Design of efficient reversible floating-point arithmetic unit on field programmable gate array platform and its performance analysis

Girija Sanjeevaiah[1], Sangeetha Bhandari Gajanan[2]
[1]Department of Electronics and communications, Dr. Ambedkar Institute of Technology, Bangalore, India
[2]Department of Electronics and communications, RNS Institute of Technology, Bangalore, India

Article Info

Article history:

Received Dec 28, 2021
Revised Jul 13, 2022
Accepted Aug 10, 2022

Keywords:

Adder
Arithmetic unit
Divider
Floating point
Multiplier
Reversible gates
Subtractor

ABSTRACT

The reversible logic gates are used to improve the power dissipation in modern computer applications. The floating-point numbers with reversible features are added advantage to performing complex algorithms with high-performance computations. This manuscript implements an efficient reversible floating-point arithmetic (RFPA) unit, and its performance metrics are realized in detail. The RFP adder/subtractor (A/S), RFP multiplier, and RFP divider units are designed as a part of the RFP arithmetic unit. The RFPA unit is designed by considering basic reversible gates. The mantissa part of the RFP multiplier is created using a 24x24 Wallace tree multiplier. In contrast, the reciprocal unit of the RFP divider is designed using Newton Raphson's method. The RFPA unit and its submodules are executed in parallel by utilizing one clock cycle individually. The RFPA unit and its submodules are synthesized separately on the Vivado IDE environment and obtained the implementation results on Artix-7 field programmable gate array (FPGA). The RFPA unit utilizes only 18.44% slice look-up tables (LUTs) by consuming the 0.891 W total power on Artix-7 FPGA. The RFPA unit sub-models are compared with existing approaches with better performance metrics and chip resource utilization improvements.

Corresponding Author:

Girija Sanjeevaiah
Department of Electronics and communications Engineering, Dr. Ambedkar Institute of Technology
Bangalore, India
Email: girija.pari@gmail.com

1. INTRODUCTION

The low-power design is the prime factor when designing the high-performance, very large-scale integrated (VLSI) system. The speed and dynamic ranges features are to be considered while creating the low-power designs which fit the portable devices. The high-integration density and high-speed features are merged to emphasize the high throughput computation in portable devices with a significant reduction in heat dissipation factor. There is enormous development in electronic semiconductor industries to improve parameters like chip area, power, and time features by optimizing the electronic architecture framework-one effective way to avoid power loss in the VLSI circuits is by using the reversible gates. The reversible logic gates map the input vectors and recover quickly from the output vectors and vice-versa. The reversible logic gates are used in enormous quantum computing applications, communications, low-power applications, digital signal processing (DSP), and many more to improve power dissipation factors. The power factor reduction in reversible logic gates depends upon the usage of the reversible logic gates, quantum cost (QC), number of constant input (CI) usage, and garbage output (GO) generation [1], [2].

The numbers are fixed-point or floating-point in computer machines, depending on the requirements and application usage. Most fields like mathematics, DSP applications, engineering, finance, and technology require performing operations by manipulating real numbers. The floating-point numbers support a wide range of real values, but it needs a high-performance processing unit (hardware) or advanced high-end software for operations implementation. So, floating-point arithmetic unit is used to perform robust computation for real-time application on hardware with optimization features, improving chip area, power, and timing complexity [3]–[5]. Most complex algorithms are impractical to implement in real-time scenarios, even in modern computers, using conventional representation approaches. Floating-point numbers are represented per IEEE 754 standards and vary precision by providing the standard specific format. The reversible features are added to the floating-point numbers to improve the computational performance by reducing the hardware complexity [6]. Many existing approaches have been available for designing reversible floating-point arithmetic operations like adders, subtractors, multipliers, and dividers. The parity preserving reversible gate is introduced in reversible floating-point (RFP) adders to improve the Nanometric area and computational performance metrics [7]. The realization of new reversible gates with multi-functional operations is added advantage to enhancing the performance metrics in arithmetic and logic unit (ALU) and RFP arithmetic units for immense real-time applications [8]–[10].

The efficient reversible floating-point arithmetic (RFPA) Unit is designed in this manuscript. The RFPA unit is synthesized and implemented in the Artix-7 field programmable gate array (FPGA) environment. The proposed RFPA unit offers cost-effective solutions to real-time applications by providing better latency, power, and less chip area utilization. The proposed RFPA unit sub-modules are compared with existing approaches with better improvement in garbage output (GO), constant inputs (CI), and quantum cost (QC). The significant contribution of the work is listed as follows:

– The proposed RFPA unit is constructed using basic reversible gates with gate-level architectures. The RFAP unit executes each arithmetic operation concurrently and is combinational.
– The RFAP unit offers low latency by executing each arithmetic operation in a single clock cycle.
– The RFP Adder or subtractor module is designed as a single module, so the user can perform addition or subtraction by changing the mode input.
– The shifting operation during the normalization process is not considered in both the RFP adder and RFP Multiplier modules to reduce the hardware complexity.
– The RFP divider is constructed using RFP adders and multipliers for reciprocal operations to improve accuracy.

Section 2 provides an overview of reversible gates used in the RFPA unit. The proposed RFPA unit architecture is explained in detail in section 3. The results and discussion of the RFPA unit and its sub-modules are analyzed with comparison in section 4. Finally, it concludes overall works with improvements in section 5.

This section explains the existing RFP modules like adders, subtractors, multipliers, dividers, and arithmetic units with its findings. Nachtigal *et al.* [11] presented the RFP adder module as IEEE 754 specifications. The RFP adder mainly has a swap, alignment unit, conversion unit, normalization unit, and rounding unit. The work analyzes the performance metrics like GO, CI, and QC. The design approach is conventional and utilizes more QC. Nguyen and Meter [12] describe the efficient RFP adder for a quantum computing system. The work introduces new reversible gates for the construction submodules of the RFP adder unit with fault-tolerant features. The design reduces the 68% quantum cost compared to conventional adder units. Alaghemand and Haghparast [13] present the new RFP adder unit with performance improvements. The design analyzes the performance metrics like GO, CI, and QC and improves them with existing approaches. AnanthaLakshmi and Sudha [14] describe the RFP adder module with synchronous features. The design introduces a new reversible gate for data–flip-flop construction by reducing the transistor count. The work achieves 411 mW of power, executes at two clock cycles, and operates at 41 MHz on Virtex-5 FPGA.

