High Throughput Architecture for the Advanced Encryption Standard Algorithm

Salma Hesham*, Mohamed A. Abd El Ghany** and Klaus Hofmann***
Electronics Department, German University in Cairo, Cairo, Egypt*
Electronics Department, German University in Cairo, Egypt. Integrated Electronic Systems Lab, TU Darmstadt, Germany**
Integrated Electronic Systems Lab, TU Darmstadt, Germany***
E-mails: {salma.hesham, mohamed.abdel-ghany}@guc.edu.eg, klaus.hofmann@ies.tu-darmstadt.de

***Abstract*—A high throughput architecture is proposed for an efficient implementation of the Advanced Encryption Standard (AES) Algorithm. The presented architecture is adapted for AES encryptor-only as well as integrated AES encryptor/decryptor designs. The *SubBytes*/*InvSubBytes* operations are implemented using composite field arithmetic in order to exploit the sub-pipelining advantage within the loop-unrolling methodology. The proposed architecture minimizes the critical path delay through the modification of the *SubBytes*/*InvSubBytes* as well as the *KeyExpansion* modules. Compared to previously reported AES encryptors and integrated AES encryptors/decryptors designs, the proposed architecture provides an efficiency improvement of 61% and 29% respectively.**

Keywords—Advanced encryption standard, FPGA, sub-pipelining, composite field arithmetic, throughput.

I. INTRODUCTION

The continuously growing number of internet and wireless communications users has marked security as a crucial designing factor for reliable communications. The Advanced Encryption Standard (AES) algorithm [1] was approved in October 2000 by the National Institute of Standards and Technologies to become the new encryption standard for its high flexibility and strong security.

Several hardware implementations for the AES algorithm were previously presented targeting either ASIC as [2-4], or FPGA as [5-11]. The AES algorithm operates on 128-bit data blocks using a cipher key of possible lengths 128/192/256-bits throughout 10/12/14 iterative rounds respectively. Each round consists of a set of transformations namely: *SubBytes*, *ShiftRows*, *MixColumns*, *AddRoundKey* or their corresponding inverses during decryption. All operations are performed on the 128-bits arranged in a 4 × 4 matrix of bytes called *State*. In a parallel manner, the input cipher key is processed through a *KeyExpansion* module to produce 10/12/14 round keys used in each respective round [1]. The architecture of the AES round unit is the core which distinguishes each hardware design. The implementation of the *SubBytes*/*InvSubBytes* and *KeyExpansion* modules is the design key for the AES round. The *SubBytes/InvSubBytes* modules are implemented either using storage memory units of size 256 bytes as [9-11] or using Composite Field Arithmetic (CFA) computations as [2-8]. Similarly, the *KeyExpansion* module follows either the pre-computing and storage method as [12], or the on-the-fly computation as [4-7]. In addition, the AES complete structure may adopt the loop-unrolling approach aiming for high throughput or the iterative looping approach aiming for area minimization.

In this paper, the presented 128-bit AES round unit architecture targets high throughput with optimized area. The sub-pipelined technique is utilized in order to break down the critical path delay as shown in Fig. 1. Accordingly, *SubBytes*/*InvSubbytes* are implemented using CFA for better exploit of the sub-pipelining approach. In order to minimize the critical path delay, two sub-functions in the proposed *SubBytes/InvSubBytes* architecture are combined. Moreover, the *KeyExpansion* module is modified through the parallelization of its sub-steps which ensures a further enhancement in the critical delay. The proposed AES round architecture is adapted for both types of design: encryptors-only as well as integrated encryptors/decryptors.

The rest of this paper is organized as follows. Section II, presents the proposed AES encryption round architecture. In Section III, the adaptation of the proposed architecture for the integrated encryption/decryption design is presented. In Section IV, the implementation results are provided and compared to previously reported FPGA designs. Finally, the work is concluded in Section V.

II. AES ENCRYPTIONROUND ARCHITECTURE

The Proposed AES encryption round architecture is abstractly described by the block diagram shown in Fig. 1. Through the following subsections the implementation of each module is presented followed by the description of the sub-pipelining scheme utilized.

