International Journal of Research

Available at https://edupediapublications.org/journals

P-ISSN: 2348-6848

E-ISSN: 2348-795X

Volume 07 Issue 06 June 2020

Realization of High Speed FPU Adder

O. Prasad (Research Scholar) [1], K.Kanthi Kinnera (Associate Professor) [2]
Gonna institute of information Technology and Science, Gonnavanipalem, Andhra Pradesh 530053

Abstract:

Floating point unit (FPU) is a part of computer system specially designed to carry out operation on floating point number. This paper shows review of IEEE floating point unit (FPU) which will perform addition function on 64 bit operand that uses the IEEE-754 standard. Floating point numbers representation can support a much wider range of values than fixed point representation. In this paper proposes the area and power efficient ripple carry adder compare to the conventional adder (carry skip adder). The work is to implement and analyses floating point adder operation and hardware module were implemented using VERILOG and synthesized using Xilinx ISE suite.

Keywords: Carry skip adder, Ripple carry adder, FPU and VERILOG.

I. Introduction

There are several ways to represent real numbers on computers. Floating point representation, in particular the standard IEEE format, is by far the most common way of representing an approximation to real numbers in computers because it is efficiently handled in most large computer processors. Binary fixed point is usually used in special-purpose applications on embedded processors that can only do integer arithmetic, but decimal fixed point is common in commercial applications. Fixed point places a radix point somewhere in the middle of the digits, and is eqallent to using integers that represent portions of some unit. Fixed point has a fixed window of representation, which limits it from representing very large or very small numbers.

Also, fixed-point is prone to a loss of precision when two large numbers are divided. Floating point solves a number of representation problems. Floating point employs a sort of “sliding window” of precision appropriate to the scale of the number. This allows it to represent numbers from1, 000,000,000, and 0.0000000000001 with ease.

Floating-point representation over fixed-point (and integer) representation is that it can support a much wider range of values. Floating-point representation is the most common solution basically represents real value in scientific notation. Scientific notation represents numbers as a base number and an exponent. For example, 123.456 could be represented as 1.23456 x 10^2. Floating-point numbers are typically packed in to a computer datum as the sign bit, the exponent field, and the signific and (mantissa), from left to right. In computing, floating-point describes a system for representing numbers that would be too large or too small to be represented as integers. Numbers are in general represented approximately to fixed number of significant digits and scaled using an exponent. The for the scaling is normally 2, 10or 16. The typical number that can be represented exactly is of the form:

Significant digits x base exponent …… (1)

The term floating-point refers to the fact that the radix point (Decimal point or more commonly used in computers, Binary point) can “float”; that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated separately in the internal representation, and floating-point representation can thus be thought of as a computer realization of scientific notation.

IEEE 754 Floating Point Formats

IEEE 754 specifies four formats for representing floating-point values:

1. Single-precision (32-bit)

2. Double-precision (64-bit)

3. Single-extended precision (≥ 43-bit, not commonly used)

4. Double-extended precision (≥ 79-bit, usually implemented with 80 bits).

Single Precision Floating Point Numbers

The single-precision number is 32 bit wide. The single-precision number has three main fields that are sign, exponent and mantissa. The 24-bit mantissa (the leading one is implicit) 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. Thus, a total of 32 bits is needed for single-precision number representation.

To achieve a bias equal to 2n−1− 1 is added to the actual exponent in order to obtain the stored exponent. This equals 127 for an eight-bit exponent of the single-precision format. The addition of bias allows

P-ISSN: 2348-6848
E-ISSN: 2348-795X
Volume 07 Issue 06 June 2020

the use of an exponent in the range from −127 to +128, corresponding to a range of 0 to 255 for single precision number. The single-precision format offers a range from 2−127 to 2+127, which is equivalent to 10−38 to 10+38

Sign: 1-bit wide and used to denote the sign of the number i.e. 0 indicate positive number and 1 represent negative number.

Exponent: 8-bit wide and signed exponent in excess-127 representation.

Mantissa: 23-bit wide and fractional component.

