A High Throughput, Low Latency 105 Gbps Four-Pipeline Stage AES

DOI: https://doi.org/10.54654/isj.v1i21.1029

Tran Sy Nam, Luong The Dung, Nguyen Van Long

***Abstract*—The emergence of new technologies such as 6G, IoT, big data has heightened the significance of ensuring the security of high-throughput data. These technologies require even more high-performance encryption solutions to keep up with the increasing demands for secure and efficient data processing. This paper present a high-throughput, low latency 4-pipeline stages architecture of AES with efficient throughput per resource (Mbps/slice). The proposed implementation achieves a high throughput of 105.7 Gbps with an efficiency of 31.48 Mbps/slice. In comparison to state-of-the-art works, our design surpasses the majority of existing designs in terms of throughput and latency.**

***Tóm tắt*— Sự xuất hiện của các công nghệ mới như 6G, IoT, big data cho thấy tầm quan trọng của việc đảm bảo tính bảo mật của dữ liệu có thông lượng cao. Những công nghệ này đòi hỏi các giải pháp mã hóa hiệu suất cao hơn nữa để đáp ứng nhu cầu ngày càng tăng về xử lý dữ liệu an toàn và hiệu quả. Bài báo này trình bày một kiến trúc pipeline bốn giai đoạn tốc độ cao, độ trễ thấp của AES và có thể cài đặt hiệu quả theo thông số Mbps/slice. Kiến trúc đề xuất đạt được thông lượng cao 105.7 Gbps với độ hiệu quả 31.48 Mbps/slice. So với các công trình tiên tiến, thiết kế của nhóm tác giả vượt qua phần lớn các thiết kế hiện có về tốc độ và độ trễ.**

Keywords*— *AES, hardware implementation, pipeline, on-the-fly.

Từ khóa*— *AES; cài đặt phần cứng; pipeline; on-the-fly.

I. INTRODUCTION

In the ever-evolving landscape of technology, we have witnessed a remarkable acceleration in the pace at which data can be processed. Ethernet speeds, for instance, have soared to astonishing levels, with modern networks now capable of reaching speeds of up to 100 Gbps or even higher [1]. However, this impressive surge in data transmission capabilities has not been met with a proportional enhancement in security services. In fact, many security protocols, particularly those implemented in software, often find themselves constrained, struggling to cope with processing speeds that typically range between a modest 1 to 10 Gbps.

This incongruity between the breakneck pace of data transmission and the limitations of security services has given rise to a significant challenge. Applications, driven by the imperative of delivering data with minimal latency, have increasingly begun to bypass security services that cannot match the lightning-fast speed of data transfer. As a consequence, this circumvention of security measures can lead to a multitude of issues, including reduced throughput, heightened latency, and a consequent deterioration of the Quality of Service (QoS) experienced by users.

In light of these growing concerns and the increasing asymmetry between data processing speeds and security capabilities, it becomes evident that the study and development of high-performance cryptographic algorithms, particularly block ciphers, take on extraordinary importance. These algorithms hold the promise of bridging the gap between the rapid advancement of technology and the imperative need to ensure data integrity, confidentiality, and authenticity. As we delve into the realm of high-performance cryptography, we embark on a journey to empower security services, enabling them to keep pace with the swiftly

This manuscript is received on April 13, 2024. It is commented on May 28, 2024 and is accepted on June 14, 2024 by the first reviewer. It is commented on June 19, 2024 and is accepted on June 25, 2024 by the second reviewer.

evolving data landscape. This endeavor ensures that security remains an unwavering pillar in our digital age.

AES (Advanced Encryption Standard) remains one of the most prevalent symmetric key encryption algorithms. Over the years, it has garnered substantial attention from researchers in terms of both security and implementation. AES can be implemented in software or hardware. The advantages of implementing AES in software include ease of use, upgradability, portability, and flexibility. However, software implementation has lower performance and a reduced level of physical security, particularly concerning key storage issues [2]. In contrast, hardware implementation offers high-speed processing and enhanced physical security, as they are difficult for attackers to read or modify. Hardware implementation can be achieved using ASIC (Application-Specific Integrated Circuit) or FPGA (Field-Programmable Gate Array) technologies. FPGA implementation presents certain benefits when compared to ASIC implementation, including re-configurability and a reduced design cost. In this paper, we present a high throughput, low latency 105 Gbps 4-pipeline staged AES. We also demonstrate the method for finding the optimal architecture.