A. SubBytes Transformation using CFA

The *SubBytes* transformation processes the 128-bit state value on a byte level. Each byte is substituted by its multiplicative inverse followed by an affine transformation [1]. The main complexity lies in computing the multiplicative

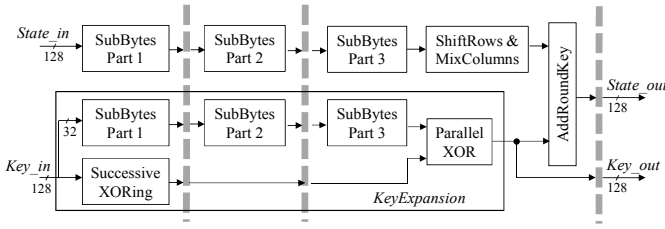

Fig. 1 Proposed AES sub-pipelined encryption round block diagram

978-1-4799-4558-0/14/$31.00

inverse in $GF(2^8)$. The CFA implementation guarantees a memory-free low-cost design, with optimum exploit of the sub-pipelining technique. In addition, the complexity of the multiplicative inverse computation is reduced from $GF(2^8)$ to $GF(2^4)$. Using CFA, any element A in $GF(2^8)$ is mapped using a mapping matrix T into an element B in the composite field $GF((2^4)^2)$ with the form $B = b_1x + b_0$ where b_1, b_0 are in $GF(2^4)$. The mapping matrix T is derived based on the defining irreducible polynomials of the fields $GF(2^8)$ and $GF((2^4)^2)$ following the algorithm presented in [13]. The multiplicative inverse of the mapped value B is computed by

$$B^{-1} = b_1 \varphi^{-1} x + (b_1 + b_0.\rho) \varphi^{-1}, \quad (1)$$

where $\varphi = (b_1^2\delta + b_0 (b_1\rho + b_0))$ is in $GF(2^4)$, while δ and ρ are constant coefficients in $GF(2^4)$ [13]. The value B^{-1} is then mapped back to $GF(2^8)$ using the inverse mapping matrix T^{-1}. The *SubBytes* operation is then finalized by the affine transformation. There are different possible optimum ρ, δ and T combinations for efficient hardware implementation of the *SubBytes* module [2-5]. In this work, the values used for ρ and δ are {0001} and {1001} respectively, with mapping matrices T and T^{-1} provided in [4]. For ρ = {0001}, the hardware realization of φ in (1) requires an *XOR* gate to compute $(b_1 + b_0)$, a $GF(2^4)$ multiplier, squarer, constant multiplier by δ and a final *XOR* gate. In addition, a $GF(2^4)$ multiplicative inverse module together with two $GF(2^4)$ multipliers are needed to complete the computation of (1).

Fig. 2 shows the proposed *SubBytes* architecture, where the computation of (b_1+b_0) is combined with the mapping matrix T. The proposed combination eliminates 1 *XOR* gate delay from the critical path of the *SubBytes* module. This is accomplished by deriving the bit level equations of the 4-bit output $O = (b_1 + b_0)$ in terms of the original input Byte $A = \{a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0\}$ as follows:

$$O_3 = a_7 + a_5 + a_4 + a_2, \quad (3)$$
$$O_2 = a_7 + a_3 + a_4 + a_1, \quad (4)$$
$$O_1 = a_6 + a_5, \quad (5)$$
$$O_0 = a_6 + a_3 + a_2 + a_1 + a_0. \quad (6)$$

The sub-expressions needed to compute (3), (4), (5) and (6) are all commonly shared with the T matrix module except the sub-expression term $(a_4 + a_2)$. Therefore, the proposed enhancement in the critical delay does not add any extra cost to the hardware design. On the other hand, following the conventional implementations, the $GF(2^4)$ squarer is combined with the constant multiplier and the final inverse mapping T^{-1} is combined with the affine transformation. Furthermore, the $GF(2^4)$ multiplicative inverse module is implemented using derived bit equations since further decomposition into $GF((2^2)^2)$ will not lead to neither better area nor better delay [5].

B. ShiftRows, MixColumns, AddRoundKey Transformations

The *ShiftRows* transformation is implemented through simple signal wiring, where the 4 rows of the state matrix are cyclically shifted to the left by 0, 1, 2 and 3 offset bytes respectively. Subsequently, the *MixColumns* transformation processes the shifted state matrix by multiplying each column with the fixed polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ modulo $x^4 + 1$. The realization of the *MixColumns* module follows the design presented in [11]. Finally, to perform the *AddRoundKey* transformation, the 128-bit output of the *MixColumns* is directly *XOR*ed with the round key K_r supplied from the *KeyExpansion* module.

