A Further Optimized Mix Column Architecture Design for the Advanced Encryption Standard

Sudhir Rao Rupanagudi
WorldServe Education
Bengaluru, India
sudhir@worldserve.in

Varsha G Bhat
WorldServe Education
Bengaluru, India

Darshan G
APS College of Engineering
Bengaluru, India

Darshan S
APS College of Engineering
Bengaluru, India

Valliveti Vidya J
APS College of Engineering
Bengaluru, India

Padmavathi P
APS College of Engineering
Bengaluru, India

Shreya K. Gurikar
PES University
Bengaluru, India

Nivedita Sindhu
PES University
Bengaluru, India

***Abstract*— With the evolution of The Internet, there has been a huge spurt in online transactions and also an increase in sharing of private, confidential and sensitive information over the web. This in turn has increased the requirement of highly secure and swift methodologies to protect such data using modern cryptographic techniques such as the Advanced Encryption Standard (AES). In order to achieve the same, this paper discusses significant and novel modifications to the existing hardware architecture of the mix column step of the AES algorithm. By adopting these techniques, a speed efficiency of over 1.41 times was achieved as compared to previous algorithms. Moreover, in a VLSI perspective, an average area optimization of 3 times was also achieved. All experiments were conducted using the Xilinx Artix-7 series of FPGA.**

***Keywords*— AES; Cryptography; FPGA; Look-up Table; Mix Column; Multiplication; Network Security; Splitting method; VLSI; Vedic Mathematics**

I. Introduction

The dawn of the 21st century has brought along with it a huge amalgamation of technological breakthroughs and novel inventions. Most of these discoveries cater to easing the life of people in their day to day life activities which include communication [1], agriculture [2], transportation [3] and the likes. At the same time, it must be noted that most of these innovations are currently backed by complex algorithms generally setup and executed on servers situated elsewhere using the power of cloud computing and wireless communication. These systems can also be controlled wirelessly through virtual private networks and also remote desktop access solutions. Though this may seem advantageous, it comes along with it a major disadvantage as well. If not vigilant, these systems can unfortunately be also controlled by unauthorized personnel through hacking, obtaining credentials through phishing and other network security breaches [4]. Hence, highly efficient and quick methodologies to secure such systems, becomes the need of the hour.

The past few years has seen a large increase in different methodologies to provide network security to various installations, the most common of these methods being cryptography [5]. Whether it is social networking, online transactions, social security or controlling smart applications through the Internet of Things (IoT), cryptographic methods assist in prevention of data loss or cyber theft. The word Cryptography, which is derived from two Greek words 'krypto' and 'graphene' which mean hidden and writing respectively [6], is the science of protecting vital information, where the data is encrypted or scrambled in such a way that only the one who possesses the knowledge of deciphering it, can understand it [7].

Amongst the various cryptographic techniques which exist, the most popular methods include the Data Encryption Standard (DES), Triple DES (3DES) and the Advanced Encryption Standard (AES) [8]. Amid these, due to the advantages of the AES standard over the rest [9], it is considered as the most preferred algorithm. Currently, a further advanced variant of the AES – the Rijndael algorithm, is also popularly used, mainly due to its heightened level of security [10]. The four major steps involved in the Rijndael AES algorithm can be seen in Fig. 1 and have been elaborated in detail in [11].

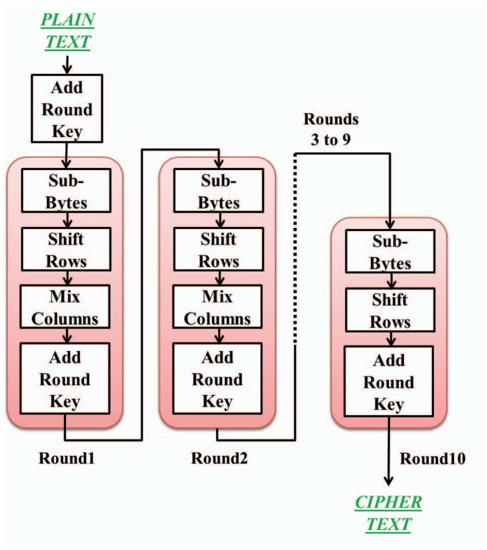

Fig. 1. Steps involved in the Rijndael AES algorithm

As shown in Fig. 1, it can be clearly seen that the input data is first subjected to the add-round key step and is then made to undergo 10 rounds of AES wherein the first nine rounds involve all the 4 steps mentioned above and the tenth round excludes the mix column step [10].

