Accelerating Cryptographic Algorithms on RISC-V cores using Carryless Multiplication

Simi Sukumaran*, Tripti S Warrier
CarS Lab, Department of Electronics, CUSAT
Cochin, India
simisukumaran@cusat.ac.in, tripti@cusat.ac.in

Babu P S, Neel Gala
Incore Semiconductors
Chennai, India
babu.ps@incoresemi.com, neelgala@incoresemi.com

Abstract— **Edge computing emerges as a critical paradigm in the wake of Internet of Things (IoT) and 5G New Radio (5GNR). It catalyzes the demand for energy-efficient devices that have resilient CPUs with lean physical footprints. Mitigating the security challenges in these networked devices necessitates Bit Manipulation Instruction (BMI) inclusive architectures to improve Galois Field (GF) arithmetic, which is a fundamental step for most cryptographic algorithms. All major Instruction Set Architectures (ISA), including RISC-V incorporate dedicated instructions for carryless multiplication, recognizing its significant contribution in cryptographic applications. Acknowledging the fact, this paper introduces a novel approach to enhance the performance of GF arithmetic using carryless multiplication. The approach presents a promising avenue by improving the execution cycle counts of a real-world cryptographic application like the Advanced Encryption Scheme (AES) and can be scaled to all GF-based cryptographic algorithms. The proposed GF algorithm effectively maps the Carryless Multiplication Instruction of the ratified RISC-V Zbc extension. Evaluations indicate about 4.5x performance improvement for multiple schemes of AES using an open-source RISC-V core (SweRV-EL2™ 1.3) without incurring any additional overhead in terms of area as well as compiler support.**

Keywords- Cryptography; Galois Field Arithmetic; RISC-V; AES;

I. INTRODUCTION

The Internet of Things (IoT) [1] and communication network topologies, such as 5G New Radio (5GNR) [2], are driving the transition from cloud to edge computing. The stringent architectural constraints of portable devices necessitate optimal performance at the least cost of area and power. This motivates researchers to derive the best from current designs through modifications to enhance crucial and frequent applications such as cryptography and error correction. Most measures that tackle the security and safety concerns according to today's device design trends rely predominantly on bit manipulation algorithms. The RISC-V community has been working on the introduction of BMI for the past couple of years and has been successful in ratifying four key BMI sub-extensions: Zba/b/c/s [3] for which GNU Compiler Collection (GCC) support is also available. The Zbc extension endows carryless multiplication (CLMUL) instructions, engaging which the GF arithmetic and any allied cryptographic algorithms like Advanced Encryption Standard (AES), Reed-Solomn and even post quantum cryptographic algorithms can be accelerated.

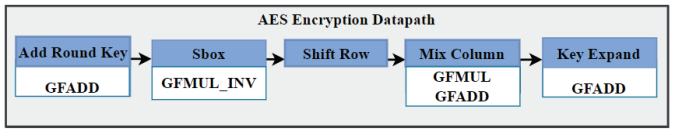

Figure 1: Datapath showing GF operations in AES encryption process

AES plays a critical role in minimizing the security vulnerabilities among networked IoT devices by guaranteeing secure encrypted communication. The functional breakdown of AES reveals strong reliance on GF arithmetic. Fig. 1 depicts the GF arithmetic units employed during different stages of AES encryption. Similarly, most of the cryptographic schemes have repeated instances of Galois Field Multiplication (*GFMUL*), Inversion (*GFMUL_INV*), and Addition (*GFADD*). This promotes the development of flexible galois field processors [4], dedicated hardware and hardware-software co-designs [5] to enhance the performance and security of edge devices. The *GFADD* represents a simple xor operation, whereas the *GFMUL* corresponds to a carryless multiplication followed by reduction logic [6], as illustrated in Fig. 2. Inversion uses the repeated square and multiply approach [7], which also aligns with the requirement for optimum Galois Field Multiplication.

The above facts demonstrate that optimized GF arithmetic employing CLMUL implementations provides the necessary impetus to boost cryptographic applications. Inspired by these observations, this work proposes a novel approach to modify the GFMUL algorithm using three CLMUL instructions, one for multiplication and the rest for polynomial reduction. Further, we simulate, validate and evaluate modified GFMUL and GFMUL_INV algorithms on SweRV-EL2 core with inbuilt Zbc extension [3] and measure the performance improvement

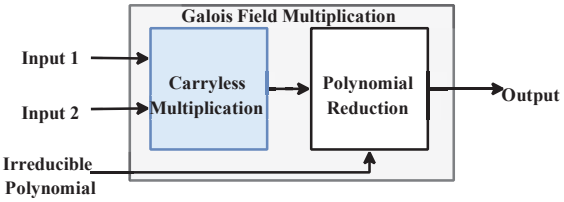

Figure 2: Galois Field multiplication using two stage logic

obtained on a real-world cryptographic application - AES. Our work stands distinct from the state-of -art works by exploiting the existing CLMUL instructions for polynomial reduction rather than custom instructions in RISC-V processors. Hence, it eliminates the modification of the RISC-V GNU toolchain and exempts any area overhead on the core for incorporating such custom instructions.