II. Literature review

This section discusses the recent high throughput implementations of AES presented in the literature. Design in [5] take 1 clock cycle per round, achieving a speed of 2.4 Gbps and an efficiency of 5.96 Mbps/Slice with a relatively high number of BRAMs (10 BRAMs). Bulens et al. in [7] designed an architecture implementing Sbox using LUT, on-the-fly key schedule, and 4-stage pipeline. This enables achieving a speed of 4.1 Gbps and an efficiency of 10.4 Mbps/Slice with the case of 4 data blocks being processed simultaneously. The design in [8] has a similar datapath to [7], however, implementing Sbox using BRAM and on-the-fly key schedule allows achieving a speed of 1.3 Gbps and an efficiency of 4.38 Mbps/Slice. However, these works focus on balancing speed and area using an iterative looping technique or a low datapath and, thus, cannot achieve extremely high speed.

The pipeline architecture helps the design achieve significantly higher speeds for non-feedback cipher modes. Rahimunnisa et al. [9] proposed an improved architecture from [6] to achieve a speed of 37.1 Gbps and an efficiency of 22.3 Mbps/Slice. However, the design by Rahimunnisa et al. [9] uses a large number of BRAMs to save resources, leading to lower frequency and not very high speed. Liu et al. [3] proposed a pipeline architecture with routed ShiftRows, LUT-based Sbox, and logic-based MixColumns, capable of achieving speeds up to 66 Gbps with an efficiency of 19.20 Mbps/Slice. In [4], Liu further improved the AES architecture to achieve a speed of 75.92 Gbps with an efficiency of 17.50 Mbps/Slice. The work of Liu et al. [4] uses 5 6-LUTs and 4 FFs to implement 1 output bit of the S-box. The delay between these components is also very high. In this paper, we propose a method to reduce the LUTs and FFs required for 1 output bit of the S-box and also to reduce the delay between these components. The work of Oukili [13] achieved a speed of 93.73 Gbps and an efficiency of 16.27 Mbps/Slice through an 8-stage pipeline. However, the architecture by Oukili [13] uses a large number of registers for the S-box (5 intermediate registers), leading to high latency. The work of Baby Chellam [14] achieved 104.06 Gbps with an efficiency of 39.7 Mbps/Slice. However, Baby Chellam [14] only mentions unrolling all 16 S-boxes of SubBytes using LUT and does not explain the S-box implementation in detail. Furthermore, [14] achieved 104.06 Gbps with a 5-stage pipeline architecture. In this paper, we show that our method can reduce one pipeline stage while achieving higher frequency and reducing latency by 47% compared to [14].

III. Background

A. AES algorithm

The Advanced Encryption Standard is a 128-bit block cipher that supports key sizes of 128 bits, 192 bits, and 256 bits, corresponding

to 10, 12, and 14 rounds, respectively. AES was standardized in 2001 as FIPS-197 by the US National Institute of Standards and Technology (NIST). The AES encryption process is illustrated in Figure 1.

The 128-bit block is initially divided into 16 bytes in the form of a 4x4 matrix, known as the state matrix. All AES calculations operate on this matrix. There are four fundamental steps in an AES encryption round: AddRoundKey, SubBytes, ShiftRows, and MixColumns.

AddRoundKey transformation

In the AddRoundKey transformation, the State array is combined with the Round Key array using a bitwise XOR operation.

