FPGA Implementation of Antilogarithmic Computation using Fixed Point Architecture at SoC Integration

DR Prashanth Bachanna, Assistant Professor
Electronics and Communication Engineering
Institute of Aeronautical Engineering
Hyderabad, Telangana, India
b.prashanth@iare.ac.in

Sunkaraboina Aravind
Electronics and Communication Engineering
Institute of Aeronautical Engineering
Hyderabad, Telangana, India
sunkaraboinaaravind123@gmail.com

Ramavath Bhanu Prasad
Electronics and Communication Engineering
Institute of Aeronautical Engineering
Hyderabad, Telangana, India
bhanuprasadramavath56@gmail.com

Praisy Moses
Electronics and Communication Engineering
Institute of Aeronautical Engineering
Hyderabad, Telangana, India
mosespraisy0001@gmail.com

Abstract*—In recent studies works in device system-on-chip (SoC) and Embedded structures applications need optimization of electricity, latency, area, and improvement of throughput. all these packages are complex in operations and to validate FPGA is a high-quality appropriate device with minimal time. maximum of operations in DSP packages uses constant factor and records route are broadly used and attention of complicated arithmetic operations using logarithmic variety systems. so one can improve throughput and optimize electricity, region, and postpone, a novel antilogarithmic architecture has been found on FPGA. The proposed antilogarithmic function uses piecewise linear approximation (PLA), main one detector (LoD), barrel shifter (BS) and Reconfigurable residue variety system (RRNS), Parallel Prefix Adder (PPA) is included in the design. The SoC stage layout has been synthesized in Vivado Design Suite 2018.1 and tested on Artix-7 FPGA. The device utilization demonstrates how little FPGA aid the architecture uses. moreover, we have tested the approximation result through mistake analysis. in line with the mistake looked at, there may be a 2.4 percent. mistake for negative numbers and a zero.24 percentage errors for tremendous values. by the usage of greater bits for the fractional bit illustration, the inaccuracy may be reduced even greater. tool. The tool utilization demonstrates the greatest latency and minimum good judgment sources, and the same design is tested at the FPGA. additionally, we've got examined the approximation result via mistake, analysis. The LOD designs and an approximation adder for summing logarithms can be applied to beautify the Mitchell logarithmic (ML) multiplier's hardware performance. In contrast to the authentic Mitchell multiplier, this layout lowers hardware charges by way of 21 percent.eight, while in contrast to the alternative present-day generation, it lowers fees using 17.5 percentage.

***Index Terms*—Antilogarithm; fixed-point architecture, leading one detector, approximate adder, Mitchell logarithmic multiplier and FPGA.**

I. INTRODUCTION

It is known from the literature that binary arithmetic is useful and accurate in the material [1], [2]. Digital designers can now use logarithms instead of arithmetic for binary operations. Compared to teaching arithmetic, it works equally well with addition and saves a lot of mathematics [3], [4] [5]. In modern times, logarithmic operations are popular because they are better equipped than binary arithmetic, but they still lack the precision of binary arithmetic [6]. The following main steps are involved in the calculation in logarithmic systems (LNS): log transformation, computation phase and iteration [7, 8]. The real-time DSP implementation is the most important factor affecting the hardware and performance of the LNS- based arithmetic operation super-set. Common in arithmetic are logarithm and anti-log transformations [9, 10]. Since the anti-log transformation is the determining factor, converters are necessary to speed up the arithmetic operation [11, 12]. In many computer systems, partitioning is an active operation that uses a lot of resources and energy. Approximate arithmetic can reduce hardware costs and competition when used in ap-plications where the probability is not accepted. Approximate circuits are used in forensics, such as machine learning, digital programming (DSP), and many other computer problems [1]. In cases where speed is more important than accuracy, logarithmic numbers can be a better choice than decimal numbers because they do not produce partial multiplications. The logarithmic technique has proven useful in many DSP applications, but it incurs a slight loss of accuracy in arithmetic operations [2]. Delays play an important role in image and signal processing applications, especially in hardware devices