II. Proposed Work

The lower footprint requirement of lean embedded processors, along with the RISC-V GCC compiler's failure to render the CLMUL (carryless multiplication) instruction in embedded and crypto benchmarks emphasize the need to quantify the significance of these instructions. This section modifies the basic GF multiplication method utilizing carryless multiplication and maps it to AES- an essential cryptographic application.

A. Mathematical Basis of GF(2^m)

Popularly known as binary extension field, GF(2^m) represents elements with polynomial degrees up to m-1. The polynomial representation of GF(2^m) is shown in Equation (1). It involves a unit element (α^0), a zero element ($\alpha^{-\infty}$), and a primitive element (α). Equation (2) represents the irreducible polynomial f(x) associated with GF(2^m), where α (primitive element) forms its root.

$$F(\propto) = a_{m-1} \propto^{m-1} + \cdots . + a_1 \propto + a_0 ; \quad a_i \in GF(2), 0 \leq i \leq m-1 \tag{1}$$

$$f(x) = x_m + f_{m-1}x_{m-1} + \cdots + f_1 x + f0; \tag{2}$$

Arithmetic operations in GF(2^m) mandate a polynomial reduction, equivalent to the modulo using the irreducible field polynomial when the degree of the result exceeds m-1. All the arithmetic operations explained below use a(x) and b(x) represented by Equation (3) as inputs.

$$a(x) = a_{m-1}x^{m-1} + \cdots + a_1 x + a_0$$
$$b(x) = b_{m-1}x^{m-1} + \cdots + b_1 x + b_0 \tag{3}$$
$$a_{i,} \in GF(2), 0 \leq i \leq m-1$$

GFADD is a simple, yet most repeatedly used operation in cryptographic algorithms. It carries out direct bitwise XOR of the corresponding coefficients of the inputs as in Equation (4) and does not require polynomial reduction as it fits in the field.

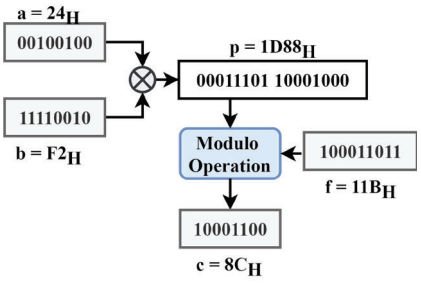

Figure 3: GF(2^8) Multiplication using standard logic

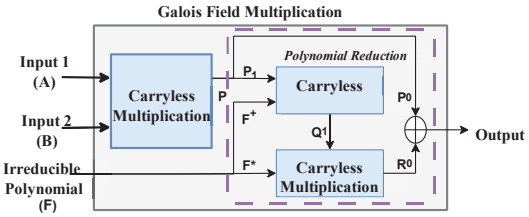

Figure 4: Modified Galois Field multiplication logic. Polynomial reduction is also broken down to make use of carryless multiplication. P, Q, and R are intermediate carryless multiplication results.

$$s(x) = (a_{m-1} \oplus b_{m-1})x^{m-1} + \cdots + (a_0 \oplus b_0) \tag{4}$$

GFMUL is a two-step process comprising of a carryless multiplication followed by polynomial reduction, as shown in Fig. 2. The result of carryless multiplication is of degree 2m-2 and should be reduced using the irreducible polynomial to keep the final result c(x)=p(x) mod f(x) within the field. Fig. 3 represents a GF(2^8) example of *GFMUL* with field polynomial $f(x) = x^8+x^4+x^3+x+1$ (11BH).

B. Modified Galois Multiplication Algorithm

Galois Multiplication being a complex but inevitable operation in cryptography is explored and implemented in different ways. Kuo et al. employs an appealing two-step process using carryless multiplication followed by polynomial reduction as shown in Fig. 2 [6]. The work in this paper modifies the *GFMUL* logic using three carryless multiplications as shown in Fig. 4. Here the polynomial reduction is also broken down into two carryless multiplications. The mathematical grounds of this mapping is derived from the explanations given by Gureon et al., but is not included here due to space considerations [8].