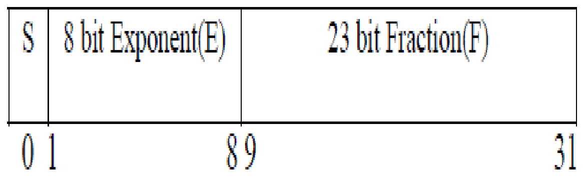


Figure 1: Single-precision floating-point number representation

The excess-127 representation mentioned when discussing the exponent portion above, is utilized to efficiently compare the relative sizes of two floating point numbers. Instead of storing the exponent (E) as a signed number, we store its unsigned integer representation (Eí = E +127). This gives us a range for Eí of 0 <= Eí <= 255.

While the 0 and 255 end values are used to represent special numbers (exact 0, infinity and denormal numbers), the operating range of E0 becomes 1 <= Eí<=• 254, thus, limiting the range of E to -126 <= E <= 127. In double-precision numbers, an excess-1023 representation is utilized.

Double Precision Floating Point Numbers

Figure shows the double precision floating point number representation. The double precision number is 64 bit wide. The double precision number has three main fields which are sign, exponent and mantissa. The 52-bit mantissa (the leading one is implicit), while an 11-bit exponent to an implied base of 2 provides a scale factor with a reasonable range.

Thus, a total of 64 bits is needed for single-precision number representation. To achieve a bias equal to 2n−1− 1 is added to the actual exponent in order to obtain the stored exponent. This equal 1023 for an 11-bit exponent of the double-precision format. The addition of bias allows the use of an exponent in the range from −1023 to +1024, corresponding to a range of 0ñ2047 for double precision number. The double precision format offers a range from 2−1023 to 2+1023, which is equivalent to 10−308 to 10+308

Sign: 1-bit wide and used to denote the sign of the number i.e. 0 indicate positive number and 1 represent negative number. **Exponent**: 11-bit wide and signed exponent in excess- 1023 representation. **Mantissa**: 52-bit wide and fractional component.

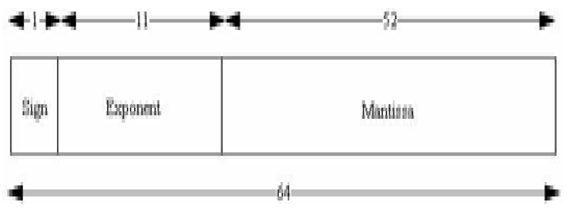


Figure 2: Double-precision floating-point number representation

In IEEE standard is that the mantissa component is always normalized. The latter implies that the decimal point is placed to the right of the first (nonzero) significant digit. Hence, the 23 bits stored in the M field actually represent the fractional part of the mantissa, that is, the bits to the right of the binary point. As aforementioned, the most significant bit of the mantissa is always equal to 1, due to binary normalization. Revisiting our original formula representation, let us now normalize the 32-bit single-precision floating-point number representation as:

(-1) S * (1 +M) * 2E^{-127}……(2)

S represents the sign (1-bit), M represents the mantissa (23-bit) and E represents the exponent (8-bit)

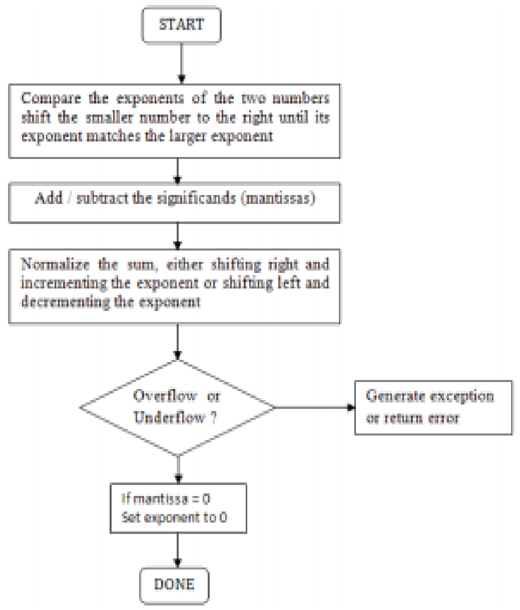

START
Compare the exponents of the two numbers shift the smaller number to the right until its exponent matches the larger exponent
Add / subtract the significands (mantissas)
Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Overflow or Underflow ?
Generate exception or return error
If mantissa = 0 Set exponent to 0
DONE

Figure 3 Flowchart for floating point addition

II. ANALYSIS OF REPORTED WORK

Floating point unit required less hardware. The low power optimizing technique multi threshold voltage (MVT) is used for reducing the power consumption of arithmetic unit. The power saving for slow High compared to typical library. Arithmetic unit has been designed to perform pack, unpack and rounding arithmetic operations on floating point number.

