HIGH SPEED AREA EFFICIENT VLSI ARCHITECTURE OF THREE-OPERAND BINARY ADDER

Mr.Madgula Mahesh[1]., G.Santhoshi[2]., J.Rajeshwari[3]., N.Gayatri[4] ., R.Sushma[5]
1 Assistant Professor, Department of ECE., Malla Reddy College of Engineering for Women., Maisammaguda., Medchal., TS, India (madgulamahesh786@gmail.com)
2, 3, 4, 5 B.Tech ECE, (19RG1A0475, 19RG1A0477, 19RG1A04A5, 19RG1A04B7),
Malla Reddy College of Engineering for Women., Maisammaguda., Medchal., TS, India

Abstract—

When it comes to cryptography and PRBG algorithms, the modular arithmetic is often performed using a three-operand binary adder as the fundamental functional unit. A common method for doing three-operand addition is the carry-save adder (CS3A). The CS3A's large propagation delay, however, is due to its ripple-carry stage (n). Additionally, the critical path time may be drastically decreased by using a parallel prefix two-operand adder like Han-Carlson (HCA) for three-operand addition, although at the expense of extra hardware. Because of this, we propose a new high-speed and area-efficient adder architecture that uses pre-compute bitwise addition followed by carry prefix computation logic to perform the three-operand binary addition, thereby requiring significantly less area, low power, and drastically reducing the adder delay to O. (log2 n). The suggested architecture is generated using a commercially available 32nm CMOS technology library and implemented on an FPGA device for functional validation. The suggested adder was shown to be 3.12 times, 5.31 times, and 9.28 times quicker than the CS3A in post-synthesis findings for 32-bit, 64-bit, and 128-bit architectures, respectively. It outperforms the HC3A adder in terms of area, power dissipation, and latency. Also, compared to other methods for adding three operands, the suggested adder has the lowest average delay and peak delay (ADP and PDP, respectively).

Key words

Adders with three operands, the carry-save adder (CSA), the Han-Carlson adder (HCA), and modular arithmetic.

INTRODUCTION

It is essential to implement the cryptographic algorithms on hardware [1]-[3] in order to achieve optimum system performance while maintaining physical security. Different cryptographic algorithms typically use modular arithmetic's for the arithmetic operations, including modular exponentiation, modular multiplication, and modular addition [4]. This means that the efficiency with which the congruential modular arith metic operation is implemented directly affects the efficacy of the cryptographic method. The Mont Gnomery algorithm [5]-[7], whose critical operation is based on three-operand binary addition [6]-[8], is the most effective method for implementing modular multiplication and exponentiation. In the year 2020, the three-operand Manuscript was received on March 31; edited on July 18; and approved on August 4. Publication date is August 21, 2020, and the most recent revision was made on October 30, 2020. Associate Editor M. M. Kermani suggested I read this piece. Amit Kumar Panda is the author to contact for correspondence. Amit Kumar Panda may be reached at amit@hyderabad.bits-pilani.ac.in, and he works in the Department of EEE at BITS Pilani Hyderabad in Hyderabad, India. Rakesh Palisetty may be reached at rakeshp@kluniversity.in, and he works in the ECE department at KL Deemed to be University in Vaddeswaram, India (postal code: 522502). Kailash Chandra Ray may be reached at kcr@iitp.ac.in or the Department of Electrical Engineering, IIT Patna, Patna 801106, India. Some of the figures in this article have colour counterparts that may be seen at http://ieeexplore.ieee.org.

TRI-OPERAND BINARY ADDING METHODS

Congruential modular arithmetic architectures [5]-[8] and LCG-based PRBG techniques, such as CLCG [9], MDCLCG [10], and CVLCG [11], rely heavily on the three-operand binary addition as a key arithmetic operation. Two-operand adders or a single-stage three-operand adder may both be used to do this. When doing a three-operand binary addition, the carry-save adder (CSA) is the method of choice [9–14]. The sum of three operands is calculated in two steps. An initial full-adding array is used. Each complete adder takes in three binary digits, ai, bi, and ci, and simultaneously calculates the "carry" bit and the "sum" bit. The second step is a ripple-carry adder, which generates "sum" and "carry-out" signals of size n and one bit, respectively, after performing three-operand addition. In the ripple-carry stage, an n-count of complete adders are used to spread the "carry-out" signal. This means that the delay grows proportionally as the number of bits grows. In Fig. 1 we see the key route delay marked by a dashed line and the overall design of the three-operand carry-save adder. The evaluation reveals that the critical path delay is affected by the carry propagation delay of the ripple carry stage and proceeds as follows:

$$T_{CS3A} = (n+1)T_{FA} = 3T_X + 2nT_G$$

Similarly, the total area is evaluated as follows,

$$A_{CS3A} = 2nA_{FA} = 4nA_X + 6nA_G$$

Here, AG and TG indicate the area and propagation delay of basic 2-input gate (AND/OR/NAND/NOR) respectively. AX and TX indicate the area and propagation delay of 2-input XOR gate respectively. The major drawback of the CS3A is the larger critical path delay which increases with an increase of bit length. This critical propagation path delay influences the overall latency of the congruential modular arithmetic based cryptography and PRBG architectures, where three-operand adder is the primary component.

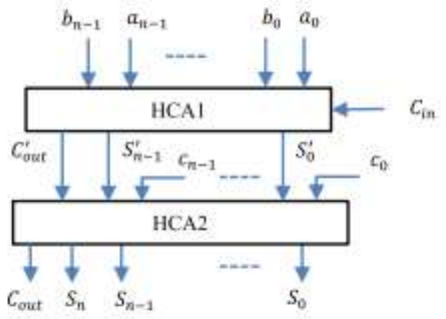


Fig. 1. Block level architecture of HCA-based three-operand adder (HC3A).

Hence, to shorten the critical path delay, two stages of parallel prefix two-operand adder can also be used. In literature, parallel prefix or logarithmic prefix adders are the fastest twooperand adder techniques [16], [17]. These adder techniques have six different topologies, such as Brent-Kung, Sklansky, Knowles, Ladner-Fischer, Kogge-Stone (KS) and Han-Carlson (HC). Among these, Han-Carlson is the fastest one when bit size increases (i.e. n > 16) [17]. In recent years, various such kind of parallel prefix two-operand adders, i.e., Ling [18], Jackson-Talwar [19], ultra-fast adder [20], hybrid PPFCSL [21] and hybrid Han-Carlson [22] are also discussed in the literature. The ultra-fast adder [20] is reported as the fastest one, and it is even faster than the Han-Carlson by three gates delay. However, it consumes comparatively twotimes large gate area than the Han-Carlson adder. On the other hand, the hybrid Han-Carlson adder [22] is designed with two Brent-Kung stages each at the beginning and the end, and with Kogge-Stone stages in the middle. This resultant a slightly higher delay (two gates delay) than the HanCarlson adder, with a 10% to 18% reduction in the gate complexity [22].

PROPOSED THREE-OPERAND ADDER ARCHITECTURE

This section presents a new adder technique and its VLSI architecture to perform the three-operand addition in modular arithmetic. The proposed adder technique is a parallel prefix adder. However, it has four-stage structures instead three-stage structures in prefix adder to compute the addition of three binary input operands such as bit-addition logic, base logic, PG (propagate and generate) logic and sum logic. The logical expression of all these four stages is defined as follows,

Stage-1: Bit Addition Logic

$$S'_i = a_i \oplus b_i \oplus c_i,$$
$$cy_i = a_i \cdot b_i + b_i \cdot c_i + c_i \cdot a_i$$

Stage-2: Base Logic:

$$G_{i:i} = G_i = S'_i \cdot cy_{i-1}, \quad G_{0:0} = G_0 = S'_0 \cdot C_{in}$$
$$P_{i:i} = P_i = S'_i \oplus cy_{i-1}, \quad P_{0:0} = P_0 = S'_0 \oplus C_{in}$$

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: Sum Logic:

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

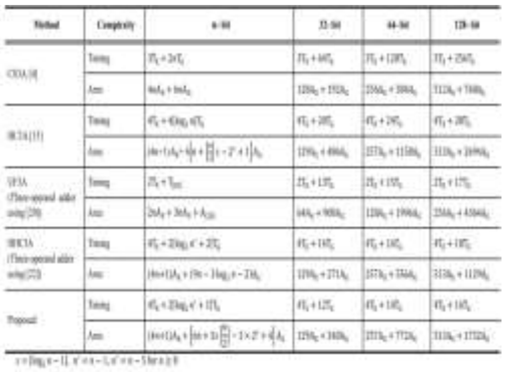
Figure 1 depicts the proposed VLSI design and internal structure of the three-operand binary adder. Three n-bit binary inputs are added in four steps using the new adder method. Bitwise addition of three n-bit binary input operands is carried out in the first stage (bit-addition logic), with the array of complete adders computing "sum (S I and "carry (cy I signals as shown in Fig. 1. (a). Bit-addition logic is shown in Fig. 1, and the corresponding logical formulas for calculating the sum (S I and carry (cy I signals are described in Stage-1 (b). After the first stage, the generate (Gi) and propagate (Pi) signals are computed using the "sum (S I bit of the current full adder and the "carry" bit of the full adder to its right (base logic). According to Fig. 1(a), the "squared saltire-cell" is used to represent the Gi and Pi signal computations, and there are n + 1 saltire-cells in the first stage of logic. Fig. 1(b) is a logic diagram depicting the saltire-cell, which is actualized by the following logical equation,

$$G_{i:i} = G_i = S'_i \cdot cy_{i-1};$$
$$P_{i:i} = P_i = S'_i \oplus cy_{i-1}$$

When adding three operands, the suggested adder method additionally takes into account the external carry-input signal (Cin). When calculating the G0 (S 0 Cin) in the first saltire-cell of the base logic, this extra carry-input signal (Cin) is used as an input. Third, a carry calculation step combining black and grey cell logics, "gener ate and propagate logic" (PG), is used to calculate the carry bit in advance. Fig. 3(b) shows a logical representation of a black and grey cell, which uses the following logical equation to determine the carry generation of Gi: j and the propagation of Pi: j signals:

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

TABLE I AREA AND TIMING COMPLEXITY OF THREE-OPERAND ADDER



Method	Complexity	n-bit	32-bit	64-bit	128-bit
CS3A [8]	Timing	$3T_X + 2nT_G$	$3T_X + 64T_G$	$3T_X + 128T_G$	$3T_X + 256T_G$
	Area	$4nA_X + 6nA_G$	$128A_X + 192A_G$	$256A_X + 384A_G$	$512A_X + 768A_G$
HC3A [15]	Timing	$4T_X + 4\log_2 n T_G$	$4T_X + 20T_G$	$4T_X + 24T_G$	$4T_X + 28T_G$
	Area	$(4n+1)A_X + 6\left[n + \left\lceil\frac{n}{2}\right\rceil s - 2^s + 1\right]A_G$	$129A_X + 496A_G$	$257A_X + 1158A_G$	$513A_X + 2694A_G$
UF3A (Three-operand adder using [20])	Timing	$2T_X + T_{UF2A}$	$2T_X + 13T_G$	$2T_X + 15T_G$	$2T_X + 17T_G$
	Area	$2nA_X + 3nA_G + A_{UF2A}$	$64A_X + 908A_G$	$128A_X + 1996A_G$	$256A_X + 4504A_G$
HHC3A (Three-operand adder using [22])	Timing	$4T_X + 2\lceil\log_2 n^* + 2\rceil T_G$	$4T_X + 14T_G$	$4T_X + 16T_G$	$4T_X + 18T_G$
	Area	$(4n+1)A_X + (9n - 3\log_2 n - 2)A_G$	$129A_X + 271A_G$	$257A_X + 556A_G$	$513A_X + 1129A_G$
Proposed	Timing	$4T_X + 2\lceil\log_2 n' + 1\rceil T_G$	$4T_X + 12T_G$	$4T_X + 14T_G$	$4T_X + 16T_G$
	Area	$(4n+1)A_X + \left[6n + 3s\left\lceil\frac{n}{2}\right\rceil - 3\times 2^s + 4\right]A_G$	$129A_X + 360A_G$	$257A_X + 772A_G$	$513A_X + 1732A_G$