such as FPGAs. The text [14] describes the delay using the approximation method, which uses the encoding algorithm to generate a logarithmic inverse algorithm; this can reduce the delay and error due to the coefficients by up to 20. Multipliers consume more slowly while summing a part of the product. To avoid the use of multipliers, an additional method is added to the design to optimize the area and power consumption. It is already a project in [15], which generates data for DSP applications. A new algorithm is developed and implemented using hardware-free and low-power correction circuitry for the correction of anti-log. These VLSI solutions are smaller compared to other hardware solutions used in the literature. The converter is designed and manufactured using 0.6/spl mu/m CMOS process technology and the connection logic architecture requires 1500/spl lambda//spl times/2800/spl lambda/ dead space. The 32-bit anti-log converter consumes 81 mW, operates at 100 MHz, and determines the anti-log in just one hour [15].

For real-time applications in today's systems, especially for multimedia, video/signature, music, DSP, and graphics, it is difficult to adapt to space-saving codes [1-4]. Now, as the popularity of the required number of applications increases, hair roots, advanced filters, power of quadratic operations, inversions, and exponential operations are needed [4]. VLSI implementations of these complex arithmetic systems use floating point cell (FP) bits, are complex, slow, power consuming, and require extensive semiconductor technology [3-5]. In embedded systems where variable numbers need to be represented by a suitable decimal number, some form of measurement is important. Fixed arrays are suitable for the construction of widely used arrays due to their simple data and robust arithmetic. In addition, the information method of security-based circuits is fast, space-saving, and energy-saving [4-5]. FPGAs are widely used as solutions for all kinds of complex applications with minimum design time. Advanced FPGAs have many common devices including adders, multipliers, embedded memory, and embedded processors on a single FPGA chip in addition to rich logic devices [5-8]. Fixed-point arithmetic provides a numerical data method that can be used with FPGA macro elements. The Logarithmic Number System (LNS) can be used with the FP Number System to help construct complex arithmetic operations.

II. PROPOSED BINARY ANTI-LOGARITHMIC FUNCTION FOR DSP APPLICATIONS

The design concept of anti-logarithmicThe arithmetic is simpler than point arithmetic because it only requires percentage data. As a result, fixed-point units use less power and occupy less space. Also, small FPGA structures can be easily implemented using fixed hardware. We can also take advantage of FPGA macro elements that are readily available and can be modified to meet the needs of arithmetic operations at higher clock speeds. The data path of the proposed architecture is used by the 2.6.12 fixed-point format, as shown in Figure 1. 2. The piecewise linear approximation method is a space-saving method that uses binary anti-log units. Let X [20] 1 If X =

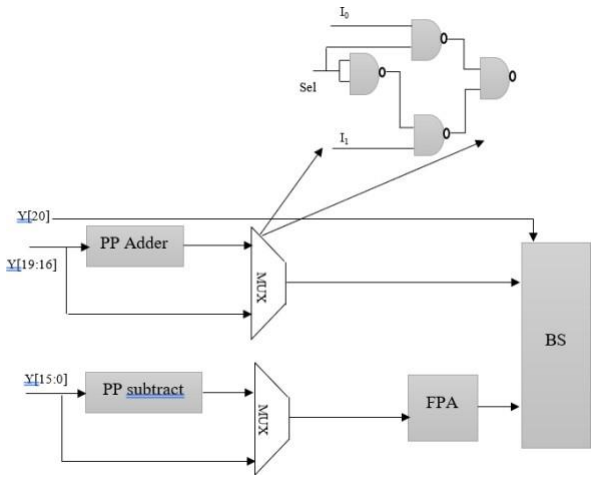

Fig. 1. functional diagram of optimal latency MUX-based antilog computation unit

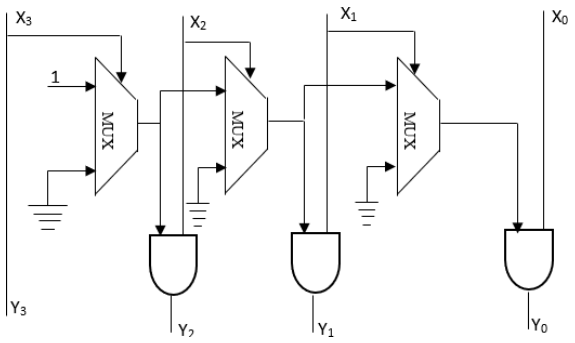

Fig. 2. Proposed LoD for 4-bit for sign detection in antilog function

occurs, the input is negative; otherwise, X[20] 0 = means that the input number is positive. The anti-numeric value can be calculated using the PA number as shown in (1)

