INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924

DESIGN AND IMPLEMENTATION OF AREA EFFICIENT TERNARY OPERAND ADDER FOR MODULAR ARITHMETIC

Nallipogula Mounika[1], A.Balachandra Reddy[2]
[1] M.tech scholar, Dept of ECE (VLSI), Sree Rama Engineering College, Tirupati, A.P., India
Email Id: mouni4056@gmail.com
[2] Associate Professor, Dept. of ECE, Sree Rama Engineering College, Tirupati, A.P., India
Email Id: balutest@gmail.com

Abstract**—** Parallel Prefix Adders have been established as the most efficient circuits for binary addition. Their regular structure and fast performance makes them particularly attractive for VLSI implementation. In many cryptography and pseudorandom bit generator (PRBG) methods, the fundamental functional unit is a three-operand binary adder that performs modular arithmetic. The carry save adder is a popular method for doing three-operand addition. The ripple-carry stage of the Carry save adder, on the other hand, results in a large propagation delay. Furthermore, a parallel prefix two-operand adder like Han-Carlson (HCA) can be utilized for three-operand addition, which significantly decreases the critical path time at the expense of extra hardware. In this method, for the purpose of three operand addition sklansky parallel prefix adder is used to improve performance of three operand adder in terms of Hardware efficiency.

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. So the overall performance of Modular arithmetic and cryptographic applications depends on the fundamental blocks used. Based on this fundamental module, the performance of the top module varies.

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 n-bit RCA is depicted in the figure below.

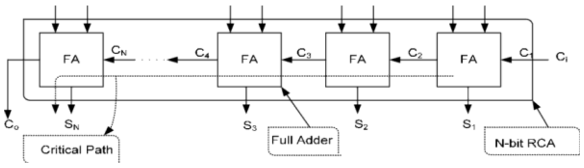


Fig.1 n-bit RCA

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.

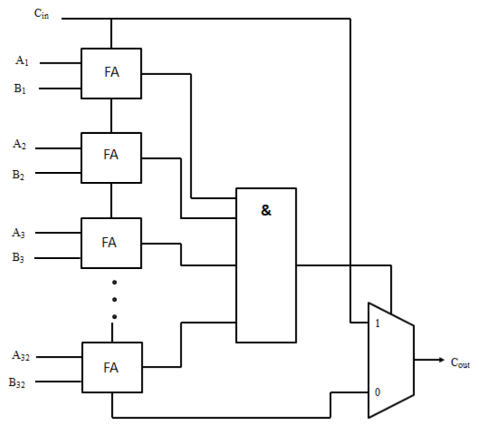


Fig.2 Logic diagram of Carry Skip Adder(CSKA)

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.

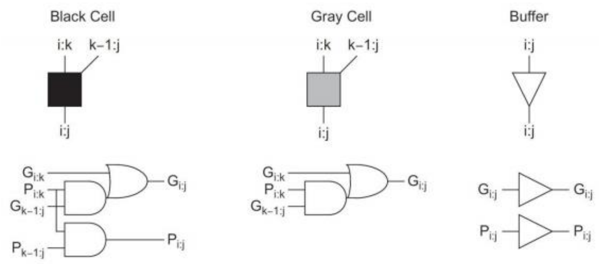


Fig.3 Black Cells, Gray Cells and Buffer cell used for carry generation stage

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.

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

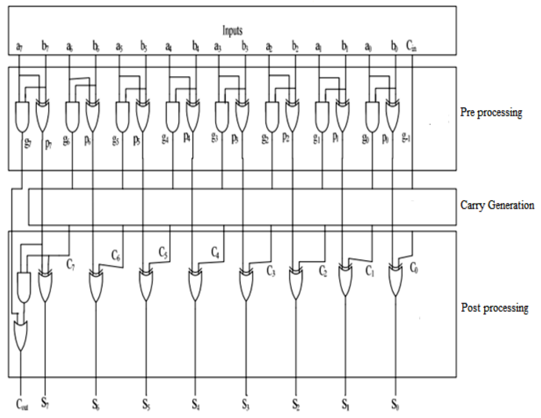

Inputs
a_7 b_7 a_6 b_6 a_5 b_5 a_4 b_4 a_3 b_3 a_2 b_2 a_1 b_1 a_0 b_0 C_{in}
Pre processing
Carry Generation
Post processing
C_7 C_6 C_5 C_4 C_3 C_2 C_1 C_0
C_{out} S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0

Fig.4 Overall architecture of parallel prefix adder.

ISSN 2454-9924

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.

