2023 10th International Conference on Signal Processing and Integrated Networks (SPIN) | 978-1-6654-9099-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/SPIN57001.2023.10116674

FPGA IMPLEMENTATION OF A HIGH THROUGHPUT LOW POWER ADVANCED ENCRYPTION STANDARD (AES-128) CIPHER

Jaideep kala
Department of Electronics and Communication
Delhi Technological University
New Delhi, India
jaideepkala_2k21spd04@dtu.ac.in

Jeebananda Panda
Department of Electronics and Communication
Delhi Technological University
New Delhi, India
jpanda@dce.ac.in

Lavi Tanwar
Department of Electronics and Communication
Delhi Technological University
New Delhi, India
lavi.tanwar@dtu.ac.in

***Abstract*— Ensuring secure transmission and storage of digital information is critical for any organization to function properly. To address this issue, encryption algorithms are commonly used. Advanced Encryption Standard (AES) has been globally adopted as the mainstay cipher algorithm for securing transmission networks and storage devices due to its easy implementation and compatibility with both hardware and software applications. Another reason for its widespread popularity is its uncompromisable nature against existing brute force attacks, making it practically unbreakable on existing computing power. AES implementation for battery operated devices requires an algorithm with low power consumption and high-speed encryption/decryption of digital data. This paper proposes an FPGA implementation of a high throughput parallel pipelined 128-bit AES algorithm with a low power key expansion mechanism for iterative stages. A 128-bit symmetric key has been used for undertaking 10 rounds of transformations. All the encryption and decryption transformations are simulated using iterative design methodology in order to minimize hardware consumption. Xilinx Artix-7 FPGA device is used for hardware evaluation and Verilog HDL for programming. Simulation and synthesis task has been performed on Xilinx Vivado v2021.1 IDE. The results exhibit high-rate encryption of 68 Gb/s and low energy consumption of 7 pJ/bit.**

Keywords*— *AES, Encryption, FPGA, Low Power, Secure, Simulation, Throughput.

I. Introduction

Information security is a critical part of any communication system. At the heart of this security system is a cryptographic algorithm that manipulates the input message or plain text into cipher text with the help of a key [1]. An Encryption algorithm ensures that data is safely transmitted from sender to the receiver without any unauthorized manipulations or attacks. Advanced encryption standard (AES) is widely used encryption algorithm that has been adopted globally by governments and organizations for secure communication and data storage. Developed by National Institute of Standards and Technology in the year 2001, AES is a symmetric key cipher with a fixed block size of 128 bits. Meaning it uses the same key for encryption and decryption process and all operations are performed on 128 bits of data [2]. AES is the successor of Data encryption standard (DES) algorithm which was prone to hacking hence needed to be replaced by a much stronger algorithm. AES utilizes cipher key of sizes 128,192 and 256 bits for encryption hence making it one of the most secure and robust algorithms. It has proven to be safe from brute force attacks with not a single registered case of its failure against known attacks. Encryption techniques like AES are based on Shannon’s theory of confusion and diffusion (1945) [3]. Here confusion aims at complicating the relationship between cipher message and symmetric key whereas diffusion aims at dispersing the features of input message throughout the encrypted message. It can be efficiently used for both hardware and software applications One of the major limitations of AES algorithm is its high computational complexity which leads to high power consumption, making it less suitable for battery operated devices. Another area for improvement in AES is the speed of the algorithm [4]. Faster encryption and decryption processes are highly desirable for real time communication and data storage. For hardware implementation, AES algorithms that takes less area for implementation, has high throughput and low power consumption are preferable and is a topic of constant research [5].

