EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Design and Comparison of 24-bit Three Operand Adders using Parallel Prefix method for Efficient Computations

S. Usha[1*] and M. Kanthimathi[2]

[1]Associate Professor, Sri Sairam Engineering College, Scholar, Anna University, Chennai, India
[2] Professor, Sri Sairam Engineering College, Supervisor, Anna University, Chennai, India

Abstract

Binary Three-operand adder serves as a foundation block used within security and Pseudo Random-Bit Generator (PRBG) systems. Binary Three-operand adder was designed using Carry Save Adder but this consumes more delay. Therefore, a Parallel Prefix Adder (PPA) method can be utilized for faster operation. The canonical types of PPA result in a lesser path delay of approximately O (log2 n). These adders can be designed for 8, 16, 24 or n bits. But this work is focused on developing a 24-bit three-operand adder that takes three 24-bit binary numbers as input and generates a 24-bit sum output and a carry using five different PPA methods The proposed summing circuits are operationalized with Hardware-Description-Language (HDL) using Verilog, and then subjected to synthesis using Field -Programmable Gate- Array (FPGA) Vertex 5. On comparing the proposed adders, it shows that the delay and the size occupied are significantly less in the Sklansky PPA. These faster three-operand adders can be utilized for three-operand multiplication in image processing applications and Internet of Things (IoT) security systems.

Keywords: PPA, Three-Operand Adder, Modular Arithmetic, FPGA

Received on 11 November 2023, accepted on 23 January 2024, published on 01 February 2024

doi: 10.4108/eetsis.5004

*Corresponding author. Email: usha.ece@sairam.edu.in

1. Introduction

Developing cryptographic algorithms in hardware is crucial to attaining optimal system achievement and ensuring the security of systems. Many cryptographic algorithms rely on modular arithmetic for various math operations like modulo exponentiation, residue addition, and modular product. Therefore, the efficacy of cryptanalysis methods depends on how well these congruential arithmetic operations are implemented. One notable approach for efficiently performing residue multiplication operations is the Montgomery method, which heavily depends on a three-operand binary addition technique [1].

This algorithm has shown outstanding efficiency in executing these crucial cryptographic operations. At the moment, technological progress is taking place at an exceptionally rapid speed over a very short period. The circuits that are now being designed include billions of components yet need very little space, move very quickly, and use very little power. Therefore, considering area, speed, and power is essential to designing every circuit. It is necessary to build a circuit with low space limitations and low delay limits to meet the requirements of the present trend.

The arithmetic units are the primary building elements of any processing unit, and they are responsible for carrying out a variety of arithmetic operations. The addition process is particularly crucial in this context among all the mathematical operations. In the literature survey of adder designs, various adder algorithms starting from Ripple Carry Adder to Approximate Adder have been researched. All of these additions will only be able to handle a total of two operand inputs; but to support three operand inputs, there is a need to recreate the building block as a component with support for the carry chain approach.

Either one adder that handles three inputs or two adders that handle two binary inputs each may be used to perform the addition of three operands. When it comes to adding three operands used in cryptography algorithms and various bit-generation approaches, the PPA is an efficient mechanism pertaining to size and speed. To minimize latency on the critical path, like a two-operand PPA, the PPA algorithm can also be utilized for the summation of three-operand inputs. It not only makes a difference in O (log2 n) in terms of the delay on the critical path, but it also makes a difference in the order of O in terms of the area (n log2 n).

As a result, it is vital to build an effective structure in a Very Large Integrated Circuit designed for the rapid three-operand addition of binary numbers with the fewest gates required by the hardware. Therefore, in this paper, a new faster, size optimized PPA technique is proposed for three operands binary addition. For analyzing PPA, in terms of performance, five different approaches, utilizing conventional PPA including Sklansky PPA were employed for adding three 24-bit numbers.

To achieve the addition of three operands, a pre-compute of the bitwise addition is done and then carry-prefix computing logic is performed. This technique consumes a significant power, but this will take an amount less size while simultaneously cutting down on the propagation delay when compared to the PPA two-operand adder. Besides, the recommended adder structure is enacted using the Verilog-HDL, followed by synthesis on Vertex 5 FPGA, and finally, it is implemented on Xilinx for functional validation.

