IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ERROR TOLERANT HIGH ACCURATE APPROXIMATE ADDER IN FPGA FOR REAL-TIME IoT PROCESSORS

Shanubha Mari Jero[1], Merfeena[2], Dyana Christilda[3]

[1]Student, Department of Electronics and Communication Engineering, Vins Christian College of Engineering, Tamil Nadu, India.

[2]Student, Department of Electronics and Communication Engineering, Vins Christian College of Engineering, Tamil Nadu, India.

[3]Professor, Department of Electronics and Communication Engineering, Vins Christian College of Engineering, Tamil Nadu, India.

Abstract: In reality, RISC processors and Digital signal processors (DSP) are used to construct the majority of embedded devices, and MAC (multiplications and/or accumulator) units are crucial here. During the multiplication and addition processes, the executions in MAC units frequently include a variety of crucial actions. Different adders and multipliers are required for constructing more effective MAC units, whereas approximation adders are typically developed based on their individual needs. In contrast to prior simple accurate adders, the innovative approximate adder proposed in this study delivers great speed and requires minimal complexity. The proposed adder's efficiency breakthrough paves the possibility for its incorporation into real-time IOT (Internet of Things) processors. This newly developed error-tolerant, highly accurate approximate adder was created to get over the implementation issues that parallel prefix adders have. In this situation, computing with fewer calculation nodes, maximum depth, and higher latency can be used to overcome the complexity of executions. The suggested approximation adder is synthesised and simulated in Xilinx Vivado22.2 using Verilog code.

Keywords: Image processing, MAC unit, and approximate adders.

I. INTRODUCTION

In digital circuits, computer arithmetic is essential for creating approximative algorithms that make good use of the hardware. Adders are implemented in ALUs in a variety of processor types to carry out fundamental operations. The output may not be as accurate as expected when a circuit is equipped with few components. In contrast, contemporary technical advancements anticipate not only moderate circuit utilisation but also low power consumption, an improvement in performance speed, and a small chip size. This anticipation demonstrates the close relationship between algorithms and demand in contemporary applications.

To add two binary numbers of any length together, adders can be concatenated. Half adders and complete adders are the two types of adders. While complete adders are employed in multiple bit addition, digital processors, etc., half-adder applications are found in calculators, computers, digital measuring instruments, etc.

The most efficient binary adders, such as the Ripple carry adder, Carry choose adder, Carry look ahead adder, Common Boolean logic adder, etc., are being used to update the basic approximation adders. In these applications, a delay in the generation of carry output causes the time it takes for carry to spread from one stage to the next, and the addition of each carry to the outcomes of the forthcoming stages. The number of bit additions rises together with the carry generation. This limitation is taken into consideration for the device's

efficient operation. Approximate adders are created to fulfil the Parallel Prefix Adder (PPA) accuracy in order to do away with these drawbacks. They are regarded as being the easiest to design and create adders. The Carry Tree adders are proven to perform better in VLSI designs and are effective in binary addition. This development in ETA design takes into account the Carry tree implementation of the Sklansky, Koggestone, Brent Kung, Ladner- Fischer, etc. adders.

The programme is run primarily to achieve high speed and low power consumption while obtaining the output with less accuracy. We employ error-tolerant adders because precision is not a top priority in this situation. The primary goal of this approach is efficient processing, even in intricate circuits. The high-speed MAC unit was created with IoT applications in mind as a result. Error Tolerant High Accurate Adder is used for comparing the performances of various approximation adders during designation.

II. LITERATURE SURVEY

The Accuracy-Configurable Radix-4 Adder with a Dynamic Output Modification Scheme was developed by K. Tsai, Y. Chang, and others. It makes use of the power gating technique to stoutly turn on or off the partial logic gates of an adder element to calculate accurate or inaccurate results. The ACRA's compute output demonstrates an excellent trade-off between power usage, propagation delay time, and error distance. But compared to the other Energy Quality Scalable Adder (EQSA), ACRA consumed more electricity.

M. E. Elbtity et al. [2020] suggested an accelerator design that makes use of approximation MAC units in the convolutional layer, parallel memory access, and N-way high speed. The chip space and power consumption are decreased by using the approximate adder with OR operations on LSBs (AOL). The approximation MAC units in CNN can be set up in several ways to attain more accuracy than the standard CNN architecture. Comparatively approximate multipliers have a larger effect than comparative approximate adders, and FC layers are more sensitive to approximate units than Convolutional layers.

In order to reduce the overall error distance caused by inexact addition, Sunghyun Kim et al. [2016] presented an adaptive approximate adder (A3) employing a modified XNOR-based adder and an adaptive idea to set the approximation bits during runtime. The authors of this study came to the conclusion that the bit lengths for the precise adder and the approximative adder can be determined using a Simple XOR operation. This technique also demonstrated that both the error rate and the error distance may be significantly improved. This claim's restriction is thought to be the low accuracy and complexity range that results from trying to add more bits..

