ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

Double Precision IEEE-754 Floating-Point Adder Design Based on FPGA

Adarsha KM[1], Ashwini SS[2], Dr. MZ Kurian[3]

PG Student [VLS& ES], Dept. of ECE, Sri Siddhartha Institute of Technology, Tumkur, Karnataka, India[1]

Assistant professor, Dept. of ECE, Sri Siddhartha Institute of Technology, Tumkur, Karnataka, India[2]

HOD, Dept. of ECE, Sri Siddhartha Institute of Technology, Tumkur, Karnataka, India[3]

ABSTRACT:Because of dynamic representation capabilities and large spectrum of numbers can be represented with limited number of bits, floating-point numbers are being widely adapted in the fields of scientific applications. A floating-point arithmetic unit is specifically designed to carry out on floating-point numbers and is one of the most common part of any computing system in the area of binary applications. Floating-point additions are the most frequent floating-point operations and floating-point adders are therefore critically important components in signal processing and embedded platforms. This review paper presents the survey of related works of different algorithms/techniques which are important for implementation of double precision floating point adder with reduced latency based on FPGAs. As per the review paper the basic design deals with the floating point adder implementation, the proposed design handles delay optimization.

KEYWORDS: Floating-Point Addition, IEEE-754, FPGA, Delay Optimization.

I.INTRODUCTION

In digital systems IEEE-754 [8] standard is used for representing the floating-point numbers and are of different types according to their mantissa bit length. There are half, single, double and quadruple precision binary numbers having a mantissa of bit length 16, 32, 64 and 128 respectively in accordance with IEEE-754 2008 standard format. Using double precision binary numbers large spectrum of numbers can be expressed with a limited number of bits and hence these are being most widely used in the area of scientific and engineering applications. Double precision representation is the IEEE-754 standard of 64-bit format with 1 bit for sign, 11 bits of exponent and 52 mantissa bits, thus provides greater dynamic range as shown in fig 1.

Length = 1	Length = 11	Length = 52
Sign (S)	Exponent (E)	Significand

Fig 1: General Representation of IEEE 754-2008 Double Precision Floating-Point Numbers

In the implementation of high performance systems, modern FPGAs are considering as a great valuable quality hardware prototyping tools. Every year FPGAs are coming out with more features and are being established as most valuable tools in the implementation of high performance systems to perform computations at high clock frequencies. FPGAs have dedicated multipliers, memories and carry chains, most of the blocks are aiming at signal processing computations and even general processors can be incorporated. Although FPGAs have the advantage of re-programmability over general purpose processors with the parallel processing advantage and speed of custom hardware improves fixed point computations but it is harder to implement high performance floating-point computations on FPGAs. One of the major challenge during the implementation is the requirement of the normalization in a floating-point addition. To handle arithmetic operations, floating-point unit is designed and it also performs functions such as exponential calculations. This paper presents the review of an implementation of the double precision floating point

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

adder which can perform the addition within two clock cycles with considerable reduction in latency. Over the fixed point and integer representation floating point-representation has its own advantages like it can support wide range of values.

II.RELATED WORKS

A.OPTIMIZATION TECHNIQUES [4]

A detailed examination of optimization techniques enables designer to demonstrate how these methods can be grouped to achieve an overall fast floating-point adder implementation. In general, considerable reduction of latency by parallel paths requires balancing the delay of the parallel paths. In [4] such a balance was achieved by a gate level consideration of the design. A brief overview of the optimization techniques used in paper [4] are given below:

- A two path design with a nonstandard separation criterion. Instead of dividing based on magnitude of the exponent difference, defining a separation criterion that also considers whether the operation is effective subtraction and the significand difference. Here the advantage is that, alignment shift and normalization shift take place only in one of the paths and the full exponent difference is calculated only in one path and also this separation technique requires rounding to take place only in one path.
- IEEE rounding modes are reduced to three modes and use of injection based rounding.
- A simple design can be achieved by using unconditional pre-shifts for effective subtractions to reduce to two the number of binades that sum and difference of significand may belong to.
- Exponent's difference and significand's differences sign magnitude representation is derived from one's compliment representation of the difference.
- Sum and the incremented sum of the significands can be computed using the parallel-prefix adder.
- Using of recoding's to estimate the number of leading zeros in the non-redundant representation of a number represented as a borrow-save number.
- Post-normalization will take place before the rounding decision is ready.