The work is structured as stated: Section 2 provides a review of the literature and the developments made previously are presented. The applications and techniques of the suggested 24-bit three-input adders of various types are described in Section 3. Validation of the results and implementation is presented in Section 4. Finally, Section 5 concludes the work.

2. Literature Review

Ravi Payal et al. [2] introduced a carry-look-ahead adder design where the generation of a carry network is structured using prefix trees, resulting in the development of two distinct types of PPA given by Kogge-Stone and Ladner-Fischer. This adder is widely employed in industries for meeting performance targets. In [3], the authors stated that the area requirements and circuit complexity are reduced in the classical PPA structures. A very low latency additional circuitry is given in [4]. A new scheme that offers a reduction in component count and fewer logic levels is proposed in [5].

Jackson et al. [6] introduced innovative adder designs that achieve reduced complexity across all addition stages. A multiplication algorithm based on redundant binary representation, applicable to both unsigned and signed integers, as discussed in [7]. To decrease the power consumed by the multiplier, an adapted full-adder approach is employed [8]. Additionally, [9] employed a new configuration of the Wallace-tree multiplier, incorporating PPA in the final stage. To compress partial products, three innovative 4:2-type compressors were suggested and integrated into multipliers [10].

Amir Fathi et al. [11] devised ultrahigh-speed compressors to demonstrate enhanced speed performance and power-delay product efficiency. In [12], a novel digit-serial structure for executing the multiplication of three operands featuring a low-complexity implementation was achieved by leveraging a newly developed Karatsuba algorithm [13] which can be used in cryptography applications.

In [14], a fresh method for a three-operand multiplier is given, featuring a basic two-level radix-4 recoding technique to minimize costs and latency compared to other techniques. An elliptic curve cryptographic processor that supports 256-bit point multiplication is proposed in [15]. Montgomery modular multiplier that uses a configurable carry save adder is proposed to attain higher speed and significant area-time product improvement is developed in [16].

The literature review highlights the necessity for a high-speed, simple design adder to expedite operations, serving as a foundational component in various arithmetic circuits. This study aims to address this particular challenge.

3. Applications and Techniques

Binary three-operand addition is one of the important math operations and this has been utilized in modular arithmetic architecture and Linear Congruential Generator based methods. The use of adder approaches will always appear in two operand addition. Furthermore, the proposed effort of this application concentrated on Three Operand Binary Multiplication. It is necessary to build a circuit with low area constraints and low delay limits to meet the requirements of the present trend.

The numerous arithmetic operations are carried out by a variety of arithmetic units, which are important building elements of any processing unit. Multiplication operation is one of the most essential mathematical procedures. In the literary analysis of multiplier designs, many different methods for multiplying are investigated. These algorithms include the Binary, Booth, Array, Dadda and Wallace tree multipliers.

The tree multiplier by Wallace is beneficial in comparison to other kinds of multipliers. All this multiplication is carried out with only two operand inputs here. However, three-operand multiplication is necessary for a greater number of applications and algorithms. In this case, it is decided to utilize a Wallace Tree Multiplier for Three operand Multiplications. To perform this, a Cascaded Method Architecture is chosen to construct a three-operand Wallace Tree Multiplication, which is given in Fig.1.

