International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

High-Speed,Area-Efficient VLSI Architectureof Three-Operand Binary Adder

Dr. B. SUDHA RANI[1], M. K. DHANUSH[2], M. NANDINI[3], N. DINESH[4], M. YASWANTHREDDY[5], P. V. HARI PRIYA[6]

[1]Associate Professor, Dept. of ECE, S V College of Engineering, Tirupati, A.P, India.

[23456]B.Tech Students, Dept. of ECE, S V College of Engineering, Tirupati, A.P, India.

ABSTRACT

Three-operand binary adder is the basic functional unit to perform the modular arithmetic in various cryptography and pseudorandom bit generator (PRBG) algorithms and also used in many applications. Carry save adder (CS3A) is the widely used technique to perform the three-operand addition. In carry save adder at final stage uses ripple carry adder which will cause large critical path delay. Moreover, a parallel prefix two-operand adder such as Han-Carlson (HCA) can also be used for three-operand addition that significantly reduces the critical path delay with more area complexity. Hence, a new high-speed and area-efficient adder architecture is proposed using pre-compute bitwise addition followed by carry prefix computation logic to perform the three-operand binary addition that consumes substantially less area and less delay. When compare to existing design like three operand carry save adder and two operand based three operand Han-Carlson adder the proposed design consumes less area and less delay. The synthesis and simulation are verified by using Xilinx ISE 14.7 Tool.

Keywords: Three-operand adder, carry save adder (CSA), Han-Carlson adder (HCA), modular arithmetic.

1. INTRODUCTION

To achieve optimal system performance while maintaining physical security, it is necessary to implement the cryptography algorithms on hardware [1]–[3]. Modular arithmetic=such as modular exponentiation, modular multiplication and modular addition is frequently used for the arithmetic operations in various cryptography algorithms [4]. Therefore, the performance of the cryptography algorithm depends on the efficient implementation of the congruential modular arithmetic operation. The most efficient approach to implement

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

the modular multiplication and exponentiation is the Montgomery algorithm [5]–[7] whose critical operation is based binary addition is also a primary arithmetic operation in the linear congruential generator (LCG) based pseudo-random bit generators (PRBG) such as coupled LCG (CLCG) [9], modified dual-CLCG (MDCLCG) [10] and coupled variable input

LCG (CVLCG) [11]. Modified dual-CLCG (MDCLCG) is the most secure and highly random PRBG method among all the LCG-based and other existing PRBG methods. It is polynomial-time unpredictable and secure if for bit greater than 32-bits. Therefore, the security of the MDCLCG enhances with the increase of operand size. However, the area and critical path delay increases linearly since its hardware architecture consists of four three-operand modulo-2n adders, two comparators, four multiplexer's area in previous papers. Hence, the performance of the MDCLCG can be improved by the efficient implementation of the three-operand adder.

2. LITERATURE REVIEW

Parallel Prefix adders are arguably the most commonly used arithmetic units. They have been extensively investigated at architecture level, register transfer level (RTL), gate level, circuit level as well as layout level giving rise to a plethora of mathematical formulations, topologies and implementations. This paper contributes significantly to the understanding of these parallel prefix adders in a couple of ways. Firstly, it attempts to describe various such parallel prefix adders in elegant and consistent formulations. Secondly, a new family of parallel prefix adders is proposed at architecture level. The estimates of the area-throughput characteristics for an instance of this family are also presented. While the speeds achieved by this instance match those achieved by the state of the art adders, their area characteristics improved than existing but still it has more hardware complexity.

Summary of literature: In this paper various parallel prefix adder have been proposed like ling adder with different topology and Weinberger-Smith Recurrence which consist of less delay and more area.

3. EXISTING METHOD

1. Carry Save Adder:

The three-operand binary addition is one of the critical arithmetic operation in the congruential modular arithmetic architectures various existing methods and

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

LCG-based PRBG methods such as CLCG [9], MDCLCG [10] and CVLCG [11]. It can be implemented either by using two stages of two-operand adders or one stage of three-operand adder. Carry-save adder (CSA) is the commonly used technique to perform the three-operand binary addition [9]–[14]. It computes the addition of three operands in two stages. The first stage is the array of full adders. Each full adder computes "carry" bit and "sum" bit concurrently from three binary input $, bi\ and\ ci$. The second stage is the ripple-carry adder that computes the final n-bit size "sum" and one-bit size "carry-out" signals at the output of three-operand addition. The "carry-out" signal is propagated through the n number of full adders in the ripple-carry stage. Therefore, the delay increases linearly with the increase of bit length. The architecture of the three-operand carry-save adder is shown below figure. Where critical path delay is highlighted with a dashed line. It shows that the critical path delay depends on the carry propagation delay of ripple carry stage and is evaluated as follows.

