International Journal of Research and Applications (July - Sep © 2014 Transactions) 1(3): 120-131
International Conference on Emerging Trends in Engineering & Technology (ICETET-2014)
ISSN (online): 2349-0020
ELECTRONICS RESEARCH ARTICLE

# Implementation of AES-GCM encryption algorithm for high performance and low power architecture Using FPGA

**V. Arun [1], K. Vanisree [2] and D. Laxma Reddy [3]**
[1]Asst.Prof.ECE,MLRIT,Dundigal, Hyderabad-500043,Telangana
[2]Assoc. Prof. ECE, ACE College of Engineering, R. R. Dt.- 501 301,Telangana
[3] Asst.Prof.ECE,MLRIT,Dundigal, Hyderabad-500043,Telangana

**ABSTRACT**

Evaluation of the Advanced Encryption Standard (AES) algorithm in FPGA is proposed here. This Evaluation is compared with other works to show the efficiency. Here we are concerned about two major purposes. The first is to define some of the terms and concepts behind basic cryptographic methods, and to offer a way to compare the myriad cryptographic schemes in use today. The second is to provide some real examples of cryptography in use today. The design uses an iterative looping approach with block and key size of 128 bits, lookup table implementation of S-box. This gives low complexity architecture and easily achieves low latency as well as high throughput. Simulation results, performance results are presented and compared with previous reported designs. Since its acceptance as the adopted symmetric-key algorithm, the Advanced Encryption Standard (AES) and its recently standardized authentication Galois/Counter Mode (GCM) have been utilized in various security-constrained applications. Many of the AES-GCM applications are power and resource constrained and requires efficient hardware implementations. In this project, AES-GCM algorithms are evaluated and optimized to identify the high-performance and low-power architectures. The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data. The Cipher Block Chaining (CBC) mode is a confidentiality mode whose encryption process features the combining ("chaining") of the plaintext blocks with the previous Cipher text blocks. The CBC mode requires an IV to combine with the first plaintext block. The IV need not be secret, but it must be unpredictable. Also, the integrity of the IV should be protected. Galois/Counter Mode (GCM) is a block cipher mode of operation that uses universal hashing over a binary Galois field to provide authenticated encryption. Galois Hash is used for authentication, and the Advanced Encryption Standard (AES) block cipher is used for encryption in counter mode of operation. To obtain the least-complexity S-box, the formulations for the Galois Field (GF) sub-field inversions in GF ($2^4$) are optimized By conducting exhaustive simulations for the input transitions, we analyze the synthesis of the AES S-boxes considering the switching activities, gate-level net lists, and parasitic information. Finally, by implementation of AES-GCM the high-performance GF ($2^{128}$) multiplier architectures, gives the detailed information of its performance. An optimized coding for the implementation of Advanced Encryption Standard-Galois Counter Mode has been developed. The speed factor of the algorithm implementation has been targeted and a software code in Verilog HDL has been developed. This implementation is useful in wireless security like military communication and mobile telephony where there is a grayer emphasis on the speed of communication.

Index Terms— Cipher block chaining, GaliosField, Advanced Encryption Standard, finite field, Galois/Counter Mode, high performance.

## I. INTRODUCTION

Data Encryption Standard (DES) is the most common SKC scheme used today; DES was designed by IBM in the 1970s and adopted by the National Bureau of Standards (NBS) [now the National Institute for Standards and Technology (NIST)] in 1977 for commercial and unclassified government applications. DES is a block-cipher employing a 56-bit key that operates on 64-bit blocks. Symmetric-key ciphers use the same key for encryption and decryption, or to be more precise, the key used for decryption is computationally easy to compute given the key used for encryption. In turn, symmetric-key

ciphers fall into two categories: block ciphers and stream ciphers. Stream ciphers encrypt the plaintext one bit at a time, in contrast to block ciphers, which operate on a block of bits of a predefined length. Most popular block ciphers are DES, IDEA and AES, and most popular stream cipher is RC6.DES has a complex set of rules and transformations that were designed specifically to yield fast hardware implementations and slow software implementations The Advanced Encryption Standard-Galois/Counter Mode AES-GCM) provides authentication and confidentiality for sensitive data simultaneously. In the AESGCM, data confidentiality is provided by the Advanced Encryption Standard (AES). This paper explores the area-throughput trade-off for an ASIC implementation of the Advanced Encryption Standard (AES). Different pipelined implementations of the AES algorithm as well as the design decisions and the area optimizations that lead to a low area and high throughput AES encryption processor are presented. With loop unrolling and outer-round pipelining techniques, throughputs of 30 Gbits/s to 70 Gbits/s are achievable in a 0.18-_m CMOS technology. Moreover, by pipelining the composite field implementation of the byte substitution phase of the AES algorithm (inner-round pipelining), the area consumption is reduced up to 35 percent. By designing an offline key scheduling unit for the AES processor the area cost is further reduced by 28 percent, which results in a total reduction of 48 percent while the same throughput is maintained. Therefore, the over 30 Gbits/s, fully pipelined AES processor operating in the counter mode of operation can be used for the encryption of data on optical links. The authentication of the AES-GCM is provided by the Galois/Counter Mode (GCM) using a universal hash function. The AES-GCM has been used for a number of applications such as the new LAN security standard WLAN 802.1ae (MACSec) and Fibre Channel Security Protocols (FC-SP). Moreover, it has been utilized in a number of cores from industry. In addition, two AES-GCM software-based implementations have been presented. Among the transformations in the AES encryption, the SubBytes (S-boxes) is the only non-linear one, requiring the highest area and consuming much of the AES power. Therefore, the performance metrics of the S-boxes affect those for the entire AES encryption significantly. For low-complexity implementations, the S-box can be realized using logic gates in composite fields. These S boxes can also be pipelined for achieving high performance. On the other hand, the S-boxes based on look-up tables (LUTs) could be area-efficient when implemented utilizing the memory resources available on FPGAs. In this paper, logic-gate optimizes and comprehensive synthesis of more than 40 different S-boxes are used for deriving their performance metrics. This paper presents the area-throughput trade-offs of a fully pipelined, ultra high speed AES encryption processor. Different pipelined architectures that can achieve the required throughput for the above application and the area optimization opportunities for such designs are explored.

