Improved Method for Parallel AES-GCM Cores Using FPGAs

Karim M. Abdellatif, R. Chotin-Avot, and H. Mehrez
LIP6-SoC Laboratory, University of Paris VI, France
{karim.abdellatif, roselyne.chotin-avot, habib.mehrez}@lip6.fr

***Abstract*—This paper proposes an efficient method for implementing parallel AES-GCM cores using FPGAs. The proposed method improves the performance of the parallel architecture (Throughput/Slice). Presented architectures can be used for applications which require encryption and authentication with slow changing keys like Virtual Private Networks (VPNs). Our architectures were evaluated using Virtex5 FPGAs. It is shown that the performance of the presented parallel AES-GCM architecture outperforms the previously reported ones.**

***Keywords*-Parallel AES-GCM, FPGAs**

I. Introduction

AES-GCM [1] simultaneously provides confidentiality, integrity and authenticity assurances on the data. It supports not only high speed authenticated encryption but also protection against bit-flipping attacks. It can be implemented in hardware to achieve high speeds with low cost and low latency. Software implementations can achieve excellent performance by using table-driven field operations. GCM was designed to meet the need for an authenticated encryption mode that can efficiently achieve speeds of 10 Gbps and higher in hardware. It contains an AES engine in CTR mode and a Galois Hash (GHASH) module.

It is well-suited for wireless, optical, and magnetic recording systems due to its multi-Gbps authenticated encryption speed, outstanding performance, minimal computational latency as well as high intrinsic degree of pipelining and parallelism. New communication standards like IEEE 802.1ae [2] and NIST 800-38D have considered employing GCM to enhance their performance.

Although in ASIC technologies, several architectures of the AES-GCM reaching the 100 Gbps throughput have been demonstrated by [3], to the best of our knowledge no real designs for field-programmable gate array (FPGA) devices reaching the same performances have been so far presented. Therefore, we present efficient method for implementing parallel AES-GCM architectures by taking the advantage of slow changing key applications.

Section II presents a background of AES-GCM. After that, our proposed parallel AES-GCM is presented in **Section III**. Then, we report our implementation results in **Section IV**. Finally, **Section V** concludes our work.

II. Related Work of AES-GCM

As shown in Fig. 1, the GHASH function (authentication part) is composed of chained GF(2^{128}) multipliers and bit-wise exclusive-OR (XOR) operations. Serial implementation of GF(2^{128}) multiplier performs the multiplication process in 128 clock cycles. Parallel method can be implemented like [4] and it takes only one clock cycle.

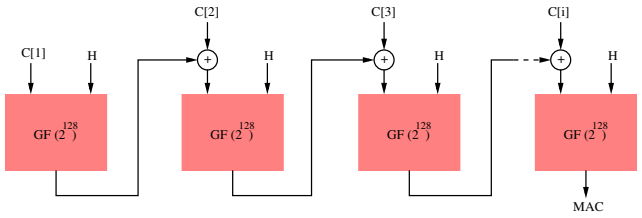

Figure 1. GHASH operation

Karatsuba Ofman Algorithm (KOA) was used by [5] to reduce the complexity (consumed area) of the GF(2^{128}) multiplier. In order to reduce the data path of KOA multiplier, pipelining concept was accomplished by [6]. Although the use of pipelining concept for KOA decreases the data path and increases the operating frequency, the number of clock cycles to process a number of 128-bits is increased. This is because the output is fed back and XORed to the next input and there is a latency resulting from pipelining. An example of this problem is shown in [6], their GF multiplier achieves the multiplication of 8 frames of 128-bits in 19 clock cycles because of using the pipelining concept. Hence, their throughput is calculated as follows:

$$Throughput(Mbps) = F_{max(MHz)} \times 128 \times (8/19) \quad (1)$$

If **H** is fixed, the multiplier is called fixed operand GF(2^{128}) multiplier [7] which can be used efficiently (smaller area) on FPGAs as the circuit is specialized for **H** and a new reconfiguration is uploaded into the FPGA with the new specialization in case of changing the key.

