


---

**Proceedings Paper:**
Kavun, E.B. orcid.org/0000-0003-3193-8440, Mentens, N., Vliegen, J. et al. (1 more author) (2020) Efficient utilization of DSPs and BRAMs revisited : new AES-GCM recipes on FPGAs. In: 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig). 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), 09-11 Dec 2019, Cancun, Mexico. IEEE . ISBN 9781728119588

https://doi.org/10.1109/reconfig48160.2019.8994730

---



# Efficient Utilization of DSPs and BRAMs Revisited: New AES-GCM Recipes on FPGAs

Elif Bilge Kavun
*The University of Sheffield*, Sheffield, UK
e.kavun@sheffield.ac.uk

Nele Mentens
*imec-COSIC and ES&S, ESAT, KU Leuven*, Leuven, Belgium
nele.mentens@kuleuven.be

Jo Vliegen
*imec-COSIC and ES&S, ESAT, KU Leuven*, Leuven, Belgium
jo.vliegen@kuleuven.be

Tolga Yalçın
*Northern Arizona University*, Flagstaff, AZ, US
tolga.yalcin@nau.edu

***Abstract*—In 2008, Drimer et al. proposed different AES implementations on a Xilinx Virtex-5 FPGA, making efficient use of the DSP slices and BRAM tiles available on the device. Inspired by their work, we evaluate the feasibility of extending AES with the popular GCM mode of operation, still concentrating on the optimal use of DSP slices and BRAM tiles. We make use of a Xilinx Zynq UltraScale+ MPSoC FPGA with improved DSP features. For the AES part, we implement Drimer's round-based and unrolled pipelined architectures differently, still using DSPs and BRAMs efficiently based on the AES Tbox approach. On top of AES, we append the GCM mode of operation, where we use DSP slices to support the GCM finite field multiplication. This allows us to implement AES-GCM with a small amount of FFs and LUTs. We propose two implementations: a relatively compact round-based design and a faster unrolled design.**

## I. Introduction

FPGA vendors make heterogeneous devices that contain dedicated cores on the silicon die, next to the traditional reconfigurable gates. Although such a versatile array of different cores makes that for every application there is a fit, it also means that some dedicated cores are unused, while others are intensively used. In order to optimize the occupation of the dedicated cores, it is a challenge for the hardware designer to use the dedicated cores for applications that they were not originally intended for. This paper is inspired by the work of Drimer et al. [1], which implements the Advanced Encryption Standard (AES) [2] on a Xilinx Virtex-5 device, mostly using DSP slices and Block RAM (BRAM) cores. Since symmetric ciphers are almost always used in combination with a mode of operation, we extend [1] by implementing both AES and the Galois/Counter Mode (GCM) [3] on a recent Xilinx Virtex UltraScale+ FPGA [4], still concentrating on maximizing the use of DSP slices and BRAM cores while minimizing the use of flip-flops (FFs) and look-up tables (LUTs).

## II. Implementation

Two architectures are implemented: a round-based architecture, performing both AES and the multiplication in 10 cycles, and a fully unrolled pipelined architecture, executing both parts in one cycle. The flow of the pipelined unrolled architecture is the same as the round-based architecture. The only difference is that all rounds are implemented in an unrolled pipelined fashion for faster execution with more area utilization. Since all round constants are fixed in the unrolled version, there is no need for control logic. Fig. 1 shows the overall AES-GCM FPGA architecture.

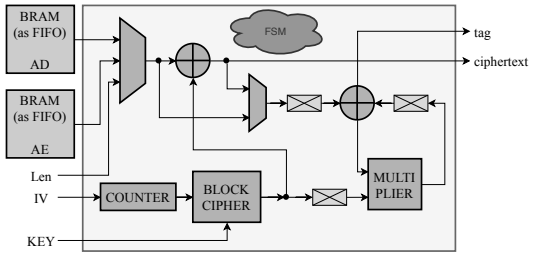


Fig. 1. The overall AES-GCM architecture, in which AD and AE stand for associated data and data for authenticated encryption, respectively. The block cipher is the AES core and the multiplier is the GCM core.