From the descriptions of each step stated in [11], it can be concluded that the most mathematically intense and time consuming step of the AES, is the mixed column stage [12]. The main reason for the same is the involvement of multiplication in the Galois-field domain with the Maximum Distance Separable (MDS) matrix shown in Fig. 2. It is also a well known fact that an increase in time to perform encryption can increase the probability of succeeding to 'break' the algorithm [13]. Similarly, with respect to decryption, a faster execution time relates to better back end performance [14]. Hence, a need arises to decrease the time required to perform the mix column operation of the AES and also reduce the vulnerability of the algorithm. This in turn forms the main motivation for this paper.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \qquad \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

(a) (b)

Fig. 2. The MDS matrices used for Mix-column (a) for encryption and (b) for decryption

In this paper, two methodologies, which can be used in conjunction with each other in order to reduce the execution time of the mix column step of the AES algorithm, have been described. The paper also focuses on area efficiency as well, keeping in mind a hardware VLSI implementation of the algorithm. Section 2 mentions the various existing methodologies which prevail for performing the mix column operation, and also elaborates on one of the most recent methods published in this regard. In Section 3, the two novel approaches have been discussed in detail. Section 4 gives a comparative study in terms of hardware area and speed of all the methods elaborated in this paper. The conclusion and future scope of this paper can be seen in section 5.

II. Existing Methodologies

As explained in the previous section, improvising the speed of execution of the AES algorithm can in turn lead to a minimal probability of cracking the same. This can be achieved by reducing the time taken to perform the mix column step of the Advanced Encryption Standard. The following section elaborates a few of the techniques already existing in literature to perform the same and also their disadvantages.

In [15], the authors describe a method to perform the mix column operation by multiplying the data obtained from the previous step of the AES algorithm, with an irreducible polynomial. This multiplication is performed over the Galois Field - GF 2^8 with the polynomial $x^8 + x^4 + x^3 + x + 1$ [11]. [16] - [19] also discuss a similar approach, though this time with a pipelined architecture. Though pipelining does reduce the time taken to obtain the product of the multiplication involved and also the throughput, the initial overhead and internal propagation delay of the XOR gates within the architecture can create a detrimental effect on the speed of execution. Also, in most of these approaches, a large number of XOR gates are utilized in turn increasing the area overhead of the design.

Another popular method to perform the mix column multiplication is by utilizing pre calculated multiplication look up tables (LUT). The authors in [20] created two LUT's – a logarithmic L table and an exponential E Table, which can be used together to find the product of two numbers in the Galois field. Since the values are pre-computed, this method has an upper edge with respect to calculation time in comparison with other methods. However, in an area-on-chip perspective the method proves to be futile due to its enormous memory requirement overhead [21].

In [22], the authors describe a method to reduce the on chip area requirement by the aforementioned method, by combining the L table and E tables into a single LUT. By doing so the area requirement reduces by 50%. However, the authors claim that this methodology would be viable only for encryption since the MDS matrix comprises of 2, 3 and 1 of which multiplication with 2 could be implemented with a shifter and that with 3 could be carried out with this new LUT. They further state that this method would prove to be pointless for decryption since there would be an increase in the total number of LUTs – one for each input 0xEh, 0xBh, 0xDh and 0x9h., thus increasing the on-chip area two fold.