Nagamani *et al.* [15] presented an RFP adder module for DSP applications. The work uses a carry look ahead (CLA) adder and Brent-Kung (BK) adder for mantissa calculation. The design realizes the performance metrics using both CLA and BK adders, improving existing approaches. AnanthaLakshmi and Sudha [16] present the Fused 32-bit RFP arithmetic unit for DSP applications. The work includes different arithmetic components like RFP adder/subtractor, RFP multiplier, RFP divider, RFP square root, and performance realization. The Finite impulse response (FIR) filter and fast Fourier transformation (FFT) modules are designed and verified with simulation results to validate the design modules. Khan and Sundhari [17] present the pipelined double-precision RFP adder/Subtractor modules with a fault-tolerant feature on FPGA. The work realizes both single and double-precision RFP units with its performance. The double-precision RFP adder/subtractor module works at 370.3 MHz by utilizing 71% logic units, with a dynamic power consumption of 51.97 mW.

Nachtigal *et al.* [18] elaborate on the RFP multiplier architecture using an operand decomposition mechanism. The design uses the Wallace tree multiplier for mantissa calculation with the help of 4:2 compressors units. The RFP multiplier realizes the performance metrics by obtaining a delay of 912 and QC of 6,957. Jenath and Nagarajan [19] present the RFP Multiplier module on FPGA. The work uses a standard conventional 24x24 multiplier for mantissa calculation. The designed multiplier is compared with the existing approach with better improvement in QC and Delay parameters. Malathi *et al.* [20] present the single-precision RFP multiplier with cost-effective features. The design uses a TSG gate for cost optimization and is adopted for 24x24 multiplier calculation. Arunachalam *et al.* [21] presented the RFP multiplier with new gates. The works introduce the three reversible gates for sub-model designs used in the RFP multiplier. The work optimizes the QC more than the existing approach. Jain *et al.* [22] describe single-precision and double precision-based RFP multipliers. The reversible 4:2 compressors are used for multiplier design for mantissa calculation. The work realizes the performance metrics for both the design modules with improvements to existing works.

Kamaraj and Marichamy [23] describe the RFP division architecture with fault-tolerant features. The design uses KMD gates which support many logic functions with optimization. The restoring and non-restoring division units are designed as per IEEE 754 standards. The work analyzed the performance metrics and compared them with existing approaches with better improvements. Muñoz *et al.* [24] present the FP library for arithmetic units using FPGA. The arithmetic units like FP adder/ subtractor, FP multiplier, FP divider, FP square root units are designed and realized on FPGA. The FP arithmetic core analyzes the chip area, power, and tradeoff dependence parameters on FPGA. Jamal and Babu [25] present the RFP divider using high-speed division array modules. The work realizes the performance metrics like QC, GO, and CI for 2-bit, 4-bit, 8-bit, and 16-bit divider modules. Gayathri *et al.* [26] elaborate on the Single-precision RFP division unit with T-count optimization. The work uses restoring and non-restoring algorithms for division unit design concerning quantum Clifford plus T gate set. In contrast, the RFP division module using the Goldschmidt algorithm is designed. The results of restoring and non-restoring Goldschmidt division algorithms are discussed in detail. Przybyl [27] discuss the fixed-point arithmetic unit on FPGA for embedded applications with scalable features. The real-number calculation is accessible on real-time processing applications, providing faster and better processing efficiency. The reversible logic gates are used extensively in most of the processors for high speed computations [28]–[31] and also in image processing [32] and video processing applications [33] for high resolution outcomes.

2. REVERSIBLE GATES

The reversible gates are used to implement more than one operation. Reversible gates are modeled using the quantum gate library. Reversible gates and a few modules used in the RFP arithmetic unit design are illustrated in Figure 1 (in appendix). The Feynman gate (FG) is a 2x2 gate with a QC of 1, represented in Figure 1(a). The Fredkin gate (FR) is a 3x3 gate with a QC of 5, and it is illustrated in Figure 1(b). The Peres gate (PG) is a 3x3 gate with a QC of 4, and it is represented in Figure 1(c). The multiple controlled Fredkin (MCF) gate [1] is a 3x3 gate used for the construction of reversible AND and OR gates, and it is represented in Figures 1(d) and 1(e), respectively. The reversible AND and OR gates use the QC of 3 with one CI and two GOs. The modified Fredkin gate (MFG) is a 3x3 gate with a QC of 4, represented in Figure 1(f). The reversible multiplexor (MUX) gate is designed using MFG, illustrated in Figure 1(g). The MUX gate uses two GO with a QC of 4. The reversible half adder/subtractor (RHAS) is designed using two FGs and one PG. The RHAS utilizes one CI, two GO with a QC of 6, represented in Figure 1(h). The reversible full adder/subtractor (RFAS) is designed using two FGs and two PGs. The RFAS utilizes one CI, three GO with a QC of 10, represented in Figure 1(i).

3. REVERSIBLE FLOATING-POINT ARITHMETIC (RFPA) UNIT

The single-precision 32-bit Floating point number representation is illustrated in Figure 2 as per IEEE 754 format. It mainly contains a 1-bit sign (31st bit), 8-bit exponent (30:23), and 23-bit mantissa (22:0) for the formation of a single-precision 32-bit FP number. This representation is used further in the RFP arithmetic unit and its sub-modules.

The 32-bit RFP arithmetic unit architecture is illustrated in Figure 3. The RFPA unit consists of a 32-bit reversible multiplexor (MUX), RFP adder, RFP subtractor, RFP multiplier, and RFP divider units. The 2-bit mode is used as a select line to the MUX unit to select the corresponding units. RFP adder output is chosen if the mode is “00”. Similarly, for “01”, the RFP subtractor unit output, for “10”, the RFP multiplier unit output, and for “11”, the RFP divider output is selected. The detailed description of the submodules of the RFP arithmetic unit is explained in the below section.

