Vol. 06, No. 3 (2024) 903-914, doi: 10.24874/PES06.03.002

Proceedings on Engineering Sciences

www.pesjournal.net

DESIGN AND PERFORMANCE ANALYSIS OF TERNARY LOGIC BASED ALU USING DOUBLE PRECISION FLOATING POINT

Nagarathna R[1]
A R Aswatha

Received 25.10.2023.
Received in revised form 26.11.2023.
Accepted 07.01.2024.
UDC – 004.62

Keywords:

Digital Circuits, Ternary, Floating point, ALU

A B S T R A C T

In digital circuits, particularly space signal applications, the detection/estimation of phase (angle) like milli degree is challenging and involves many complex operations. To estimate milli degree (10-3) or more, floating point operations (like double precision adders, subtractions, multiplications, and divisions) are the significant components and power and area consumption; delays are more in existing works. Existing works mainly concentrate on clock-based synchronous operations, which require more hold, and clock distribution is another major problem due to extra circuitry requirements. The proposed pipelined and clock-efficient distribution system (CEDS) is incorporated in the Floating-Point Unit (FPU) design to address these issues. It includes a double-precision adder, multiplication, division, and subtraction. These FPU and CEDS can be used to detect Milli Degree for GHz frequency in Space applications more accurately. The floating points are essential in phase shift operation used in the space-borne system for effective correction of error checking rather than execution of error analysis in the real-time scenario. Modern digital circuits can hardly improve due to limitations of physical area consumption. The substitution for these limitations, the new digital ternary logic (0, 1, 2), is the solution because of its higher number of digital circuits involved in floating points. The ternary logic is the best solution for the minimization of number bits to optimize memory utilization, and ternary logic has Negative Ternary Inverter (NTI), Ternary Decrement Cycling Inverter (DCI), Standard Ternary Inverter (STI,) and Positive Ternary Inverter (PTI). The proposed design is successfully designed and validated on Zybo Z7-10 (XC7Z010-1CLG400C) FPGA development board. The results show a 45% reduction in power consumption, delay, and area utilization.

1. INTRODUCTION

The implementation of asynchronous floating-point adders has yet to be studied in this work. Most studies mainly concentrate on the performance of asynchronous floating-point operations such as multiplication and division (Jaiswal, 2014). This is because of the difficulties involved in implementing an Floating Point Adder (FPA). Mantissa shifting for matching the exponent, aligned mantissa addition, rounding, and normalization of the output calculated with different latencies are some of the operations involved during floating-point accumulation. Since the synchronous FPA estimates the frequency of the clock depending on

[1] Corresponding author: Nagarathna R
Email: nagarathnatce@dayanandasagar.edu

the worst-case delay, there is no necessity to worry about the process completion because any process can terminate before or within the worst-case delay.

On the other hand, completing the process early can improve the addition function from 60% to 90% (Jaiswal, 2015). By substituting the clock pulse with REQ as well as ACK signals, an asynchronous floating-point adder takes the benefit of early completion detection. It thereby minimizes the time required for processing from worst-case to average-case delay. The contemporary Asynchronous Floating-Point Adder (AFPA) design presented by Noche as well as Jose (Jaiswal, 2016), Sheikh as well as Manohar (Savas, 2017), Jun, as well as Wang (Sangeetha, 2018 and Hiratkar 2016), is addressed in the following section; these are the only architectures presented in the research work that explains the implementation of AFPA design. The MTNCL technique presented in (Havaldar, 2016) provides no information regarding the AFPA implementation. This technique is included in the work because it compares the efficiency of floating-point operations such as addition and subtraction for asynchronous and synchronous floating-point processors.

Single-Precision AFPA: Noche and Jose (Spoorthi, 2020) presented a variable latency approach for implementing a single-precision Asynchronous Floating-Point Unit (AFPU) which is discussed in this section. All existing floating-point units' implementations are primarily concentrated on multiplication or division operations. Initially, this design presents the asynchronous implementation of a single-precision floating-point adder (Spoorthi, 2020), including additional arithmetic operations. The data path, as well as the control path in this design, makes use of dual-rail differential cascode voltage switch (DCVS) logic as well as complementary metal-oxide-semiconductor (CMOS) logic, respectively. To build as well as to test the arithmetic unit, Cadence software is utilized. The AFPU is constructed at the transistor level using a 3.3 V supply voltage with a 0.35 um process. The AFPU performance for addition operations is discussed in this research work. The primary elements of the data path considered for addition operations are registers and adders. Bidirectional shift registers, and a rounding bit, are often employed to design the shifter circuit and are essential for normalization and matching the exponent (Malkapur, 2020).

9-bit Carry Lookahead Adders (CLA) is used to develop an adder necessary for determining the exponent difference, whereas 25-bit is used to create an adder necessary for deciding mantissa addition (Ushasree, 2013). To choose inputs for registers and adders, the DCVS multiplexers are often utilized. However, if the dual-rail inputs are not active, then OR gates can be used to optimize the design. Logic gates, SR latches, and C-elements are present in the AFPU's control circuitry (Wang 2019).

1.1 Operand-Optimized Double-Precision AFPA

Even though the time required for processing excludes the time needed for computing the rounding logic, Noche, as well as Joes, claim to minimize the time needed for completing the process of single-precision AFPA (Malkapur, 2020). Furthermore, the architecture does not employ any other energy-saving strategies because it is entirely non-pipelined. Many asynchronous pipelining approaches optimize performance (Wang 2019 and Wang 2019). Pipelining is a strategy in which several operations are run concurrently for distinct data values to maximize the output. As stated in this section, Sheikh and Manohar (Zeng, 2015) developed an operand-optimized Double-Precision AFPA (DPAFPA) with all four-rounding logic.

