Design of High Throughput and Low Latency Double Precision Floating Point Arithmetic Unit for Space Signal Applications

Ponduri Sivaprasad [1], Dr.V.Anandi [2], Dr.Satyanaryana Murthy [3]

[1]Research Scholar, Visvesvaraya Technological University, Belagavi, Karnataka 590018, India
[2]Associate Professor, Ramaiah Institute of Technology, MSRIT Post, Bangalore-560054, Karnataka, India
[3]Principal, Gonna Institute of Information Technology and Sciences, Visakhapatnam,Andhra Pradesh 530053, India

Abstract
For space signal processing systems, reliability, accuracy, and performance are major concerns for the detection of accurate phase estimation, for and most of the functionality of the system, the data is acquiring high speed and supervising continuously for validation of correct data. For more accuracy, fixed points arithmetic operations have got lot of data losses and single-precision floating point operations also has data losses. All existing double-precision floating point arithmetic operations utilizes dual rail coding to perform complete detections and also required the circuit to receive acknowledge on completion execution and it leads to worst-case delay irrespective of the actual completion time. With help of modified double precision floating point operations, we can obtain more reliability by using memory based synchronization architecture and high accurate phase detection. In space applications, milli degree estimation is major challenge and plays important role, in order to estimation milli degree, the Double Precision Floating Point (DPFP) based arithmetic operations are designed using Verilog hardware description language and synthesized with help of Xilinx Design Suite 14.7 ISE software tool and finally implemented on Virtex-5 FPGA development board. All arithmetic operations use ternary logic at lower level module design to optimize area and latency. The proposed architecture of double precision floating point arithmetic operations is good enough in terms of power optimization, high speed, optimal delays, hardware utilizations (Slices and LUT's) and smaller-sized edge device. The synthesized results show that proposed DPFP based ALU design for estimation of milli-degree reduces the overall latency to 23%, throughput is improved by 13% and power consumption is reducing to 31% as compared to existing works.

Keywords:
Floating point based ALU, Ternary logic, Signal processing, milli-degree estimation, FPGA.

1. Introduction

In general, division has a substantially larger latency than other arithmetic operations. Despite their infrequent use, in [1] division is focused and have a considerable impact on total different delays. As a result, high-performance divider design has become a hot topic in high throughput and speed computational operations. Efficient parallel divisions are necessary for application of 3D graphics in computers, image/video/signal processing applications that rely heavily on division computational [2] [6]. It has effective parallel division's technique is often widely required for optimization of power optimization in terms of consumptions and latency in image/signal processing and multimedia applications [7]. An Improving of performance by lowering pipeline delay is more challenging than expanding hardware size in current microprocessors. The primary applications for latest complex operations which are used in semiconductor area is that hardware utilization grows rapidly while clock speeds get congested. Furthermore, reducing pipeline delay might cause for elimination of various hardware logic elements. For specialised ASIC processor, like GPU and these processors utilizes more number of registers per bits in parallel, it can reduce the number of register bits required by minimising parallel latency. Furthermore, because the pipeline latency decision may have an impact on the overall architecture, _xing the pipeline latency of a specific unit may be required [9] describes a Taylor series expansion-based high-radix pipelinable division algorithm. In these algorithms having more LUT than other techniques, variation to this pipelinable division technique was presented in [4] to drastically reduce the LUT size. In contrast to [4,] [5] suggests a further adjustment for DPFP values that greatly decreases chip space, particularly by optimizing number of LUT size from 62 KB to 3.7 KB using taylor series [10]. The performance and dependability of space-based systems are critical in DSP applications. To ensure the authenticity of obtained data, perfect functionality of system should often repeatedly be testing and monitoring are necessary. We create a new digital circuit in this study that can detect minute phase deviations in time-varying analogue signals. The angle

Manuscript received September 5, 2022
Manuscript revised September 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.9.105

estimations for

a single channel input vs a known signal system, also for two channels with data collected simultaneously. In space borne systems, phase shift data can be utilised to either confirm correct system performance or highlight potential problems. Data capture system with giga samples per second (GSamp/s). NASA Surface Water Ocean Topography was created mainly for space related applications. Microwave waves will be used to monitor the Earth's environment by a number of planned NASA remote sensing satellite projects. The use of signal phase to characterise obtained data is a typical feature of these missions [1]. In each mission will use signal phase in a unique way, the fact that their research outcomes are dependent on it emphasises the importance of precise signal phase measurement [2].