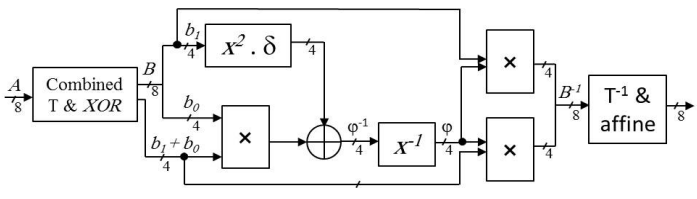

Fig. 2 Proposed SubBytes architecture

C. Proposed KeyExpansion Module

The key expansion algorithm computes each 128-bit round key $K_r = (w_{r,0}, w_{r,1}, w_{r,2}, w_{r,3})$ column by column using the following equations:

$$w_{r,0} = RotWord(SubWord(w_{r-1,3})) + w_{r-1,0} + RCON[r], \quad (7)$$
$$w_{r,i} = w_{r,i-1} + w_{r-1,i}, \quad for\ i = 1, 2, 3, \quad (8)$$

where r expresses the round number from 1 to 10 with K_0 being the input cipher key, $RCON[r]$ is the hexadecimal value $\{00,00,00,x^{r-1}\}$ and $w_{r,i}$ is the 32-bit word column i in K_r. The *SubWord* function in (7) performs the *SubBytes* transformation on each of the four bytes of the column $w_{r-1,3}$. The conventional hardware implementation of (7) and (8) is realized by the *SubWord* function followed by a successive *XOR*ing to calculate each column $w_{r,i}$ from the previous column $w_{r,i-1}$.

The proposed *KeyExpansion* architecture computes the successive *XOR*ing in parallel with the *SubWord* function as shown in Fig. 3. Consequently, the remaining part is only to *XOR* the output of *SubWord* function with all four columns in parallel. Therefore, the critical path delay is decreased by 3 *XOR* gates through the parallelization of the *KeyExpansion* steps compared to the conventional *KeyExpansion* structure in [5].

D. Sub-pipelining Architecture

The proposed AES round architecture adopts the sub-pipelining technique through the insertion of three-level registers to break down the critical path delay as shown in Fig.1. The placement of the registers is chosen such that the resulting stages delays are balanced while trying to optimize the number of pipelining registers used. Fig. 4 shows the complete sub-pipelined AES encryption round architecture. The vertical grey dashed lines represent the added sub-pipelining registers. The resulting stages delays are provided on the top of each stage respectively as illustrated in Fig. 4. Due to the proposed modifications in the *SubBytes* and the *KeyExpansion* modules, the presented sub-pipelined architecture minimizes the critical path delay to *6 XOR* gates as demonstrated in Fig. 4.

III. INTEGRATED AES ENCRYPTION/DECRYPTION ROUND ARCHITECTURE

The AES decryption process performs the inverse encryption transformations in the following reverse order: *AddRoundKey*, *InMixColumns*, *InvShiftRows*, *InvSubBytes*. Furthermore, the parallel *KeyExpansion* is executed in the reverse order starting from the final key value. In order to follow the same transformations order of the encryption procedure, the *AES* decryption is implemented using the equivalent Inverse cipher method [1]. Accordingly, hardware sharing techniques between each transformation and its corresponding inverse can be easily