$s = \lfloor\log_2 n - 1\rfloor$, $n' = n - 1$, $n^* = n - 5$ for $n \geq 8$

Given that the number of prefix calculation steps is (log2 n + 1), it is evident that the carry propagate chain is the primary cause of the adder's overall delay in the suggested implementation. The third phase, represented by sum logic, entails deriving the "sum (Si)" bits from the carry produce Gi: j and carry propagate Pi bits using the logical formula Si = Gi + Pi + Si (Pi Gi1:0). The carryout (Cout) is the output signal that is directly produced by the carry generate bit (Gn:0). Complexity in Terms of Both Space and Time The suggested adder consists of four separate phases: the bit-addition phase, the base phase, the PG phase, and the sum phase. In PG logic, the performance of the adder is very sensitive to the number of prefix stages. The maximum propagation gate delay (Tprop) of the proposed three-operand adder design may therefore be stated as follows.

$$\begin{aligned} T_{prop} &= T_{bitadd} + T_{base} + T_{PG} + T_{sum} \\ &\approx 2T_X + T_X + 2\lceil\log_2 n' + 1\rceil T_G + T_X \\ &\approx 4T_X + 2\lceil\log_2 n' + 1\rceil T_G \end{aligned}$$

Similarly, the hardware area (A_{prop}) of the proposed three-operand adder can be estimated as:

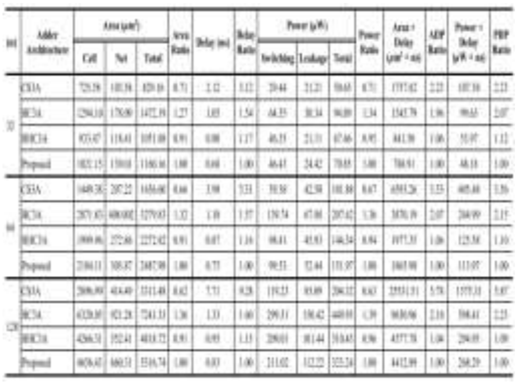
$$\begin{aligned} A_{prop} &= A_{bitadd} + A_{base} + A_{PG} + A_{sum} \\ &\approx (2nA_X + 3nA_G) + (n+1)(A_X + A_G) \\ &\quad + \left[2n + 3s\left\lceil\frac{n}{2}\right\rceil - 3\times 2^s + 3\right]A_G + nA_X \\ A_{prop} &\approx (4n+1)A_X + \left[6n + 3s\left\lceil\frac{n}{2}\right\rceil - 3\times 2^s + 4\right]A_G \end{aligned}$$

Here, s = log2 n − 1 and n = n − 1. In order to have a comprehensive assessment, the concept of hybrid Han-Carlson two-operand adder given in [22] is extended to develop a three-operand adder architecture. The first order timing-area complexity of this hybrid Han-Carlson three-operand adder (HHC3A) is evaluated as follows:

$$\begin{aligned} T_{HHC3A} &= T_{bitadd} + T_{base} + T_{PG} + T_{sum} \\ &\approx 2T_X + T_X + 2\lceil\log_2 n^* + 2\rceil T_G + T_X \\ &\approx 4T_X + 2\lceil\log_2 n^* + 2\rceil T_G \\ A_{HHC3A} &= A_{bitadd} + A_{base} + A_{PG} + A_{sum} \\ &\approx (2nA_X + 3nA_G) + (n+1)(A_X + A_G) \\ &\quad + (5n - 3\log_2 n - 3)A_G + nA_X \\ A_{HHC3A} &\approx (4n+1)A_X + (9n - 3\log_2 n - 2)A_G \end{aligned}$$

For n >= 8, n = n 5. Comparable work is done to construct the three-operand adder (UF3A) from the ultra fast two-operand adder (UF2A) presented in [20], and the area time complexity is calculated. Area and time complexity for the proposed three-operand adder design, as well as the CS3A, HC3A, HHC3A, and UF3A three-operand adder architectures, are summarised in Table I. The paper claims that the suggested adder design reduces critical latency by 43.4%, 50.2%, and 55.6% compared to the HC3A adder architecture for 32-, 64-, and 128-bit operand sizes, respectively. In addition, the suggested adder has a far lower critical latency than the CS3A adder does, by 81.2%, 89.1%, and 93.7%, respectively, for 32 bits, 64 bits, and 128 bits, respectively. A hybrid Han-Carlson based HHC3A adder is also noticed, and it is shown to use 10%-18% less gate area but be slower by a two gates delay compared to the suggested adder. However, UF3A, a three-operand adder based on ultra-fast two-operand technology, is the fastest of its kind. While it is twice as big in gate area as the HHC3A and projected three-operand adders, it is three gates quicker than the former.

