International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

VLSI Implementation of Ternary Operand Binary Addition Using Parallel Prefix Adder for Area Efficiency

[1]N. Mounika, [2]Dr.B.Lalitha

[1]PG Student, Department of Electronics and Communication Engineering, Sree Rama Engineering College, Rami Reddy Nagar, Karakambadi Road, Tirupati, Andhra Pradesh, India

[2]Professor, Department of Electronics and Communication Engineering, Sree Rama Engineering College, Rami Reddy Nagar, Karakambadi Road, Tirupati, Andhra Pradesh, India

ARTICLEINFO

Article History:

Accepted: 01 July 2023
Published: 24 July 2023

Publication Issue
Volume 10, Issue 4
July-August-2023

Page Number
247-254

ABSTRACT

Cryptographic applications and pseudo random generator perform modular arithmetic three operand is the basic fundamental unit used in all these applications. For performing three operand additions, CSA (carry save adder) is one of the most widely used adders. But in CSA in the final stage, carry is propagated which impacts the delay. Prefix parallel adders are therefore employed to get around this. The parallel prefix adders use more space even though performance in terms of latency is improved. Parallel prefix adders can also be used to build three operand adders. A brand-new, high-speed, and hardware-efficient adder technique is used to boost performance in terms of latency and area. This adder approach uses four stages to achieve three operand addition. Since Han Carlson adder is used in third stage, the suggested adder is not area efficient. To overcome this, in this paper we are replacing the Han Carlson parallel prefix adder with sklansky adder.

Keywords : CSA (Carry Save Adder), Sklansky Adder.

I. INTRODUCTION

Three operand additions is the most commonly used in modular multiplication which are widely used in cryptography applications. For maintaining optimum system performance along with retaining physical security, cryptography algorithms must be implemented on hardware [1]. In the cryptographic applications, modular arithmetic is used in which the three operand adder is the basic block [4]. In these applications which involve modular arithmetic, three operand addition is required it is basic fundamental operation in these application. Hence designing an efficient three operand adder is the need of the day.

With the rapid advancement in data communication, internet services data privacy can become an

important issue to be dealt with. To provide security, cryptographic applications are mostly used. Three operand adders is the basic fundamental unit used in this cryptography applications and modular arithmetic. As a result, the choice of essential building blocks affects how well modular arithmetic and cryptographic applications perform overall. The top module's performance varies depending on this foundational module.

When the addition is done between two operands or two input n-bit numbers, the adder is referred to as a two operand adder. For implementing two operand operations, different types of adders named as ripple carry adders (RCA), parallel prefix adders (PPA), carry skip adders (CSKA), and so on are available.

Consider ripple carry adder. It is the most commonly used adder for performing two operand addition.it is simply designed by cascading the 1 bit full adder cells. The 4bit RCA is depicted in the figure below.

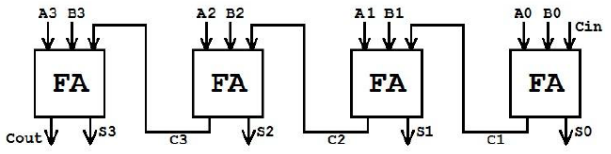

Fig.1 4-bit RCA

The Major drawback in this adder is its critical path delay. The second full adder has wait until the first full adder operation is performed. Similarly for performing n-bit addition it has to wait until (n-1) operation. Hence delay is more in case of ripple carry adder.

Consider carry skip adder. In this adder based on the skip logic, the carry will be either propagated or skipped. When carry is skipped the delay is effectively reduced. But the carry is skipped only for one case. But in the remaining cases the carry is being propagated which significantly impacts the performance of the adder and only the application it is used. The block diagram of carry skip adder is shown below.

From the figure 2, it can be observed that carry Propagation in the above adder is skipped only if all the propagate signals that are individually generated from all the individual full adders is logic high and for the other cases the carry is propagated. The CSKA Performance is improved compared to RCA. But the Speed in this adder is not as effectively improved.