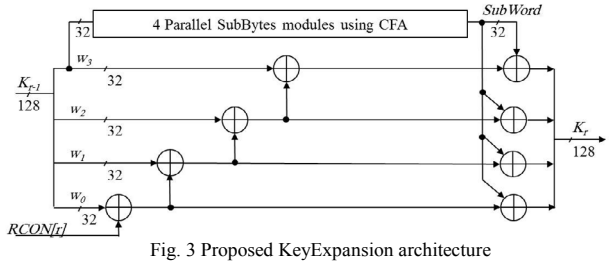

Fig. 3 Proposed KeyExpansion architecture

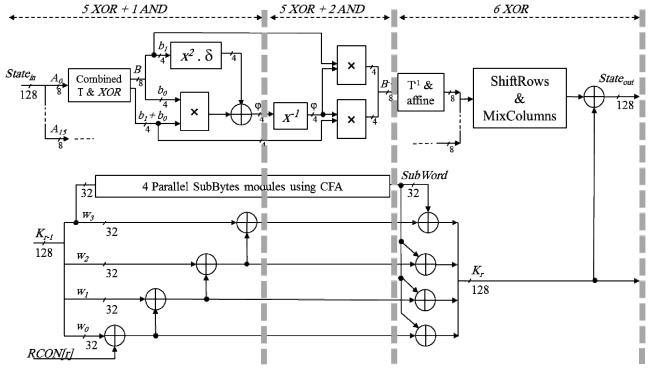

Fig. 4 Proposed AES encryption round architecture

applied. In the following subsections, the proposed integration of each two reciprocal operations is described. The *AddRoundKey* transformation is the inverse of itself.

A. Proposed SubBytes/InvSubBytes Implementation

The *SubBytes* and *InvSubBytes* operartions have in common the computation of the multiplicative inverse using CFA. However, they differ in the affine and inverse affine transforms required at the end of the *SubBytes* and the start of the *InvSubBytes* operations respectively. The architecture in Fig. 2 is adapted for the integrated *SubBytes*/*InvSubBytes* as shown in Fig. 5. Two extra blocks are added: the proposed combined block merged with the inverse affine transform at the beginning in addition to the inverse mapping block at the end. Moreover, two multiplexers are used to select between the encryption or the decryption paths by setting the selection lines to '*0*' or '*1*' respectively. For the combined block with the merged inverse affine transform, the bit level equations of the *4*-bit output $O = b_1 + b_0$ are derived in a similar way to (3), (4), (5) and (6).

B. Proposed ShiftRows/InvShiftRowsImplementation

The *InvShiftRows* transformation restores the shifted rows to their original places through a cyclic shift to the right by the same offset bytes of the *ShiftRows* operation. By examining the location of each byte in the state matrix following a *ShiftRows* or an *InShiftRows* operation the following facts are deduced:

- Row 1 is kept un-shifted in both cases,
- The offset shifting for row 3 is 2 bytes which makes it a symmetric shift whether to the left (*ShiftRows*) or to the right (*InvShiftRows*).
- Only rows 2 and 4 need multiplexing to select the corresponding signal wiring for *ShiftRows* or *InvShiftRows* based on the active mode of encryption or decryption respectively.

C. Proposed MixColumns/InvMixColumns Implementation

The *InvMixColumns* operation multiplies each column in the state matrix with the fixed polynomial $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$ modulo $x^4 + 1$. The polynomial $a^{-1}(x)$ can be decomposed into the *MixColumns* polynomial $a(x)$ provided in Section II.C, in addition to the two polynomials $m(x) = \{08\}(x^3 + x^2 + x + 1)$ and $n(x) = \{04\}\ (x^2 + 1)$ such that $a^{-1}(x) = a(x) + m(x) + n(x)$. Consequently, the multiplications required in the *InvMixColumns* module are reduced to either a multiplication by {02}, {08} or {04}. The architecture of the integrated modules, as shown in Fig. 5, consists of the *MixColumns* block parallel with the two blocks implementing multiplication by $m(x)$ and $n(x)$ respectively. A multiplexer is used to select either the output of the *MixColumns* block or the *XOR*ing of the outputs of the three blocks together.

D. Proposed Key Expansion for Integrated Design

The AES decryption process utilizes the same round key values however in the reverse order. Therefore, the decryption is only initiated when the last key value, K_{10} in case of 128-bit key, is available. Consequently, the *KeyExpansion* module starts to generate the round keys in a reverse manner using the reciprocal equations of (7) and (8). The on-the-fly respective round key K_{11-r-1} is generated from the available K_{11-r} during each decryption round r from 1 to 10. The utilized equivalent inverse cipher method implies the execution of the *InvMixColumns* on each round key before it enters the *AddRoundKey* transformation except for K_{10} and K_0 [1]. The proposed *KeyExpansion* architecture described in Section II.C is adapted, as shown in Fig. 5, to implement (7) and (8) as well as their reciprocals for encryption and decryption respectively. The proposed parallelization of the successive *XOR*ing with the *SubWord* computation improves the critical path delay by *3 XOR* gates and 2 multiplexers delays compared to the conventional *KeyExpansion* in [4].

