Vlsi Design of Nano Aes Security Algorithm for Image Encryption

M.Abimanyu,S.Annamalai,K.Mariyappan,M.Mummoorthy,UG Student,Department of Electronics and Communication Engineering,Guided by,**Mr.A.R.Mahendren,M.E**

Asst.prof,Department of Electronics and Communication Engineering,
Arasu Engineering College,Kumbakonam-612 501

***Abstract*—This brief proposes an area-efficient AES design approach considering both application-specific integrated circuits (ASIC) and field-programmable gate arrays (FPGA) implementation characteristics. This brief focuses on optimizing and analyzing the design approach of Subbytes and MixColumns, which take up the most significant portion of AES hardware area. Furthermore, this brief presents an area-efficient AES intellectual property (IP) design by analyzing the trade-off relationship between area and clock cycles based on the datapath variations. The proposed AES IPs were designed using Verilog HDL and synthesized using Samsung 28nm standard cell library for performance comparison. The proposed AES IPs show the advanced normalized area efficiency of 70% over the existing AES design with the same datapath. Furthermore, the Xilinx UltraScale+ KCU116 evaluation board (XCKU5P) was used for FPGA verification and performance analysis. As a result, the FPGA implementation results show up to 36% better look-up table (LUT) utilization efficiency than the Xilinx AES IP, and up to 17.9 times better than the existing AES implementation results.**

***Index Terms*—Cryptography, AES, ASIC, FPGA, datapath.**

I. Introduction

ADVANCED encryption standard (AES) [1] is a symmetric-key encryption method adopted as a standard through a public offering by the U.S. National Institute of Standards and Technology (NIST). The AES has a similar structure to the data encryption standard (DES), in which functions are repeatedly used. However, unlike the DES, where security vulnerabilities have been found, the AES uses a substitution-permutation network (SPN) structure in the encryption and decryption process. The SPN requires an inverse encryption process for decryption in contrast with the Feistel structure, which employs the same procedure in both encryption and decryption in the DES. The AES supports key lengths of 128/192/256 bits and is secure against all known block encryption attacks. Since the AES shows strong computation efficiency and high flexibility in hardware implementation, it is widely used.

Research on the hardware implementation of the AES was conducted from various perspectives on an application-specific integrated circuit (ASIC) and a field-programmable gate array (FPGA) [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. The pipeline architectures [2] and unrolled architectures [3] showed high performances. However, [2] and [3] have disadvantages by having large areas. References [4] and [5] proposed design methods to reduce the datapath for the area efficiency in hardware implementation. Reference [4] designed the AES with a very low area at 8-bit datapath. Reference [5] defined the optimal datapath by analyzing the trade-off relationship and conducted the AES design at 32-bit datapath. References [6], [7], [8] implemented the AES with considering various constraints in FPGA.

Our previous work [13] proposed a resource-efficient AES design method considering the FPGA implementation characteristics. Based on this, this brief proposes an area-efficient AES intellectual property (IP) design approach based on the analysis of the implementation characteristics of ASIC as well as FPGA. This brief focuses on optimizing and analyzing the design approach of Subbytes and MixColumns, which take up the most significant portion of the AES hardware area. Furthermore, this brief presents an area-efficient AES IP design by analyzing the trade-off relationship between area (ASIC gate count, FPGA resource utilization) and clock cycle based on the datapath variations of each module.

The rest of this brief consists of the following. Section II proposes the area-efficient AES design methods considering ASIC and FPGA implementation characteristics. In addition, Section II shows the analysis of the trade-off relationship between area and clock cycles. Section III analyzes the implementation results with other AES designs, and finally, conclusions are drawn in Section IV.

II. Proposed Design Method

A. SubBytes Design Analysis

Subbytes [1], a byte-wise non-linear substitution operation, occupies the largest area in the AES hardware. There are

Manuscript received 10 April 2023; revised 26 May 2023; accepted 3 July 2023. Date of publication 11 July 2023; date of current version 25 September 2023. This work was supported in part by the National Research and Development Program through the National Research Foundation of Korea (NRF) funded by the MSIT under Grant NRF-2021R1A2C2010228, and in part by the Ministry of Science and Information and Communications Technology (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2022-2020-0-01461. This brief was recommended by Associate Editor X. Xue. *(Corresponding author: Myung Hoon Sunwoo.)*

Useok Lee, Ho Keun Kim, and Myung Hoon Sunwoo are with the Department of Electrical and Computer Engineering, Ajou University, Suwon 16499, South Korea (e-mail: aqua3693@ajou.ac.kr; hokeun92@ajou.ac.kr; sunwoo@ajou.ac.kr).

