Improved Method for Parallel AES-GCM Cores Using FPGAs

Karim Moussa Ali Abdellatif, Roselyne Chotin-A vot, Habib Mehrez

To cite this version:
Karim Moussa Ali Abdellatif, Roselyne Chotin-A vot, Habib Mehrez. Improved Method for Parallel AES-GCM Cores Using FPGAs. ReConFig 2013 - International Conference on Reconfigurable Computing and FPGAs, Dec 2013, Cancun, Mexico. pp.1-4, 10.1109/ReConFig.2013.6732299. hal-01160904

HAL Id: hal-01160904
https://hal.sorbonne-universite.fr/hal-01160904
Submitted on 8 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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 bitwise 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.

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:

\[ \text{Throughput (Mbps)} = F_{\text{max}}(MHz) \times 128 \times (8/19) \] (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
architecture. As a result of using key-synthesized AES, where all keys are precomputed and synthesized into the architecture of AES. 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 function for block $i$ is defined in Eq. (3)

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

$$X_i = (C_i \oplus X_{i-1}) \times H$$
$$= (C_i \times H) \oplus (X_{i-1} \times H)$$
$$= (C_i \times H) \oplus (C_{i-1} \times X_{i-2} \times H^2)$$
$$= (C_i \times H) \oplus (C_{i-1} \times H^2) \oplus ((C_{i-2} \oplus X_{i-3}) \times H^3)$$
$$= (C_i \times H) \oplus (C_{i-1} \times H^2) \oplus (C_{i-2} \times H^3)$$
$$\oplus (C_{i-3} \times X_{i-4} \times H^4)$$
$$\oplus (C_{i-3} \times H^4) \oplus (C_{i-2} \times H^{i-1}) \oplus (C_1 \times H^i)$$  (4)
### Table I

**HARDWARE COMPARISON**

<table>
<thead>
<tr>
<th>FPGA type</th>
<th>Design</th>
<th>Key</th>
<th>SubBytes</th>
<th>Slices</th>
<th>BRAM</th>
<th>Max-Freq MHz</th>
<th>Thr. Gbit/s</th>
<th>Thr./Slice Mbps/Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>Virtex5</td>
<td>AES-GCM</td>
<td>°</td>
<td>LUT</td>
<td>3211</td>
<td>0</td>
<td>216.3</td>
<td>27.7</td>
</tr>
<tr>
<td>This work</td>
<td>Virtex5</td>
<td>4-parallel AES-GCM</td>
<td>°</td>
<td>LUT</td>
<td>12152</td>
<td>0</td>
<td>200</td>
<td>102.4</td>
</tr>
<tr>
<td>[6]</td>
<td>Virtex5</td>
<td>AES-GCM</td>
<td>●</td>
<td>LUT</td>
<td>4628</td>
<td>41</td>
<td>324</td>
<td>17.5</td>
</tr>
<tr>
<td>[8]</td>
<td>Virtex5</td>
<td>4-parallel AES-GCM</td>
<td>●</td>
<td>LUT</td>
<td>14799</td>
<td>0</td>
<td>233</td>
<td>48.8</td>
</tr>
</tbody>
</table>

#### 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 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).

Acknowledgment

This work is a part of the project Robust FPGA ANR 11 INS-02, funded by The French National Research Agency, The Pole Systematic and The Pole Minalogic.

References


