Design and Analysis of an Approximate Adder with Hybrid Error Reduction

Ishan Jha
IMT2022562
Department of ECE
International Institute of Information Technology Bangalore
Bangalore- 560100, Karnataka, India
Email: Ishan.Jha@iiitb.ac.in

Shannon Muthanna I B
IMT2022552
Department of ECE
International Institute of Information Technology Bangalore
Bangalore- 560100, Karnataka, India
Email: shannon.muthanna@iiitb.ac.in

Sreyas Janamanchi
IMT2022554
Department of ECE
International Institute of Information Technology Bangalore
Bangalore- 560100, Karnataka, India
Email:sreyas.janamanchi@iiitb.ac.in

M V Chirag
IMT2022583
Department of ECE
International Institute of Information Technology Bangalore
Bangalore- 560100, Karnataka, India
Email: mv.chirag@iiitb.ac.in

R Harshavardhan
IMT2022515
Department of ECE
International Institute of Information Technology Bangalore
Bangalore- 560100, Karnataka, India
Email: r.harshavardhan@iiitb.ac.in

Aryan Mishra
IMT2022502
Department of ECE
International Institute of Information Technology Bangalore
Bangalore- 560100, Karnataka, India
Email: aryan.mishra@iiitb.ac.in

https://github.com/Sreyas-J/ApproxAdder

***Abstract*—Energy efficiency is a critical requirement in modern computing, particularly for system-on-chips (SoCs). Approximate computing offers significant energy savings by trading off computation accuracy, which is acceptable in many error-resilient applications like digital signal processing. Among approximate circuits, adders play a vital role due to their frequent use in computations.**

This report presents the implementation and analysis of an energy-efficient approximate adder with a hybrid error reduction scheme. Extensive comparisons highlight its superior performance, making it a competitive solution for error-tolerant applications.

The link of the Github repository containing all the reports of each Genus simulation and other details about the project are present in the following at the following URL: https://github.com/Sreyas-J/ApproxAdder.

Keywords: Adders

I. Introduction

Adder circuits are fundamental components in digital electronics, serving as the backbone of arithmetic operations in computers and various digital systems. Their ability to perform basic mathematical calculations, such as addition, is essential for the functioning of devices ranging from simple calculators to complex processors. The significance of adder circuits stems from their widespread application, efficiency, and role in enabling advanced computational capabilities

The importance of adder circuits extends beyond simple addition. They are vital in data processing, signal processing, and memory addressing. In central processing units (CPUs), adders are used not only for arithmetic computations but also for calculating memory addresses during data retrieval. They are integral in the design of digital filters and other systems requiring high-speed data manipulation, such as digital signal processors (DSPs).

At their most basic, adder circuits are designed to add two binary numbers. This operation is the foundation for more complex arithmetic functions like subtraction, multiplication, and division. Digital systems often rely on adders to execute these tasks efficiently. For example, subtractors are often implemented using adders with additional logic for handling two's complement arithmetic. This demonstrates how adders form the core of arithmetic logic units (ALUs), which are integral parts of microprocessors.

Adder circuits have evolved to meet the demands of modern technology. For instance, faster and more efficient adders, like carry-lookahead adders and parallel prefix adders, are critical in reducing the delay associated with binary addition. This has a direct impact on the overall performance of modern computing devices, as faster arithmetic operations enable quicker data processing and improved user experiences.

The aim of this paper is to do a comparative study of various kind of adder circuits, and compare their efficiencies. The efficiency of a particular adder circuit will be a combination of measures of various parameters such as energy consumption, total area occupied, total delay, accuracy of output of adder circuit, etc.

II. LOA : Lower-Part OR Adder

A LOA (Lower-Order Approximation) adder is a type of approximate adder circuit designed for applications where exact accuracy is not critical but speed, power efficiency, or area optimization is prioritized. LOA adders are commonly used in areas like image processing, neural networks, and signal processing, where minor errors are tolerable and can be compensated for by the system.

The LOA adder divides the addition process into two regions: a Lower-Order Region (Approximate Region) and a Higher-Order Region (Accurate Region).

In the lower order (or the approximate) region, the addition operation is approximated to simplify the logic. These simplifications may involve ignoring carry propagation or using XOR gates to compute approximate sums. However the main goal is to reduce circuit complexity and delay in this region. Meanwhile in the higher order (or accurate) region, the higher-order bits are added using exact full adders. The carry propagation is accurately handled in this region to maintain precision for significant bits.[3]

The circuit diagram of a LOA adder is given below in figure (II.1).

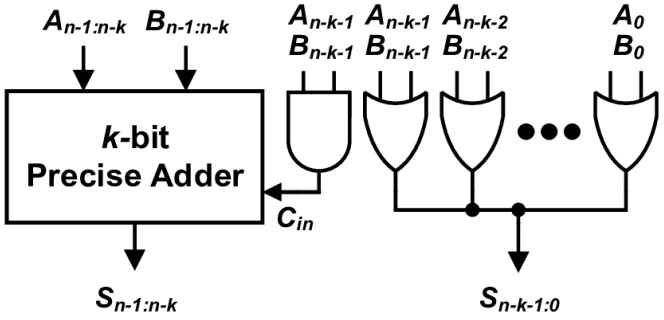

Fig. II.1: Circuit diagram of LOA Adder

Pros:

- High performance and speed improvement due to reduced carry propagation.
- It is energy efficient as it consumes lower power than traditional adders.
- Uses less hardware for the approximate region making it resource optimized.

Cons:

- There is a tradeoff between accuracy and execution speed, as due to approximation, errors are introduced in the final result.
- LOA adders have a limited application scope as they are unsuitable for applications requiring high precision.