Jeahack Lee is with the SoC Platform Research Center, Korea Electronics Technology Institute, Seongnam 13509, South Korea (e-mail: jhk507@keti.re.kr).

Digital Object Identifier 10.1109/TCSII.2023.3293999

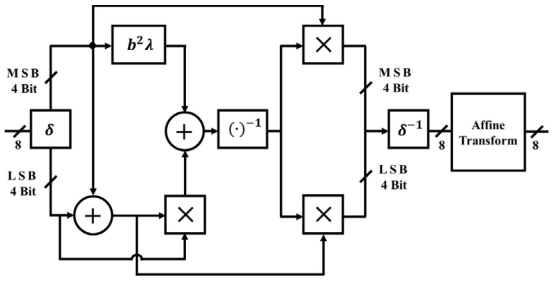

Fig. 1. CFA-based SubBytes architecture.

TABLE I
SUBBYTES DESIGN METHOD COMPARISON OF ASIC SYNTHESIS RESULTS

	L_{Path}	8	16	32	64	128
Method-A	GEs	513	1027	2054	4108	8216
	CPD	0.70ns				
Method-B	GEs	287	574	1149	2298	4596
	CPD	1.87ns				

two ways to design SubBytes. The first method (Method-A) is designing the S-Box as a memory-based look-up table (LUT) [1]. The second method (Method-B) is calculating the inverse of the multiplication in $GF(2^8)$ and designing the operation according to the Affine transform as a combinational logic [14].

For the design of Method-B, composite field arithmetic (CFA) of (1) can be applied [14].

$$(bx+c)^{-1} = b(b^2\lambda + c(b+c))^{-1}x + (c+b)(b^2\lambda + c(b+c))^{-1} \quad (1)$$

The SubBytes operation consists of the isomorphic (δ), inverse isomorphic (δ^{-1}), squarer (b^2), lambda multiplier (λ), multiplicative inversion $((\cdot)^{-1})$, $GF(2^4)$ multiplier, and Affine transform. The architecture of SubBytes can be designed, as shown in Fig. 1.

Table I presents the ASIC synthesis results of the SubBytes module based on different design methods, where L_{path} means the length of the datapath. Implementation of Method-B using a combination circuit, as implemented through [6], shows an average of 44% lower gate equivalent (GE) than the LUT-based Method-A, where GE can be calculated by an equivalent number of 2-input NAND gates. However, Method-A has a 62% shorter critical path delay (CPD) than Method-B. This means that a faster circuit can be achieved when designed with Method-A, which can impact the overall AES operation speed if applied to the same AES architecture, as confirmed by the results in Fig. 6 that will be presented later.

Table II shows the comparison results of FPGA resource utilization according to SubBytes design methods. FPGA has dedicated design resources such as block random access memory (BRAM) as well as LUTs and registers that are used flexibly for hardware implementation. Therefore, a direct comparison like Table I is not possible. For the SubBytes implementation, Method-A uses BRAM and Method-B uses LUTs. The LUT mentioned in Method-B means one of the FPGA design resources, not a look-up table as in Method-A.

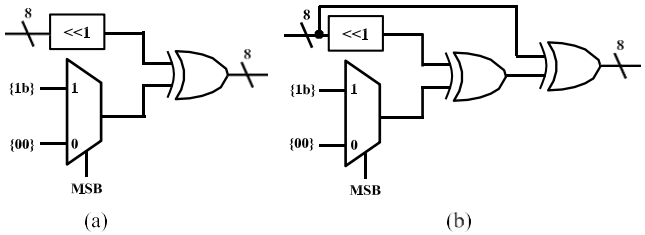

Fig. 2. (a) Optimized 0×02 multiplier. (b) Optimized 0×03 multiplier.