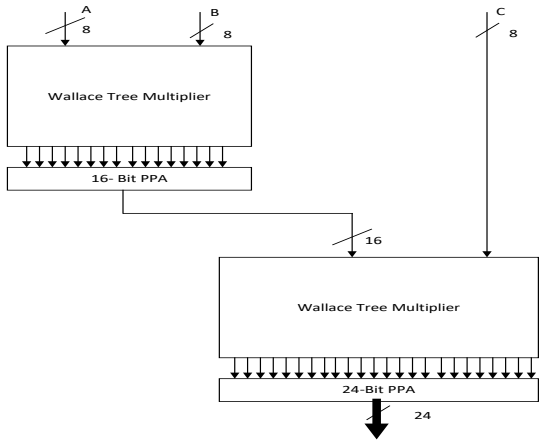

Figure 1. Three Operand Cascaded Multiplier based on Wallace Tree

The cascaded Multiplier established on Wallace Tree with three operands is given in Fig 1. It will be created with two operands for the Wallace Tree Multiplier, making it a cascaded method. The first Wallace Tree Multiplier will output 16 bits from a PPA, and when the second Wallace Tree Multiplier receives these 16 bits as input, it will perform multi-bit multiplication and the final output is 24 bits from a PPA. This application of three-operand Wallace tree multiplication may be used in Image Processing, Signal Processing, Arithmetic operations, microprocessors, Controller, and so on.

This work suggests a 24-bit three-operand binary adder method as a way to minimize the complexity of three-operand-multiplier applications. During multiplication, the first step is producing partial products, the function of a Wallace-tree multiplier is similar to any other multiplier. In the second step of the process, the Wallace-tree multiplier combines the partial products from the preceding three rows. After that, the newly formed sum along with carry, is added to the subsequent row of partial products. Repetition of this adding procedure is continued until the completion of the formation of the final items. For this method of adding rows together, both half and full adders are used. Therefore, the function that adders perform in the production of final product terms is a very significant one.

The functioning of the multiplication is going to be impacted in some way by the speed of the addition. The PPA adder structure that was used in the development of the Wallace-tree multiplier plays a vital part in the method of enriching the performance of the multiplication operation. In this particular piece of research, the three-operand adder is designed using five major PPAs. The widely used PPAs are - the Kogge-stone, Brent-Kung, Han-Carlson, Ladner-Fischer and Sklansky. The architecture of these mentioned PPA is the same; with the primary differentiator being the arrangement of the prefix network which can use black and grey cells, and the connections between them.

3.1. Techniques

To achieve the addition of three operands required by modular operations, the adder mechanism is detailed in this part. The adder that uses prefix computation in parallel is a PPA. This PPA is an efficient circuit and gives the best performance when compared with the other adders.

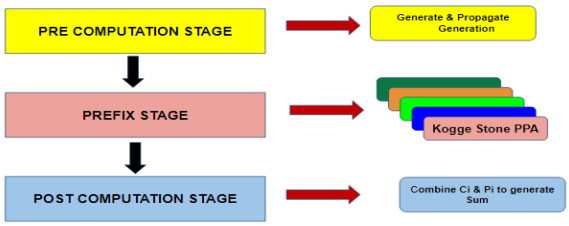

Figure 2. Traditional PPA for two operands addition

To find the sum of three binary input operands, three operands PPA use four steps instead of three steps as in traditional prefix adders as given in Figure 2. The intended three-operand adder with four steps is shown in Figure 3. These structures use full adders in bit-addition logic, XOR and AND gates, propagate logic, generate logic, and XOR gates are used in the final sum logic.

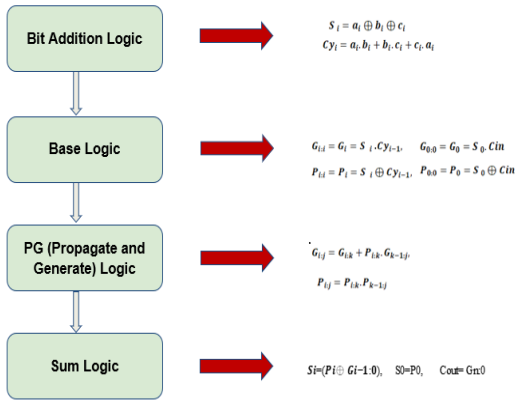

Figure 3. Steps involved in Three-Operand Addition

The sequential order of four steps is provided with their expressions.