2. Related Work On Floating Point Operations

Explaining research chronological, including research design, research procedure (in the form of algorithms, Pseudocode or other), how to test and data acquisition [5]–[7]. Floating point implementation on a Field Programmable Gate Array (FPGA) is a newer field that has seen significant achievements, due to FPGA development advantages like speed and latency and low cost than ASIC design. [2] Proposed DPFP and Single Precision Floating Point (SPFP) arithmetic operations, which were subsequently validated on an FPGA for image/signal processing HDL programming scripts. This study's main purpose is to look at the area and timing of single and DPFP and MAC units. On the Spartan 6 FPGA, the described model is both simulated and implemented [1]. The complex module's floating-point multiplication, i.e. mantissa multiplications, was performed using an efficient Karatsuba technique, allowing for more efficient use of the in-built DSP48E blocks on Xilinx Virtex-5[2]. The suggested technique beats current solutions in terms of throughput; additionally, due to changes in the DSP slice multiplier architecture, a few Altera FPGAs achieve higher clock rates [3]. The framework's floating-point multiplier takes care of overflow, underflow, and rounding. Traditional floating-point multipliers are written in Verilog programming language, synthesised, and verified using the ISE Simulator [4]. They are based on Vedic mathematics and are written in Verilog programming language.

In [5] proposed semi-parallel iterative design is built and validated using FPGA, and the findings show that the proposed work outperforms with less latency when compared to fixed point multipliers with double precision presented in this study work [5]. Based on the examination of numerous look-up table-based algorithms in the previous work, the fundamental blocks of algorithms such as multiplier(s) and adder(s)) are re-engineered to enable the advantages of area and timing to operate efficiently. We use Wong and Goto's algorithms as a foundation for our optimization approaches, and the suggested algorithm's performance is compared to that of other algorithms in the literature based on performance and scalability measures. The Wong and Go to division algorithms' precision, i.e. latency area, is improved by 26.94 percent [6]. Binary addition is the most basic function in arithmetic modules, and the adder is the processor's basic arithmetic component. A full adder is an important feature in DSP architecture, microprocessor, and microcontroller applications, as well as data processing modules. The most common purpose of parallel multipliers is to achieve faster processing rates at the expense of improved area efficiency [7]. This research paper presents an adaptive design of a speed, power optimized multipliers using shift and add techniques. In [8] also covers the design of the Braun multiplier and Wallace multiplier and tested in RTL Compiler using Cadence, as well as simulation and the development of test circuits for each module that makes up the multiplier. Additionally, with the assistance of the Cadence tool [8]. MOSFETS, which operate in a weak inversion zone, are frequently utilised in this design to achieve low power dissipation. On a 0.5V supply, the multiplier is built up of four Exponential approximation circuits. Tanner tool produces findings and simulations using 180nm technology [9].

An array multiplier and a Booth multiplier were used to create the Finite Impulse Response Filter, and the results were compared to various constraints. The suggested filters are written in Verilog HDL and run on the Xilinx 14.7 ISE platform. In the area of delay, there has been progress [10]. The design's functionality is tested using simulation [11] multiplier is a more sophisticated version of the tree-based multiplier. The Wallace tree multiplier uses the Carry-Save addition method to reduce latency. Vedic mathematics is used to create the Vedic multiplier. In Vedic multiplication, there are 16 tantras, commencing with "Urdhva Tiryakbhyam" [12]. In VLSI systems, cutting the consumed power requires reducing the minimum supply needs. This study offers a high-performance capacitance multiplier that can run on as little as 0.25 V of power. Furthermore, using the same CMOS method, The output currents generated are 98 times more than the quiescent [13]. A stopband filter with a complementary-defected ground structure is presented in this study (DGS)[15]. The IEEE 754 standard is used to express binary floating-point numbers [17]. The Karatsuba multiplication method is

written in Verilog HDL and does not require a pipelined design. This multiplier can do significant multiplication, sign bit operations, and exponent arithmetic. With an 8-clock cycle delay, the system uses three steps of pipelining [16].

