IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 6, Issue 1, Ver. I (Jan. -Feb. 2016), PP 60-66
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Synthesis and Simulation of 64 Bit Floating Point Unit with Square Root Operation

Mohit Kumar Goswami[1], Anushree[2]
Assistant professor, E&C, Hardayal Technical Campus, Mathura, INDIA [1]
Assistant professor, E&C, Hindustan College of science & technology Mathura, INDIA2

Abstract: Floating point numerical operations being widely used in large set of signal processing computation, scientific, commerce and finance calculation.This implementation involves approach for computing four floating pointnumerical operationswith square root operation also. In top-down design approach, four arithmetic modules: addition/subtraction, multiplication, division, and square root are combined to form a double precision floating point unit.In this paper the proposed design compliant with IEEE-754 format a 64 bit floating point unit is designed which is also handling rounding, overflow, underflow&various exceptions for each operation.Verilog and Questa-Sim design tool used for synthesis and simulation continuously for precise results.
Keywords: Double precision, Floating point unit, Square root, Hardware description language (HDL), IEEE-754

I. INTRODUCTION

It is most proficient method for speaking to genuine no. In PCs. Gliding point expansion is most broadly utilized operation as a part of DSP/Math processors, robots, air activity controller, advanced computers [1]. Digital number-crunching operations are vital in the outline of computerized processors and application particular frameworks. Math circuits frame an essential class of circuits in advanced frameworks. Progresses in the expansive scale incorporation (VLSI) circuit innovation, numerous intricate circuits have turned out to be effortlessly feasible today. The customary PC math systems, as well as the whimsical ones are worth examination in new outlines. In the advanced sign preparing, applications including more exactness and range coasting point representation is utilized. One of the Applications is sound preparing. A skimming - point math and rationale unit, for the most part term as Floating Point Unit is a part of a PC framework extraordinarily intended to complete operations on coasting point numbers. Skimming point portray a framework for speaking to numbers that would be too long or too short to be re-exhibited as whole number. Gliding point re-presentation held its determination and precision contrasted with the altered point number framework re-presentation. The standard accommodates numerous firmly related configurations, contrasting in just a couple points of interest, similar to single exactness, twofold accuracy, two fold developed [3]. The square root is being examined here usage of effective square root administrator for figuring square foundation of twofold drifting point information, it bolster continuous sub-current and four adjusting modes. Status banner utilized for taking care of special cases are not set.

The IEEE 754 arrangements standard determines:
Basic and developed gliding point number configurations
Add, subtract, duplicate, partition, square root, leftover portion, and think about operations
Conversions in the middle of whole number and skimming point designs
Conversions between diverse gliding point groups
Conversions between fundamental organizations coasting point numbers and decimal strings
Floating-point special cases and their taking care of, including IEEE 754 standard presents two distinctive drifting point groups, Binary trade design and Decimal between change position. the IEEE754 single exactness parallel organization representation; it comprises of an one piece sign (S), an eight piece type (E), and a twenty three piece part (M or Mantissa).If the example is more noteworthy than 0 and littler than 255, and there is 1 in the MSB of the huge then the number is said to be a standardized number [2].

II. BACKGROUND

In the IEEE Standard 754 for Binary Floating-Point Arithmetic different work has done in this.

The IEEE (Institute of Electrical and Electronics Engineers) has created a Standard to characterize coasting point representation and math. In spite of the fact that there are different representations, it is the most widely recognized representation utilized for drifting point numbers.

The standard brought out by the IEEE come to be known as IEEE 754.

With regards to their exactness and width in bits, the standard characterizes two gatherings: fundamental and broadened group. The augmented arrangement is execution subordinate and doesn't concern this proposed work.

The essential organization is further isolated into single-accuracy configuration with 32-bits wide, and twofold exactness position with 64-bits wide. The three fundamental segments are the sign, type, and mantissa.
[8] Use of Look-up Table,Vedic Approach: Dvanand Yoga Sutra, Paper and Pencil Method are more various square root algorithms have studied and implemented. By utilizing single accuracy and twofold exactness Floating point Adder/subtraction, multiplier, divider and a great deal more modules had outline and execute. The square root calculations and usage have been tended to principally in three headings: Newton-Raphson technique SRT-Redundant strategy and Non-Redundant system.

