Electronics and Telecommunication
BSIOTR, Wagholi,
Pune, India
*pranaliranpise2015@gmail.com*

Prof. D.S Bhosale
Professor, Dept. of Electronics and Telecommunication
BSIOTR, Wagholi,
Pune, India
*bhosale.dilip@rediffmail.com*

***Abstract*——** In this paper, we propose a Internet-based Cloud computing service that provides computing, storage and networking services to multiple users. Computing capacity runs out quickly in cloud computing services with the increase in data size. To fill the shortage of computation capacity, we propose to adopt variable precision by implementing unum (universal number). Unum is a number format different from IEEE Standard for Floating-Point Arithmetic – IEEE 754 floats. Compared with IEEE 754 floats, the outstanding features of unum are clearance of rounding errors, high information-per-bit and variable precision. As a candidate replacement of IEEE 754 floats, the application of unum can improve the precision in computing. It decreases the bit width for high precision numbers. However, unum was only implemented in software model before due to technical complexity, in order to validate the performance on chip, we implement this arithmetic on FPGA for the very first time. We also implement an unum based 16-point FFT on FPGA. We validate the design and compare the bit width in computing with IEEE 754 floats. The experimental results of comparison show that unum arithmetic can ensure correctness even in some extreme arithmetic cases in which IEEE 754 floats cannot work properly, furthermore the bit width of unum is much less than IEEE 754 floats in the same precision.

***Keywords-*** *FPGA, Cloud computing, MATLAB, Bandwidth, Floating Point Number*

*****

## I. INTRODUCTION

Cloud computing is a new paradigm to meet elastic computing requirements of users by providing scalable computing, storage and networking services. To meet the ever-growing requirement of computation capacity along with the increase in data volumes, intensive research has been carried out to improve performance by optimizing resource utilization. However, it was rarely reported in the literature to improve cloud computing capacity by exploiting variable computation accuracy since users are educated to use standard accuracy only, such as IEEE 754 floats.

To enable variable precision computing, unum was first introduced publicly by John L. Gustafson in 2013, the goal of this number format is to overcome the weakness of existing floats standard – IEEE 754 floats. With a book introduce unum is published in 2015, the research group of Gustafson has done some pioneer work as they found that the main weaknesses of IEEE 754 floats are rounding errors and too large bit width in computing.

Unum floats and IEEE 754 floats have different definition of bit string. Bit string of IEEE 754 floats consists of 3 parts: sign bit, exponent bits and fraction bits. For IEEE 754-style floats if the bit width is constricted, it cannot represent an exact number, then the actual value will be rounded to an approximate value. As the computation process goes on, the accumulation of rounding errors will become treacherous and the result would deviate from exact value enormously, especially in cloud computing with sensitive numeric analysis. The rounding errors also effect the use of parallel methods.

Being different from IEEE 754 floats, bit string of unum floats is comprised of 6 parts: sign bit, exponent bits, fraction bits, ubit bit, exponent size bits and fraction size bits, exponent size and fraction size bits can be 0. Unum was only implemented in software model before due to technical complexity, we would be the first team to implement this arithmetic logic on FPGA to the best of our knowledge. The main contributions of our work would be: (1) Unum basic arithmetic addition, subtraction, multiplication and division implementation on FPGA, with comparison of computing results and power dissipation against IEEE 754 floats arithmetic module. An application: unum arithmetic based 16-point FFT implementation on FPGA and the comparison with IEEE 754 floats based FFT module, which is a common computation core in cloud computing.