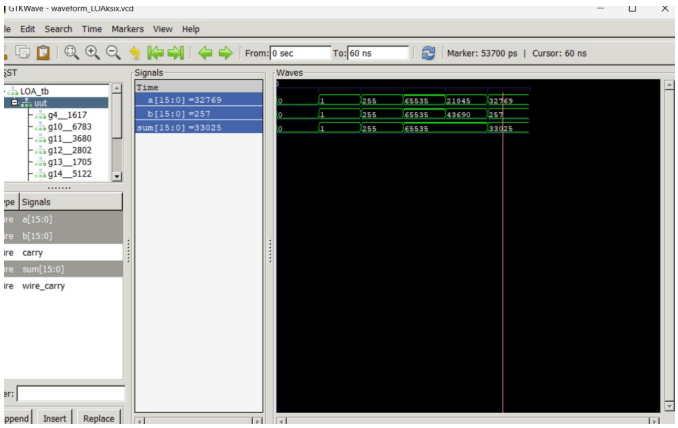

Fig. II.2: Genus Simulation of an LOA adder for k=6

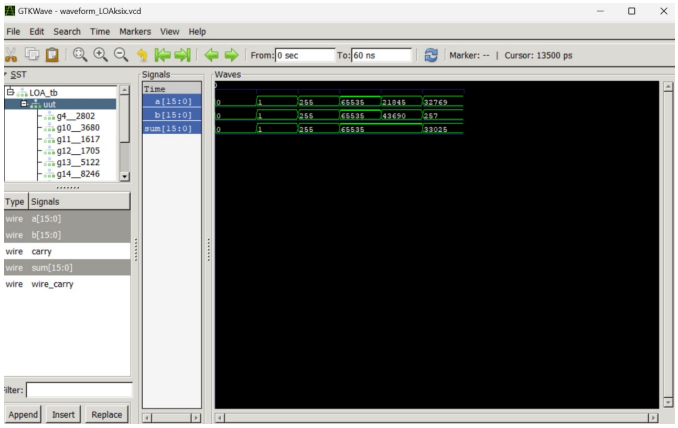

Fig. II.3: Genus Simulation of an LOA adder for k=7

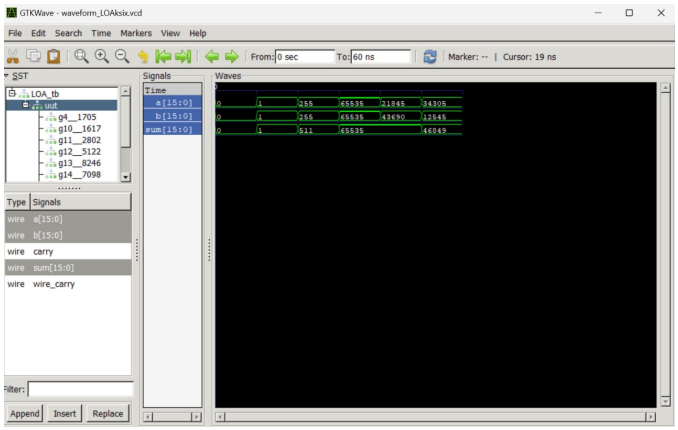

Fig. II.4: Genus Simulation of an LOA adder for k=8

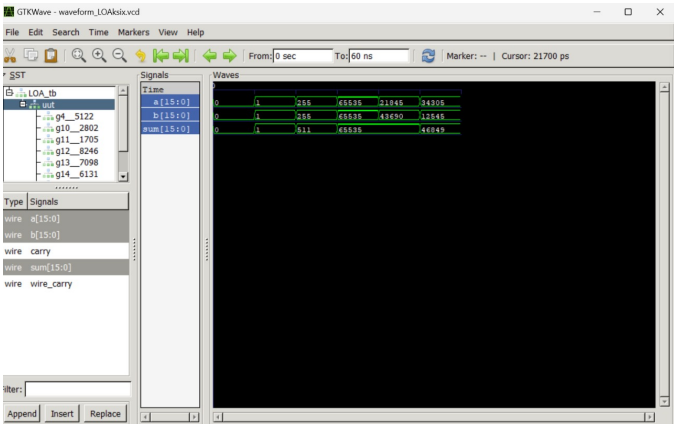

Fig. II.5: Genus Simulation of an LOA adder for k=9

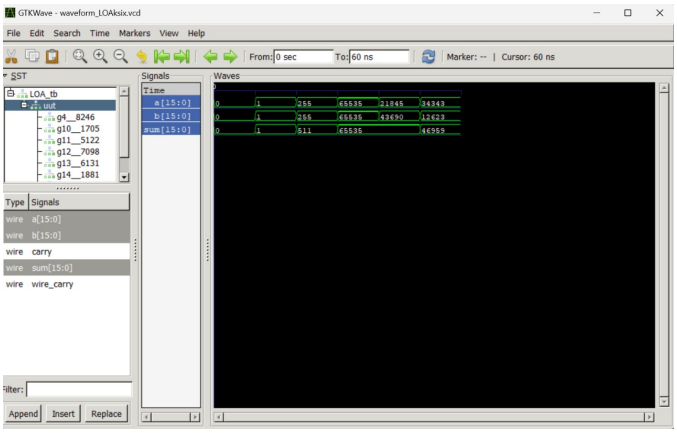

Fig. II.6: Genus Simulation of an LOA adder for k=10

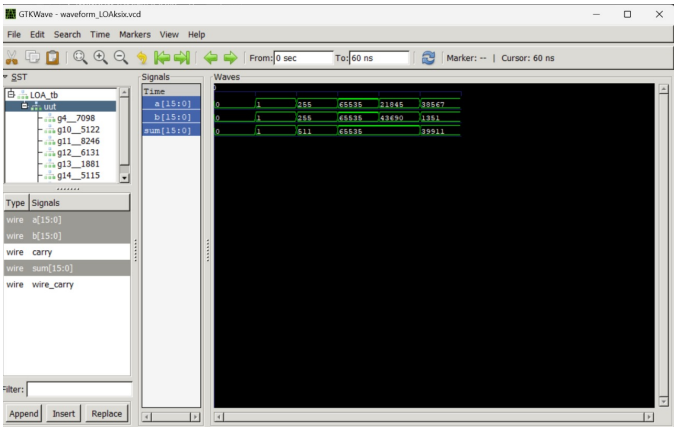

Fig. II.7: Genus Simulation of an LOA adder for k=11

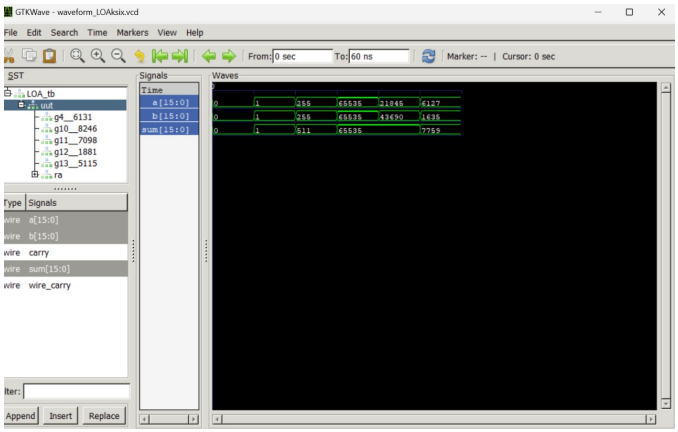

Fig. II.8: Genus Simulation of an LOA adder for k=12

III. LOAWA : LOA WITHOUT AND OPERATION

Similar to the LOA adder we have an adder introduced by Albicoocco et al. [1], known as the LOAWA (LOA without the AND operation), employs the OR operation to implement the (n-k)bit approximate addition for the LSBs. However, unlike the LOA, it omits the carry prediction scheme to minimize hardware costs, albeit at the cost of reduced accuracy.

The circuit diagram of a LOAWA adder is shown in figure (III.1).

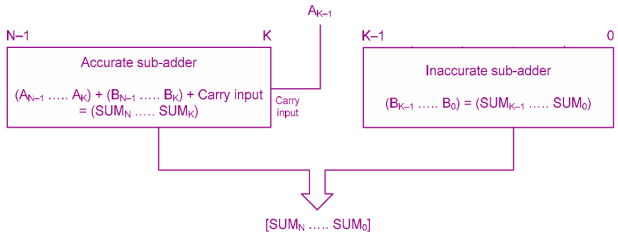

Fig. III.1: Circuit Diagram of an LOAWA Adder

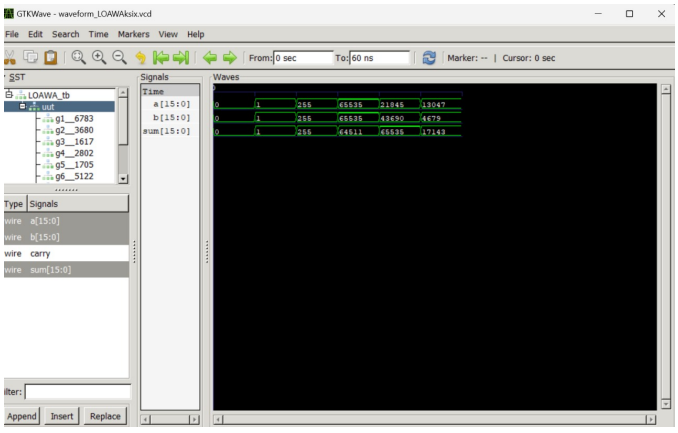

Fig. III.2: Genus Simulation of an LOAWA adder for k=6

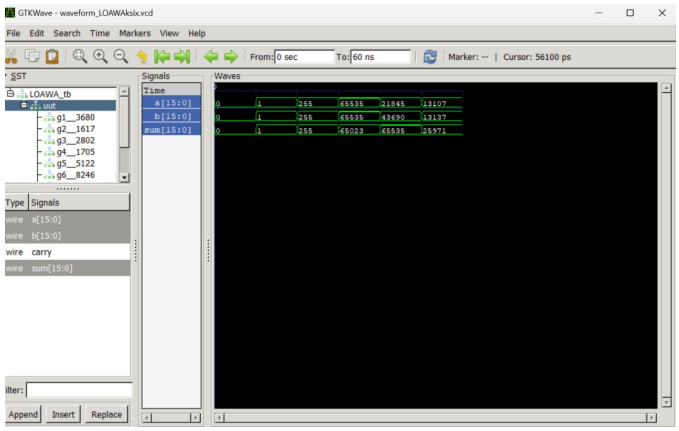

Fig. III.3: Genus Simulation of an LOAWA adder for k=7

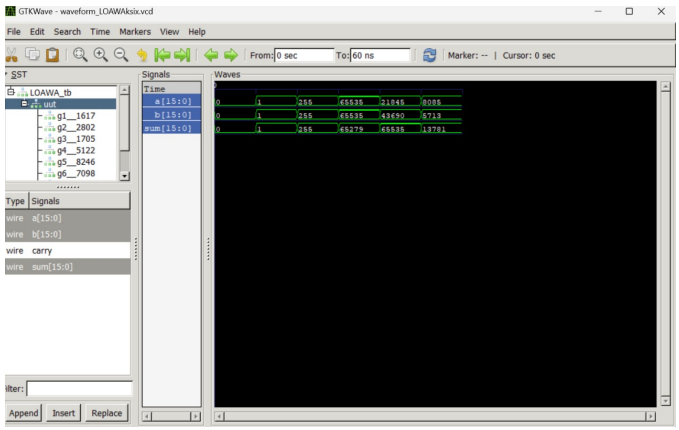

Fig. III.4: Genus Simulation of an LOAWA adder for k=8

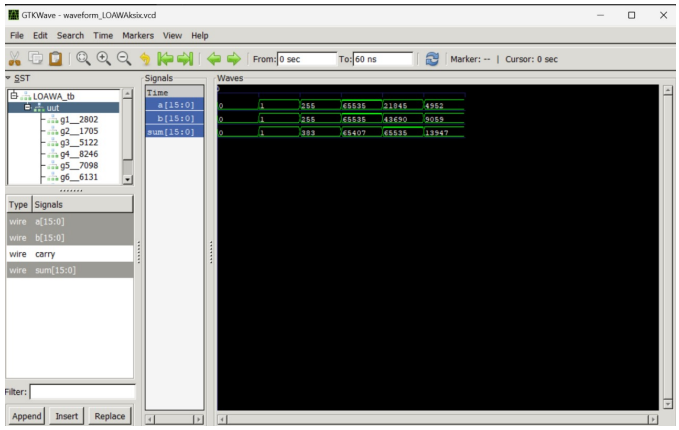

Fig. III.5: Genus Simulation of an LOAWA adder for k=9

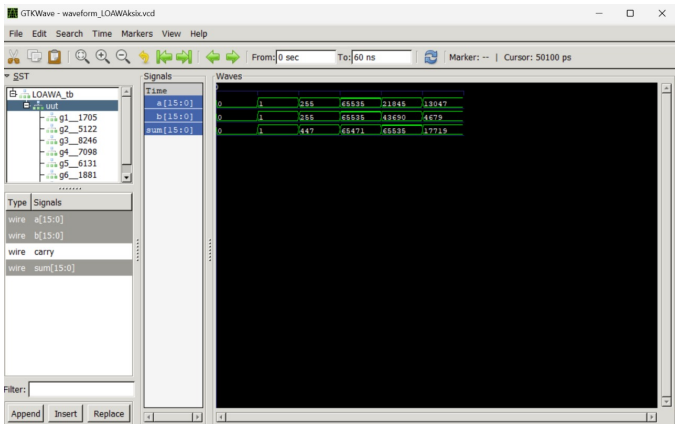

Fig. III.6: Genus Simulation of an LOAWA adder for k=10

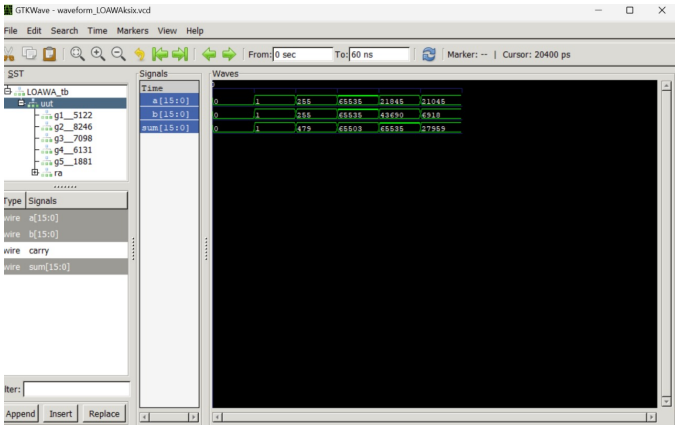

Fig. III.7: Genus Simulation of an LOAWA adder for k=11

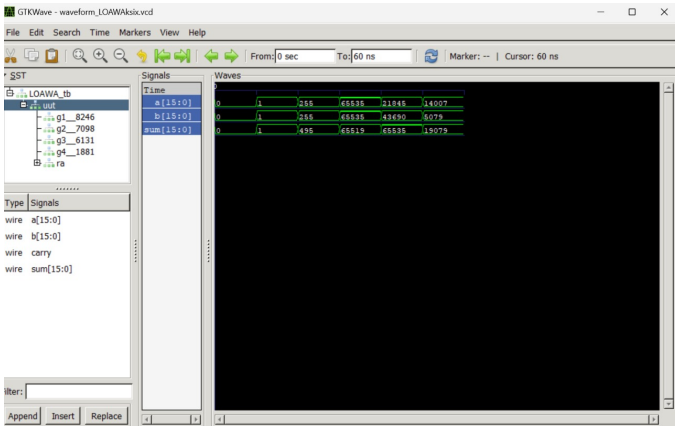

Fig. III.8: Genus Simulation of an LOAWA adder for k=12

IV. OLOCA : Optimized Lower-part Constant-OR Adder

An OLOCA (Optimized Lower-part Constant Approximate) adder is a type of approximate adder designed for applications where high performance, reduced power consumption, and area efficiency are more critical than precise accuracy.

The OLOCA adder splits the addition process into two distinct regions: the Lower-Part Approximate Region and the Higher-Part Accurate Region.