III. IMPLEMENTATION

The double precision floating point unit performs addition, subtraction, multiplication and divide square root operations. The block diagram of double precision floating point. It consists of 5 sub blocks. The blocks are
1 FPU add
2 FPU sub
3 FPU mul
4 FPU div
5 FPU sqrt

3.1 Addition and subtraction

In this part we are representing how to add and subtract two 32-bits floating point numbers. For example, there are two floating point numbers as follows such that both these floating point numbers can represent in the following form. And there arithmetic operation (i.e. add/sub) can be easily understood by seeing the following process of addition and subtraction. [(-1) ^s1 * f1 * 2^e1] +/- [(-1) ^s2 * f2 * 2^e2]
Suppose e1>e2, then we can write it as, [(-1) ^s1 * f1 * 2^e1] +/- [(-1) ^s2 * f2' * 2^e2] (where, f2' = f2/ 2^ (e1-e2))
The result is-[(-1) ^s1 * (f1 +/- f2') * 2^e1] [6][4]

3.2 Multiplication

Suppose there are two floating point numbers as follows such that both these floating point numbers canbe representingthe following form. And there multiplication can be easily understood by seeing the following process of multiplication.
[(-1) ^s1 * f1 * 2^e1] * [-1) ^s2 * f2 * 2^e2] = (-1) ^ (s1 xor s2) * (f1 * f2) * 2^ (e1+e2)
Since 1 < (f1 * f2) < 4, result may need to be normalized. Conceptually, multiply operation is somewhat like simpler. If we look at two numbers-
(-1)^s1 * f1 * 2^e1 – 1st floating point number.
(-1) ^s2 * f2 * 2^e2 – 2nd floating point number.
The sign of the result is exclusive or of signs - if both are 1's or both are 0's – result will be zero. Fraction part is the product of two fractions and the exponent gets summed. There is no initial alignment which is required. But off course, we need to do normalization and rounding, because the product (f1 * f2), which are individually in the range 1 to 2. The resulted product would in range 1 to 4. That means the larger side, it may exceed two and it may require one small adjustment. If it exceed by 2, we may divide it by 2 and increment the exponent part [5].

3.3 Division

Suppose there are two floating point numbers as follows such that both these floating point numbers can represent in the following form. And there division can be easily understood by seeing the following process of addition and subtraction.
[(-1) ^s1 * f1 * 2^e1] / [(-1)^s2 * f2 * 2^e2] = (-1) ^ (s1 xor s2) * (f1 / f2) * 2^ (e1 – e2).
Since, 0.5 < (f1 / f2) < 2, result may need to be normalized.(Assume f2 not equal to 0). Divide is similar. Two numbers are taken in same way.
If we look at same numbers-
[(-1) ^s1 * f1 * 2^e1] – 1st. floating point number.
[(-1) ^s2 * f2 * 2^e2] – 2nd floating point number.
There division can be brought straight away. SIGN is written exactly in same way. Sign of result is exclusive or of signs. If both are 1's or 0's result will be zero. The fraction gets divided and exponent gets subtracted. Now, ratio of fractions would be in the range-
0.5 < (f1 / f2) < 2.
So it will not exceed 2, because each one is in the range 1 and 2. But it can become small; hence we may need to normalize. So before doing all above process, we had to check, whether f2 is '0' or not [4].

3.4 Square root module

*1) Sign Bit:*Sign bit of the result is same as the sign bit of the originalnumber.

2) Exponent Computation: Exponent of the result depends on the biased exponent of the number. If biased exponent is odd, then 1023 is added to it and final sum is right shifted (divide by 2 operation).

$$E_r = (E_a+1023)/2$$

If biased exponent is even, then 1022 is added and final sum is right shifted (divide by 2 operation). In addition, shift lag is set to indicate that the mantissa should be shifted to the left by 1 bit before computing its square root

$$E_r = (E_a+1022)/2$$

3) Mantissa (Square Root) Evaluation:The block of code which carry out square-root computation is based on iterative approach where it deals with two registers namely TEMP and ANS of 56-bit wide andof 55 bit wide.Consider an example of evaluating square root of 16 $(10000)_2$.

InstateTEMP as 0000… 0000 and ANS as 0100… 0000.