From the results of implementation in [4], one can see that the presentation of technology-independent analysis and optimization of the IEEE floating point addition implementation based on the logical effort hardware model and determined optimal gate size and optimal buffer insertion. The adder receives normalized numbers, supports all IEEE rounding modes, and registers the outputs correctly normalized rounded sum/difference in the format required by the IEEE standard. The presented algorithm was a two staged pipeline partitioned into two parallel paths called R-path and the N-path. The final result is selected between the outcomes of the two paths. The floating-point adder designed achieved a low latency by combining various optimization techniques.

B.PIPELINED FPGA ARCHITECTURE [5]

The advantage of the pipelining architecture mechanism is that, regardless of having a higher input and output sequential length, it offers an unmatched throughput by virtue of their assembly structure. Pipelined FPGA architectures significantly reduce the latency and some of the commercial (open source) floating-point cores assume the precision to be the critical parameter. To get optimal throughput rates, the number of pipelining stages also has to be considered as a parameter. Some of the floating-point cores use a common format with conversion to and from the IEEE-754 technical standard at interfaces to other resources in the system. Various tools have been developed to automate the optimization and generation of the floating-point units.

C.PIPELINED PACKET-FORWARDING PARADIGM [2]

Paper [2] presents the pipelined adder design and algorithm for a floating-point addition employing a packet forwarding pipeline paradigm. To handle data hazards in deeply pipelined floating-point pipeline, their proposed work and packet forwarding format constitute a new paradigm. The two algorithms addition and rounding form a phase of four stage execution pipeline with each stage responsible for implementation in a minimum clock period. The first two clock cycles perform addition and last two clock cycles perform the rounding. There are two format given to the adder one

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

operand in standard binary format (Fig 2) at the start of the cycle first and other operand is in the packet forwarding floating-point format, (Fig 3) which is divided into four parts: the sign bit, the exponent string, the principle part of the significand, and carry-round packet. The second operand's first three parts are input at the start of the cycle one and the carry-round packet is input at the start of the input two. According to the pipelined packet-forwarding paradigm, a floating-point adder is implemented by a pipeline of 4 stages as depicted in Fig.4. The first two stages perform the addition operation and the last two stages deal with rounding. The registered result at the output is in two formats one in packet forwarding floating-point format at the end of the cycle two and three to allow forwarding with an effective latency of two cycles and another, in the standard IEEE 754 at the end of cycle four. Fig 5 depicts an execution of successive dependent floating-point operations. The latency is 2 clock cycles and there is only one stall cycle between successive dependent operations.