Direct comparison is difficult because each resource uses different physical cells, but Method-B is superior in utilization compared to the total available resources. On the other hand, Method-A can be seen as superior in design flexibility because LUTs have more usability when implementing hardware.

B. MixColumns Optimization

The MixColumns operations [1] are column-by-column multiplication in *GF*, the same operations as polynomials multiplication followed by modulo operation with the irreducible polynomial (2).

$$f(x) = x^8 + x^4 + x^3 + x + 1 \quad (2)$$

The MixColumns operations can be simplified as multiplication of fixed third-degree polynomial (3) followed by modulo operation with (4).

$$m(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (3)$$

$$n(x) = x^4 + 1 \quad (4)$$

These operations can be expressed as a matrix-based equation as in (5).

$$\begin{bmatrix} c_0' \\ c_1' \\ c_2' \\ c_3' \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad (5)$$

Then, the byte-wise operation for each column in (5) can be expressed as (6)–(9).

$$c_0' = (\{02\}\cdot c_0) \oplus (\{03\}\cdot c_1) \oplus c_2 \oplus c_3 \quad (6)$$

$$c_1' = c_0 \oplus (\{02\}\cdot c_1) \oplus (\{03\}\cdot c_2) \oplus c_3 \quad (7)$$

$$c_2' = c_0 \oplus c_1 \oplus (\{02\}\cdot c_2) \oplus (\{03\}\cdot c_3) \quad (8)$$

$$c_3' = (\{03\}\cdot c_0) \oplus c_1 \oplus c_2 \oplus (\{02\}\cdot c_3) \quad (9)$$

The byte operation method for (5) is defined as the *xtime* operation [1]. If the most significant bit (MSB) is 1 in this operation, left shifting is performed, followed by a bitwise XOR operation with {1b}. For higher-dimensional values, the *xtime* operation is repeated.

Therefore, we optimized the {02} multiplier as Fig. 2(a). In addition, since {03} is a continuous XOR operation of {01} and {02}, we optimized it as in Fig. 2(b). Since the result of multiplication with {01} is itself, multiplier implementation is not required. Fig. 3 shows the proposed 32-bit MixColumns architecture using the optimized multipliers of Fig. 2(a) and Fig. 2(b).

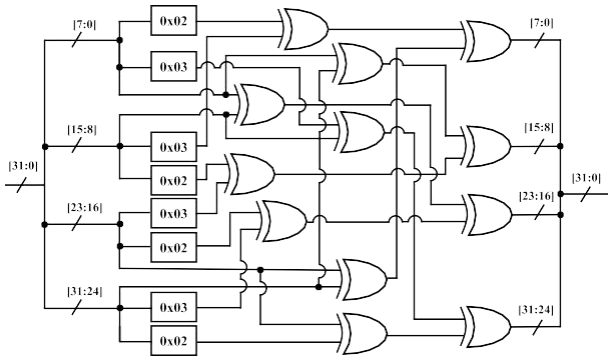

Fig. 3. Proposed 32-bit datapath MixColumns architecture.

TABLE II
SUBBYTES DESIGN METHOD COMPARISON OF FPGA IMPLEMENTATION RESULTS

	L_{Path}	8	16	32	64	128
Method-A	BRAM[a]	0.5	0.5	1	2	4
	Util.%[b]	0.18	0.18	0.36	0.71	1.43
	CPD	0.80ns				
Method-B	LUTs	85	170	340	680	1360
	Util.%[b]	0.04	0.08	0.16	0.31	0.63
	CPD	1.11ns				

[a] 36k BRAMs
[b] Based on Xilinx XCKU5P

TABLE III
ASIC SYNTHESIS AREA COMPARISON OF MIXCOLUMNS

L_{Path}	8	16	32	64	128
GEs	142			279	534
CPD	0.25ns			0.22ns	0.20ns

TABLE IV
MIXCOLUMNS DESIGN RESOURCE UTILIZATION COMPARISON

L_{Path}	8	16	32	64	128
LUTs	32[c]			64	128
Util.%[c]	0.01			0.03	0.06
CPD	0.71ns				

[c] Based on Xilinx XCKU5P

Tables III and IV show the ASIC area and the FPGA design resource utilization results when implementing MixColumns. In the MixColumns logic, the area is relatively small compared to the SubBytes implementation result, as shown in Tables I and II. Although $L_{path} = 8$ and 16 MixColumnns can be implemented, additional resources are required due to the structural limitations of the MixColumns operation, which is performed in units of columns. Therefore, it requires more resources than the MixColumnns of 32-bit and consumes additional clock cycles. Thus, in this brief, $L_{path} = 32$ MixColumns is used as the basic design module.