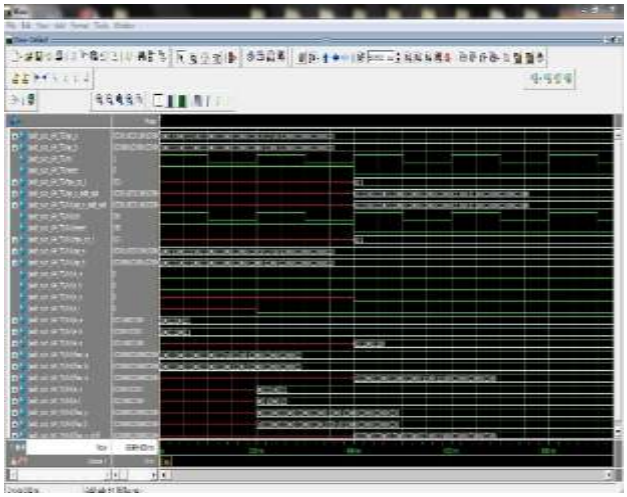
The emphasis procedure is completed. On first emphasis, TEMP is stacked with mantissa and after that is contrasted and ANS, shift operation is done relying upon the examination results. In the event that it is more prominent or equivalent to ANS, the substance of TEMP is Subtracted from those of ANS, moved to one side, and put away in TEMP. The substance of ANS are moved to one side by one piece beginning from the present pointer position Then, a 1 is inserted in the current bit position in ANS. Note that in each iteration, a pointer points to the current bit that will be replaced in ANS.

In the event that TEMP is not as much as ANS, its substance are moved to one side and put away in TEMP. The substances of ANS are moved to one side by one piece beginning from the present pointer position. At that point, a 0 is embedded in the present piece position bit in ANS.

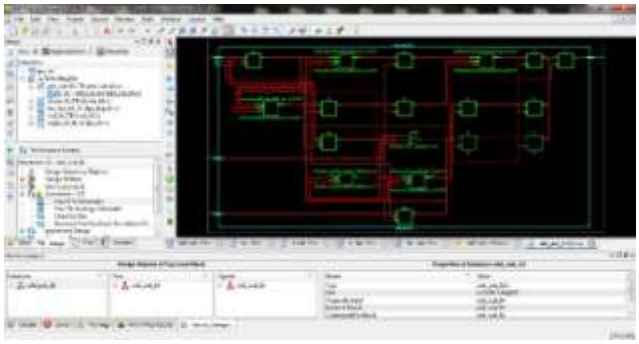
After the last cycle, the substance of ANS, except for the two slightest noteworthy bits are the bits which considered as conclusive result.

IV RESULTS

4.1 Addition and subtraction module



Simulation window- addition and subtraction module

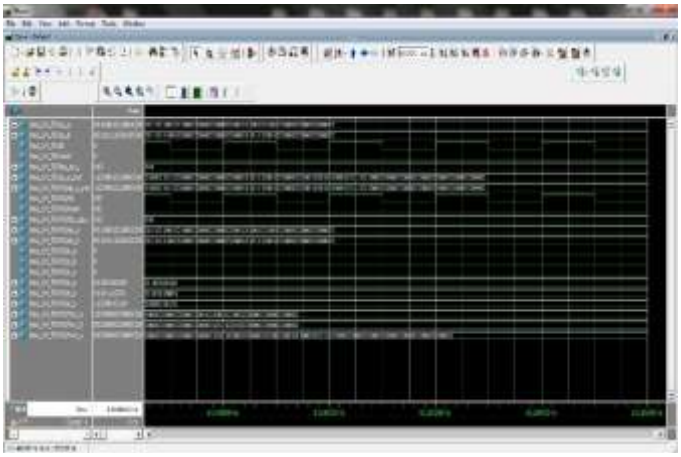


RTL window- addition and subtraction module

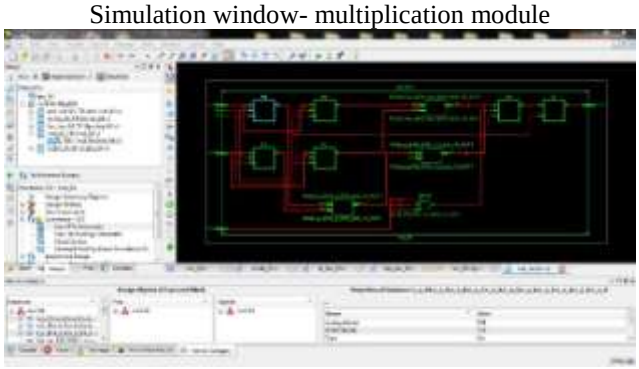
4.2 Multiplication module

Inputs op_a and op_b are of 32-bit binary floating point.

op_a = 10110010010010000000000000000
op_b = 00110100010000000000000000000
After giving the select line value, fpu_op_i = 010. We are getting output, and we will the output of size 64-bits floating point number.
Op_o_mul=1010100110001001000000000000000000000000000000000000000000000000