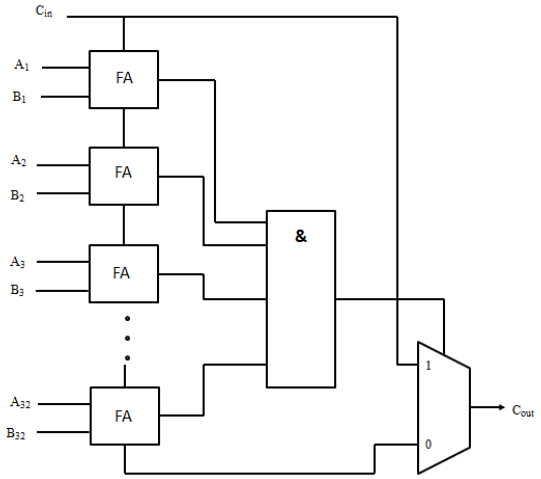

Fig.2. Logic diagram of Carry Skip Adder (CSKA)

Consider parallel prefix adder. In parallel prefix adder, the entire operation is performed in three stages to produce the final sum output. They are

1. Pre-processing.
2. Generation of carry.
3. Final processing.

Pre-processing: In this stage, from the inputs operands A and B, the propagate and generate signals are generated.

$$Pi = Ai \oplus Bi \dots\dots (1)$$
$$Gi = Ai.Bi \dots\dots (2)$$

Generation of carry: Here in this case, the using propagate and generate signals, the carry is generated for each bit. The operation performed in this stage is parallel. The expressions for are shown below

$$P_{(i:k)} = P_{(i:j)} \cdot P_{(j-1:k)} \dots\dots (3)$$
$$G_{(i:k)} = G_{(i:j)} + \left(G_{(j-1:k)} \cdot P_{(i:j)}\right) \dots\dots (4)$$

This carry is generated by the use of various cell structures known as Gray cells, Black cells, and Buffer cells. Final carry is calculated by using these mentioned cells. Various parallel prefix architectures

can be designed by varying this carry generation architecture.

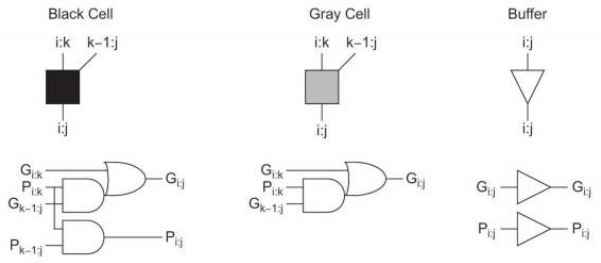

Fig.3 Black Cells, Gray Cells and Buffer cell used for carry generation stage

Final processing stage:

This stage generates the final sum based on the propagate and carry values produced in previous stages. Equation 5 depicts the Boolean expression.

$$s_i = p_i \oplus c_{i-1} \ldots\ldots\ldots (5)$$

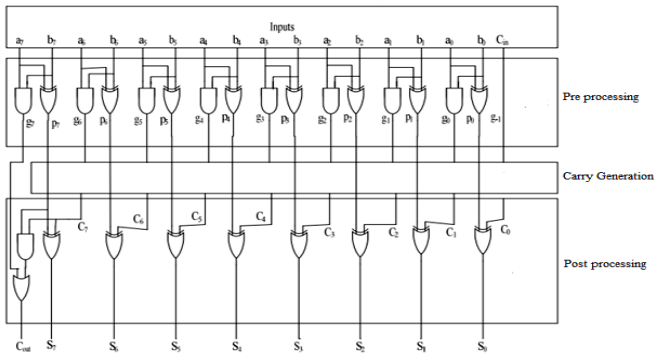

Fig.4 Overall architecture of parallel prefix adder.