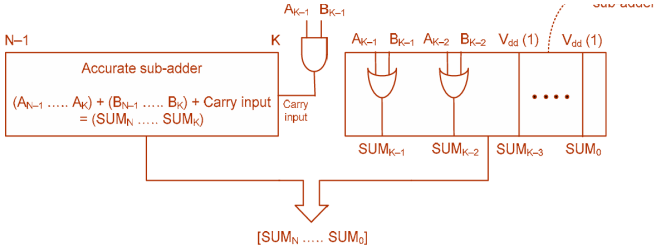

Fig. IV.1: Circuit Diagram of an OLOCA Adder

In the Lower-Part Approximate Region, instead of performing full addition with carry propagation, OLOCA simplifies the design by fixing the least significant bits (LSBs) of the sum to a constant value of 1. This reduces both power consumption and delay in the lower-order bits. In the Higher-Part Accurate Region, the OLOCA adder uses exact full adders to handle the significant bits with precise carry propagation, ensuring the accuracy of the most important part of the result.

By combining approximate computation in the lower part with exact addition in the higher part, the OLOCA adder achieves a balance between error tolerance and optimized performance, making it suitable for error-resilient applications.

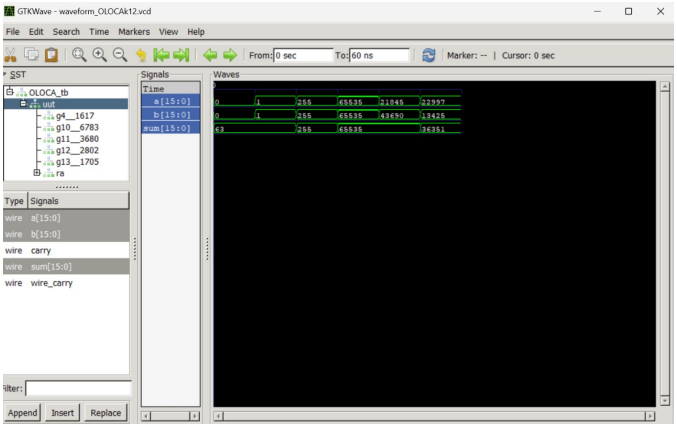

Fig. IV.2: Genus Simulation of an OLOCA adder for k=6

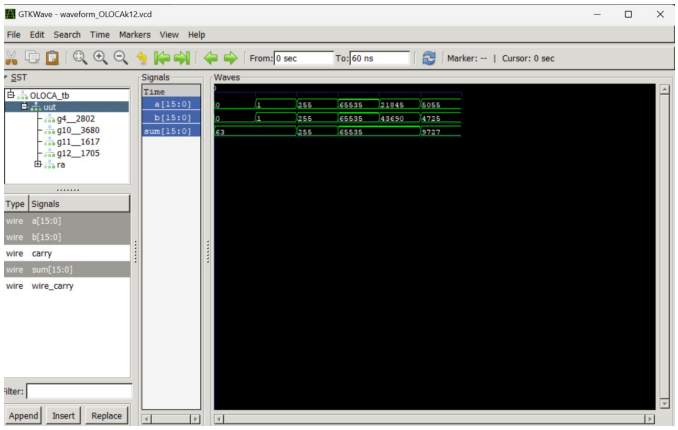

Fig. IV.3: Genus Simulation of an OLOCA adder for k=7

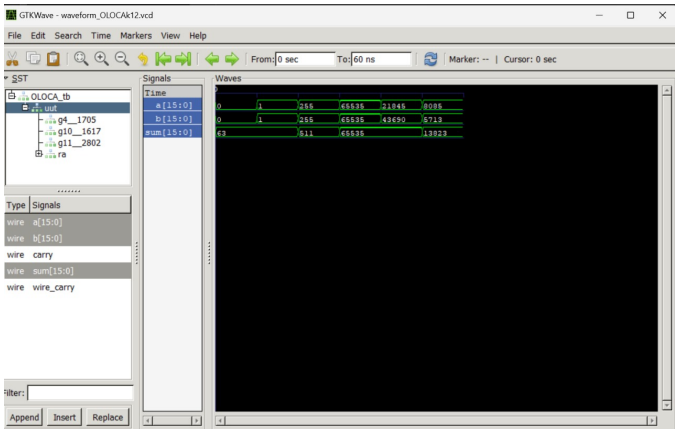

Fig. IV.4: Genus Simulation of an OLOCA adder for k=8

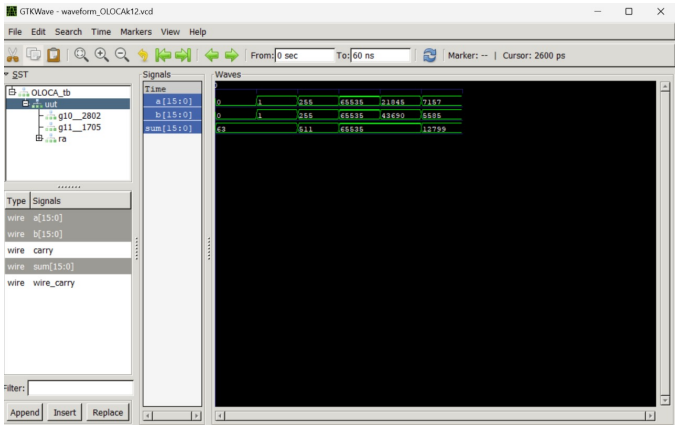

Fig. IV.5: Genus Simulation of an OLOCA adder for k=9

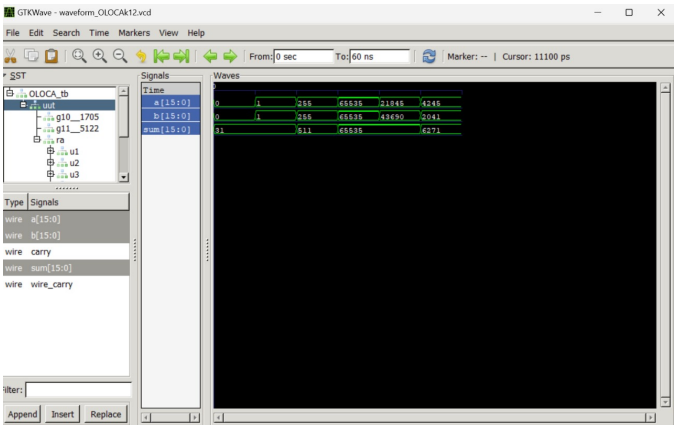

Fig. IV.6: Genus Simulation of an OLOCA adder for k=10

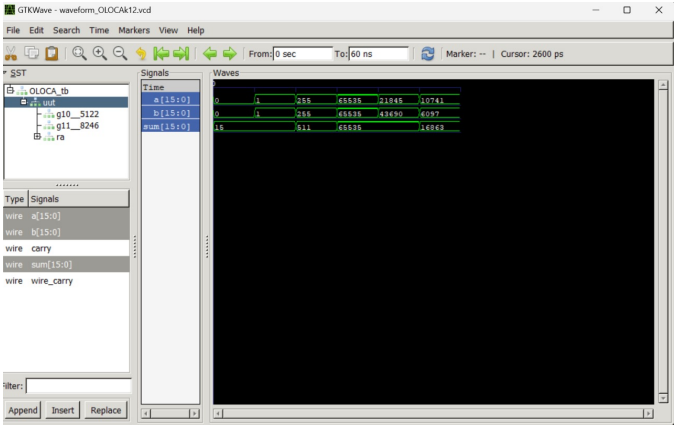

Fig. IV.7: Genus Simulation of an OLOCA adder for k=11

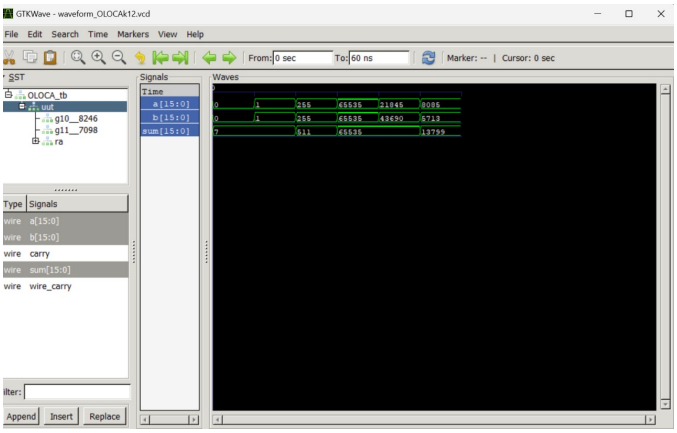

Fig. IV.8: Genus Simulation of an OLOCA adder for k=12

V. RCA : Ripple Carry Adder

The simplest way to build an N-bit carry propagate adder is to chain together N full adders. The C(out) of one stage acts as the C(in) of the next stage. This is called a ripple-carry adder.[2]

A Ripple Carry Adder (RCA) is a basic digital circuit used to perform binary addition. It consists of a series of full adders

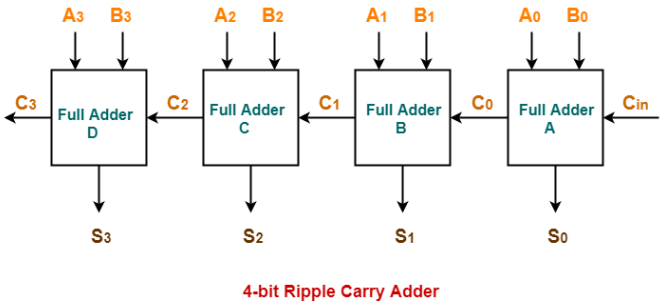

Fig. V.1: Circuit diagram of RCA Adder

connected in cascade, where the carry output of one stage is fed as the carry input to the next stage. This sequential propagation of the carry gives the adder its name, as the carry ”ripples” through the circuit from the least significant bit (LSB) to the most significant bit (MSB).

An RCA is composed of multiple full adder circuits, each responsible for adding a single pair of bits along with an input carry. For an n-bit addition, the RCA consists of n-full adders. Each full adder adds three inputs: two binary digits (Ai and Bi) and a carry input (Cin) from the previous stage and generated two outputs: the binary sum of the inputs and the carry generated from the addition which will be propagated to the next stage. The carry generated at each stage must propagate through all subsequent stages before the final sum is available. This sequential carry propagation results in a delay proportional to the number of bits, making the RCA slower for large bit-widths.

Pros:

- The design and implementation of a RCA adder are straightforward, making it easy to construct and understand.
- RCA adders require fewer gates compared to more advanced adders, making it cost-effective for small bit-width additions.

Cons:

- The propagation delay increases linearly with the number of bits due to sequential carry propagation, limiting its speed for larger bit-widths.
- The cumulative delay makes it inefficient for applications requiring fast arithmetic operations.

The circuit diagram of a RCA adder is given in figure (V.1).