3. Proposed Double Precision Floating Arithmetic Operation For Estimation Of MILLI Degree

Multiplication, division, subtraction, and addition are

3.1 Double precision floating point Multiplication

The standard for floating-point arithmetic [1] includes three upgraded operations as new proposed operations: augmented Addition, augmented Subtraction, and augmented Multiplication. This is these homogeneous functions take two binary floating-point values as input and return two FP numbers in the 754 format as the input. a problem that our multiplication algorithms do not seek to solve

Algorithm 1: Double precision floating point multiplication

: Inputs: A and B operands of 64 bits,: Output: double precision product

Step1: Separate mantissa's (Ma and Mb), exponents (Ea and Eb) and sign bits from given 64 bits operands

Step2: Add exponents and bias value i.e 1023 is subtracted from its value. i.e compute Eout = Ea + Eb - Ebias. (Ebias = 1023),

Step3: Implicit bit addition at MSB side of each mantissa's ({1'b1,Ma}, {1'b1, Mb}). M=53 bit, **Step4:** Multipliy mantissa's like normal integer multiplier and output is 106 bit.

Step5: Keep MSB 53 bit in one variable i.e Mout= output[105:54],

Step6: Check MSB of it. (Mout[53]), whether MSB bit is logic 1 or logic 0,

Step7: If it is one then Efinal= Eout + 1 is called post normalization

Step8: Finally the product is Mfinal= {Mout << 1} is called post normalization.

In exiting floating point format, directly combining sign bit, exponents and mantissa's to final product of multiplications as shown in Fig. But this process will consume more delay, area and power consumption and not suitable for complex operations like estimation of milli degree for space signal applications or any other complex applications. The mantissa's output are measured with help normal integer multiplication of both operands as shown step4 in algorithm 1. At last mantissa's bits rearranged through exponents value and then combine to produce final product which is in terms of double precision floating point format as shown in Fig. After successful completion of all arithmetic operations using double precision floating point format as shown in Fig, the following points are summarized. When the computation is finished, the input operands are divided into sign, exponent, mantissa's. Each segments perform a calculation appropriate for the operation, such as addition, subtraction, division, and multiplication.

Sign: The sign of both inputs is determined by performing an XOR operation on the two input signs.

Exponents: If an exponents differ; then larger exponent value is subtracted from lessor. The mantissa bits are shifted to the right for the input with a smaller exponent to align the two integers to the same decimal point.

Mantissa: Mantissa measures the value of the Mantissa's help fixed point operations. The Mantissa bit addition/subtraction operation may produce a result that is one bit greater than the Mantissa bit of both inputs. To acquire exact results, we increased the number of Mantissa bits for both inputs, then performed Mantissa addition/subtraction based on the Mantissa calculation findings, whether MSB is 0 or 1, and a normalizer is not necessary if MSB is zero. If MSB is 1, the normalizer moves the previously calculated Exponent and Mantissa bits to obtain the final combined values. At last, each measured values are combined into one common floating point format to get final results of each arithmetic operation output