TABLE II PHYSICAL SYNTHESIS RESULTS AND COMPARISONS OF THREE-OPERAND ADDER TECHNIQUES



Bit	Adder Architecture	Area (µm²)			Area Ratio	Delay (ns)	Delay Ratio	Power (µW)			Power Ratio	Area × Delay (µm² × ns)	ADP Ratio	Power × Delay (µW × ns)	PDP Ratio
		Cell	Net	Total				Switching	Leakage	Total					
32	CS3A	725.26	103.56	829.16	0.71	2.12	3.12	29.44	21.21	50.65	0.71	1757.42	2.22	107.38	2.23
	HC3A	1294.10	178.09	1472.19	1.27	1.05	1.54	64.35	30.34	94.69	1.34	1545.79	1.96	99.43	2.07
	HHC3A	975.47	116.41	1091.88	0.91	0.80	1.17	46.35	21.11	67.46	0.95	873.50	1.06	53.97	1.12
	Proposed	1025.15	139.01	1164.16	1.00	0.68	1.00	46.41	24.42	70.83	1.00	788.91	1.00	48.18	1.00
64	CS3A	1449.36	207.22	1656.60	0.66	3.98	5.53	59.58	42.50	101.88	0.67	6593.26	3.35	405.48	3.56
	HC3A	2871.43	408.00	3279.43	1.32	1.18	1.57	159.74	47.88	207.62	1.36	3870.39	2.07	244.99	2.15
	HHC3A	1999.86	272.66	2272.52	0.91	0.87	1.16	98.41	45.93	144.34	0.94	1977.10	1.06	125.58	1.10
	Proposed	2138.11	308.87	2487.98	1.00	0.75	1.00	99.53	52.44	151.97	1.00	1865.98	1.00	113.97	1.00
128	CS3A	2896.99	414.49	3311.48	0.62	7.71	9.28	119.23	85.09	204.32	0.63	25531.31	5.78	1575.31	5.87
	HC3A	6328.85	912.28	7241.13	1.36	1.33	1.60	299.31	150.42	449.73	1.39	9630.96	2.18	598.41	2.23
	HHC3A	4266.31	552.41	4818.72	0.91	0.95	1.13	209.01	101.44	310.45	0.96	4577.78	1.04	294.93	1.09
	Proposed	4656.47	660.53	5316.74	1.00	0.83	1.00	211.02	112.22	323.24	1.00	4412.89	1.00	268.29	1.00

DUAL-CLCG PERFORMANCE WITH THE PROPOSED THREE-OPERAND ADDER

For the quickest possible encryption and decryption, the hardware security of IoT applications requires a stream-cipher based high data rate, lightweight cryptography approach. The important component of stream-cipher based encryption/decryption is the key generator or pseudorandom bit generator (PRBG). Among the currently available PRBG algorithms, modified dual-CLCG (MDCLCG) is the most efficient option for stream-cipher based hardware security. However, the MDCLCG approach relies on the bit size of the congruential modulus to provide a high level of security. If n is less than 32 bits, it is secure and unpredictable in polynomial time [10]. As can be seen in Fig. 5, the MDLCG technique relies on LCG for its hardware design, with a three-operand modulo 2n adder serving as the fundamental computational arithmetic block. With four registers and multiplexers, two magnitude comparators, and four carry-save adders (CS3A) with modulo-2n operands, the MDCLCG architecture shown in [10] is built. As the bit size grows, the performance of the MDCLCG design is affected by the greater carry propagation gate delay in the CS3A adder. To evaluate the MDCLCG's performance, we will swap out the CS3A adder with the HHC3A and the suggested adder designs. The suggested adder's architecture is revised further to provide for the MDCLCG method's addition operation with three operands modulo 2n. For three-operand modulo-2n addition, we can thus assess the area (AP3OA) and time complexity (TP3OA) of the suggested adder design as follows.