Different applications like linear congruential generator, modified dual coupled linear congruential generator and coupled variable input linear congruential generator use three operand addition as the fundamental unit. Among the mentioned LCG, MDCLCG is considered as the most secure. Its security enhances with the increasing bit width. However as the operand size increase, the delay and area also increases linearly. Since MDCLCG is more secure compared to other LCG, It is widely used. Since three operand is the basic fundamental unit, if the performance of the three operand adder is improved in terms of power delay and area obviously the overall performance of MDCLCG.

Two parallel prefix adder with two input operands or one three-operand adder may be used to do three-operand binary addition Operation. In various cryptographic and Pseudo Random Bit generator (PRBG) methods CSA is most commonly recommended for performing three operand addition.

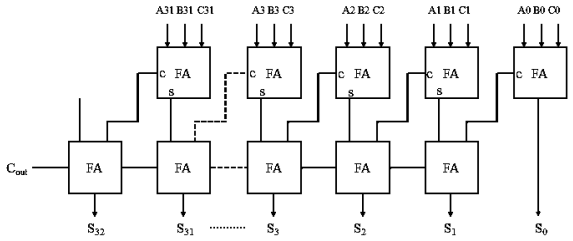

Fig.5 Block diagram of n-bit CSA

The modified dual CLCG (MDCLCG) and other cryptographic implementations on internet of things (IOT) -based hardware systems perform noticeably worse overall as a result of the delay in the Carry save adder (CSA) final stage.

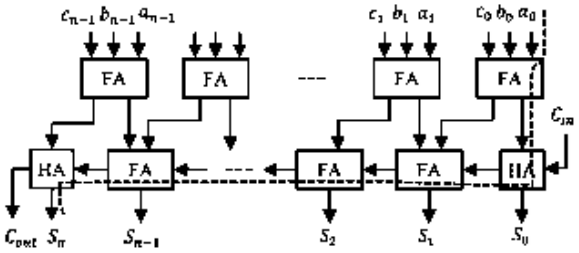

Fig.6 Carry propagation chain in CSA

In the carry save adder, all the intermediate carries are not propagated but the final stage carry is propagated. Hence in CSA final stage which performs carry propagation significantly impacts the performance.

The delay, which is CSA's primary flaw, is decreased but the area is increased by utilizing these parallel prefix adders. For the three-operand binary addition,

2 parallel prefix adders with two operands may be used. Here is a parallel prefix adder module with a three operand adder.

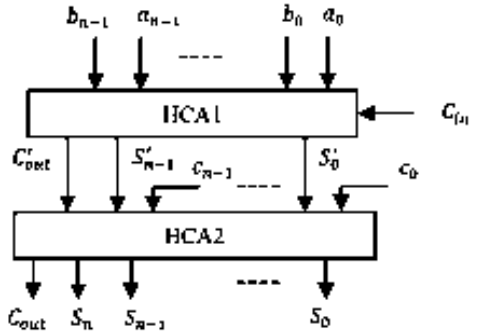

Fig.7 Block diagram of three operand adder using two parallel prefix adders

II. RELATED WORKS

Previously, carry save adder (CSA) is the most commonly used multi operand adder (three operand). In this adder, the although the carry is not propagated in the intermediate stages but in the final stage carry is propagated. Due to the carry propagation in the final stage of CSA, the delay is more. The multi (three in this case) operand adder is the fundamental component in the cryptography applications and pseudo random bit generation applications .since the delay of this three operand adder using CSA is high, it impacts the overall performance of the above mentioned applications. Hence it is not considered as the best choice. To overcome this delay drawback, parallel prefix adders is used. The conventional parallel prefix adders have 3 stages. The computations for all steps are carried out concurrently in parallel in parallel prefix adders, which improves performance with respect of delay. Two parallel prefix adders must be designed in order to construct a three operand adder. The region is the main problem, despite the fact that latency has enhanced. As a result, this method uses a new parallel architecture consisting of 4 phases to create a three operand adder. The third step, which consists of propagate and generation, uses a parallel prefix adder.