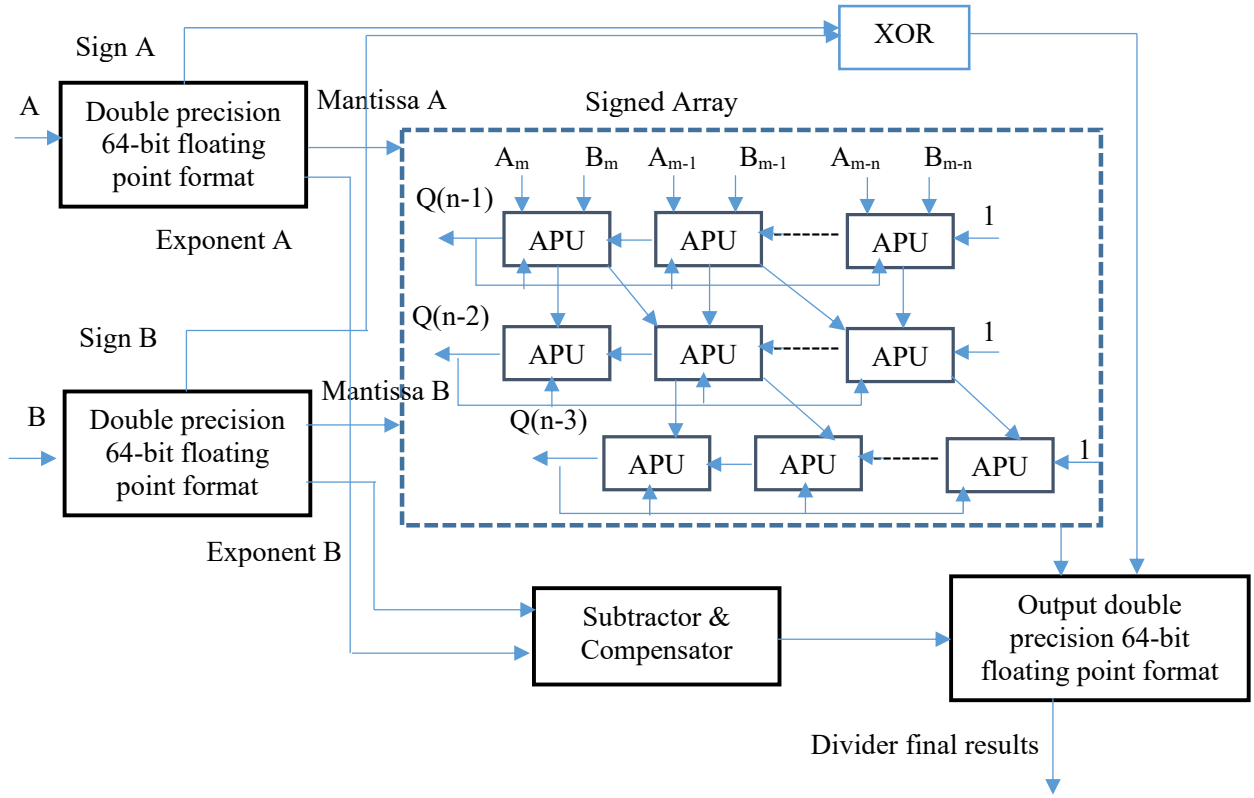

Fig.1. Proposed internal architecture of divider 64bit double precision floating point operation using signed array