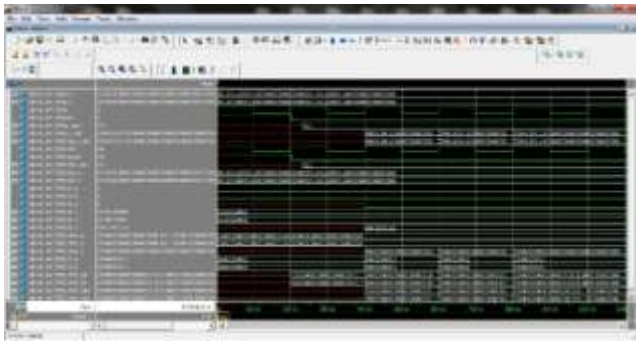
Simulation window- multiplication module



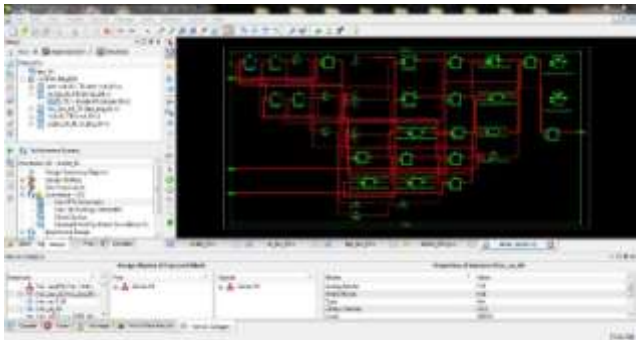
RTL Window- multiplication module

4.3 Division module

Inputs op_a and op_b are of 32-bit binary floating point.
op_a = 00110111000011000000000000000
op_b = 10110010000110000000000000000
After giving the select line value, fpu_op_i = 011. We are getting output, and output will be of 32-bits floating point number.
Op_o_div=11000100000000000000000000000001



Simulation window-division window

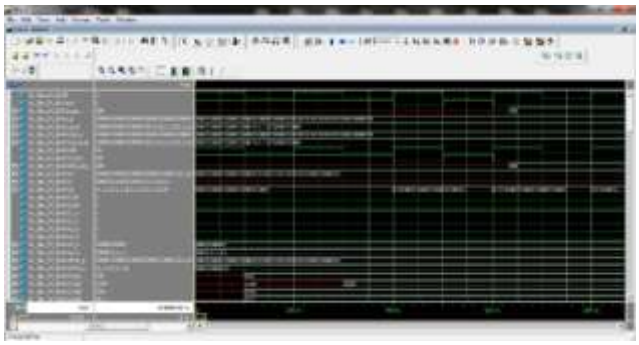


RTL Window-division module

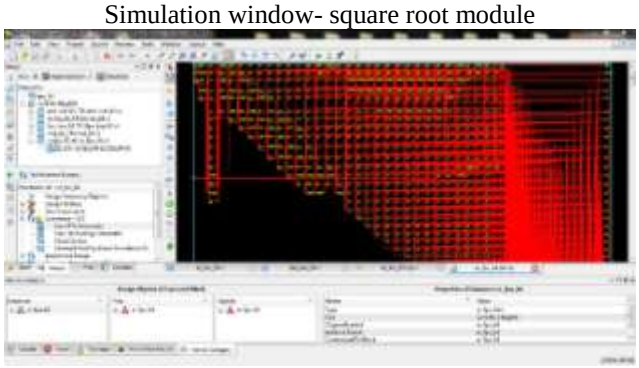
4.4 Square root module

Input op_a are of 32-bit binary floating point. op_a = 00001000010111001010100000000

After giving the select line value, fpu_op_i = 100. We are getting output, and output will be of 32-bits floating point number. Op_o_sr = 00100011110110111101 1



