FPGA Parallel-Pipelined AES-GCM Core for 100G Ethernet Applications

Luca Henzen and Wolfgang Fichtner
Integrated Systems Laboratory, ETH Zurich, Switzerland
E-mail: {henzen, fw}@iis.ee.ethz.ch

***Abstract*—The forthcoming IEEE 802.3ba Ethernet standard will provide data transmission at a bandwidth of 100 Gbit/s. Currently, the fastest cryptographic primitive approved by the U.S. National Institute for Standard and Technology, that combines data encryption and authentication, is the Galois/Counter Mode (GCM) of operation. If the feasibility to increase the speed of the GCM up to 100 Gbit/s on ASIC technologies has already been demonstrated, the FPGA implementation of the GCM in secure 100G Ethernet network systems arises some important structural issues. In this paper, we report on an efficient FPGA architecture of the GCM combined with the AES block cipher. With the parallelization of four pipelined AES-GCM cores we were able to reach the speed required by the new Ethernet standard. Furthermore, the time-critical binary field multiplication of the authentication process relies on four pipelined 2-step Karatsuba-Ofman multipliers.**

I. Introduction

The Galois/Counter Mode (GCM) of operation [1] has been standardized by the U.S National Institute for Standard and Technology in order to provide a high-speed algorithm for authenticated encryption and decryption. GCM was the answer to the growing demand of confidentiality and data integrity in modern multi-gigabit communication systems. In the last years, GCM has indeed been applied in numerous standards, such as the IETF RFC 4106 for IPsec Encapsulating Security Payload [2] or the IEEE P1619.1 for authenticated encryption with length expansion for storage devices [3]. Moreover, the combination of the GCM with the counter mode of the block cipher AES (Advanced Encryption Standard [4]) has been recommended in the IEEE 802.1AE standard for Media Access Control (MAC) Security. Several hardware implementations have already been proposed; in [5] the authors presented a 97.9 Gbps AES-GCM core in 0.13 µm CMOS technology for the IEEE 802.1AE, while in [6] a complete link encryptor based on the AES-GCM has been investigated and tested in 2G Fibre Channel networks.

Although in ASIC technologies, several architectures of the AES-GCM reaching the 100 Gbps throughput have been demonstrated [7], [8], to the best of our knowledge no designs for field-programmable gate array (FPGA) devices reaching the same performances have been so far presented. Following the road map of the IEEE P802.3ba standard for 40G and 100G Ethernet [9], the final approval is scheduled in June 2010. The possibility to secure Ethernet traffic at this speed using reconfigurable hardware becomes therefore crucial.

In this work, we present a GCM architecture combined with the AES, which is able to fully support 100 Gbps speed. Exploiting the parallelization of four cores plus the extensive utilization of pipelining, we were able to design three different 100G AES-GCM implementations for Xilinx Virtex-5 FPGAs.

II. GCM Authenticated Encryption

The GCM is a block cipher mode of operation that is able to encrypt or decrypt data, providing at the same time authentication and data integrity . In practice, it combines a block cipher in the counter mode with universal hashing over the binary field GF(2^{128}). In this work, we used the Advanced Encryption Standard (AES) [4] for encryption and decryption, supporting key sizes of 128, 192 and 256 bits.

The target AES-GCM algorithm takes as input a plaintext P (or input message), split into 128-bit sequences $P_1, P_2, \ldots, P_n^*$, an initialization vector IV, some additional authenticated data $A = (A_1, A_2, \ldots, A_m^*)$, and the secret key K. The size in bits of the final blocks P_n^* and A_m^* is defined by u and v, with $1 \leq u, v \leq 128$. The following equation defines the authenticated encryption of GCM:

$$\begin{array}{rcll} H & = & \text{Enc}(K, 0^{128}) & \\ Y_0 & = & \begin{cases} IV||0^{31}1 & \text{if } \text{len}(IV) = 96 \\ \text{GHASH}(H, \{\}, IV) & \text{otherwise} \end{cases} & \\ Y_i & = & \text{incr}(Y_{i-1}) & i = 1, 2, \ldots, n \\ C_i & = & P_i \oplus \text{Enc}(K, Y_i) & i = 1, 2, \ldots, n-1 \\ C_n^* & = & P_n^* \oplus \text{MSB}_u(\text{Enc}(K, Y_n)) & \\ T & = & \text{MSB}_t(\text{GHASH}(H, A, C) \oplus \text{Enc}(K, Y_0)). & \end{array} \quad (1)$$

$\text{Enc}(K, Y)$ denotes the AES encryption of the block Y with the key K. The GHASH() function in the last lines takes the $(m + n + 1)$-block input composed by H, A, and C, and compresses it to a single 128-bit block. The compression is performed by the sequential multiplication of A and C with the pre-computed term H over the Galois field GF(2^{128}).