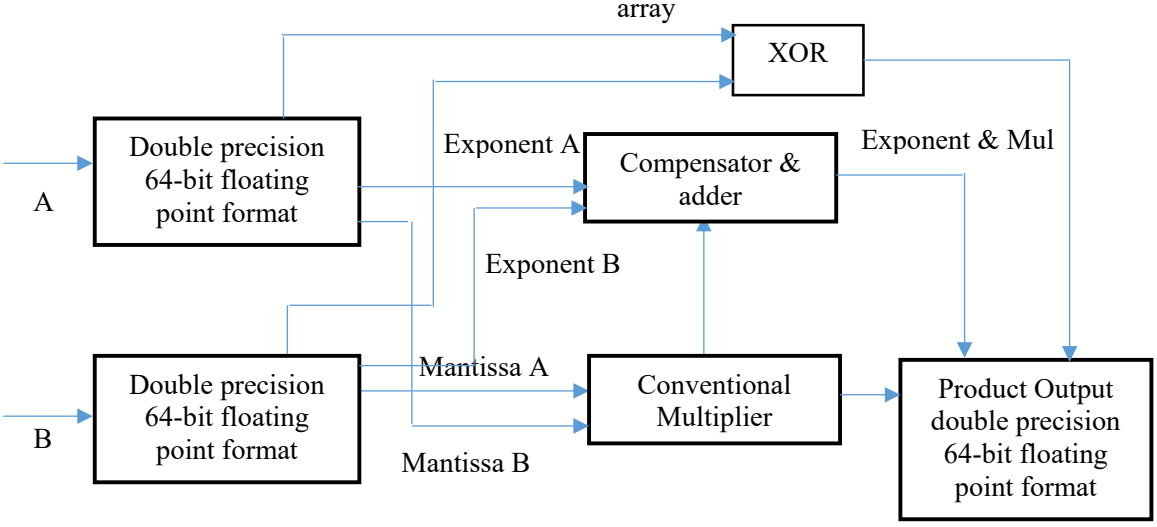

Fig.2. Proposed architecture of double precision floating point multiplication

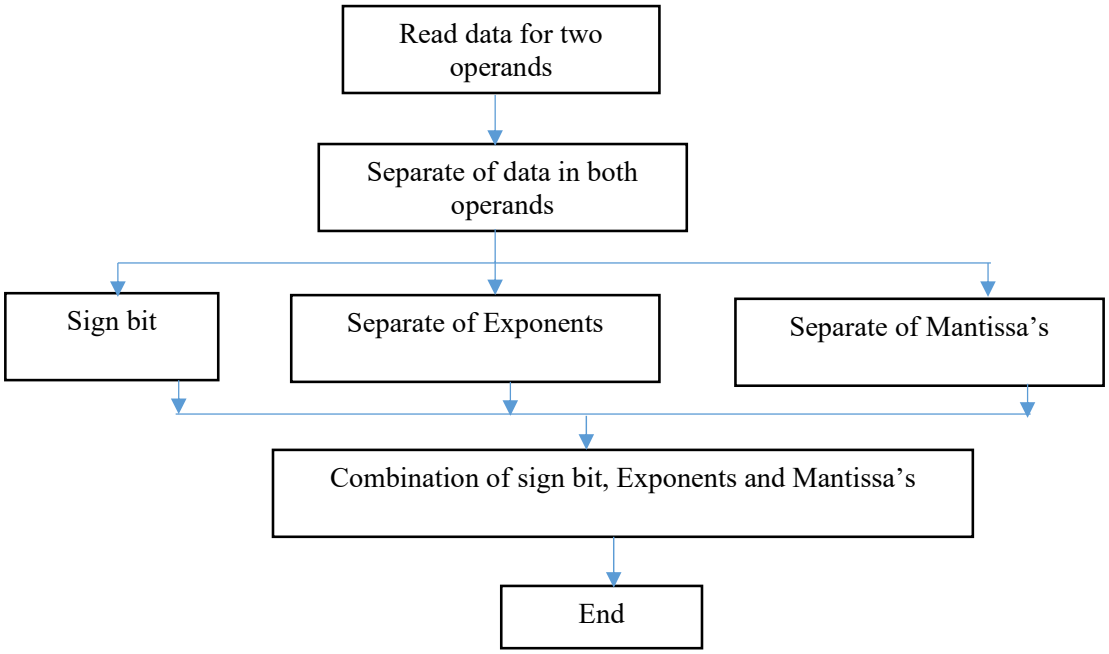

Fig.3. Existing floating point operations process [8, 12].