Two methods of pipelined AES (composite field and BRAMs) were accomplished with KOA multiplier by [5] using virtex4 FPGA. Zhou et al. [6] used three methods for pipelined AES implementation (composite field, BRAMs, and LUT) with pipelined KOA to increase the operating frequency of the overall architecture. Henzen et al. [8] proposed 4-parallel AES-GCM using pipelined KOA. Their design achieved the authentication of 18 frames of 128-bits in 11 clock cycles because of the latency resulting from the pipelined KOA. As a result, their throughput is calculated

978-1-4799-2079-2/13/$31.00 ©2013 IEEE

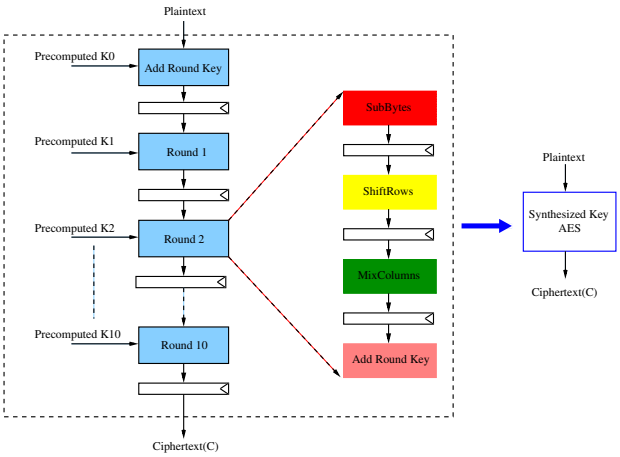

Figure 2. key-synthesized based AES

as follows:

$$Throughput(Mbps) = F_{max(MHz)} \times 128 \times (18/11) \quad (2)$$

III. Efficient Parallel AES-GCM for Slow Changing Key Applications Using FPGAs

Virtual Private Networks (VPNs) are widely employed to connect private local area networks to remote locations. VPNs use AES-GCM for encryption and authentication. In this kind of networks, the secret key used for encryption and authentication is changed weekly, monthly or yearly. Current commercial security appliances of VPNs allow a throughput from 40 to 60 Gbit/s [9],[10]. Another example of slow changing keys application is embedded system memory protection [11] which requires infrequent key changes over weeks or months.

The previous applications are considered slow key changing applications. Therefore, implementing the key expansion is particularly expensive in terms of hardware cost. The GF multiplier used for authentication is a challenge because its data path is longer than AES and pipelining method does not solve this problem as described before.

AES has a key expansion or key schedule operation, which takes the main key and derives from it subkeys K_r (10, 12, and 14 for AES-128, AES-192, and AES-256, respectively), where **r** denotes the corresponding round number. For our case, we concentrate on AES-128.

In our hardware implementation, constant key specialization on the FPGA is used. The precomputed keys are generated using a **C** code. After, these keys are synthesized into the architecture of AES. As a result, the key expansion scheme is reduced from the architecture of AES. As we look for high speed architectures, subpipelining is used to obtain high throughput. Fig. 2 shows synthesized key AES, where all keys are precomputed and synthesized into the architecture. As a result of using key-synthesized AES, the

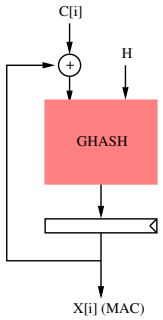

Figure 3. GHASH operation

operand H of the GHASH function is also fixed because it is generated by applying the block cipher to the zero block. Therefore, the proposed multiplier by [7] is suitable because it is based on fixed operand multiplier.

In order to implement parallel architectures of AES-GCM using key-synthesized method, parallel GHASHs must be constructed to meet the requirement of key-synthesized method (i.e, one of the two operands of every GHASH must be fixed).

Previous parallel schemes of GHASH [3], [12], [7] are not suitable because the two operands of every GHASH are varied during the running time operation. As a result, their architectures are not suitable for key-synthesized approach. Therefore, constructing parallel GHASHs which have a fixed operand for every GHASH multiplier is very important for high speed applications.

Fig. 3 shows the GF(2^{128}) multiplication between an **H** value and a 128-bit input value **C**. $GHASH_H$ function for block i is defined in Eq. 3.

$$X_i = (C_i \oplus X_{i-1}) \times H \quad (3)$$

In order to accomplish parallel architectures for higher throughput, Eq. 3 is rewritten to fit the parallel scheme as shown in Eq. 4. For every multiplier, there is a fixed operand as shown in Fig. 4. For example, $Multiplier_i$ has H as an operand, $Multiplier_{i-1}$ has H^2, ..., and $Multiplier_1$ has H^i .