The authentication tag T consists in the first t bits of the exclusive-OR (XOR) between the encryption of the IV and the output $X_{m+n+1} = \text{GHASH}(H, A, C)$, where

$$X_i = \begin{cases} 0 & i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A_m^*||0^{128-v})) \cdot H & i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C_n^*||0^{128-u})) \cdot H & i = m+n \\ (X_{m+n} \oplus (\text{len}(A)||\text{len}(C))) \cdot H & i = m+n+1 \end{cases} \quad (2)$$

The last multiplication is computed using the concatenation of the 64-bit sizes of A and C.

Since the block cipher is in the counter mode, the authenticated decryption in GCM is simply computed by exchanging the input of the GHASH function with the incoming data, in this case the received ciphertext. The new generated tag T' is then compared with the received tag T. In case of mismatch, the decrypted plaintext is completely discarded.

A. Parallel Multiplication

Originally proposed in [1], [10], the parallelization process of the multiply operation in (2) has been further investigated in [8]. The goal is to divide the sequential *multiplication-addition* steps into several parallel computations, which generate the same final result. By defining a parallelization degree q, the final X block could indeed be expressed as the sum of q sub-terms Q_i:

$$X_{m+n+1} = Q_q \oplus Q_{q-1} \oplus \ldots \oplus Q_1, \tag{3}$$

where

$$\begin{aligned} Q_q &= (((I_1 H^q \oplus I_{q+1})H^q \oplus I_{2q+1})H^q \oplus \ldots)H^q \\ Q_{q-1} &= (((I_2 H^q \oplus I_{q+2})H^q \oplus I_{2q+2})H^q \oplus \ldots)H^{q-1} \\ &\vdots \\ Q_1 &= (((I_q H^q \oplus I_{2q})H^q \oplus I_{3q})H^q \ldots)H, \end{aligned} \tag{4}$$

and

$$\begin{aligned} &(I_1, I_2, \ldots, I_{m+n+1}) = \\ &(A_1, \ldots, A_m^* || 0^{128-v}, C_1, \ldots, C_n^* || 0^{128-u}, \text{len}(A) || \text{len}(C)). \end{aligned} \tag{5}$$

If the length of the input blocks $m+n+1$ is not a multiple of q, the last q multiplications are accordingly shifted through the Q_i terms. More important is that the different Q_i could be computed separately and then XORed only at the end of the authentication process.

III. Architecture

The achievement of throughput rates up to 100 Gbps in state-of-the-art FPGA devices is almost impossible with a single AES core in combination with a bit-parallel multiplier computing the tag. Even with modern FPGAs the maximal speed of a standard architecture is limited to 40-50 Gbps [11], [12]. In order to support the new Ethernet standard IEEE 802.3ba, we then decided to exploit parallelization and pipelining both in the AES and the GHASH function.

A. Multi-core AES Design

A block cipher in counter mode of operation does not require to feedback the output of the previous block to compute the next one. Aiming at speed, this feature allows the insertion of pipeline stages in unrolled implementations. In order to support the three key sizes, the AES architecture relies on 14 unrolled rounds separated by pipeline registers. The most critical transformation within a single round is the *SubBytes* process. Several approaches to design the *SubBytes* have been developed and implemented in FPGA. We selected three main solutions:

- The entire computation is replaced by a substitution table S-box. Two 2048-bit S-boxes are then stored inside a dual-port block RAM (BRAM).
- The use of composite field to reduce the complexity of the *SubBytes* [13] leads to a plain combinatoric circuit implemented in FPGA logic.
- As proposed in [14], a single 2048-bit S-box could efficiently be stored in 32 6-input LUTs of a Xilinx Virtex-5 FPGA.