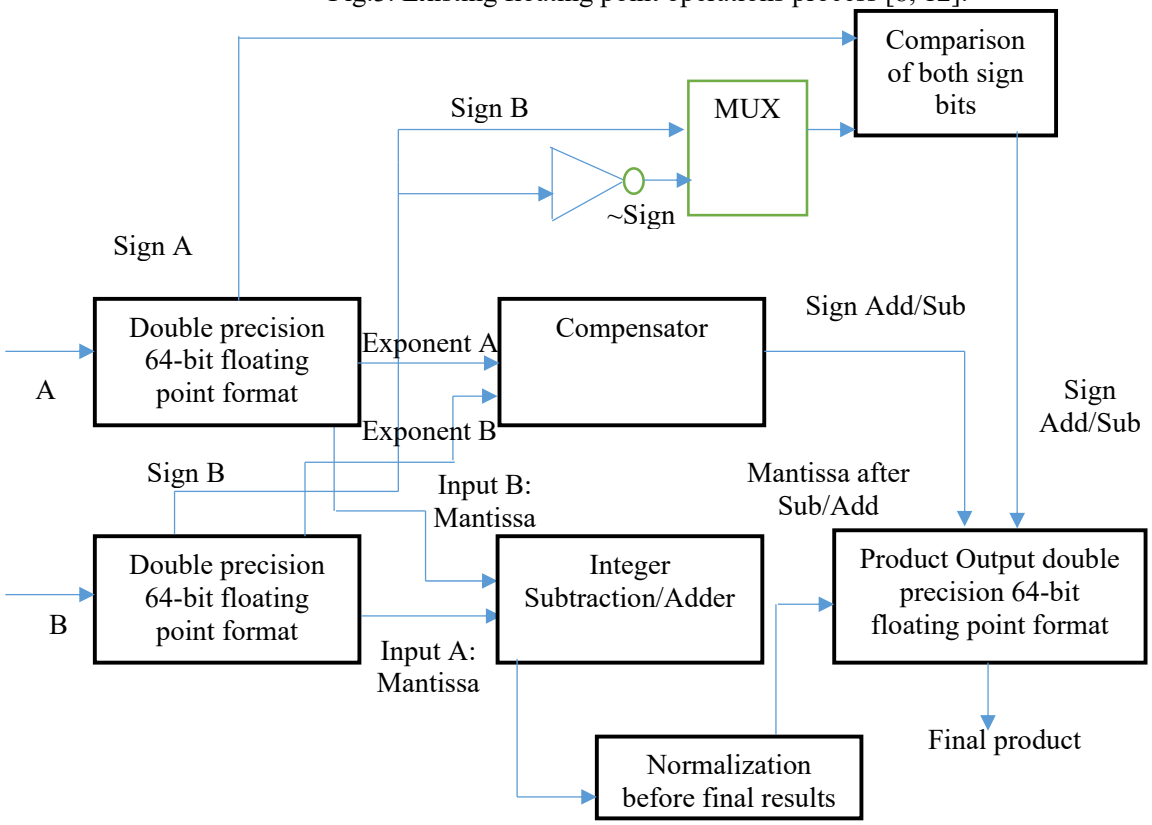

Fig.4. Proposed detailed double precision floating point operation of all arithmetic operations for Milli degree estimation.

4. Results and Discussion

The Xilinx Design Suite 14.7 ISE software tool's Design Compiler, which can synthesise Verilog HDL designs to digital circuits for SoC, is used to compare several existing formats and proposed double precision arithmetic operations. The floating-point format for each bit width is N (S, E, M), where N is the total number of bits, S is a sign bit, E is an exponent, and M is a Mantissa. The lower the space and energy used in floating-point multiplication, as shown in Table 1, the fewer bits used. The multiplier circuit's most remarkable feature is that, unlike the adder/subtractor, its energy consumption skyrockets. Finally, Table compares division operators in a variety of floating-point formats. Despite the fact that the operation delay time remains constant when compared to other operators, the smaller the number of bits employed for the floating-point division operation, the lower the area or energy consumption. Many test cases have been run to determine the appropriate arithmetic architecture without sacrificing millidegree estimation accuracyBased on testing, we fed our trained models 10K test images from the MNIST dataset to measure inference accuracy as shown Fig.. It can operate at frequencies 271 MHz and verified the throughput. It uses 5.44 times less energy and takes up five times less space than its predecessors. We tested our mixed-precision architecture on Virtex-7 handwritten dataset and were able to reach over 63 percent accuracy. Our accelerator is ideal for IoT applications due to its small size, power consumption, and accuracy.

.