Step-1: Addition of three bits :

$$Sum'_i = A_i \oplus B_i \oplus C_i$$
$$Carry_i = A_i . B_i + B_i . C_i + C_i . A_i. \quad (1)$$

Step-2: Propagate and Generate generation step:

$$Gen_{i:1} = Gen_i = Sum'_i . Carry_{i-1}$$
$$Pro_{i:1} = Pro_i = Sum'_i \oplus Carry_{i-1},$$
$$Gen_{0:0} = Gen_0 = Sum'_0 . Cin$$
$$Pro_{0:0} = Pro_0 = Sum'_0 \oplus C_{in} \quad (2)$$

Step-3: Group Propagate and Generate generation step:

$$Gen_{i:j} = Gen_{i:k} + Pro_{i:k} . Gen_{k-1:j}$$
$$Pro_{i:j} = Pro_{i:k} . Pro_{k-1:j} \quad (3)$$

Step-4: Final Sum step:

$$Sum_i = (Pro_i \oplus Gen_{i-1:0}),\ Sum_0 = Pro_0,\ C_{out} = Gen_{n:0} \quad (4)$$

The novel adder method involves the execution of the addition operation on three n-bit operands in a total of four separate steps as mentioned in the equations. During the initial step (bit-addition logic), n full adders are used to conduct the bit-wise summation of three operands of n-bits. Each full adder is responsible for computing the sum (Sum_i) and carry ($Carry_i$) outputs for the addition. The logical equations for calculating the sum (Sum_i) and carry ($Carry_i$) signals are stated in Step 1, and Figure 4 depicts the gates used in the first step, base logic, final sum logic, and cells for group generate and propagate. Step 2 depicts the implementation of Step 1's logical expressions.

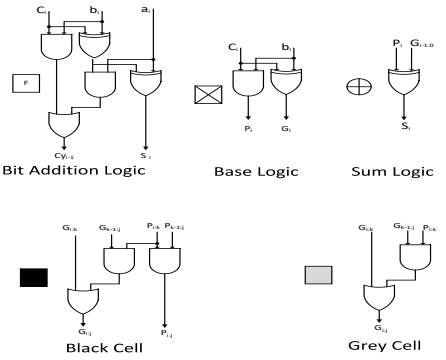

Figure 4. Logic Gate Diagram of bit-addition, base, sum logic, black and grey cell.

The "sum" (Sum_i) output bit from the current active full adder and the "carry" from the right-most neighbor (base logic) are combined to form the generate (Gen_i), propagate (Pro_i) bits in the first step. With the following logical phrase, the Gen_i and Pro_i signals can be calculated as the "squared saltire-cell," where n represents the used saltire-cells initially.

$$Gen_{i:i} = Gen_i = Sum'_i . Carry_{i-1}$$
$$Pro_{i:i} = Pro_i = Sum'_i \oplus Carry_{i-1} \quad (5)$$

In the presented adder method, the carry-input which is given externally, denoted by *Cin*, is taken into account while performing the addition of three n-bit inputs. During the process of calculating Gen_0 ($Sum'_0 \bullet Cin$) in the first saltire-cell of the logic, this extra carry-input signal, which is denoted by *Cin*, is used as input to the base step. The next step is the carry calculation stage, also known as the "generate and propagate logic" (*PG*) stage, and it is used to calculate the carry bit in advance. This stage is a mixture of grey, and black cells. The expression that computes the generates $Gen_{i:j}$, propagate $Pro_{i:j}$ is given below, which depicts the gate diagram of a grey and black cells logic.

$$Gen_{i:j} = Gen_{i:k} + Pro_{i:k} . Gen_{k-1:j}$$
$$Pro_{i:j} = Pro_{i:k} . Pro_{k-1:j} \quad (6)$$

The number of calculations in the suggested adder is as given in ($\log_2 n+1$), and hence, the latency of the planned adder is mostly driven by this chain of carry inputs. The last step is given as sum logic where the "sum (Sum_i)" bits are derived from the generate $Gen_{i:j}$ and propagate Pro_i using the formula, $Sum_i = (Pro_i \oplus Gen_{i-1:0})$. The carry output signal (C_{out}) is simply retrieved from the $Gen_{n:0}$.