SubBytes transformation

SubBytes employs an S-box in its transformation and represents the sole non-linear transformation within AES. The S-box is the only non-linear component in AES, making it resistant against known attacks. Nonetheless, this property also introduces performance and area constraints, thereby acting as a bottleneck. The S-box is calculated through a composition of the multiplicative inverse over $GF(2^8)$ and an affine transformation over $GF(2)$:

$$S(x) = Affine(x^{-1})$$

$$Affine = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

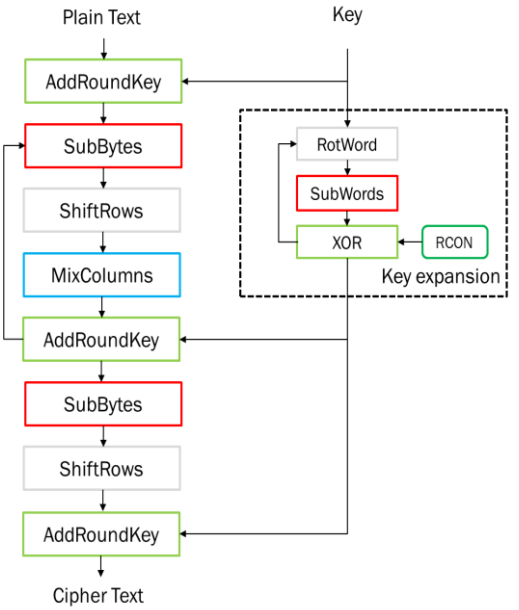

Figure 1. AES encryption process

ShiftRows transformation

ShiftRows is implemented by shifting each row of the matrix by a specific number of bytes as Figure 2:

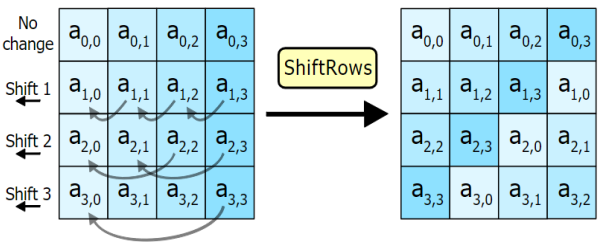

Figure 2. ShiftRows transformation in AES-128

MixColumns transformation

The MixColumns transformation is a linear diffusion process which operates independently on each column of the State matrix. In this transformation, each column of the State array is multiplied modulo x^4+1 with a fixed polynomial $c(x) = 03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02$.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Each round requires a round key generated by the key expansion scheme. The key expansion scheme involves three main steps: RotWord, SubWord, and XOR. Both SubBytes

and SubWord operations are performed using the S-box, while RotWord is analogous to ShiftRows.

B. Hardware implementation techniques

Block ciphers are commonly built based on iterative structures like SPN and Feistel, utilizing round functions. Various techniques are available for implementing block ciphers [19]:

- Iterative looping
- Loop unrolling
- Partial pipelining, full pipelining
- Partial pipelining with subpipelining (multi-stages pipeline)

Iterative looping is an effective technique for minimizing hardware requirements when implementing looping architectures. In this approach, only one round function is implemented, and the n-round block cipher needs to be repeated n times to perform both encryption and decryption. While this solution maintains low latency between registers, it does require a large number of clock cycles to execute.

The iterative looping technique is a specific case of loop unrolling, where only one loop can be unrolled. In contrast, the loop unrolling architecture permits multiple loops to be unrolled. While this approach optimizes the number of clock cycles needed for encryption and decryption, it results in increased latency of the registers, leading to a reduction in the system frequency.

The partially pipelined or fully pipelined structure offers the advantage of high speed by concurrently processing multiple data blocks. This is achieved through the implementation of round functions and the inclusion of registers to store data between rounds. For instance, in the case of the full pipeline architecture of AES, the design outputs 128-bit blocks of ciphertext per clock cycle. Nevertheless, this architecture demands significantly more hardware compared to loop unrolling. Furthermore, the pipeline architecture is particularly effective for modes without data feedback, such as ECB, CTR, and GCM. However, for data feedback modes like CBC, CFB, and OFB, the ciphertext of one block must be ready before the next block can be encrypted. Therefore, encrypting multiple blocks in a pipeline is not feasible for these modes.

Subpipelining within a pipeline architecture (multi-stages pipeline) is beneficial when the round function of a pipeline architecture becomes highly complex, resulting in substantial delays between pipeline layers. By introducing subpipeline stages, the round function of each pipeline stage is divided into smaller blocks. This will reduce the delay between stages but increase the number of clocks to perform the encryption.