Here the value of f/k varies according to the sign bit. Here in PLA, the maximum information (f) is approximately 0 → 1 If the data is negative and outside the predefined range, we reduce the fractional number by one and subtract the fractional number by one. In this way, negative binary integers can be approximated in the same way. Equation (1) is an anti-log function and its binary code is Y. Anti-log implementation is provided. Here, the desired value 2f in (1) is found using fractional partial approximation (FPA) design cells. After determining the part, use the barrel shifter (BS) to change the calculated FPA output value to the left or right by the value of the characteristic part according to (1). 1. Working diagram of anti-log counting unit based on optimum delay MUX LoD is the main subsystem module in this application and optimizes the delay, area, and power consumption initially. LoD is designed for 4-bit as shown in Figure 2 and then continues with 16. .-bit and 32-bit. For 32-bit LoD, 8x LoD is initialized and integrated. 4-bit LoD gives the result of 4- bit binary output by setting the bits to the expected "1" and

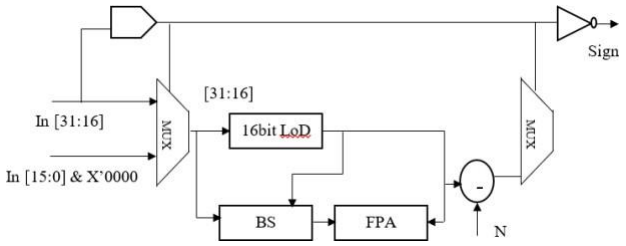

Fig. 3. Block diagram of the proposed anti-logarithmic circuit with Optimal LoD

the other bits to ”0”. There are two main modules in the approximate logarithmic function of the multiplier: LoD and adder; the latter is the logarithm of the input to the adder. We plan to use RRNS and PPA algorithms to process these models to improve the area, power consumption, and delay. 32-bit LoD is generated and implemented using 4-bit LoD, and with the help of eight 4-bit LoD modules measured in parallel, 32 bits are divided into 2 parts of 4 bits each. During the second step, 4-bit LoD counts the first bit in two 16-bit fields, bits 31 to 16 and bits 15 to 0. In the third step, 2-bit LoD (e.g. ”000”, ”01”, ”10” connection) enables the selection of two levels of 4-bit MUX. 4-bit LoD of the second level If the result is logical -1, it will be displayed in the selected row, otherwise, it will be -0000 and the same result will be generated by the multiplexer. The 8-bit output of MUX forms the input selection for eight sets of 4-bit MUX in the final processing stage. One of the 8 bits used to select the line will be a logical ”1” and MUS- will send 4 bits from the first level 4-bit LoD for the decision. Finally, through simulation, Mitchell determined the maximum number of objects that can be estimated in LOD without compromising the accuracy of the final object. From the actual analysis given in the next section, we see that the four key 4-bit LOD in Figure 3 will always be approximated by the constant hexadecimal variable x”0400” as shown in Figure 4. Since we previously estimated four minimum 4-bit LODs in the first stage, now we can estimate the minimum 4-bit LOD in the second stage, as shown in Figure 2 ”LODA A Initial, 16-bit LSB: false and LoD input greater than 216 means there is no error in the estimated product, otherwise less than 216 will cause sign error. Synthesis and simulations have been performed to estimate the maximum number of objects in LoD without affecting the accuracy of the end of the RRNS multiplier, 4 LSB