The DPAFPA's performance was compared to that of a high-performance baseline AFPA, with the following operating parameters: In a 65 nm bulk CMOS process at the typical- typical (TT) corner, the temperature was found to be 25 ^{o}C with a supply voltage of 1 V. The standard AFPA employs a 56-bit Hybrid Kogge Stone Carry Select Adder (HKSCSA) to add mantissa to add mantissa. The adder produces two tentative sum outputs for two distinct carry-in values and based on the actual carry-in value, the final production is chosen at the last stage. The dual-rail protocol is utilized with 1-of-4 encoding and radix-4 arithmetic to maximize the energy and speed constraints. The Leading One Predictor (LOP) methodology is used for normalization and addition operation. The LOP estimates the shift amount, and the end outcome must be slightly shifted if the predicted shift amount is incorrect. To normalize the summation result, the data path is categorized into two pipelines, i.e., Left and Right. Whenever a significant left shift is expected due to a subtraction operation, the left pipeline is utilized, and the proper channel is used in all other situations. Thirty pipeline phases are employed in data paths with a minimum latency increase. In the case of all data computations, the pre-charge enable half-buffer (PCEHB) pipeline is used, which is quicker as well as more energy-efficient than the actual pre-charge half-buffer (PCHB) pipeline. Additionally, it employs the weak condition half-buffer (WCHB) for simple buffers and tokens, even though PCEHB is more energy efficient.

It also demonstrates that addition consumes the most power after the proper shift operation. When compared to baseline AFPA, the DPAFPA design presented by Sheikh showed enhanced power savings by incorporating the following modifications:

1. An interleaved asynchronous adder is substituted for HKSCSA, which employs two radix-4 ripple-carry adders. Simultaneously both the ripple-carry adders work for various input operands; even odd

operand pairs are added by one adder, whereas odd and another adder adds even operand pairs case of radix-4 arithmetic, the maximum carry-chain length for around 90% of scenarios is 7. At the same time, an interleaved adder's energy or operation requirement is 2.9 pJ/op for carrying distances below 15, and the bandwidth required is 2.2 GHz. On the other hand, the 56-bit adder (HKSCSA) employed by a standard FPA requires 13.6 pJ/op and seems to have a throughput of 2.17 GHz. As a result, in contrast to HKSCSA, an interleaved adder minimizes power usage by more than four times. Interleaved adders also lower the number of transistors utilized for a 56-bit adder by 35%.

2. The right shifter is made up of three pipeline stages: Stage 1, stage 2, as well as stage 3, shifts the mantissa from 0 to 3 bits, 0, 4, 8, or 12 bits, as well as by 0, 16, 32, or 48 bits respectively. In standard AFPA, the calculation time of the shifter to shift the mantissa from 0 - 55 bits are constant. The long path and short path are the two different paths of the shifter in the architecture of AFPA presented by Sheikh. This enables the shifter to choose one way based on the number of shifts and skip the other. The shifter's architecture is based on data and can reduce power consumption.
3. One pipeline, either left or right, is utilized for normalization in the modified technique of LOP. Before setting the LOP phase, one must choose which channel to use, i.e., left or right. Compared to standard AFPA, the left channel saves up to 13% power, and the proper channel can save up to 18%.
4. The rounding operation, 53-bit mantissa incrementor, left/right 1-bit shifter, as well as computation of the final value of the exponent, are all managed by the post-add proper pipeline. In contrast to the carry-select incrementor used by the standard AFPA, the DPAFPA architecture incorporates an interleaved incrementor analogous to an interleaved adder. This improves the energy efficiency of DPAFPA.
5. The zero input operands are detected by the architecture. When one or both operands are zero, the final result can be computed rather than utilizing the AFPA's power-consuming chunks.

When contrasted to the standard AFPA that requires 69.3 pJ/op, the suggested DPAFPA design consumes 30.2 pJ/op, resulting in a 56.7% drop in energy consumption. DPFPA's efficiency is also contrasted with Quinnell's synchronous FPA, which is regarded as one of the rarely designed fully implemented FPA that provides a reasonable standard for examining DPFPA's performance. A standard-cell library and a 65 nm SOI (Silicon-On-Insulator) technology were used to design the synchronous FPA. Table 1 compares the performance of DPAFPA, standard FPA, and the synchronous design of FPA proposed by Sheikh and Manohar. The efficiency of a floating-point unit is measured in GFLOPS (gigaflops) (FLOPS— floating-point operations per second). The suggested DPFPA seems to have a high GFLOPS/W, enabling the asynchronous design strategy appropriate for improving the circuit's performance. Only non-zero operands are evaluated in the input set for standard AFPA and DPFPA for correct shift values varying from 0 to 3.

The circuit's power consumption can be minimized by the application of interleaved adders as well as shifters. The shifter design is divided and deployed using pipelines, reducing the circuit's processing time and power consumption. AFPA is designed and tested using PRISM, which is regarded as a gate-level simulation tool that employs 10 billion random input operands, including one billion archived inputs from an original application standard. Exceptions such as NaN, Zero, Infinity, and Denormal numbers are also evaluated in the design. The implementation of DPAFPA primarily concentrates on minimizing the power consumption and energy or operation with the application of pipelining approach along with reduced processing time.

1.2 Double-Precision AFPA with Operand-Dependent Delay Elements

The technique of desynchronization outperforms the synchronous architecture and can be utilized for designing AFPA. On the other hand, during resynchronization, the clock signal is substituted by worst-case delay models; therefore, it cannot benefit from the event-driven feature of asynchronous circuits. In the speed of various sub-operation necessary for executing floating-point addition and an AFPA design with operand-dependent delay components is evaluated. The standard synchronous FPA proposed in with the FAR/CLOSE path structure has introduced a balanced 56-bit shifter with LOP and rounding by injection approach. Xu and Wang updated this synchronous FPA to take advantage of its event-driven nature by using asynchronous logic with variable-length delay components. Several AFPA sub-operations involving various calculation times must be determined to choose the delay models. Minimum six operations have been recognized and are processed at multiple speeds.