### A. AES Core

Drimer et al. used the AES optimization technique T-table approach [2]. In the LUT realization of AES SubBytes step, it is possible to append the MixColumns step together with ShiftRows to the LUT and re-define the table as 8-bit input, 32-bit output. T-table is defined as $y \rightarrow (2S(x), S(x), S(x), 3S(x))$ or as its shifted versions due to ShiftRows as a result of the multiplication with the MixColumns coefficients. Using T-tables, it is possible to use the existing BRAM resources on an FPGA efficiently while minimizing LUT and FF utilization. Furthermore, as in the case of Drimer et al., DSP slices are also used to realize certain AES steps which again results in less LUT utilization. Our AES core takes 128-bit data and key as inputs. We define two types of T-tables for data substitution, $T$ and $T'$ (8K each), where $T$ and $T'$ correspond to the concatenation of $2S$, $S$, $S$, $3S$ and $S$, 0, 0, 0, respectively. We do not need decryption thanks to GCM mode, which enables us to implement the last round without using any additional logic. AES core block diagram can be seen in Fig. 2. For the key scheduling part, additional BRAMs are required due to 4 Sbox calls in the last 32-bit word of the round key. The rc values for the 8-bit round constant addition is also stored in BRAM. We also use BRAMs instead of registers by storing a one-to-one mapping LUT, which we call "Bbox".

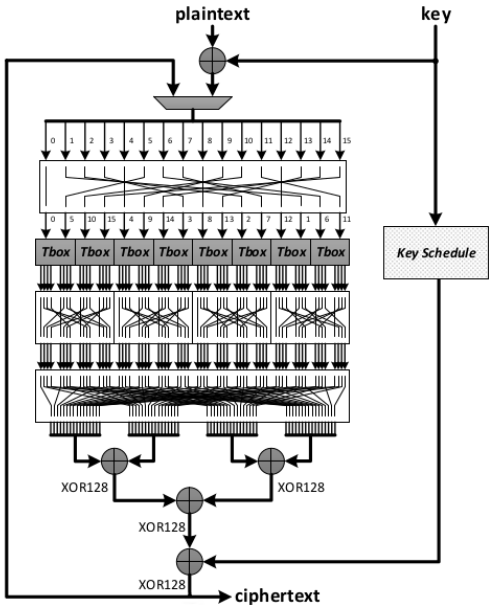

Fig. 2. Block diagram of the AES core

### B. GCM Core

The DSP slices on an UltraScale+ can have the +/-/l operator in the second stage perform a 3-input XOR. This new feature is utilized to execute the 3-input addition in the for-loop of the GCM algorithm's GF($2^{128}$) multiplier. The other operations in the for-loop are a multiplication with $x$, i.e. a shift operation that is handled through the routing outside of the DSP slice, and multiplications of a GF(2) element with a GF($2^{128}$) element, namely $a_i \cdot B(x)$ and $t_m \cdot P(x)$. We use a multiplexer that has the GF(2) element at its selection input and the GF($2^{128}$) element at one of its data inputs, while the second data input is fed with zeros. This leads to the DSP slice mapping in the top part of Fig. 3, showing the two multiplications through two multiplexers and the addition through a three-input XOR. In order to map this operation on the 48-bit DSP slices, we need at least three slices ($3*48 > 128$). The bottom DSP slice in the figure is used to store the value of the irreducible polynomial $P$ (which is fixed in the GCM specification). This way, the connection of $P$ makes use of the dedicated routing in between the DSP slices. This is shown in Fig. 3, in which the register that stores the bits of $P$ is indicated with a rectangle with a cross inside. Since $P$ is sparse ($P = x^{128} + x^7 + x^2 + x + 1$), we only need one DSP slice for the most-significant bits. In total, this leads to 3 DSP slices utilizing multiplexers and an XOR (as shown in the top part of Fig. 3), and one DSP slice utilizing a register (as shown in the bottom part of Fig. 3), thus 4 DSP slices in total for one the GF($2^{128}$) multiplication. To execute a 128-bit wide modular multiplication in a single clock cycle, 512 (128 $\times$ 4) DSP slices are required. When performing one operation in 10 clock cycles, 52 (13$\times$ 4) DSP slices are required. These two variants are used in the unrolled and round-based AES-GCM implementations, respectively.