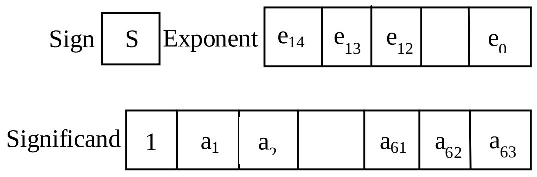

Fig 2. Standard IEEE-754 Floating-Point operand format

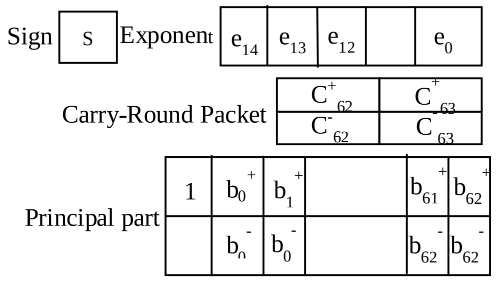

Fig 3. Packet-forwarding floating-point operand format

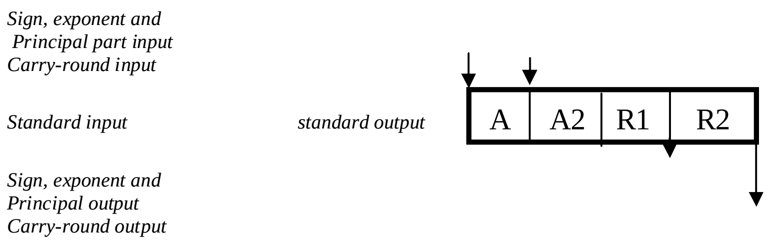

Fig 4. Input-Output schedule of pipeline

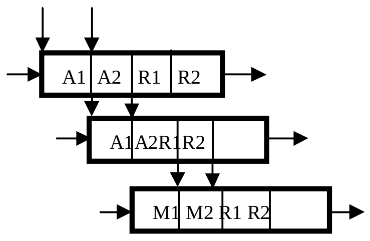

Fig 5.Execution of successive dependent operations

D.FLAGGED PREFIX ADDITION [1]

Paper [1] describes the reduced latency IEEE floating point standard architectures which uses flagged prefix addition to merge rounding with the significand addition. They presented architecture of a floating-point adder with latency of two cycle with potentially the same cycle time that also employs flagged prefix addition as the modified architecture for their three cycle latency and supports all rounding modes. A flagged prefix adder is a modified parallel prefix type carry look ahead adder which calculates the longest string of one's from the second LSB upwards and sets them as flag bits to indicate those bits which can be inverted for incrementing a two's compliment number. A three cycle floating-point adder using a flagged prefix integer adder is similar in design to the traditional three stage pipelined adder, in that rounding of the significand, which is normally done after the normalization shift is merged with the significand addition in the second pipeline stage. A two cycle floating-point adder has separate data paths depending on whether or not the data requires a large alignment shift or large normalization shift.

N.Kikkeri, P.M Seidel (2007) reported an implementation of highly optimized double precision IEEE floating-point adder, which was based strongly on optimization techniques such as nonstandard dividing of the two paths, a simple rounding algorithm, unification of rounding cases for addition and subtraction, one's complement subtraction based sign magnitude computation of a difference, compound adders and fast circuits for approximate counting of leading zeros from borrow save representation. They reported fully gate-level verification and is important to increase the confidence in the resulting implementation as opposed to verification on high level abstractions where the representation of the inputs and the blocks may be different from the actual hardware. They used Verilog code for their design then compiled in altera qwartus2 to realize the implementation. The implemented design was having a successfully reduction in latency on an altera starix device and their design was implemented as a two stage pipeline to handle data hazards in deeply pipelining stages. [7].

Purna Ramesh Addanki, Venkata Nagartna Tilak Alapati and Mallikarjuna Prasad Avana (2013) presented a high speed floating-point double precision adder/subtractor and multiplier, which are implemented on a virtex-6 FPGA using Verilog language. Their proposed designs were compliant with IEEE 754 standard format and handles overflow, underflow cases and rounding mode. The IEEE standard specifies four rounding modes and the rounding odes are selected for various bit combinations of mode. Based on the changes in the rounding to the mantissa corresponding changes has to be made in the exponent path also. They showed that, their presented design was achieved high operating frequency with better accuracy and considerably good performance. [9].

III.PROPOSED DESIGN WORK

Proposed implementation work is following the similar design method as in [1]. Floating-Point adder is partitioned into two stage pipeline as shown in Fig 6. Two stage pipeline is divided into two paths namely R and N paths, these are being selected based on the exponent difference of two operands. The new algorithm is being achieved at by bringing a few implementation changes in the algorithm of [1]. The two pipeline stages are going to execute in two different clock cycles. Let, two input operands which are in the IEEE-754 format, represented by their bit vectors as, (SA, EA, FA) and (SB, EB, FB) and let SOP defines the operation going to perform (0-for addition and 1-for subtraction).