C. Trade-Off Analysis According to L_{path} Variation

This brief employs a round-based AES design approach [15]. The AES round module consists of SubBytes, ShiftRow, MixColumns, and AddRoundKey. ShiftRow is implemented with simple wires or shift registers depending on L_{path}.

AddRoundKey requires parallel XOR gates equal to L_{path}. Based on the results in Tables I–IV and AddRoundKey implementation characteristic, we formulated the number of GEs and LUTs of combinational logic required for designing round modules in ASIC and FPGA as shown in (10) and (11).

$$GEs_{Round} = \frac{531 \times L_{Path}}{8} + 142 \times MAX\left(1, \frac{L_{Path}}{32}\right) \quad (10)$$

$$LUTs_{Round} = \frac{93 \times L_{Path}}{8} + MAX(L_{Path}, 32) \quad (11)$$

At this point, registers for pipeline architecture and serial-parallel substitution, including ShiftRow, have not been considered. For example, the ShiftRow module with an L_{path} of 128 in ASIC results uses only 128 GEs. However, if the L_{path} is not 128, the ShiftRow module can be implemented as a mixed circuit of shift registers and wires. For example, if the L_{path} is 8 and 32, GEs of 1326 and 1383 are required, respectively. Because of this, it is difficult to grasp the linear relationship and can produce different results depending on the design method.

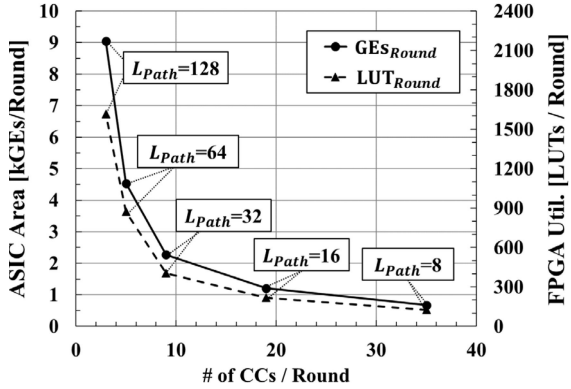

Fig. 4. Trade-off relationship between the area and clock cycles with the change of L_{path}.

The relationship between the area and clock cycles in a round can be found in Fig. 4. GEs_{Round} and $LUTs_{Round}$ denote the number of GEs and LUTs required for a round module, respectively. This brief proposes AES-128 encryption IPs for three L_{path} based on ASIC and FPGA implementation characteristics shown in Fig. 4. The three L_{path} for the proposed AES IPs are as follows: $L_{path} = 32$ with the sharpest slope change, $L_{path} = 8$ with the least LUT utilization, and $L_{path} = 128$ with the least CC consumption.

D. Proposed AES Architecture

Fig. 5 shows the basic architecture of the proposed round-based AES. Each module is designed and modified for each target L_{path}. SubBytes, MixColumns, and AddRoundKey inside the round module are in parallel according to the target L_{path}. Shift registers were used in serial-parallel I/O combination and distribution, including ShiftRows operations. The round module is used for a total of 10 repetitions.

The *En_sig* is applied to the global counter, which outputs *data_in_sel*, *last_rnd_sig*, *rnd_sig*, and *key_in_sel* signals. The *data_in_sel* determines the input of the round module.

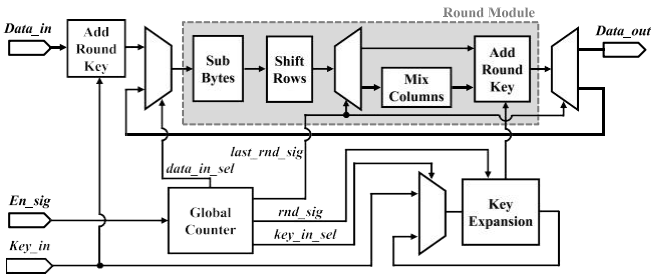

Fig. 5. Proposed AES architecture.