II. Literature survey

There have been significant efforts in the past aimed at reducing the power consumption and increasing throughput of the AES algorithm. Bui et al. [6] proposed a hardware optimization strategy for AES implementation in ultra-low power IoT applications to provide multilevel security using different key sizes. Power and energy optimization has been performed for both data path and key expansion. This led to significant reduction in energy per bit to a value of 1 pJ/b at 10 MHz at 0.6 V and throughput of 28 Mb/s. Duran et al. [7] proposed an AES-128/256 S-box acceleration scheme which uses a custom S-box unit connected as a logic unit. All S-box calculations were performed using pipelined and pure combinational approach resulting in lower memory access and lower energy consumption of 9.7 pJ/bit. A large portion of the energy consumed in an AES circuit is during the substitution process, hence S-box architecture plays a crucial role. Morioka et al. [8] proposed a low power S-box architecture in which signal arrival time at gates are very close if their depth from main input is identical. This led to a minimalistic power consumption of 29 µW at 10 MHz using 0.13 µm CMOS technology. In recent studies, pipelining of AES architecture has proven to significantly increase the throughput of the encryption system and accompanied power reduction. Chellappa et al. [9] proposed a fully pipelined 256-bit AES design with pulse clocked latches connecting the pipeline stages that can be made transparent when in use resulting a 7.6% decrease in energy. This design could deliver 64 Gb/s encryption when fabricated on 90 nm technology. Oukili et al. [10] presented a 5-stage pipeline S-box design to increase the maximum speed and frequency of the AES system. S-box transformations plays a crucial role in the complexity of AES algorithm therefore parallel processing using pipelined stages helped the proposed method to achieve a throughput of 79 Gbps. Kshirsagar et al. [11] proposed interchanging of byte substitution and shift rows operations in the AES implementation which helped to streamline the processing of 16 data blocks into 4 parallel blocks of data. This led to significant reduction in hardware area consumption by 56%

and increasing the throughput by 4.25%. Contributions of the proposed algorithm are:

- This paper intends to incorporate power saving strategies and obtaining high throughput using parallel pipelined 128-bit AES.
- Maintaining design simplicity.
- Ease of implementation.
- Low hardware area consumption.

III. Methodology

AES algorithm makes use of iterative process to obscure the relationship between the key and the cipher text. Each iteration step performs fixed number of substitutions and permutations to encrypt the input message. The number of iterations depends on the size of the cipher key used. Key can be of size 128,192 and 256 bits and subsequently the number of iterations are 10,12 or 14 respectively. Similarly, during the decryption process the encrypted message is passed through these iterative steps in reverse order to obtain the original input message. Each of these steps consists of four processes named (1) Substitute bytes (2) Shift rows (3) Mix Column (4) Add round keys the details of which has been discussed in brief in this section [12]. Before the first iteration is performed, a pre round transformation takes place along with key expansion in which the size of cipher key is extended from 4 words (in case of 128 bits key) to 44 words, where each word is of size 4 bytes. Four words of this expanded key is then supplied to each of the 10 iterative rounds as well as to the pre-round transformation. All operations on the data are performed on a block size of 128 bits in the AES irrespective of the key size, hence the input message is divided into 4x4 matrices where each element in the matrix is of size 1 byte. After each operation the results are stored in a 4x4 intermediate state matrix on which further operations are performed [13]. Figure 1 shows the flow of input data through N rounds of transformation where each round is provided with extended key. The cipher key is first XORed with the input data matrix in the pre-round transformation to produce the state matrix which then acts as the input for subsequent rounds.

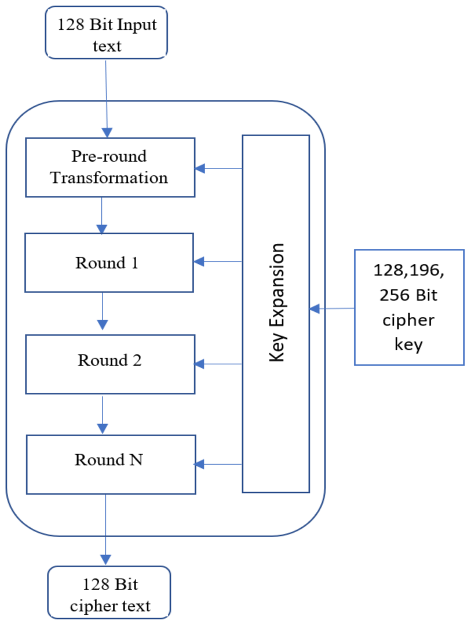


Fig.1. Block diagram of N rounds 128-bit AES algorithm.