A. PROPOSED FPA FOR ANTILOG FUNCTION

The fraction of antilog 2 is derived with the help of a piecewise approach, where for every 19 bits 8 predefined values are stored in ROM and defined as 2 Here I am different from 0 and 7. The design is being worked on. In ROM, the first 8 bits of the values stored in MSB represents and the remaining 11 bits represent 4. In 2 The first 3 bits are used for ROM address and are added to the time of. Use

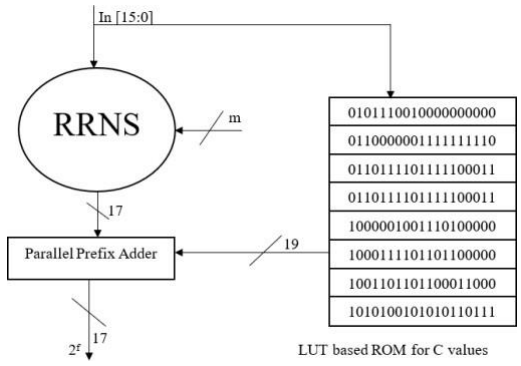

Fig. 4. Proposed RRNS and PPA for FPA along with LUT-based ROM

Parallel Prefix Appenders (PPAs). PPA also uses the sum of all partial components of the equation. Prime integers serve as the moduli in the predefined moduli set used by the RNS to carry out arithmetic calculations. Using moduli set values and total elements used as moduli, the range of input operands that the RNS can withstand devoid of truncating the results is statistically generated. Additionally, each module and related computations are performed as independent networks in L equivalent routes during RNS computation, significantly reducing path propagation latency. Furthermore, the RNS system employs modified PPA topology-based accumulation to reduce the path latency.

III. SOFTWARE REQUIREMENTS

1. The software environment must: FPGA development tools: Xilinx Vivado, Intel Quartus Prime, or other FPGA synthesis tools for coding, simulation, and hardware imple- mentation. for verifying the behavior of VHDL code before synthesis. Manage tools to manage policies and changes. Fixed-point architecture considerations: Fixed-point represen- tation: Define a bit width between the input and output code as required. ¿ Range and Precision: Ensure that fixed points provide sufficient range and precision for anti-log operation. and underflow handling: Overflow and underflow handling logic in fixed-point arithmetic. Functional design: Mathematical functions: The main function is to calculate the anti-log (inverse of the logarithm), for x= e y (natural logarithm) orx=10 y (logarithm to base 10) To use poly- nomial approximations, the approximation method can be used (Taylor series, etc.), the function of the table to find the exponential value, or the CORDIC (Coordinate Rota- tion Digital Computer) algorithm. Output: Fixed-point anti-numerical value. Data Path Design: Piping and Parallelism: If performance is important, use pipelines and parallelism to optimize logarithmic returns to increase and decrease latency. The machine or controller needs to manage the computation phase, especially when dealing with back-end algorithms or lookup tables. LUS) index. SoC Integration: Interfacing with

SoC: Enable AXI (Advanced Extensible Interface) protocol or other SoC transport to communicate with ARM core or other components. Interrupt management between FPGA and SoC for efficient data transfer: Create an interrupt line to signal the SoC processor when processing is completed. FPGA and SoC subsystems are required. Experimental and practical: Simulation: According to the design of the FPGA board. Optimization: Resource Usage: Reduce the usage of logic devices (LUTs, flip-flops) by optimizing algorithms and data paths. (in LUT).

IV. DESIGN AND IMPLEMENTATION

1. Understanding fixed-point notation Fixed-point format: In fixed-point notation, numbers are stored as a fixed number divided into numerators and fractions. For example, a 16-bit number can contain 8 bits for digits and 8 bits for decimals (type Q8.8). Select the appropriate fixed-point format for input and output. Anti-log Function Anti-log is the inverse of the logarithm. For base-10 logarithms, the anti-log is given by: ¡ In hardware, computing the anti-log can be done with a lookup table (LUT) or expansion (such as the Taylor series or CORDIC algorithm). 3. FPGA Algorithm Design There are several ways to implement the anti-log function:

a. Lookup Table (LUT) Precomputed Values: A simple approach is to precompute the value of the anti-log function and store it in a lookup table. For example, if the input is a fixed-point number in the range [0, 1], you can compute the inverse value in discrete steps (e.g., every 0.01) and store it in memory. using linear interpolation of the LUT values. CORDIC algorithm Iterative algorithm: The CORDIC (Co-ordinate Rotation Digital Computer) algorithm can compute trigonometric functions, hyperbolic functions, and exponential functions. It is particularly suitable for FPGAs because it uses only switches, adders, and LUTs. It moves and adds values through 2 x iterations. Taylor Series Expansion Another method is to use the Taylor series approximation for anti- log functions. For example, the series can be converted to constant integers, but this method can be more computationally intensive than LUT-based methods. 4. Fixed-Point Arithmetic Fixed-Point Arithmetic Addition/subtraction: Easy because it uses the same bit width. bit width (twice the operand width), so appropriate scaling and truncation are required. FPGA Using HDL Design: Use Verilog or VHDL to define the behavior of anti-log functions. Create memory or allocate RAM on the FPGA if you are using a LUT-based design. For the CORDIC system, recalculations are implemented using shift-add logic. Design accuracy and performance. SoC Integration Processor Integration: If the FPGA is part of the SoC (such as Xilinx or Intel SoC FPGA), the code protection device can be integrated as an accelerator device connected to the processor operation (such as ARM core). Interface: Use AXI (Advanced Extensible Interface) to connect the FPGA fabric to the processor. The processor can send the input value to the FPGA via the AXI bus, and the FPGA returns the anti- log value. You may need to perform tasks such as sending data, performing calculations, and reading back the results. ¿

7. Optimization Latency and throughput: optimize the design for specific applications. For urgent applications, ensure that the delay numbers meet the time requirements of the system. The balance of resources depends on the complexity of the FPGA device and the back-end processing. Verification and Validation Design by Analogy: Use HDL simulation tools to verify that an inverse function behaves correctly when given various inputs. Check the embedded hardware. If necessary, adjust the correctness to ensure the result is acceptable. Final integration and implementation After analyzing the counter-measures together with other components in the SoC, such as memory, I/O interfaces, and other computations. Platform: Use Xilinx (Vivado, Zynq SoC) or Intel (Quartus, Intel SoC) platform for implementation. Direct the library to work.

V. RESULT

1. Performance measurement Speed: FPGA implementations can be very similar, allowing for faster development compared to software solutions running on traditional CPUs. Especially when using pipelines or components, you should see an improvement in the speed of the anti-log operation.) or efficient use of resources. Point-to-point representation of precision Precision: Point-to-point computation is often chosen for FPGAs due to its lower resources compared to floating point. However, this comes at the expense of accuracy. Careful scaling and selection of the number of components in integer and fractional values is important to reduce quantization error. The results should include an analysis of how well the FPGA computes the true variable count, possibly measured as mean squared error (MSE) or maximum error (MAE). Resource Usage Logic elements and scratchpads: FPGA resources such as LUTs (lookup tables), FFs (flip-flops), and DSP slices (for arithmetic operations) will be heavily utilized, and the results should include information on how the FPGA resources were utilized. The power efficiency of FPGAs compared to other platforms such as CPUs is particularly important in the SoC environment, where power consumption is often limited. Comparison with other architectures Fixed-point vs. floating-point: The results should compare the resource utilization, speed, and accuracy of fixed-point and floating-point. Floating-point implementation of FPGAs. FPGA results are compared to the equivalent implementation on a CPU or GPU, particularly in terms of speed, processing power, and accuracy. Integration and Integration in SoC Interfaces: Since the anti-log function is part of the SoC, the results should indicate the degree of integration of the FPGA module with other objects (e.g., memory interfaces, communication buses). Delays or conflicts due to communication overhead or interference should be minimized. Discussion: In a SoC, multiple IP cores may compete for resources (such as memory bandwidth). It is useful to examine how the anti-log computation unit fits into the SoC environment and how it affects the overall performance of the system. Scalability and Flexibility Design scalability: The results should show how the design scales with large input or anti-log operations. The implementation can be easily modified or extended to compute other mathematical functions (e.g.