**II. Galois/Counter Mode of Operation (GCM)**

By definition, the Galois/Counter Mode is a block cipher mode of operation that uses universal hashing over a binary Galois field whose purpose is to provide authenticated encryption. This paper proposes an efficient solution to combine Rijndael encryption and decryption in one FPGA design, with a strong focus on low area constraints. The proposed design gets into the smallest Xilinx FPGAs3, deals with data streams of 208 Mbps, uses 163 slices and 3 RAM blocks and improves by 68% the best-known similar designs in terms of ratio Throughput=Area. They also proposed implementations in other FPGA Families (Xilinx Vertex-II) and comparisons with similar DES, triple-DES and AES implementations. It can achieve data rates of 21.3 Gbps in Vertex-II FPGAs. The encryption/decryption mode can be changed on a cycle-by-cycle basis with no dead cycles. For the AES, the best similar RAM-based solution unrolls the 10 cipher rounds and pipelines them in an encryption-only process. This implementation in a Vertex-E FPGA produces a throughput of 11.8 Gbps and allows the key to be changed at every cycle. This DES implementation reaches higher throughput than the corresponding

AES implementation. The input and output for the AES algorithm each consist of sequences of 128 bits (digits with values of 0 or 1). These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits. Other input, output and Cipher Key lengths are not permitted by this standard. The bits within such sequences will be numbered starting at zero and ending at one less than the sequence length (block length or key length). The number i attached to a bit is known as its index and will be in one of the ranges $0 \le i < 128$, $0 \le i < 192$ or $0 \le i < 256$ depending on the block length and key For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by Nb = 4, which reflects the number of 32-bit words (number of columns) in the State. For the AES algorithm, the length of the Cipher Key, K, is 128, 192, or 256 bits. The key length is represented by Nk = 4, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8.T.

**A. GCM Encryption and Decryption - Inputs and Outputs**

Encryption**:** The GCM encryption routine expects four inputs:

- A secret key K, to be used with the underlying block cipher. AES is defined to support key lengths of 128-, 192- or 256-bits long.
- An initialization vector IV that (in principle) can be of any length between 1 and $2^{64}$ bits.
- A plaintext P that can be of any length between 0 and ($2^{39}$◎ 256) bits.
- Additional authenticated data AAD that can be of any length between 0 and $2^{64}$ bits.

This procedure has two outputs:

- A cipher text C that has the same length as the input plaintext P.
- An authentication tag T, which in our case is of length exactly 128 bits.

Below we denote the GCM encryption routine (using AES) by

(C,T) := GCM-AES-enc(K; IV, P, AAD)

The security of GCM relies on the secret key being secret, and on the IV being used as a nonce. That is, GCM only offers security as long as the same value for the IV is never used for encryption of more than one plaintext under the same key.

Decryption**:** The GCM decryption routine has five inputs:

- the key K,
- initialization vector IV,
- cipher text C,
- additional authenticated data AAD, and
- tag T, all as above.
-

Its output is either the plaintext P as above, or the special signal fail (indicates that the inputs are not authentic). Below we denote the GCM decryption routine (using AES) by

P/fail: = GCM-AES-dec (K; IV, C, AAD, T)

A cipher text C, initialization vector IV, additional authenticated data A and tag T are authentic for key K when they are generated by the encrypt operation with inputs K, IV, A and P, for some plaintext P. The authenticated decrypt operation will, with high probability, return **FAIL** whenever its inputs were not created by the encrypt operation with the identical key. The additional authenticated data A is used to protect information that needs to be authenticated, but which must be left unencrypted. When using GCM to secure a network protocol, this input could include addresses, ports, sequence numbers, protocol version numbers, and other fields that indicate how the plaintext should be handled, forwarded, or processed. In many situations, it is desirable to authenticate these fields, though they must be left in the clear to allow the network or system to function

properly. When this data is included in the AAD, authentication is provided without copying the data into the cipher text. The primary purpose of the IV is to be a nonce, that is, to be distinct for each invocation of the encryption operation for a fixed key. It is acceptable for the IV to be generated randomly, as long as the distinctness of the IV values is highly likely. The IV is authenticated, and it is not necessary to include it in the AAD field. Both confidentiality and message authentication is provided on the plaintext. The strength of the authentication of P, IV and A is determined by the length t of the authentication tag. When the length of P is zero, GCM acts as a MAC on the input A. The mode of operation that uses GCM as a stand-alone message authentication code is denoted as GMAC.

**B. ENCRYPTION**

Let n and u denote the unique pair of positive integers such that the total number of bits in the plaintext is (n − 1)128 + u, where 1 ≤ u ≤ 128. The plaintext consists of a sequence of n bit strings, in which the bit length of the last bit string is u, and the bit length of the other bit strings is 128. The sequence is denoted $P_1, P_2 \ldots P_{n-1}, P^*_n$, and the bit strings are called data blocks, although the last bit string, $P^*_n$ , may not be a complete block. Similarly, the cipher text is denoted as $C_1, C_2 \ldots C_{n-1}, C^*_n$, where the number of bits in the final block $C^*_n$ is u. The additional authenticated data A is denoted as $A_1, A_2 \ldots A_{m-1}, A^*_m$ , where the last bit string $A^*_m$ may be a partial block of length v, and m and v denote the unique pair of positive integers such that the total number of bits in A is (m − 1)128 + v and 1 ≤ v ≤ 128. The authenticated encryption operation is defined by the following equations:

$$H = E(K, 0^{128})$$

$$Y0 = IV \,||\, \begin{cases} 0^{31}1 & \text{if len (IV)} = 96 \\ GHASH(H, \{\}, IV) & \text{Otherwise.} \end{cases}$$

$$Yi = incr(Y_{i-1}) \text{ for } i = 1, \ldots, n \qquad (1)$$

$$C_i = P_i \oplus E(K, Y_i) \qquad \text{for } i = 1 \ldots n-1$$

$$C^*_n = P^*_n \oplus MSB_u(E(K, Y_n))$$

$$T = MSB_t(GHASH(H, A, C) \oplus E(K, Y_0))$$

Successive counter values are generated using the function incr(), which treats the rightmost 32 bits of its argument as a nonnegative integer with the least significant bit on the right, and increments this value modulo $2^{32}$. More formally, the value of incr(F || I) is F || (I + 1 mod 232). The encryption process is illustrated in Figure 1.