A. Key Expansion

The cipher key is supplied by the user as a plain text which is first converted to hexadecimal form. Consider a 128 bit key arranged in the form of a 4x4 matrix with each column representing a 4 byte word as shown in Figure 2. This 4-word representation of the original key is expanded to 44 words and supplied to the pre-round transformation stage along with 10 transformation rounds [14]. The first 4 words of the expanded key representation is the original key itself which is XORed with the input message (in hex) and the intermediate state matrix is passed on to round 1 as input.

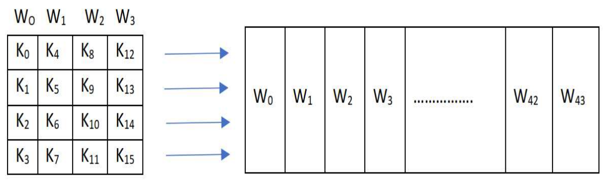


Fig.2. Key Expansion process for 128-bit symmetric key.

Figure 3 depicts the process of obtaining the next four words (W_4-W_7) from the first 4 words (W_0-W_3) of the original cipher key using the "g" function.

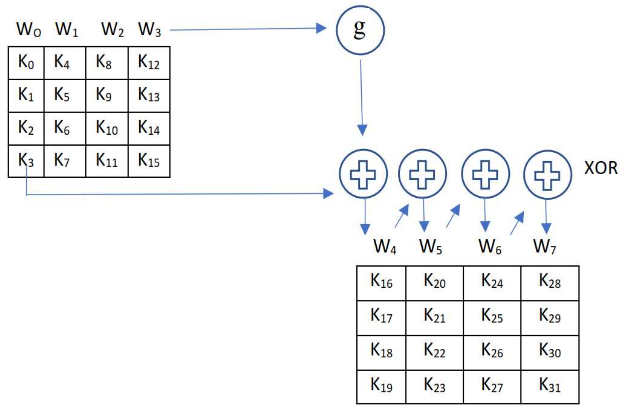


Fig.3. Key expansion operation for obtaining next 4 key words.

Mathematically this can be defined as:

$$W_4 = W_0 \oplus g(W_3) \quad (1)$$

$$W_5 = W_4 \oplus W_1 \quad (2)$$

$$W_6 = W_5 \oplus W_2 \quad (3)$$

$$W_7 = W_6 \oplus W_3 \quad (4)$$

Here $g(W_3)$ is obtained by performing a 3-step process, one-byte circular left shift of the word W_3 to get X_1 and then performing a byte substitution on each byte of X_1 using S-box to get Y_1.

Finally,

$$g(W_3) = Y_1 \oplus R_{CON}[j] \quad (5)$$

Here $R_{CON}[j]$ is the round constant described for each iteration round as shown in Table 1.

Table 1. Round constant table for 10 rounds of transformation in AES.

R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Figure 4 depicts the four processes in round 1 to 9 of the 128-bit AES algorithm with a cipher key size of 128 bit. The final round R10 has only 3 internal processes as the Mix column operation is excluded from it.