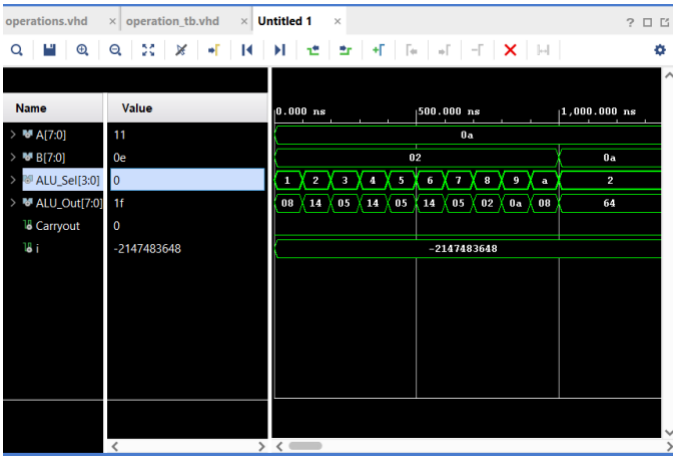

Fig. 5. simulation

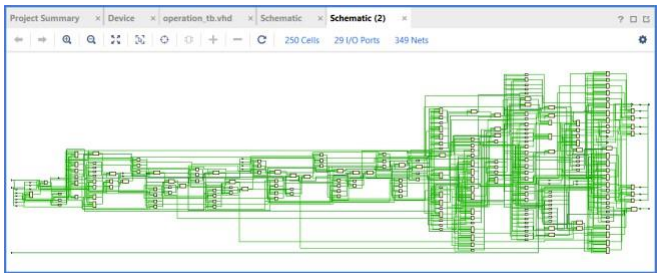

Fig. 6. schematic Diagram

logarithms, exponential functions) or to accommodate different levels of precision in the fixed representation. Using the Optimization Process Pipelines: Designers can use pipelines to increase the depth of the pipeline, and the impact on performance will be significant. degree and its impact on utilization and employment. Synthesis and Implementation Results Synthesis Reports: These typically include time spent, details of resource usage, and downtime. These results will show that the design meets performance goals. Case Study or Practical Application Case Study: If FPGA anti-log computing is used for a specific application (such as configuration, audio processing, or financial computing), the results should show the improvements and advantages of this FPGA solution compared to software. .use Make a comparison.

VI. CONCLUSION

Using logarithmic, complex arithmetic can be handled more easily. This research proposed a new model for FPGA to implement anti-log evaluation method . The proposed antilog circuit uses the PLA technique. The same generator can process both positive and negative binary numbers. Create a special BS that moves the input data to the left or right by the appropriate amount. Xilinx Artix-7 xc7ALX100t device im- plementation Proposed architecture. Device utilization shows how much of the FPGA resources are used by the device. We also analyzed the estimated values through error analysis. The inaccuracy can be further reduced by using more objects to represent the decimal number. The application of the higher FPA resolution to RNS units is the subject of this paper. The results of the integrated equipment presented in this work show that the hardware design speed, durability and performance are directly affected by all the processing levels that need to be improved during the computation of the return logarithm. Both RAM-based FPA functions are used to reduce DSP implementation, which can reduce the performance penalty in the gap in the FPA design. Complex arithmetic can be handled more easily by using logarithmic. This study suggested a novel model for the FPGA that uses the anti-log evaluation technique. The PLA approach is employed in the suggested antilog circuit. Binary numbers that are positive or negative can be processed by the same generator. Make a unique BS that shifts the input data by the necessary amount to the left or right. Proposed architecture for the Xilinx Artix-7 xc7ALX100t device implementation. The device's utilization indicates the percentage of FPGA resources that it is using. We also used error analysis to examine the estimated values. By representing the decimal number with more items, the error can be further decreased. This research focuses on applying the greater FPA resolution to RNS units. The outcomes of the

VII. REFERENCES

[1]. Nandan, D. (2020). An efficient antilogarithmic con- verter by means of the use of correction scheme for DSP processor. Traitement du sign, 37(1): seventy seven-eighty three. https://doi.org/10.18280/ts.370110

[2]. B. Xiong, Y. Li, S. Li, S. Fan and Y. Chang, "half-Precision Logarithmic mathematics Unit primarily based on the Fused Logarithmic and Antilogarithmic Converter," in IEEE Transactions on Very huge Scale Integration (VLSI) systems, vol. 30, no. 2, pp. 243-247, Feb. 2022, doi: 10.1109/TVLSI.2021.3136229.