VI. HOERAA : High-Order Error-Reducing Approximate Adder

The HOERAA (High-Order Error-Reducing Approximate Adder) is a type of approximate adder designed to achieve a balance between computational efficiency and accuracy in digital systems. It is commonly used in error-tolerant applications, such as image and video processing, neural networks, and machine learning, where some level of error is acceptable.

The HOERAA adder modifies the addition process to reduce hardware complexity, power consumption, and delay while maintaining a relatively high level of accuracy for critical

bits. It operates by applying different approximations to the lower-order and higher-order bits. By focusing on accurate computation of higher-order bits, HOERAA ensures minimal deviation in the final result, even with approximations in lower-order bits and is designed for applications where minor errors in output are acceptable, such as multimedia processing.

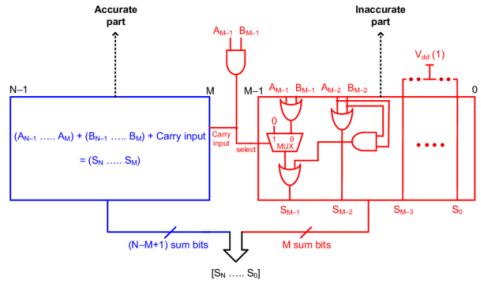

Fig. VI.1: Circuit diagram of a HOERAA Adder

Pros:

- The use of approximate logic for lower-order bits significantly reduces energy usage.
- Carry propagation delays are minimized, leading to faster addition operations.
- The design of a HOERAA adder can be modified for the approximation level to balance between speed and precision based on application needs.

Cons:

- While effective in reducing errors, it may not be suitable for applications requiring exact computation.
- The error-reducing mechanism can increase design complexity in comparison to simpler accurate adders such as RCA.

The circuit diagram for a HOERAA adder is shown in figure (VI.1).

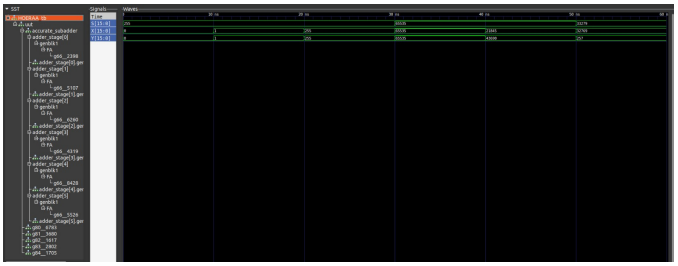

Fig. VI.2: Genus Simulation of an HOERAA adder for k=6

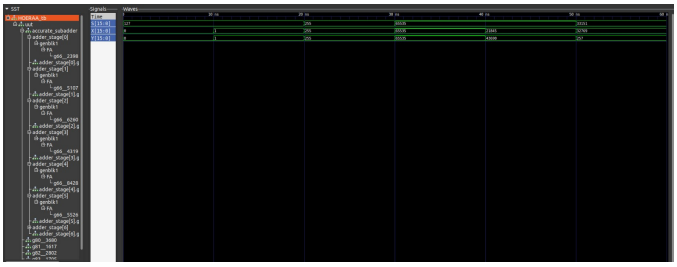

Fig. VI.3: Genus Simulation of an HOERAA adder for k=7

[h!]

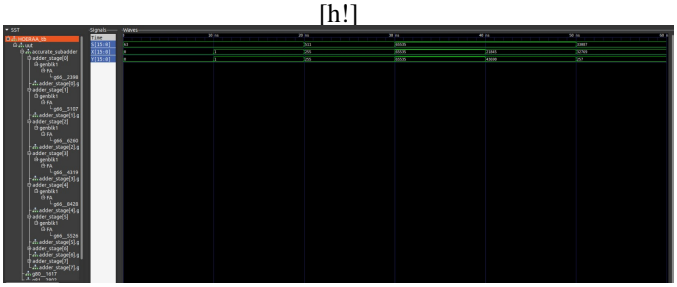

Fig. VI.4: Genus Simulation of an HOERAA adder for k=8

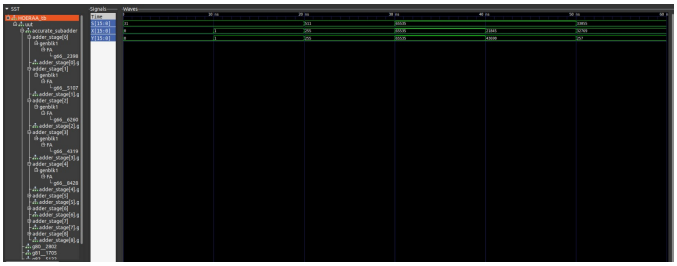

Fig. VI.5: Genus Simulation of an HOERAA adder for k=9

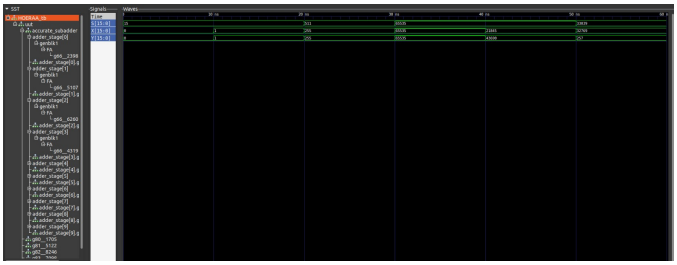

Fig. VI.6: Genus Simulation of an HOERAA adder for k=10

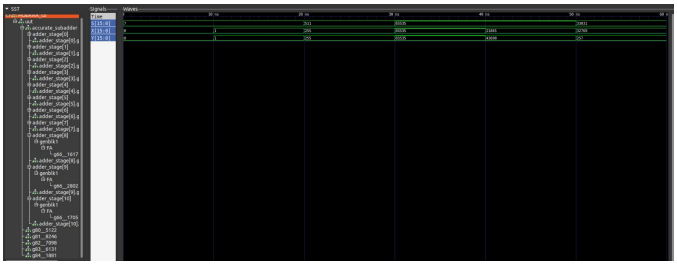

Fig. VI.7: Genus Simulation of an HOERAA adder for k=11

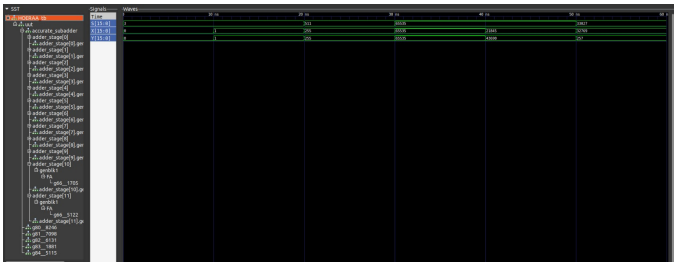

Fig. VI.8: Genus Simulation of an HOERAA adder for k=12

VII. CLA : Carry Look Ahead Adder

A carry look-ahead adder (CLA) reduces the propagation delay by introducing more complex hardware. The CLA modifies the RCA structure by predicting the carry for the next adder using the carry of the current adder. The variables "Carry Generate" and "Carry Propagate" have been introduced to help reduce delay. Carry Generate is a flag indicating that a carry will be generated regardless of the input bits. "Carry Propagate" refers to transferring the carry from the current summation to the next one. The summation is calculated using the XOR operation of "Carry Propagate" and the current carry. It improves performance by reducing the delay caused by carry propagation in conventional adders like the Ripple Carry Adder.

Pros:

- CLA minimizes the carry propagation delay by calculating all carry bits simultaneously making it significantly faster than ripple carry adders, especially for larger bit-widths.
- CLAs are suitable for higher-bit addition since their delay does not grow linearly with the number of bits, unlike ripple carry adders.
- Due to its speed, the CLA is widely used in processors, digital signal processors (DSPs), and other high-performance systems where quick arithmetic operations are critical.

Cons:

- The logic for generate and propagate signals, as well as the carry lookahead logic, requires additional gates and interconnections, increasing the complexity of circuits for large bit numbers, thus making it hard to design and implement.
- The increased number of logic gates results in higher power consumption compared to simpler adders like ripple carry adders. These increased number of gates and interconnections result in a larger silicon area, which could be a limitation in a resource-constrained system.

The circuit diagram for a carry look ahead adder is given in figure (VII.1).

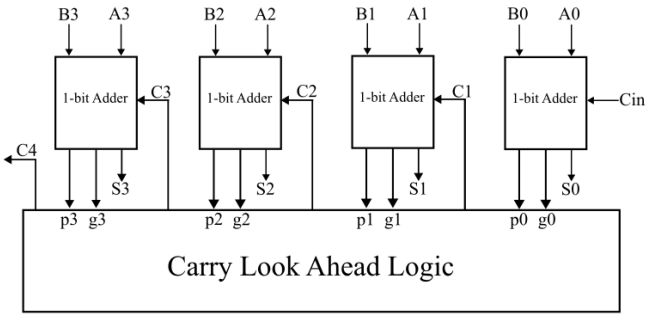

Fig. VII.1: Circuit diagram of a CLA Adder

VIII. ETA-I : Error Tolerant Adder Type-I

In many digital systems, precise arithmetic operations are often unnecessary, especially in error-tolerant applications like digital signal processing (DSP), image processing, and approximate computing. Error Tolerant Adder I (ETAI) is designed to provide an efficient trade-off between computational accuracy, delay, and power consumption by selectively tolerating errors in low-significance bits. This adder is particularly beneficial in applications where small errors do not significantly impact system performance.

Error Tolerant Adder I (ETAI) adopts a hybrid addition mechanism by splitting the input operands into two distinct sections:[4]

Accurate Part (First k Bits): The higher-order bits, critical for maintaining the overall accuracy of the computation, are processed using a precise addition method. Inaccurate Part (Next n-k Bits): The lower-order bits, which have less significance in most applications, are handled with an approximate addition technique to reduce delay and power consumption.

The division into accurate and inaccurate parts allows ETAI to exploit the trade-off between computational accuracy and system performance by focusing precision where it matters most, while tolerating errors in less significant bits.

The accurate part consists of the first k higher-order bits, which are summed using a conventional k-bit precise adder. This block ensures that no approximation occurs in these bits, preserving the correctness of the most significant portion of the result. The addition in this segment follows the standard rules of binary addition, with full carry propagation, ensuring that higher-order bit results remain consistent with traditional addition methods.[4]

The inaccurate part, which processes the next (n-k) bits, consists of two main sub-blocks: the control block and the carry-free addition block. Together, these components implement the approximate addition mechanism, effectively reducing the delay and power consumption associated with traditional carry propagation.