The required floating-point addition operation will be performed by the below formulae:

$$rnd(sum) = rnd((-1)^{SA}.2^{EA}.FA + (-1)^{(SOP+SB)}.2^{EB}.FB)$$

Where,

$$S.EFF = SA \oplus SB \oplus SOP$$

So,

$$sum = (-1)^{Sl}.2^{El}.(Fl + (-1)^{S.EFF}(Fs.2^{-|\delta|}))$$

Where δ is the exponent difference and (Sl, El, Fl), (Ss, Es, Fs) are the small and large operands respectively.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

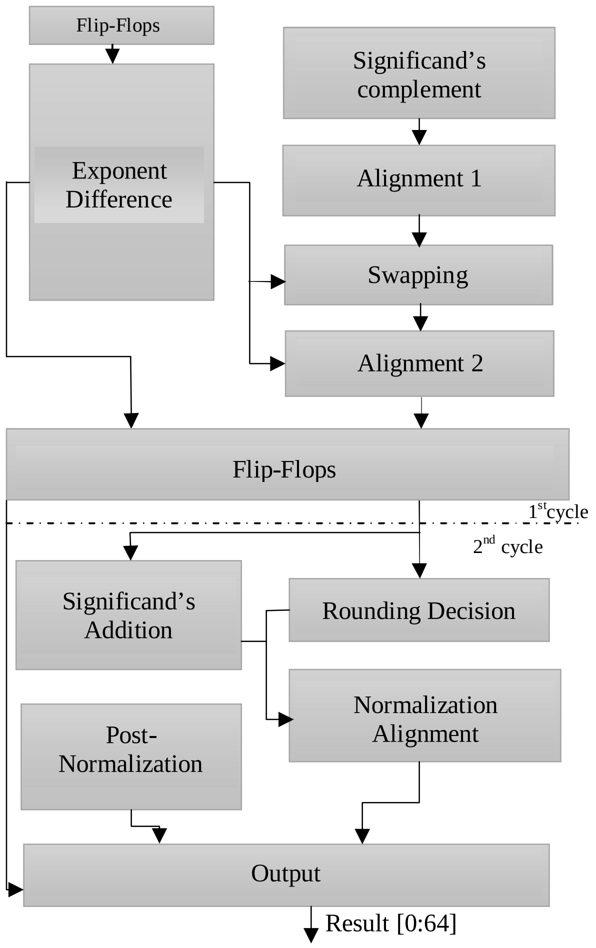

Fig. 6. Higher Level Representation of the Algorithm

A.OVERVIEW OFFIRST CLOCK CYCLE

Fig.7 is the first stage of the pipelining mechanism.Exponent difference will be calculated for two ranges: the medium exponent is in the range of [-63, 64], and the big exponent difference is in the range of [-∞, -64] and [65, ∞]. The outputs of the exponent difference module are given as follows:SIGN_MED and MAG_MAD are the sign-magnitude representation of exponent difference (δ). If δ is in the medium exponent difference interval.

$$(-1)^{SIGN_{MED}\cdot\{MAG_{MED}\}} = \begin{cases} \delta - 1 & if\ 64 \geq \delta \geq 1 \\ \delta & if\ 0 \geq \delta \geq -63 \\ "don't\ care & otherwise \end{cases}$$

In the big exponent difference interval range, the required alignment shift is at least 64 bit positions. SIGN_BIG is the sign bit of the exponent difference δ. IS_BIG is a flag defined by:

$$IS\_BIG = \begin{cases} 1, & if\ \delta \geq 65\ or\ \delta \leq -64 \\ 0, & otherwise \end{cases}$$

An 11-bit adder is used to subtract the exponents and the carry out is used to identify if e1 is smaller than e2. If the result is negative, it has to be complemented and a 1 has to be added to it in order to get the required difference. The exponent difference module can be implemented by cascading a seven bit adder with five bit adder. The seven bit adder calculates the lazy 1s complement exponent difference in the medium interval. This difference is converted to sign and magnitude representations as shown in fig 6. Using cascading adders one can evaluate the exponent difference in parallel with finding out whether the exponent difference is in the big interval or range. Signal SIGN_BIG is the MSB of the lazy 1s complement exponent difference, IS_BIG signal is determining by OR-ing the bits in the positions [6:10] of magnitude of the lazy 1s complement exponent difference and explains why the medium interval is not symmetric around zero.One's complement box is responsible for computing the signals FAO, FBO, and S.EFF. The FAO and FBO signals are specified by

$$FAO[0:52], FBO[0:52] = \begin{cases} FA[0:52], FB[0:52], & if\ S.EFF = 0 \\ not(FA[0:52]), not(FB[0:52]), & otherwise \end{cases}$$

If the range of exponent difference is in the medium interval then *pre-shift* and *align* region computations are considered as relevant. If an effective subtraction takes place then the significands are pre-shifted. Based upon the signal *sign_big*the large operand is selected in the swap region and similarly based upon the signal *sign_med* small operand is selected for the medium exponent difference and based on *sign_big* large exponent difference range is selected as shown in fig.6.Align 2 region deals with pre-shifting the minuend in case an effective subtraction takes place.