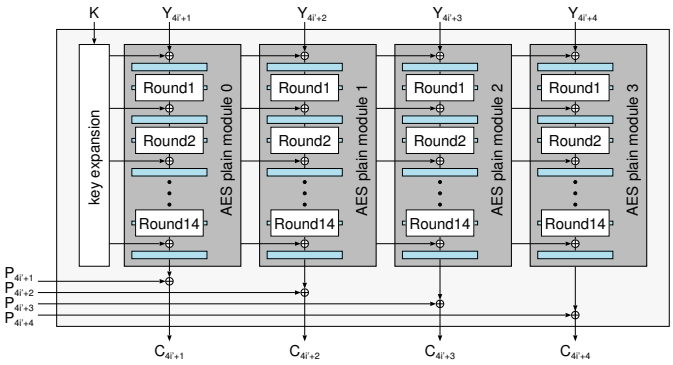

Figure 1. Block diagram of the multi-core AES-128. The blue blocks identify pipeline register stages. Each round has an additional internal pipeline stage. Connections are 128-bit wide, while $i' = 0, 1, \ldots, \lceil \frac{n}{4} \rceil - 1$.

Since the use of BRAMs introduces an additional cycle of latency, we decided to add an *in-round* pipeline stage into the composite and the LUT-based approaches.

Nevertheless, a single pipelined AES is not able to achieve 100 Gbps. This limitation forces the introduction of a multi-core construction. We instantiated four parallel AES cores sharing the same circuit to perform the key expansion transformation. The four AES plain modules work consequently using the same round keys. Fig. 1 shows a schematic overview of the main components. The resulting multi-core design is thus able to process a 512-bit block (4×128 bits) of plaintext at each clock cycle, since the ciphertext is generated by directly XORing the four 128-bit blocks $P_{4i'+1}$, $P_{4i'+2}$, $P_{4i'+3}$, and $P_{4i'+4}$ with the output strings of the four plain modules ($i' = 0, 1, \ldots, \lceil \frac{n}{4} \rceil - 1$). Particular attention has to be taken with the generation of these outputs. As pointed out in [15], the same counter should not be used twice with the same key. This means that the input counter should not be equal for the four AES modules. This could easily be avoided, by fixing with different values two bits of the four input counters $Y_{4i'+1}$, $Y_{4i'+2}$, $Y_{4i'+3}$, and $Y_{4i'+4}$.

B. The Parallel Pipelined GHASH Design

To combine the authentication core with the multi-core AES, a design solution based on four parallel binary-field multipliers has been investigated. Due to the high-speed re-

quirement, we were forced to further insert pipelining into each multiplier. This is due to the fact that the speed of a plain multiplier is mainly defined by the size of the binary field, in this work the large GF(2^{128}). We adopted the 2-step Karatsuba-Ofman (KO) algorithm and designed a 4-stage pipelined architecture similar to [16]. The overview of the implemented multiplier is depicted in Fig. 2.

More precisely, the single step KO algorithm splits two m-bit inputs A and B into four terms A_h, A_l, B_h, and B_l, where the index identifies the highest or lowest $\frac{m}{2}$ bits (*split* phase). The result R is the combination of the outputs of three multiplications between these four terms:

$$\begin{array}{lcl} R_l & = & A_l B_l \\ R_{hl} & = & (A_h + A_l)(B_h + B_l) \\ R_h & = & A_h B_h \\ R & = & R_h x^m + x^{\frac{m}{2}}(R_h + R_{hl} + R_l) + R_l. \end{array} \quad (6)$$

The last line computes the final result of the multiplication between A and B, by aligning the intermediates results R_l, R_{hl}, and R_h (*align* phase).

We applied this approach twice recursively, in order to reduce a large 128-bit multiplier into nine 32-bit multipliers (see the "2-s mult" region in Fig. 2). The basic bit-parallel multiplier has then be implemented to carry out the 32-bit multiplication. In this way, the overall complexity of the computation could be decreased allowing the insertion of registers to shorten the longest path. As can be seen in Fig. 2, a single KO multiplier hosts four pipeline stages. The first comes after the *split* phases, the second after the multipliers, while the last two stages isolate the binary-field reduction from the two *align* phases.