For applications involving medical image processing, gradient filters, and Gaussian filters, P.L. Lahari et al. [2020] presented a truncated (MAC) multiplier and accumulator unit with reduced delay and reduced area consumption. Additionally, the algorithms for image and video processing applications can be realised with less time thanks to truncated based designs. Their proposal uses the proposed approximate adder to demonstrate an effective 16x16 truncated MAC unit delay realisation. When compared to earlier approximate adder designs, it produces the best area and least amount of delay..

Through generalising an architectural template for approximate adders, Ayad Dalloo, A. Najafi, et al. [2018] developed an ideal approximate adder. When compared to the previously mentioned best architectures, the suggested adder OLOCA exhibits a significant reduction in both hardware cost and error metrics. The mathematical analysis and additional experimental findings have demonstrated the superiority of OLOCA over the current approximate adders.

III. EXISTING METHOD

ARITHMETIC LOGIC UNIT

A digital circuit that can execute both arithmetic and logical operations is known as an arithmetic logic unit (ALU). It is renowned for showing the essential components of a computer's central processor unit (CPU). Along with the ALUs, modern CPUs also have a control unit (CU). The control unit instructs the ALU on the operations to be carried out on the data, and the ALU stores the outcome in an output register. Transferring data between these registers, the ALU, and the memory is the responsibility of the control unit.

The fundamental ALUs accept two data inputs and an opcode, which is a group of control signals. The ALU's function is represented by the opcode and the carry-in. C0 = 0 yields a+b when the ALU is set to add, while C0 = 1 produces a+b+1.

ADDERS

To gauge the speed of arithmetic processes, the addition is one of the simplest and most frequently utilised procedures. The addition of two binary numbers is the fundamental operation of microprocessors, digital signal processing (DSP), and other data-processing application-specific ICs (Integrated Circuits). Numerous algorithms have been proposed in order to support high-speed parallel addition as a result of the need for efficient operation in contemporary applications. Because of this, speed and area are typically traded off. The importance of the issue has been demonstrated by this, and binary adders are made to be the fundamental components of VLSI (Very Large-Scale Integration) circuits.

SINGLE BIT ADDITION

Adders are divided into two primary groups. Half-adder and full-adder, respectively. Less complicated half-adder addition is used in several fundamental circuits. Two input binary bits, known as input bits A and B, make up the adder. The sum (S) and carry (Cout) would be the obtained outputs when the two inputs are added to the equation. Figure (2), which is below Figure (1), displays the block schematic for both the half-adder and full-adder circuits.

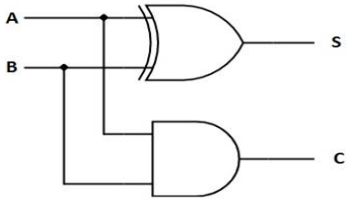

Fig.1. Half Adder Circuit

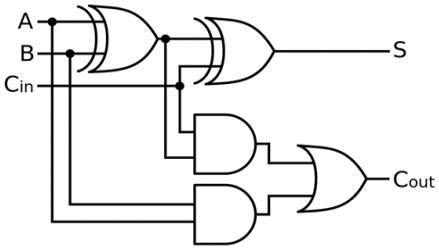

Fig. 2. Full Adder Circuit

MULTIPLE BIT ADDITION

When N-bit adders are taken into account, the two inputs are taken as "AN..., A1," "BN,..., B1," and "Cin," which is also treated as the propagation adder's third input." With these inputs, the output is calculated as the sum of the most significant bit's carry coefficient and the SN,..., S1, S1 bits. The text A0 often calls the least significant bit rather than absorbing the larger value. Cary-propagate adders (CPAs) are adders that operate in such a way that the carry into one bit affects the carry into every other succeeding bits. In this case, Cin has an impact on the total and carry bits. The carry-out from one bit is provided as the input (Cin) to the subsequent step of the connection in the most fundamental and basic architecture that uses the carry propagation adder. Faster adders look ahead to predict the carry-out of a multi-bit group by computing group PG (Propagate-Generate) signals in Fig. 3. This is typically done to show how the multi-bit group can propagate a carry-in or how to generate a carry-out. Long adders employ look-ahead structures with numerous levels to increase speed.

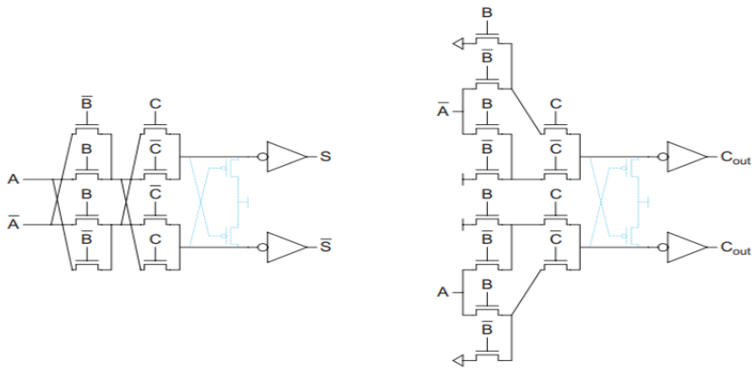

Fig.3. Complementary Pass transistor Logic (CPL) adder

- **CARRY RIPPLE ADDER**

An N-bit carry ripple adder (also known as ripple carry adder) is created by cascading a one-bit full adder N times. The adder's name implies that carry spreads throughout all of the adder's phases. The key delay path runs from the 0-bit inputs up through the Ci's to the n-1 bit because the addition is not complete until the nth adder computes its Sn-1 output; the outcome at that step depends upon Ci input, and so on down the line. The rippling time of the carry across the N stages determines the adder delay. Because addiction is a self-dual function, it is possible to eliminate the inverters on the outputs to reduce this delay. A full adder inversion receives complementary inputs and generates true outputs.