The authors in [23] suggest performing multiplication by splitting the coefficients of the MDS matrix into powers of 2 using the distributive nature of the values and then carry out the multiplication operation using bit shifting. Though this method is area friendly, the utilization of barrel shifters introduces a combinational path delay which in turn compromises the total speed efficiency. Another aspect of major importance is that the output of the above mentioned operation does not compute the final product since the resultant is not yet limited to the GF 2^8. Hence there is a requirement of further processing to confine the output to an 8 bit number. In order to do so, the resultant obtained above is XORed with the generator polynomial mentioned previously which is now converted in hexadecimal format as 0x11Bh. The XORing is performed by bit shifting 0x11Bh to the left such that its most significant bit (MSB) now corresponds to that of the resultant. This process is repeated until the final product obtained is less than or equal to 8 bits [22]. Fig. 3 shows an example for more clarity. It can be clearly seen that the whole process requires an additional 4Δt seconds for a 12 bit number and the same would increase in case of a partial resultant with an increased bit-width. This further adds to the processing time proving disadvantageous in terms of speed efficiency.

$$\begin{array}{rr|l}
\text{FFF}_h\text{:} & \text{1111 1111 1111} & \\
\text{Step 1 - 11B}_h \ll 3\text{:} & \oplus\ \text{1000 1101 1000} & \Delta t \\
\hline
 & \text{0111 0010 0111} & \\
\text{Step 2 - 11B}_h \ll 2\text{:} & \oplus\ \text{100 0110 1100} & \Delta t \\
\hline
 & \text{011 0100 1011} & \\
\text{Step 3 - 11B}_h \ll 1\text{:} & \oplus\ \text{10 0011 0110} & \Delta t \\
\hline
 & \text{01 0111 1101} & \\
\text{Step 4 - 11B}_h \ll 0\text{:} & \oplus\ \text{1 0001 1011} & \Delta t \\
\hline
 & \text{0 0110 0110} & \\
\hline
\end{array}$$

Fig. 3. Example showing a 4Δt second requirement for reducing a 12 bit number to an 8 bit number

In order to overcome the disadvantage of the previous method due its use of barrel shifters, the authors in [24] came up with a methodology to perform multiplication using a modification of the ancient Vedic mathematic sutra or rule – the Urdhwa Tiryakbhyam Sutra which was discovered by Sri Bharti Krishna Tirthaji [25], between 1911 and 1918. The main advantage of this method was a reduction in hardware and also an improvisation in speed, in comparison with popular techniques of multiplication such as the Booth or Modified Booth approaches [26]. This was mainly due to the fact that multiplication was now reduced to a series of 16 equations mainly comprising of only XOR gates and AND gates. In continuation to this research, the authors in [22] further reduced these equations to 11 and in turn reduced the total number of XOR gates required to 21 and AND gates to around 32. Though promising, this method too suffers the same issue as the splitting method described previously, since the resultant obtained has to be further reduced to a Galois field - 2^8. This implies a further 4Δt second delay to obtain one multiplication output for a 12 bit partial result.

In the next section, the proposed methodology is described wherein an earnest attempt has been made to further reduce the hardware required in comparison to the methods described in this section. In addition to this, a novel methodology to reduce the total time duration for reduction of the result to Galois field by a mere Δt second has also been elaborated.

III. Proposed Methodologies

As mentioned in Section 1, the main aim of the work carried out in this paper is to improvise the speed efficiency of the AES algorithm and also to minimize the on-chip area occupied as well. In order to achieve the same, two major optimizations were identified and are elaborated below. The first deals with the optimization of the multiplication unit of the mix column step whilst the second method addresses improving the efficiency of the stage which reduces the multiplier output to Galois field - 2^8.

A. Performing the mix column operation utilizing traditional multiplication

As explained in the previous section and Section 1, the first step in the mix column operation is to multiply the input with the coefficients of the MDS matrix. It can be seen from Fig.2 that the values to be multiplied for encryption are 2, 3 and 1. Multiplication by 2 can be performed by bit shifting the input by one position to the left. There is no operation required for multiplication with 1, since the product is the input itself. With respect to multiplication by 3, if the traditional approach is to be followed, the steps along with the output obtained for an example can be seen in Fig. 4.