IV. PROPOSED AES PIPELINE ARCHITECTURE

Because the SubBytes component typically consumes the largest number of logic gates, many studies have been published to optimize resources for S-boxes. The RAM or ROM implementation of S-boxes leads to a high critical path, resulting in low frequency. On the other hand, the logic-based implementation of S-boxes requires many pipeline stages, resulting in high latency. In our implementation, we optimized the lookup table of the S-box architecture using only native logic elements in each FPGA slice. This has proven to achieve the lowest resource usage and the lowest critical path**.**

Specifically, implementing 1 output bit of our S-box requires 4 6-input LUTs, 2 MUXF7, 1 MUXF8, and 1 FF (Figure 3). This architecture uses only native logic elements in a single slice, making it the most optimal because there is no delay between logic elements. For an 8-bit output of our S-box, 32 6-LUTs, 16 MUXF7s, 8 MUXF8s, and 8 FFs are needed. The best logic-based implementation of the S-box by Canright [16] requires 48 6-input LUTs, while the best lookup table implementation of the S-box requires 32 6-input LUTs [17]. Our proposed S-box implementation also offers the lowest number of LUTs. The S-box implementation in [4] requires 5 6-input LUTs and 4 FFs for 1 output bit, necessitating more LUTs and resulting in high interconnect

between logic elements. The authors in [13] implemented the S-box using a logic-based method, requiring a large number of registers, which leads to high latency (5 intermediate registers). The S-box implementation in [14] only mentions unrolling all 16 S-boxes in SubBytes without providing details, thus we have no evaluation.

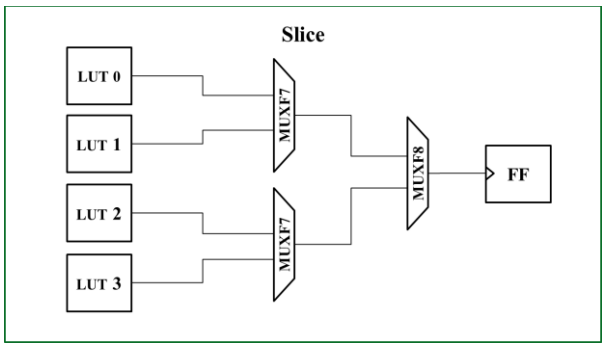

Figure 3. The proposed 1-bit output Sbox

The remaining low logic level operations in an AES round, including ShiftRows, MixColumns, and AddRoundKey, will be implemented using LUTs in our architecture.

The studies indicates that in AES pipeline architectures, the critical path lies in the key schedule scheme [3], [4], [18]. To keep up with the speed of the encryption pipeline, the key schedule is also pipelined using the on-the-fly method. With this solution, user keys can be changed arbitrarily in each clock cycle corresponding to a plaintext.

In the original key schedule scheme, the round key is calculated as follows:

$$K'_0 = K_0 \oplus Rcon \oplus SubWord\,(RotWord\,(K_3))$$
$$K'_1 = K'_0 \oplus K_1$$
$$K'_2 = K'_1 \oplus K_2$$
$$K'_3 = K'_2 \oplus K_3$$

With K_0, K_1, K_2, K_3 are 32 bit words of the last round key and K'_0, K'_1, K'_2, K'_3 are 32 bit words of the current round key. This scheme has a large delay because the following keys depend on the results of the previous keys. To reduce the critical path for the expansion scheme, at least 4 registers are needed to store the intermediate results. This approach can be found in works like [14], which uses 5 registers, or [13], which uses 7 registers.

To further reduce latency, we propose a modified key schedule scheme. Let 32 bit words P_0, P_1, P_2, P_3:

$$P_0 = K_0 \oplus Rcon$$
$$P_1 = K_1 \oplus P_0$$
$$P_2 = K_2 \oplus P_1$$
$$P_3 = K_3 \oplus P_2$$

Then 32 bit words K'_0, K'_1, K'_2, K'_3 are calculated as:

$$K'_0 = P_0 \oplus SubWord\,(RotWord\,(K_3))$$
$$K'_1 = P_1 \oplus SubWord\,(RotWord\,(K_3))$$
$$K'_2 = P_2 \oplus SubWord\,(RotWord\,(K_3))$$
$$K'_3 = P_3 \oplus SubWord\,(RotWord\,(K_3))$$

It can be seen that in the modified key schedule scheme, at least 2 registers are used for intermediate results instead of 4. This scheme will reduce both latency and resources in comparison with the original.