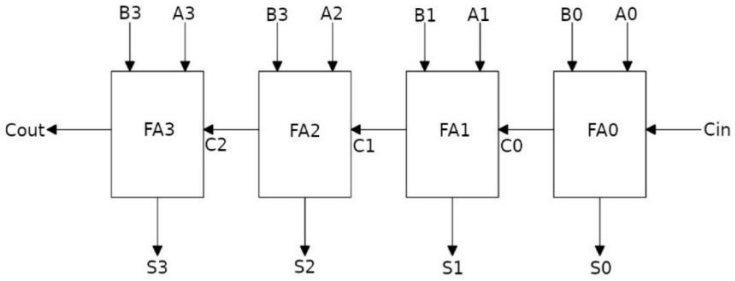

Fig.4. 4-bit ripple carry adder

- **CARRY LOOK-AHEAD ADDER**

The adders also accelerate as the carry chain does. Carry look-ahead adders may not be utilised much in VLSI, but they provide as an example of several essential concepts that can be applied to many adders. The carry computation in this adder is divided into two easy steps that begin with the computation of two intermediate values. The adder inputs are taken as Ai's and Bi's, just like the previous ones, and P (propagate) and G (generate) are computed from them.

Pi = Ai + Bi **- (1)**

Gi = Ai . Bi **- (2)**

When Gi = 1, the ith part of the sum will be carried out, and a carry will then be formed. The carry from the i-1th bit will propagate to the subsequent bit stage when Pi = 1. Both the sum and carry equations could be rewritten as follows in terms of P and G

Si = Ci ⊕ Pi⊕ Gi **- (3)**

Ci+1 = Gi + Pi Ci **- (4)**

The carry formula is simpler and thus easier to recursively expand when expressed in terms of P and G:

Ci + 1 = Gi + Pi . (Gi-1 + Pi-1 . Ci-1)

= Gi + Pi Gi-1 + Pi Pi-1. (Gi-2 + Pi¬-2 Ci-2)

=Gi + Pi Gi-1 + Pi Pi-1. Gi-2 + Pi Pi-1. Pi¬-2 Ci-2 **- (5)**

The restrictions highlight how slower larger gates are as compared to smaller gates. In general, four carry levels could be expanded.

Each unit generates its own P and G values, which are used to feed the carry-look ahead unit at the next level of the tree. The carry-look ahead units in fig. 5 can be connected recursively to build a tree.

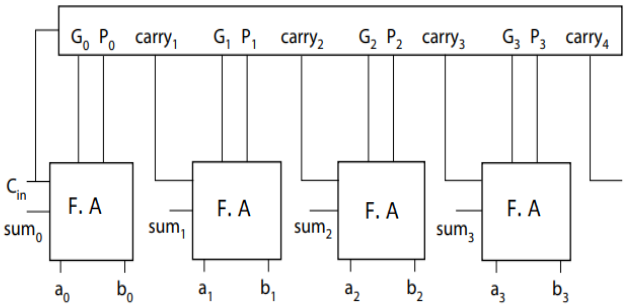

Fig.5. Carry look ahead adder structure

- **CARRY SELECT ADDER**

In the carry-select adder shown in Figure 6, the computation is carried out by comparing the addition of two alternative carry-in versions and then choosing the appropriate one. The carry-select adder is typically divided into m-bit stages, just like the carry-skip adder. In the second stage, computation is done using two numbers, one assuming carry-in is 0 and the other assuming it is 1. These candidate outcomes were all calculated using the corresponding adder structures' favourites.