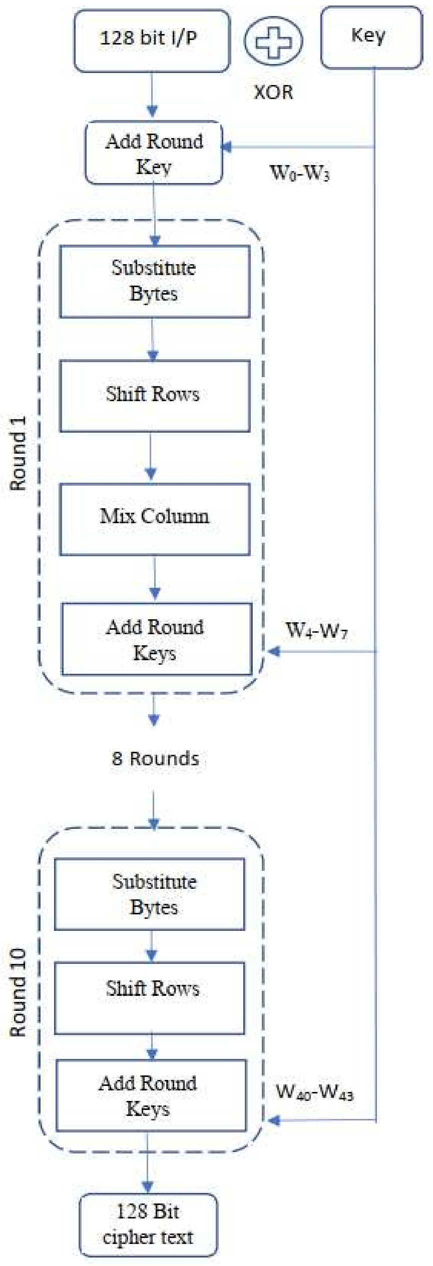
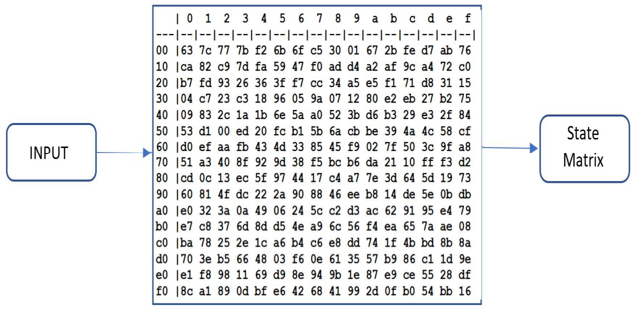


Fig.4. XOR of expanded key with each round of AES.

B. Substitute Bytes

Substitute byte forms the first step in each round. Here the output of pre-round transformation is used as an input to this step in which the elements of intermediate state matrix are replaced using an S-box table as shown in Figure 5. S-box is a major component of any cryptography algorithm, which performs substitution. It is a 16x16 Look Up Table (LUT) with its elements ranging from 00 to FF. Substitution has the largest share in power consumption and is one of the most complex process in the entire algorithm [15].



INPUT
State Matrix

```
   | 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
00 |63 7c 77 7b f2 6b 6f c5 30 01 67 2b fe d7 ab 76
10 |ca 82 c9 7d fa 59 47 f0 ad d4 a2 af 9c a4 72 c0
20 |b7 fd 93 26 36 3f f7 cc 34 a5 e5 f1 71 d8 31 15
30 |04 c7 23 c3 18 96 05 9a 07 12 80 e2 eb 27 b2 75
40 |09 83 2c 1a 1b 6e 5a a0 52 3b d6 b3 29 e3 2f 84
50 |53 d1 00 ed 20 fc b1 5b 6a cb be 39 4a 4c 58 cf
60 |d0 ef aa fb 43 4d 33 85 45 f9 02 7f 50 3c 9f a8
70 |51 a3 40 8f 92 9d 38 f5 bc b6 da 21 10 ff f3 d2
80 |cd 0c 13 ec 5f 97 44 17 c4 a7 7e 3d 64 5d 19 73
90 |60 81 4f dc 22 2a 90 88 46 ee b8 14 de 5e 0b db
a0 |e0 32 3a 0a 49 06 24 5c c2 d3 ac 62 91 95 e4 79
b0 |e7 c8 37 6d 8d d5 4e a9 6c 56 f4 ea 65 7a ae 08
c0 |ba 78 25 2e 1c a6 b4 c6 e8 dd 74 1f 4b bd 8b 8a
d0 |70 3e b5 66 48 03 f6 0e 61 35 57 b9 86 c1 1d 9e
e0 |e1 f8 98 11 69 d9 8e 94 9b 1e 87 e9 ce 55 28 df
f0 |8c a1 89 0d bf e6 42 68 41 99 2d 0f b0 54 bb 16
```

Fig. 5. Substitution byte transformation using 16x16 S-box

C. Shift Rows

The result obtained from substitution then undergoes shift rows operation in which circular left shift is performed on the state array as shown in figure 6. There is no shift in the first row of the matrix, a 1-byte circular left shift in second row, followed by 2 and 3 bytes circular left shift in row 3 and 4 respectively [16].