The control block monitors the addition process in the approximate part by generating a control signal. The control signal is designed to determine whether the addition process

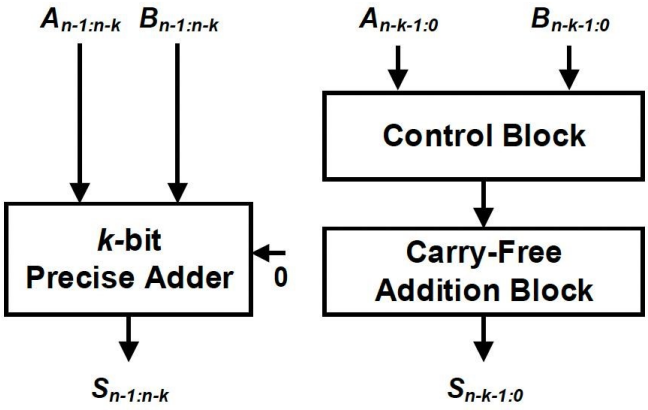

Fig. VIII.1: Circuit diagram of an ETA-I Adder

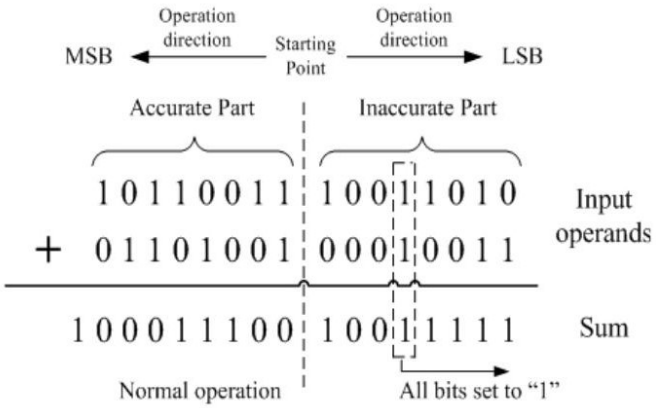

Fig. VIII.2: Working of an ETA-I Adder

should continue as normal or terminate early. At every step, it checks the values of the corresponding input bits. If the input bits at a given position are (1,1), the control signal is set high, indicating that further bitwise addition will be bypassed, and all subsequent sum bits will be set to 1. In all other cases the control signal remains low, allowing normal addition to proceed. The control block ensures that the approximate addition process terminates efficiently whenever a critical carry condition (1+1) is encountered, thereby avoiding unnecessary delay.[4]

The carry-free addition block performs a modified XOR-based operation to compute the sum of the lower-order bits without carry propagation.

Pros:

- By eliminating carry propagation in the lower-order bits, ETAI achieves faster computation compared to traditional adders.
- This simplified addition mechanism significantly reduces power consumption, making ETAI suitable for low-power applications.
- Similar to other adders described previously, the configurable division between the accurate and inaccurate parts allows flexibility in balancing accuracy and performance.

Cons:

- Errors introduced in the approximate part can propagate to the final result, which may not be acceptable in high-precision applications.
- For small-number inputs, the relative magnitude of errors can become significant, leading to poor performance.
- ETAI is unsuitable for applications requiring high levels of precision, such as control systems or financial computations.

The circuit diagram of an ETA-I adder and an example of how it works is shown in figure (VIII.1) and figure (VIII.2) respectively.

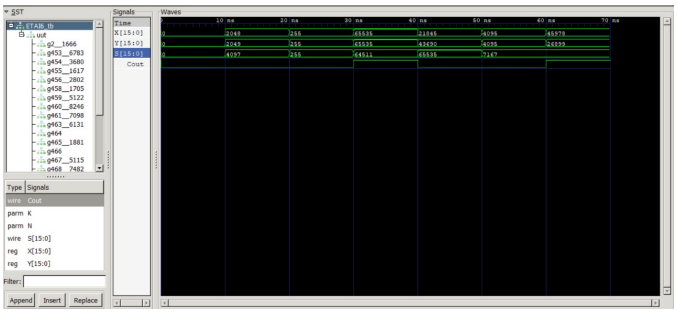

Fig. VIII.3: Genus Simulation of an ETA-I adder for k=6

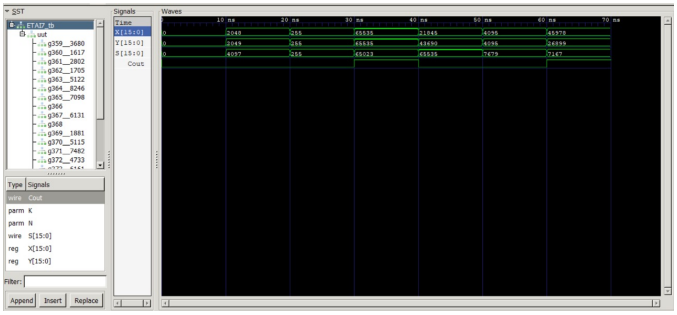

Fig. VIII.4: Genus Simulation of an ETA-I adder for k=7

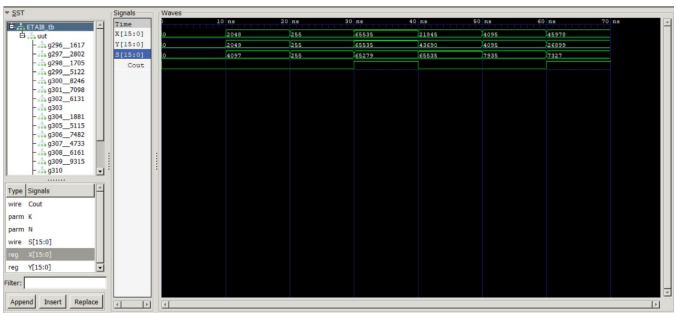

Fig. VIII.5: Genus Simulation of an ETA-I adder for k=8

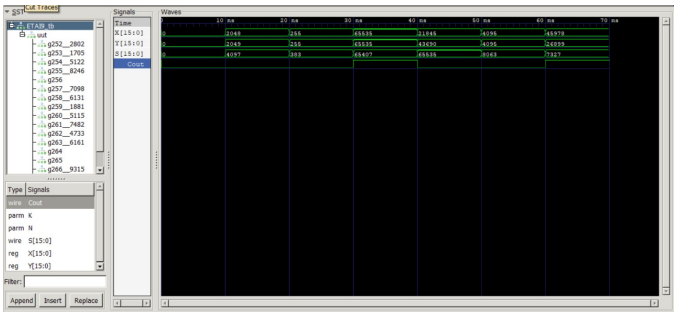

Fig. VIII.6: Genus Simulation of an ETA-I adder for k=9

Fig. VIII.7: Genus Simulation of an ETA-I adder for k=10

Fig. VIII.8: Genus Simulation of an ETA-I adder for k=11

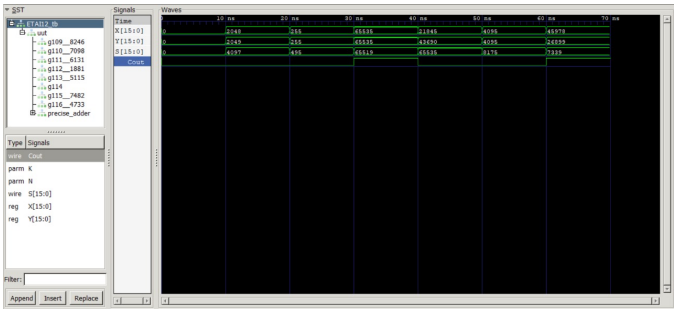

Fig. VIII.9: Genus Simulation of an ETA-I adder for k=12

IX. CPETA : Carry Predicting Error Tolerant Adder

The accuracy of ETA-I degrades due to a lack of carry predictor. Also, the modified XOR gate was implemented at the transistor level, resulting in design overhead. To enhance accuracy and enable a gate-level implementation, CPETA was introduced. The design of CPETA comprises two sections: the accurate part and the inaccurate part. The accurate part includes a precise adder, such as the Ripple Carry Adder (RCA) and the Carry Lookahead Adder (CLA). The proposed adder takes a carry-in signal that is speculated by an AND operation of n-k-1 LSB input bits. The sum generator includes two NOR gates that produce the correct sum for the n-k-1 bit position.

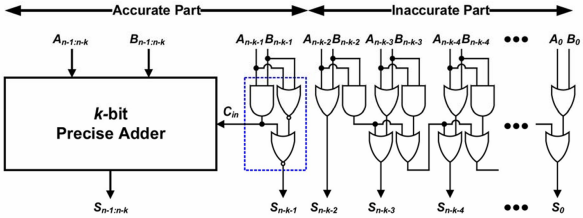

Fig. IX.1: Circuit diagram of a CPETA Adder

The inaccurate part consists of only OR and AND logic gates to produce approximate summation outputs for n-k-2 input bits, whereas the ETAI uses transistor-level implementation to achieve the same result. The inaccurate part sets all the remaining LSB output to 1 if both input bits are 1 when checking from the n-k-1 input bit position to the LSB as the conventional ETAI does.

The circuit diagram for a CPETA adder is shown in figure (IX.1).

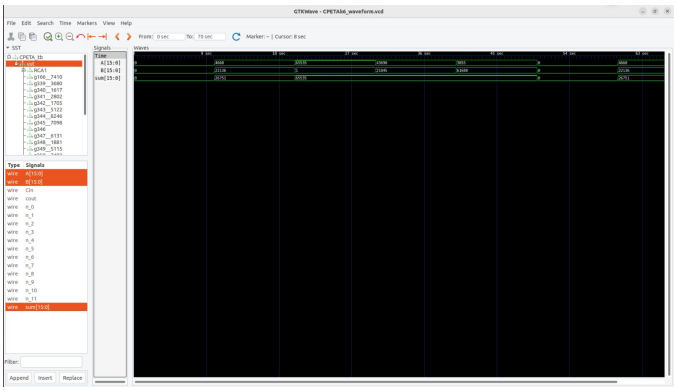

Fig. IX.2: Genus Simulation of a CPETA adder for k=6

X. ECPETA : Enhanced Carry Predicting Error Tolerant Adder

ECPETA is a modified form of the previous adder, CPETA. ECPETA utilizes both the (n-k-1)th and (n-k-2)th LSB inputs with an additional OR gate to predict the carry to improve the accuracy.