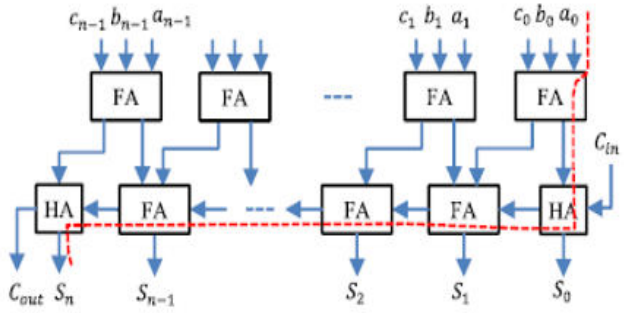


Fig.1: Three-operand carry-save adder (CS3A) showing critical path delay.

Drawback:

The major drawback of the CS3A is the larger critical path delay which increases with an increase of bit length.

2. Han-Carlson Adder:

To shorten the critical path delaya high speed parallel prefix Han-Carlson two-operand adder (HC2A) is further employed in the modified radix-2 Montgomery modular multiplier (MR2MMM). The three operand addition is performed using the two two-operand Han-Carlson adders in the modified radix-2 Montgomery modular algorithm. The HC2A based three-operand adder architecture is shown in below figure represent the characteristics diagram and first-order architecture of the Han-Carlson based three-operand adder.

The two stages of two-operand Han-Carlson adders (HC2A- 1 and HC2A-2) compute the addition of three operands. The detailed

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

architecture of two-operand Han-Carlson adder (HC2A). It has three stages such as base logic, PG (propagate and generate) logic and sum logic [16]. The logical diagram of base cell in base logic and sum cell in sum logic.

Drawback:

The HCA-based three-operand binary adder (HC3A) [15] greatly reduces the critical path delay in comparison with the three-operand carry-save binary adder. However, the disadvantages is area increases with increase of bit length of the adder.Therefore, to minimize this trade-off between area and delay, a new high-speed, area-efficient three-operand adder technique and its efficient VLSI architecture is proposed.

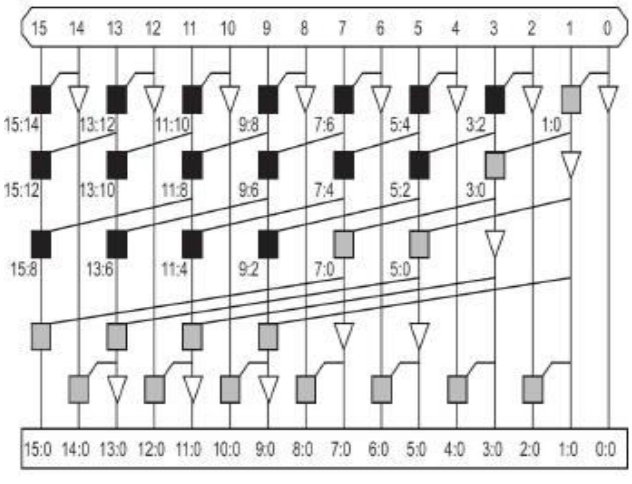


Fig.2: The carry generation structure of Han-Carlson adders

4. PROPOSED METHOD

Here we proposed 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 are 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:**

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

The proposed VLSI architecture of the three-operand binary adder and its internal structure is shown in above figure. The new

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

adder technique performs the addition of three n-bit binary inputs in four different stages. In the first stage (bit-addition logic), the bitwise addition of three n-bit binary input operands is performed with the array of full adders, and each full adder computes "sum $(S\,i\,)$" and "carry $(cyi\,)$" signals as highlighted in Fig. 3(a). The logical expressions for computing sum $(Si\,)$ and carry $(cyi\,)$ signals are defined in Stage-1, and the logicaldiagram of the bit-addition logic is shown in fig.