Error and error condition in the mathematical processing will be reported in a consistent manner regardless of implementation floating point Arithmetic unit has been designed and suitable algorithm has been developed to perform operations such as addition, subtraction, multiplication and division. The algorithm can be implemented in pipelined way to reduce the delay and increase the computation time for operation. The result between the hardware and software are matching this will clear the gap between hardware implementation and the software simulation.

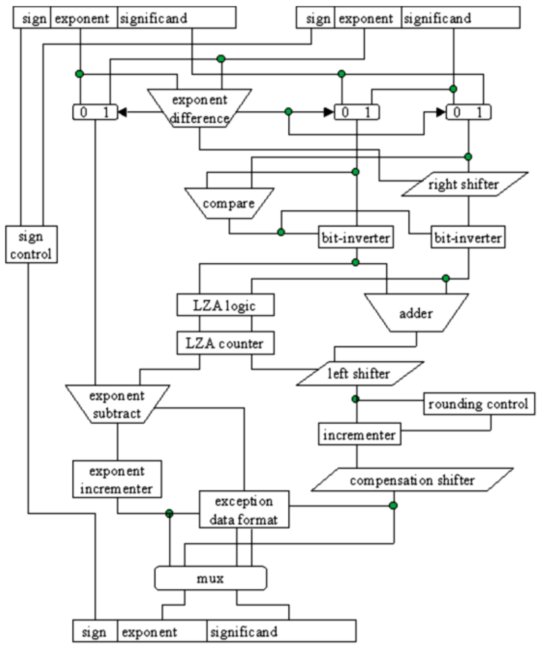


Figure 4: Floating point adder architecture

III. Adders for Floating point addition:

Carry skip adder:

Carry skip adder contains some special blocks that are useful for detecting the bits that are to be added. Here the carry will be either generated or propagated. Carry by-pass adder is also known as carry skip adder. The signal that is produced by this circuit is known as "propagation signal". The carry signal is transmitted through all the stages of blocks and the propagation time is depended on the position of carry that has been generated. If there is no need to calculate the carry then only the time which is required to compute the sum value is considered. The below block diagram has four multiplexers and is considered as 16 bit carry skip adder. The implementation of the circuit is shown below.

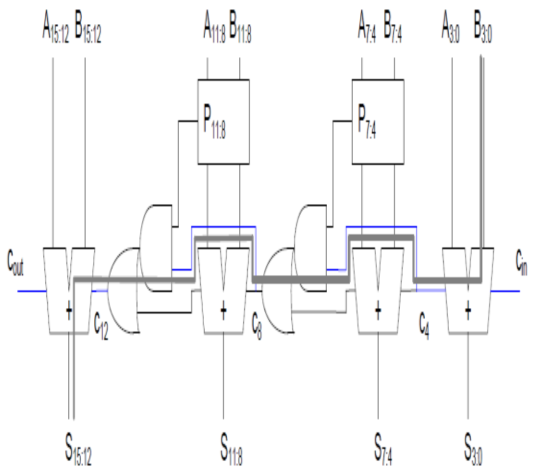


Fig.5: Structure of Carry Skip Adder

A **carry-skip adder** (also known as a **carry-bypass adder**) is an adder implementation that improves on the delay of a ripple-carry adder with little effort compared to other adders. The improvement of the worst-case delay is achieved by using several carry-skip adders to form a block-carry-skip adder. The worst case for a simple one level carry-ripple-adder occurs, when the propagate condition is true for each digit pair (ai, bi).

The *n*-bit-carry-skip adder consists of a *n*-bit-carry-ripple-chain, a *n*-input AND-gate and one multiplexer. Each propagate bit pi, that is provided by the carry-ripple-chain is connected to the *n*-input AND-gate. The resulting bit is used as the select bit of a multiplexer that switches either the last carry-bit cn or the carry-in c0 to the carry-out signal cout.

Ripple Carry Adder:

Arithmetic operations like addition, subtraction, multiplication, division are basic operations to be implemented in digital computers using basic gates like AND, OR, NOR, NAND etc. Among all the arithmetic operations if we can implement addition then it is easy to perform multiplication (by repeated addition), subtraction (by negating one operand) or division (repeated subtraction). Half Adders can be used to add two one bit binary numbers. It is also possible to create a logical circuit using multiple full adders to add N-bit binary numbers. Each full adder inputs a Cin, which is the Cout of the previous adder.