## II. LITERATURE SURVEY

[1], Junjie Hou, Yongxin Zhu, Yulan Shen, Mengjun Li, Qian Wu and Han Wu

School of Microelectronics, Shanghai Jiao Tong University, Shanghai, China 200240240 on “Enhancing Precision and Bandwidth in Cloud Computing: Implementation of a Novel Floating-Point Format on FPGA”

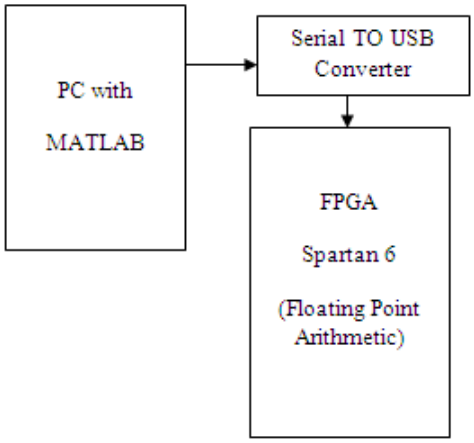
Cloud computing is a type of Internet-based service computing that provides computing, storage and networking services to multiple users. With the increase of data size,

 ISSN: 2454-4248
141– 144

computing capacity runs out quickly in cloud computing services. To fill the shortage of computation capacity, we propose to adopt variable precision by implementing unum (universal number), which is a number format different from IEEE Standard for Floating-Point Arithmetic – IEEE 754 floats. Compared with IEEE 754 floats, the outstanding features of unum are clearance of rounding errors, high information-per-bit and variable precision. As a candidate replacement of IEEE 754 floats, the application of unum can improve the precision in computing, decrease the bit width for high precision numbers.

However, unum was only implemented in software model before due to technical complexity, in order to validate the performance on chip, we implement this arithmetic on FPGA for the first time. We also implement an unum based 16-point FFT on FPGA. We validate the design and compare the bit width in computing with IEEE 754 floats, evaluate the power dissipation on FPGA. The experimental results of comparison show that unum arithmetic can ensure correctness even in some extreme arithmetic cases in which IEEE 754 floats cannot work properly, furthermore the bit width of unum is much less than IEEE 754 floats in the same precision.

## III. PROPOSED SYSTEM




.

Figure 1. System Architecture

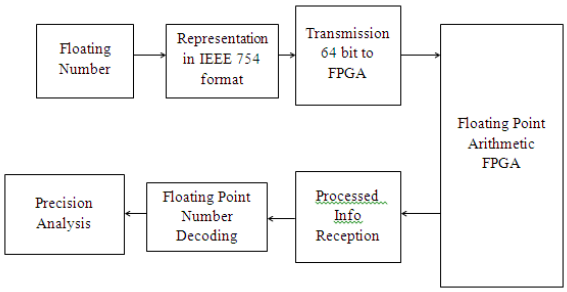



Figure 2. Block diagram of Image Processing

Our proposed system is as shown in above fig. Matlab is responsible for creating the GUI and taking the input operands from user.

After acquisition of operand, it is converted into IEEE 754 format. Block of 64 bits is then sent to FPGA for data compression. Reconfigurable FPGA is performing the task of arithmetic on received data

IEEE 754 Representation

In high precision numerical calculation, especially in scientific calculation the precision of number format have a great influence on the validity of computing results, the accumulation of rounding errors may become hazardous than we expected and cause many confusing paradoxes.

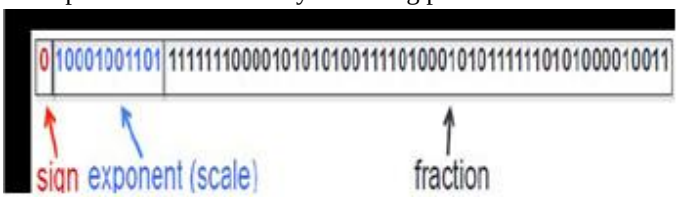



Figure 3. IEEE 754 Double Float

Design Flow

This section examines the flow for design using FPGA. This is the entire process for designing a device that guarantees that we will not overlook any steps and that we will have the best chance of getting back a working prototype of functions correctly in our system. The design flow consists of the steps as shown in the below flow diagram.