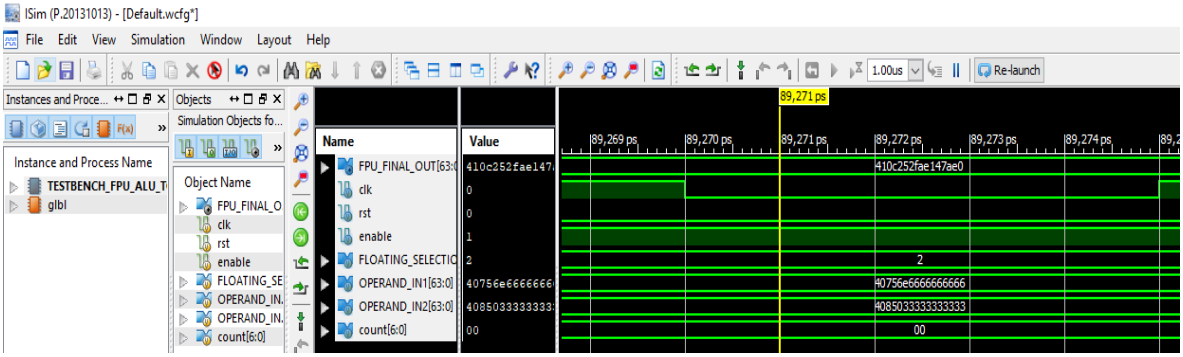

Fig. Simulated results of Double precision floating point for multiplication for two operand values such as A= 342.9 (in Hexadecimal= 40756e6666666666) and B= 672.4 (in Hexadecimal= 4085033333333333) and result (A*B) is 2,30,565.96 (in Hexadecimal= 410c252fae147ae0)

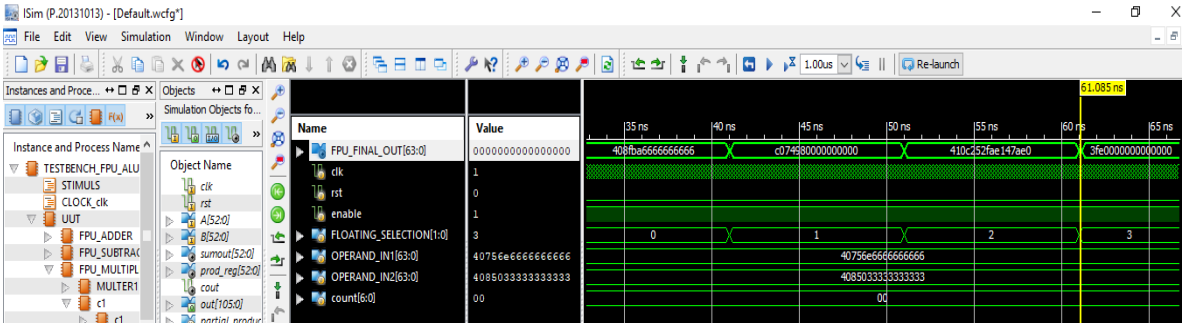

Fig. Simulated results of Double precision floating point for ALU for two operand values

Table.1. Traditional multiplier and proposed Multiplier comparisons

Multiplier	Slices (area)	LUT	Delay (ns)	Power (mW)
Narjes Hasanikhah [3]	140,818	230,751	7.86	5,264
Niu [4]	12710	---	4.69	263.5
Proposed ZP & FRBM	**2231**	**5000**	**4.11**	**88**

5. Conclusion

The impact of efficient FP ALU for high-speed FP formats on the implementation of FP arithmetic in FPGAs without native FP capability is investigated in this work. In comparison to the IEEE-754 standard, the proposed converters to/from any radix allow working at high speeds inside the FPGA and returning output numbers in IEEE-754 format if required for applications using data from an IEEE-754 FP source, all while increasing the number of significant bits used and dynamic range. The results show that the large area and latency benefits of utilising FPGAs to build high-speed FP adders may outweigh the limitations of using FPGAs to implement high-radix FP multipliers.

This work allows determining the optimal allowable radix of the FP arithmetic representation as a function of the ratio between the amount of additions and multiplications in the targeted algorithm for algorithms that comprise both addition and multiplications. Depending on the needs of the designer's unique application, this guidance allows the designer to pick between maximum speed, area, or a trade-off option. In several operations, such as dot product, matrix by matrix multiplications, convolutions, and others, the number of addition equals the number of multiplications. Many other operations, such as matrix and vector addition or average computation, rely on addition. Signal processing, machine learning, deep learning, computer graphics, wireless communications, and other disciplines all make extensive use of these techniques. Engineers can utilise this method to build more efficient structures in FPGA-based systems while preserving or even improving the precision of operations across the board.

References

[1]. B. Zhou, G. Wang, G. Jie, Q. Liu and Z. Wang, "A High-Speed Floating-Point Multiply-Accumulator Based on FPGAs," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 10, pp. 1782-1789, Oct. 2021, doi: 10.1109/TVLSI.2021.3105268.

[2]. W. Mao et al., "A Configurable Floating-Point Multiple-Precision Processing Element for HPC and AI Converged Computing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 2, pp. 213-226, Feb. 2022, doi: 10.1109/TVLSI.2021.3128435.