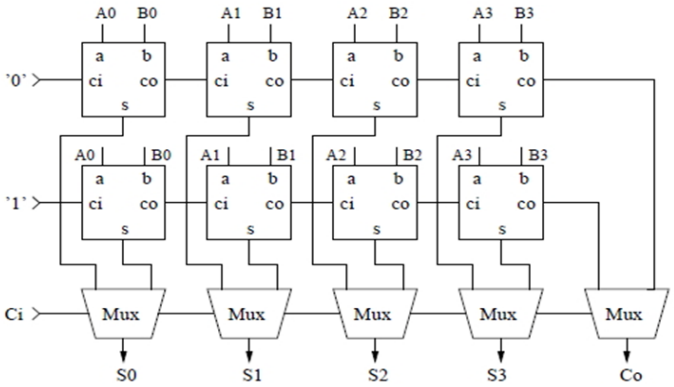

Fig.6. Carry select adder

TREE ADDERS

The delay of carry-lookahead (or carry-skip, carry-select) adders predominates for broad adders that perform sophisticated arithmetic operations (approximately N > 16 bits). This delay is caused by sending the carry through many stages of look-ahead adders. By scanning across the look-ahead blocks, this time can be reduced. In order to achieve the delayed growth with log N, a multilevel tree of look-ahead structures is built. These adders are also known as tree adders, logarithmic adders, parallel-prefix adders, multilevel-look-ahead adders, or just look-ahead adders. The look-ahead tree can be constructed in a variety of ways, offering trade-offs between the logic stages, the number of logic gates, the approximate fanout on each gate, and the amount of writing between stages. The primary fundamental trees are Sklansky, Kogge-Stone, and Brent-Kung architectures.

The log2 N stages and a fanout of 2 are achieved using the Kogge-Stone tree (Fig. 7). This may have the drawback of increasing the cost of rerouting numerous lengthy wires between phases. There are more PG cells at the tree; even if the surrounding region might not be affected, assuming the adder's layout is on a regular grid, this should result in higher power usage. The Kogge-Stone tree is employed in numerous high-performance 32-bit and 64-bit adders in spite of the expenses.

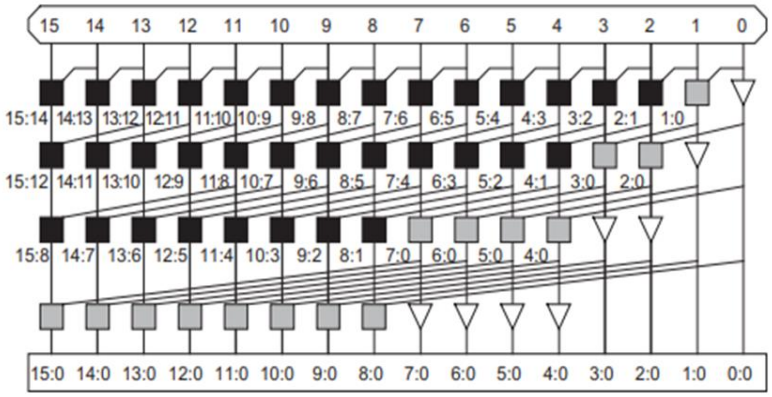

Fig.7.Kogge- Stone Tree

To cut the delay to log2 N stages, the Sklansky adder, also known as the divide-and-conquer tree (Fig. 8), computes intermediate prefixes in addition to large group prefixes. The use of this adder results in each level's fanouts being doubled: The gates spread out to other columns in [8, 4, 2, 1]. Unless the high fanout gates are scaled properly or the crucial signals are buffered before being used for intermediate prefixes, this leads to subpar performance on wide adders. Since each cell needs a variety of sizes of transistors, even though larger gates can stretch into adjacent columns, there may be some regularity in the layout's transistor scaling. It should be noted that the recursive doubling in this tree is comparable to the conditional sum adders. With the right amount of buffering, the fanouts might be cut down to [8, 1, 1, 1].

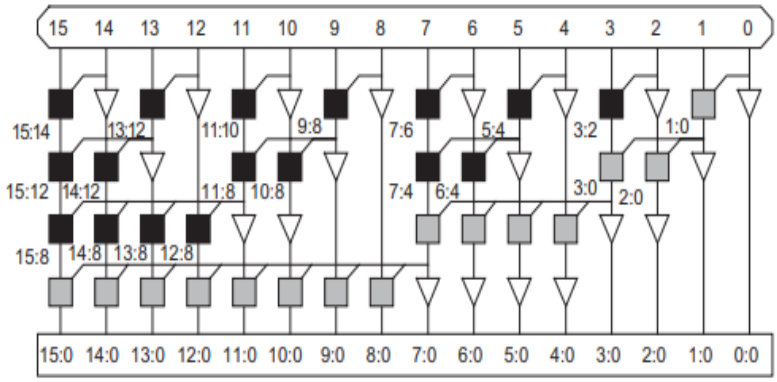

Fig.8. The Divide-and-Conquer Tree

The Brent-Kung tree (Fig. 9) can calculate the prefixes for 2-bit groups. They typically find prefixes for 4-bit groups, which find prefixes for 8-bit groups, and so forth. The respected carries-in to each bit will then be computed as the prefixes fan back down. The 2log2 N - 1 step is necessary in this tree. It is agreed that there will only be two fans at each stage. Although the diagram suggests using buffers to reduce fanout and gate loading, in reality, buffers are typically left out.