**C. GCM encryption operation**

As the name suggests, GCM mode combines the well-known counter mode of encryption with the new Galois mode of authentication. The key feature is that the Galois field multiplication used for authentication can be easily computed in parallel thus permitting higher throughput than the authentication algorithms that use chaining modes, like CBC. The GF ($2^{128}$) field used is defined by the polynomial $x^{128}+x^7+x^2+1$. The function GHASH is defined by GHASH (H, A, C) = $X_{m+n+1}$, where H is a string of 128 zeros encrypted using the block cipher, A is data which is only authenticated (not encrypted), C is the cipher text, m is the number of 128 bit blocks in A, n is the number of 128 bit blocks in C (the final blocks of A and C need not be exactly 128 bits), and the variable $X_i$ for i = 0, ..., m + n + 1 is defined as

$$Xi = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1 \ldots m-1 \\ (X_{m-1} \oplus (A^*_m \,||\, 0^{128-v})) \cdot H & \text{for } i = m \\ (Xi-1 \oplus C_i) \cdot H & \text{for } i = m+1 \ldots m+n-1 \end{cases} \qquad (2)$$

$$(X_{m+n-1} \oplus (C_m^* \,||\, 0^{128-u})) \cdot H$$

for i = m + n

$$(X_{m+n} \oplus (\text{len}(A) \,||\, \text{len}(C))) \cdot H$$

for i = m + n + 1.

where v is the bit length of the final block of A, u is the bit length of the final block of C, and || denotes concatenation of bit strings.

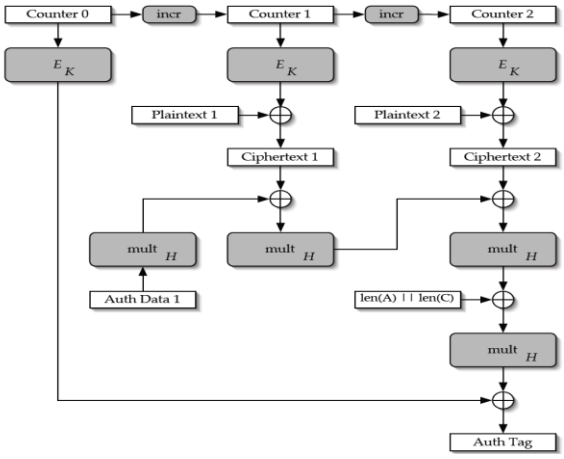



**Figure 1**: The authenticated encryption operation.

For simplicity, a case with only a single block of additional authenticated data (labeled Auth Data 1) and two blocks of plaintext is shown. Here $E_K$ denotes the block cipher encryption using the key K, $mult_H$ denotes multiplication in GF ($2^{128}$) by the hash key H, and incr denotes the counter increment function.

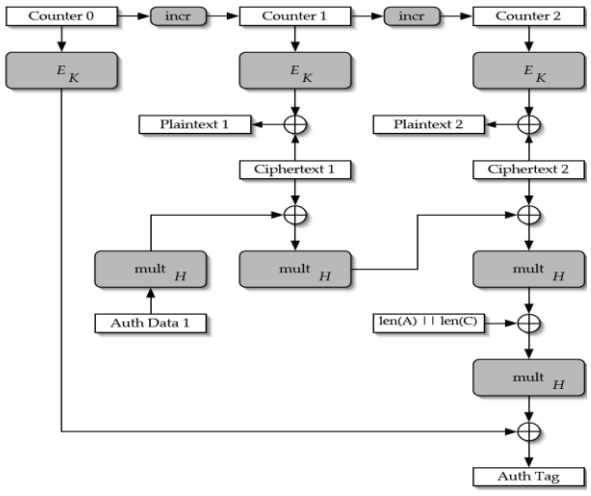



**Figure 2**: The authenticated decryption operation, showing the same case as in Figure 1.

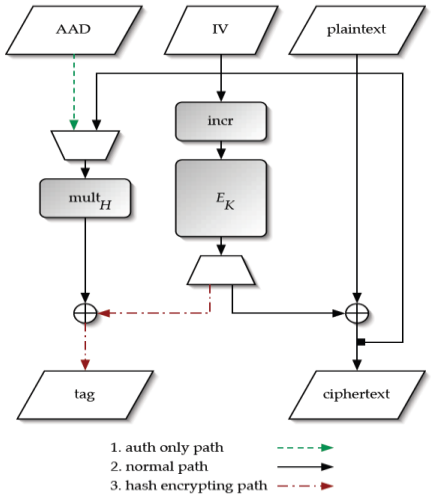



**Figure 3** :A hardware implementation of GCM, showing the different data paths through the circuit.

### D. MULTIPLICATION IN GF ($2^{128}$)

The multiplication operation is defined as an operation on bit vectors in order to simplify the specification. This definition corresponds to the particular choice of the field representation used in GCM. Each element is a vector of 128 bits. The $i^{th}$ bit of an element X is denoted as $X_i$. The leftmost bit is $X_0$, and the rightmost bit is $X_{127}$. The multiplication operation uses the special element R = 11100001 || $0^{120}$, and is defined in Algorithm 1. The function right shift () moves the bits of its 7

**Algorithm 1** Multiplication in GF ($2^{128}$). Computes the value of Z = X · Y, where X, Y and

Z ∈ GF($2^{128}$).