The following example shown in Figure 5 gives the working of a 3-bit three operand Kogge-stone adder where the inputs are 4,3 and 7 in decimal with a carry input as 1 and the results are obtained as per the technique used and Cout is zero for this case.

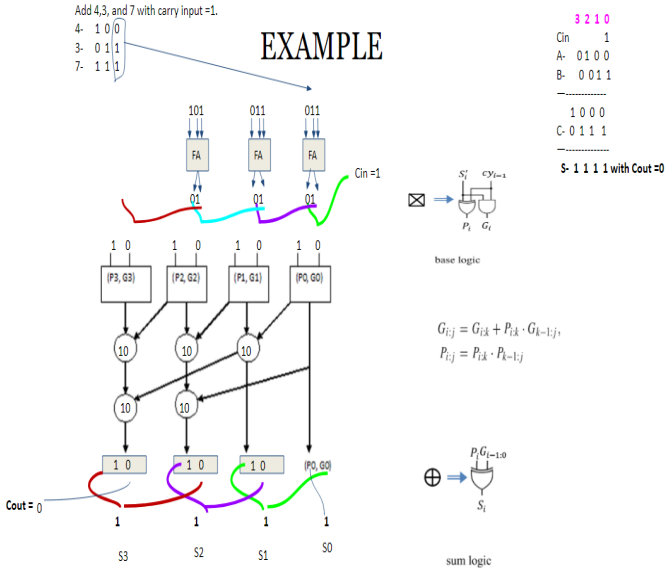

Figure 5. Block diagram of a 3 bit three operand Kogge stone PPA

3.2. Proposed 24-Bit Three Operand Adder Using Kogge-Stone PPA

The Kogge-Stone PPA is particularly useful in high-speed applications, even though it needs a large amount of space

and additional power. The structure has a latency that may be given as $\log_2 n$, The number of nodes is $[n(\log_2 n)-n+1]$. The plan of this adder will be difficult to understand due to the high number of connections that are required. Figure 6 shows the construction of 24-bit Three-operand Kogge-stone adder. This structure has 58 black cells and 23 grey cells.

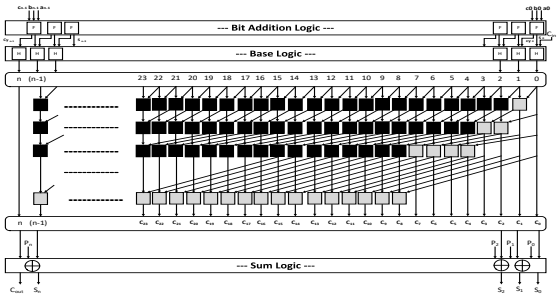

Figure 6. Structure of Proposed Three Operand 24-bit Kogge Stone Adder

3.3. Proposed 24-Bit Three Operand Adder Using Brent Kung PPA

Figure 7 depicts the structure of the Proposed Three Operand 24-bit Brent Kung Adder. This adder maintains the maximum depth while having fewer computing nodes, which accounts for the higher delay. When compared to the complexity of the interconnections in a Kogge-stone adder, the grey and black logic cells are less. The latency of the structure may be expressed as $[2(\log_2 n)-2]$, while the number of calculation nodes is $[2n-2-\log_2 n]$.

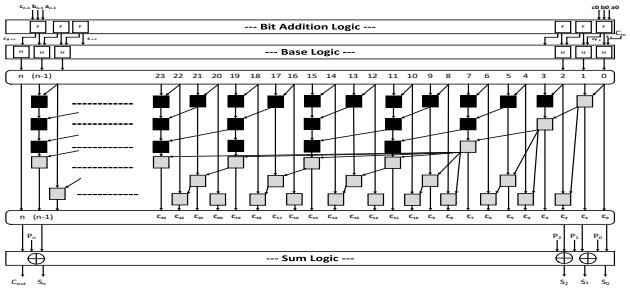

Figure 7. Structure of Proposed Three Operand 24-bit Brent Kung Adder

This adder comprises 20 and 23 black and grey cells respectively indicating that its overall area is much less than that of Kogge stone and Sklansky adders. When compared with the Kogge-stone adder, this adder's structure is far more straightforward. Also, the amount of fan out for this structure is much reduced in comparison to the Sklansky adder.