1.3 Multi-Threshold NULL Convention Logic (MTNCL)

A Multi-Threshold NULL Convention Logic (MTNCL) or Sleep Convention Logic (SCL) has been developed by Liang et al. in which is an integration of Multi-Threshold CMOS (MTCMOS) as well as NULL Convention Logic (NCL). Low Vt, i.e., high leakage current, rapid speed, and high Vt, i.e., low leakage current, slow pace, the transistors with various threshold voltages (Vt) are used in MTCMOS. To retain efficiency with minimal leakage, low Vt and high Vt are integrated with the design of MTCMOS. Once the circuit is not functioning, the MTCMOS enters into sleep mode and thereby helps in consuming minimal

power. On the other hand, sustaining sleep signals necessitates complex logic due to the time requirements as the synchronous circuits render the problem of transistor sizing and logic block partitioning. NCL, on the other hand, employs an asynchronous dual-rail design, which necessitates the application of two wires to compute a single bit, as well as a spacer or NULL signal, as seen in Figure 6. When MTCMOS and NCL are combined in MTNCL, the circuit can use sleep mode during NULL logic without worrying about the clock issues. The power-gating high Vt transistor is introduced in the pull-down network, which modifies the MTNCL architecture. The Static MTNCL threshold gate structure (SMTNCL) removes two bypass transistors and eliminates the output wake-up glitch.

In the case of single-precision floating-point co-processors, Liang et al. compared synchronous MTCMOS design with various NCL designs. The efficiency of co-processors is presented for operations such as addition, subtraction, and multiplication. So, only adding and subtraction operations performance is talked about in this research work. To handle data and NULL, an average time TDD is used by the MTNCL circuits, which is analogous to the synchronous clock period. Since the designs of multi-threshold do not offer any specific AFPA architecture, they are also included in this survey research due to the limited availability of AFPA literature. Table 2 shows that the comparison is limited to simple NCL designs (Low and High Vt), the optimum MTNCL design, and a synchronous MTNCL design. To explain the claimed optimal MTNCL (SMTNCL with SECRII w/o nsleep) design, it would be necessary to have a basic knowledge of existing SMTNCL architectures. The Early Completion Input-Incomplete (ECII) characteristic of MTNCL's fundamental architecture sets a process to sleep when all its inputs are NULL. To minimize power dissipation, a modified architecture called SECII sets the NCL circuit's combinational logic to sleep during the NULL cycle. When the circuit is not active, another form known as SECRII sets the completion, registration logic, and combinational logic to sleep. When such an SMTNCL circuit is integrated with bitwise MTNCL, the nsleep signal is no longer required, offering the SMTNCL a SECRII w/o nsleep design. When compared to synchronous MTCMOS design, the architecture proposed by Liang et al. found that this architecture was simulated for 25 sets of randomly selected floating-point integers. It utilized less than 86% of energy, three orders of magnitude less idle power, and 14% less area, and speed is slower, not less than 2.

2. LITERATURE SURVEY

Recently, there has been an increasing demand by users for DSP processors that perform efficiently. Therefore, hardware capable of processing high-speed signals and performing arithmetic floating-point operations is required to fulfill this requirement or demand. Initially, fixed-point algorithms were used in large numbers for implementing the algorithms on FPGA. Implementing floating points on Field Programmable Gate Array is considered one of the developing fields with recent advancements because FPGA development consumes less time and costs less, unlike the ASIC design. In this research work, we develop single and double-precision floating-point arithmetic operations. The same has been deployed on Field Programmable Gate Array for signal processing with the module of MAC through Verilog programming language. The primary aim of this work is to evaluate the area and the timing of floating-point units (FPUs) and the MAC units with single and double precision. On the Spartan 6 FPGA, the presented model is simulated and implemented (Ramesh, 2013).

In the case of floating-point (FP) multipliers, the hardware architectures based on FPGA are introduced in this research work. The deployment of single-precision (SP), double precision (DP), double-extended precision (DEP), as well as quadruple precision (QP) are all possible with the presented multiplier architectures. The conventional computational flow for floating-point multiplication is addressed in this research work. With the application of efficient Karatsuba methodology, the floating-point multiplication has been carried out for the complex module, i.e., mantissa multiplications, thereby enhancing the application of available in-built 25x18 DSP48E blocks on Xilinx Virtex-5 as well as on later FPGA devices. Compared to other conventional techniques, the proposed architecture shows enhanced performance with 1 DSP48 for SP, 3 DSP48 for DP, 6 DSP48 for DEP, and 18 DSP48 for QP multipliers (Ramesh, 2013).

This research presents a novel approach for dividing floating-point numbers depicted in the format of IEEE-754 single-precision (binary32). The suggested technique mainly relies on a multiplier as well as an inverter which is deployed as the integration of Parabolic Synthesis as well as second-degree interpolation. The proposed method is synthesized on a Xilinx Ultrascale FPGA and is independently implemented with or without pipeline stages. The proposed implementations show enhanced resource usage and latency performance compared to conventional methods. The suggested approach performs better than traditional techniques in terms of throughput; furthermore, a few Altera FPGAs achieve higher clock rates owing to variations in the DSP slice multiplier design (Mehta, 2013).

A floating-point number can simultaneously develop a high level of precision and a wide range of numbers. Floating-point multiplication is widely used in a variety of scientific and technological computations. Rapid, as well as efficient multipliers with a relatively small area as well as reduced power consumption, are required. In this research work, developing an IEEE-754 format

multiplier using Vedic Urdhva - Tiryagbhyam math concepts to support single-precision and double-precision format floating-point numbers has been carried out. The floating-point Multiplier presented in this framework handles overflow, underflow, and rounding. The presented work and traditional floating-point multipliers are based on Vedic mathematics, written in Verilog programming language, synthesized, and tested using the ISE Simulator (Defour, 2019).

A semi-parallel iterative decimal multiplier is presented in this research work. Compared to other conventional implementations presented in position, the proposed Multiplier employs BCD-8421 encoding, and recoding is not required for this framework. A new iterative partial product reduction technique, as well as semi-parallel partial product generation for faster multiplication, is employed in this work; a decimal 4:2 Adder is employed for partial product reduction. The proposed semi-parallel iterative design is implemented and validated using FPGA, and the results demonstrate that the proposed work outperforms with reduced delay in contrast to that of decimal multipliers and binary multipliers with double precision have been discussed in this research work (Kiran, 2017).