This kind of adder is a Ripple Carry Adder, since each carry bit "ripples" to the next full adder. The first (and only the first) full adder may be replaced by a half adder. The block diagram of 4-bit Ripple Carry Adder is shown here below

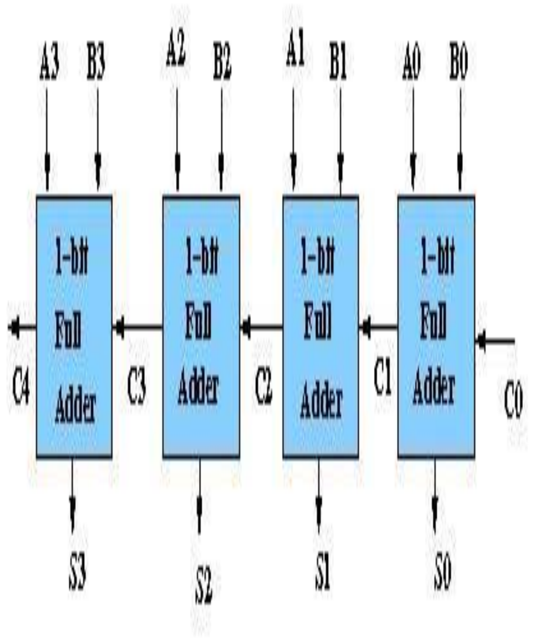


Figure 6: Structure for Ripple carry adder

The layout of ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic. In a 32-bit [ripple carry] adder, there are 32 full adders, so the critical path

P-ISSN: 2348-6848
E-ISSN: 2348-795X
Volume 07 Issue 06 June 2020

(worst case) delay is 31 * 2(for carry propagation) + 3(for sum) = 65 gate delays.

IV. Results:

The use of Verilog for modeling is especially appealing since it provides formal description of the system and allows the use of specific description styles to cover the different abstraction levels (Architectural, register transfer, logic level and test bench) employed in the design is verified through synthesis, which is done in a bottom fashion, small modules are simulated in separate test benches before they are integrated and tested as a whole.