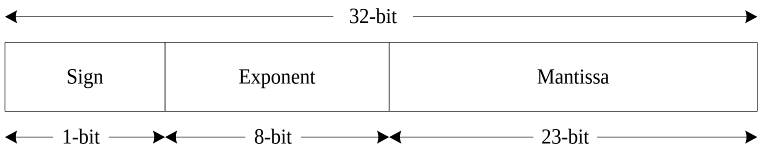


Figure 2. 32-bit single-precision floating point number representation

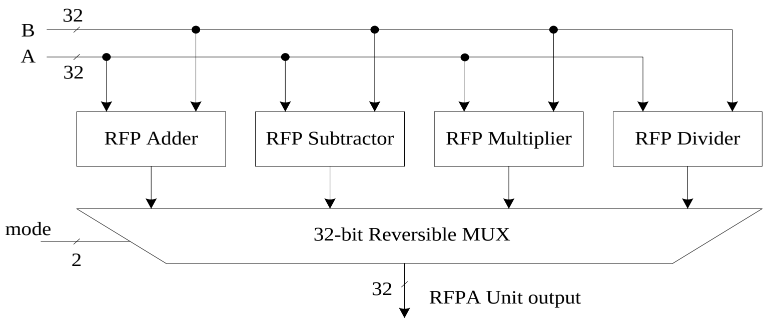


Figure 3. RFP arithmetic unit architecture

3.1. 32-bit RFP Adder/Subtractor

The representation of 32-bit RFP Adder/Subtractor architecture is illustrated in Figure 4. The 32-bit RFP Adder/Subtractor module uses reversible logic gates by framing the sub-modules like an adder, subtractor, multiplexers, barrel shifters (left and right), and comparators. The 32-bit RFP Adder/Subtractor architecture is constructed using the following steps:

a. First, consider the first 31 bits of A and B for comparison; if (A [30:0] < B [30:0]), then swap else, keep it as it is.
b. Extract the exponent and mantissa bits using step a (first step). For exponent: EA and EB using A [30:23] and B [30:23], For mantissa: MA and MB using A [22:0] and B [22:0].
c. Check whether all the bits of EA and EB are one; if yes, then the number is infinity or not a number (NAN). It means an exception.
d. Perform XOR operation of MSB bits (A [31] and B [31]) to generate the final sign bit.
e. If all the bits of EA and EB are zero, then the number is denormalized, and an implicit bit of the corresponding mantissa (MA or MB) is set to zero.
f. Perform the exponent difference calculation and shift either mantissa of MA or MB according to the difference ($EA > EB$).
g. If exponents are the same ($EA == EB$), then perform 24-bit addition using mantissa data and barrel shifted data.
h. Finally, normalize and remove the implicit bit to form addition's final 23-bit mantissa output.
i. Find the maximum exponent value using EA and EB to generate the 8-bit final exponent bit.
j. To concatenate the final sign bit with 8-bit final exponent and final 23-bit mantissa data to form the 32-bit RFP adder output.
k. For subtraction operation, consider the right-shifted output for 2's complement operation and then perform addition with mantissa data.
l. The added output is used for the normalization and later performs the shifting operation (left) using the exponent (EA) value.
m. The final 8-bit exponent and 23-bit mantissa value is generated after normalization and shifting operation for subtraction operation.
n. Finally, concatenate the final sign bit with 8-bit final exponent and final 23-bit mantissa data to form the 32-bit RFP subtraction output.

In the RFP adder/subtractor module, the 31-bit comparator unit is designed using NOT, FG, and PGs to generate A less than B (A<B) operation. The swapping unit is created using two reversible 32-bit MUXs with the select line of A<B. The denormalized and implicit bit addition is performed using two reversible 8-bit OR reduction units and two 24-bit MUXs. The exponent difference calculation is performed using 8-bit reversible adder and subtractor units. The right shifting operation is performed using a 24-bit reversible Barrel shifter. The maximum exponent value is calculated using an 8-bit reversible adder and an

8-bit MUX unit. The normalization with the removal of implicit bit operation for addition operation is performed using 23-bit MUX. The 24-bit 2’s complement unit is designed using NOT gates and a 24-bit reversible adder. The normalize and shift function uses a 24-bit barrel shifter (left), 25-bit two’s complement unit, and 8-bit reversible subtractor units.