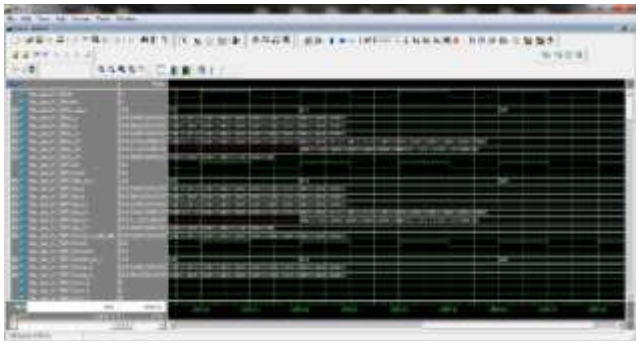
Simulation window- square root module



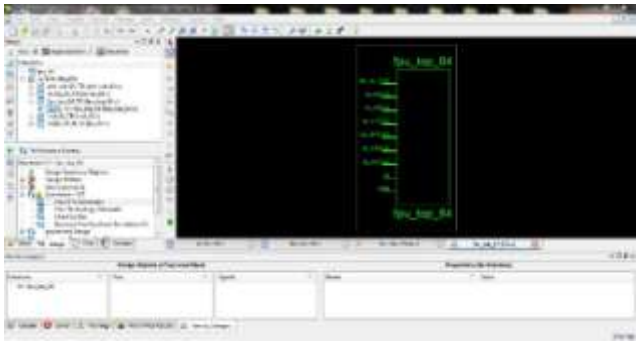
RTL window-square root module

4.5 Top level module

Subsequent to giving inputs op_an and op_b, to every one of the four sub modules, i.e. Expansion/Subtraction module, Multiplication module, Division module and Square root module, we will the required result by applying qualities to a specific selecting pin.



Simulation window- top level module



RTL window- top level module

Figure representing the register transfer level window of a Top level module. In this particular register transfer window we are now able to see the detailed interconnections between the several transistor blocks of addition/subtraction module

V. CONCLUSION

Till now, we are getting just four functionalities: expansion, subtraction, duplication, and division, from already proposed outlines

In this particular design we had further introduced one more functionality of square root, such that this design will be now capable of doing five operations: addition, subtraction, multiplication, division, and square root of double precision numbers. Each module is independent from each other. The modules are realized and validated using verilog simulation in the questasim and synthesis using Xilinx ISE Design Suite 13.3.

More work can be done in this particular project regarding power consumption. We can simply use the concept of clock gating, to resolve the problem. Such that only the required sub module will perform its operation, and all other sub module will remain off, on applying the values select pins.

References

[1] PreethiSudhaGollamudi, M. Kamaraju‘’Design of High Performance IEEE- 754 Single Precision (32 bit) Floating Point Adder Using VHDL’’ inInternational Journal of Engineering Research & Technology (IJERT)Vol. 2 Issue 7, July – 2013.

[2] Addankipurna Ramesh,Ch.Pradeep “Fpga Based Implementation of Double PrecisionFloating Point Adder/Subtractor Using Verilog”International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012).

[3] SARC International Conference, 30th CHAITANYA A. KSHIRSAGAR, P.M. PALSODKAR “An FPGA Implementation of IEEE - 754 Double Precision Floating Point Unit Using Verilog” Proceedings of 4thSARC International Conference, 30th March-2014, Nagpur, India.

[4] Swathi. G. R, Veena. R“Design of IEEE-754 Double Precision Floating Point Unit Using Verilog”in International Journal of Engineering Research & Technology (IJERT)International Journal of Engineering Research & Technology (IJERT)Vol. 3 Issue 4, April – 2014.

[5] KusumaKeerthi “Design of High Performance Single Precision Floating Point Multiplier in (ITSI-TEEE)ISSN (PRINT) 2320 – 8945, Volume -2, Issue -1,2014.

[6] Ravi Payal “Simulation and Synthesis Model for the Addition of Single Precision Floating Point Numbers Using VERILOG” International Journal of Engineering Research & Technology (IJERT)ISSN: 2278-0181Vol. 2 Issue 9, September – 2013.

[7] G. Even, S. M. Mueller, and P.-M. Seidel, "A dual precision IEEEfloating-point multiplier," Integer.'. VLSI J, Vol. 29, no. 2, pp. 167-1802000.

[8] Li, Y., Chu, W., “Implementation ofSingle Precision Floating Point Square Rooton FPGAs”, In Proc. 5th IEEE Symposium On Field Programmable Custom ComputingMachines 1997, pp .226-232.

[9] IEEE standard for floating-point arithmetic(IEEESTD 754-2008),revision of IEEE-754 in august (2008).

[10] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE-754, 1985.

[11] UshasreeG, RDhanabalSarat Kumar Sahoo “Implementation of a High Speed Single Precision Floating Point Unit using Verilog” *International Journal of Computer Applications (0975 – 8887) National conference on VSLI and Embedded systems 2013.*

[12] NajibGhatte, ShilpaPatil, Deepak Bhoir “Double Precision Floating Point Square Root Computation” International Journal of Engineering Trends and Technology (IJETT) – Volume 13 Number 6 – Jul 2014.

About the Author

Mohit Kumar Goswami, B.Tech. in Electronics & Communication from CET Moradabad in 2012,M.Tech.pursuing in VLSI from HCST, Mathura. He has an interest in VHDL, VLSI Design, and Embedded System. Also he has published a paper in MANET in STM journal.

Anushree, B.tech in E&C and M.tech in VLSI design from HCST Mathura. she has published more than ten papers in IEEE conference, national and international journals. Her area of research is VLSI interconnects, DSP& low power.