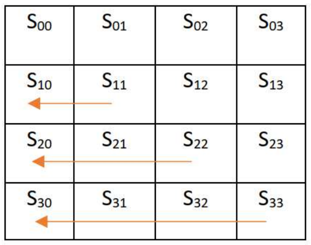


Fig 6. Shift rows operation on 128-bit block size data

D. Mix Columns

In this step, the state matrix obtained from shift row operation undergoes word by word multiplication with a constant matrix as shown in figure 7 [17].

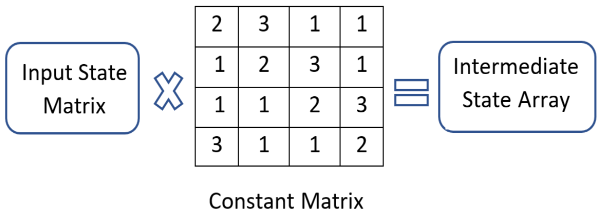


Fig. 7. Multiplication of state matrix with constant matrix

E. Add Round Keys

In this final step round keys are XORed with the output obtained from mix column operation and the results are passed on to the next round [18]. After all the 4 steps are performed iteratively for all the 10 rounds final cipher text is obtained. This encrypted message is in hexadecimal form. For performing decryption, a similar approach is undertaken where the encrypted message is passed through the 10 rounds of transformation with inverted Mix Columns and inverted shift rows. An inverted S-box is used in the substitute bytes step and the output obtained is the original message in hexadecimal form which can be further converted to plain text.

F. Pipelined Architecture

All operation inside the AES is performed on 128-bit block size of data, that means the input message (in hex) needs to be broken to and presented in 128-bit 4x4 matrices before being processed inside the AES [19]. Since the data blocks passes through the 10 iterative rounds in a sequential manner therefore only one block is processed at a given time in a particular round leaving the subsequent stages unused or idle. Parallel pipelining enables multiple data blocks to be processed parallelly in different stages hence making the encryption process of large messages or images faster [20].

IV. Results

To perform encryption of "*sample message 1*" with 128 bit key "*secure password1*" the resulting 128-bit encrypted output will be "786e4e6532761c57e253cc34814c233c".

The decryption module will take this 128-bit hexadecimal encrypted data as input and same 128 bit key as input to generate the decipher text message "sample message 1". The energy consumed per bit in transformation during the entire process was 7 pJ/bit and throughput of encrypted results was 68 Gbps. Figure 8 and 9 shows the produced encryption and decryption outputs.

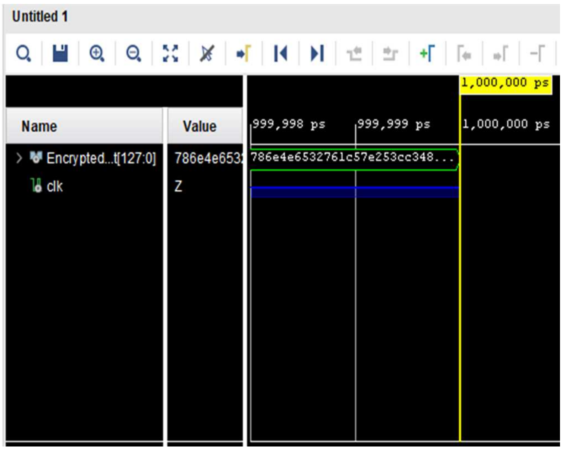


Fig. 8. Encrypted 128-bit hexadecimal output.