The Algorithm 1 gives a stepwise walk through this modified approach with two N-bit inputs (of degree N-1) and an N+1 bit field polynomial (of degree N). As long as the degrees of inputs (A and B) and field polynomial (F) are N-1 and N, respectively, F+ and F remain the same. Hence, using the proposed algorithm, the *GFMUL* result can be attained with three carryless multiplications and minor logic to extract the MSB and LSB of the intermediate results. Fig. 5 clearly demonstrates the steps depicted by Algorithm 1 using GF(2^8). This degree is chosen because many of the real world cryptographic applications, like AES make use of GF(2^8) with Field polynomial as $F(x)=x^8+x^4+x^3+x+1$ (11BH). Here, as F is of degree 8 and A, B are of degree 7, F^+ is same as F (11BH) and its lower significant Byte (lsB) forms F* (1BH). The results for the modified algorithm

Algorithm 1: Proposed GFMUL Algorithm

Input: A,B : N bits, Field Polynomial (F): N+1 bits
Output: C : N bits

STEP 1: P : [P1:P0] = carryless multiplication(A,B)
STEP 2: Q : [Q1:Q0] = carryless multiplication(P1,F^+)
where F^+ = result of X^{2N} divided by F
STEP 3: R : [R1:R0] = carryless multiplication(Q1,F*)
where F*= lower order N bits of F
STEP 4: Output : C = R0 xor P0

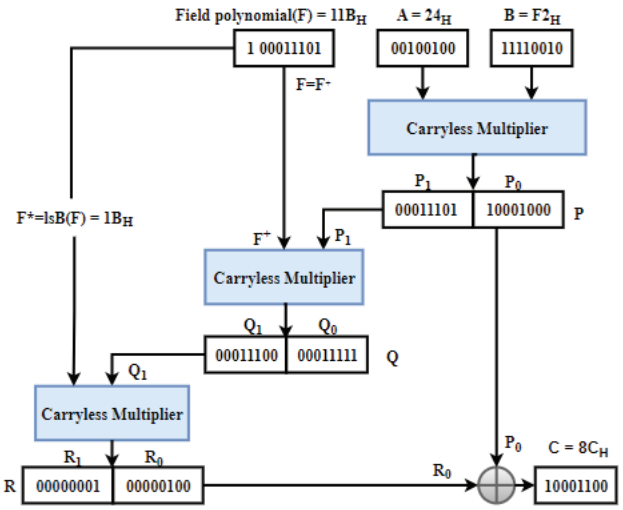

Figure 5: An eight bit *GFMUL* example using the modified algorithm

are verified for all possible 8-bit input combinations against the standard algorithm. The modified *GFMUL* algorithm when compiled for GF(2^8) using *RISC-V GCC compiler*, generates the instruction disassembly which gives three instances of CLMUL instruction as expected. This algorithm can be further extended to higher polynomial degrees but gets limited with the width of the RISC-V register.

C. Application mapping and evaluation on SWERV-EL2

The proposed *GFMUL* algorithm using CLMUL instruction can be used to accelerate various application that use GF(2^8) arithmetic operations. AES is the GF(2^8) based cryptographic application selected for the performance evaluation and is applied to multiple encryption schemes of AES like CBC, ECB and CTR for key-lengths 128, 192 and 256. SweRV-EL2 known as VeeR-EL2 at the time of writing is an open-source RISC-V core from Western Digital with support for the ratified bit manipulation extensions (Zba/b/c/s). Thus, by default, it has the hardware for carryless multiplication in the execution pipeline and deprecates any extra hardware for executing the proposed algorithm.

The notion of the work is to reduce the cycles of execution taken by the *GFMUL* in cryptographic applications using the CLMUL instructions. The evaluation strategy hence involves a standard variant that is compiled and executed using basic RISC-V instructions and a proposed variant, that requires the Zbc instruction support, that is available in SWERV-EL2.The AES algorithm in the repository mentioned in Table I has been modified to accommodate these two variants (Standard and Proposed) for three encryption Schemes (CBC, CTR and ECB) using multiple key lengths. Table I shows the record of tools, its configurations and the repositories used in this work. The simulation environment consists of a verilated model using SWERV-EL2 core connected to a virtual memory via AXI bus as shown in Fig. 6. AES is modified to have versions with and without CLMUL instruction based GF arithmetic, compiled by the RISC-V GNU toolchain and the resulting binary is fed into virtual memory for simulation. The verilated top module is modified to display the number of cycles executed and the total instructions retired in the console output for each run. This is accomplished by reading the values of MCYCLE and MINSRET controls status registers (CSR) of RISC-V ISA using the CSR read instruction. The counts corresponding to the standard and the proposed CLMUL implementation are extracted to calculate the percentage reduction in cycles executed. All the steps followed for the simulation here are available in the open-source SweRV repository [9].