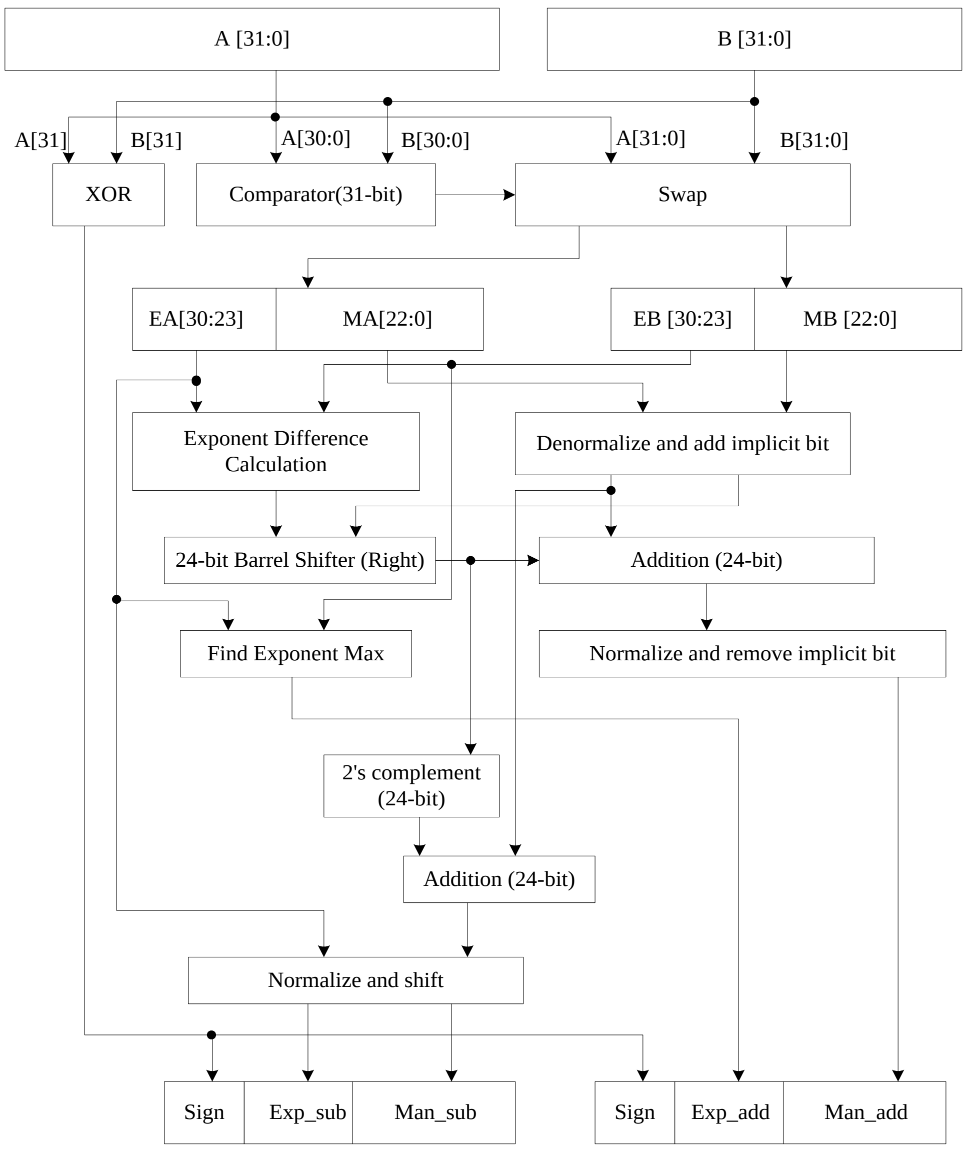


Figure 4. Representation of 32-bit RFP adder/subtractor

3.2. 32-bit RFP multiplier

The 32-bit RFP multiplier architecture is illustrated in Figure 5 as per IEEE 754 standards. The 32-bit RFP multiplier module contains a reversible 24x24 Wallace tree multiplier, 8-bit adder/subtractor units, 24-bit adders, and a normalization unit. The 32-bit RFP multiplier architecture is constructed using the following steps:

a. First, check whether all the bits of exponents EA and EB are one; if yes, the number is either infinity or not a number (NAN). i.e., exception.
b. If all the bits of EA and EB are zero, then the number is denormalized, and the implicit bit of the corresponding mantissa (MA or MB) is set to zero.
c. Perform 24-bit reversible Wallace tree multiplication for MA and MB bits to generate 48-bit mantissa product.
d. The MSB of the mantissa product (47th bit) will be active as the select line to find out the round of bit.

e. Perform normalization:
   - If the MSB of the mantissa product is 1, then the mantissa product is already normalized and considers the next 23-bits after the MSB bit.
   - If the MSB of the mantissa product is 0, so the next bit is always 1, and consider after next-to-next bit of 23-bits. So, no need to perform any of the shifting operations.
   - Add the round-off bit to the normalized result to generate the final 24-bit mantissa output.
f. Perform EA+EB-127 operation to generate the final exponent output.
g. Perform XOR operation of MSB bits (A [31] and B [31]) to generate the final sign bit.
h. Concatenate final sign bit, final exponent, and final mantissa output to generate the final 32-bit RFP multiplier output.

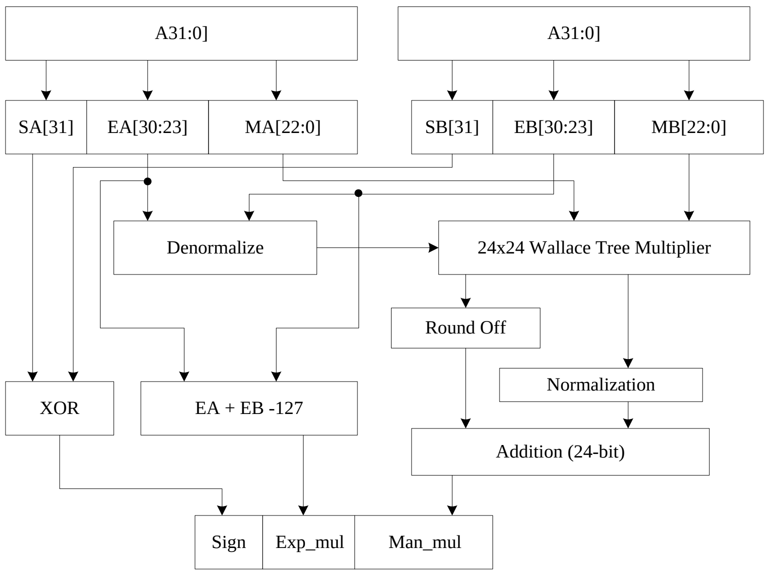


Figure 5. Representation of 32-bit RFP multiplier

The 24×24 reversible Wallace tree multiplier uses nine 8-bit reversible Wallace tree multipliers and eight 48-bit reversible adder units. The single 8-bit reversible Wallace tree multiplier uses reversible AND gates, reversible half, and full adders. The round-off bit was generated using two 24-bit reversible OR reduction operations and one multiplexer unit. The normalization unit is constructed using a 24-bit reversible MUX unit and one 24-bit reversible adder to generate a 24-bit mantissa. The EA+EB-127 operation uses an 8-bit reversible adder and 9-bit reversible adder units to generate the final 8-bit exponent bits.

3.3. 32-bit RFP divider

The 32-bit RFP divider is illustrated in Figure 6. The reciprocal unit of the RFP divider is designed using Newton-Raphson’s method [28]. It mainly contains the exponent difference calculation using 8-bit reversible subtractor and adder units, followed by reciprocal units using RFP adders and RFP multipliers. The 32-bit RFP divider is constructed using the following steps and is as follows:

a. First, check whether all the bits of EA and EB are one; if yes, then the number will be either infinity or not a number (NAN). i.e., exception
b. Perform XOR operation of MSB bits to generate the final sign bit.
c. Calculate the exponent difference (EA-EB).
d. Formation of dividend (D_n) using exponent difference and MA.
e. Construction of Divisor (D_d) using MB.
f. Generate the final Exponent and mantissa results using Newton-Raphson’s method
g. Concatenate the final sign bit, exponent, and mantissa output to generate the final 32-bit RFP Division output.

The reciprocal unit is constructed using Newton-Raphson’s method as follows:

- Define the Coefficient values used in reciprocal calculations: C_1 = 2.8235, C_2 = -2.17647 and C_3 = 2.
- First, calculate the iteration (I_0) using (1) for the reciprocal unit ($1/D_d$) of the divisor value (D_d).

$$I_0 = C_1 + C_2 D_d \tag{1}$$

– Compute the accurate iterations I_1, I_2, and I_3 for the reciprocal unit (for single-precision iteration value is fixed to 3). The successive i^{th} iteration (I_{i+1}) is calculated using (2) as:

$$I_{i+1} = I_i(C_3 + D_d I_i) \tag{2}$$

where i = 0, 1 and 2.

– Lastly, calculate the quotient (Q) value by multiplying the dividend (D_n) using a reciprocal unit of the divisor value (D_d), and it is represented using (3) as (3):

$$Q = D_n . I_3 \tag{3}$$

The performance parameters of the RFP arithmetic unit's sub-modules like GC, CI, GO, and QC is summarized in Table 1. The normalization and shift unit and Wallace tree multiplier unit consume more resources than the corresponding RFP subtractor and RFP multiplier. The 24-bit barrel shifter and 2's complementor are part of the normalization unit in RFPA/S unit.