On close inspection it can be clearly seen that the final product can be obtained by simply XORing the input with a bit shifted version of the same. This can be mathematically represented by (1).

$$P_3 = A \oplus (A << 1) \qquad (1)$$

Where P_3 is the partial product obtained and A is the input.

```
     1111 1111   (A = FF_h)
   x 0000 0011   (03_h)
  -------------
     1111 1111   (A << 0) ⎤ Partial
  ⊕ 1 1111 1110   (A << 1) ⎦ Products
  -------------
   1 0000 0001   (P_3)
```

Fig. 4. Multiplication by 0x3h using the traditional approach

This further implies that, barring a bit shifter for multiplication by 2 and 9 XOR gates for multiplication with 3, no additional hardware is required for an input to be multiplied with a single row of the MDS encryption matrix viz. 2, 3 and 1. This forms a true improvisation in terms of hardware in comparison with the requirement of 14 XOR gates and 32 AND gates which were needed for performing mix column multiplication for the same input set as elaborated in [22].

With respect to decryption, the MDS values to be multiplied are 0xEh, 0xBh, 0xDh and 0x9h as shown in Fig. 2b. In the binary notation these values are represented as 1110b, 1011b, 1101b and 1001b. It can be clearly seen that in order to multiply the input with 0xEh through the traditional approach, the input has to be shifted once, twice and thrice and XORed together. This can be represented by (2) and can be best understood with the help of the example shown in Fig. 5.

$$P_E = (A << 3) \oplus (A << 2) \oplus (A << 1) \qquad (2)$$

Where P_E is the partial product obtained and A is the input.

```
          1111 1111   (A = FF_h)
        x 0000 1110   (0E_h)
  ------------------
          0000 0000   (A x 0)  ⎤
        1 1111 1110   (A << 1) ⎥  Partial
       11 1111 1100   (A << 2) ⎥  Products
  ⊕   111 1111 1000   (A << 3) ⎦
  ------------------
      101 1111 1010   (P_E)
```

Fig. 5. Multiplication by 0xEh using the traditional approach

Though the same approach can be followed for multiplication with 0xBh, 0xDh and 0x9h, it can be closely observed that all three values when represented in binary have a common MSB and LSB bit, which is 1. In other words, multiplication of the input with 0x9h can be first performed and the result of the same could be used for multiplication with 0xBh and 0xDh. Utilizing the same traditional approach applied in the above examples, multiplication with 0x9h can be performed by XORing the input with a three-time bit shifted version of the same. This can be seen in (3).

$$P_9 = A \oplus (A << 3) \qquad (3)$$

Where P_9 is the partial product obtained.

The result P_9 thus obtained can now be XORed with a single-bit left shifted version of the input for multiplication with 0xBh and with a two-bit left shifted version for obtaining the product with 0xDh. An example for multiplication with all 3 numbers can be seen in Fig. 6 and the mathematical representation of the same can be seen in (4) and (5).

$$P_B = P_9 \oplus (A << 1) \qquad (4)$$

$$P_D = P_9 \oplus (A << 2) \qquad (5)$$

As was with the case of encryption, decryption by adopting the traditional multiplication approach elaborated above also shows a promising reduction in hardware and also time consumed. In total, for performing a single round of multiplication with the MDS matrix values 0xEh, 0xBh, 0xDh and 0x9h, only 35 XOR gates are required as opposed to the Vedic approach which required 84 XOR gates and 128 AND gates [22].

```
     1111 1111    (A = FFh)
   x 0000 1001    (09h)
  ---------------
     1111 1111    (A << 0)             ┐
   0 0000 0000    ((A << 1) x 0)       │  Partial
  00 0000 0000    ((A << 2) x 0)       │  Products
⊕ 111 1111 1000   (A << 3)             ┘
  ---------------
  111 0000 0111   (P9)

A x 0Dh = (A << 2)⊕P9                A x 0Bh = (A << 1) ⊕P9
  ------------------                   ------------------
  11 1111 1100    (A << 2)              1 1111 1110    (A << 1)
⊕ 111 0000 0111   (P9)               ⊕ 111 0000 0111   (P9)
  ------------------                   ------------------
  100 1111 1011   (PD)                 110 1111 1001   (PB)
```

Fig. 6. Galois multiplication with 0x9h and utilizing the result for multiplication with 0xDh and 0xBh.