TABLE I. RECORD OF TOOLS AND REPOSITORIES USED FOR EVALUATION

Core	
SWERV-EL2	Core™ 1.3: RISC-V from Western Digital Target= typical_pd [No ICCm, AXI4 bus interface] Frequency= 25 Mhz
Synthesis and Implementation	
Vivado	Version=v2023.2 (64-bit)
Target board	Nexys 4 DDR; Part: XC7A100T-1CSG324C
Simulation	
Verilator	v4.212
RISC-V GNU Toolcahin	32 bit GCC: version:13.2.0(gc891d8dc23e) Compiler flags: -mabi=ilp32 -march=rv32imac_zbc_zicsr -O3 -fomit-frame-pointer -fPIC -no-pie
Algorithms	
AES	Repository: https://github.com/kokke/tiny-AES-c

TABLE II. CLOCK CYCLES AND INSTRUCTION COUNTS FOR GFMUL

Function	Cycles		Instructions		Reduction (%)	
	Standard	Proposed	Standard	Proposed	Cycle	Instr
GFMUL						
GF(2^4)	128	57	64	21	55.47	67.19
GF(2^8)	175	63	106	21	64	80.19
GF(2^{16})	344	65	240	26	81.1	89.17
GFMUL_INV						
GF(2^4)	378	131	296	85	65.34	71.28
GF(2^8)	1250	274	1126	223	78.08	80.2
GF(2^{16})	6356	631	5928	573	90.07	90.33

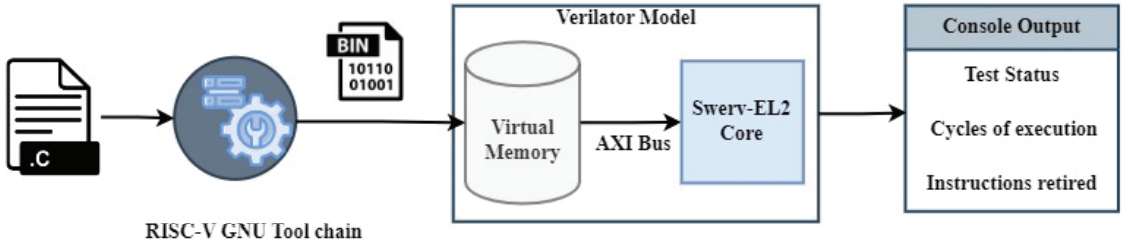

Figure 6: Verilated model of SWERV-EL2 for simulation of GF operations in AES encryption

III. RESULTS AND DISCUSSIONS

This section evaluates the performance of the proposed modification in Galois multiplication in terms of execution cycles and retired instructions. Though this work lays focus on GF(2^8) owing to the degree of field in AES , the proposed algorithm is scalable to other polynomial degrees as well. Table II shows the count of execution cycles and instructions returned count of *GFMUL* and *GFMUL_INV* for degrees 4, 8, and 16. It shows that the percentage of reduction in cycle count increases from 55% to 81% as the degree of polynomial or the number of bits in the arithmetic logic increases from 4 to 16. This trend makes it evident that the performance of the modified algorithm improves with higher degrees of Galois field arithmetic. Similarly, the inversion algorithm is modified using the proposed *GFMUL* and examined for multiple degrees. The results show cycle count reduction of 65%, 78% and 90% for degrees 4, 8 and 16 respectively over the standard execution.

CBC, ECB and CTR schemes of AES exhibits similar results of more than 75% reduction in execution cycles for encryption and decryption algorithms as portrayed by Fig. 7. The results marks a comparable performance with respect to the state-of-the-art (SOTA) solution with a small depreciation in cycle reduction [6]. This stems from the absence of dedicated hardware for polynomial reduction using custom instructions. In SOTA, the polynomial reduction logic as such is substituted with FFRED instruction (Custom), which incurs an area of 7% equivalent to 268 LUT's on NEXYS DDR4 FPGA board. On contrary, the proposed approach does not add any additional hardware and uses the already existing carryless multiplier twice for polynomial reduction. These additional instances of CLMUL instructions in the algorithm account for the aforementioned depreciation in the perfomance.