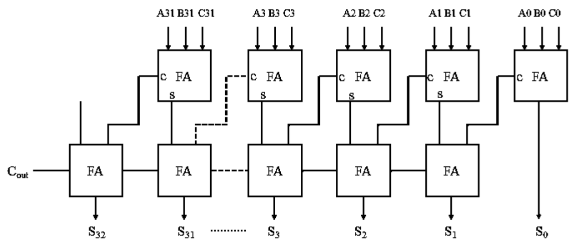


Fig.5 Block diagram of n-bit CSA

The delay in the final stage of Carry save adder (CSA) has a significant impact on the overall performance of the modified dual CLCG (MDCLCG) and other cryptographic implementations on internet of things (IOT) -based hardware systems.

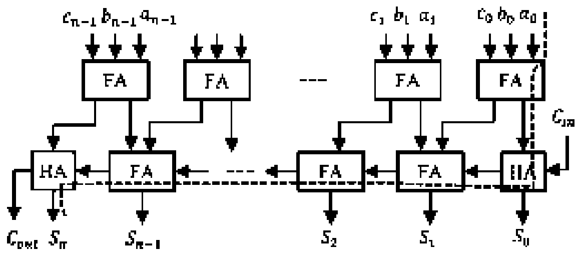

c_{n-1} b_{n-1} a_{n-1}
c_1 b_1 a_1
c_0 b_0 a_0
FA
HA
C_{in}
C_{out} S_n S_{n-1} S_2 S_1 S_0

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.

By using these parallel prefix adders, the delay which is the major drawback in CSA is reduced but increases the area. In order to perform the three-operand binary addition two operand two parallel prefix adders can be used. The block of three operand adder using parallel prefix adder is shown below.

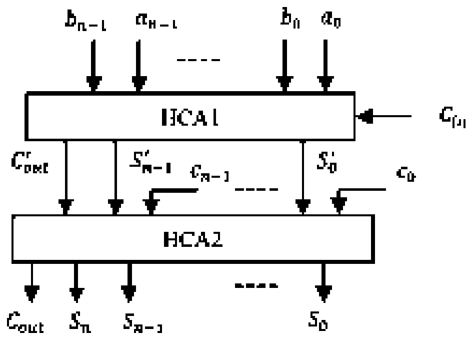

b_{n-1} a_{n-1} b_0 a_0
HCA1
C_{in}
C'_{out} S'_{n-1} c_{n-1} S'_0 c_0
HCA2
C_{out} S_n S_{n-1} S_0

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

ISSN 2454-9924

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. In parallel prefix adders, the computations of all the stages are performed parallel, hence the performance in terms of delay is improved. To design three operand adder two parallel prefix adders are needed. Although delay is improved, the area is the major drawback. so in this method, three operand adder is designed by using a new parallel architecture compromising of 4 stages . In the third stage (propagate and generate), a parallel prefix adder is used. In this work, in the propagate generate stage the Han Carlson adder is used. The block diagram of proposed Han Carlson adder is shown below.

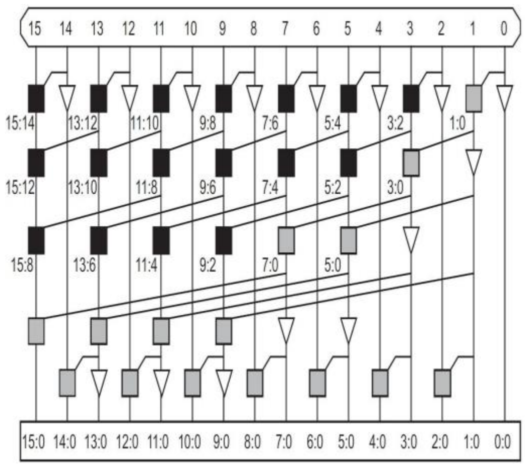


Fig: 8 16 bit Han Carlson adder

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.

II. IMPLEMENTATION

In this method a four-stage parallel prefix design is designed with sklansky adder in the third stage. The three operand adder perform addition in 4 stages namely Bitwise addition , base logic, propagate and generate logic and final sum stage CSA (carry save adder) was once one of the most widely used adder architectures for multi-operand adds. However, the use of CSA architecture in high-performance applications poses a challenge. Because the carry is propagated in the last step of the carry save adder design, the latency is greatly affected. To get around this, we choose to design the three operand addition using two parallel prefix adders. Despite the fact that the delay has been reduced, the hardware usage has increased significantly. To further improve the performance in terms of area and speed a four stage parallel prefix architecture 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:

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924

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

$$Cout = G_{n:0}$$

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

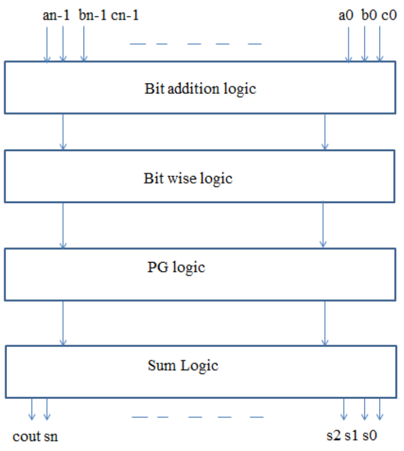


Fig.9(a) Block diagram of proposed three-operand adder

Using two xor gates and three and gates, the bit wise addition generates the partial sum and partial carry using inputs a,b,c. Consider the following scenario. Assume a=1, b=1, and c=0; the partial sum and carry will be s=0 and cy=1 since s= a xor b xor c.

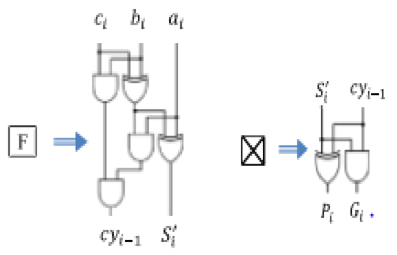
The generated partial sums and carries are fed into the proposed architecture's base logic, which is the second step. From the partial sum and carry signals created in the first stage, this base logic now creates propagate and generate intermediate outputs. In the same case as for bit wise addition, s=0,cy=1 is seen in the preceding example. As a result, they're now regarded inputs in the Base Logic. The results are displayed below.

$$P = s\ xor\ cy,$$

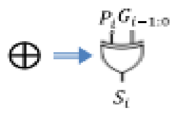
$$P = 0xor1 = 1$$

$$G = s\ and\ cy,$$

$$G = 0and1 = 0$$



Bitwise Addition Base Logic



Final addition logic

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

The propagate and generate signal created at the output of the second stage is now fed into the propagate and generate block in the third stage. This signals are used by the propagate and generate block, which generates the final carry. Gray and black cells, as well as buffers, are used in the propagate and generate block. The Han Carlson adder was previously utilized at this level. We are replacing this adder with the sklansky adder in our approach to increase the area efficiency because it uses more space..

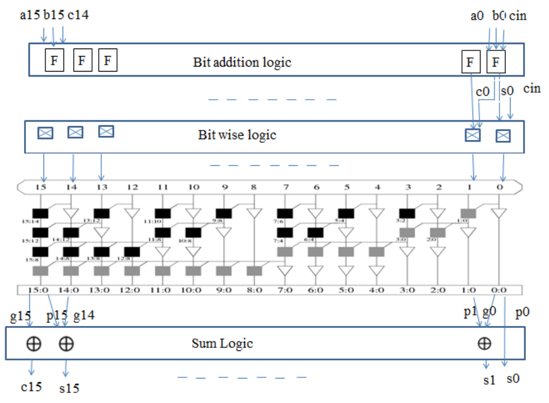


Fig.10 Proposed 16 bit three-operand adder with Sklansky Adder

Only gray, black, and buffer cells will be present in any parallel prefix adder design. Both propagate and generate must be computed for black cells. Only generate is computed for gray cells. The output stage receives the same input from the buffer cell. Consider the following scenario for performing propagates and generate stage activities. Below is a block schematic of a 16 bit sklansky adder.

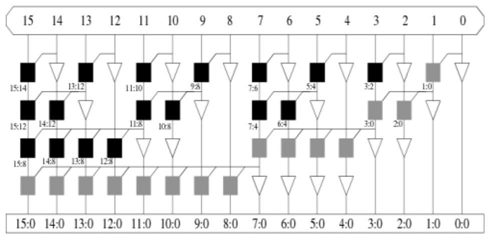


Fig.11 16 bit sklansky adder

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:

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

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

The output will be P=0;

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

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

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

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

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

III. RESULTS AND DISCUSSIONS

The block diagram of 64bit three operand adder, the technology schematic, and simulation results for the proposed three operand adder are shown below. Simulation results show that the proposed Sklansky based three operand adder shows better results in terms of area.