In the propagate generation step of this study, the Han Carlson adder is used. The Han Carlson adder block schematic is displayed below.

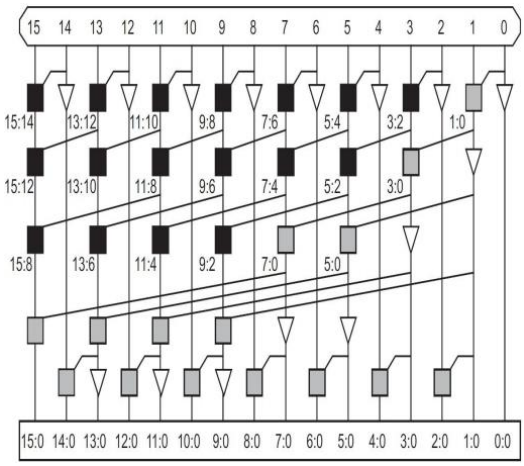

Fig: 8 16 bit Han Carlson adder

Although calculation of carry using this architecture, is done parallel, from the above figure, it can be observed that the number of black cell count is more. Hence the hardware efficiency of that adder is less. If the area of the basic adder is more, there will not be any space to add the extra logic or to incorporate more features in the chip. Hence designing an area efficient three operand adder is the need of the day.

III. IMPLEMENTATION

A novel parallel prefix architecture employing four stages is proposed. They are

1. Bitwise addition.
2. Base Logic.
3. Propagate and Generate logic (PG).
4. Final addition.

Previously CSA(carry save adder) is one of the most commonly used adder architecture for performing multi operand additions. But the problem of using CSA architecture in high performance applications. Because in the carry save adder architecture, the carry is being propagated in the final stage which significantly impacts the delay. To overcome this, we preferred using two parallel prefix adders to perform three operand addition. Although delay is improved

here the consumed is large. To further improve the performance in terms of area and speed a four stage parallel prefix architecture is used. In this, in the propagate and generate stage Han Carlson adder is used. Although the performance is enhanced area is more. To overcome this, the Han Carlson in the propagate and generate stage is replaced with the sklansky adder in our proposed method. The logical expressions for those stages are shown below.

Stage-1: Bitwise Addition:

$$s_i = a_i \oplus b_i \oplus c_i$$
$$cyi = a_i \cdot b_i + b_i \cdot c_i + c_i \cdot a_i$$

Stage-2: Base Logic:

$$G_{i:i} = G_i = s_i \cdot cy_{i-1}$$
$$G_{0:0} = G_0 = s_0 \cdot cin$$
$$P_{i:i} = P_i = s_i \oplus cy_{i-1}$$
$$P_{0:0} = P_0 = s_0 \oplus cin$$

Stage-3: PG (Generate and Propagate) Logic:

$$G_{i:j} = G_{i:k} + P_{i:K} \cdot G_{K-1:j},$$
$$P_{i:j} = P_{i:k} \cdot P_{K-1:j}$$

Stage-4: Final addition:

$$S_i = (P_i \oplus G_{i-1:0}), S_0 = P_0,$$
$$Cout = G_{n:0}$$

Figure 9 depicts block diagram of the novel three-operand binary adder. This novel adder architecture is shown in the figure below.

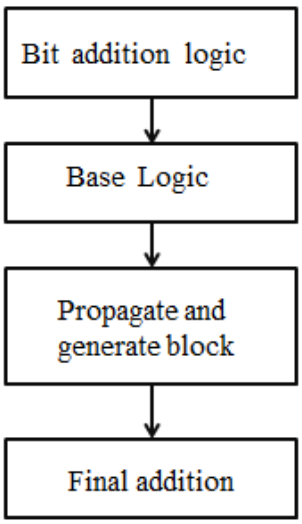

Fig.9(a) Block diagram of proposed three-operand adder

The bit wise addition uses input a,b,c to generate the partial sum and partial carries using 2 xor gates and 3 and gates. Consider an example. Let us assume a=1, b=1 and c=0; since s= a xor b xor c, the partial sum and carry will be s=0 and cy=1;