$$\begin{aligned} X_i &= (C_i \oplus X_{i-1}) \times H \\ &= (C_i \times H) \oplus (X_{i-1} \times H) \\ &= (C_i \times H) \oplus [(C_{i-1} \oplus X_{i-2}) \times H^2] \\ &= (C_i \times H) \oplus (C_{i-1} \times H^2) \oplus [(C_{i-2} \oplus X_{i-3}) \times H^3] \\ &= (C_i \times H) \oplus (C_{i-1} \times H^2) \oplus (C_{i-2} \times H^3) \\ &\oplus [(C_{i-3} \oplus X_{i-4}) \times H^4] \\ &= \underbrace{(C_i \times H)} \oplus \underbrace{(C_{i-1} \times H^2)} \oplus \underbrace{(C_{i-2} \times H^3)} \\ &\oplus \underbrace{(C_{i-3} \times H^4)} .... \oplus \underbrace{(C_2 \times H^{i-1})} \oplus \underbrace{(C_1 \times H^i)} \end{aligned} \quad (4)$$

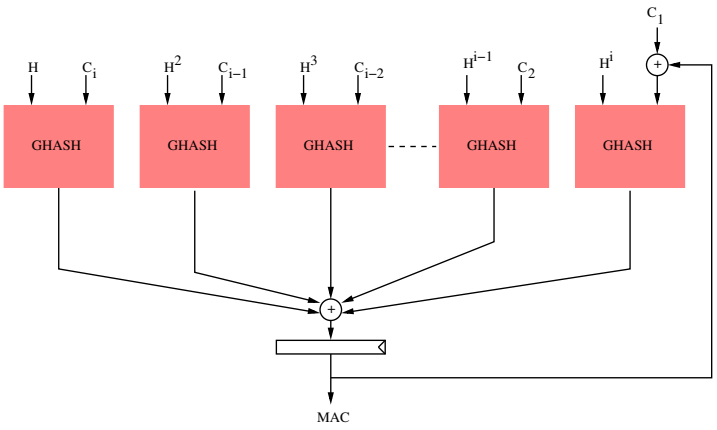

Figure 4. Proposed parallel GHASH with fixed operand during running time operation

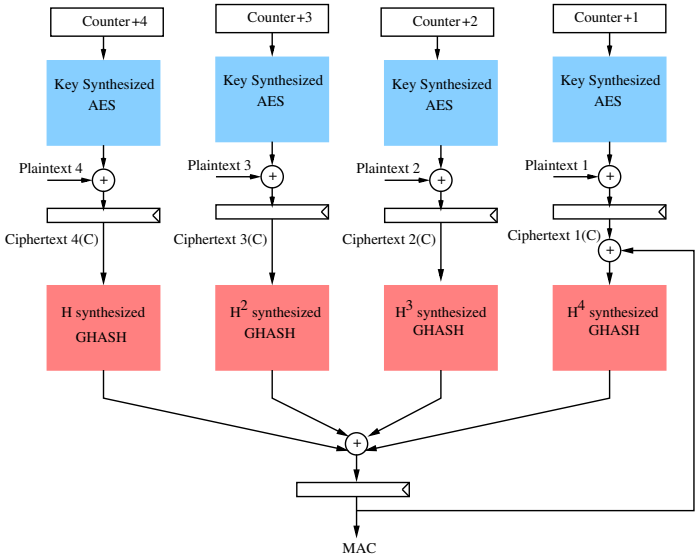

Figure 5. Presented 4-parallel AES-GCM using key-synthesized method

Unlike previous work, this method makes the parallel architecture of GHASH suitable for key-synthesized method as the values H^i, H^{i-1}, ... H get synthesized into the architecture of the parallel GHASH. After solving the problem of parallel GHASHs using key-synthesized method, these parallel multipliers are combined with parallel AES cores to perform parallel AES-GCM. Fig. 5 shows an example of 4-parallel AES-GCM architecture using key-synthesized method. For parallel pipelined AES, the round keys are precomputed and synthesized into the architecture. In terms of parallel GHASH, the used operands (H, H^2, H^3, H^4) are also precomputed and synthesized into the architecture. VPNs infrastructure [9] can benefit from our key-synthesized parallel AES-GCM due to the nature of slow changing key operation.

IV. Hardware comparison

We coded two architectures (AES-GCM and 4-parallel AES-GCM) in VHDL and targeted to Virtex5 (XC5VLX220). ModelSim 6.5c was used for simulation. Xilinx Synthesize Technology (XST) is used to perform the synthesize and ISE9.2 was adopted to run the Place And Route (PAR).

Table I shows the hardware comparison between our results and previous work. Note the filled dots in the ”Key” column. Key is synthesized into the architecture when denoted by ○, otherwise, the key schedule is implemented when denoted by ●.