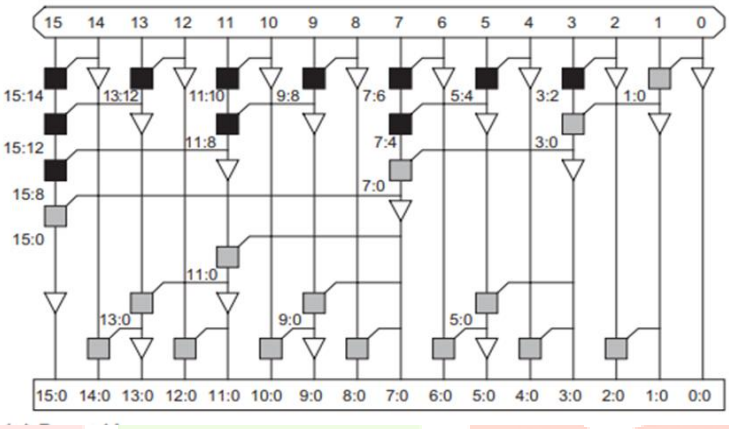

Fig.9. Brent-Kung Tree

Implementing a Sklansky or Kogge-Stone tree adder results in a shorter critical path when compared to the other two tree adders.

Ttree≈ tpg + [log2 N] tAO + tXOR **(6)**

The ideal tree adder will have wire tracks that are not more than one (Gi:j and Pi:j bundle) between each row and logic that is implemented at log2 N levels with a fanout that never exceeds two. The fundamental tree architecture represents examples that come close to the ideal but differ in one specific way.

PROPOSED METHODOLOGY

APPROXIMATE ADDERS

The major factors to be taken into account when building a digital circuit are the circuit's low power consumption, small design footprint, and high-performance speed. Approximate adders are primarily made to meet these requirements under circumstances where it is acceptable for errors to occur. In practically all arithmetic operations, including multiplication, subtraction, and division in digital circuits, these adders are regarded as the fundamental building blocks.

Numerous advancements in contemporary technology have the audacity to implement their designs using approximation adders, where certain faults are thought to be acceptable. In fields of implementation including machine learning, image processing, digital signal processing (DSP), Internet of Things (IoT), wireless communication, etc., approximate computations are highly sought-after and have gained popularity. These applications take advantage of the area, latency, and power advantages that come with mistake. The following approximation adder equations are included in the current approach and are hence suggested to ensure implementation efficiency**.**

APPROXIMATE ADDER 1

Both the sum and the carry are approximated. In conclusion, it is accurate in 6 out of 8 cases, while the carry is accurate in 7 out of 8 cases. The following logical equations carry the sum and carry approximation.

S=A' (B'Cin +BCin') **-(7)**

Cout=A + BCin **-(8)**

APPROXIMATE ADDER 2

When this adder type is taken into account, the approximation is carried out for the total alone and is accurate in 6 out of 8 situations. Since there is no approximation made here and the carry ensures that all outputs are error-free, it remains accurate in all situations. The following equations represent the sum (s) and carry (Cout).

S=AB'Cin' +ABCin **-(9)**

Cout= AB + BCin + ACin **-(10)**

APPROXIMATE ADDER 3

Both the sum and the carry are approximated in this instance. Six out of eight examples involve a precise amount, and seven out of eight involve a precise carry. These are the acknowledged sum and carry formulae.

S= A'B'Cin +AB'Cin' **-(11)**

Cout= (A+ B) Cin **-(12)**

APPROXIMATE ADDER 4

Here, both the sum and the carry are approximated. For 8 examples, the exactness of the total and carry is tested. Up to 5 out of 8 cases are accurate for the sum, while 6 out of 8 cases are accurate for carry.

S=A'(B + Cin) **- (13)**

Cout=B **- (14)**

APPROXIMATE ADDER 5

Here, both the sum and the carry are approximated. Eight cases are verified for accuracy in the case of the sum and carry. Up to 6 out of 8 cases are calculated for the preciseness of the total, and 6 out of 8 cases are calculated for the preciseness of the carry. The following formulae are used to determine the respected sum and carry.

S = ABCin + A'Cin' + B' **- (15)**

Cout= B **- (16)**

INPUT			OUTPUT											
			F. A		AA1		AA2		AA3		AA4		AA5	
A	B	C_{in}	S	Co	S	Co	S	Co	S	Co	S	Co	S	Co
0	0	0	0	0	0	0	0	0	0	0	0	0	1*	0
0	0	1	1	0	1	0	0*	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0*	0	0*	0	1	1*	1	1*
0	1	1	0	1	0	1	0	1	0	1	1*	1	0	1
1	0	0	1	0	0*	1*	1	0	1	0	0*	0	1	0
1	0	1	0	1	0	1	0	1	0	1	0	0*	1*	0*
1	1	0	0	1	0	1	0	1	0	0*	0	1	0	1
1	1	1	1	1	0*	1	1	1	0*	1	0*	1	1	1

Table. 1. Truth table for approximate adders (1 to 5)

The truth table comparison between the full adder and the approximate adders 1 to 5 is shown in Table 1. In the table above, the wrong output is represented by the Star symbol. In adders, you can either approximate both the sum and the carry, or you can approximate just the sum. Adders 1, 3, 4, and 5 are approximated by approximating both sums and carry, while adder 2 is approximated by approximating sum alone while leaving carry unchanged.