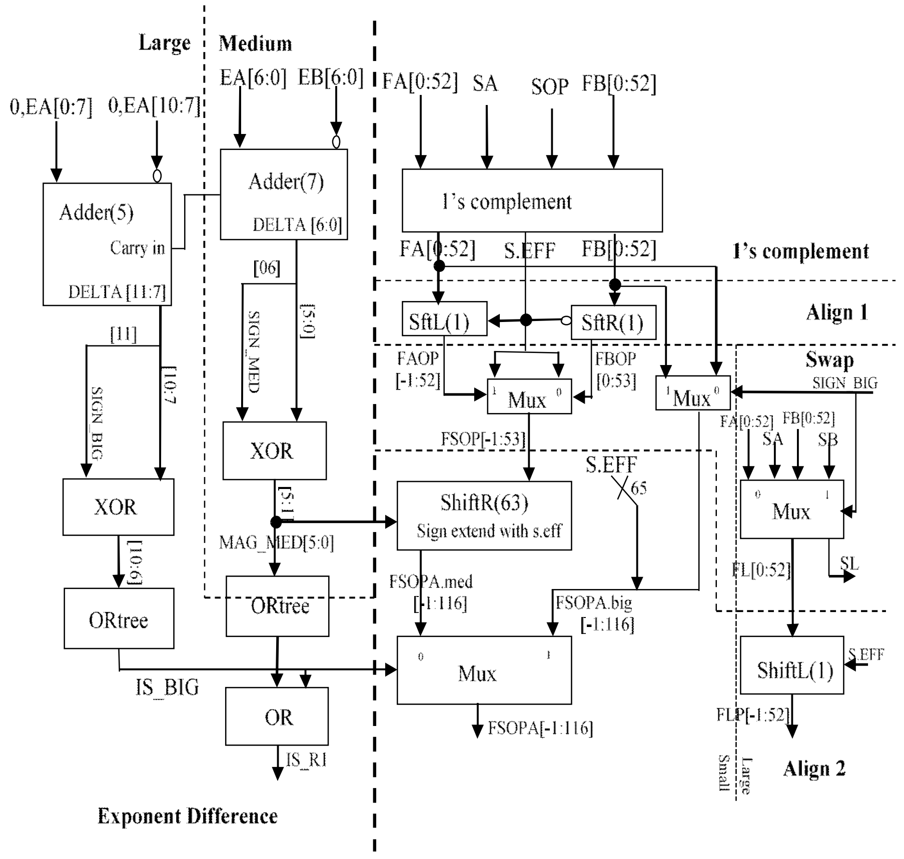

Fig. 6. Block Level Representation of the 1st Cycle of the Algorithm

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

B.OVERVIEW OF SECOND CLOCK CYCLE

This is the second stage of the pipelining mechanism. Here the two significands which are pre-processed are added and the result is rounded in accordance with the IEEE-754 standard floating-point rounding algorithm.Inputs to the second cycle are: the sign bit SL, exponent representation *el*, the significand strings FLP[-1:52] and FSOPA[-1:116], and the rounding mode. At the end rounding algorithm will be implemented and it is normalized. The output result will be a sixty four bit binary floating-point number.