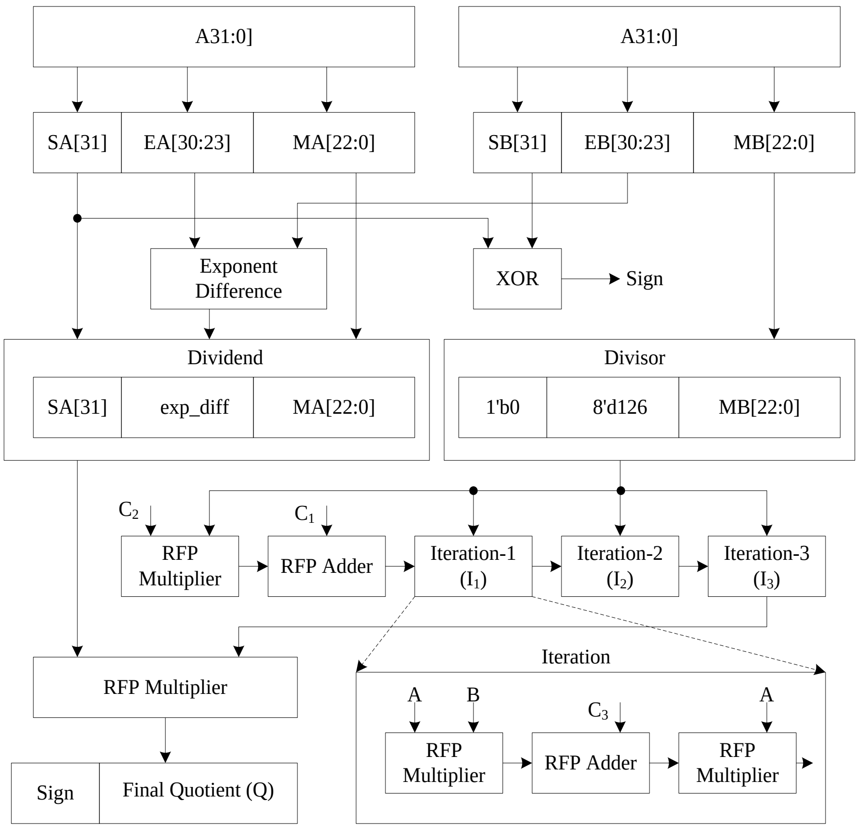


Figure 6. Representation of 32-bit RFP divider

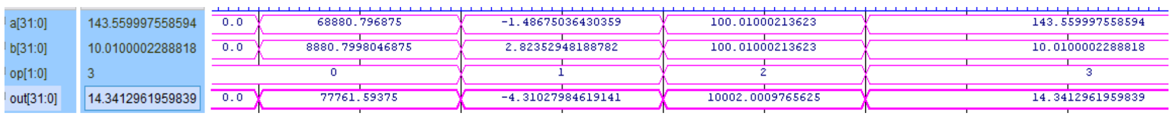
Table 1. Summary of RFPA unit's sub modules

RFP sub modules	GC	CI	GO	QC
32-bit multiplexer	32	0	64	128
8-bit comparator	74	23	36	118
comparator_31-bit	304	92	151	486
8-bit full adder/subtractor (FAS)	31	8	23	76
24-bit full adder/subtractor (FAS)	95	24	71	236
24-bit Barrel Shifter	120	0	240	480
24-bit 2's complementor	119	24	71	236
Normalization and shift unit	276	32	344	812
24×24 Wallace tree multiplier	1,616	744	1,272	3,758

4. RESULTS AND DISCUSSION

This section implements and analyzes the results and discussion of the RFP arithmetic unit and its sub-modules. The simulation results, synthesis results, performance metrics realization, and comparative analysis are discussed. The RFPA unit is synthesized and implemented on Artix-7 FPGA (XC7A100T-3CSG 324) using the Vivado IDE environment. The performance metrics like CI, GO, and QC parameters are realized for each sub-module of the RFPA unit. The Synthesis results of the RFPA unit include slice LUTs, total power consumption, and latency (Clock cycles), which are tabulated and discussed. The performance metrics and synthesis results are compared with similar existing approaches, with better improvements.

The simulation results of the RFPA unit are illustrated in Figure 7. The two 32-bit inputs (a and b), 2-bit selection modes (op) are defined, and 32-bit output (RFPA unit) is obtained as per design. The functional simulation results are verified with theoretical results. If mode (op) selection is zero, it performs the RFP addition, 1 for RFP subtraction, 2 for RFP multiplication, and 3 for RFP division.



```
a[31:0]    143.559997558594   0.0  68880.796875       -1.48675036430359   100.01000213623     143.559997558594
b[31:0]    10.0100002288818   0.0  8880.7998046875    2.82352948188782    100.01000213623     10.0100002288818
op[1:0]    3                       0                  1                   2                   3
out[31:0]  14.3412961959839   0.0  77761.59375        -4.31027984619141   10002.0009765625    14.3412961959839
```

Figure 7. Simulation results of RFPA unit

The performance analysis of the RFPA unit and its modules is tabulated in Table 2. The RFP adder utilizes 891 gate count (GC), CI of 193, GO of 973, and QC of 2,328. In contrast, the RFP Subtractor uses GC of 1,248, CI of 241, GO of 1,351, and QC of 3,272. The RFP multiplier utilizes GC of 1,881, CI of 861, GO of 1,596, and QC of 4,489. The RFP divider operates GC of 17,359, CI of 7,705, GO of 14,765, and QC of 45,464. Overall, the RFPA unit uses GC of 21,457, CI of 9,000, GO of 18,877, and QC of 55,937. The graphical representation of performance metrics of the RFPA unit is illustrated in Figure 8.

Table 2. Performance analysis of RFPA unit and its modules

RFP units	GC	CI	GO	QC
RFP adder	891	193	973	2328
RFP subtractor	1,248	241	1,351	3272
RFP multiplier	1,881	861	1,596	4,489
RFP divider	17,359	7,705	14,765	45,464
RFP arithmetic unit	21,475	9,000	18,877	55,937

The RFP subtractor unit has 2's completion operation, normalized and shift process, consuming more GC and QC. The RFP multiplier uses a 24×24 Wallace multiplier for mantissa calculation, and it consumes additional GC and QC. The reciprocal unit in the RFP divider module is designed using four RFP adders and four RFP multipliers that utilize more GC and QC.