Z ← 0, V ← X
**for** i = 0 to 127 **do**
  **if** Yi = 1 **then**
  Z ← Z ⊕ V
**end if**
**if** $V_{127}$ = 0 **then**
  V ← right shift (V)
**else**
V ← right shift(V ) ⊕ R
**end if**
**end for**
**return** Z

argument one bit to the right. More formally, whenever W = right shift (V ), then $W_i = V_{i-1}$ for $1 \le i \le 127$ and $W_0 = 0$.

Authenticated encryption and decryption are the two functions within the GCM. The authenticated encryption performs two tasks; encrypting the confidential data and computing and authentication tag. The authenticated decryption function decrypts the confidential data and verifies the tag. The data flow of the authenticated encryption is shown in Fig. 3. As seen in this figure, the mechanism for the confidentiality of data is a variation of the block cipher counter mode of operation, denoted by $GCTR_K$ (Galois Counter with the key K). Then, the function $GCTR_K$ performs the block cipher counter mode with the Initial Counter Block (ICB) and its increments ($CB_2 - CB_i$) and the plaintext blocks ($P_1 - P_i$) as the inputs.

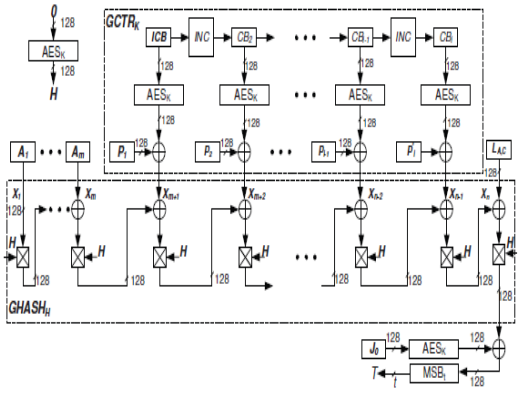



**Figure 4** The GCM authenticated encryption data flow.

Galois Hash ($GHASH_H$) function is constructed by GF ($2^{128}$) multiplications with a fixed parameter, called the hash subkey (H). The $GHASH_H$ function calculates

$$\sum_{j=1}^{n} X_j H^{n-j+1} = X_1 \cdot H^n \oplus X_2 \cdot H^{n-1} \oplus \ldots \oplus X_n \cdot H, \quad (1)$$

where $X_1$ to $X_n$ are the n, 128-bit blocks of the input. It is noted that the hash subkey is generated by applying the AES to the zero block, i.e., 0 = (0, 0... 0) ∈ GF ($2^{128}$). Then, the $GHASH_H$ function calculates (1). All the arithmetic operations in (1), i.e., additions, GF multiplications, and exponentiations are performed over GF ($2^{128}$) constructed by the irreducible polynomial $P(x) = x^{128} + x^7 + x^2 + x + 1$. As seen in Fig. 3.2, the total number of input blocks to $GHASH_H$ is n = m + i + 1, where m and i are the number of blocks for the additional authenticated data ($A_1 - A_m$) and the output of $GCTR_K$, respectively. Eventually, the authentication tag T with length of t bits is derived. In the authenticated decryption, the same $GHASH_H$ procedure is performed on the authenticated data and ciphertext blocks to verify the tag.

## III. HIGH PERFORMANCE GCM PARALLEL ARCHITECTURE

High-performance parallel architectures for GCM improve the throughput and the latency of the structures for $GHASH_H$. They also remove the need for consecutive GF ($2^{128}$) multiplications with H for deriving (1). Because of the low complexity of the implementations of these exponents, we take advantage of these low-cost hash subkey powers in the proposed high-performance architectures. We utilize the powers in the form of $H^{2^j}$ to obtain the other powers of the hash subkey with the least number of GF multiplications over GF ($2^{128}$) for proposed architectures. For instance, we derive $H+ = H^2 \times H$ or $H^6 = H^4 \times H^2$. This architecture is based on the composite field GF (($2^4$)$^2$). Algorithm 1 is used for obtaining the key formulation for the proposed GHASHH function. Although there is no restriction in choosing q, i.e., the number of parallel adder-multipliers, we use $q = 2^j$, $1 \le j \le \log_2(n)$. This leads to lower number of clock cycles and higher throughput needed for the implementations.

**Algorithm 1** The proposed high-performance approach for implementing the GCM.

Inputs: $X_p \in$ GF ($2^{128}$), $1 \le p \le n$, and $H2j \in$ GF ($2^{128}$),

$0 \le j \le \log_2(q)$.

Output: $GHASH(X, H) = \Sigma^{n}_{j-1} X_j H^{n-j+1}$.

1: **for** i = 1 to q **do**

2: tempi ← Xi

3: **for** j = 1 to (n/q) − 1 **do**

4: tempi = (tempi × $H_q \oplus X_{i+jq}$)

5: **end for**

6: Let q − i + 1 = ($a_0^{(i)}$, . . . , $alog_2(q)^{(i)}$)$_2$

7: tempi = tempi × ($Ha_0^{(i)q}$ ×H2 ×. . .×$Ha^{(i)}log_2(q)$)

8: **end for**

9: GHASH(X, H) =$\Sigma^{q}_{i-1}$ tempi

10: **return** GHASH(X, H).

In Algorithm1, the output GHASH(X,H) is obtained as follows:

$X1 \cdot H^q \times \ldots \times H^q \oplus X_2 \cdot H^q \times \ldots \times H^q \times H^{q-1} \oplus \ldots$

$\oplus X_j \cdot H_q \times \ldots \times H_q \times H_{q-j+1} \oplus \ldots$

$\oplus X_q \cdot H_q \times \ldots \times H_q \ldots \oplus X_n H,$ (1)

where all operations are performed over GF($2^{128}$) constructed by the irreducible polynomial $P(x) = x^{128} + x^7 + x^2 + x + 1$ and ⊕ comprises 128 XOR gates.