Several optimization methodologies have been presented in this research work for the algorithms based on a look-up table for double-precision floating-point arithmetic. The fundamental blocks of algorithms such as Multiplier (s) as well as an adder(s)) are re-engineered to enable the area's advantages and timing to operate efficiently based on evaluating various look-up table-based algorithms in the literary work. We design different look-up table optimization techniques for the algorithm proposed. In the double-precision floating-point module, we look at the trade-offs of exact rounding (0.5ulp) (unit in the last place). We utilize Wong and Goto's algorithms as a basic model to substantiate our optimization methods. The proposed algorithm's performance is compared with other algorithms based on performance and scalability metrics. The accuracy, i.e., the latency area of the Wong and Goto division algorithm, is enhanced by 26.94 percent (Merchant, 2016).

The most basic function in arithmetic modules is binary addition, and the Adder is considered the processor's essential arithmetic component. Full Adder is one of the critical features in Digital Signal Processing (DSP) architecture, microprocessor microcontroller applications, and data processing modules. Parallel multipliers are often used to accomplish better processing speeds at the cost of increased area efficiency. The performance comparison of various Full adder cells based on the transistor count is presented in this research work. This framework uses a cadence tool with 180nm technology and a 4-bit, 8-bit Braun multiplier architecture for effective layout implementation (Zhang, 2019).

The perception of power-efficient multipliers is used in this research work. It is considered one of the essential parts of all VLSI system designs because they offer high speed with low power consumption, which is one of the essential concerns for any VLSI design. With the help of shift and add techniques, an adaptive implementation of a high-speed, low-power multiplier is presented in this research work. This paper also presents the performance of the Braun multiplier and the Wallace Multiplier using the Cadence (Encounter) RTL Compiler along with simulation, which further involves developing the Test circuit for every module integrated to form the Multiplier. The Braun multiplier, as well as the Wallace multiplier, are modeled by designing a circuit diagram for each of the building blocks like the AND, OR, NOT, EXOR gates, Half Adder, as well as Full Adder, as well as evaluating each of the above blocks with a test circuit in this work. Further, with the help of the Cadence tool, these test circuits are simulated and synthesized (Oh, 2005).

Multipliers are essential in analog applications. Artificial neural networks, image processing, and modulators are some multipliers' applications. With the help of the Exponential Approximation circuit, the performance evaluation and implementation of low power and low andOS analog Multipliers are presented in this research work. MOSFETS are often used in this setup to accomplish low power dissipation by functioning in a weak inversion region. The Multiplier comprises four Exponential approximation circuits that execute on a 0.5V supply. Tanner tool uses 180nm technology to produce results and simulations (Oh, 2005).

Optimization of the area and a significant decrease in power consumption are essential factors for designing and implementing the DSP processor. The Finite Impulse Response Filter is considered one of the most critical components in the design and deployment of a DSP processor. Adder blocks, flip flops, and multiplier blocks are the three fundamental components of the Finite Impulse Response (FIR) Filter. Array and Booth multiplier were used to develop the Finite Impulse Response Filter and were compared with various constraints. The recommended filters are written in Verilog HDL programming language and executed with the help of Xilinx 14.7 ISE tools. Development has been seen for the area and delay (Jalaja, 2016).

This research mainly concentrates on a fixed-width, parallel multiplier design where the partial product array's eight least significant columns are compressed. It accepts two n-bit numbers as input, and the output obtained is the n-bit product. The Baugh-Wooley Multiplier is recommended in the case of 2's complement multiplication. Three multiplication units are used in the design to achieve a specified output. The combinational blocks are used in all of these units. The delay is successfully reduced through the parallel operation. The high efficiency of the circuit is driven by

substituting the inefficient design elements with efficient ones. Simulation is used to test the design's functionality (Jaiswal, 2014).

The deployment of 4 distinct 32-bit multiplier architectures is discussed in this research work, and the comparative study of multipliers' applications, speed, area, and power has been discussed. Booth multiplier, Wallace Tree Multiplier, Vedic Multiplier, and Dadda Multiplier are the four different multipliers defined in this work. Verilog programming language is used to develop and execute the multipliers, and the Xilinx ISE tool is used to synthesize the code. An enhanced version of the tree-based Multiplier is a Wallace tree multiplier. To minimize latency, the Wallace tree multiplier uses the Carry-Save addition algorithm. The basis of the Vedic Multiplier is Vedic mathematics. In Vedic multiplication, there are 16 tantras, with "Urdhva Tiryakbhyam" proving to be the best sutra (Jaiswal, 2015).

One of the significant parts of reducing the consumed power in VLSI systems is reducing the minimum supply requirements. A high-performance capacitance multiplier introduced in this research work can operate with supplies as low as ±0.25 V. It is designed on dynamically biased class-AB current mirrors to achieve maximum current efficiency. Conceptual assertions represented by the 11 capacitance multiplier factor measurements are manufactured in 180-nm CMOS technology. Furthermore, the same CMOS process is used to design and fabricate low-voltage precision rectifiers depending on similar class-AB current mirrors. Compared to quiescent currents, the produced output currents are 100 times greater (Jaiswal, 2016).

A unique low-power multiplication algorithm, as well as the architecture of VLSI, are presented in this research work. The proposed algorithm is simple and successively utilizes a 2n-1 constant number for multiplicand and a multiplier to determine NxN unsigned binary number multiplication. Compared to the traditional Multiplier, the proposed Multiplier illustrates that the reusability of hardware resources results in reduced power consumption and improves power delay products. The experimental results are further compared with the traditional Multiplier and a constant multiplier based on the result analysis of retiming. The proposed framework is structurally substantiated and synthesized using cadence EDA tools and has been deployed using 45nm technology libraries (Savas, 2017).

A standard stopband filter along with a complementary-defected ground structure (DGS) is presented in this work. The filter employs two distinct DGS patterns: On both sides of the filter, a Π -shaped DGS pattern is used, and in the middle, a button-headed H-shaped DGS pattern is used. Mutual inductance and mutual capacitance are employed between DGS patterns by the filter to enhance the filter's in-band gain-flatness, which is further beneficial for extending the bandwidth and improving the rejection ratio at low cut-off frequencies. The differential signal under the DGS filter is approximately stable, as welandommon-mode noise can be significantly lowered by 15 dB from 3.2 to 12.4 GHz as per the measured welandated results (Sangeetha, 2018).