3.4. Proposed 24-Bit Three Operand Adder Using Sklansky PPA

The layout of a carry prefix network has a low logic depth, but this comes at the expense of a somewhat large fan out for some of the compute nodes. The structure has $n/2 \log_2 n$ computation nodes, which results in a latency that may be expressed as log2n. Along the critical route, the adder fan outgrows66 dramatically from the inputs to the outputs, which causes a significant rise in the amount of delay. The adder's performance deteriorates as the bits grow. Figure 8 illustrates the construction of a 24-bit Three Operand Sklansky adder, which is comprised of 29 and 23 black and grey cells. This indicates that its size is also smaller compared with a Kogge-stone adder; nevertheless, the electrical effort of the compute node increases, thereby increasing the adder's latency.

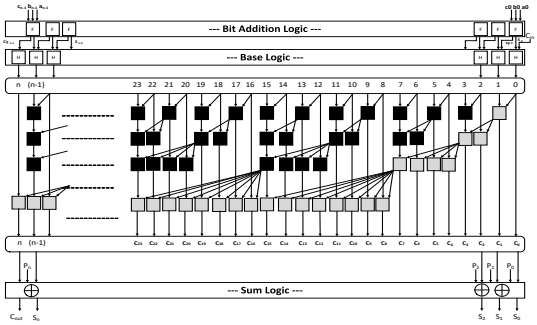

Figure 8. Structure of Proposed Three Operand 24-bit Sklansky Adder

3.5. Proposed 24-Bit Three Operand Ladner Fischer PPA

The Ladner Fischer adder design was constructed from the design of Sklansky adder, whose delay is $(\log_2 n)+1$. The number of nodes is $[(n/2)\log_2 n]$. Figure 9 illustrates the construction of a 24-bit Ladner-Fischer type three operand adder which is similar to that of the Three Operand Brent-Kung adder has 20 and 23 black and grey cells.

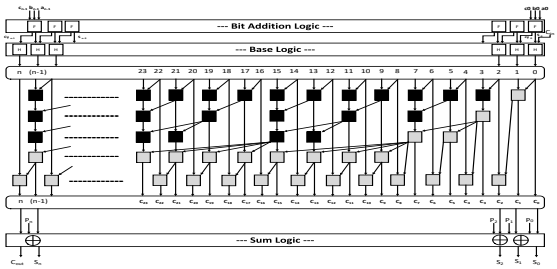

Figure 9. Structure of Proposed Three Operand 24-bit Ladner Fischer Adder

3.6. Proposed 24-Bit Three Operand Adder using Han Carlson PPA

The blend of Kogge-stone and Brent-Kung adder is the Han-carlson adder which is a hybrid design. The second block is divided into five phases, like the Brent Kung Adder and the levels that follow it resemble a Kogge-Stone Adder. The latency of the structure may be expressed as $(\log_2 n)+1$, and it has computation nodes that are $[(n/2)\log_2 n]$. Figure 10 depicts the construction of a 24-bit three-operand Han Carlson adder. It is made up of 29 and 23 black and grey cells respectively, which suggests that its size and interlinked complexity are lower than that of a Kogge-stone adder. However, its radix is higher, resulting in a longer delay.