$$A_{P3OA} \approx (4n-2)A_X + \left[6n + 3s\left\lceil \frac{n'}{2} \right\rceil - 3\times 2^s\right] A_G$$
$$T_{P3OA} \approx 4T_X + 2\left\lceil \log_2 n' + 1 \right\rceil T_G$$

Here, s = log2 n − 1 and n = n − 1. Similarly, the area (Amgc) and time (Tmgc) complexity of the magnitude comparator is evaluated in [10] which is further highlighted as follows,

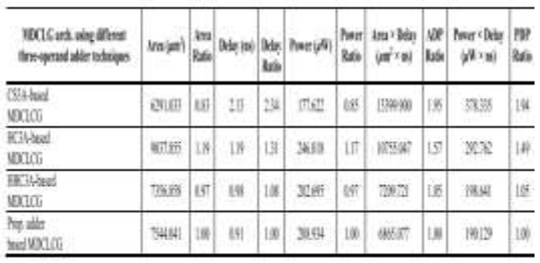
$$A_{mgc} \approx (n-1)\left[9A_G + 4A_N\right]$$
$$T_{mgc} \approx 4T_X + 4\left(\log_2 n\right) T_G$$

Therefore, the area and time complexity of the MDCLCG architecture using the proposed adder and the magnitude comparator can be evaluated as follows

$$\begin{aligned} T_{MDCLCG} &\approx T_{3oa} + T_{mx} \\ &\approx 4T_X + 2\left\lceil \log_2 n' + 2 \right\rceil T_G \\ A_{MDCLCG} &= 4\left(A_{mx} + A_{3oa} + A_{rg}\right) + 2A_{cmp} + A_X \\ &\approx (16n-7)A_X + +4(3n-2)A_N + 4nA_{FF} \\ &\quad + 2\left[27n + 6s\left\lceil \frac{n'}{2} \right\rceil - 6\times 2^s - 5\right] A_G \end{aligned}$$

Thus, the critical path delay of the proposed adder based MDCLCG architecture is drastically reduced in the order of O(log2 n), and hence, the overall latency is significantly improved. The 32- bit architecture of MDCLCG method based on the proposed three-operand adder is implemented using Verilog HDL and thereafter synthesized using Synopsys

TABLE III PHYSICAL SYNTHESIS RESULTS OF 32- BIT MODIFIED DUAL-CLCG ARCHITECTURE USING PROPOSED THREE-OPERAND ADDER TECHNIQUE



MDCLG arch. using different three-operand adder techniques	Area (μm^2)	Area Ratio	Delay (ns)	Delay Ratio	Power (μW)	Power Ratio	Area × Delay ($\mu m^2 \times ns$)	ADP Ratio	Power × Delay ($\mu W \times ns$)	PDP Ratio
CS3A-based MDCLCG	6291.033	0.83	2.13	2.34	177.622	0.85	13399.900	1.95	378.335	1.94
HC3A-based MDCLCG	9037.855	1.19	1.19	1.31	246.018	1.17	10755.047	1.57	292.762	1.49
HHC3A-based MDCLCG	7356.858	0.97	0.98	1.08	202.695	0.97	7209.721	1.05	198.641	1.05
Prop. adder based MDCLCG	7544.041	1.00	0.91	1.00	208.934	1.00	6865.077	1.00	190.129	1.00

Measurements of the chip's central region, its cycle time, and its power consumption may be extracted using a design compiler based on SAED's 32nm CMOS technology library. The HHC3A adder is then modified to create the 32-bit MDCLCG architecture for impartial evaluation, and the resulting design is implemented in Verilog HDL using the same coding language. The 32-bit MDCLCG architecture is developed with CS3A and HC3A in the same Verilog-HDL coding fashion. The same technology library and software environment is used to generate each of these designs. Maximum combinational gate delay, core area, power consumption, average design time, and peak design time are some of the physical synthesis characteristics provided in Table III. For 32-bit operand size, the proposed adder-based MDCLCG design has been reported to be 2.34 times quicker than the CS3A-based, HC3A-based, and HHC3A-based architectures. And much as how it uses 16.5% less space and 15.1% less power than the HC3A based MDCLCG design, so too does it have better thermal efficiency.