E. Sub-pipelining Architecture

The sub-pipelining technique is applied on the AES integrated encryption/decryption round architecture by adding three levels of pipelining registers as shown in Fig. 5. The stages division is similar to the one shown in Fig. 4 for the AES encryption round. However, the *AddRoundKey* is moved from stage 3 to stage 1 of next round in order to optimize the balancing between the stages delays without extra registers cost. As illustrated in Fig. 5, stage 3 represents the critical path which consists of *7 XOR* gates + *3 MUX*s delays.

IV. Results and Comparisons

In this section, the performance of the proposed architectures is evaluated and compared with previously reported FPGA designs. The presented results are for the loop-unrolled approach where the proposed round architecture is duplicated through *10* successive units referring to the *10* rounds of the AES algorithm. Table I presents the results of the encryption architecture compared to the reported encryption designs in [9] and [5]. The proposed architecture operates at a maximum frequency of 453.45MHz providing a throughput of 58Gbps with 5,337 occupied slices on a Virtex-5 FPGA. Defining the efficiency performance metric as throughput per unit slice, the proposed architecture provides an efficiency of 10.88Mbps/Slice. In order to present a reliable comparison, the proposed architecture is implemented over similar devices for each reference. In [9], the LUT-based approach is adopted for the implementation of the *SubBytes*, while the *KeyExpansion* module is implemented using partial and dynamic reconfiguration. Furthermore, the AES round is divided into 4 sub-pipelining stages. However, the exploit of the sub-pipelining technique is not optimum due to the use of the storage LUTs with unbreakable delays for the *SubBytes* occupying alone 10,240 slices, equivalent to 80 BRAMs [9]. On the other hand, the 3 sub-pipelining stages design presented in [5] implements the *SubBytes* using CFA while the *KeyExpansion* is implemented using the conventional on-the-fly structure. As provided in Table I, the proposed architecture provides an efficiency improvement of 55% and

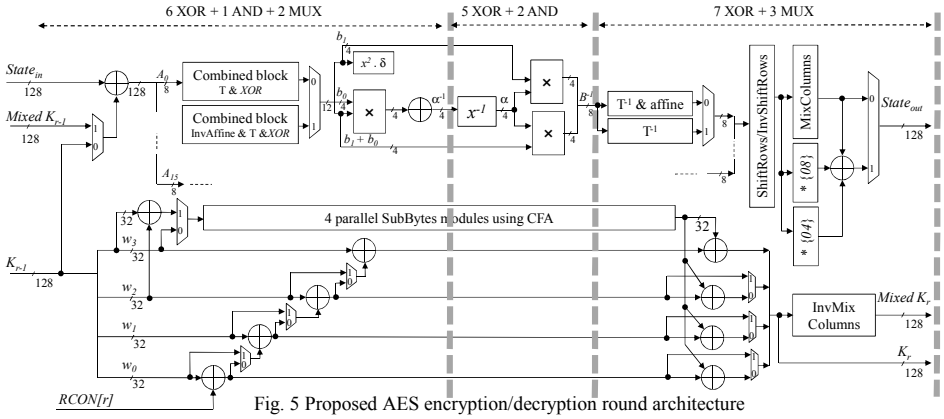

Fig. 5 Proposed AES encryption/decryption round architecture

TABLE I. RESULTS COMPARISON OF ENCRYPTION DESIGN

Parameters	XC5VLX50-3	XC2V6000-6		XCV800-6	
	This work	This work	[9]	This work	[5]
Freq.(MHz)	453.45	227.7	194.5	112.45	71.8
Throu.(Gbps)	58.04	29.15	24.9	14.4	9.184
Latency (ns)	68.36	136.15	210	275.7	431.75
Slices	5,337	10,025	13,816	9,483	9,406
Eff(Mbps/slice)	10.875	2.907	1.802	1.518	0.976

61% over the designs in [5] and [9] respectively with a higher throughput and lower latency.

Similarly, the performance of the proposed integrated encryption/ decryption design is analyzed and compared with the reported design in [6] as provided in Table II. The proposed architecture can operate at a maximum frequency of 294.8MHz providing a throughput of 37.7Gbps with an efficiency of 5.6Mbps on aVirtex-5 FPGA. Compared to the integrated encryption/decryption design in [6], the proposed architecture provides a 29% efficiency improvement with a higher throughput and lower latency.