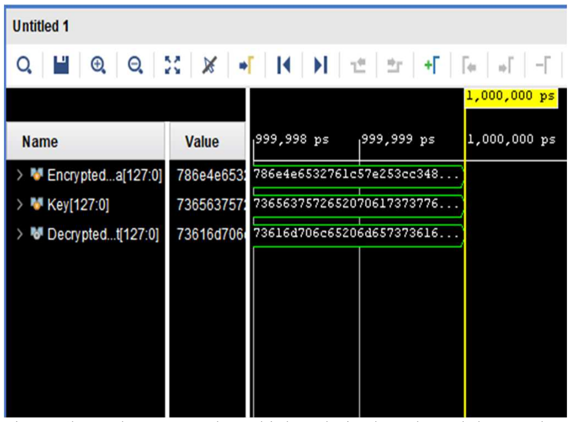
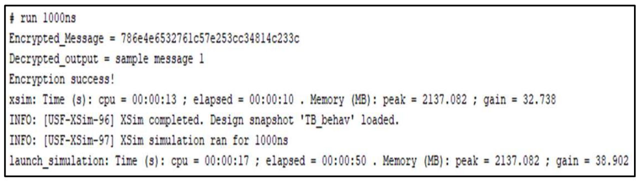


Fig. 9. Shows the Encrypted 128-bit hexadecimal results and decrypted message.

Xilinx Artix-7 FPGA device based on 28nm technology was used for hardware evaluation and Verilog HDL for programming. This FPGA provides highest performance/watt and has 215,360 logic cells for design implementation. For the AES-128 implementation, a total of 23,689 logic cells were used which comes to about 11% resource utilization on the FPGA. Simulation and synthesis task has been performed on Xilinx Vivado v2021.1 IDE. Figure 10 shows simulation of the given algorithm run for 1000ns and the encryption status in Vivado tcl console, peak memory used, gain, and time elapsed in producing simulation results.



```
# run 1000ns
Encrypted_Message = 786e4e6532761c57e253cc34814c233c
Decrypted_output = sample message 1
Encryption success!
xsim: Time (s): cpu = 00:00:13 ; elapsed = 00:00:10 . Memory (MB): peak = 2137.082 ; gain = 32.738
INFO: [USF-XSim-96] XSim completed. Design snapshot 'TB_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:17 ; elapsed = 00:00:50 . Memory (MB): peak = 2137.082 ; gain = 38.902
```

Fig. 10. simulation runtime and status of encryption.

Figure 11. shows the AES top module in Xilinx Vivado simulator with 128 bits of input data and cipher key and the clock pulse. Dataout[127:0] is the encrypted output message in hex form. Figure 12. Shows the Rounds operation module with datain[127:0], keyin[127:0], clk, rc[3;0] input vectors and keyout[127:0], rndout[127:0] output vectors.

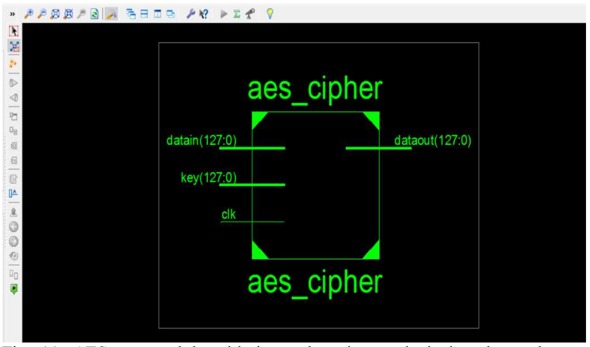


Fig. 11. AES top module with input data, key and clock pulse and encrypted output.