APPROXIMATE ADDER 6

Both the sum and the carry are approximated. 5 of the 8 situations have the correct sum, while 6 have the correct carry. Below are the formulae for solving sum and Cout.

S= A'B + Cin **(17)**

Cout= B **-(18)**

APPROXIMATE ADDER 7

5 out of 8 times when using this kind of approximation, the sum is accurate. In carry, 6 out of 8 instances involve the fault. Below are the logic equations for sum and carry.

S = (B'+ AB) Cin **-(19)**

Cout = BCin **(20)**

APPROXIMATE ADDER 8

Overall, 5 out of 8 cases have the correct outcome, and all carry cases have the correct output. So, each of the three examples has a mistake. As opposed to carry, where 7 out of 8 scenarios are right. The following list contains the equations for sum and carry.

S= B'(A+ Cin) **- (21)**

Cout= ACin + BCin **- (22)**

APPROXIMATE ADDER 9

Here, the total alone is the only thing that is approximated. Six out of eight times, the result is received perfectly for sum, and all are accurate for carry. 9 will be deemed the better when comparing approximate adders 8 and 9 because it produces fewer inaccuracy. Below are the formulae for resolving sum and carry.

S= (A'B' + AB) Cin **- (23)**

Cout= AB + BCin + ACin **- (24)**

APPROXIMATE ADDER 10

Only 5 of the total cases are accurate. based on these three situations of incorrect observation. For every case, the carry is unchanged. 8 fewer errors are returned when this adder's sum is checked with another adder. Below are the logical equations for resolving sum and carry.

S= A'B + Cin **(25)**

Cout = AB + BCin + ACin **(26)**

INPUT			**OUTPUT**											
			F. A		**AA6**		**AA7**		**AA8**		**AA9**		**AA10**	
A	**B**	**Cin**	**S**	**Co**	**S**	**Co**	**S**	**Co**	**S**	**Co**	**S**	**Co**	**S**	**Co**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	1*	0*	0	0*	0	0*	0	1	0
0	1	1	0	1	1*	1	0	1	0	1	0	1	1*	1
1	0	0	1	0	0*	0	0*	0	1	0	0*	0	0*	0
1	0	1	0	1	1*	0*	1*	0*	1*	1	0	1	1*	1
1	1	0	0	1	0	1	0	0*	0	0*	0	1	0	1
1	1	1	1	1	1	1	1	1	0*	1	1	1	1	1

Table 2. The truth table for approximate adders (6 to 10)

The contrast between the full adder and the approximations of adders 6 to 10 is shown in Table 2. In the approximation of adders, the star sign in the bits represents inaccurate output. Out of all the approximate adders, adders 2, 9 and 10 approximate 2 in sum while maintaining the carry. Adders 1, 3, 5, 6, 7 and 8 include

approximation in both sum and carry as compared to other adders. The approximate adder 2,9 and 10 uses sum-only approximation. The present adders' approximation was created to maintain output accuracy and minimise errors. The processors' performance is compared for efficiency using the suggested approximate adder 11, which is very accurate with minimal errors.

ETHAA – ERROR TOLERANT HIGH ACCURATE ADDER

The design of the approximation adder sacrifices accuracy. Both sum and carry are calculated while taking the outputs' approximation into account. There was one inaccuracy in the sum output and none in the carry output as a result of applying the derived equations. Adders 9 and 11 have a lower delay rate when compared to the existing adder formulae. Below are the logical equations for sum and carry calculations.

S = (A+B) Cin' + (AB+A'B') Cin **-(27)**

Cout = AB'Cin + B(A+Cin) **-(28)**

INPUT			**OUTPUT**			
			Full Adder		**ETHAA (Proposed)**	
A	**B**	**C_{in}**	**Sum**	**C_{out}**	**Sum**	**C_{out}**
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	1	0	0	1	1*	1
1	1	1	1	1	1	1

Table 3. The truth table of the proposed approximate adder

The approximation result of proposed adder 11, where accuracy is reached for 7 out of 8 cases in sum output, is shown in the truth table above. When compared to the full adder result, the proposed equation gives accuracy in carry in all 8 circumstances. Incorrect cases are indicated by the star symbol. The advantage of this adder over other approximate adders is its fast execution.

ERROR RATE CALCULATION

There are certain metrics for analysing the approximation of the adders that can be used to compare the approximate adders. The Error Rate (ER) for any multi-bit adder is the proportion of incorrect responses to all test instances. The following equation can be used to represent the error rate.

ER = No. of erroneous results / Total test cases **(29)**

The Error Distance (ED) is the difference between the accurate result (S) and the erroneous result (S*)

ED = |S – S*| **(30)**

Approximate adders	Error Rate		Error Distance	
	Sum O/P	Carry O/P	Sum O/P	Carry O/P
AA1	0.25	0.125	2	1
AA2	0.25	0	2	0
AA3	0.25	0.125	2	1
AA4	0.375	0.25	3	2
AA5	0.25	0.25	2	2
AA6	0.375	0.25	3	2
AA7	0.375	0.25	3	2
AA8	0.25	0.125	2	1
AA9	0.25	0	2	0
AA10	0.375	0	3	0
ETHAA (Proposed)	0.125	0	1	0

Table 4. ER and ED comparison of the Approximate adders.