To preserve the correct tag computation, the parallelization degree q has then been set to 16 (4 parallelization $\times$ 4 pipelining). This means that the input blocks I_i are processed into 16 independent accumulators $Q_1, Q_2, \ldots, Q_{16}$.

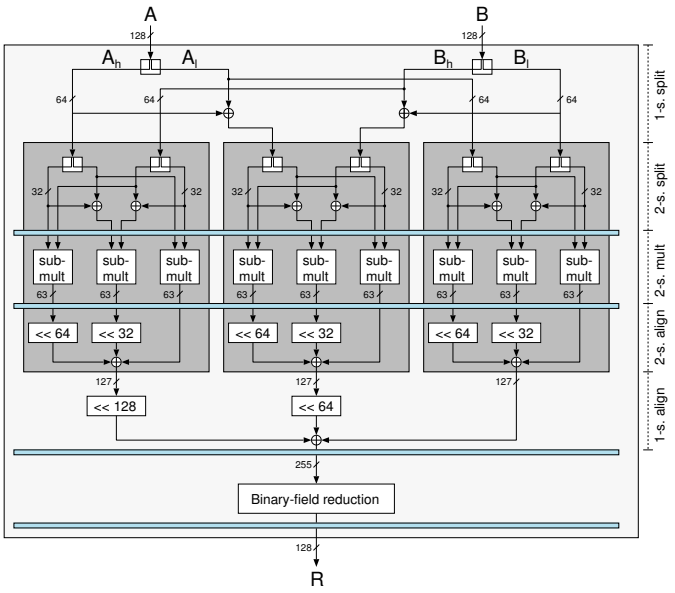

Figure 2. Architecture overview of the pipelined 2-step Karatsuba Multiplier. The blue lines represent the four pipeline stages.

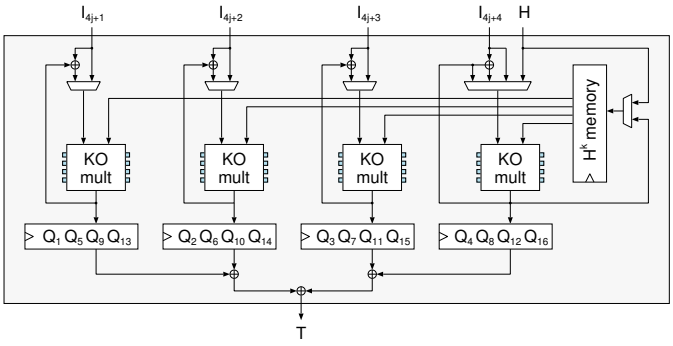

Figure 3. Architecture of the parallel authentication core. The four *KO mult* modules are based on the pipelined 2-step Karatsuba-Ofman multiplier. Connections are 128-bit wide, $j = 0, 1, \ldots, \lceil \frac{m+n+1}{4} \rceil - 1$.

The block diagram of the resulting GHASH module is illustrated in Fig. 3. Each input multiplexer selects the first operand of the multiplication, while the second, i.e. the terms H^k with $1 \leq k \leq 16$, comes from a dedicated memory module. The 16 accumulators Q_i are also stored in register-based memories. The authentication tag T is obtained by XORing the 16 outputs of these memories.

C. GCM Design

The overall GCM architecture is based on the combination of the multi-core AES and the parallel-pipelined authentication core. The powers of H are computed whenever the key is updated. After the encryption of the 128-bit zeros term, 17 cycles are indeed needed to compute and store inside a dedicated memory the 16 H^k terms. After this task the AES-GCM core is ready to encrypt messages with the new key.

In (4), the last multiplication of the accumulators Q_i is done using scaling power terms. This involves the *a priori* knowledge by the GCM core of the total message length. While most of the blocks are multiplied with H^{16}, the last 14 blocks must be multiplied with lower power terms. In Table I the input pairs of the four multipliers are given for a 18 128-bit blocks plaintext with three 128-bit bl ocks of authenticated data. Note that the sum of $m + n + 1$, i.e. $3 + 18 + 1 = 22$, is expressly not a multiple of $q = 16$.