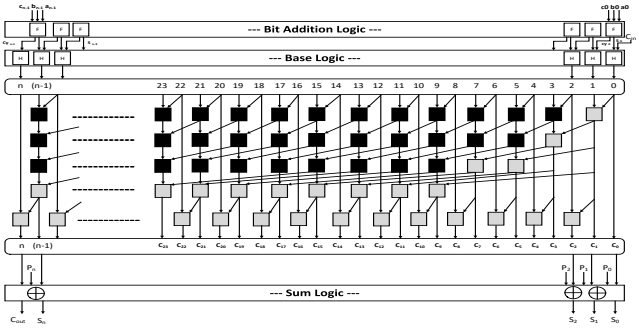

Figure 10. Structure of Proposed Three Operand 24-bit Han Carlson Adder

4. Results and Implementation

In this study, five different three-operand PPA structures are constructed with input sizes of 24 bits using Verilog HDL, Figure 9 displays the simulation for the recommended three-operand-adder and Register Transfer Level (RTL) Schematic of 24-bit Three-Operand Han-Carlson PPA is depicted in Figure 11. The RTL schematic of the 24-bit Three operand Han Carlson adder is shown in Figure 12.

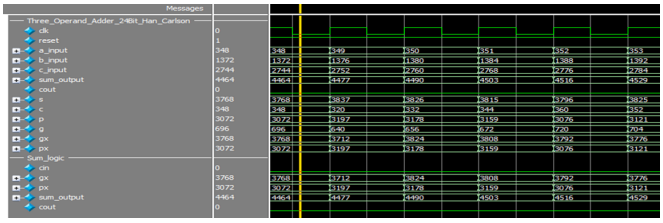

Figure 11. Simulation result of 24-bit Three Operand PPA

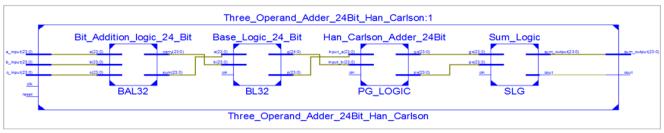

Figure 12. 24-bit Three-Operand Han-Carlson PPAs Register Transfer Level Schematic

The outputs of the three-operand PPA synthesis are compared, and the results are given in Table 1 in terms of size (in the number of Lookup tables), delay (in nanoseconds), power (in watts), Fan out, and also in the bar chart as shown in Figure 13. It is found that the operand 24bit Sklansky PPA has lower latency than other adders. Also, the number of Lookup tables occupied is minimized than the existing structures. The delay is 15% and the size occupied is 60% less in the Sklansky PPA when compared with the other adders. So, in applications that require a faster operation and area optimization that involves the 24-bit addition of three operands, Sklansky PPA is preferred over other 24-bit three-operand adders. The fanout of the Sklansky PPA is also higher.

Table 1. Comparisons of 24-Bit Three Operand PPA Synthesized Results using Xilinx Vertex-5 FPGA

	Comparisons of 24-bit Three Operand Parallel Prefix Addition				
	Kogge-Stone	**Brent-Kung**	**Sklansky**	**Ladner-Fischer**	**Han-Carlson**
LUT	113	80	77	80	83
Occupied Slice Register	37	22	24	23	24
IOB	97	97	97	97	97
Delay (ns)	7.676	8.868	**7.531**	8.182	8.366
Power(W)	3.516	3.516	3.516	3.516	3.516
Fanout	1.56	1.39	1.55	1.43	1.39

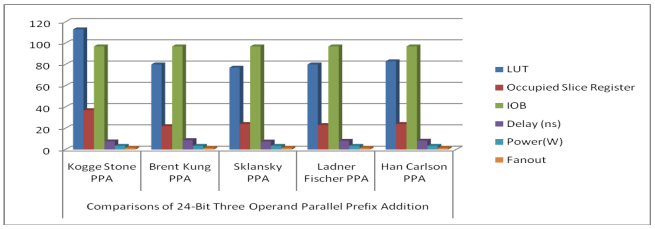

Figure 13. Performance comparison of five different 24-bit Three Operand PPA

5. Conclusion and Future Work

A high-speed efficient adder approach and Very Large-Scale Integrated design of it were proposed in this study to execute addition and to facilitate calculations in residual arithmetic efficiently, which was utilized in applications such as PRBG and cryptography encryption. To calculate the sum of three operands for a larger bit width, the

suggested three-operand adder method was a PPA-based three-operand adder that made use of four-stage structures. Since the adder was the limiting element in digital signal processing applications and cryptographic implementations, an optimized carry path could be enabled by these PPAs based on three operand adders.