$$rnd(sum) = rnd((-1)^{SA}.2^{EA}.FA + (-1)^{(SOP+SB)}.2^{EB}.FB)$$

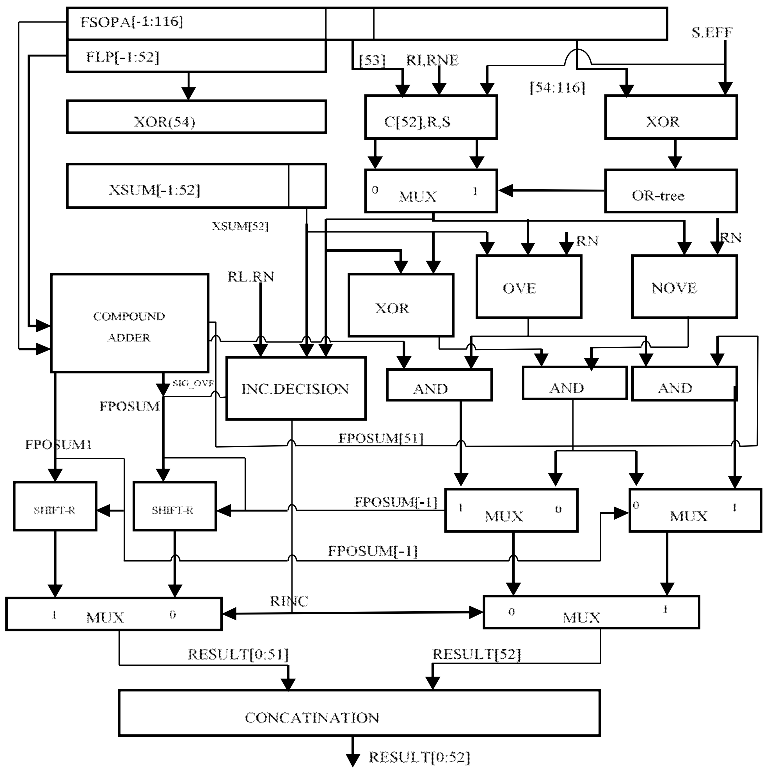

Fig 7. Block Level Representation of the 2nd Cycle of the Algorithm

IV.RESULT AND CONCLUSION

Survey of different algorithms/techniques which are essential for implementation of floating-point arithmetic is discussed. Overview of the proposed implementation work is shown. We have seen that the double precision floating-point unit compared to single precision floating-point provides high speed and gives more accuracy. FPGA based embedded systems have a higher advantage of lower computational aspects, and the double precision floating-point adder supports the IEEE standard format.

REFERENCES

[1] A. Beaumont-Smith, N. Burgess, S. Lefrere, C. Lim, "Reduced Latency IEEE Floating-Point Standard Adder Architectures," Proc. of 14th IEEE Symposium on Computer Arithmetic, pp. 35-43, 1999.

[2] A. Nielsen, D. Matula, e.N. Lyu, G. Even, "IEEE Compliant Floating-Point Adder that Conforms with the Pipelined Packet-Forwarding Paradigm," IEEE Trans. on Computers, vol. 49, no. 1, pp. 33-47, Jan. 2000.

[3] Paschalakis, S., Lee, P., "Double Precision Floating-Point Arithmetic on FPGAs", *In Proc. 2003 2nd IEEE International Conference on Field Programmable Technology (FPT '03)*, Tokyo, Japan, 2003.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

[4] Peter-Michael Seidel, Guy Even, "Delay-Optimized Implementation of IEEE Floating-Point Addition", IEEE Trans. on Computers, vol. 53, no. 2, pp. 97-113, Feb. 2004.

[5] Akshay Sharma, Katherine Compton, Carl Ebeling and Scott Hauck, "Exploration of Pipelined FPGA Interconnect Structures" February 22–24, 2004, Monterey, California, USA.

[6] Xilinx Inc., *Floating-Point Operator v3.0*. Product Specification, 2005.

[7] N. Kikkeri, P.M. Seidel, "An FPGA Implementation of a Fully Verified Double Precision IEEE Floating-Point Adder", Proc. of IEEE International Conference on Application-specific Systems, Architectures and Processors, pp. 83-88, 9-11 July 2007.

[8] IEEE Computer Society, "IEEE Standard for Floating-Point Arithmetic", IEEE Std. 754^{TM}-2008 (IEEE Std 754-1985), Aug, 2008.

[9] Purna Ramesh Addanki, Venkata Nagaratna Tilak Alapati and Mallikarjuna Prasad Avana, "An FPGA Based High Speed IEEE – 754 Double Precision Floating Point Adder\Subtractor and Multiplier Using Verilog", International journal of advanced science and technology. Vol. 52, March, 2013.

[10] *Xilinx*; Virtex-4 User Guide.