Similar to the splitting technique and the Vedic method explained in the previous section, the resultant obtained performing the above method also requires to be reduced to the Galois Field - 2^8. An efficient method to do the same has been elaborated next.

B. Reducing the product obtained to the Galois field - 2^8

As mentioned in the previous section and also in Section 1, the resultant products obtained after multiplication with the MDS matrix have to be confined to the Galois field - 2^8. The existing procedure to do the same has been elaborated in the previous section and it can be seen that a processing time of 4Δt seconds was required to obtain the final resultant in the case of a 12 bit product. This would prove to be highly inefficient in terms of speed and in order to overcome this issue a novel methodology has been elaborated below.

In the case of encryption, since the maximum value with which any input is to be multiplied is 0x3h, the maximum bit width an output could possibly attain is 9 bits. Hence in a worst case scenario wherein the product is 9 bits, it can be seen that a single time XORing with 0x11Bh would suffice to reduce the partial output. Therefore for encryption the maximum time for obtaining the final output is by default Δt seconds. However, this is not the case with its decryption counterpart.

With respect to decryption, the largest probable bit width of the product resulting from the multiplication operation would be around 12 bits. In such a case the total number of times 0x11Bh is to be left shifted and XORed is 4, which is equal to the total number of '1's preceding the 8th bit of the partial product. This can be better understood from the example already shown in Fig. 3. However, instead of shifting and XORing 0x11Bh at each step of the reduction process, this calculation can actually be pre-computed, stored and then finally XORed with the product obtained. Fig. 7a shows the pre-calculated value obtained in the case the first 4 bits of the partial product obtained are '1' and Fig. 7b shows the same XORed with the input utilized in Fig. 3.

It can be clearly seen that both results (Fig. 3 and Fig 7b) obtained coincide with each other. Also, with the help of pre-calculation the total time required to perform reduction to Galois field reduces to only Δt seconds. Table 1 shows the pre-computed values of 0x11Bh for a few combinations of the first 4 bits of the 12 bit product – B_{11} to B_8 where B represents the bit position. Though it might seem there is an area overhead due to a requirement of a look up table, this is in turn negligible in comparison with the overall bit shifters required and also the time delay expenditure.

```
  1000 1101 1000   (11Bh << 3)
   100 0110 1100   (11Bh << 2)            FFh:          1111 1111 1111  ┐
    10 0011 0110   (11Bh << 1)            PV of 11Bh:  ⊕1111 1001 1001  ┘ Δt
⊕    1 0001 1011   (11Bh)                              ---------------
  ---------------                         Final result:  0000 0110 0110
  1111 1001 1001   (PV of 11Bh)

              (a)                                        (b)
```

Fig. 7. (a) Pre-calculated value (PV) of the number to be XORed in the case the first four bits of the product is 1. (b) Resultant obtained after XORing (PV) with the product in just Δt seconds

TABLE I. A FEW PRECOMPUTED VALUES OF 11BH BASED ON MSB OF THE PRODUCT OBTAINED

Most significant bits of partial products (B_{11} - B_8)	Precomputed values of 11Bh to be XORed
0000	N/A
0110	65Ah
1011	BF5h
1111	F99h

The next section gives a comparison in terms of speed and area of the existing methodologies mentioned in the previous section, along with the combination of the above mentioned methods.