In addition, it has the lowest ADP and PDP of all MDCLCG designs, including those based on CS3A and HC3A. Reports indicate a 48.7 percentage point and 36.2 percentage point decrease in ADP, respectively, as compared to CS3A and HC3A-based MDCLCG frameworks. When compared to CS3A and HC3A based MDCLCG designs, it has been shown to reduce PDP by 49.7 and 35.1%, respectively. HHC3A-based MDCLCG architecture, on the other hand, uses much less space than HC3A-based and suggested adder-based MDCLCG designs. However, it is 7.1% slower than the adder-based MDCLG design that was suggested.

In addition, it shows ADP and PDP improvements of 5% and 4.5% over the suggested adder-based MDCLCG architecture, respectively. The next section details the use of Synopsys Verilog Compiler Simulation (VCE) and Design Vision Environ meant (DVE) in functionally validating the proposed adder and demonstrating its implementation in the MDCLCG approach.

Functional Verification and Simulation Following Synthesis

The part below uses the post-synthesis simulation data from the Synopsis VCS tool to verify the output results of the proposed adder and its application in the area of PRBG, namely, Modified dual CLCG (MDCLCG). DVE is a graphical interface tool used for troubleshooting and examining the waveforms. The suggested three-operand adder is used to create a 32-bit MDLCG architecture for the benefit of the reader.

utilising Synopsys VCS tool for simulation of Verilog-HDL code in order to verify functionality. Starting with the numbers (5183, 91356, 39771, 7392) for the initial seeds (x0), (y0), (p0), and (q0), the 32-bit MDCLCG architecture produces the sequence as follows: a1 = 65, b1 = 117, a2 = 16385, b2 = 221, a3 = 4097, b3 = 21359, a4 = 65537, b4 = 533, m =

$$\begin{aligned} x_1 &= (a_1 \times x_0 + b_1) = \left(2^{r_1} \times x_0 + x_0 + b_1\right) \bmod 2^{32} \\ &= (65 \times 5183 + 117) \bmod 2^{32} \\ &= \left(2^6 \times 5183 + 5183 + 117\right) \bmod 2^{32} \\ &= (331712 + 5183 + 117) \bmod 2^{32} \\ &= 337012 \bmod 2^{32} = 337012 \end{aligned}$$

In the above equation the three input modulo-2n addition is performed using the proposed three-operand adder technique. Similarly, the sequences x2, x3, can be computed as follows

$$X_i = \{337012, 21905897, 1423883422, 2358109331, \ldots\}$$

In the same way, other sequences such as Yi , Pi and Qi are also computed as follows,

$$\begin{aligned} Y_i &= \{1496868281, 1923524246, 474752883, 640215120, \ldots\} \\ P_i &= \{162963146, 1940099641, 2898752936, 606226711, \ldots\} \\ Q_i &= \{484450037, 1003823370, 1558127391, 2147362100, \ldots\} \end{aligned}$$

The sequences Bi and Ci in the MDCLCG architecture are generated by comparing Xi with Yi and Pi with Qi respectively using the magnitude comparator as follows,

$$\begin{aligned} B_i &= \{0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, \ldots\} \\ C_i &= \{0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, \ldots\} \end{aligned}$$

The pseudorandom bit sequence Zi is obtained by Bi ⊕ Ci as highlighted below

$$Z_i = \{0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, \ldots\}$$

depicts the post-synthesis simulation using the graphical interface tool DVE to verify the design functionality after the gate-level netlist of the 32-bit MDCLCG architecture has been built using the Synopsys VCS tool at 2ns clock constraint. Calculated values of Zi, xi, yi, and Bi are similar to those obtained by mathematical formulation performed while putting the MDCLCG architecture into practise using the suggested adder (see Fig. 1).

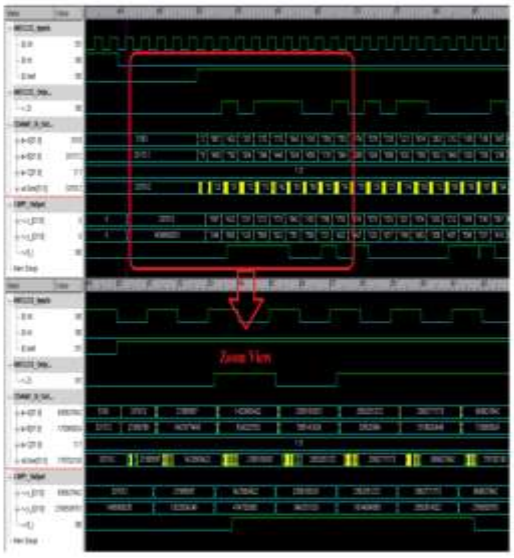


Fig. 3 . Post-synthesis simulation results of the 32-bit MDCLCG architecture.