[3]. L. Gao et al., "DPF-ECC: A Framework for Efficient ECC With Double Precision Floating-Point Computing Power," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3988-4002, 2021, doi: 10.1109/TIFS.2021.3098987.

[4]. Z. Galias, "Periodic Orbits of the Logistic Map in Single and Double Precision Implementations," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 11, pp. 3471-3475, Nov. 2021, doi: 10.1109/TCSII.2021.3081604.

[5]. M. Fasi and M. Mikaitis, "Algorithms for Stochastically Rounded Elementary Arithmetic Operations in IEEE 754 Floating-Point Arithmetic," in IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 3, pp. 1451-1466, 1 July-Sept. 2021, doi: 10.1109/TETC.2021.3069165.

[6]. F. Zaruba, F. Schuiki, T. Hoefler and L. Benini, "Snitch: A Tiny Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads," in IEEE Transactions on Computers, vol. 70, no. 11, pp. 1845-1860, 1 Nov. 2021, doi: 10.1109/TC.2020.3027900

[7]. N. S. Mangalath, R. A. Priya and P. Malathi, "An efficient universal multi-mode floating point multiplier using Vedic mathematics," 2014 International Conference on Advances in Communication and Computing Technologies (ICACACT 2014), 2014, pp. 1-4, doi: 10.1109/EIC.2015.7230724.

[8]. M. K. Jaiswal and N. Chandrachoodan, "Efficient Implementation of IEEE Double Precision Floating-Point Multiplier on FPGA," 2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems, 2008, pp. 1-4, doi: 10.1109/ICIINFS.2008.4798393.

[9]. Shanmugapriyan S and Sivanandam K, "Area efficient run time reconfigurable architecture for double precision multiplier," 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), 2015, pp. 1-6, doi: 10.1109/ISCO.2015.7282355.

[10]. A. P. Ramesh, A. V. N. Tilak and A. M. Prasad, "An FPGA based high speed IEEE-754 double precision floating point multiplier using Verilog," 2013 International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT), 2013, pp. 1-5, doi: 10.1109/ICEVENT.2013.6496575.

[11]. S. Nair and T. S. B. Sudarshan, "An asynchronous double precision floating point multiplier," 2015 IEEE International Conference on Electrical, Computer and Communication

Technologies (ICECCT), 2015, pp. 1-5, doi: 10.1109/ICECCT.2015.7226153.
[12]. A. Jain, R. Jain and J. Jain, "Design of Reversible Single Precision and Double Precision Floating Point Multipliers," 2018 International Conference on Advanced Computation and Telecommunication (ICACAT), 2018, pp. 1-4, doi: 10.1109/ICACAT.2018.8933712.
[13]. Y. S. Rao, M. Kamaraju and D. V. S. Ramanjaneyulu, "An FPGA implementation of high speed and area efficient double-precision floating point multiplier using Urdhva Tiryagbhyam technique," 2015 Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG), 2015, pp. 271-276, doi: 10.1109/PCCCTSG.2015.7503923.
[14]. F. Mhaboobkhan, K. Kokila, R. Jothikha and K. L. Preethikha, "Design of Pipelined Parity Preserving Double Precision Reversible Floating Point Multiplier Using 90 nm Technology," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 739-744, doi: 10.1109/ICACCS48705.2020.9074209.
[15]. P. Anuhya and R. Dhanabal, "Asic Implementation of Efficient Floating Point Multiplier," 2018 4th International Conference on Electrical Energy Systems (ICEES), 2018, pp. 138-141, doi: 10.1109/ICEES.2018.8443236.
[16]. K. Manolopoulos, D. Reisis and V. A. Chouliaras, "An efficient multiple precision floating-point multiplier," 2011 18th IEEE International Conference on Electronics, Circuits, and Systems, 2011, pp. 153-156, doi: 10.1109/ICECS.2011.6122237
[17]. M. K. Jaiswal and H. K. -. So, "DSP48E efficient floating point multiplier architectures on FPGA," 2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID), 2017, pp. 1-6, doi: 10.1109/ICVD.2017.7913322.