As mentioned, pipeline techniques reduce the critical path by inserting registers between operations with large delays. This helps increase the frequency and speed of the design. However, adding registers also increases resource usage and clock cycles, resulting in increased latency for the entire design. The task of finding the optimal number of registers to balance speed, resources, and latency is not a simple one. Based on the proposed S-box and key schedule scheme, the following register insertion locations are available:

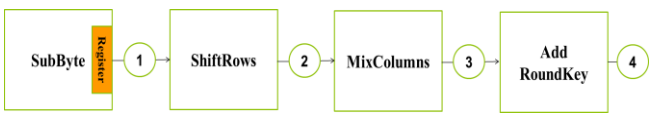

Figure 4. Register insertion locations of round AES.

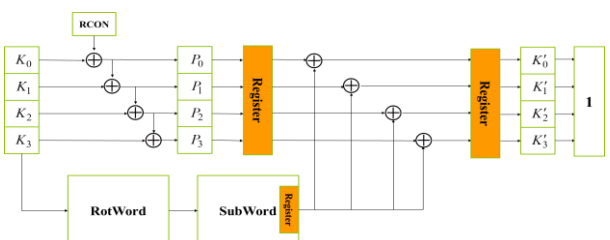

Figure 5. Register insertion locations of key schedule scheme AES

We propose Algorithm 1 to find the optimal number and location of registers.

Algorithm 1. Finding the optimal number and location of registers

k : number of inserted registers.

n : maxmum number of inserted registers.

$C_{n\,j}^{k}$: Location j of registers in combinations C_n^k .

$$C_n^k = (1,\cdots,1,\cdots,0)$$
$$\underbrace{\qquad\qquad}_{n}^{k}$$

Input: Proposed S-box, proposed key schedule scheme.

Output: Number and location of registers

$k = 0$, $location = 0$.

Find maximum frequency F_0 and area (slices) A_0. Calculate throughput $T_0 = F_0 \times 128$.

Calculate throughput/area ratio: $E_0 = \dfrac{T_0}{A_0}$

Calculate latency:

$L_0 = number\_clock \times \dfrac{1}{F_0}$.

Calculate the ratio of throughput/area to latency $H_0 = \dfrac{E_0}{L_0}$.

$E_{\max} = E_0$, $L_{\min} = L_0$, $H_{\max} = H_0$

for ($i = 1; i < n; i++$) **do**

  For $j = 0; j < C_n^i; j++$:

    Insert i registers at localtion j in AES round. The operation with highest logic level (SubBytes) will be inserted first. Insert or remove the registers in the proposed key schedule scheme to match with $i+1$ piplined stages.

    Find maximum frequency $F_{i,j}$ and area $A_{i,j}$.

    Calculate $E_{i,j}$, $L_{i,j}$, $H_{i,j}$.

    if ($H_{i,j} \geq H_{\max}$) **then**

      $H_{\max} = H_{i,j}$

      $E_{\max} = E_{i,j}$; $L_{\min} = L_{i,j}$

      $k = i$; $location = C_{n\,j}^{i}$

end

return k, $location$, $E_{\max}$, $L_{\min}$;

The algorithm finds the optimal number k of registers and their locations in the architecture based on the throughput/area to latency ratio H. The higher the value of parameter H, the higher the throughput-to-area ratio and the lower the latency. Using the proposed algorithm, we are able to find the optimal pipeline architecture with $k = 3$ for for AES-128, AES-192 and $k = 2$ for AES-256. The location of registers is as follows:

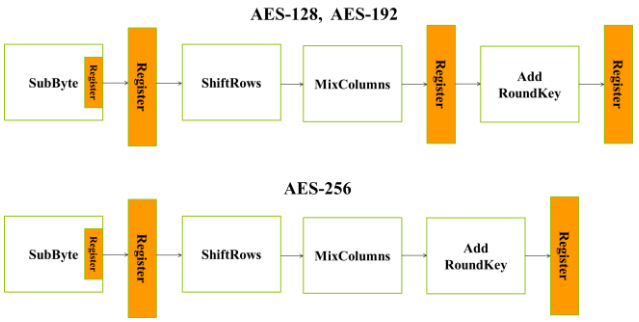

Figure 6. The optimal pipeline architecture of round AES

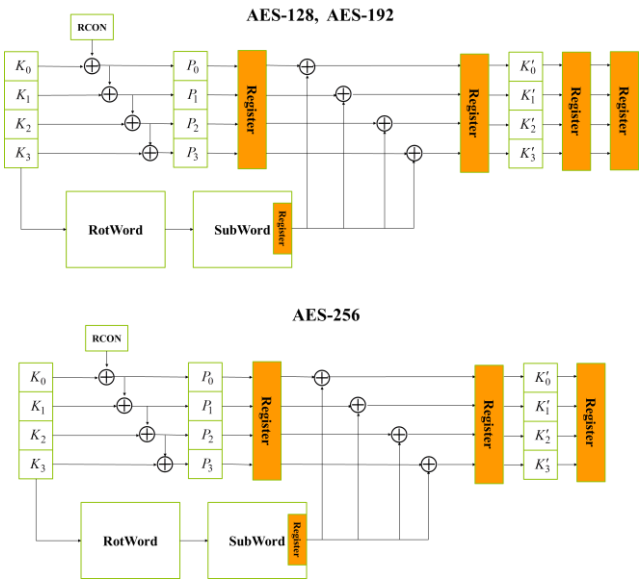

Figure 7. The optimal pipeline architecture of key schedule scheme AES

V. RESULTS AND DISSCUSSIONS

The frequency and latency of proposed AES with different key sizes in Virtex-7 FPGA device are shown in the following table:

TABLE 1. FREQUENCY AND LATENCY OF PROPOSED AES ARCHITECTURE

	Frequency (MHz)	**Clock cycle**	**Latency (ns)**
AES-128	826	40	48.4
AES-192	689	61	88.45
AES-256	740	42	56.7

The 4-stage pipeline architecture of AES-128 achieves the highest frequency and thus the highest throughput among AES versions. This architecture is able to encrypt 128-bit data input in each clock cycle. The ciphertexts are available after a latency of 40 clock cycles.

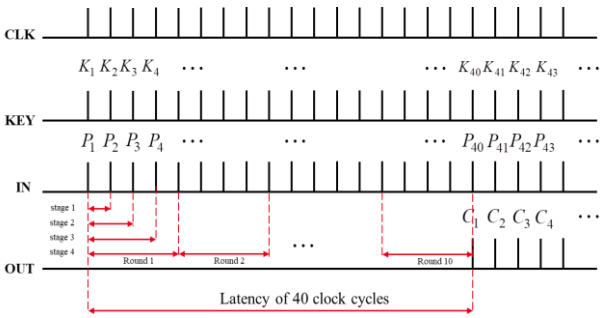

Figure 8. Input/output timing characteristics of proposed AES-128.

The results are compared with other implementations in Table 2. The speeds and latency of all designs in Table 2 are calculated using the formulas: $Speed = F_{\max} \times 128$ and

$$Latency = number\_clock \times \frac{1}{F_{\max}}.$$

The proposed AES design achieves the highest throughput of 105.7 Gbps, with an efficiency of 31.48 Mbps/Slice. Design [14] has higher efficiency; however, it also has a greater latency, exceeding the proposed design's latency by 47%. This is because [14] takes 58 clock cycles for the ciphertexts to become available, compared with 40 clock cycles in our implementation. Only the latency of [3] is lower than that of the proposed design, but the speed and efficiency of [3] are relatively low. The proposed AES design is encryption-only and can be applied in non-feedback modes such as ECB, CTR, and GCM. The resources and parameters reported here are for ECB mode of operation. Other modes like CTR and GCM will require more resources and control logic.

VI. CONCLUSION

This work proposes a high-throughput, low latency hardware architecture of AES for implementation in FPGA. The high-throughput, low latency of the design can be achieve by efficient implementation of AES functions combined with various methods, including subpipelining, look-up tables and on-the-fly key schedule scheme. The proposed implementation can help address the need for high-speed security services in 100 Gbps Gigabit Ethernet environments or crypto hardware acceleration in HSM and PCIe.