One can re-write (1) so that only the exponentiations of the hash subkey to the powers of 2 in the form of $H^{2j}$ are utilized. This method of exponentiation is based on the binary exponentiation. As seen from this algorithm, for the exponentiations $H^{q-i+1}$, 1 ≤ i ≤ q, one can use the binary representation of q − i + 1 as ($a_0$ (i), . . . , $alog_2(q)^{(i)}$)$_2$ . The hardware implementation of Algorithm 1 has been presented in Fig. 4.1. For implementing Algorithm 1 in hardware, in total, (n/q) +$log_2$ (q) clock cycles are needed. For the first (n/q) −1 clock cycles, the GF ($2^{128}$) multiplications by $H^q$ are performed. This is achieved by a simple control unit selecting $H^q$. Then, for the next $log_2$ (q) clock cycles, the other exponentiations are used. These include the powers of the hash subkey in the form of $H^{2j}$ and a number of field elements 1 = (0, 0... 1) ∈ GF ($2^{128}$) for bypassing the GF ($2^{128}$) multiplication operations. We note that if n is not a multiple of q, one need to add q − mod (n, q) blocks containing 0 = (0, 0... 0) ∈ GF ($2^{128}$) to the beginning of the n blocks to make the total blocks processed multiple of q. Performing this, the hash computation can be done normally based on the presented procedure. Finally, in one clock cycle, the result becomes$\Sigma^{n}_{j-1} X_j H^{n-j+1}$.

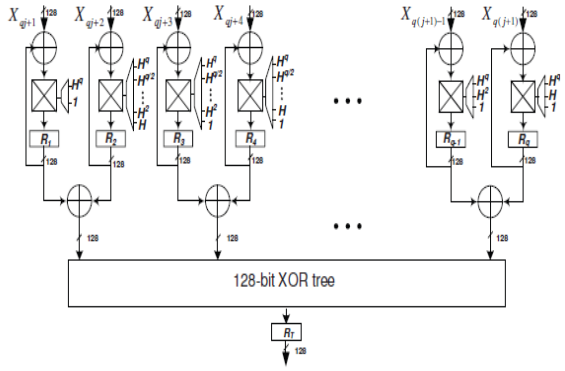



Figure 5. The hardware architecture of the proposed high performance GCM $GHASH_H$ function

**GF ($2^{128}$) Multipliers for the GCM -** Different types of GF ($2^{128}$) multipliers are utilized in the literature for implementing the GF ($2^{128}$) multiplications in (1). The multiplications have been performed using bit-parallel, digit-serial, and hybrid multipliers in composite fields. The efficiency of different multipliers, including the sub-quadratic ones, are compared. A high-speed AES-GCM core has been presented. It is noted that the considered GF ($2^{128}$) multipliers in these works include the Mastrovito multiplier with quadratic space complexity, the Karatsuba-Ofman multiplier and the GF ($2^{128}$) multiplier. We have considered the bit-parallel GF ($2^{128}$) multiplier which has quadratic hardware complexity. It is noted that this GF ($2^{128}$) multiplier has lower timing complexity compared to the sub-quadratic hardware complexity GF ($2^{128}$) multipliers. However, we note that according to the latency of the proposed architectures, i.e., (n/q) + $log_2$ (q), increasing the number of parallel structures (q) results in having higher throughputs. Fig.5presents the proposed architecture for the AESGCM for q = 8 parallel structures. The AES-128 pipeline registers are shown by dashed lines in Fig. 4.5. As seen in this figure, 10 clock cycles are needed for obtaining the cipher text. After these first 10 clock cycles, the results are obtained after each clock cycle. According to Fig. 5, 8

parallel AES-128 structures are implemented as part of $GCTR_K$ to provide inputs to $GHASH_H$. As seen in this figure, the function $GCTR_K$ performs the AES counter mode with the Initial Counter Block (ICB) and its one-increments ($CB_i$). Moreover, q = 8 increments (using INC 8 module) and the plaintext blocks ($P_i$) are used as the inputs. It is assumed that the data is encrypted and the IV in the GCM is 96 bits which is recommended for high throughput implementation

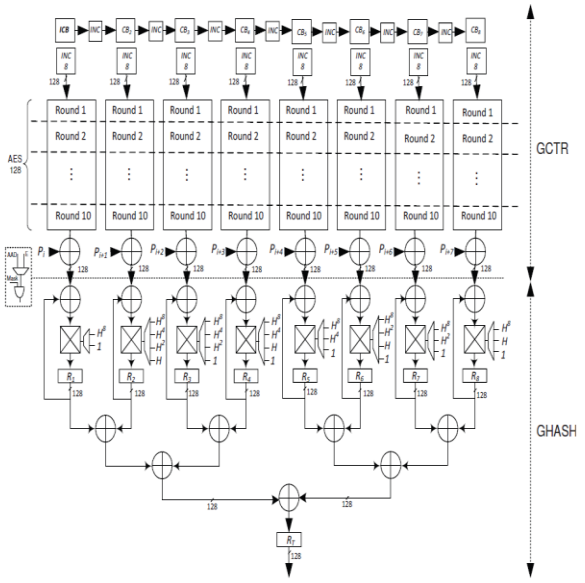



Figure 6. The proposed AES-GCM high-performance architecture for q = 8 (mod (n, q) = 0).