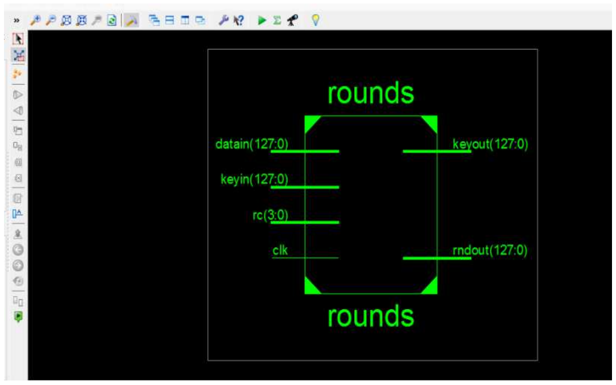


Fig. 12. Round transformation module with its input and output vectors

Table 2. shown below provides a variation in the energy/bit (pJ/b) of the design with varying temperature in kelvin at various input supply voltages (V).

Table 2. Energy/bit variation with temperature and input voltage

S. No	Temperature (K)	Input Voltage (V)	Energy/bit (pJ/b)
1	290	0.9	7
2	300	1	7.09
3	310	1.05	7.23
4	320	1.1	8.10

Table 3. shown below provides a variation in the dynamic power (W) and the leakage power (W) at various input supply voltages (V).

Table 3. Input voltage vs dynamic and leakage power

S. No	Input Voltage (V)	Dynamic Power (W)	Leakage Power (W)
1	0.9	0.69	0.09
2	1	0.83	0.11
3	1.05	0.9	0.12
4	1.1	1.01	0.14

Table 4 shown below provides a comparison of the throughput and energy consumed in per bit transformation of the presented parallel pipelined architecture with existing work.

Table 4. Result comparison of presented method with existing research.

S.No	Author	Algorithm	Through-put	Energy	Frequency
1	D.H Bui (2017) [6]	Data Path optimization	28 Mb/s	1 pJ/bit	10 MHz
2	C. Duran (2022) [7]	S-box acceleration	-	9.7 pJ/bit	100 MHZ
3	S. Chellapa (2015) [9]	Fully Pipelined	64 Gbps	-	500 MHz
4	This Paper	Parallel Pipelining	68 Gbps	7 pJ/bit	531.2 MHz

V. Conclusions

This paper proposes an efficient high throughput pipelined architecture of 128-bit AES cipher algorithm. A detailed study of the transformation processes in encryption/decryption of data has been presented to analyze the results obtained from each stage of the AES. The results exhibit high-rate encryption of 68 Gbps and a low energy consumption of 7 pJ/bit. Both encryption and decryption processes have been demonstrated for an alphanumeric text message using a 128-bit symmetric cipher key. This architecture proved to be more efficient and easier to implement as compared to other techniques adopted for low energy AES. Making it useful for applications in battery operated devices which have speed and power consumption constraints.

References

[1] W. H. Baker and L. Wallace, "Is Information Security Under Control?: Investigating Quality in Information Security Management," in IEEE Security & Privacy, vol. 5, no. 1, pp. 36-44, Jan.-Feb. 2007, doi: 10.1109/MSP.2007.11.

[2] H. S. Deshpande, K. J. Karande and A. O. Mulani, "Efficient implementation of AES algorithm on FPGA," 2014 International Conference on Communication and Signal Processing, pp. 1895-1899, 2014 doi: 10.1109/ICCSP.2014.6950174.

[3] C. E. Shannon, "Communication theory of secrecy systems," in The Bell System Technical Journal, vol. 28, no. 4, pp. 656-715, Oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.

[4] O. Hajihassani, S. K. Monfared, S. H. Khasteh and S. Gorgin, "Fast AES Implementation: A High-Throughput Bitsliced Approach," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 10, pp. 2211-2222, 1 Oct. 2019, doi: 10.1109/TPDS.2019.2911278.

[5] P. Hamalainen, T. Alho, M. Hannikainen and T. D. Hamalainen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core," 9th EUROMICRO Conference on Digital System Design (DSD'06), pp. 577-583, 2006 doi: 10.1109/DSD.2006.40.