The Karatsuba algorithm is used to develop an effective floating-point multiplier in this research work. Multiplications consume time as well as power and are used extensively in digital signal processing algorithms as well as in media applications. IEEE 754 format is used to represent floating-point numbers in binary form. The algorithm of Karatsuba multiplication is not dependent on the pipelined design and is implemented using Verilog HDL. Significant accumulation, the sign bit, and exponent arithmetic operations are implemented using this Multiplier. The design employs three pipelining stages with an 8-clock cycle latency (Hiratkar, 2016).

In the case of any modern computing system, floating-point multiplication is considered one of the vital components. The architecture of a customizable dual-mode double-precision floating-point multiplier that can handle two-parallel single-precision multiplication is introduced in this research work. This centralized double-precision dual (two-parallel) single-precision structure is the DPdSP Multiplier. The proposed framework is based on a typical advanced floating-point multiplication flow that can handle both standard and sub-normal operands, thereby presenting the ability of exceptional case handling. The suggested framework is typically implemented as an ASIC (UMC 90nm). Compared to conventional techniques, the proposed framework shows enhanced performance in terms of area, time, area×period, and throughput complexity measurements. By providing extra computation assistance, the proposed dual-mode architecture improves the measurements of the design (Havaldar, 2016).

Since the integer representation would no longer be feasible for representing very small or large values, an extensive range is necessary. The floating-point picture based on the IEEE-754 standard can depict these values. The IEEE 754-2008 standard is employed to implement a high-speed ASIC implementation of a floating-point arithmetic module that can execute addition, division subtr, action, and multiplication functions on 32-bit operands. The pre-normalization and post-normalization modules and exceptional handling ability are also all addressed in this research work. With the help of Verilog HDL, the proposed algorithms are designed as well as the Adder, subtractor, Multiplier, and divider, as well as square root's RTL code for is synthesized through Cadence RTL compiler, and the proposed architecture is intended for 180nm TSMC technology (Spoorthi, 2020).

In the case of the Floating Point (FP) division, this research work introduces a dynamically configurable and area-efficient multi-precision architecture. In the

technological and scientific areas, the division of floating-point FP is considered one of the fundamental computations. The double-precision (DP) division that can process dual (two-parallel) as well as single-precision (SP) computations are carried out by the DPdSP FP divider proposed in this work. The proposed design was primarily based on calculating division on series expansion strategy. We used 0.18m technology ASIC application with "OSUcells Cell Library" to monitor the proposed framework. The presented structure showed enhanced performance for throughput and area with the product of site and delay (Malkapur, 2020).

A unique adaptable multiple-precision multiply-accumulate (MAC) module for deep neural network training and inference is introduced in this research work. The MAC module supports some of the operations, such as fixed-point and floating-point. The presented module facilitates one 16-bit MAC operation, two 8-bit multiplications sum, and a 16-bit addend in floating-point format. The exponent's bit-width, as well as mantissa, can be efficiently replaced to make the MAC module to be more flexible. The suggested MAC module also facilitates fixed-point operations by defining the exponent's bit-width to zero. The MAC module of the proposed work can also be subdivided to assist four 4-bit multiplications as welandbit addend. The presented module of MAC enables accumulating eight 1-bit logic AND operations with minimum accuracy to help binary neural networks. The suggested unit of MAC offers greater flexibility with 21.8% computation complexity compared to a typical MAC module with 16-bit half-precision (Ushasree, 2013).

The implementation of hardware for analyzing arbitrary roots of a single-precision floating-point number is described in this research work. The implemented structure is primarily based on the GH CORDIC algorithm (Generalized Hyperbolic Coordinate Rotation Digital Computer). The system proposed in this work can determine the Nth root (Nx2) of a single-precision floating-point number using various floating-point numbers. Several measurements, such as precision, power consumption, efficiency comparison, and many more, were conducted after the successful implementation of the structure. However, according to the results obtained, the suggested technique determines the Nth root of a positive single-precision floating-point number along with a relative error of about 10 to 7. It thereby presents an error-flattening output (Wang, 2019).

To speed up media and data streaming, the floating-point module in a CELL processor's synergistic processor unit uses a fully pipelined 4-way SIMD module. This module aims to maximize the efficiency of critical single-precision multiply-add operations by assisting 32-bit single-precision floating-point and 16-bit integer operands with two distinct latencies. It consumes less power by making use of fine-grained clock gating. The architecture, logic, circuits, and implementation are co-designed to fulfill the parameters such as performance, energy, and area (Wang, 2019).
In the computations of various scientific and signal processing, the Floating Point (FP) multiplication has been extensively used, and expansion is considered one of the widely accepted arithmetic operations. Furthermore, the suggested model proposed in this work is compatible with IEEE-754 and thus can manage overflow, underflow, rounding, and a variety of exception scenarios. The structure accomplished a frequency of 414.714 MHz with 648 slices area (Zeng, 2015).

The exponential function is effectively computed with the help of double-precision numbers only when rounding is correctly performed. The exponential function must be determined with high precision to achieve the accurately rounded exponential with some arguments, and sometimes higher accuracy is required for small ideas. This research work introduces small-argument algorithms that are simple as well as fast. The proposed algorithm is integrated with other conventional methods to ensure optimum and average processing time. With the help of double-precision arithmetic, all the suggested algorithms accurately compute rounded exponential functions for all the rounding modes. Predetermined tables are often employed in the argument reduction phase. These algorithms are implemented by writing the code in C language and are found to be user-friendly.

3. PROBLEM STATEMENT

Repeating-point addition and subtraction are frequently employed arithmetic operations in most research areas. There are very few research papers on the design of an asynchronous floating-point adder. The conventional AFPA designs use dual-rail coding, which necessitates a wide application area because they are faster and consume less power than their standard FPA counterparts. This research work compared several performance aspects with their respective baseline FPA with all four contemporary AFPA designs. Since all the current models have multiple performance parameters, a complete evaluation is impossible.