TABLE 2. RESOURCE SUMMARY FOR AES DESIGN

Work	Key size	Device	Pipeline stage	Slice	BRAM	Freq (MHz)	Latency (ns)	Speed (Gbps)	Efficiency (Mbps/Slice)
Our	128	xc7vx690t	4	3357	0	**826**	48.4	**105.7**	31.48
Our	192	xc7vx690t	4	3845	0	689	88.45	88.192	22.93
Our	256	xc7vx690t	3	4416	0	740	56.7	94.72	21.45
Baby Chellam [14]	128	xc7vx690t	5	2617	0	813	71.34	104.06	**39.7**
Oukili [13]	128	xv6vlx240t	8	5759	0	732	104	93.73	16.27
Liu [4]	128	xc7vx690t	5	4339	0	593	84	75.92	17.50
Liu [3]	128	xc7vx690t	2	3436	0	516.8	**38.6**	66.10	19.20
Rahimunnisa [9]	128	xc6vlx75t	?	1664	48	505.5	?	37.1	22.3
Shahbazi [12]	128	xc5vlx85	?	5974	0	622.4	?	79.7	13.3
Kouzehzar [11]	128	xc5vfx70t	5	9756	0	460	108.5	60	6.1
El Maraghy [8]	128	xc5vlx50	5	303	8+2	425	93.35	1.327	4.38
Soltani [10]	128	xc6vlx240t	5	28,520	0	803.988	63.24	102.9	3.6

REFERENCES

[1]. D'Ambrosia, John. "*100 gigabit Ethernet and beyond* [Commentary]." IEEE communications magazine 48.3 (2010): S6-S13.

[2]. R. Doud, "*Hardware crypto solutions Boost VPN*," Electron. Eng. Times, pp. 57–64, Apr. 12, 1999.

[3]. Liu Q, Xu Z , Yuan Y. *A 66.1 Gbps single-pipeline AES on FPGA*. In: 2013 international conference on, field-programmable technology (FPT). IEEE; 2013, p. 378–81.

[4]. Liu, Qiang, Zhenyu Xu, and Ye Yuan. "High throughput and secure advanced encryption standard on field programmable gate array with fine pipelining and enhanced key expansion." IET Computers & Digital Techniques 9.3 (2015): 175-184.

[5]. Chaves R, Kuzmanov G, Vassiliadis S, Sousa L. *Reconfigurable memory based AES coprocessor*. In: 20th international parallel and distributed processing symposium IPDPS, IEEE; 2006. p. 8.

[6]. Chodowiec P, Gaj K. *Very compact FPGA implementation of the AES algorithm*. In: Cryptographic hardware and embedded systems-CHES. Springer; 2003. p. 319–33.

[7]. Bulens P, Standaert FX, Quisquater JJ, Pellegrin P, Rouvroy G. *Implementation of the AES-128 on virtex-5 FPGAs*. In: Progress in cryptology–AFRICACRYPT. Springer; 2008. p. 16–26.

[8]. El Maraghy M, Hesham S, Abd El Ghany MA. *Real-time efficient FPGA implementation of aes algorithm*. In: 2013 IEEE 26th international, SOC conference (SOCC). IEEE; 2013. p. 203–8.

[9]. Rahimunnisa, K., et al. "FPGA implementation of AES algorithm for high throughput using folded parallel architecture." Security and Communication Networks 7.11 (2014): 2225-2236.

[10]. Soltani, Abolfazl, and Saeed Sharifian. "*An ultra-high throughput and fully pipelined implementation of AES algorithm on FPGA.*" Microprocessors and Microsystems 39.7 (2015): 480-493.

[11]. Kouzehzar, Hossein, Meisam Nesary Moghadam, and Pooya Torkzadeh. "*A high data rate pipelined architecture of AES encryption/decryption in storage area networks.*" Electrical Engineering (ICEE), Iranian Conference on. IEEE, 2018.

[12]. Shahbazi, Karim, and Seok-Bum Ko. "*High throughput and area-efficient FPGA implementation of AES for high-traffic applications.*" IET Computers & Digital Techniques 14.6 (2020): 344-352.

[13]. Oukili, Soufiane, and Seddik Bri. "*Hardware implementation of AES algorithm with logic S-box.*" Journal of Circuits, Systems and Computers 26.09 (2017): 1750141.