The architecture shown in Fig. 5assumes that the number of blocks n is a multiple of the number of parallel structures q and there is no additional authenticated data (AAD). In case that n is not a multiple of q, one can append q − mod (n, q) zero blocks at the beginning of the blocks for which hash is computed. This is done by adding a masking gate along the dotted line as shown in Fig. 4.5. Moreover, in this case, the counter blocks and accordingly $P_i$'s in Fig. 5start from the q−mod (n, q) +1 column, i.e., the first actual input block. We also note that in case AAD is present, additional multiplexers are placed at the output of the GCTR block in Fig. 4.5along the dotted line so that instead of encrypted data, the AAD is fed to the architecture. When the AAD is done, the counter blocks provide the encrypted data. Finally, in Fig. 5and as the last processed block, the output of the GCTR block in the rightmost column is masked and $L_{A,\,C}$ (number for n) is fed (using an extra multiplexer which is not shown in Fig.5for the sake of brevity). $AES_K(J_0)$ and $H = AES_K(0)$ can be also obtained or pre-computed in Fig. 5. The results of our synthesis for the AES-GCM using the FPGA Vertex Xilinx tool are shown in the results The synthesis are based on the case for q = 8 parallel addition-multiplications using the bit-parallel GF ($2^{128}$) multiplier which has quadratic hardware complexity. For achieving low hardware complexity for the AES-GCM, we have also synthesized six different steps for the Karatsuba-Ofman multipliers.

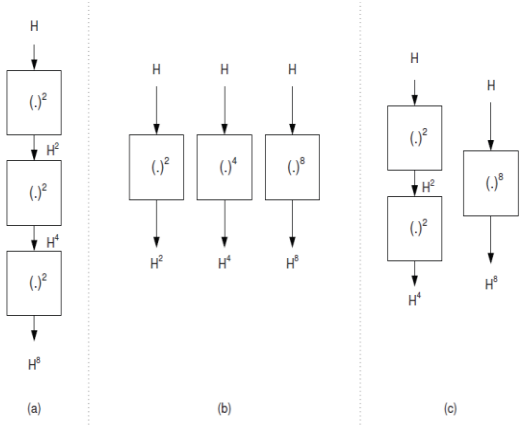



Figure 7(a) Cascade, (b) parallel, and (c) hybrid realization methods for hashSub key exponentiations

## IV. GCM-AES BLOCK SPECIFICATION

GCM-AES (Galois Counter Mode – Advanced Encryption Standard) is an authenticated encryption mode designed by David McGrew and John Viega. This aims to explore hardware implementation of GCM-AES mode of operation specifically targeting FPGA (Field Programmable Gate Arrays). The aim of such an implementation is to benchmark GCM-AES on FPGA in terms of area, power and speed. GCM-AES has been implemented as a full duplex block which means that the design consists of separate encryption-authentication and decryption-verification blocks. Thus, it can carry out encryption-authentication and decryption-verification operations simultaneously.

**Encryption and Authentication Block**

- GCM-AES encryption block works on one single frame (Message + AAD) at any given time. A frame consists of one or more AAD blocks or zero or more message blocks. Specifically, the

encryption block works on one message block or AAD block at any time.

- The default block length is 128 bits.
- A single control word starts the operation of GCM-AES encryption block with the Setup phase. This phase is done once per frame. After twenty clock cycles of latency incurred from the setup phase, the encryption block is ready to accept a message or AAD block.
- The encryption block expects one or more AAD blocks where the last AAD's block length need not be 128 bits. It should however be a multiple of a byte. Similarly, the encryption block expects zero or more message blocks where the last message block length need not be the default block length. It should as well be a multiple of a byte.
- The design requires one or more AAD blocks to be input first and then zero or more plain text blocks.
- The current implementation is capable of handling any message or AAD blocks per frame. It takes 10 clock cycles to encrypt a message block (default block length or less than that) with 10 cycle AES-128 implementation when the corresponding encrypted cipher text is produced. The GCM-AES encryption block relies on AES-128 encryption block for encryption and Galois Field multiplication for authentication. Galois Field Multiplier used in this implementation produces result in 8 clock cycles. Cipher text is not produced in case of AAD block.
- The length of the frame does not need to be known by the encryption block.
  The encryption block works on a frame with the following format:

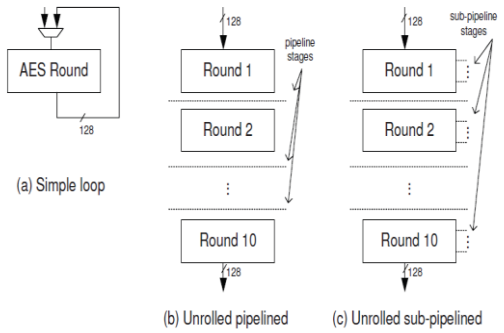



Figure 7. The AES-128 structure for (a) simple loop, (b) unrolled pipelined, and (c) unrolled sub-pipelined architectures

**Decryption and Verification Block**

1 The GCM-AES decryption block is similar to GCM-AES encryption block. Thanks to Counter Mode. Tag calculation is exactly the same as GCM-AES encryption. The computed tag is compared against the provided tag. If the tags match, decrypted plain texts are considered valid.

2 The GCM-AES decryption block works on the similar frame format as GCM-AES encryption block where now Message Blocks are replaced by zero or more Cipher text blocks.

**GCMAES Encryption and Authentication**

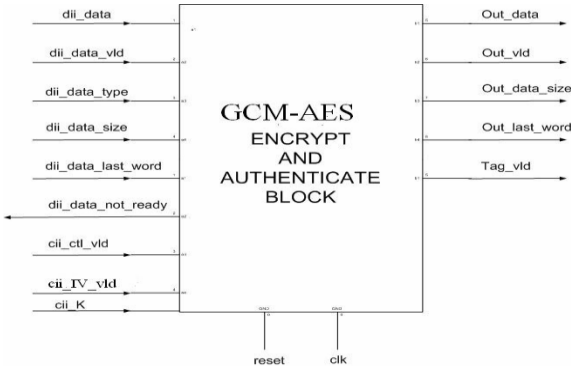



**Figure 8** Interface diagram GCM-AES Encryption block

GCM-AES design as described is coded in Verilog hardware descriptive language HDL. All simulations are done in Modelsim-Altera 6.5e (Quartus II 10.0sp1) Starter Edition Modelsim. XILINX ISE 9.2 is used for FPGA design flow using VirtexE technology (Family = VirtexE, Device = XCv400e, Package = bg560, Speed = -8).