TABLE II. RESULTS COMPARISON OF INTEGRATED ENCRYPTION/DECRYPTION DESIGN

Parameters	XC5VLX50-3	XC5VLX110T-1	
	This work	This work	[6]
Freq.(MHz)	294.8	232.9	202.3
Throu.(Gbps)	37.7	29.8	25.89
Latency (ns)	105.2	133.08	147
Slices	6,741	7,933	8,896
Eff(Mbps/slice)	5.6	3.758	2.9103

V. CONCLUSIONS

A high throughput efficient architecture is proposed for AES encryptor round and adapted for integrated AES encryptor/decryptor. The *SubBytes/InvSubBytes* operations are implemented using composite field arithmetic in order to exploit the sub-pipelining advantage on the loop-unrolling methodology. The proposed architecture allows the minimization of the critical path delay in both encryption and decryption processes. This is achieved through the proposed modifications in the *SubBytes/InvSubBytes* and *KeyExpansion* modules. The proposed encryption design provides a 58Gbps throughput with 5,337 slices on a Virtex-5 FPGA. On the other hand, the proposed integrated encryption/decryption design provides a 37.7Gbps throughput with slice occupation of 6,741 on a Virtex-5 FPGA. Compared to previously reported FPGA designs, the proposed architectures for the encryption and the integrated encryption/decryption provide an efficiency improvement of 61% and 29% respectively.

REFERENCES

[1] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," 2001.

[2] S. K. Mathew, et al. "53 Gbps native GF(24)2 composite field AES-encrypt/decrypt accelerator for content-protection in 45nm high-performance microprocessors," *IEEE Journal of Solid-State Circuits,* vol. 46, no. 4, pp. 767-776, April 2011.

[3] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient high performance parallel hardware architectures for the AES-GCM," *IEEE Transactions on Computers,* vol. 61, no. 8, pp. 1165-1178, August 2012.

[4] S.-F. Hsiaso, M.-C. Chen and C.-S. Tu, "Memory-free low cost designs of Advanced Encryption Standard using common subexpression elemination for subfunctions in transformations," *IEEE Transactions on Circuits and Systems,* vol. 53, no. 3, pp. 615-626, March 2006.

[5] X. Zhang and K. K. Parhi, "High speed VLSI architectures for the AES algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 12, no. 9, pp. 957-967, September 2004.

[6] S. K. Reddy S, R. Sakthivel and P. Praneeth, "VLSI implementation of AES crypto processor for high throughput," *International Journal of Advanced Engineering Sciences and Technologies,* vol. 6, no. 1, pp. 022-026, 2011.

[7] J. Chu and M. Benaissa, "Low area memory-free FPGA implementation of the AES algorithm," in *22nd International Conference on Field Programmable Logic and Applications*, pp. 623-626, August 2012.

[8] T. A. Pham, S. H. Mohammad and H. Yu, "Area and power optimisation for AES encryption module implmentation on FPGA," in *18th International Conference on Automation and Computing*, pp. 1-6, September 2012.

[9] J. M. Granado-Criado, M. A. Vega-Rodríguez, J. M. Sánchez-Pérez and J. A. Gómez-Pulido, "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," *Integartion, the VLSI Journal,* vol. 43, pp. 72-80, January 2010.

[10] K. Rahimunnisa, P. Karthigaikumar, S. Rasheed, J. Jayakumar and S. SureshKumar, "FPGA implementation of AES algorithm for high throughput using folded parallel architecture," *Journal of Security and Communication Networks,* vol. 5, no. 10, October 2012.

[11] M. El Maraghi, S. Hesham and M. A. Abd El Ghany, "Real-time efficient FPGA implementation of AES algorithm," in *26th Internation System on Chip Conference*, pp. 203-208, September 2013.

[12] M. Fayed, M. W. El-Kharashi and F. Gebali, "A high speed fully-pipelined VLSI architecture for real-time AES," in *International Conference on Information and Communications Technology*, December 2006.

[13] C. Paar, "Efficient VLSI architecture for bit-parallel computations in Galois field," Ph.D. dissertation, Essen, Germany, 1994.