IV. RESULTS

Since most of the previous methodologies were designed on different hardware and also given that a fair and just comparison is required, the approaches mentioned in the previous two sections were redesigned, synthesized and implemented for a Xilinx Artix 7 (XA7A15TCSG324-2I) Series FPGA. Synthesis and implementation were performed for a timing constraint of 10ns, to ensure a proper comparison. Also, it was found through experimentation that previous evaluations, as elaborated in [22], [23] and [24], failed to take into account the time required for reduction of the product to Galois field - 2^8. Since this paper concentrates on a novel method to improvise the timing efficiency of the reduction process as well, the total speed calculated is a summation of both the time taken for multiplication as well as confining the same to an 8 bit number.

Further to this since [22] has already proved that the modified Vedic approach is better than its predecessors, comparison with those methods have not been performed. The synthesis, post synthesis and also place and route for all designs were performed using the Vivado 2017.2 version. All designs were compared in terms of speed of execution and on-chip area occupied. All values were tabulated and the same can be seen in Table 2.

It can be clearly seen from Table 2 that in terms of area, the proposed approaches for encryption and decryption occupy 2.78 and 2.98 times less area than the previous respective approaches using Vedic math. This can also be seen in a logic gate perspective as the total number of gates required for decryption is a mere 35 by the proposed method, as opposed to 212 which was required by the Vedic approach. In terms of speed, the new approaches for encryption and decryption are 1.6 times and 1.41 times faster than their respective Vedic approach counterparts. Thus, this establishes that both methods prove promising in terms of area and speed efficiency.

TABLE II. COMPARISON OF AREA AND SPEED OF VARIOUS MIX COLUMN METHODS

Approach	Area (in %)	Speed (in ns)
Modified Vedic Math method for encryption [22]	3.83	2.370
Modified Vedic Math method for decryption [22]	10.06	2.939
Proposed approach for encryption	1.38	1.478
Proposed approach for decryption	3.38	2.084

V. CONCLUSION AND FUTURE SCOPE

In this paper, two novel approaches to perform the Galois Field multiplication (2^8) for the mix column stage of AES, have been elaborated and explained. In both methods, a significant improvisation in area and speed were obtained. Similar types of methodologies could be explored for the other stages of AES as well. Application of these techniques, to other encryption methods such as visual cryptography, could open up a new paradigm into the world of research in the fields of network security and management.

REFERENCES

[1] S. R. Rupanagudi et al., "A low area & low power SOC design for the baseband demodulator of an indoor local positioning system," 2015 International Conference on Computing and Network Communications (CoCoNet), Trivandrum, 2015, pp. 689-695.

[2] S. R. Rupanagudi, B. S. Ranjani, P. Nagaraj and V. G. Bhat, "A cost effective tomato maturity grading system using image processing for farmers," 2014 International Conference on Contemporary Computing and Informatics (IC3I), Mysore, 2014, pp. 7-12.

[3] S. R. Rupanagudi et al., "A novel and secure methodology for keyless ignition and controlling an automobile using air gestures," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 1416-1422.

[4] I. M. Abdul Ghani Azmi, S. Zulhuda and S. P. Wigati Jarot, "Data breach on the critical information infrastructures: Lessons from the Wikileaks," Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), Kuala Lumpur, 2012, pp. 306-311.

[5] S. Chandra, S. Paira, S. S. Alam and G. Sanyal, "A comparative survey of Symmetric and Asymmetric Key Cryptography," 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), Hosur, 2014, pp. 83-93.

[6] M. Meena, A. Komathi, " A Study and Comparative Analysis of Cryptographic Algorithms for Various File Formats", International Journal of Science and Research (IJSR), Volume 5 Issue 8, August 2016, pp. 991 - 995

[7] Fischer, T. (2014, Dec). Private and Public Key Cryptography and Ransomware [Online]. Available: https://www.cisecurity.org/wp-content/uploads/2017/03/Cryptography.pdf

[8] Rodriguez-Henriquez, F., Saqib, N.A., Díaz Pérez, A., Koc, C.K. , "A Brief Introduction to Modern Cryptography", in Cryptographic Algorithms on Reconfigurable, Springer, Boston, MA, 2015, ch. 2, pp. 7-8.