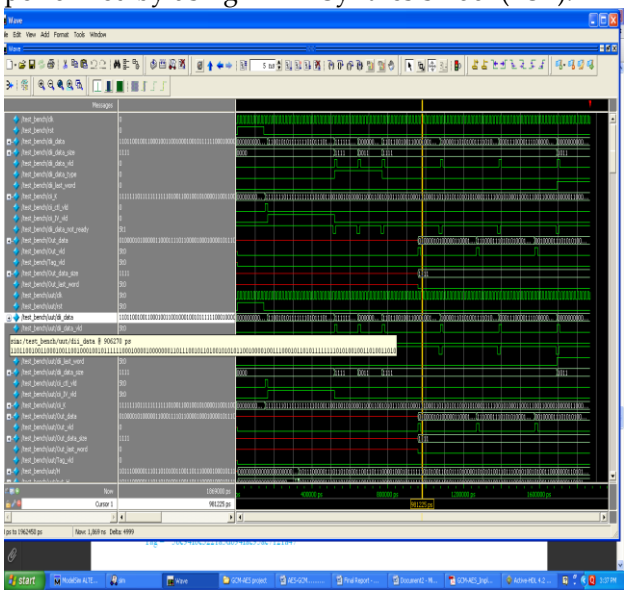
## V. RESULTS

We used test case for GCM-AES Encryption-Auth simulations. It is reproduced here for convenience

K = feffe9928665731c6d6a8f9467308308
P = d9313225f88406e5a55909c5aff5269a
86a7a9531534f7da2e4c303d8a318a72
1c3c0c95956809532fcf0e2449a6b525
b16aedf5aa0de657ba637b39A
AAD= feedfacedeadbeeffeedfacedeadbeefabaddad2
IV = cafebabefacedbaddecaf888
The corresponding cipher text and Tag is
CT = 42831ec2217774244b7221b784d0d49c
e3aa212f2c02a4e035c17e2329aca12e

21d514b25466931c7d8f6a5aac84aa05
1ba30b396a0aac973d58e091

All simulations are done in Mentor Graphic's Modelsim. The design is mapped to FPGA belonging to VirtexE technology (Family = VirtexE, Device = XCv400e, Package = bg560, Speed = -8). Synthesis is performed by using Xilinx Synthesis Tool (XST).



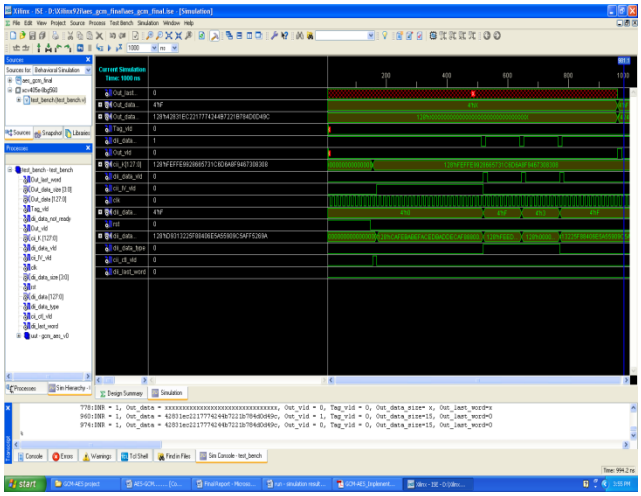


Figure 8 Simulation of GCM-AES Enc-Auth block

Figure 8 shows simulation of GCM-AES Encryption-Authentication block. Decryption-Verify block is exactly the same. Thus, only simulation of Encryption-Authentication block suffices.

The synthesis report generated is shown below.