The generated partial sum and carries are given as input to base logic which is the second stage in the proposed architecture. Now this base logic generates propagate and generate intermediate outputs from the partial sum and carry signals generated at the first stage. Considering the same example that is considered for bit wise addition, it is observed that s=0,cy=1 in the above example. So now they are considered as inputs in the Base Logic. the outputs are shown below

$$P = s\ xor\ cy,$$
$$P = 0xor1 = 1$$
$$G = s\ and\ cy,$$
$$G = 0and1 = 0$$

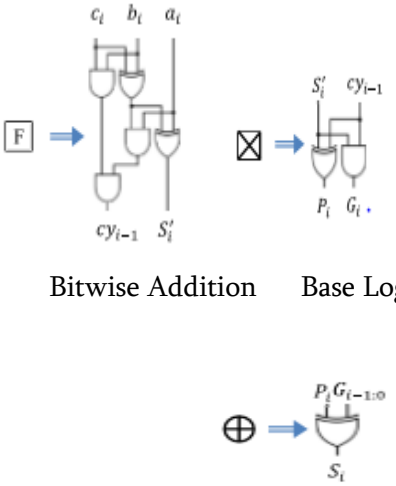

Fig.9(b)Gate level architectures for bitwise addition, Base Logic and Final addition.

Now in the third stage, using the propagate and generate signal produced at the output of second stage are given as input to the propagate and generate block. The propagate and generate block uses this signals and generate the final carries using this block. The propagate and generate block uses gray cells and black cells and buffers. Here in this stage, Han Carlson adder is previously used. Since it consumes more area, in our method, we are replacing this adder

with the sklansky adder to improve the area efficiency.

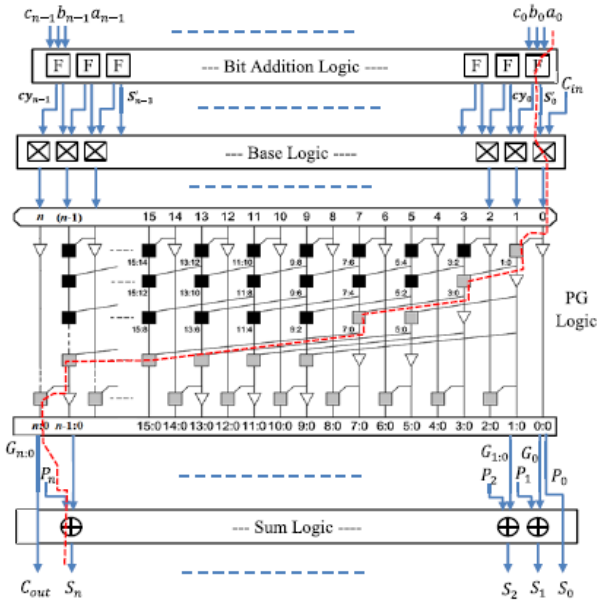

Fig.10 Proposed three-operand adder with Sklansky Adder

For any parallel prefix adder architecture, only gray cell, black cell and buffer cells will be present. For black cell, both propagate and generate need to be calculated. For gray cell, only generate is calculated. The buffer cell passes the same input to the output stage. Let us consider an example for performing operations of propagates and generate stage. The block diagram of 16 bit sklansky adder is shown below.

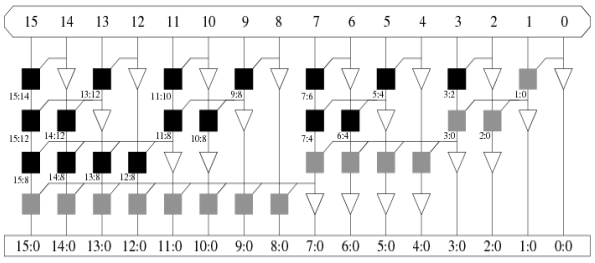

Fig.11 16 bit sklansky adder

From the figure3, it can be observed that the transistor count of the black cell is more compared to the gray cell. It can be clearly observed from figure 8 and 11 that the black cell count in Han Carlson adder when compared to the sklansky adder. Since black cell has larger gate count and Han Carlson adder has high number of black cells, the area occupied by the Han Carlson adder compared to sklansky adder. hence from this, it can be observed that Han Carlson adder is not as area efficient as sklansky adder.