The efficacy of the suggested Error Tolerant High Accurate adder is shown by the comparison of Error Rate and Error Distance above.

RESULT

For a variety of IoT-based real-time applications in the VLSI era, low-power, high-speed, and area-efficient circuits are preferred. Any signal processing for medical imaging applications must take into account the importance of arithmetic circuits, adders, and multipliers. Performance of the circuits as a whole is impacted by the adders' and multipliers' performance. In order to achieve high performance, efficient adder and multiplier circuits are required. This work proposes a speed, area, and power-efficient approximation adder design for the MAC unit that is appropriate for real-time IoT-based applications.

Three chosen adders and the suggested approximation adders are compared for efficient applications in terms of speed, area use, and power.

Ripple-carry adders and a multiplexer often make up the carry-select adder. Once the two results have been computed, the proper carry-in is known, and the multiplexer is then used to pick the correct carry-in, the correct sum, and the correct carry-out.

Due to the elimination of the pairs of ripple carry adders, the Carry Select adder has fewer logic gates (low area), but its power consumption is unchanged.

The Kogge-stone adder, which is regarded as the quickest adder that concentrates on design time and is a good substitute for high-performance applications, is the second adder. Due to limited fan-out and minimal logic depth, this nature is swift. Compared to the Kogge-Stone adder, which is significantly easier to construct, the Brent-Kung adder requires less modules for implementation. It is also simpler because there are less connections between the modules. Due to its lower complexity, this adder was chosen to be compared to the suggested adder. The fan-out of the Brent-Kung adder is one of its drawbacks.

Attributes	Carry Select Adder	Kogge-Stone Adder	Brent-Kung Adder	Proposed ETHAA
No. of LUT slices	25	51	24	15
Bonded IOB	50	50	49	50
Path delay in ns	13.721	12. 713	11.838	10.062
Power (w)	11.894	11.838	11.238	9.765

Table 5. comparison of four adders.

Table 5 compares the proposed approximate adder's performance to that of the current adders. The area used by the adders is determined by the number of slices LUTs that the FPGA has taken up. The delay is specified in terms of nanoseconds. The amount of power used during execution is calculated in Watt units (w). The only adder with fewer numbers than the other two, the Brent Kung adder, has the same bonded IOBs as the other two.

The outcome demonstrates that the proposed approximate adder is more effective than other conventional adders in terms of delay, area, and power. Comparing the slice LUTs utilisation to the existing adders, it is likewise quite low.

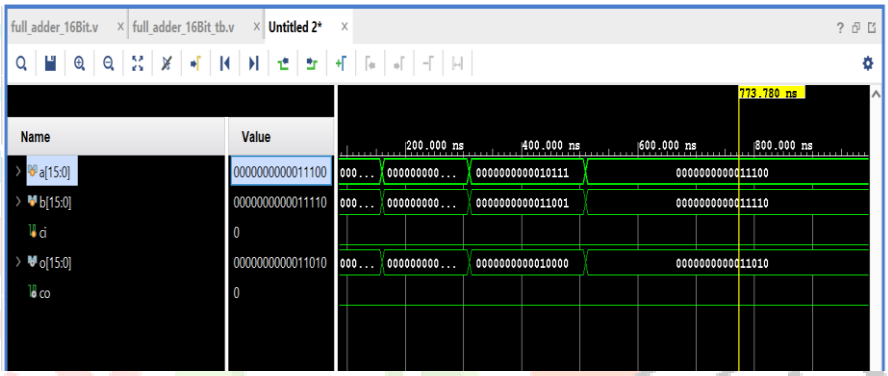

Fig.13. Simulation diagram.

The simulation output for the suggested approximate adder is shown in Figure 13. The suggested ETHAA's technological schematic is shown in the figure below.

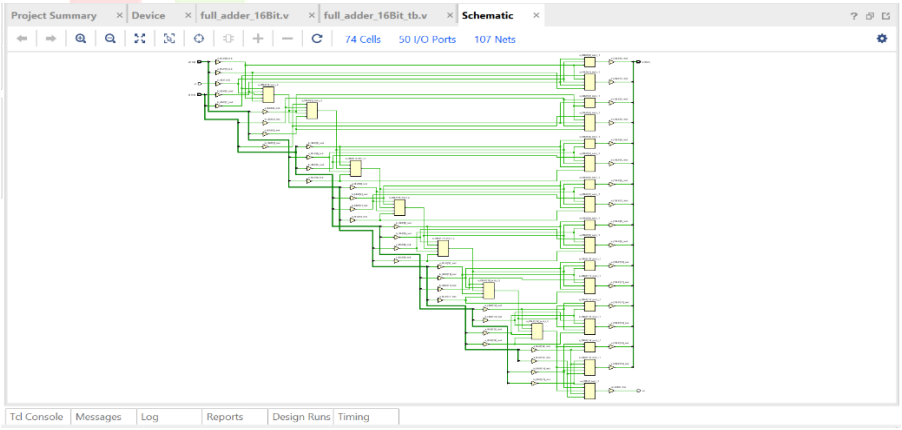

Fig.14. The schematic diagram of the proposed adder.