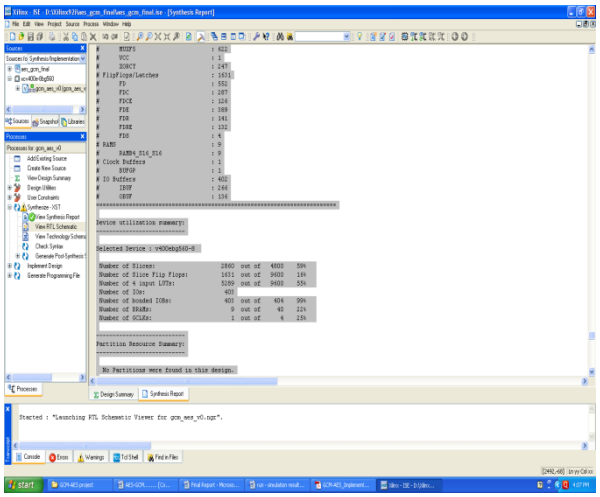


**Figure 9** Synthesis report of GCM-AES Enc-Auth block

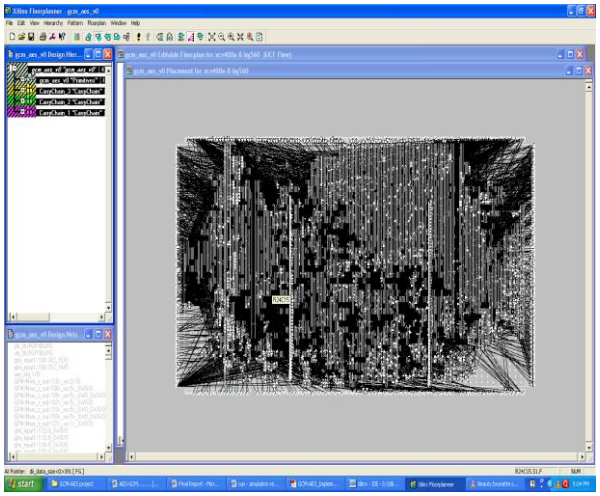


**Figure 10** Floor plan design of GCM-AES Enc-Auth block

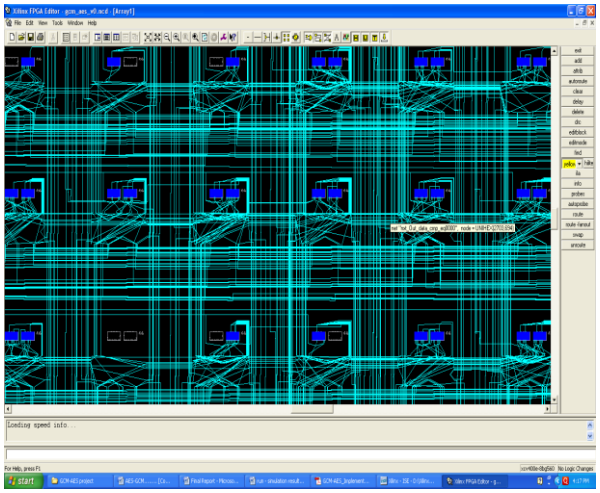


**Figure 11** FPGA Schematic of GCM-AES Enc-Auth block

**Table 1** Description of GCM-AES Encryption block

| | | | |
|---|---|---|---|
| dii_data<br><br>dii stands for data input interface. | Input | 128 | Data input that is either Nonce, message or AAD block |
| dii_data_vld | Input | 1 | When asserted (=1), dii_data contains either message or AAD block |
| dii_data_type | Input | 1 | When asserted, dii_data contains AAD block. When deasserted (= 0), dii_data contains message block |
| dii_data_size | Input | 4 | Describes the size of valid dii_data. It ranges from 0-15 where 0 indicates valid message consists of 1 byte in the LSB of dii_data and 15 indicate full block length. Its value may change from 15 on the last message or AAD block |
| dii_data_last_word | Input | 1 | When asserted, dii_data contains the last message or AAD block |
| dii_data_not_ready | Output | 1 | It is asserted by the GCM-AES indicating that it currently in the Setup phase or working with one message or AAD block and cannot accept an additional message or AAD block. |
| cii_ctl_vld<br><br>cii stands for control input interface | Input | 1 | When asserted, starts the execution of GCM-AES encryption block and triggers Setup phase |
| cii_IV_vld | Input | 1 | When asserted, dii_data contains IV value |
| cii_K | Input | 128 | It contains secret key used in GCM-AES block |
| Out_data | Output | 128 | It contains either the cipher text or Tag_data |
| Out_vld | Output | 1 | When asserted, it indicates Out_data contains cipher text |
| Out_data_size | Output | 4 | It describes the number valid bytes in Out_data |
| Out_last_word | Output | 1 | It describes whether the cipher text is the last cipher text |
| Tag_vld | Output | 1 | When asserted, Out_data contains Tag data. |

| PIN | Direction | Size (bits) | Description |
|---|---|---|---|
| clk | Input | 1 | Design clock |
| reset | Input | 1 | Design reset |

**VI CONCLUSION and FUTURE SCENARIO**

In this project, we have obtained optimized building blocks for the AES-GCM to propose efficient and high performance architectures. For the AES, through logic gate minimizations for the inversion in GF ($2^4$), the areas of the S-boxes have been reduced. We have also evaluated and compared the performance of different S-boxes using Xilinx tool. Furthermore, through exhaustive searches for the input patterns,

| Message length in Bytes |
|---|
| AAD length in Bytes |
| Key |
| Nonce |
| Message Block 1 |
| Message Block 2 |
| : |
| Message Block N |
| AAD Block 1 |
| AAD Block 2 |
| : |
| AAD Block M |

We have performed simulation-based results using modelsim tool for different S-boxes to reach more accurate results compared to the statistical methods. We have also proposed high-performance and efficient architectures for the GCM. For the case study of q = 8 parallel structures in $GHASH_H$, we have performed a hardware complexity reduction technique for the hash subkey exponentiations, having their timing complexities intact. Based on the available resources and performance goals to achieve, one can choose the proposed AES-GCM architectures to fulfill the constraints of different applications.

In future the performance of the proposed efficient architectures for the AES-GCM and their fault detection approaches can be benchmarked using application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) hardware platforms. Larger devices can be chosen to have enough number of slices needed. Another future work for the FPGA platform can be explored noting that the AES is utilized for bit stream security mechanisms. Specifically, the AES decryption is hardware-implemented in many recent FPGAs. Incorporating the proposed hardware countermeasures and evaluating their effectiveness in counteracting internal/malicious faults on FPGAs would be an interesting future research topic. Finally, one can work on devising reliable architectures for the recently standardized GCM, which provides data authentication to block ciphers such as the AES. To the best of my knowledge, the aforementioned research on reliability of these architectures will be carried out for the first time.

## REFERENCES

1 National Institute of Standards and Technologies, "Announcing the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication, no. 197, Nov. 2001.
2 M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST SP 800-38D, 2007.
3 Algotronics Ltd.: GCM Extension for AES G3 Core, 2007.
4 E. K¨asper and P. Schwabe, "Faster and Timing-Attack Resistant AES-GCM," In Proc. of CHES 2009, LNCS 5747, pp. 1-17, 2009.
5 K. Jankowski and P. Laurent, "Packed AES-GCM Algorithm Suitable for AES/PCLMULQDQ Instructions," IEEE Trans. Computers, vol. 60, no. 1, pp. 135-138, Jan. 2011.

## AUTHOR'S BIOGRAPHY

**Mr.ARUN** presently working as Assistant Professor in MLR instuitute of technology completed B.Tech in Electronics and communication in JNTU Hyderabad , M.Tech in Embedded systems from, JNTU Hyderabad. His interested areas are Embedded systems ,VLSI and image processing.

**Mrs.K.VANISREE,** working as Associate professor, HITS COE,Hyderabad, pursuinh P.Hd from JNTUH. Her areas of interest are communications, Image processing and signal processing

**Mr. Laxmareddy presently** working as Assistant Professor in MLR instuitute of technology completed B.Tech in ECE from Mother Theresa College of Engineering , Hyderabad and M.Tech in computer communications from RRS college of engineering and Technology, JNTU Hyderabad. His interested areas in communications and image processing.