In order to configure the correct H^k inputs, the GCM controller needs to know already at the second cycle that the block P_5 must be multiplied with H^{15}, scaling the powers of H in the next four cycles. This problem is solved by adding a 4-stage buffer at the input of the AES-GCM. The gray module in Fig. 4 acts like a shift register with controlled output. Thanks to this component, the AES-GCM is informed in advance of the configuration and total length of the message. The output multiplexer is used in case of small messages, i.e. $n < 16$. The input controller selects indeed the correct buffer output, depending on the computed size of the plaintext. In spite of an increase of the total system latency by four cycles, the AES-GCM is able to process data sequentially without having any prior information on the message size.

Table I
INPUT PAIRS OF THE FOUR MULTIPLIERS. THE FLOW CORRESPONDS TO A 288 BYTES PLAINTEXT WITH 48 BYTES OF AUTHENTICATED DATA.

Clk	I_{4j+1}	H^k	I_{4j+2}	H^k	I_{4j+3}	H^k	I_{4j+4}	H^k
1	A_1	H^{16}	A_2	H^{16}	A_3	H^{16}	P_1	H^{16}
2	P_2	H^{16}	P_3	H^{16}	P_4	H^{16}	P_5	H^{15}
3	P_6	H^{14}	P_7	H^{13}	P_8	H^{12}	P_9	H^{11}
4	P_{10}	H^{10}	P_{11}	H^9	P_{12}	H^8	P_{13}	H^7
5	P_{14}	H^6	P_{15}	H^5	P_{16}	H^4	P_{17}	H^3
6	P_{18}	H^2	len^a	H	-	-	-	-

[a] $\text{len}(A)||\text{len}(C)$

Because of the pipeling architecture applied in the GHASH core, the correct authentication tag T appears four cycles after the insertion of the last message blocks. In Table I, T would then be generated at the 11th cycles.

IV. RESULTS AND COMPARISON

The parallel-pipelined AES-GCM core has been coded in functional VHDL in three architectures, differing by their *SubBytes* implementation (see Sec. III-A). The three designs have then been synthesized using Synplify Pro, while place and route has been done with the Xilinx ISE Design Suite. Target FPGAs were two Xilinx Virtex-5 chips, i.e., the XC5VLX220 with speed grade -2 for the cores with composite and LUT-based *SubBytes*, and the XC5VSX240T (-2) for the BRAM *SubBytes*. The choice of different FPGAs is motivated by the large amount of required BRAMs in the last design. The 36 Kb BRAMs are indeed dual-port memories that could store only two S-box tables. In total, the mulit-core AES requires 450 BRAMs ($4\times14\times16$ S-boxes for the rounds and four for the key expansion). Such large amount of BRAMs is only available in bigger Virtex-5 FPGAs like the XC5VSX240T chip. Table II summarizes the performances and proposes a comparison with the results of [16]. Note that their implementations are based on a single-core pipelined AES-GCM.

We optimized the area of the AES-GCM cores for the same maximal frequency of 233 MHz. The three designs locate indeed their crytical path inside the key expansion module of the AES core. This frequency is suitably enough to guarantee the speed requirements imposed by the forthcoming 100 Gigabit Ethernet standard. Although the design using BRAMs for the *SubBytes* operation consumes less logic, it is penalized by the huge memory demand to store the S-boxes. We point out the core, using the LUT approach. This core fits in 14.8 kslices, about 43 % of the total space available in the XC5VLX220 chip, without requiring additional storing capacity.

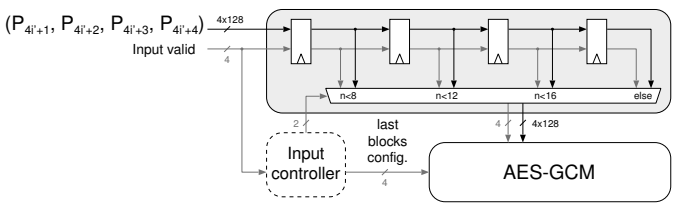

Figure 4. Block diagram of the input buffer architecture, $i' = 0, 1, \ldots, \lceil \frac{n}{4} \rceil - 1$.

Table II
FPGA PERFORMANCE COMPARISON OF THE HASH FUNCTIONS. FPGA FAMILY IS THE XILINX VIRTEX-5 WITH SPEED GRADE -2.