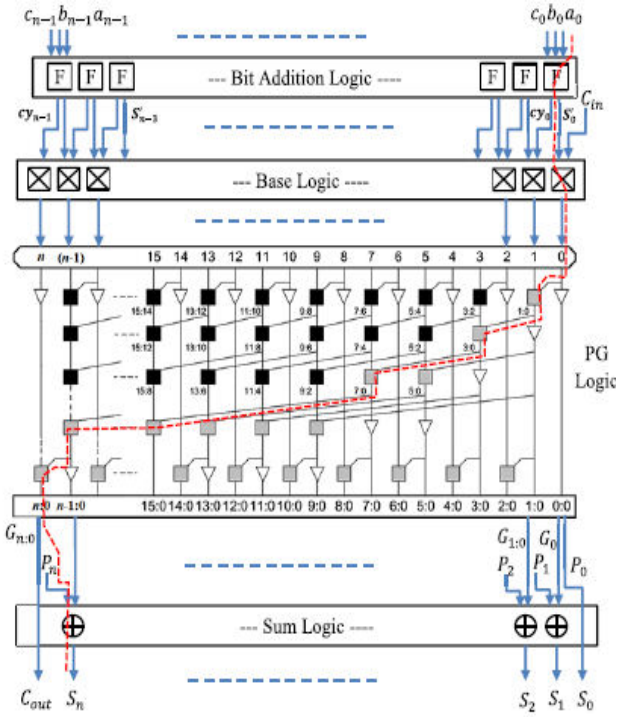


Fig.3: Proposed three-operand adder

In the first stage, the output signal "sum $(Si\,)$" bit of currentfull adder and the output signal "carry" bit of its right-adjacentfull adder are used together to compute the generate $Gi$and propagate $(Pi\,)$ signals in the second stage (base logic).The computation of Gi and Pi signals are represented bythe "squared saltire-cell" as shown in Fig. 3(a) and there are$n$+1 number of saltire-cells in the base logic stage. The logicdiagram of the saltire-cell is shown in Fig. 3(b), and it isrealized by the following logical expression,

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

The third stage is the carry computation stage called "generate and propagate logic" (PG) to pre-compute the carry bit and is the combination of black and grey cell logics. The logical diagram of black and grey cell is shown in Fig. 3(b) that computes the carry generate $Gi{:}\,j$ and propagate Pi: j signals with the following logical expression,

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

The number of prefix computation stages for the proposed adder is *(*log2 n+1*)*, and therefore, the critical path delay of the

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

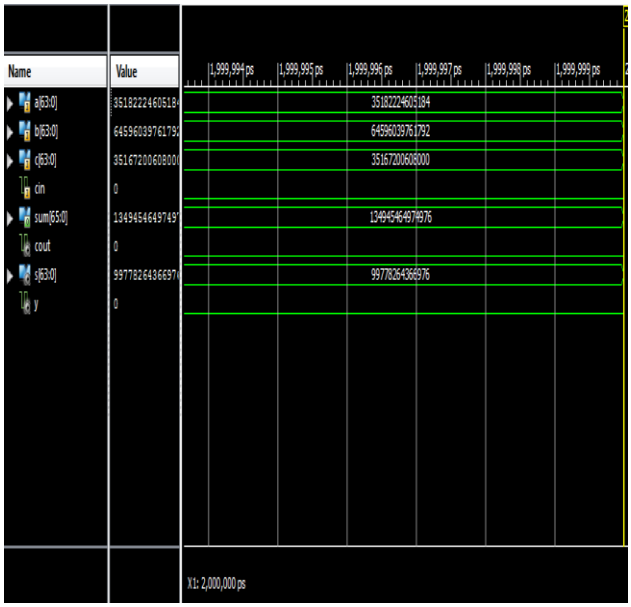
proposed adder is mainly influenced by this carry propagate chain. The final stage is represented as sum logic in which the "sum (Si)" bits are computed from the carry generate Gi: j and carry propagate Pi bits using the logical expression

$Si = (Pi \oplus Gi - 1:0)$. The carryout signal $(Cout)$ is directly obtained from the carry generate bit Gn:

5. METHODS OR TECHNIQUES USED

The proposed adder is implemented in Verilog HDL stimulated in Xilinx ISE Design Suite 14.5. Verilog HDL is a hardware description language. It is a language used for describing a digital system like network system. It is very easy for designing and debugging.

6. SIMULATIONRESULTS



Existing method

HC3A Adder

Area report:

```
Device utilization summary:
---------------------------

Selected Device : 7vx330tffg1157-2


Slice Logic Utilization:
 Number of Slice LUTs:                  444   out
    Number used as Logic:               444   out

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    444
   Number with an unused Flip Flop:     444   out
   Number with an unused LUT:             0   out
   Number of fully used LUT-FF pairs:     0   out
   Number of unique control sets:         0

IO Utilization:
 Number of IOs:                         259
 Number of bonded IOBs:                 259   out
```