CONCLUSION

IoT technology is developing quickly, and this has increased the demand for CPUs with high speed, low power, and small form factors. In many real-time applications, they are crucial. The ALU unit and MAC units found in all digital processors are used to carry out some complicated and arithmetic computations. It needs an effective multiplier and an adder to function well. Numerous approximative adder equations are constructed to produce an equation with the least amount of error in order to satisfy the futuristic need. During the comparison, the equation with the highest degree of accuracy is suggested, carried out, and found to have the desired efficiency. By deriving both the Sum and Carry equations, the suggested adder achieves the highest accuracy possible. Utilising Verilog code, Xilinx Vivado is used to synthesise and simulate this adder equation.

REFERENCES

[1]. K. -L. Tsai, Y. -J. Chang, C. -H. Wang and C. -T. Chiang, "Accuracy-Configurable Radix-4 Adder with a Dynamic Output Modification Scheme," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 8, pp. 3328-3336, Aug. 2021.

[2]. Neil. West and David. Harris, "CMOS VLSI design, circuits and system perspective" Fourth Edition, Pearson Education India, 2015 pp.429-450.

[3]. M. E. Elbtity, H. -W. Son, D. -Y. Lee and H. Kim, "High Speed, Approximate Arithmetic Based Convolutional Neural Network Accelerator," 2020 International SoC Design Conference (ISOCC), Yeosu, Korea (South), 2020, pp. 71-72.

[4]. G. Anusha, P. Deepa," Design of approximate adders and multipliers for error-tolerant image processing, Microprocessors, and Microsystems, Volume 72,2020,102940, ISSN 0141-9331.

[5]. M. Bharathi and Y. J. M. Shirur, "Optimized Synthesis of Dadda Multiplier Using ParallelPrefix Adders," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2019, pp. 288-292.

[6]. T. Nomani, M. Mohsin, Z. Pervaiz, and M. Shafique, "xUAVs: Towards Efficient Approximate Computing for UAVs—Low Power Approximate Adders with Single LUT Delay for FPGA-Based Aerial Imaging Optimization," in IEEE Access, vol. 8, pp. 102982-102996, 2020.

[7]. A. Dalloo, A. Najafi and A. Garcia-Ortiz, "Systematic Design of an Approximate Adder: The Optimized Lower Part Constant-OR Adder," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 26, no. 8, pp. 1595-1599, Aug. 2018.

[8]. W. Liu, T. Zhang, E. McLarnon, M. O'Neill, P. Montuschi and F. Lombardi, "Design and Analysis of Majority Logic-Based Approximate Adders and Multipliers," in IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 3, pp. 1609-1624, 1 July-Sept. 2021.

[9]. Ansari, Mohammad Saeed & Jiang, Honglan& Cockburn, Bruce & Han, Jie. (2018). Low-Power Approximate Multipliers Using Encoded Partial Products and Approximate Compressors. IEEE Journal on Emerging and Selected Topics in Circuits and Systems. PP. 1-1.

[10]. J. Lee, H. Seo, H. Seok, and Y. Kim, "A Novel Approximate Adder Design Using Error Reduced Carry Prediction and Constant Truncation," in IEEE Access, vol. 9, pp. 119939-119953, 2021.

[11]. F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha and M. Pedram, "Block-Based Carry Speculative Approximate Adder for Energy-Efficient Applications," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 1, pp. 137-141, Jan. 2020

[12]. W. Ahmad, B. Ayrancioglu and I. Hamzaoglu, "Low Error Efficient Approximate Adders for FPGAs," in IEEE Access, vol. 9, pp. 117232-117243, 2021.

[13]. A. Sadeghi, R. Ghasemi, H. Ghasemian, and N. Shiri, "High Efficient GDI-CNTFET-Based Approximate Full Adder for Next Generation of Computer Architectures," in IEEE Embedded Systems Letters, vol. 15, no. 1, pp. 33-36, March 2023.

[14]. S. Kim and Y. Kim, "Adaptive approximate adder (A3) to reduce error distance for image processor," 2016 International SoC Design Conference (ISOCC), Jeju, Korea (South), 2016, pp. 295-296.

[15]. P. L. Lahari, M. Bharathi and Y. Jyothi Shirur, "An Efficient Truncated MAC using Approximate Adders for Image and Video Processing Applications," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 1039-1043.

[16]. R. Ishida, T. Sato, and T. Ukezono, "Approximate Adder Generation for Image Processing Using Convolutional Neural Network," 2018 International SoC Design Conference (ISOCC), Daegu, Korea (South), 2018, pp. 38-39.

[17]. V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, Switzerland, 2017, pp. 258-261.

[18]. Wyne Wolf, “Modern VLSI Design: IP based design”, Fourth Edition, Pearson Education, pp.352-360, Dec 2008.