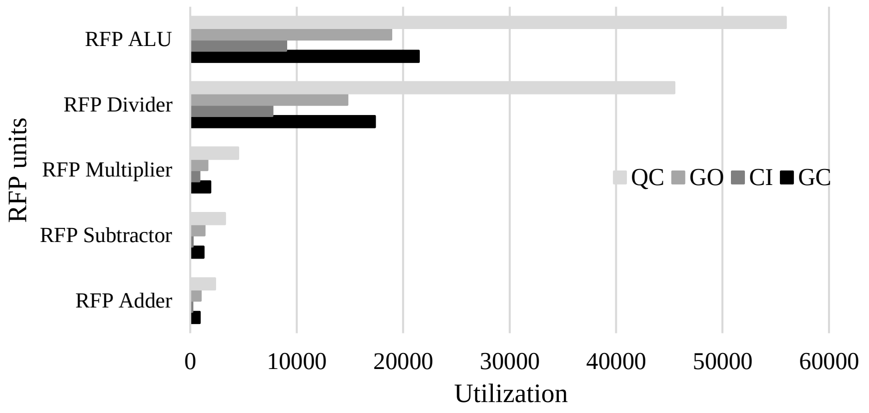


Figure 8. Graphical representation of performance metrics of RFPA unit

The resources utilization of the RFPA unit and its sub-modules on Artix-7 FPGA is tabulated in Table 3. The RFP adder/subtractor module utilizes 562 LUTs and consumes 0.14 W of total power. In contrast, the RFP multiplier module uses 1,085 LUTs and consumes 0.157 W of total power. The RFP divider module utilizes 9,803 LUTs and consumes 0.841 W of total power. The final RFPA unit utilizes 11,693 LUTs and consumes 0.891 W of total power after implementation on Artix-7 FPGA. The RFPA unit and its submodules modules are designed using reversible gates and are processed in parallel operations. The RFPA unit and its submodules are simulated individually and executes in a single clock cycle. Overall, the RFPA unit consumes 18.44% LUTs on Artix-7 FPGA.

The performance metrics comparison of RFP sub-modules with existing approaches is tabulated in Table 4. The performance parameters like GC, CI, GO, and QC compares existing approaches [11]–[16], [18]–[22]. The proposed RFP adder improves the CI by 92.7%, GO by 66.74%, and QC by 68.78% than the existing adder [11]. Similarly, the proposed RFP adder improves the CI by 74.5%, with a QC of 52.24% more than the existing adder [12]. The proposed RFP adder improves the CI by 65.03% and QC by 29.79% more than the existing adder [13]. The proposed RFP adder improves the GC by 3.04%, CI by 64.84%, and QC by 34.73% than the existing adder [14]. The proposed RFP adder improves the GC by 33.35%, CI by 75.19%, and QC by 60.02% more than the existing adder [15]. In contrast, the proposed RFP subtractor improves the QC by 11.78% more than the existing subtractor [16]. The proposed RFP multiplier improves the QC of 35.47%, 31.48%, 33.92%, 53.10%, and 24.05% than the exiting multipliers [18]–[22] respectively.

Table 3. Resources utilization of RFPA unit on Artix-7 FPGA

FP units	Slice LUTs	Total power (W)	Latency
RFP A/S	562	0.14	1
RFP multiplier	1,085	0.157	1
RFP divider	9,803	0.841	1
RFP arithmetic unit	11,693	0.891	1

Table 4. Performance metrics comparison of RFP sub modules with existing approaches

RFP A/S designs	GC	CI	GO	QC
RFP Adder				
Nactigal *et al.* [11]	NA	2,646	2,926	7,458
Nguyen and Meter [12]	NA	757	824	4,873
Alaghemand and Haghparast [13]	NA	552	635	3,316
AnanthaLakshmi and Sudha [14]	919	549	841	3,567
Nagamani *et al.* [15]	1,337	778	898	5,823
Proposed RFP adder	891	193	973	2,328
RFP subtractor				
AnanthaLakshmi and Sudha [16]	966	NA	888	3,709
Proposed RFP subtractor	1,248	241	1,351	3,272
RFP multiplier				
Nactigal *et al.* [18]	NA	NA	1,491	6,957
Jenath and Nagarajan [19]	NA	NA	1,723	6,552
Malathi *et al.* [20]	426	NA	1,296	6,794
Arunashalam *et al.* [21]	NA	608	1,879	9,573
Jain *et al.* [22]	NA	NA	1,156	5,911
Proposed RFP multiplier	1,881	861	1,596	4,489

The RFP adders [11]-[15] are used a conditional right shifter for the normalization process of RFP adders. In contrast, the proposed RFP adder does not use any conditional shifter during the normalization process, which reduces the 20% of the GC and QC in the design process. The RFP subtractor [16] uses TR Gates to construct a full adder/subtractor, which increases the QC rather than the proposed subtractor. The Wallace tree multiplier is used in most RFP multipliers [18]–[22] for partial product generation (PPG) and is constructed using compressor-tree units. The PPG using compressor-tree increases the hardware complexity and QC.

The resource comparison of RFP units with existing approaches to different FPGAs is tabulated in Table 5. The proposed RFP A/S module utilizes less than 80.81% of LUTs, consumes 65.93% of less power and 50% of latency than the existing RFP adder [16]. In contrast, the proposed RFP A/S module utilizes less than 57.04% of LUTs, consumes 78.09% of less power and 80% of latency than the existing RFP A/S [17]. The proposed RFP A/S module utilizes less than 19.48% of LUTs, consumes 65.85% of less power and 50% of latency than the existing RFP subtractor [16]. The proposed RFP divider module consumes 17.95% less power and 90% less latency than the current RFP divider [24]. The proposed RFP divider module consumes 15.64% less power and 85.71% less latency than the existing RFP divider [16].

Table 5. Resource comparison of RFP units with existing approaches on FPGA

RFP-AS	RFP Unit	FPGA	LUTs	Power (W)	Latency
AnanthaLakshmi and Sudha [16]	RFP Adder	Virtex-5	2,930	0.411	2
Kahn and Sundhari [17]	RFP A/S	Cyclone-10	596	0.639	5
AnanthaLakshmi and Sudha Ref [16]	RFP Subtractor	Virtex-5	698	0.41	2
Proposed RFP A/S	RFP A/S	Artix-7	562	0.14	1
Mu'noz *et al.* [24]	RFP Divider	Virtex-5	NA	1.025	10
AnanthaLakshmi and Sudha Ref [16]	RFP Divider	Virtex-5	NA	0.997	7
Proposed RFP division	RFP Divider	Artix-7	9,803	0.841	1

The existing RFP adders/subtractor [16], [17] are designed in sequential nature and consume more area and latency. The current RFP dividers [16], [24] are designed using Goldschmidt's algorithm and executed sequentially. These dividers consume more power and latency during reciprocal calculation than the proposed divider.