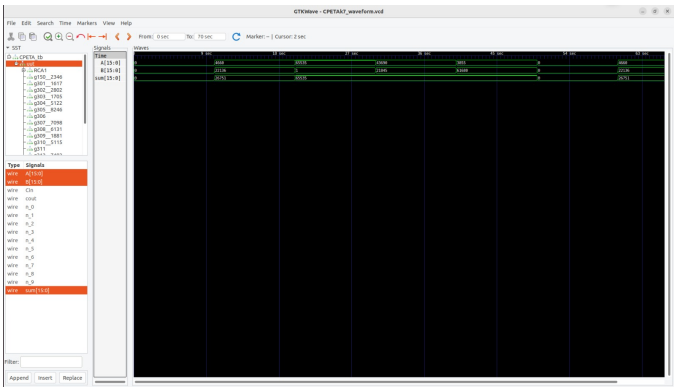

Fig. IX.3: Genus Simulation of a CPETA adder for k=7

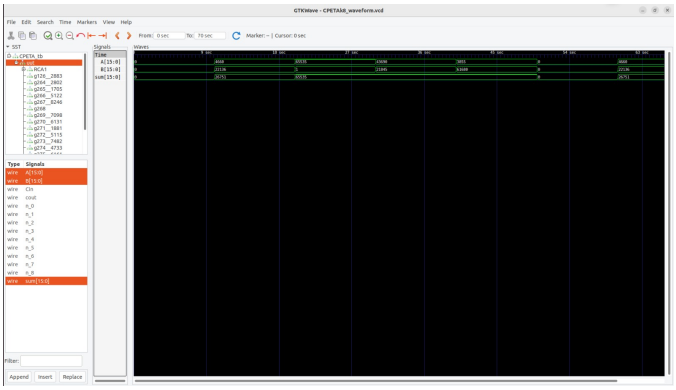

Fig. IX.4: Genus Simulation of a CPETA adder for k=8

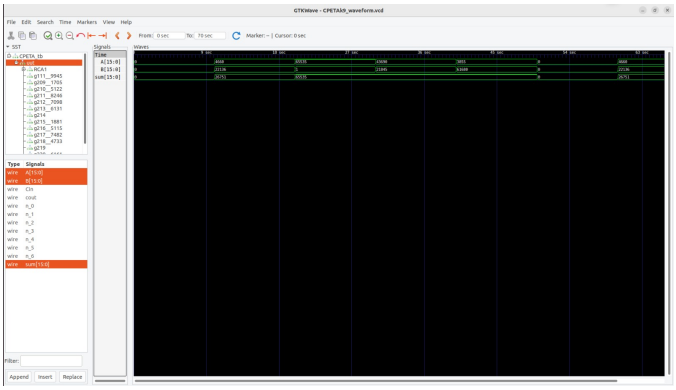

Fig. IX.5: Genus Simulation of a CPETA adder for k=9

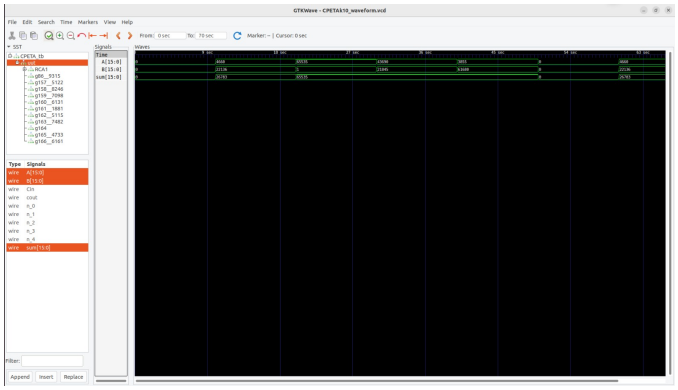

Fig. IX.6: Genus Simulation of a CPETA adder for k=10

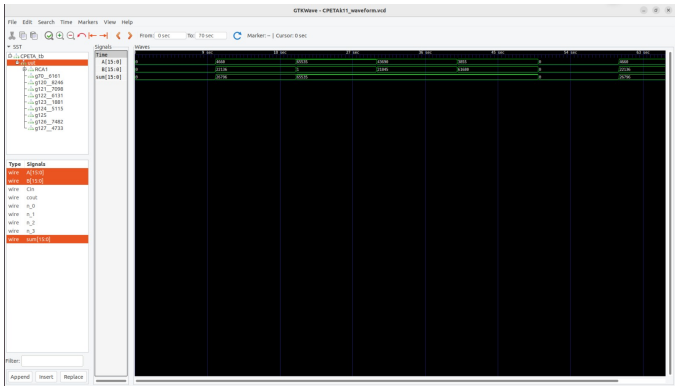

Fig. IX.7: Genus Simulation of a CPETA adder for k=11

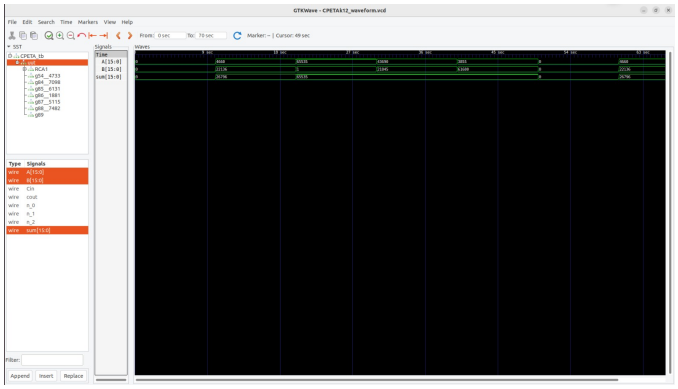

Fig. IX.8: Genus Simulation of a CPETA adder for k=12

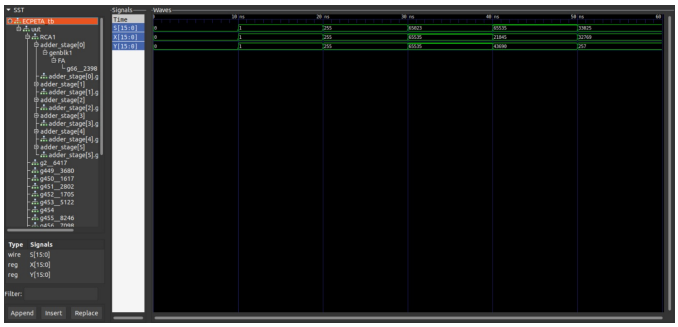

Fig. X.1: Circuit diagram of an ECPETA Adder

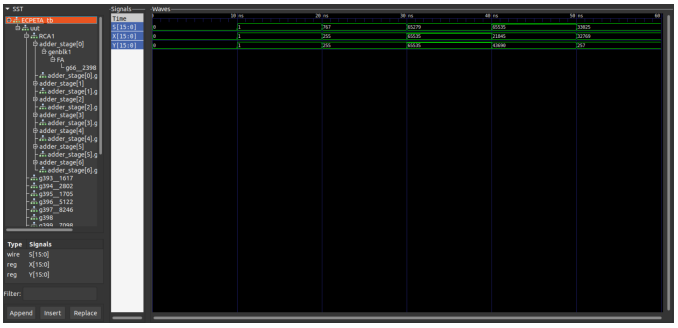

Fig. X.2: Genus Simulation of a ECPETA adder for k=7

[h!]

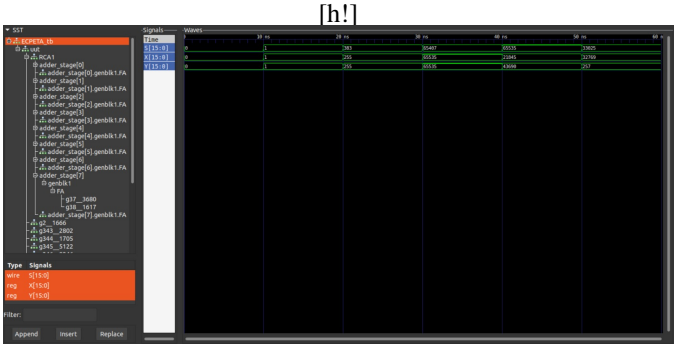

Fig. X.3: Genus Simulation of a ECPETA adder for k=8

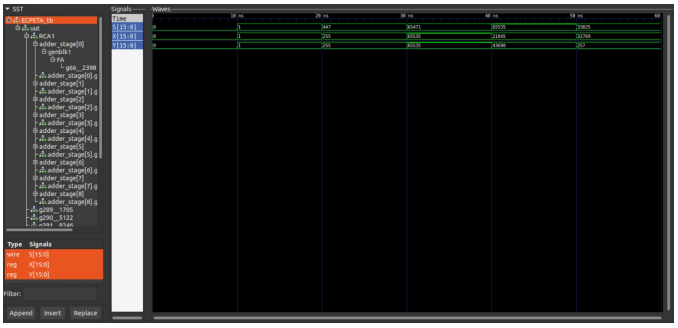

Fig. X.4: Genus Simulation of a ECPETA adder for k=9

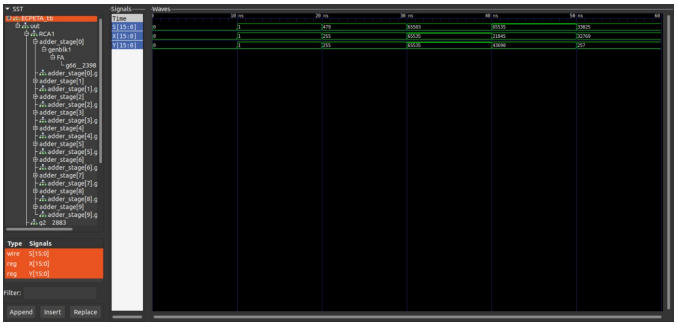

Fig. X.5: Genus Simulation of a ECPETA adder for k=10

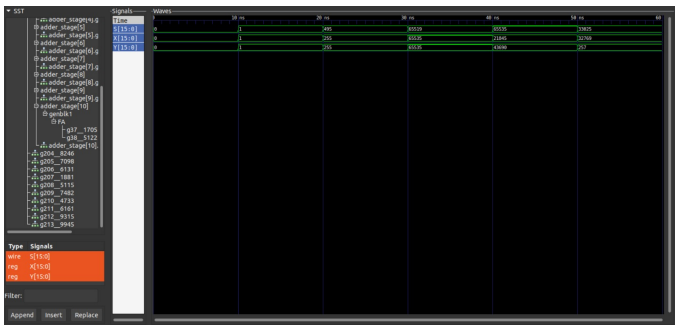

Fig. X.6: Genus Simulation of a ECPETA adder for k=11

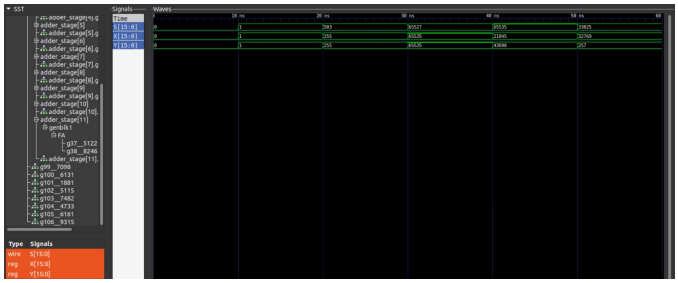

Fig. X.7: Genus Simulation of a ECPETA adder for k=12

XI. HERLOA : Hybrid Error Reduction Lower Paer OR Adder

HERLOA is formulated by applying a hybrid error reduction scheme to the LOA. The proposed design uses a 'k' bit precise adder for 'k' MSB bits combined with an approximate 'n-k' part for the 'n-k' LSB bits. The precise adder used is a Ripple Carry Adder. The approximate part uses OR and XOR operations. The carry is determined by performing the AND operation on the 'n-k-1' bits of the inputs, which is analogous to the LOA. The intermediate sum of the approximate part is calculated using the LOA (Lowest Order Approximation) scheme. The distinction is that an XOR gate is used for the last bit of the approximate part, which enhances its accuracy relative to the standard LOA method. The proposed adder performs further error reduction only when the 'n-k-2'th bits of the inputs are one.