[6] D. -H. Bui, D. Puschini, S. Bacles-Min, E. Beigné and X. -T. Tran, "AES Datapath Optimization Strategies for Low-Power Low-Energy Multisecurity-Level Internet-of-Things Applications," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 12, pp. 3281-3290, Dec. 2017, doi: 10.1109/TVLSI.2017.2716386.

[7] C. Duran and E. Roa, "A 10pJ/bit 256b AES-SoC Exploiting Memory Access Acceleration," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 3, pp. 1612-1616, March 2022, doi: 10.1109/TCSII.2021.3126984.

[8] M. Sumio, and A. Satoh. "An optimized S-Box circuit architecture for low power AES design." In *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 172-186. Springer, Berlin, Heidelberg, 2002.

[9] S. Chellappa, C. Ramamurthy, V. Vashishtha and L. T. Clark, "Advanced encryption system with dynamic pipeline reconfiguration for minimum energy operation," Sixteenth International Symposium on Quality Electronic Design, pp. 201-206, 2015 doi: 10.1109/ISQED.2015.7085425.

[10] S. Oukili and S. Bri, "High speed efficient advanced encryption standard implementation," 2017 International Symposium on Networks, Computers and Communications (ISNCC), 2017, pp. 1-4, doi: 10.1109/ISNCC.2017.8071975.

[11] R. V. Kshirsagar and M. V. Vyawahare, "FPGA Implementation of High Speed VLSI Architectures for AES Algorithm," 2012 Fifth International Conference on Emerging Trends in Engineering and Technology, pp. 239-242, 2012 doi: 10.1109/ICETET.2012.53.

[12] K. -L. Tsai, Y. -L. Huang, F. -Y. Leu, I. You, Y. -L. Huang and C. -H. Tsai, "AES-128 Based Secure Low Power Communication for LoRaWAN IoT Environments," in IEEE Access, vol. 6, pp. 45325-45334, 2018, doi: 10.1109/ACCESS.2018.2852563.

[13] C. Lu and S. Tseng, "Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter," Proceedings IEEE International Conference on Application- Specific Systems, Architectures, and Processors, pp. 277-285, 2002 doi: 10.1109/ASAP.2002.1030726.

[14] B. Subramanyan, V. M. Chhabria and T. G. S. Babu, "Image Encryption Based on AES Key Expansion," 2011 Second International Conference on Emerging Applications of Information Technology, pp. 217-220, 2011 doi: 10.1109/EAIT.2011.60.

[15] S. Kumar, V. K. Sharma and K. K. Mahapatra, "Low latency VLSI architecture of S-box for AES encryption," 2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT), pp. 694-698, 2013,doi: 10.1109/ICCPCT.2013.6528906.

[16] B. Bhat, A. W. Ali and A. Gupta, "DES and AES performance evaluation," International Conference on Computing, Communication & Automation, pp. 887-890, 2015 doi: 10.1109/CCAA.2015.7148500.

[17] K. Wu, Ramesh Karri, G. Kuznetsov and M. Goessel, "Low cost concurrent error detection for the advanced encryption standard," 2004 International Conferce on Test, pp. 1242-1248, 2004, doi: 10.1109/TEST.2004.1387397.

[18] F. J. D'souza and D. Panchal, "Advanced encryption standard (AES) security enhancement using hybrid approach," 2017 International Conference on Computing, Communication and Automation (ICCCA), pp. 647-652, 2017, doi: 10.1109/CCAA.2017.8229881.

[19] D. Punia and B. Singh, "Speed Optimization of the AES Algorithm Using Pipeline Hardware Architecture," 2019 International Conference on Communication and Electronics Systems (ICCES), pp. 2070-2074, 2019, doi: 10.1109/ICCES45898.2019.9002086.

[20] Q. Liu, Z. Xu and Y. Yuan, "A 66.1 Gbps single-pipeline AES on FPGA," 2013 International Conference on Field-Programmable Technology (FPT), pp. 378-381, 2013, doi: 10.1109/FPT.2013.6718392.