On the other hand, the implementation of AFPA is the only design presented in this research work, and it demonstrates that the asynchronous methodology can enhance AFPA performance. It also explains the possible consequence of an AFPA developed using a bundled data protocol and a completion detection methodology. Based on the literature survey, power, area, and latency are more than existing methods.

4. PROPOSED DOUBLE PRECISION BASED ALU DESIGN

The following process was carried out to implement a latency-effective pipelined hardware divider. Initially, the size of the Look-up table and MUL bit width information was determined by conducting an error analysis on the implemented divider architecture in cases 1 and 2. Further, we choose the best possible block size with the help of Look-Up Table size and MUL bit-width, which is determined from the essential information. At last, by using optimal block size as a reference, the architecture of the divider is developed. Fig. 1 and 2 illustrate that once the LUT is accessed in the very first process, value A is generated in the second phase. The total computational latency is increased because the two tasks are carried out one after the other. The implemented architecture parallelizes multiplication and Look Up Table access to effectively minimize latency. Jeong's algorithm can be modified to obtain the advantage of parallelism.

$$\frac{P}{R} \approx (2\text{- AR}) \text{ AP} = (2 - \frac{1}{R_h^2}(R_h - R_l)\text{ R})\frac{1}{R_h^2}(R_h - R_l)\text{ P} \quad (1)$$

It is possible to parallelize the LUT access for as well as for multiplications $(R_h - R_1)$P as well as $(R_h - R_1$)R. Based on equation (1), the procedural steps are depicted in Figure 3. Jeong's algorithm has a latency of 1 LUT C 3 MULs, whereas the algorithm presented in this work has a latency of 3 MULs. In comparison to Jeong's algorithm, the clustering framework enables one more multiplier. To exploit the advantage of parallelism, based on Singh's algorithm the following equation can be written:

$$\frac{P}{R} \approx((2 - \text{ AR})^2 - (1\text{-AR}))\text{AP} = ((2 - \frac{1}{R_h^2}(R_h - R_l)\text{ R})^2 - (1 - \frac{1}{R}(R_h - R_l)\text{ R}))\frac{1}{R_h^2}(R_h - R_l)\text{ P}. \quad (2)$$

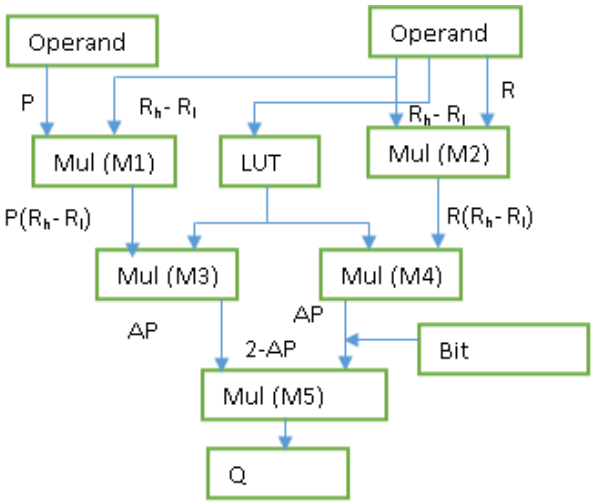

Figure 1. Block diagram of the proposed architecture DPFP division

The multiplications $(R_h - R_1)$P as well as $(R_h - R_1$)P, and LUT access for $\frac{1}{R_h^2}$, can be carried out simultaneously. Figure 4 indicates the proposed procedural steps for exploiting parallelism. The algorithm proposed in this work possesses a latency of 4 MULs along with another multiplier whereas, Singh's algorithm possesses a latency of 1 LUT C 4 MULs, whereas the proposed algorithm has a latency of 4 MULs and as well as one multiplier.

Since error analysis offers base data for determining the size of the Look-Up Table and base data for determining the bit widths of MULs, the proposed algorithm is essential for designing the architecture of the hardware divider. The error analysis for the above two scenarios is performed, and the optimal block size is determined in this section.

Table 1 shows the comparison results of delay, area cost of the proposed strategy, and the existing pipelinable division algorithms. Each algorithm's overall delay or total area cost is derived by summing every pipeline iteration's delays or area costs. Table 1 shows that Case 1 in the proposed methodology might approximately decrease the critical path time by 16 % and a 21% increase in hardware area compared to that of (Defour, 2019). With a 45% increase in the hardware area, the proposed technique in Case 2 could minimize the critical path time by approximately 7% compared to that of (Kiran, 2017). In contrast to (Defour, 2019), the proposed strategy in Case 1 can minimize the depth of the pipeline by one step and possess the same delay (12.0) as that of (Defour, 2019) within a channel. This suggests that when contrasted to (Defour, 2019) with almost the same clock frequency, the proposed methodology in Case 1 can reduce the delay of the pipeline by 25%.