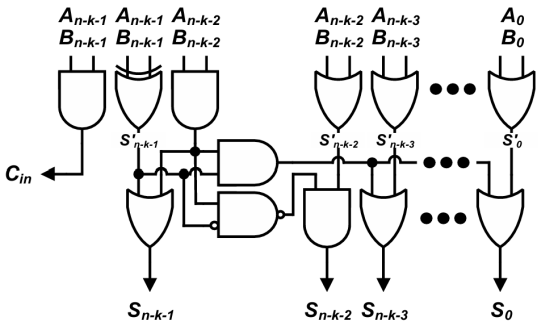

Fig. XI.1: Circuit diagram of a HERLOA Adder

In the proposed error reduction scheme, we first check if the 'n-k-2'th bits are one. If the condition is satisfied, two error reduction schemes can occur based on the 'n-k-1'th bits. If the output of the XOR operation is zero, this indicates that the bits at positions 'n-k-1' are identical, meaning they are either both ones or both zeros. Consequently, the sum bit at position 'n-k-1' will be fixed at one, while the sum bit at position 'n-k-2' will be fixed at zero. This is the correct summation of outputs at the corresponding positions under the given inputs. This scheme leads to a 2 to the power n-k-2 reduction in the error distance. If the output of the XOR gate is zero implies that a carry is generated at 'n-k-2' th bit and propagates to the precise adder. Since the carry is predicted using the AND operation of the 'n-k-1'th input bits, it will result in zero. Therefore to compensate for the carry bit all the sum bits from 'n-k-3' are set to one. Thus the calculated sum will always be greater than the approximate sum in this case.

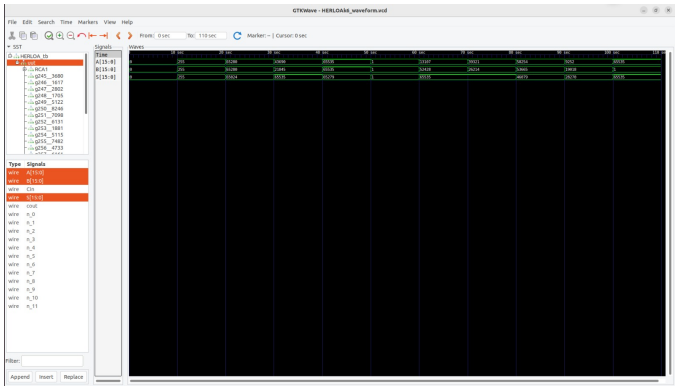

Fig. XI.2: Genus Simulation of a HERLOA adder for k=6

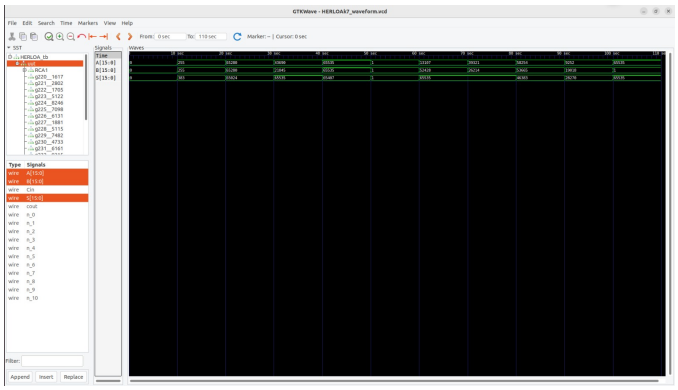

Fig. XI.3: Genus Simulation of a HERLOA adder for k=7

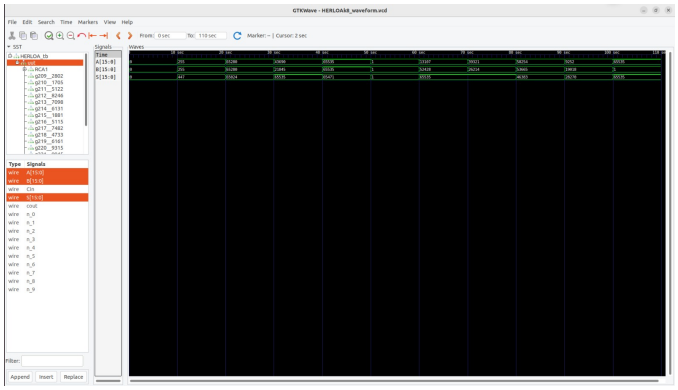

Fig. XI.4: Genus Simulation of a HERLOA adder for k=8

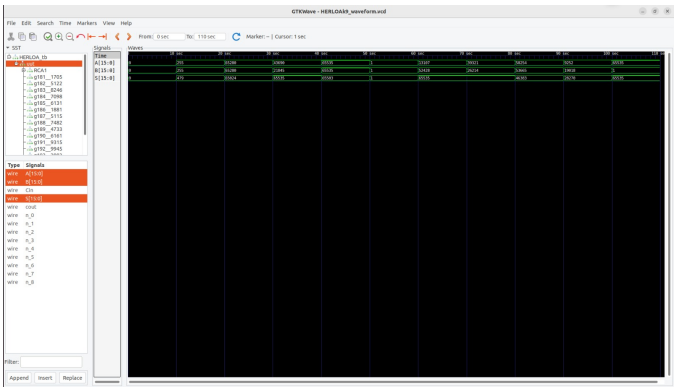

Fig. XI.5: Genus Simulation of a HERLOA adder for k=9

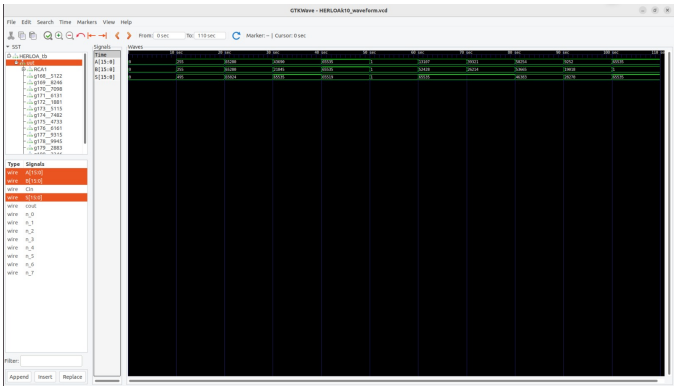

Fig. XI.6: Genus Simulation of a HERLOA adder for k=10

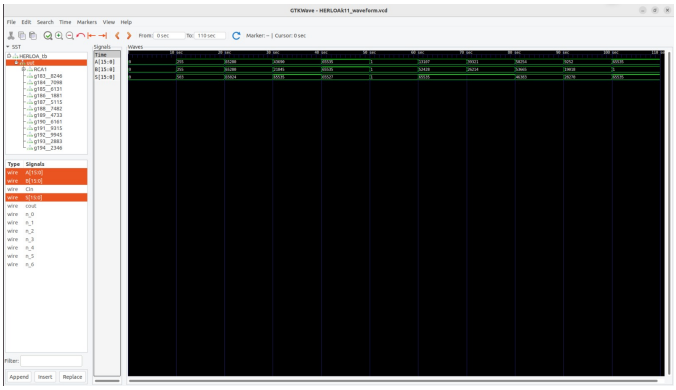

Fig. XI.7: Genus Simulation of a HERLOA adder for k=11

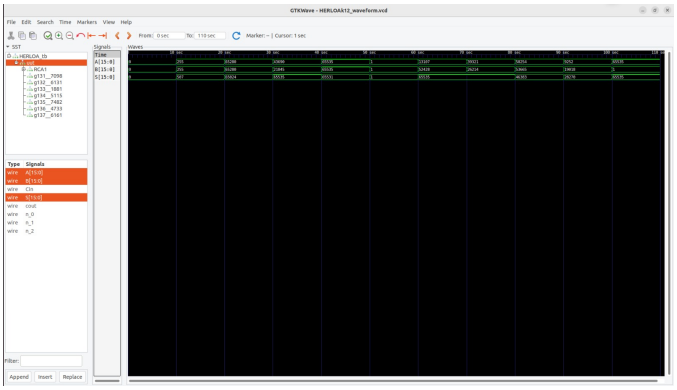

Fig. XI.8: Genus Simulation of a HERLOA adder for k=12

XII. Results

For benchmarking of all the above mentioned adders we use the following metrics: Error Rate (ER), Mean Error Distance (MED), Mean Ratio Error Distance (MRED) and Normalized Mean Error Distance (NMED).

$$MED = \frac{1}{n}\sum_{i=1}^{n} ED_i \,,$$

$$MRED = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{ED_i}{S_{i,\,accurate}}\right|,$$

$$NMED = \frac{MED}{D} = \frac{1}{n}\sum_{i=1}^{n}\frac{ED_i}{D},$$

Fig. XII.1

$$\text{Power NMED} = (\text{Total Power}) \cdot \text{NMED} \tag{1}$$

$$\text{Energy NMED} = (\text{Power} \cdot \text{Delay}) \cdot \text{NMED} \tag{2}$$

$$\text{Energy-Delay-Product NMED} = (\text{Energy} \cdot \text{Delay}) \cdot \text{NMED} \tag{3}$$

$$\text{Normalized Power NMED} = \frac{(\text{Total Power}) \cdot \text{NMED}}{(\text{Total Power of LOA}) \cdot (\text{NMED of LOA})} \tag{4}$$

$$\text{Normalized Energy NMED} = \frac{(\text{Total Energy}) \cdot \text{NMED}}{(\text{Total Energy of LOA}) \cdot (\text{NMED of LOA})} \tag{5}$$

$$\text{Normalized EDP-NMED} = \frac{\text{EDP-NMED}}{\text{EDP-NMED of LOA}} \tag{6}$$

In figure(XII.2) we can see that the proposed HERLOA adder has the least error rate. This is because it essentially has a k+1 bit precision adder with the hybrid error reduction scheme. OLOCA has the worst error rate because most values are directly set to 1.

In figure(XII.3) we can see that ECPETA adder has the least MED. This is because it essentially has a k+1 bit precision adder, the Cin is determined using 2 bits (n-k-1 and n-k-2) and 1 is propagated once carry should be generated within the inaccurate part which reduces error distance greatly. LOAWA has the worst MED because there is no Cin prediction.

HERLOA has the best trade-off's between energy,delay,power and accuracy compared to other adders. As we can see the Normalized EDP NMED is least for HERLOA. Normalized EDP NMED is a total measure of all the parameters.