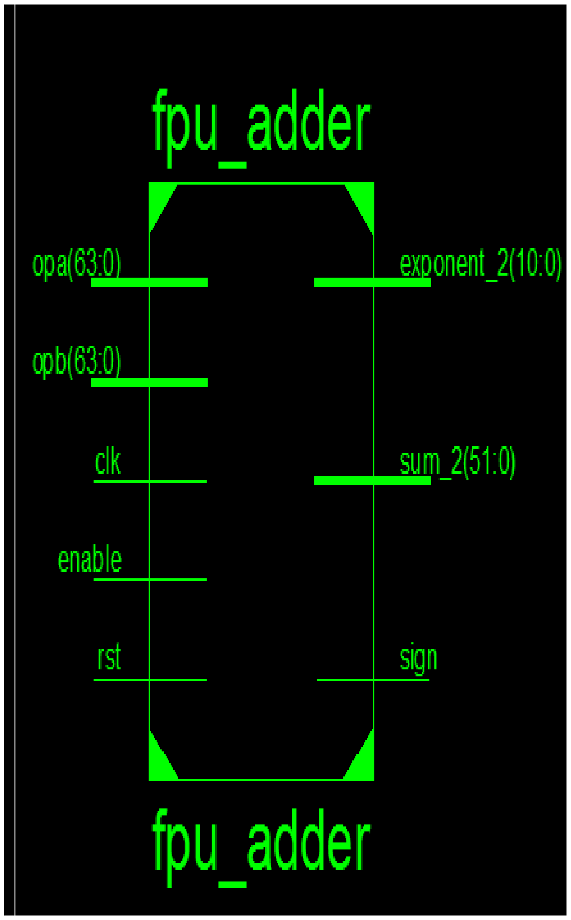


Fig 7 Top module

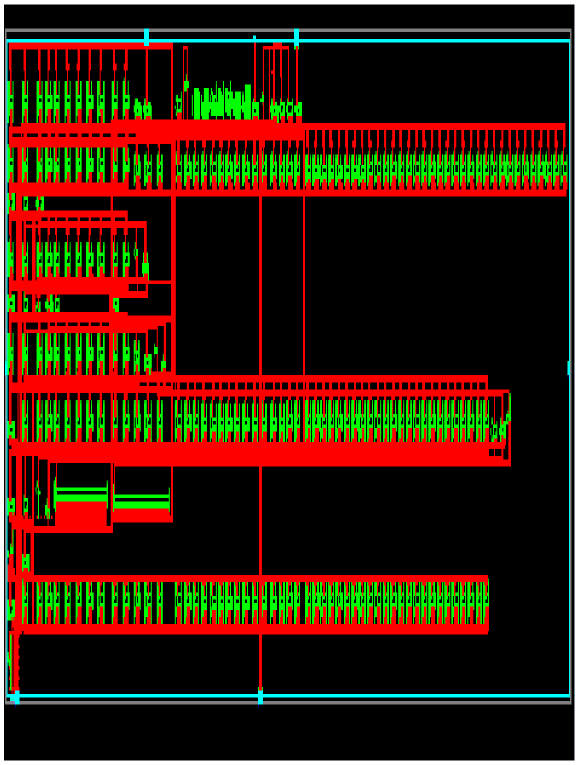
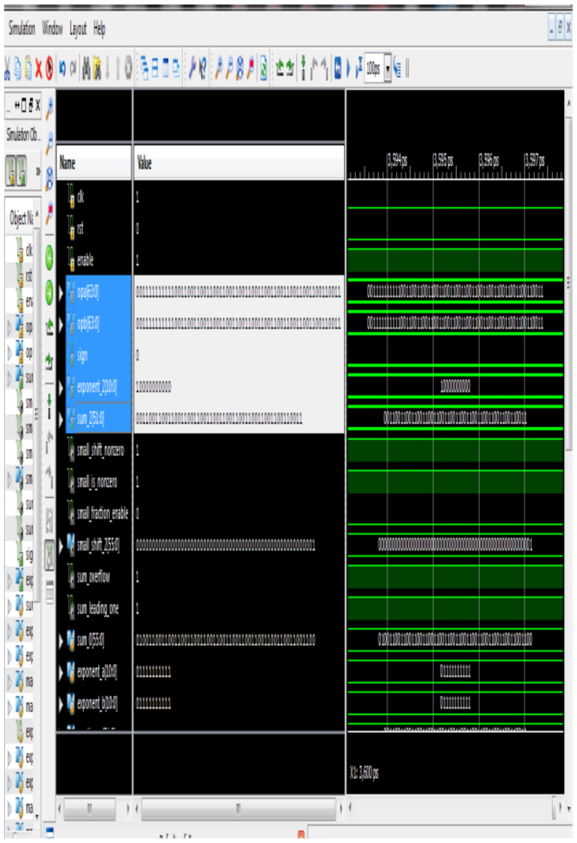


Figure 8: Diagram of RTL schematic



```
Name                   Value
clk                    1
rst                    0
enable                 1
opa[63:0]              0011111111110011001100110011001100110011001100110011001100110011
opb[63:0]              0011111111110011001100110011001100110011001100110011001100110011
sign                   0
exponent_2[10:0]       10000000000
sum_2[51:0]            0011001100110011001100110011001100110011001100110011
small_shift_nonzero    1
small_is_nonzero       1
small_fraction_enable  0
small_shift_2[55:0]    00000000000000000000000000000000000000000000000000000001
sum_overflow           1
sum_leading_one        1
sum_0[55:0]            01001100110011001100110011001100110011001100110011001100
exponent_a[10:0]       01111111111
exponent_b[10:0]       01111111111
```