Calculation procedure for black cell:

$$P = P_{n-1}\ and\ P_n$$

Suppose that P_{n-1}=1, and P_n =0,

The output will be P=0;

Now $G = (G_n\ and\ P_{n-1})\ or\ (P_n)$

Suppose that G_n =1, P_{n-1}= 0, P_n =1

The output will be G=1

Now these (P,G) values that are obtained are passed as inputs to either buffer or gray cell according to the structure. Here P_n and G_n are the present values, P_{n-1} and G_{n-1} are the previous state values.

Calculation procedure for gray cell:

Now $G = (G_n\ and\ P_{n-1})\ or\ (P_n)$

Suppose that G_n =1, P_{n-1}= 0, P_n =1

The output will be G=1.Now this G is passed as the output carry if there are no buffer cells or passed to through the buffer and is to calculate the carry bit. Buffer passes the same value that are given as input either from black cell or gray cell. In the proposed adder, Cin is considered for three-operand addition. To generate the final sum, the propagate signal from that block and carry signal from the previous block are being xored.

The final addition result is calculated using the expressions shown below.

$$S_i = (P_i \oplus G_{i-1:0}),$$
$$S_0 = P_0,$$
$$Cout = G_{n:0}$$

Let us consider an example for producing the final sum.Let us consider the initial carry input =0, and the propagate value that is generated from the first input bit a0=1, then the sum will be

$$S = p \text{ xor } cin$$
$$S = 1 \text{ xor } 0=1$$

IV. RESULTS AND DISCUSSIONS

The suggested three operand adder's block diagram, technology schematic, as well as simulation studies are shown following. Simulation results show

increased performance in terms of area for the proposed three operand adder, which is based on Sklansky.