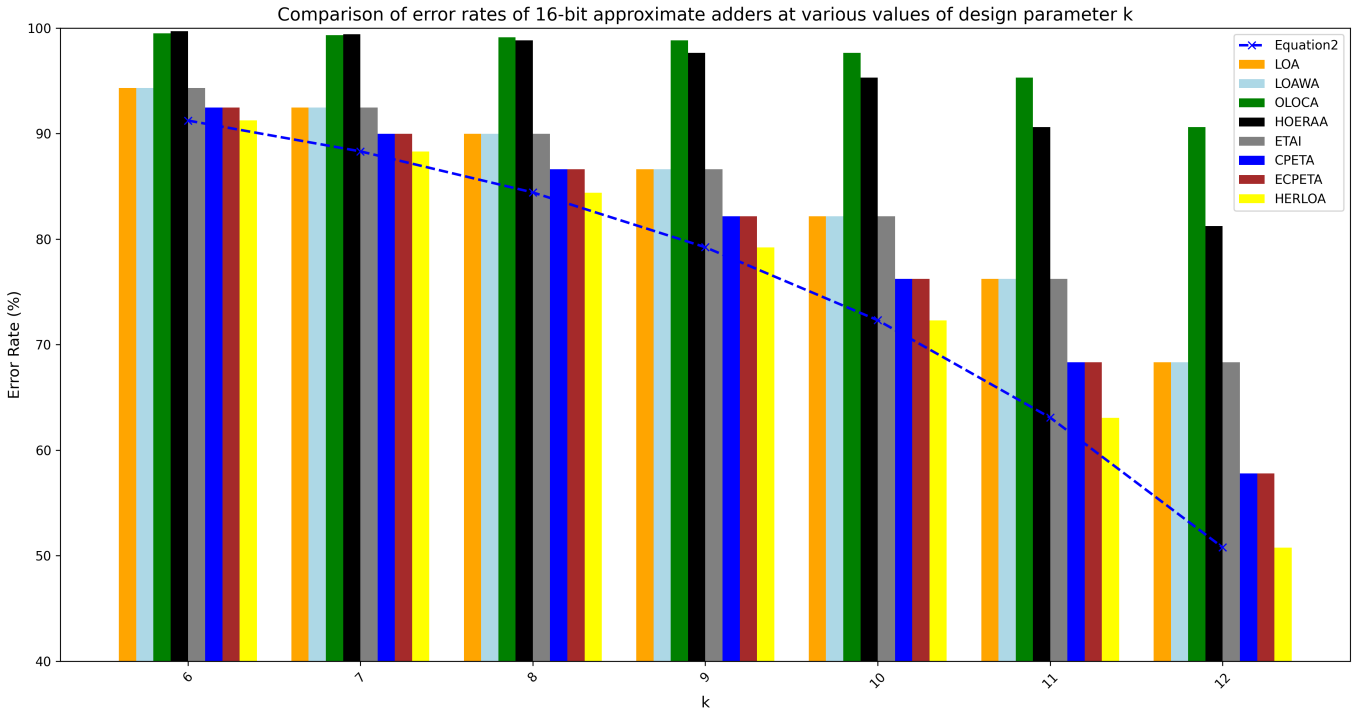

Fig. XII.2: Circuit diagram of a HOERAA Adder

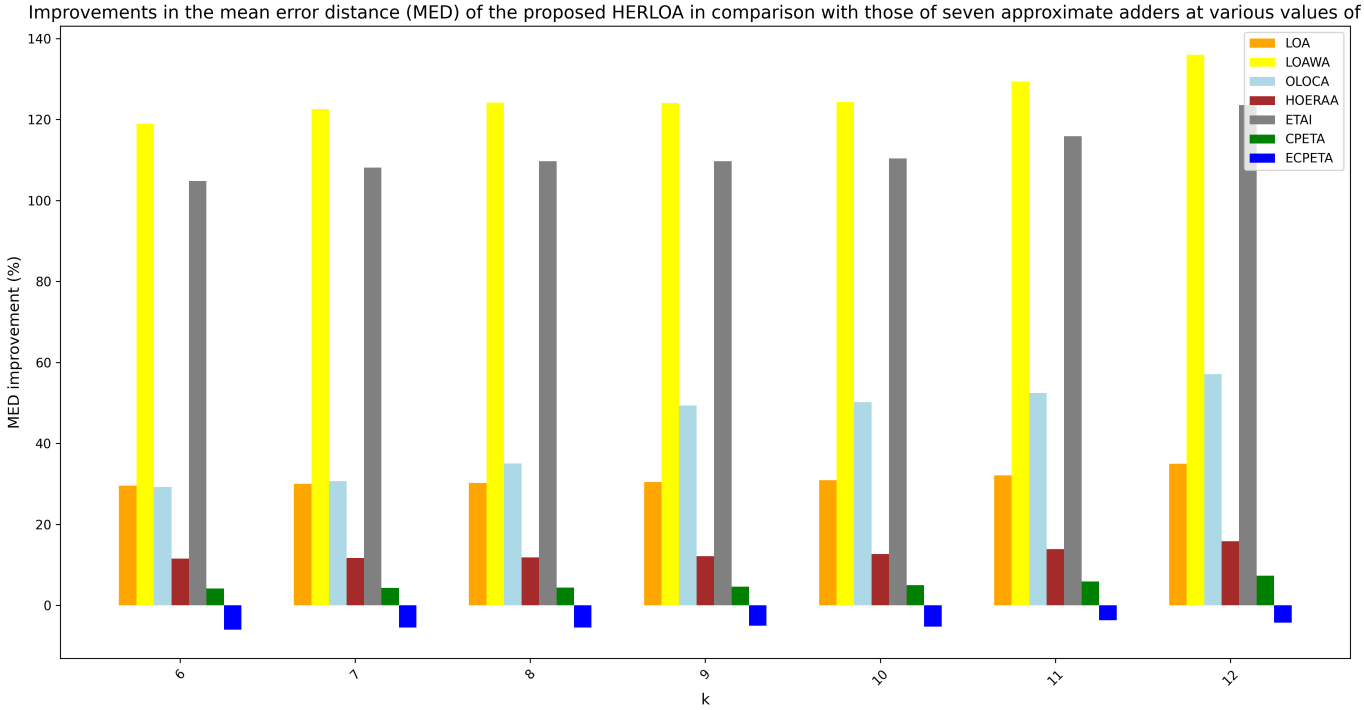

Fig. XII.3: Circuit diagram of a HOERAA Adder

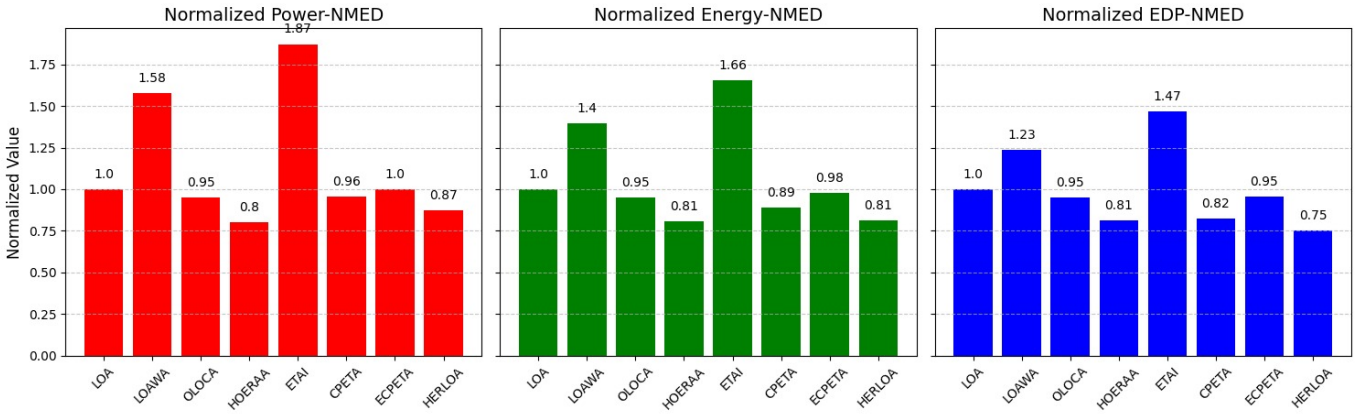

Fig. XII.4: Circuit diagram of a HOERAA Adder

k	LOA	LOAWA	OLOCA	HOERAA	ETAI	CPETA	ECPETA	HERLOA
6	5.89	6.57	5.89	5.89	6.57	5.88	5.18	5.88
7	5.18	5.88	5.18	5.17	5.88	5.17	4.45	5.17
8	4.45	5.17	4.45	4.45	5.17	4.45	3.73	4.45
9	3.73	4.45	3.73	3.73	4.45	3.73	2.98	3.73
10	2.98	3.73	2.98	2.98	3.73	2.98	2.25	2.98
11	2.25	2.98	2.25	2.25	2.98	2.25	1.47	2.25
12	1.47	2.25	1.47	1.46	2.25	1.46	0.64	1.46

TABLE I: Mean relative error distances (MREDs) of 16-bit approximate adders at various values of design parameter k

Fig. XII.5

Adder	Area (μm^2)	Delay (ps)	Power (μW)	PDP (μJ)	EDP (mJ·s)	ER (%)	MED	MRED	NMED($\times 10^{-3}$)
RCA	82.080	709	2.51	1782	1263	N/A	N/A	N/A	N/A
CLA	82.080	709	2.51	1782	1263	N/A	N/A	N/A	N/A
LOA	53.352	382	1.37	523	200	89.98	112.35	4.45	1.71
LOAWA	50.616	338	1.25	423	143	89.98	193.35	5.17	2.95
OLOCA	45.144	382	1.25	478	182	99.12	116.51	4.45	1.78
HOERAA	48.222	384	1.28	490	188	98.83	96.49	4.45	1.47
ETAI	68.742	338	1.59	537	181	89.98	180.87	5.17	2.76
CPETA	69.084	354	1.64	579	205	86.63	90.06	4.45	1.24
ECPETA	68.742	373	1.71	637	238	86.63	81.55	3.73	1.37
HERLOA	64.638	354	1.55	549	194	84.39	86.26	4.45	1.32

TABLE II: Summary of performances of various 16-bit approximate adders with $n = 16$ and $k = 8$.

Fig. XII.6

REFERENCES

[1] P. Albicocco, G. Cardarilli, A. Nannarelli, M. Petricca, and M. Re, "Imprecise arithmetic for low power image processing," in *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, ser. Asilomar Conference on Signals, Systems and Computers. Conference Record. United States: IEEE, 2012, pp. 983–987, 46th Asilomar Conference on Signals, Systems and Computers ; Conference date: 04-11-2012 Through 07-11-2012. [Online]. Available: http://www.asilomarssc.org/

[2] D. M. Harris and S. L. Harris, "5 - digital building blocks," in *Digital Design and Computer Architecture (Second Edition)*, second edition ed., D. M. Harris and S. L. Harris, Eds. Boston: Morgan Kaufmann, 2013, pp. 238–293. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123944245000057

[3] H. Seo, Y. S. Yang, and Y. Kim, "Design and analysis of an approximate adder with hybrid error reduction," *Electronics*, vol. 9, p. 471, 03 2020.

[4] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proceedings of the 2009 12th International Symposium on Integrated Circuits*, 2009, pp. 69–72.