The *key_in_sel* decides the input of the KeyExpansion. The *rnd_sig* is used as an input of KeyExpansion for generating the round key corresponding to each round. Since the MixColumns operation does not need in the last round, the global counter sends *last_rnd_sig* to the multiplexer to bypass the MixColumns operation. In addition, the *last_rnd_sig* decides the final output of the proposed AES.

III. Implementation Results

The proposed AES IPs in this brief were designed using Verilog HDL and synthesized using Samsung 28nm standard cell library and Synopsys Design Compiler for ASIC performance comparison. The Xilinx UltraScale+ KCU116 evaluation board (XCKU5P) was used for FPGA verification and performance analysis. Both Method-A (LUT-based) and Method-B (CFA-based) were considered for SubBytes design, and area and clock cycle analysis were performed according to L_{path} variation. Based on this, we designed six versions of AES IPs with different L_{path} and implementation methods. In addition, the proposed decoders were designed with a fully-pipelined architecture to achieve high operating frequency and throughput. This brief uses area efficiency based on GEs in ASIC and LUTs in FPGA, which can be represented as follows.

$$\text{Efficiency}_{\text{GEs}} = \frac{\text{Throughput}}{\text{GEs}} \tag{12}$$

$$\text{Efficiency}_{\text{LUTs}} = \frac{\text{Throughput}}{\#\text{ of LUTs}} \tag{13}$$

Fig. 6 shows the ASIC synthesis results of the proposed AES designs. As reported in Section II-A, the CFA-based SubBytes design (Method-B) has a smaller area than the LUT-based SubBytes design (Method-A), but operates at a slower speed. When both Methods-A and B are applied to the same AES architecture, Method-B affects the critical path, as shown in Fig. 6. For example, the AES designed with a 32-bit datapath has 9.6% fewer GEs in Method-B than in Method-A, but has a 51.8% slower maximum operating frequency. Since the area efficiency based on (11) is 46.5% lower in Method-B, the AES design in this brief used Method-A compared to other references.

Table V shows a comparison of synthesis results for the proposed AES designs. The proposed AES designs were not only designed for the low area but also for high operation frequency, which required a large number of registers for pipeline architecture. Therefore, the proposed AES designs use more area than [4], [5], and [16]. In addition, the proposed

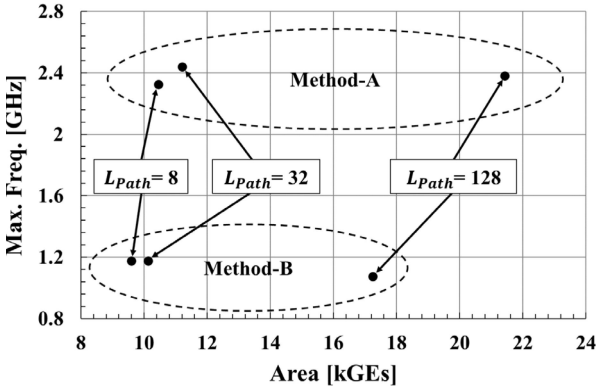

Fig. 6. ASIC synthesis results of the proposed AES designs.

TABLE V
ASIC Synthesis Results Comparison

	Proposed (Method-A)			[16]	[4]	[5]
Tech. [nm]	28			40	22	28
Voltage [V]	0.9			0.9	0.9	0.6
L_{Path} [bits]	8	32	128	8	32	32
Area [kGEs]	10.4	11.2	21.4	2.2	1.9	8.6
Power [mW]	7.10	6.33	12.01	4.39	13[e]	0.02
# of CCs	364	94	31	337	336	44
Max. Freq. [GHz]	2.3	2.4	2.4	1.3	1.1	60[MHz][f]
Max. T_P [Mbps]	818	3321	9831	494	432	170
Eff.$_{\text{GEs}}$ [Mbps/kGEs]	78.6	296.5	459.4	221.6	221.7	19.8
Normalized for 28nm and 0.9 V[d]						
Power [mW]	7.10	6.33	12.01	3.07	16.55[e]	0.05
Max. Freq. [GHz]	2.3	2.4	2.4	1.9	0.9	60[MHz][f]
Max. T_P [Mbps]	818	3321	9831	705	339	170
Eff.$_{\text{GEs}}$ [Mbps/kGEs]	78.6	296.5	459.4	316.6	174.2	19.8

[d] Frequency ∝ 1/s, and power ∝ v^2s, where s and v are the technology node and supply voltage ratio
[e] Total encrypt/decrypt accelerator power consumption
[f] Targeting low power AES for internet of things (IoT).