The novel aspect of this recommended design was the reduction of running time and size in the propagate and generate logic and the first stage of the bit-addition step, which ultimately resulted in lessening of the overall path delay. These suggested architectures were built on Xilinx for functional validation and synthesized using Vertex 5 FPGA. In addition, it was reported that the Sklansky PPA took up less space, used less power and had a reduced delay compared with other three-operand adders. The future research possibilities of this study are to develop 32, 64 or higher-order three-operand adders and to design a three-operand multiplier that can be used for image multiplication and provide hardware security systems in IoT applications.

References

[1] Amit, K, Rakesh, P. High-Speed Area Efficient VLSI Architecture of Three Operand Binary Adder. IEEE Transactions on Circuits and Systems. 2020; Vol. 67; pp. 3944-3953.

[2] Ravi, P, Mahima, G. Design and Implementation of Parallel prefix adder for improving the performance of Carry Lookahead adder. International Journal of Engineering Research and Technology. 2015: Vol. 04: pp.566-571.

[3] Chandrika, B, Poorna, K. Implementation and Estimation of Delay, Power and Area for Parallel Prefix Adders. International Journal for Modern Trends in Science and Technology. 2016: Vol. 02, pp. 41-45.

[4] Han, T, Carlson, A. Fast area-efficient VLSI adders. IEEE 8th Symp. Computer Arithmetic. (ARITH). 1987: pp. 49–56.

[5] Ling, H.: High-speed binary adder. IBM J. Res. Develop. 1981; Vol. 25, pp. 156–166 .

[6] Jackson, R, Talwar, S.: High-speed binary addition. Conf. Rec. 38th Asilomar Conf. Signals, Syst. Computer. 2004; Vol. 2, pp. 1350–1353.

[7] Takagi, Yajima.: High-Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree. IEEE Transactions on Computers. 1985; Vol. C-34, pp. 789-796.

[8] Kokila, B, Nithish, K.: Low Power Wallace Tree Multiplier Using Modified Full Adder. 3rd International Conference of Signal Processing. 2015; pp.1-4 .

[9] Yamini, D, Krishna, S.: Design and analysis of High-Speed Wallace Tree multiplier using parallel prefix adders for VLSI Circuit Designs. International Conference on Computing, Communication and Networking Technology. 2020; pp.1-6.

[10] Pei, Haoran, Yi, Zhou, Hang, He, Yajuan.: Design of ultra-low power consumption approximate 4-2 compressors based on the compensation characteristic. IEEE Transactions on Circuits and Systems. 2020; Vol. 1, pp. 1–10.

[11] Amir, F, Behbood, M, Sarkis, A.: Very Fast, High Performance 5-2 and 7-2 Compressors in CMOS Process for Rapid Parallel Accumulations. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, 2020; Vol. 28, pp.1-10.

[12] Chiou, L, Shyan, M, Lee, Y.: New Digit Serial Three Operand Multiplier over Binary Extension Fields for High-Performance Applications. IEEE International Conference on Computational Intelligence and Applications. pp.498-502 (2017).

[13] Chiou, L, Chia, F, Shyan, M, Lee, Y.: Efficient Implementation of Karatsuba Algorithm Based Three Operand Multiplication Over Binary Extension Field. IEEE Access 2018;Vol.6, pp.38234-38242.

[14] McIlhenny, R, Ercegovac, D.: On the implementation of a three-operand multiplier. Thirty-First Asilomar Conference on Signals, Systems and Computers. 1997; Vol. 2, pp. 1168-1172.

[15] Islam, M, Hossain, M, Hasan, K, Shahjalal, M, Jang, M.: FPGA implementation of a high-speed area-efficient processor for elliptic curve point multiplication over a prime field. IEEE Access, 2019; Vol. 7, pp. 178811–178826.

[16] Kuang, S, Wu, K, Lu, R.: Low-cost high-performance VLSI architecture for Montgomery modular multiplication. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems. 2016; Vol. 24, pp. 434–443.