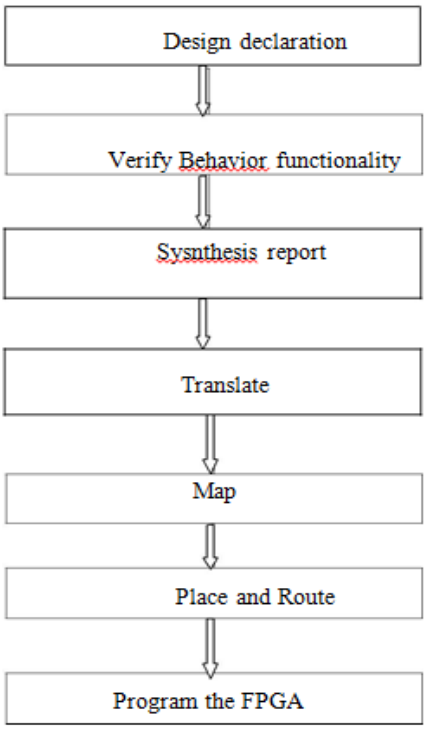
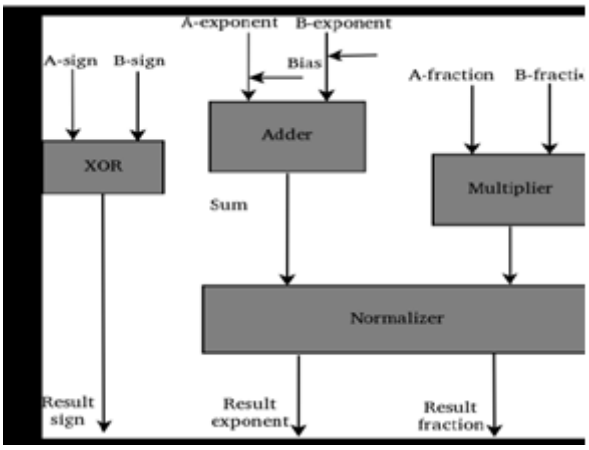



Fig. 4. Design Flow

 ISSN: 2454-4248
141– 144

Computing Architecture of Float Multiplier




Computing Architecture of floats multiplier is depicted in Fig.4, the procedures of floats multiplier require following steps:

1. Determine the sign of result
2. Add two exponent parts
3. Integer multiplication of fraction parts
4. Normalize the results

For multiplier, sign of the result can be obtained directly with sign of two operands by XOR. Then adding the exponents A − exponent and B − exponent, if the added exponent is bigger or smaller than the maximum or minimum exponent that can be represented, overflow or underflow occur. Then Integer multiplication of two fraction parts are performed, the amount of multiplied fraction bits is the sum of significant bits of two fraction parts, but only the higher bits are conserved because of the constrain of bit width. Finally, transfer the result into normalized format.

## IV. Hardware and Software Specification

### A. *Hardware Specification*

#### *1) Spartan 6 FPGA*

- Flash memory: 16 Mb SPI flash memory (M25P16)
- USB 2.0 interface for On-board flash programming
- FPGA configuration via JTAG and USB
- 8 LEDs ,Six Push Buttons and 8 way DIP switch for user defined purposes
- One VGA Connector
- One Stereo Jack
- One Micro SD Card Adapter
- Three Seven Segment Displays
- 39 IOs for user defined purposes
- On-board voltage regulators for single power rail operation

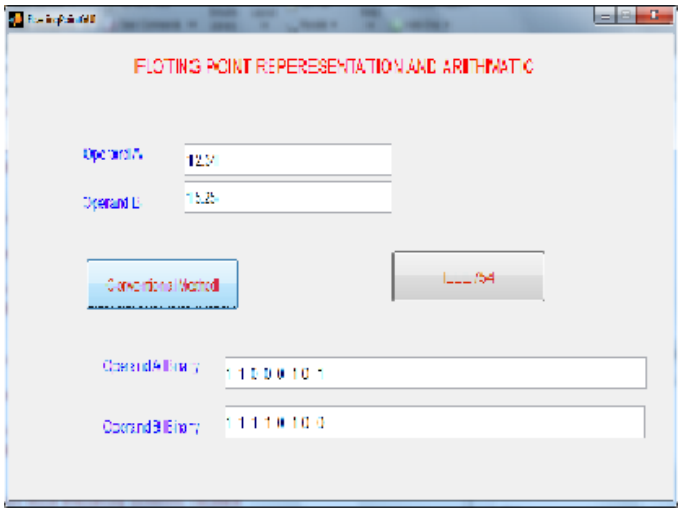
#### *2) Software Specifications*

The proposed system is implemented using C/C++ languages in MATLAB .