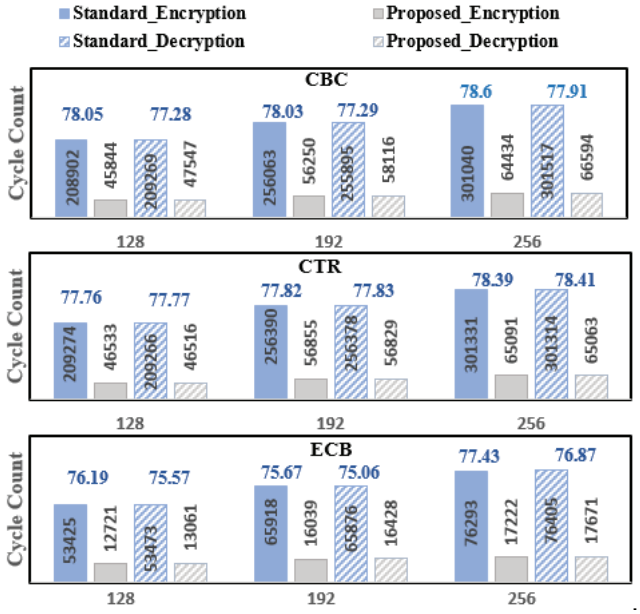

Figure 7: Plot of the Cycles counts for CBC, CTR and ECB schemes of AES for Standard and Proposed execution for various keylengths. The numbers on top of the bars indicate the percentage reduction in cycle count achieved by the proposed method corresponding to the absolute bar values.

Nevertheless, the proposed variant shows about 4.5x compared to the standard approach for AES, without any hardware modification to the existing core and also eliminates the need for compiler and ISA modifications and associated hardware overheads for custom instructions.

IV. CONCLUSION

The Security concerns on edge devices, initiate tailored optimizations of general-purpose CPUs for accelerated cryptographic applications. Galois field arithmetic - GF(2^m), is a critical step in pre and post-quantum encryption and error-correcting codes. Carryless multiplication enhances GF arithmetic, making the Zbc extension of RISC-V BMI valuable for accelerating GF-based cryptographic algorithms. This work proposes an approach to modify the Galois multiplication and related arithmetic using carryless multiplication to map the same to RISC-V ISA. Our analysis using an open-source RISC-V core revealed a significant decrease in clock cycles of over 75% for various AES schemes with a 25% decrease in static code size without the need for any additional hardware for custom instructions and related compiler overheads. Future work aims to explore carryless multiplier designs, incorporate them into an in-house RISC-V core, and evaluate more real-world applications like McEliece and Reed-Solomn.

ACKNOWLEDGMENT

This research is supported in part by C2S Project scheme of Ministry of Electronics and Information Technology (MeitY), Government of India vide project grant EE-9/2/2021-R&D-E.

REFERENCES

[1] L. Tan and N. Wang, "Future internet: The internet of things," in 2010 3rd international conference on advanced computer theory and engineering (ICACTE), vol. 5, pp. V5–376, IEEE, 2010.

[2] T. Huang, W. Yang, J. Wu, J. Ma, X. Zhang, and D. Zhang, "A survey on green 6g network: Architecture and technologies," IEEE access, vol. 7, pp. 175758–175768, 2019.

[3] R. International, "Risc-v bit-manipulation isa-extensions." https://github.com/riscv/riscv-bitmanip/blob/main/bitmanip/bitmanip.adoc, 2022.

[4] Y. Chen, S. Lu, C. Fu, D. Blaauw, R. Dreslinski Jr, T. Mudge, and H.-S. Kim, "A programmable galois field processor for the internet of things," in Proceedings of the 44th Annual International Symposium on Computer Architecture, pp. 55–68, 2017.

[5] W.-M. Lim and M. Benaissa, "Design space exploration of a hardware-software co-designed gf (2m) galois field processor for forward error correction and cryptography," in Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pp. 53–58, 2003.

[6] Y.-M. Kuo, F. Garcia-Herrero, O. Ruano, and J. A. Maestro, "Riscv galois field isa extension for non-binary error-correction codes and classical and post-quantum cryptography," IEEE Transactions on Computers, vol. 72, no. 3, pp. 682–692, 2023.

[7] X. Zhang, VLSI architectures for modern error-correcting codes. Crc Press, 2017.

[8] S. Gueron and M. Kounavis, "Efficient implementation of the galois counter mode using a carry-less multiplier and a fast reduction algorithm," Information Processing Letters, vol. 110, no. 14-15, pp. 549‑553, 2010.

[9] W. D. Corporation, "Risc-v swerv-el2 github repository." https://github.com/chipsalliance/Cores-SweRV-EL2, 2020.