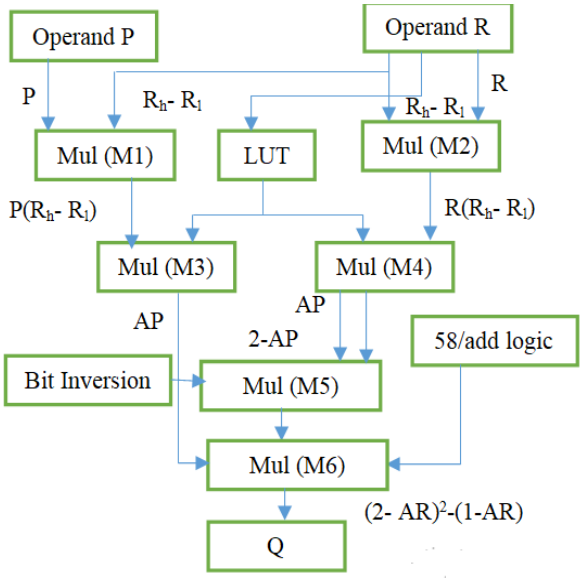

Figure 2. Block diagram of the proposed architecture for DPFP of Multiplication

This outcome corresponds to the suggested approach in Case 2, which can further reduce the uncertainty of the channel by 20% when contrasted to (Kiran, 2017) at the same clock frequency. The proposed methodology in Case 2 minimizes the area's cost by 28% and has the same pipeline depth as that of (Defour, 2019) but with an 8% slower clock frequency. For precise evaluation, the suggested architecture was synthesized using a 28nm process along with the Synopsys design compiler, and thereby the execution result was contrasted to that of existing models (Defour, 2019 and Kiran, 2017). The pipelined delay was reduced by 22% while increasing the area by 34% over (Defour, 2019), and these comparisons are illustrated in Table 2. Compared to (Kiran, 2017), Case 2 of the proposed algorithm reduces the pipeline delay by approximately 11% while increasing the area by 33%. The delay time reduction rate of Case 1 was increased from 16% to 22% when contrasted to that in (Defour, 2019), as well as the hardware area was increased from 21% to 34% and at last, the result comparisons of table 2, as well as Table 1, are carried out. In contrast to (Kiran, 2017), the delay time reduction rate of Case 2 significantly improved from 7% to 11%, whereas the hardware area reduced from 45% to 33%. Therefore, compared to the existing algorithms (Defour, 2019) and (Kiran, 2017), the size of the hardware shown in table 1 and table 2 was enhanced equally by 33%.

Power consumption is typically proportional to the area associated with it. Therefore, area-timing products (ATP) based on area as well as timing values are commonly utilized in domains similar to the one used in this work, as well as ATP outcomes are employed as indicators of power consumption (Havaldar, 2016), (Spoorthi, 2020). As a result, we used the ATP model to evaluate and analyze the preceding algorithms in terms of power consumption. The proposed algorithm's ATP outcomes are compared with the existing techniques (Defour, 2019) and (Kiran, 2017) and are illustrated in Table 4. ATP was determined in the first phase by categorizing LUT and the multiplier in 1 and two proposed scenarios. Since the delay time, as well as the power consumption of the two systems, is unique, the computation is accurate. According to the analysis of ATP, the power consumption of the suggested pipeline divider was approximately 37%. It was equal to a 33% increase in hardware size compared to conventional algorithms. It was concluded that power consumption and area are proportional to each other from these observations. The algorithm of the pipelined division was previously utilized in (Zhang, 2019). There exists a lot of discrepancies between (Hao Zhang 2019) as well as the proposed approach. Computations of fractions and error analysis on double-precision accuracy are to be performed or not are some of the critical variations offered as well (Zhang, 2019). Whenever floating-point arithmetic is employed as a fraction estimation technique in (Zhang, 2019), the hardware size or operational frequency may be higher or lower than the recommended fixed-point arithmetic technique.

5. COMPARISON WITH EXISTING MULTIPLICATION METHODS

The quotient is generated by approximating the multiplicative algorithm with the application of hardware that integrates a floating-point multiplier and a LUT.

Table.1 Comparison between the conventional adder and proposed ternary-based double precision floating point adder

Adders	**Slices**	**Delay (ns)**	**FF's**	**Power (W)**	**Slice LUT's**	**Area (Occupied Slices)**	**Memory**	**Frequency MHz**
(Spoorthi, 2020)	1140	2.87	2021	7.69m				
(Srujana, 2020)	1047	11.15	1724	36.34m				
(Spoorthi, 2020)	1268	9.24	1536	0.4 µ				
(Srujana, 2020)	992	82.21	4990	13.54m				
(Addanki, 2013)	1655	35.2	1642	0.6µ				
Proposed Ternary based double precision floating point adder	**892**	**1.340**	**1201**	**0.082** µ	**629**	**3071**	**137KB**	**127.09**

Table.2. Comparison between conventional multiplier and proposed Multiplier

Multiplier	Slices (area)	LUT	Delay (ns)	Power (mW)	Area* delay	Time* power	Area*time*power	FF's/ Memory	Frequency
(David, 2019)	12095	7620	8.4	1.56					
(Oh, 2005)	4520	9841	5.32	0.94					
Proposed ZP & FRBM	**3819**	**4951**	**3.841**	**0.088**				**2100/132**	

Table.3 Comparison between conventional adder and proposed ternary based double precision floating point subtractor

Adders	Slices	Delay (ns)	FF's	Power (W)	Slice LUT's	Area(Occupied Slices)	Memory	Frequency MHz
Kogge-Stone (Spoorthi, 2020)	7209	6.41	4821	0.97	3061			
Sklansky (Srujana, 2020)	5620	4.9	3875	2.40	2971			
Proposed Ternary-based double precision floating point subtractor	**2820**	**3.841**	**2861**	**0.088**	**1672**	**1863**	**132**	**52.637**

Table.4. Comparison between conventional Division and proposed Division

Multiplier	Slices (area)	LUT	Delay (ns)	Power (mW)	Area* delay	Time* power	Area*time*power	FF's/Memory	Frequency
(Merchant, 2016)	4921	2901	7.81	1.45					
(Jalaja, 2016)	6320	1974	4.76	1.09					
Proposed ZP & FRBM	**2085**	**1864**	**3.84**	**0.082**				**2843/133**	**180.865MHz**

6. CONCLUSION

Double precision floating-point addition and subtraction are frequently employed arithmetic operations in most research areas. There are very few research papers on the design of an asynchronous floating-point adder. The conventional AFPA designs use dual-rail coding, which necessitates a wide application area because they are faster and consume less power than their standard FPA counterparts. This research work compared several performance aspects with their respective baseline FPA with all four contemporary AFPA designs. Since all the current models have multiple performance parameters, a complete evaluation is impossible. On the other hand, the implementation of AFPA is the only design presented in this research work, and it demonstrates that the asynchronous methodology can enhance AFPA performance. It also explains the possible consequence of an AFPA developed using a bundled data protocol and a completion detection methodology. A unique low-latency pipelined divider structural design for double-precision numbers was developed in this research work. Compared to other existing techniques, the proposed architecture minimizes the pipeline's depth by one step. The algorithm presented in this work can also reduce the computational latency without increasing the size of the Look-Up Table and is implemented on two traditional divider architectures. The given divider architecture is more suitable for systems requiring double-precision division.