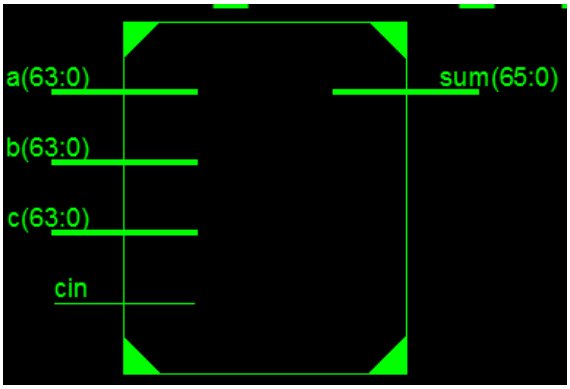

Fig 11 Block diagram of proposed three operand adder

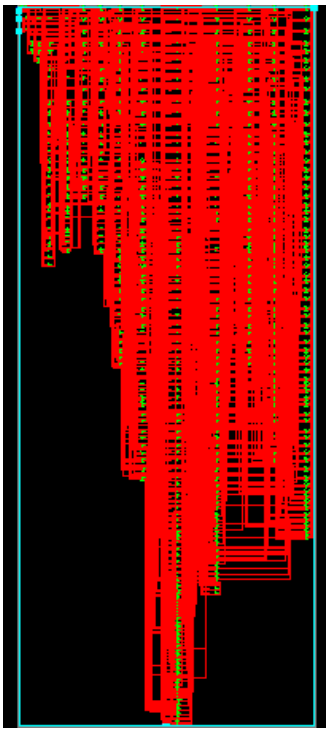

Fig 12: Technology schematic of proposed three operand adder

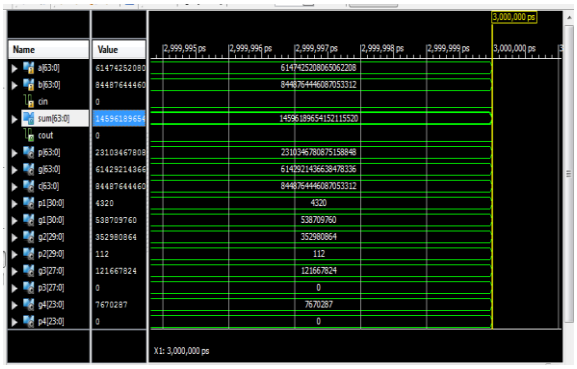

Fig 13: Simulation output of proposed Multiplier

Method.

	Area	Delay(ns)
Existing Han Carlson adder	368	7.5
Proposed sklansky adder	213	11.6

Table 1: Comparison table for Existing and Proposed methods

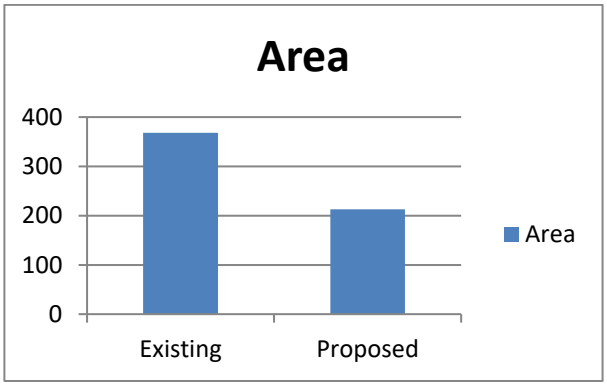

Fig.14 Comparison of area between existing and proposed multiplier

IV. CONCLUSION

In order to increase the area efficiency, a four step parallel prefix architecture in this study is devised, using Sklansky in the propagate and generate stages. Comparing the proposed adder to other parallel prefix three operand adders, such as the Han-Carlson Adder in the existing approach, the simulation results demonstrate the proposed adder's great hardware efficiency. The synthesis and simulation are verified by using Xilinx ISE tool.

V. REFERENCES

[1]. M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang, FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication

over prime field, IEEE Access, vol. 7, pp. 178811–178826, 2019.

[2]. Z. Liu, J. GroBschadl, Z. Hu, K. Jarvinen, H. Wang, and I. Verbauwhede, Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the Internet of Things, IEEE Trans. Comput., vol. 66, no. 5, pp. 773–785, May 2017.

[3]. Z. Liu, D. Liu, and X. Zou, An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor, IEEE Trans. Ind. Electron., vol. 64, no. 3, pp. 2353–2362, Mar. 2017.

[4]. B. Parhami, Computer Arithmetic: Algorithms and Hardware Design. New York, NY, USA: Oxford Univ. Press, 2000

[5]. P. L. Montgomery, Modular multiplication without trial division, Math. Comput., vol. 44, no. 170, pp. 519–521, Apr. 1985.

[6]. S. R. Kuang, K.-Y. Wu, and R.-Y. Lu, Low-cost high-performance VLSI architecture for montgomery modular multiplication, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 2, pp. 434–443, Feb. 2016.

[7]. S. R. Kuang, J.-P. Wang, K.-C. Chang, and H.-W. Hsu, Energy-efficient high-throughput montgomery modular multipliers for RSA cryptosystems, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 11, pp. 1999–2009, Nov. 2013.

[8]. S. S. Erdem, T. Yanik, and A. Celebi, A general digit-serial architecture for montgomery modular multiplication, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 5, pp. 1658–1668, May 2017.

Cite this article as :

N. Mounika, Dr. B. Lalitha, "VLSI Implementation of Ternary Operand Binary Addition Using Parallel Prefix Adder for Area Efficiency", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 10 Issue 4, pp. 247-254, July-August 2023.
Journal URL : https://ijsrst.com/IJSRST52310277