5. CONCLUSION

In this manuscript, an efficient reversible floating-point arithmetic (RFPA) Unit is designed and synthesized on Artix-7 FPGA using the Vivado IDE environment. The RFPA unit consists of RFP adder/Subtractor, RFP multiplier, and RFP divider modules. These modules are designed as per IEEE-754 standards. The corresponding module's output is the final RFPA unit output based on the selection mode. All the RFPA unit sub-modules are constructed using basic reversible gates. The simulation results of the RFPA unit are verified with theoretical values. The Synthesis results like Chip area and power consumption of the RFPA unit and its submodules are tabulated on the Artix-7 FPGA Platform. The Latency of the RFPA and its sub-modules executes in only one clock cycle. The RFPA unit utilizes 18.44% LUTs and a power of 0.891 W on Artix-7 FPGA. The performance metrics like GC, CI, GO, and QC is realized in detail for the RFPA unit and its submodules. The Proposed works are compared with existing similar works with better improvement in resource utilization (Chip area and power) and performance metrics.

APPENDIX

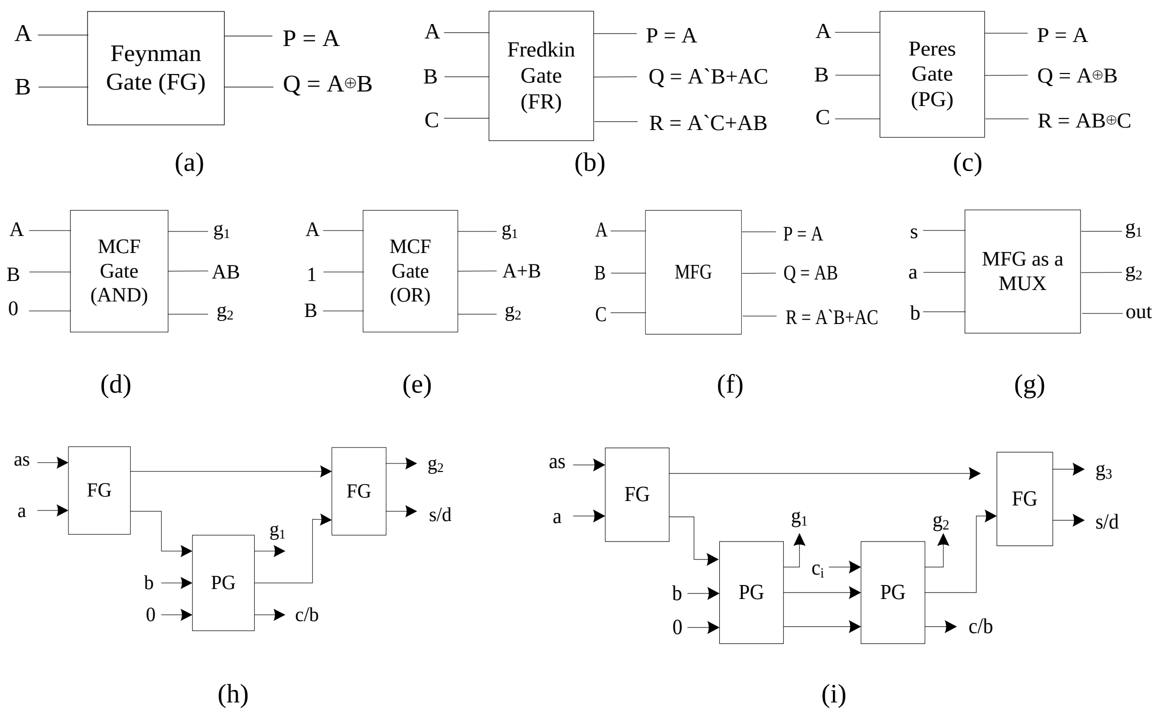


Figure 1. Reversible gates and modules used in RFPA unit design including (a) FG, (b) FR, (c) PG, (d) Reversible AND gate, (e) Reversible OR gate (f) MFG, (g) MUX gate, (h) RHAS and (i) RFAS

REFERENCES

[1] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3–4, pp. 219–253, Apr. 1982, doi: 10.1007/BF01857727.

[2] S. J. Honade, "Low power 32-bit floating point adder/subtractor design using 50nm CMOS VLSI technology," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 662–674, Aug. 2019, doi: 10.35940/ijitee.J8788.0881019.

[3] C. R. S. Hanuman, J. Kamala, and A. R. Aruna, "Implementation of multi-precision floating point divider for high speed signal processing applications," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 6038–6054, Sep. 2019, doi: 10.1007/s11227-019-02902-w.

[4] S. Kukati, D. V. Sujana, S. Udaykumar, P. Jayakrishnan, and R. Dhanabal, "Design and implementation of low power floating point arithmetic unit," in *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, Dec. 2013, pp. 205–208, doi: 10.1109/ICGCE.2013.6823429.

[5] S. Palekar and N. Narkhede, "High speed and area efficient single precision floating point arithmetic unit," in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, May 2016, pp. 1950–1954, doi: 10.1109/RTEICT.2016.7808177.

[6] J. Jain and R. Agrawal, "Design and development of efficient reversible floating point arithmetic unit," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, Apr. 2015, pp. 811–815, doi: 10.1109/CSNT.2015.215.

[7] P. Tabatabaii and M. Haghparast, "Novel design of nanometric reversible floating point adder with parity preservation capability," *International Journal of Innovative Computing and Applications*, vol. 7, no. 2, pp. 76–83, 2016, doi: 10.1504/IJICA.2016.077593.

[8] H. M. Gaur, A. K. Singh, and U. Ghanekar, "In-depth comparative analysis of reversible gates for designing logic circuits," *Procedia Computer Science*, vol. 125, pp. 810–817, 2018, doi: 10.1016/j.procs.2017.12.103.

[9] A. Dixit and V. Kapse, "Arithmetic & logic unit (ALU) design using reversible control unit," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 1, no. 6, pp. 55–60, 2012.

[10] B. Venkatachalapathi, "Implementation of high speed efficient reversible floating point arithmetic unit," in *NCTET-2K17*, 2017, pp. 76–81, doi: 10.22161/ijaers/nctet.2017.ece.23.