AES IPs consume more power than [5] and [16]. However, regarding area efficiency, especially for the proposed AES with L_{path} = 128, each design shows 45% and 163% higher normalized area efficiency than [4] and [16], respectively. In addition, for the 32-bit datapath AES, the proposed IP shows 70% higher normalized area efficiency than [4] and 15 times higher area efficiency than [5].

Table VI compares the implementation results between the proposed AES IPs and Xilinx AES IP [17]. Since [17] does not use BRAM, AES IPs with Method-B were used for comparing under the same criteria. Reference [17] is implemented in two modes: low and high throughput. The trade-off relationship between throughput and resource utilization of the two modes can be shown in Table VI. The proposed AES IPs were designed with low area as the top priority. Therefore, the comparison is performed with [17] for the low throughput mode with low resource utilization. As a result, it can be seen that all three proposed AES IPs have lower design resource usages than the low mode of [17]. In addition, compared to utilization efficiency, the proposed AES IP with L_{path} = 32 and 128 are about 10% and 36% better, respectively.

TABLE VI
FPGA IMPLEMENTATION RESULTS COMPARISON WITH XILINX AES IP

	Proposed (Method-B)			Xilinx AES IP [17][g]	
	L_{Path}= 8	L_{Path}= 32	L_{Path}= 128	T/P Mode: L	T/P Mode: H
Device	XCKU5P			KCVU27P	KCKU13P
LUTs	676	941	2195	2600	11996
Registers	543	725	907	2235	4987
Max Freq. [MHz]	556	526	500	250	250
Max T/P [Mbps]	195	717	2064	1800	25600
$Eff._{LUTs}$ [Mbps/LUTs]	0.29	0.76	0.94	0.69	2.13

[g] 128-bit key size and data input width

TABLE VII
COMPARISON OF FPGA IMPLEMENTATION RESULTS

	Proposed (L_{Path}=32)		[6]	[7]	[8]
	Method-A	Method-B			
Device	XCKU5P		XC7V X690T	XC7A100T	
LUTs	508	941	10773	3720	2814
Registers	725	725	3760	-	811
BRAM	1 5	-	-	-	2
# of CCs	94	94	20	42	44
Max Freq. [MHz]	667	526	208	292	273
T/P [Mbps]	908	717	1028	889	795
Util. Efficiency	1.79	0.76	0.10	0.24	0.28

Table VII shows the comparison results of the proposed AES IPs with the Xilinx FPGA-based AES design of [6], [7], [8]. Since L_{path} = 32 showed the optimal utilization and clock cycle trade-off relationship in Section III-C, we used L_{path} = 32 AES IPs for implementation results comparison in Table IV. Both of the proposed AES IPs use the least number of design resources such as LUT, registers, and BRAM compared to [6], [7], [8]. Although the throughputs of [6], [7], [8] are higher than the proposed AES IP with Method-B, the proposed IP shows 2.7 to 7.6 times higher utilization efficiency. In addition, in the proposed AES IP with Method-A, its throughput is higher than [7] and [8]. Furthermore, its utilization efficiency is 17.9 times better than [6], and the proposed IPs have superior efficiency in resource utilization.

IV. CONCLUSION

This brief proposed an area-efficient AES IP design method considering ASIC and FPGA implementation characteristics. First, the implementation results of SubBytes and MixColumns, the most significant components in AES, were analyzed. Furthermore, the trade-off results between the area and clock cycles in the round-based AES architecture were analyzed. Based on this, we designed six versions of AES IPs with different L_{path} and implementation methods. For ASIC synthesis results, the proposed AES IPs show a normalized area efficiency of 70% over [4] and 15 times higher area efficiency than [5] at a maximum target frequency of 60MHz.
In addition, the proposed AES IPs showed up to 36% better LUT utilization efficiency than the Xilinx AES IP [17], and achieved about 17.9 times higher utilization efficiency than [6].