ISSN 2454-9924

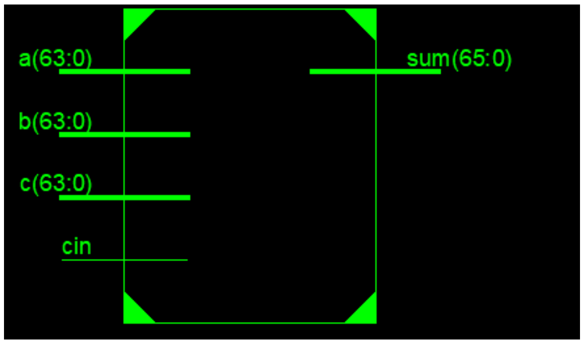


Fig 11 Block diagram of proposed three operand adder

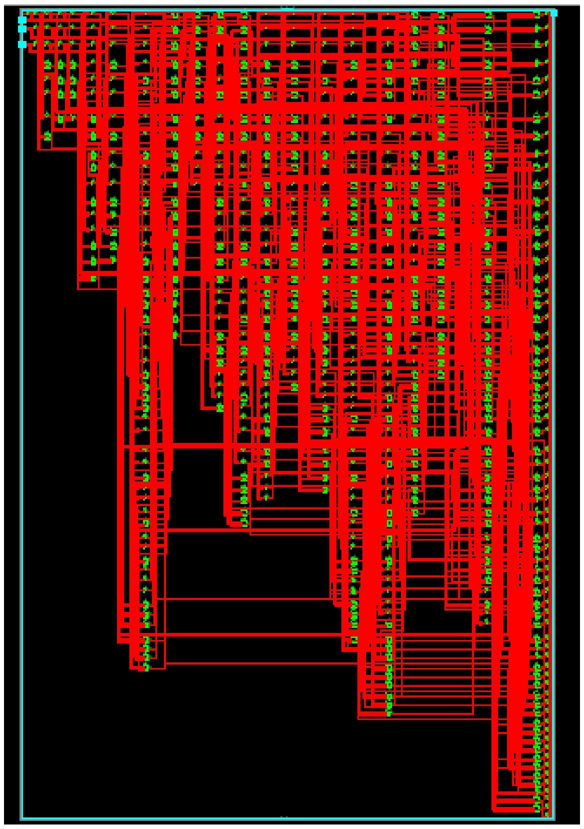


Fig 12: Technology schematic of Proposed three operand adder

Fig 13: Simulation output of proposed Multiplier Method.

	Area	Delay(ns)
Existing Han Carlson adder	462	24.7
Proposed sklansky adder	345	40.580

Table 1: Comparison table for Existing and Proposed methods

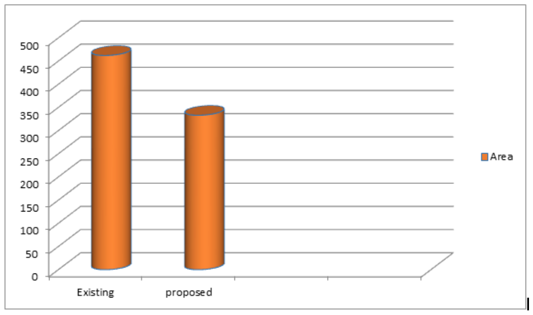


Fig.14 Comparison of area between existing and proposed multiplier

IV. CONCLUSION

In this paper, a four stage parallel prefix architecture using sklansky in its propagate and generate stage is designed to improve the area efficiency. Simulation results show that the proposed adder has high hardware efficiency when compared to other parallel prefix three operand adders like Han-Carlson Adder in existing method. 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.

ISSN 2454-9924

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.

[9] R. S. Katti and S. K. Srinivasan, “Efficient hardware implementation of a new pseudo-random bit sequence generator,” in Proc. IEEE Int. Symp. Circuits Syst., Taipei, Taiwan, May 2009, pp. 1393–1396.

[10] A. K. Panda and K. C. Ray, “Modified dual-CLCG method and its VLSI architecture for pseudorandom bit generation,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 3, pp. 989–1002, Mar. 2019.

[11] A. Kumar Panda and K. Chandra Ray, “A coupled variable input LCG method and its VLSI architecture for pseudorandom bit generation,” IEEE Trans. Instrum. Meas., vol. 69, no. 4, pp. 1011–1019, Apr. 2020.

[12] N. Weste and K. Eshraghian, Principles of CMOS VLSI Design—A Systems Perspective. Reading, MA, USA: Addison-Wesley, 1985.

[13] T. Kim, W. Jao, and S. Tjiang, “Circuit optimization using carry-saveadder cells,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 17, no. 10, pp. 974–984, Oct. 1998.

[14] A. Rezai and P. Keshavarzi, “High-throughput modular multiplication and exponentiation algorithms using multibit-scan–multibit-shift technique,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 9, pp. 1710–1719, Sep. 2015.