## III. Results

We evaluate the implementation properties of our AES-GCM architectures using Xilinx Vivado 2017.3 suite after placement and routing. The target platform is the ZCU102 evaluation kit which contains a Zynq UltraScale+ FPGA. The results are shown in Table I, in which the glue logic represents the additional registers, multiplexers and FSM.

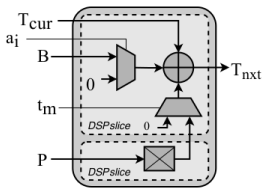

Fig. 3. The mapping of the operations on the DSP slices.

TABLE I
Implementation results of AES-GCM on ZCU102

| | LUT | FF | BRAM | DSP | $T_{min}$(ns) |
|---|---|---|---|---|---|
| unrolled PL | 899 | 1036 | 139 | 685 | 172 |
| AES | 192 | 0 | 135 | 173 | |
| MULTIPLIER | 325 | 401 | 0 | 512 | |
| glue | 682 | 635 | 4 | 0 | |
| round based | 785 | 1043 | 17.5 | 72 | 20 |
| AES | 196 | 4 | 13.5 | 20 | |
| MULTIPLIER | 156 | 398 | 0 | 52 | |
| glue | 433 | 641 | 4 | 0 | |

With respect to LUT and FF utilization, our design drastically outperforms the smallest AES-GCM implementation on FPGA, presented by Zhou et al. in [5], which reports 4628 slices on a Virtex-5 FPGA. Knowing that each slice contains 4 LUTs and 4 FFs, our implementation reduces the occupation of the LUTs and FFs by a factor 20. With respect to performance, Table I shows that both designs have a very large critical path, introducing a performance degradation in comparison to [5], which reports a maximum clock frequency of 324 MHz and thus a critical path of 3 ns.

## IV. Conclusion

We present AES-GCM architectures, tailored towards the optimal use of DSP slices and BRAM blocks. The implementation results show that we manage to reduce the occupied LUTs and FFs by a factor 20 in comparison to the smallest known AES-GCM implementation on FPGA (to our knowledge). However, the use of the DSP slices results in a relatively long critical path in the GF($2^{128}$) multiplier, leading to a significant performance degradation in comparison to related work. We can therefore conclude that the architectures proposed in this work are mainly interesting to be added as IP cores to FPGA applications that already occupy many LUTs and FFs, but have a lot of free DSP slices and BRAM tiles.

## V. Acknowledgement

This work was partially funded by the DRASTIC project (CELSA/17/033).

## References

[1] S. Drimer, T. Güneysu, and C. Paar, "DSPs, BRAMs and a Pinch of Logic: New Recipes for AES on FPGAs," in *FCCM*, pp. 99–108, IEEE, 2008.
[2] J. Daemen and V. Rijmen, *The Design of Rijndael*. Berlin, Heidelberg: Springer-Verlag, 2002.
[3] (NIST), "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," Tech. Rep. SP800-38D, U.S. Department of Commerce, 2007.
[4] Xilinx, "UltraScale Architecture DSP Slice User Guide UG473," 2019.
[5] G. Zhou, H. Michalik, and L. Hinsenkamp, "Improving Throughput of AES-GCM with Pipelined Karatsuba Multipliers on FPGAs," in *ARC*, pp. 193–203, Springer, 2009.