[9] Aleisa, N. "A Comparison of the 3DES and AES Encryption Standards." International Journal of Security and Its Applications", 2015, vol. 9, issue. 7, pp- 241-246.

[10] W. Stallings, "Advanced Encryption Standard" in Cryptography and Network Security: Principles and Practices, 4th ed., Pearson Education, India, 2006, ch. 5, sec. 2, pp. 140-160.

[11] Announcing the Advanced Encryption Standard (AES), Federal Information Processing Standards, Publication 197, November 26, 2001.

[12] S. Arrag, A. Hamdoum, A. Tragha, and S. E. Khamlich, "Design and implementation a different architectures of MixColumns in FPGA", International Journal of VLSI design and Communication Systems, Vol. No. 3, Iss. No. 4, 2012, pp. 11-22.

[13] P. Parikh and S. Narkhede, "High performance implementation of mixing of column and inv mixing of column for AES on FPGA," 2016 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC), Chennai, 2016, pp. 174-179.

[14] M. Biglari, E. Qasemi and B. Pourmohseni, "Maestro: A high performance AES encryption/decryption system," The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013), Tehran, 2013, pp. 145-148.

[15] J. Daemen, V. Rijmen, The design of Rijndael AES The Advanced Encryption Standarad, Springer-Verlag, ISBN 3-540-42580-2.

[16] V. Gopi and E. Logashanmugam, "Design and analysis of nonlinear AES S-box and mix-column transformation with the pipelined architecture," 2013 International Conference on Current Trends in Engineering and Technology (ICCTET), Coimbatore, 2013, pp. 235-238.

[17] S. S. Chawla, S. Aggarwal, S. Kamal and N. Goel, "FPGA implementation of an optimized 8-bit AES architecture: A masked S-Box and pipelined approach," 2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, 2015, pp. 1-6.

[18] J. Balamurugan and E. Logashanmugam, "Low power and high speed AES using mix column transformation," 2013 International Conference on Current Trends in Engineering and Technology (ICCTET), Coimbatore, 2013, pp. 216-219.

[19] K. Sathya and V. Manimala, "Minimized architecture for Mix Column representation," 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, 2016, pp. 1166-1171.

[20] N. R. Wagner, "The Advanced Encryption Standard (AES)", in The Laws of Cryptography with Java Code, ch. VI, sec. 20, pp. 124-126

[21] Wu SY., Lu SC., Laih C.S. (2004) Design of AES Based on Dual Cipher and Composite Field. In: Okamoto T. (eds) Topics in Cryptology – CT-RSA 2004. CT-RSA 2004. Lecture Notes in Computer Science, vol 2964. Springer, Berlin, Heidelberg

[22] S. R. Rupanagudi et al., "Optimized area and speed architectures for the mix column operation of the advanced encryption standard," 2017 International Conference on Robotics, Automation and Sciences (ICORAS), Melaka, 2017, pp. 1-5.

[23] Hua Li and Z. Friggstad, "An efficient architecture for the AES mix columns operation," 2005 IEEE International Symposium on Circuits and Systems, 2005, pp. 4637-4640 Vol. 5.

[24] S. R. Huddar, S. R. Rupanagudi, R. Ravi, S. Yadav and S. Jain, "Novel architecture for inverse mix columns for AES using ancient Vedic Mathematics on FPGA," 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, 2013, pp. 1924-1929.

[25] M. Ramalatha, K. D. Dayalan, P. Dharani and S. D. Priya, "High speed energy efficient ALU design using Vedic multiplication techniques," 2009 International Conference on Advances in Computational Tools for Engineering Applications, Zouk Mosbeh, 2009, pp. 600-603.

[26] Huddar S.R., Rupanagudi S.R., Janardhan V., Mohan S., Sandya S. (2013) Area and Speed Efficient Arithmetic Logic Unit Design Using Ancient Vedic Mathematics on FPGA. In: Unnikrishnan S., Surve S., Bhoir D. (eds) Advances in Computing, Communication, and Control. Communications in Computer and Information Science, vol 361. Springer, Berlin, Heidelberg