Figure 9: Simulation waveform of RCA

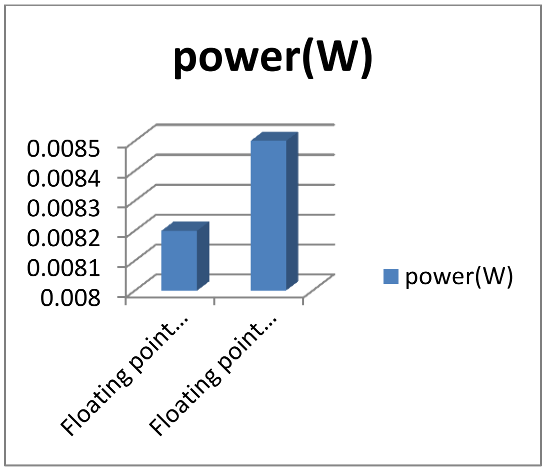


Figure 10: Comparison of Power Data graph

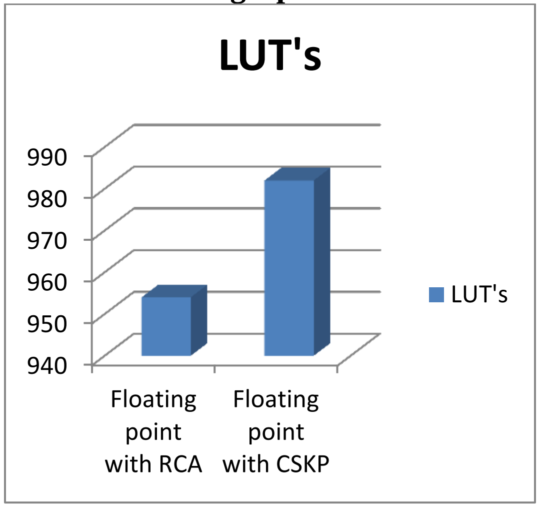


Figure 11: Comparison of Area Data graph

Parameter	Floating point with RCA	Floating point with CSKP
LUT's	954	982
Power(w)	0.0082	0.0085

Table 1: Comparison table.

V. Conclusion:

This paper shows review of IEEE floating point unit (FPU) which will perform addition function on 64 bit operand that uses the IEEE-754 standard. Floating point numbers representation can support a much wider range of values than fixed point representation. In this paper Comparison is carried out between the carry skip adder based fpu and the proposed ripple carry adder. In the proposed design area and power are reduced whose values are tabulated in table 1. The designed arithmetic unit operates on 64-bit operands'. It can be designed for 128-bit operands to enhance precision. It can be extended to have more mathematical operations like division, square root, and trigonometric functions. Floating point unit is used in DSP Processors like TMS 320X and embedded systems. The work is to implement and analyses floating point adder operation and hardware module were implemented using VERILOG and synthesized using Xilinx ISE suite

References:

[1] Charles Farnum, "Compiler Support for Floating-Point Computation" Software Practices and Experience," pp. 701-9 vol. 18, July 1988.

[2] D. Goldberg, "What every computer scientist should know about floating-point Arithmetic," pp. 5-48 in ACM Computing Surveys vol. 23-1 (1991). Logic Utilization Used Available Utilization Number of slice latches 25 13,824 1% Number of 4 input

LUTs 1604 13,824 11% Logic Distribution Number of occupied slices 831 6912 12%
Number of slices containing only related logic 831 831 100% Number of slices containing unrelated logic 0 831 0% Total number of 4 input LUTs 1605 13,824 11% Number used as logic 1604 Number of bonded IOBs 296 404 73%
Number of GCLKs 1 4 25% Number of GCLKIOBs 1 4 25% 463

[3] Guillermo Marcus, Patricia Hinojosa, Alfonso Avila and Juan Nolazco- Flores " A Fully Synthesizable Single-Precision, Floating Point Adder/Subs tractor and Multiplier in VHDL for General and Educational Use," Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, Dominican Republic, Nov.3-5, 2004.

[4] IEEE Computer Society (1985), IEEE Standard for Binary Floating- Point Arithmetic, IEEE Std 754-1985.

[5] Jim Hoff; "A Full Custom High Speed Floating Point Adder" Fermi National Accelerator Lab, 1992.

[6] John Thompson, Nandini Karra, and Michael.J.Schulte "A 64-bit decimal floating-point adder," Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design (ISVLSI'04) .

[7] W. Kahan "IEEE Standard 754 for Binary Floating-Point Arithmetic," 1996

[8] Michael L. Overton, "Numerical Computing with IEEE Floating Point Arithmetic," Published by Society for Industrial and Applied Mathematics, 2001.

[9] D. Narasimban, D. Fernandes, V. K. Raj , J. Dorenbosch , M. Bowden, V. S. Kapoor, "A 100 Mhz FPGA based floating point adder", Proceedings of IEEE custom integrated circuits conference, 1993.