References:

Defour, D., et al. (2001). Correctly rounded exponential function in double-precision arithmetic</title> *Proceedings of SPIE, the International Society for Optical Engineering/Proceedings of SPIE*. https://doi.org/10.1117/12.448644

Havaldar, S., et al. (2016). Design of Vedic IEEE 754 floating point multiplier. *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. Bangalore, India, 1131-113. https://doi.org/10.1109/RTEICT.2016.7808008

Hiratkar, S., et al. (2016). VLSI design of analog multiplier in weak inversion region. *2016 IEEE International Conference on Communication and Signal Processing (ICCSP)*. Melmaruvathur, India, 0832-0835. https://doi.org/10.1109/ICCSP.2016.7754262

Jaiswal, M. K., et al. (2014). Configurable architecture for double / two-parallel single precision floating point division. *2014 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. Tampa, FL, USA, 332-337. https://doi.org/10.1109/ISVLSI.2014.45

Jaiswal, M. K., et al. (2015). Dual-mode double precision / two-parallel single precision floating point multiplier architecture. *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. Daejeon, Korea (South), 213-218. https://doi.org/10.1109/VLSI-SoC.2015.7314418

Jaiswal, M. K., et al. (2017). DSP48E efficient floating point multiplier architectures on FPGA. *2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems*. Hyderabad, India, 1-6. https://doi.org/10.1109/ICVD.2017.7913322

Kiran, D. K., et al. (2017). VLSI implementation of Braun multiplier using full adder. *2017 IEEE International Conference on Current Trends in Computer, Electrical, Electronics, and Communication (ICCTCEEC-2017)*. Mysore, India, 499-504. https://doi.org/10.1109/CTCEEC.2017.8455157

Malkapur, S. B., et al. (2020). Design of generic floating point pipeline based arithmetic operation for DSP processor. *2020 IEEE International Conference on Inventive Research in Computing Applications (ICIRCA)*. Coimbatore, India, 1059-1064. https://doi.org/10.1109/ICIRCA48905.2020.9182948

Mehta, A., et al. (2013). Implementation of single precision floating point multiplier using Karatsuba algorithm. *2013 IEEE International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*. Chennai, India, 254-256. https://doi.org/10.1109/ICGCE.2013.6823439

Merchant, F., et al. (2016). Efficient realization of table look-up based double precision floating-point arithmetic. *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems*. Kolkata, India, 415-420. https://doi.org/10.1109/VLSID.2016.113

Oh, H. J., et al. (2005). A fully-pipelined single-precision floating-point unit in the synergistic processor element of a CELL processor. *2005 Symposium on VLSI Circuits Digest of Technical Papers*. Kyoto, Japan, 24-27. http://doi.org/10.1109/VLSIC.2005.1469325

Oh, H. J., et al. (2005). A fully-pipelined single-precision floating-point unit in the synergistic processor element of a CELL processor. *IEEE Journal of Solid-State Circuits*, *41*(4), 759–771. https://doi.org/10.1109/JSSC.2006.870924

P, S., et al. (2018). Comparison of Braun multiplier and Wallace multiplier techniques in VLSI. *2018 Fourth International Conference on Devices, Circuits and Systems (ICDCS)*. Coimbatore, India, 48-53, https://doi.org/10.1109/ICDCSyst.2018.8605173

Ramesh, A. P., et al. (2013). 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)*, Tiruvannamalai, India, 2013, pp. 1-5. https://doi.org/10.1109/ICEVENT.2013.6496575

Ramesh, A. P., et al. (2013). An FPGA based high-speed IEEE-754 double precision floating point multiplier using Verilog. *2013 IEEE International Conference on Emerging Technology Trends in Electronics, Communication and Networking*. Tiruvannamalai, India, 1-5. https://doi.org/10.1109/ICEVENT.2013.6496575

S, J., et al. (2016). Design of low power based VLSI architecture for constant multiplier and high-speed implementation using the retiming technique. *2016 IEEE Microelectronics Conference (MicroCom)*. https://doi.org/10.1109/MicroCom.2016.7522503

Savas, S., et al. (2017). Efficient single-precision floating-point division using harmonized parabolic synthesis. *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. Durgapur, India,1-6. https://doi.org/10.1109/ISVLSI.2017.28

Spoorthi, M. N., et al. (2020). A Decimal multiplier with improved speed using semi-parallel iterative approach. *2020 24th International Symposium on VLSI Design and Test (VDAT)*, Bhubaneswar, India, 1-6. https://doi.org/10.1109/VDAT50263.2020.9190260

Ushasree, G., et al. (2013). VLSI implementation of a high-speed single precision floating point unit using Verilog. *2013 IEEE Conference on Information and Communication Technologies (ICT)*. Thuckalay, India, 803-808. https://doi.org/10.9790/2834-10114954

Wang, Y., et al. (2019). GH CORDIC-based architecture for computing Nth root of single-precision floating-point number. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *28*(4), 864-875. https://doi.org/10.1109/TVLSI.2019.2959847

Zeng, Z., et al. (2015). A Wideband common-mode suppression filter with compact-defected ground structure pattern. *IEEE Transactions on Electromagnetic Compatibility*, *57*(5), 1277-1280. https://doi.org/10.1109/TEMC.2015.2440424

Zhang, H., et al. (2019). New flexible multiple-precision multiply-accumulate unit for deep neural network training and inference. *IEEE Transactions on Computers*, *69*(1), 26-38. https://doi.org/10.1109/TC.2019.2936192

Nagarathna R
Department of Electronics &Telecommunication Engineering, Dayananda Sagar College of Engineering, Visvesvaraya Technological University, Karnataka, India,
nagarathnatce@dayanandasagar.edu
ORCID 0000-0003-2783-1886

A R Aswatha
Department of Department of Electronics &Telecommunication Engineering, Dayananda Sagar College of Engineering, Visvesvaraya Technological University, Karnataka, India
aswath.ar@gmail.com
ORCID 0000-0001-8423-4071