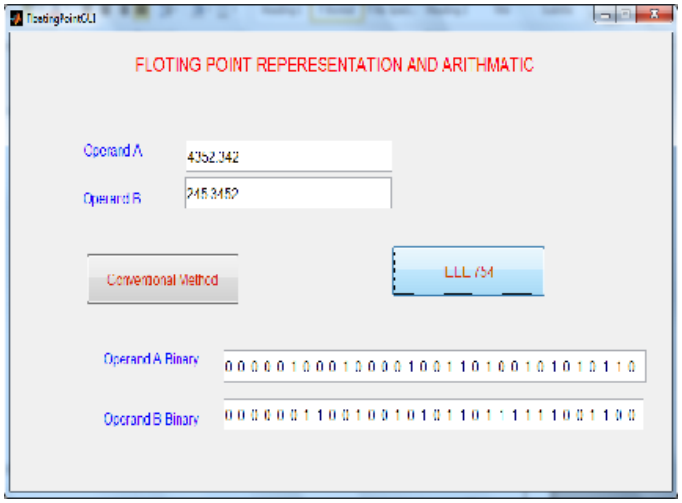
## V. Results

Results obtained from Matlab and Xilinx are as follows:

1. Conventional Method




2. IEEE 754 Method




## VI. Conclusion

During the study and implementation of floating point arithmetic and representation on FPGA, we found that our proposed architecture is less time consuming and area consuming. We are also able to enhance precision by our proposed method.

The feature of clearance of rounding error makes it more suitable for high precision computing, especially in scientific computing as well as cloud computing where sensitive numeric analytical computation is required.

Our application is expected to decrease the computational capacity and memory bandwidth requirement without losing accuracy in computing.

 ISSN: 2454-4248
141– 144

## REFERENCES

[1] Chandrima Dadi, Ping Yi, Zongming Fei, and Hui Lu. A new block-based data distribution mechanism in cloud computing. In Cyber Security and Cloud Computing (CSCloud), 2016 IEEE 3rd International Conference on, pages 54–59. IEEE, 2016.

[2] Jiayin Li, Meikang Qiu, Zhong Ming, Gang Quan, Xiao Qin, and Zonghua Gu. Online optimization for scheduling preemptable tasks on iaas cloud systems. Journal of Parallel and Distributed Computing, 72(5):666–677, 2012

[3] Meikang Qiu, Zhong Ming, Jiayin Li, Keke Gai, and Ziliang Zong. Phase-change memory optimization for green cloud with genetic algorithm. IEEE Transactions on Computers, 64(12):3528–3540, 2015.

[4] John L. Gustafson. The End of Error: Unum Computing, volume 24. CRC Press, 2015.

[5] John L Gustafson. A radical approach to computation with real numbers. In Supercomputing Frontiers, 2016.

[6] William Edmund Milne. Numerical calculus. Princeton University Press, 2015.

[7] J Brandts, M Kr´ızek, and Z Zhang. Paradoxes in numerical calculations. Neural Network World, 26(3):317, 2016.

[8] John L. Gustafson. The end of numerical error. In 2015 IEEE 22nd Symposium on Computer Arithmetic, page 74, June 2015.

[9] Grace Nordin, Peter A Milder, James C Hoe, and Markus P¨uschel. Automatic generation of customized discrete fourier transform ips. In Proceedings of the 42nd annual Design Automation Conference, pages 471–474. ACM, 2005.

[10] Gokhan Polat, Sitki Ozturk, and Mehmet Yakut. Design and implementation of 256-point radix-4 100 gbit/s fft algorithm into fpga for high-speed applications. ETRI Journal, 37(4):667–676, 2015.