The proposed AES IPs are area efficient and are suitable for systems where area efficiency is highly prioritized. However, a countermeasure against side-channel attacks is required when designing AES hardware. Therefore, if a design for side-channel attacks is applied through follow-up research, it is possible to design AES IPs with improved security and area efficiency.

ACKNOWLEDGMENT

The EDA tool was supported by the IC Design Education Center (IDEC), South Korea.

REFERENCES

1 "Advanced encryption standard," U.S. Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Rep. NIST FIPS-197, 2001.
2 S. K. Mathew et al., "53 Gbps native GF(2^4)^2 composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 767–776, Apr. 2011.
3 P. Maene and I. Verbauwhede, "Single-cycle implementations of block ciphers," in *Lightweight Cryptography for Security Privacy*, vol. 9542. Cham, Switzerland: Springer, 2016, pp. 131–147.
4 S. Mathew et al., "340 mV-1.1 V, 289 Gbps/W, 2090-gate nanoAES hardware accelerator with area-optimized encrypt/decrypt GF(2^4)^2 polynomials in 22 nm tri-gate CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 4, pp. 1048–1058, Apr. 2015.
5 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," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 12, pp. 3281–3290, Dec. 2017.
6 N. S. S. Srinivas and M. Akramuddin, "FPGA based hardware implementation of AES Rijndael algorithm for encryption and decryption," in *Proc. Int. Conf. Elect. Electron. Optim. Techn. (ICEEOT)*, 2016, pp. 1769–1776.
7 S. P. Guruprasad and B. S. Chandrasekar, "An evaluation framework for security algorithms performance realization on FPGA," in *Proc. IEEE Int. Conf. Current Trends Adv. Comput. (ICCTAC)*, 2018, pp. 1–6.
8 N. Jain, D. S. Ajnar, and P. K. Jain, "Optimization of advanced encryption standard algorithm (AES) on field programmable gate array (FPGA)," in *Proc. Int. Conf. Commun. Electron. Syst. (ICCES)*, 2019, pp. 1086–1090.
9 H. K. Kim and M. H. Sunwoo, "Low power AES using 8-bit and 32-bit datapath optimization for small Internet-of-Things (IoT)," *J. Sign. Process. Syst.*, vol. 91, 1283–1289, Dec. 2019.
10 Y. Wang and Y. Ha, "FPGA-based 40.9-Gbits/s masked AES with area optimization for storage area network," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 1, pp. 36–40, Jan. 2013.
11 K. Shahbazi and S.-B. Ko, "Area-efficient nano-AES implementation for Internet-of-Things devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 1, pp. 136–148, Jan. 2021.
12 A. Nakashima, R. Ueno, and N. Homma, "AES S-box hardware with efficiency improvement based on linear mapping optimization," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 10, pp. 3978–3982, Oct. 2022.
13 U. Lee, H. K. Kim, Y. J. Lim, and M. H. Sunwoo, "Resource-efficient FPGA implementation of advanced encryption standard," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 1165–1169.
14 P. V. S. Shastry, A. Agnihotri, D. Kachhwaha, J. Singh, and M. S. Sutaone, "A combinational logic implementation of S-box of AES," in *Proc. 2011 IEEE 54th Int. MWSCAS*, Aug. 2011, pp. 1–4.
15 P.-C. Liu, J.-H. Hsiao, H.-C. Chang, and C.-Y. Lee, "A 2.97 Gb/s DPA-resistant AES engine with self-generated random sequence," in *Proc. Eur. Solid-State Circuit Conf. (ESSCIRC)*, Sep. 2011, pp. 71–74.
16 Y. Zhang, K. Yang, M. Saligane, D. Blaauw, and D. Sylvester, "A compact 446 Gbps/W AES accelerator for mobile SoC and IoT in 40nm," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Jun. 2016, pp. 1–2.
17 *Advanced Encryption Standard (AES) Engine PG383 (V1.1), LogiCORE IP Product Guide*, Xlinx, San Jose, CA, USA, Jun. 2020.