Delay Report:

```
Data Path: a<1> to sum<65>
                                   Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             3   0.000   0.534  a_1_IBUF (a_1_IBUF)
   LUT4:I0->O            4   0.043   0.422  g1/gc1/G1 (g1/c<1>)
   LUT5:I3->O            7   0.043   0.529  g1/gc2/G1 (g1/c<3>)
   LUT5:I2->O            7   0.043   0.529  g1/gc4/G11 (g1/gc4/G1)
   LUT5:I2->O            5   0.043   0.518  g1/gc4/G1 (g1/c<7>)
   LUT6:I3->O            4   0.043   0.367  g1/gc8/G1 (g1/gc8/G)
   LUT5:I4->O            3   0.043   0.417  g1/gc8/G2 (g1/c<15>)
   LUT6:I4->O            2   0.043   0.410  g1/gc16/G11 (g1/gc16/G11)
   LUT4:I2->O            5   0.043   0.518  g1/gc16/G12 (g1/gc16/G1)
   LUT6:I3->O            4   0.043   0.367  g1/gc28/G111 (g1/gc28/G111)
   LUT5:I4->O            5   0.043   0.626  g1/gc28/G112 (g1/gc28/G11)
   LUT6:I1->O            4   0.043   0.422  g1/gc28/G (g1/c<55>)
   LUT5:I3->O            4   0.043   0.630  g1/Mxor_sum<63:1>_56_xo<0>1 (s<57>)
   LUT6:I0->O            3   0.043   0.625  g2/bc28/P1 (g2/p1<27>)
   LUT6:I0->O            1   0.043   0.613  g2/gc32/G6_SW0 (N96)
   LUT6:I0->O            1   0.043   0.405  g2/gc32/G6 (g2/gc32/G5)
   LUT5:I3->O            2   0.043   0.618  g2/gc32/G7 (cout)
   LUT6:I0->O            1   0.043   0.339  g3/carry1 (sum_65_OBUF)
   OBUF:I->O                 0.000          sum_65_OBUF (sum<65>)
  ----------------------------------------
  Total                      9.622ns (0.731ns logic, 8.891ns route)
                                     (7.6% logic, 92.4% route)
```

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

Proposed three Operand Adder:

Area Report:

```
Slice Logic Utilization:
 Number of Slice LUTs:                  368  out of  204000     0%
    Number used as Logic:               368  out of  204000     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    368
   Number with an unused Flip Flop:     368  out of    368   100%
   Number with an unused LUT:             0  out of    368     0%
   Number of fully used LUT-FF pairs:     0  out of    368     0%
   Number of unique control sets:         0

IO Utilization:
 Number of IOs:                         259
 Number of bonded IOBs:                 259  out of    600    43%
```

Delay Report:

```
Data Path: b<0> to sum<65>
                                        Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             3   0.000   0.507  b_0_IBUF (b_0_IBUF)
   LUT3:I0->O            2   0.043   0.608  g1/carry1 (c1<0>)
   LUT5:I0->O            4   0.043   0.630  pg1/gc1/G1 (x<1>)
   LUT6:I0->O            2   0.043   0.618  pg1/gc33/G1 (x<2>)
   LUT6:I0->O            4   0.043   0.512  pg1/gc2/G1 (x<3>)
   LUT6:I3->O            3   0.043   0.534  pg1/gc4/G1 (x<7>)
   LUT6:I2->O            1   0.043   0.522  pg1/gc8/G1 (pg1/gc8/G)
   LUT4:I0->O            3   0.043   0.417  pg1/gc8/G2 (x<15>)
   LUT6:I4->O            1   0.043   0.405  pg1/gc16/G1_SW0 (N4)
   LUT6:I4->O            5   0.043   0.428  pg1/gc16/G1 (x<23>)
   LUT6:I4->O            1   0.043   0.405  pg1/gc28/G111 (pg1/gc28/G11)
   LUT5:I3->O            5   0.043   0.518  pg1/gc28/G112 (x<31>)
   LUT6:I3->O            2   0.043   0.527  pg1/gc32/G4 (pg1/gc32/G3)
   LUT6:I2->O            1   0.043   0.339  h1/carry1 (sum_65_OBUF)
   OBUF:I->O                 0.000          sum_65_OBUF (sum<65>)
  ----------------------------------------
  Total                      7.531ns (0.559ns logic, 6.972ns route)
                                     (7.4% logic, 92.6% route)
```

COMPARISION OF EXISTING AND PROPOSED ADDERS IN TERMS OF AREA & DELAY:

	Area (in LUT's)	Delay(in ns)
CS3A	192	36.786
HC3A	444	9.622
Proposed Adder	368	7.531

6. CONCLUSION

In this we conclude that, a high-speed area-efficient adder technique and its VLSI architecture is proposed to perform the three operand binary addition in various cryptography algorithms. The proposed three-operand adder technique is a parallel prefix adder that uses four-stage structures to compute the addition of three input operands. The novelty of this proposed architecture is the reduction of delay and area in the prefix computation stages in PG logic and bit-addition logic that leads to an overall reduction in critical path delay. It also decrease the area complexity when compare to other parallel prefix three operand adders. Form the above comparison, the proposed three operand binary adder consists of less area and less delay when compare to existing in CS3A and HC3A three operand adder. The synthesis and simulation are verified by using Xilinx ISE tool.

7. FUTURE SCOPE

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. Hence we can modify the PG (propagate and generate) and replace with other carry propagation stages to generate the sum value. By modifying these design we get improvement in the parameters like area and delay.

8. 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.