The LUT approach for SubBytes is especially interesting on Virtex5 devices because 6-input Look-Up-Tables (LUT) combined with multiplexors allow an efficient implementation of the AES SubBytes stage. Therefore, we implemented

Table I
HARDWARE COMPARISON

	FPGA type	Design	Key	SubBytes	Slices	Max-Freq MHz	Thr. Gbit/s	Thr./Slice Mbps/Slice
This work	Virtex5	AES/GCM	○	LUT	3211	216.3	27.7	8.62
This work	Virtex5	4-parallel AES/GCM	○	LUT	12152	200	102.4	8.42
[6]	Virtex5	AES/GCM	•	LUT	4628	324	17.5	3.77
[8]	Virtex5	4-parallel AES/GCM	•	LUT	14799	233	48.8	3.29

the SubBytes of AES using LUT approach because the final target is Virtex5. On Virtex5 platform, our key-synthesized based AES-GCM core reaches the throughput of 27.7 Gbps with the area consumption of 3211 slices. By comparing our results of AES-GCM to the previous work, the comparison shows that our performance (Thr. /Slice) is better than [6]. The operating frequency presented by [6] is better than ours because they used pipelined KOA but the overall throughput is lower than ours because their GHASH achieves the multiplication of 8 frames of 128-bits in 19 clock cycles. Therefore, their throughput is calculated as described in Eq. 1.

A 4-parallel AES-GCM module operates at 200 MHz on Virtex-5. In total, it consumes 12152 Slices. This implementation achieves throughput that reaches to 102.4 Gbps. Henzen et al. [8] proposed 4-parallel AES-GCM using pipelined KOA. Their design achieves the authentication of 18 frames of 128-bits in 11 clock cycles because of the latency resulting from the pipelined KOA. As a result, their throughput is calculated as described in Eq. 2.

It is clear that our work outperforms the previously reported ones. Therefore, proposed architectures can be used efficiently for slow changing key applications like VPNs and embedded memory protection.

V. CONCLUSION

In this paper, we presented the performance improvement for implementing parallel AES-GCM cores using FPGAs. We integrated this solution by solving the complexity of the GHASH calculation (Eq. 4). With our proposed parallel AES-GCM, every multiplier has a fixed operand. Therefore, our parallel AES-GCM is suitable for key-synthesized method. Our comparison to previous work reveals that our architectures are more performance-efficient (Thr./Slice).

REFERENCES

[1] D. McGrew and J. Viega, "The Security and Performance of the Galois/Counter Mode (GCM) of Operation," *Progress in Cryptology-INDOCRYPT 2004*, pp. 377–413, 2005.

[2] "IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Security Amendment 1: Galois Counter Mode–Advanced Encryption Standard– 256 (GCM-AES-256) Cipher Suite," *IEEE*.

[3] A. Satoh, T. Sugawara, and T. Aoki, "High-Speed Pipelined Hardware Architecture for Galois Counter Mode," *Information Security*, pp. 118–129, 2007.

[4] A. Satoh, "High-Speed Hardware Architectures for Authenticated Encryption Mode GCM," *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pp. 4–pp, 2006.

[5] G. Zhou, H. Michalik, and L. Hinsenkamp, "Efficient and High-Throughput Implementations of AES-GCM on FPGAs," *International Conference on Field-Programmable Technology (FPT)*, pp. 185–192, 2007.

[6] G. Zhou and H. Michalik, "Improving Throughput of AES-GCM with Pipelined Karatsuba Multipliers on FPGAs," *Reconfigurable Computing: Architectures, Tools and Applications*, pp. 193–203, 2009.

[7] J. Crenne, P. Cotret, G. Gogniat, R. Tessier, and J. Diguet, "Efficient Key-Dependent Message Authentication in Reconfigurable Hardware," *International Conference on Field-Programmable Technology (FPT)*, pp. 1–6, 2011.

[8] L. Henzen and W. Fichtner, "FPGA Parallel-Pipelined AES-GCM Core for 100G Ethernet Applications," *Proceedings of the ESSCIRC*, pp. 202–205, 2010.

[9] C. Corporation, "Cisco ASA 5500 Series Adaptive Security Appliances," 2011. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6032/ps6094/ps6120/prod-brochure0900aecd80285492.pdf

[10] Stonesoft, "Security Engine Firewall/VPN," 2011. [Online]. Available: http://www.stonesoft.com/export/download/pdf/datasheet-stonesoft-3206.pdf

[11] R. Vaslin, G. Gogniat, J. Diguet, R. Tessier, D. Unnikrishnan, and K. Gaj, "Memory Security Management for Reconfigurable Embedded Systems," *International Conference on ICECE Technology*, pp. 153–160, 2008.

[12] J. Wang, G. Shou, Y. Hu, and Z. Guo, "High-Speed Architectures for GHASH Based on Efficient Bit-parallel Multipliers," *IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS)*, pp. 582–586, 2010.