[11] M. Nachtigal, H. Thapliyal, and N. Ranganathan, "Design of a reversible floating-point adder architecture," in *2011 11th IEEE International Conference on Nanotechnology*, Aug. 2011, pp. 451–456, doi: 10.1109/NANO.2011.6144358.

[12] T. D. Nguyen and R. Van Meter, "A resource-efficient Design for a reversible floating point adder in quantum computing," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 11, no. 2, pp. 1–18, Nov. 2014, doi: 10.1145/2629525.

[13] F. Alaghemand and M. Haghparast, "Designing and improvement of a new reversible floating point adder," *Majid Haghparast*, vol. 4, no. 4, pp. 455–461, 2015.

[14] A. V. AnanthaLakshmi and G. F. Sudha, "Design of an efficient reversible single precision floating point adder," *International Journal of Computational Intelligence Studies*, vol. 4, no. 1, pp. 2–30, 2015, doi: 10.1504/IJCISTUDIES.2015.069830.

[15] A. N. Nagamani, C. K. Kavyashree, R. M. Saraswathy, C. H. V Kartika, and V. K. Agrawal, "Design of reversible floating point adder for DSP applications," in *Proceedings of the International Conference on Signal, Networks, Computing, and Systems*, 2016, pp. 123–135.

[16] A. V. AnanthaLakshmi and G. F. Sudha, "A novel power efficient 0.64-GFlops fused 32-bit reversible floating point arithmetic unit architecture for digital signal processing applications," *Microprocessors and Microsystems*, vol. 51, pp. 366–385, Jun. 2017, doi: 10.1016/j.micpro.2017.01.002.

[17] F. M. Khan and D. R. P. M. Sundhari, "A fault tolerant pipelined double precision reversible floating point adder/subtract or in FPGA," *Annals of the Romanian Society for Cell Biology*, vol. 25, no. 6, pp. 2947–2957, 2021.

[18] M. Nachtigal, H. Thapliyal, and N. Ranganathan, "Design of a reversible single precision floating point multiplier based on operand decomposition," in *10th IEEE International Conference on Nanotechnology*, Aug. 2010, pp. 233–237, doi: 10.1109/NANO.2010.5697746.

[19] M. Jenath and V. Nagarajan, "FPGA implementation on reversible floating point multiplier," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 1, pp. 438–443, 2012.

[20] S. R. Malathi, A. Venugopal, and P. Sarathy, "A cost effective design of reversible single precision floating point multiplier," *International Conference on Innovations in Engineering and Technology (ICIET)*, vol. 2, no. 1, pp. 26–31, 2013.

[21] K. Arunachalam, M. Perumalsamy, and M. N. Subraja, "Design and implementation of floating point multiplier in reversible logic with novel gates," *International Conference on Innovations in Engineering and Technology (ICIET)*, 2016.

[22] A. Jain, R. Jain, and J. Jain, "Design of reversible single precision and double precision floating point multipliers," in *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*, Dec. 2018, pp. 1–4, doi: 10.1109/ICACAT.2018.8933712.

[23] A. Kamaraj and P. Marichamy, "Design of fault-tolerant reversible floating point division," *Journal of Microelectronics, Electronic Components and Materials*, vol. 48, no. 3, pp. 161–171, 2018.

[24] D. M. Mu`ñoz, D. F. Sanchez, C. H. Llanos, and M. Ayala-Rincón, "Tradeoff of FPGA design of a floating-point library for arithmetic operators," *Journal of Integrated Circuits and Systems*, vol. 5, no. 1, pp. 42–52, Nov. 2010, doi: 10.29292/jics.v5i1.309.

[25] L. Jamal and H. M. H. Babu, "Efficient approaches to design a reversible floating point divider," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 3004–3007, doi: 10.1109/ISCAS.2013.6572511.

[26] S. S. Gayathri, R. Kumar, S. Dhanalakshmi, G. Dooly, and D. B. Duraibabu, "T-count optimized quantum circuit designs for single-precision floating-point division," *Electronics*, vol. 10, no. 6, Mar. 2021, doi: 10.3390/electronics10060703.

[27] A. Przybył, "Fixed-point arithmetic unit with a scaling mechanism for FPGA-based embedded systems," *Electronics*, vol. 10, no. 10, May 2021, doi: 10.3390/electronics10101164.

[28] L.-K. Wang and M. J. Schulte, "Decimal floating-point division using newton-raphson iteration," in *Proceedings. 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2004.*, 2004, pp. 84–95, doi: 10.1109/ASAP.2004.1342461.

[29] S. Thakral and D. Bansal, "High functionality reversible arithmetic logic unit," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, pp. 2329–2335, Jun. 2020, doi: 10.11591/ijece.v10i3.pp2329-2335.

[30] A. Eshack and S. Krishnakumar, "Reversible logic in pipelined low power vedic multiplier," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 16, no. 3, pp. 1265–1272, Dec. 2019, doi: 10.11591/ijeecs.v16.i3.pp1265-1272.

[31] K. N. Hemalatha, S. Girija, and B. G. Sangeetha, "Optimized 64-bit reversible BCD adder for low-power applications and its comparative study," in *International Conference on Computational Intelligence and Sustainable Technologies*, 2022, pp. 349–360.

[32] A. T. Hashim and S. A. Ali, "Reversible multiple image secret sharing using discrete haar wavelet transform," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 5004–5013, Dec. 2018, doi: 10.11591/ijece.v8i6.pp5004-5013.

[33] M. K. K. and R. S. Kunte, "Framework for reversible data hiding using cost-effective encoding system for video steganography," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 5487–5496, Oct. 2020, doi: 10.11591/ijece.v10i5.pp5487-5496.

BIOGRAPHIES OF AUTHORS

Girija Sanjeevaiah has completed her graduation in Electronics and Communication Engineering from Bangalore University and master's specialization in computer science from Visvesvaraya Technological University, Karnataka. She is currently working as Assistant Professor in the Electronics and Communication Engineering department of Dr. Ambedkar Institute of Technology. Her areas of interest are Reversible logic, embedded systems, and data computation. She can be contacted at email: girija.pari@gmail.com.

Sangeetha Bhandari Gajanan is an Assistant Professor in Electronics and Communication department at RNS Institute of Technology. She received Doctorate and M. Tech degree from Visvesvaraya Technological University and B.E from UBDTCE. Her areas of research interest are low power VLSI design and thin films. Her teaching and research experience of more than 15 years. She can be contacted at email: sangeethabg@gmail.com.