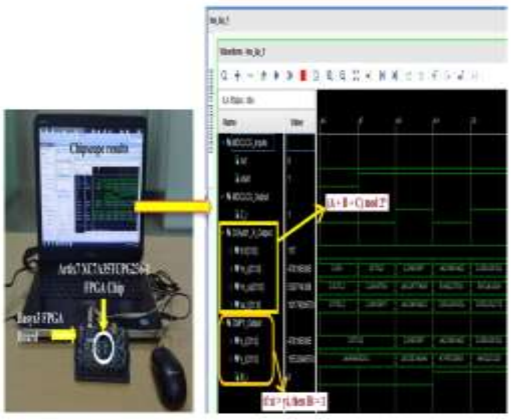


Fig. 4 . FPGA prototype in laboratory experimental setup with Real-time captured Chip scope results of the proposed 32- bit MDCLCG architecture.

In this part, the Synopsys VCS tool is used only for functional validation after synthesis. Therefore, in the next step, we build the design on FPGA and verify it on-chip via Xilinx Chip scope.

CONCLUSION

In this research, we offer a high-speed, area-efficient adder algorithm and VLSI design for performing the three-operand binary addition necessary for efficient calculation of modular arithmetic in cryptography and PRBG applications. In order to calculate the sum of three input operands, the suggested three-operand adder approach employs a parallel prefix adder with four-stage structures. The uniqueness of the suggested architecture is that it reduces critical path time, area-delay product (ADP), and power-delay product by streamlining the prefix calculation steps in PG logic and bit-addition logic (PDP). Hybrid Han-Carlson three-operand adder (HHC3A) topology is developed by extending the notion of hybrid Han-Carlson two-operand adder for the purpose of fair comparison. The HHC3A, HC3A, and CS3A are all implemented in Verilog HDL using the same coding approach as in the proposed adder design. Further, the core area, time, and power for various word sizes are synthesised using a commercially available 32nm CMOS technology library for all of these designs. Results from the physical synthesis show that the suggested adder design is 3–9 times quicker than the similar CS3A adder architecture. In addition, the suggested adder significantly outperforms the HC3A adder in terms of space usage, time path, and power dissipation. It's also important to note that the suggested adder has far lower ADP and PDP than existing methods for adding three operands together. It reduces ADP by 55.1%, 71.6%, and 82.7% compared to the CS3A adder for 32-, 64-, and 128-bit architectures, while it reduces PDP by 55.1%, 71.8%, and 82.9%.

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.

[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.

[15] A. K. Panda and K. C. Ray, "Design and FPGA prototype of 1024- bit Blum-Blum-Shub PRBG architecture," in Proc. IEEE Int. Conf. Inf. Commun. Signal Process. (ICICSP), Singapore, Sep. 2018, pp. 38–43.

[16] T. Han and D. A. Carlson, "Fast area-efficient VLSI adders," in Proc. IEEE 8th Symp. Comput. Arithmetic (ARITH), May 1987, pp. 49–56.

[17] D. L. Harris, "Parallel prefix networks that make tradeoffs between logic levels, fanout and wiring racks," U.S. Patent 0 225 706 A1, Nov. 11, 2004.

[18] H. Ling, "High-speed binary adder," IBM J. Res. Develop., vol. 25, no. 3, pp. 156–166, Mar. 1981.

[19] R. Jackson and S. Talwar, "High speed binary addition," in Proc. Conf. Rec. 38th Asilomar Conf. Signals, Syst. Comput., vol. 2. Pacific Grove, CA, USA, Nov. 2004, pp. 1350–1353.

[20] K. S. Pandey, D. K. B. N. Goel, and H. Shrimali, "An ultra-fast parallel prefix adder," in Proc. IEEE 26th Symp. Comput. Arithmetic (ARITH), Kyoto, Japan, Jun. 2019, pp. 125–134.