[14]. Baby Chellam, Manjith, and Ramasubramanian Natarajan. "*AES hardware accelerator on FPGA with improved throughput and resource efficiency.*" Arabian Journal for Science and Engineering 43 (2018): 6873-6890.

[15]. Satoh A, Morioka S, Takano K, Munetoh S. *A compact rijndael hardware architecture with S-box optimization*. In: Boyd C, editor. Advances in cryptology ASIACRYPT 2001, vol. 2248., Lecture notes in computer science. Berlin, Heidelberg: Springer; 2001. p. 239–54.

[16]. Canright D. *A very compact S-box for AES*. Springer; 2005.

[17]. Pradeep, A., Mohanty, V., Subramaniam, A. M., & Rebeiro, C. (2019). *Revisiting AES SBox composite field implementations for FPGAs*. IEEE embedded systems letters, 11(3), 85-88.

[18]. L. Henzen and W. Fichtner, "FPGA parallel-pipelined AES-GCM core for 100G Ethernet applications," ESSCIRC 2010, 2010, pp. 202-205.

[19]. Gaj, Kris, and Pawel Chodowiec. "*FPGA and ASIC implementations of AES.*" Cryptographic engineering (2009): 235-294.

[20]. Luong, T. T. (2022). Building the dynamic diffusion layer for SPN block ciphers based on direct exponent and scalar multiplication. Journal of Science and Technology on Information Security, 1(15), 38-45. https://doi.org/10.54654/isj.v1i15.836.

ABOUT THE AUTHOR

Tran Sy Nam

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: transynam1989@gmail.com

Education: Graduated with a major in Information Security of telecommunications systems at FSO Academy - Russian Federation in 2013; Graduated with a Master's degree in Cryptographic Engineering from the Academy of Cryptography Techniques in 2018.

Recent research direction: Network security, hardware implementation, embedded systems, block cipher, and the IoT.

Tên tác giả: **Trần Sỹ Nam**

Cơ quan làm việc: Viện Khoa học và Công nghệ mật mã, Ban Cơ yếu Chính phủ

Email: transynam1989@gmail.com

Quá trình đào tạo: Tốt nghiệp chuyên ngành An toàn thông tin các hệ thống viễn thông tại Học viện FSO – Liên Bang Nga vào năm 2013; Tốt nghiệp Thạc sĩ chuyên ngành Kỹ thuật mật mã tại Học viện Kỹ thuật mật mã vào năm 2018.

Hướng nghiên cứu hiện nay: Bảo mật mạng, cài đặt phần cứng, hệ thống nhúng, mã khối và IoT.

Lương Thế Dũng

Workplace: Academy of Cryptography Techniques.

Email: thedungluong1@gmail.com

Education: Received Bachelor's degree in 2001, and PhD in 2011 in Mathematical foundation for computers and computing systems from Military Technology Academy.

Recent research direction: Data privacy, cryptographic, data mining, machine learning for information security

Tên tác giả: **Lương Thế Dũng**

Cơ quan công tác: Học viện Kỹ thuật mật mã

Email: thedungluong1@gmail.com

Quá trình đào tạo: Nhận bằng Cử nhân năm 2001 và Tiến sĩ năm 2011 về Cơ sở toán học cho máy tính và hệ thống máy tính của Học viện Kỹ thuật Quân sự.

Hướng nghiên cứu hiện nay: An toàn thông tin

Nguyen Van Long

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission.

Email: longyenkk2@gmail.com

Education: Received engineering degree in 2014 and Master's degree in 2020.

Recent research direction: Microchip technology, FPGA, embedded Linux technology.

Tên tác giả: **Nguyễn Văn Long**

Cơ quan công tác: Viện Khoa học và Công nghệ mật mã, Ban Cơ yếu Chính phủ

Email: longyenkk2@gmail.com

Quá trình đào tạo: Nhận bằng kỹ sư năm 2014 và Thạc sĩ năm 2020 tại đại học Bách khoa Hà Nội, chuyên ngành Kỹ thuật điện tử.

Hướng nghiên cứu hiện nay: Công nghệ vi mạch, FPGA, công nghệ nhúng Linux.