Ref.	*SubBytes*	Area [Slices]	Area [BRAM]	Freq. [MHz]	Thr'put [Gbps]	FPGA Type
Ours	LUT	14'799	0	233	119.30	LX220
Ours	Comp.	18'505	0	233	119.30	LX220
Ours	BRAM	9'561	450	233	119.30	SX240T
[16]	LUT	5'961	0	296	37.89	LX85
[16]	Comp.	8'077	0	305	39.04	LX85
[16]	BRAM	4'115	59	287	36.74	LX85

V. CONCLUSIONS

In this paper we have presented an efficient design methodology to implement in reconfigurable hardware devices the GCM combined with the AES for authenticated encryption. Thanks to the replication of four AES cores and four binary-field multipliers we were able to demonstrate how to break the 100 Gbps speed bound in FPGA. In order to reduce the critical path of the GHASH operation, four pipeline stages have been inserted within the GF(2^{128}) multiplication. The final GCM architecture relies on a 4×4 construction and achieves 119 Gbps in Xilinx Virtex-5 devices.

REFERENCES

[1] D. McGrew and J. Viega, "The Galois/Counter Mode of operation (GCM)," May 2005, Submission to NIST Modes of Operation.

[2] J. Viega and D. McGrew, "The use of Galois/Counter Mode (GCM) in IPsec encapsulating security payload (ESP)," Network Working Group, RFC 4106, http://tools.ietf.org/html/rfc4106.

[3] "IEEE P1619.1/D23 draft standard for authenticated encryption with length expansion for storage devices," Aug. 2007.

[4] NIST, "Advanced encryption standard (AES)," Nov. 2001, Federal Information Processing Standards (FIPS) Publication 197.

[5] C. Zhang, L. Li, J. Xu, and Z. Wang, "High-throughput GCM VLSI architecture for IEEE 802.1ae applications," in *Proc. of ISCAS*, Taipei, May 2009, pp. 900–903.

[6] L. Henzen, F. Carbognani, N. Felber, and W. Fichtner, "FPGA implementation of a 2G fibre channel link encryptor with authenticated encryption mode GCM," in *Proc. of SoC*, Tampere, Nov. 2008, pp. 1–4.

[7] A. Satoh, T. Sugawara, and T. Aoki, "High-speed pipelined hardware architecture for galois counter mode," in *Information Security*, ser. LNCS, vol. 4779. Springer, Heidelberg, 2007, pp. 118–129.

[8] ——, "High-performance hardware architectures for galois counter mode," *IEEE Trans. Comput.*, vol. 58, no. 7, pp. 917–930, Jul. 2009.

[9] I. P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force, http://www.ieee802.org/3/ba/.

[10] T. Kohno, J. Viega, and D. Whiting, "The CWC-AES dual-use mode," Internet Draft, Crypto Forum Research Group. Work in progress, May 2003, http://www.zork.org/cwc/draft-irtf-cfrg-cwc-01.txt.

[11] G. Zhou, H. Michalik, and L. Hinsenkamp, "Efficient and high-throughput implementations of AES-GCM on FPGAs," in *Proc. of ICFPT*, Kitakyushu, Dec. 2007, pp. 185–192.

[12] H. Technology, "AES-GCM Core family for Xilinx FPGA," Oct. 2008, http://www.heliontech.com/aes_gcm.htm.

[13] V. Rijmen, "Efficient implementation of the Rijndael SBox," 2002.

[14] P. Bulens, F.-X. Standaert, J.-J. Quisquater, P. Pellegrin, and G. Rouvroy, "Implementation of the AES-128 on virtex-5 FPGAs," in *Progress in Cryptology AFRICACRYPT 2008*, ser. LNCS, vol. 5023. Springer, Heidelberg, 2008, pp. 16–26.

[15] M. Dworkin, "Recommendation for block cipher mode of operation: Galois/counter mode (GCM) and (GMAC)," 2007, NIST SP 800-38D.

[16] G. Zhou, H. Michalik, and L. Hinsenkamp, "Improving throughput of AES-GCM with pipelined karatsuba multipliers on FPGAs," in *Reconfigurable Computing: Architectures, Tools and Applications*, ser. LNCS, vol. 5453. Springer, Heidelberg, 2009, pp. 193–203.