[3]. C. -T. Kuo and T. -B. Juang, "A decrease error antilogarithmic converter using novel four-place piecewise-linear approximation," 2012 IEEE Asia Pacific convention on Circuits and systems, Kaohsiung, Taiwan, 2012, pp. 507-510, doi: 10.1109/APCCAS.2012.6419083.

[4]. J. Lee, J. Lee, D. Han, J. Lee, G. Park and H.-J. Yoo, "7.7 LNPU: A 25.3TFLOPS/W sparse deep-neural-community getting to know processor with fine-grained mixed precision of FP8-FP16", IEEE Int. strong-state Circuits Conf. (ISSCC) Dig. Tech. Papers, pp. 142-144, Feb. 2019.

[5]. A. Haidar, S. Tomov, J. Dongarra and N. J. Higham, "Harnessing GPU tensor cores for instant FP16 mathematics to hurry up combined-precision iterative refinement solvers", Proc. SC Int. Conf. high carry out. Comput. Netw. storage Anal., pp. 603-613, Nov. 2018.

[6]. J. Wei, A. Kuwana, H. Kobayashi, ok. Kubo and Y. Tanaka, ”Floating-factor inverse square root algorithm based on Taylor-series enlargement”, IEEE Trans. Circuits Syst. II Exp. Briefs, vol. 68, no. 7, pp. 2640-2644, Jul. 2021.

[7]. ok. Dan, ”Evolution of conventional antilogarithmic method and implementation in FPGA via VHDL,” 2014 Worldwide Conference on Advances in Computing, Communications and Informatics (ICACCI), Delhi, India, 2014, pp. 1839-1844, doi: 10.1109/ICACCI.2014.6968267.

[8]. S. Paul, N. Jayakumar and S. P. Khatri, ”a fast hardware approach for approximate green logarithm and antilogarithm computations”, IEEE Transactions on Very Huge Scale Integration (VLSI) systems, vol. 17, no. 2, pp. 269-277, Feb. 2009.

[9]. S. Paul, N. Jayakumar, and S. P. Khatri, ” A fast hardware approach for Approximate, green Logarithm and Antilogarithm Computations,” in IEEE Transactions on Very Big Scale Integration (VLSI) structures, vol. 17, no. 2, pp. 269-277, Feb. 2009, doi: 10.1109/TVLSI.2008.2003481.

[10]. Al-Tamimi, okay.; Thiyagarajan, P.; El-Sankary, okay.: continuous-time four-quadrant modulator with inherent PVT cancellation. Electron. Lett. 52(10), 807–808 (2016)

[11]. Maryan, M.M.; Azhari, S.J.: A MOS translinear cellular-based configurable block for modern-day-mode analog sign processing. Analog Integr. Circuits signal process. ninety-two (1), 1–13 (2017)

[12]. Maryan, M.M., Ghanaatian, A., Azhari, S.J. et al. Low-electricity high-speed Analog Multiplier/Divider based on a brand new current Squarer Circuit. Arab J Sci Eng forty three, 2909–2918 (2018). https://doi.org/10.1007/s13369-017-2968-2.

[13]. R. R. Selina, ”VLSI implementation of Piecewise Approximated antilogarithmic converter,” 2013 worldwide convention on communique and signal Processing, Melmaruvathur, India, 2013, pp. 763-766, doi: 10.1109/iccsp.2013.6577159.

[14]. k. H. Abed and R. E. Siferd, ”VLSI implementation of a low-power antilogarithmic converter,” in IEEE Transactions on computer systems, vol. 52, no. 9, pp. 1221-1228, Sept. 2003, doi: 10.1109/TC.2003.1228517.