VLSI design of Advanced-Features AES CryptoProcessor in the framework of the European Processor Initiative

Pietro Nannipieri, *Member, IEEE,* Stefano Di Matteo, Luca Baldanzi, Luca Crocetti, Luca Zulberti , *Student Member, IEEE,* Sergio Saponara, *Senior Member, IEEE,* and Luca Fanucci, *Fellow, IEEE,*

***Abstract*—This paper presents a cryptographic hardware accelerator supporting multiple AES based block cypher modes, including the more advanced CMAC, CCM, GCM and XTS modes. The proposed design implements advanced and innovative features in hardware, such as AES key secure management, on-chip clock randomization and access privilege mechanisms. The system has been tested in a RISC-V based System on Chip, specifically designed for this purpose, on an Ultrascale+ Xilinx FPGA, analysing resource and power consumption, together with system performances. The Cryptoprocessor has been then synthesised on a 7nm CMOS standard-cells technology; performances, complexity and power consumption information are analysed and compared with the state of the art. The proposed Cryptoprocessor is ready to be embedded within the innovative European Processor Initiative chip.**

***Index Terms*—AES, RISC-V, EPI, Cryptoprocessor**

I. Introduction

Nowadays the demand for secure data exchange is ever-growing. For this reason, cryptography has become an essential component of modern electronic systems.

Generally, an embedded system is equipped with a microprocessor in charge of executing the software; this can be a valid approach to perform security algorithms. Nevertheless, in specific application cases, the performance of the embedded microprocessor could be insufficient to comply with the latency and/or throughput requirements, for example in real-time IoT applications [1]. In these cases, a valid solution could be to privilege hardware (HW) accelerated rather than pure software (SW) implementations, for security algorithms. It is from these considerations, together with the need for a European platform for computing, that the European Processor Initiative (EPI) [2] has been launched. EPI is an H2020 project under development, started in 2018, whose aim is to design and implement a new family of low-power European processors for exascale computing, high-performance Big-Data and a range of emerging applications, including automotive [3]. Robustness and safety issues play a significant role in the framework of the targeted markets, especially automotive. The EPI processor is expected to answer the currently unaddressed need of proper architecture, at both hardware and software levels, to support the requirements of current and future cybersecurity standards and certifications.

The proposed EPI hardware security architecture relies on isolated blocks of the circuit, with specific hardware and dedicated private resources for security responsibility. Figure 1 illustrates the proposed EPI hardware security subsystem architecture, which foresees the *Cryptoprocessor-Tile* (CT) with multiple hardware cryptographic accelerators including Advanced Encryption Standard (AES), Secure Hash Algorithm (SHA)[4], Elliptic-Curve Cryptography (ECC)[5], and Random Number Generation (RNG)[6] [7]. All is controlled by a secure Micro-Controller Unit (MCU), which is a RISC-V processor.

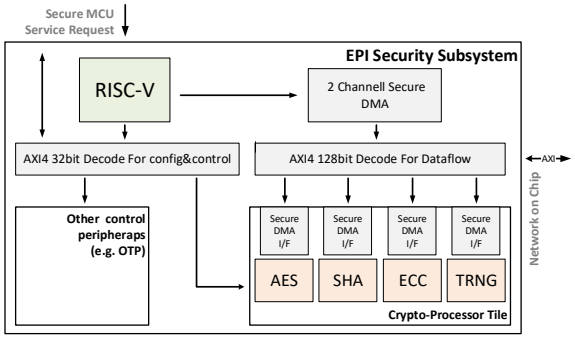

Fig. 1. High-level EPI Security Subsystem Architecture.

The security cryptoprocessor can be used both exploiting the secure MCU interface and also a dedicated secure Direct Memory Access (DMA) connection.

Among the available cryptographic functions, the AES [8] is probably the most popular symmetric-key algorithm. HW implementations of the AES are usually preferred to SW ones [9] in performance and power critical environments: with dedicated hardware, it is possible to achieve better performance with lower power consumption. They are also often coupled with other more advanced cryptographic algorithms, such as SHA3 [10] or as Authenticated Encryption with Associated Data (AEAD) services [11]. The proposed work belongs to the wider EPI CT design project: we aim to illustrate and comment on the architecture for the AES cryptoprocessor, embedded in the CT of the breaking-trough EPI processor, focusing on the main innovation introduced. Being the AES standard well established, several solutions already address the problem of the AES hardware acceleration [12] [13] [14]. There are also Systems on Chip available in the literature, but either they are performance limited or they lack the in-hardware support

Editorial version is available at 10.1109/TVLSI.2021.3129107
Manuscript received XXX, XXX; revised XXX.

for advanced features. [15] is a RISC-V based SoC with AES hardware accelerator, but its features are not declared; it is not clear which modes are supported, with which performances, making it impossible to carry out a fair and solid comparison. [16] is a very interesting implementation, like our proposed system. However, the AES processor is barely described and does not support all the features we intended to support (it seems to support only GCM). It is interesting to observe that also here the approach of the loosely coupled accelerator has been followed, also for much larger accelerators such as the ECC which requires a lot of hardware resources. [17] is a very structured solution but also here, probably since it is a commercial product, no information is given on the internal architecture and the features supported in hardware. Similar consideration applies to [18]. The main innovation of this work is to present an implementation of a complete AES cryptoprocessor, fully hardware-accelerated with advanced features, integrated with a RISC-V processor, to be employed within the upcoming EPI processor; our contributions include, but are not limited, to:

- Architectural design of the AES cryptoprocessor inside the CT, a hardware accelerator compliant to AES standard supporting 9 different block cipher modes, ensuring not only confidentiality but also integrity and AEAD.
- Hardware implementation of innovative and advanced features that are not usually hardware-accelerated: policies for secure keys storage, employment of clock randomization technique on the cryptoprocessor, the control mechanism for configuring the cryptoprocessor, preemption of cryptographic operations, panic mechanism.
- Design of a System on Chip with a RISC-V processor coupled with the AES Cryptoprocessor as a hardware accelerator. There are others examples in literature where a form of AES hardware acceleration is coupled with a RISC-V processor [19], [20], [21], [22]. Most of the available solutions are tightly coupled hardware acceleration, which exploits the extendible ISA of the RISC-V processor to accelerate in hardware small primitives. The tightly coupled approach requires greater efforts and better exploit the flexibility of the RISC-V processor. It requires a very small hardware overhead that needs to be inserted in the processor pipeline. However, the achievable improvements in terms of performances are more limited, since it is not possible to add too much logic to the processor pipeline without affecting its timing performances. Our approach explores a different region of the design space: the innovation of our proposed system is the support for advanced hardware features, such as the key management, that would not be possible to be realised in hardware within the processor pipeline. In addition, we aim for hardware acceleration of different operating modes to maximise achievable throughput. The result is a more complex design that needs to be interfaced as a separate hardware accelerator, requiring more hardware resources but achieving greater performance and more advanced features.
- System On Chip FPGA verification on a Xilinx Ultrascale+ board with preliminary analysis on complexity, throughput & performances.
- Implementation on the cutting-edge 7nm TSMC silicon technology (target technology for the EPI processor, first contribution available on this technology to the best of our knowledge), with a complete analysis of complexity, throughput, and power dissipation.

Cipher Mode	Confidentiality	Integrity	Authenticity
AES-ECB	✓	✗	✗
AES-CBC	✓	✗	✗
AES-OFB	✓	✗	✗
AES-CFB	✓	✗	✗
AES-CTR	✓	✗	✗
AES-CMAC	✗	✓	✗
AES-GCM	✓	✓	✓
AES-CCM	✓	✓	✓
AES-XTS	✓	✗	✗

TABLE I
SUPPORTED BLOCK CIPHER MODES AND RELATED PROPERTIES. ELECTRONIC CODE BOOK (ECB); CIPHER BLOCK CHAINING (CBC); CIPHER FEEDBACK (CFB); OUTPUT FEEDBACK (OFB); COUNTER (CTR); CIPHER BASED MAC (CMAC); COUNTER WITH CBC-MAC (CCM); GALOIS COUNTER MODE (GCM); XOR-ENCRYPT-XOR-BASED TWEAKED-CODEBOOK MODE WITH CIPHERTEXT STEALING (XTS).

After this introduction, the rest of the paper is organised as follows. In Section II we present and define the cryptoprocessor architecture, focusing on the innovative mechanisms (key management mechanism, clock randomization, etc..). In Section III we illustrate the architecture of the AES core embedded in the cryptoprocessor and we illustrate the architecture of the AES engine, focusing on the key and more innovative architectural portions. In Section IV we present the innovative System on Chip platform used to validate the system, based on a RISC-V processor to emulate as much as possible the target EPI security sub-system architecture. In Section V the results on various silicon technologies, especially the 7nm are shown and compared with the results available in the literature. Finally, in Section VI we will conclude this work, highlighting the achieved results.

II. AES CRYPTOPROCESSOR ADVANCED HARDWARE FEATURES

The Cryptoprocessor-Tile offers security services and algorithms based on symmetric-key and public-key cryptography, hash functions, and random numbers generation, aiming to guarantee data confidentiality, integrity, authenticity, authentication, and signature services. The AES cryptoprocessor inside the crypto-tile is responsible for the symmetric-key algorithms, supporting the AES cypher and the modes of operations reported in Table I, both in the case of AES-128 and AES-256. The architecture of the AES cryptoprocessor (Figure 2) is composed of the interface registers to handle the cryptographic operations (i.e. Configuration, Control, Error and Status registers), a state machine (i.e. FSM), data, and key registers, and the AES engine that effectively computes the AES cypher and the supported modes of operations.

A. AES cryptoprocessor functionalities

The AES cryptoprocessor integrates and offers several enhanced functionalities that increase the robustness of security boundaries and strength the resistance against misuse, unauthorised accesses, and physical attacks.

Other than six key slots that can be employed, for instance, to install 3 couples of keys for different scopes (e.g. chip keys, software context keys, and application keys), the AES cryptoprocessor also provides the possibility to redirect the output of the underlying AES engine onto internal key slots, thus allowing the user to derive and generate secure keys which are not outsourced from the AES cryptoprocessor. This feature enables the employment of the AES cryptoprocessor as a building block of pure hardware or hardware and software Root-of-Trust (RoT).

To allow restriction mechanisms on the cryptoprocessor usage, two different Privilege Levels (i.e. Supervisor and User) can be used to access a specific set of registers. The Supervisor Level is responsible for configuring the cryptoprocessor at start-up, enabling or disabling features and functionalities (e.g. supported modes of operation, reserved key slots) as well as using it normally for cryptographic operations. The User Level cannot configure processor functionalities and can only be used for cryptographic operations with the key slots reserved for it. Such a privilege mechanism allows differentiating between two different execution contexts that are strictly independent. Only the Privilege Level of the current execution context can retrieve the whole information about it such as configuration and state of the ongoing cryptographic operation, as well as input and output data. Once a cryptographic operation configuration is initiated in a Privilege Level, only this level can manage it and terminate it. To prevent denial of access, only the Supervisor Level can abort a cryptographic operation initiated with the User Level, reporting a specific error. In addition, any error occurring on a cryptographic operation or a key slot is reported separately for each Privilege Level. The Privilege Level differentiation is implemented at the interface level, exploiting the AXI4 functionalities.

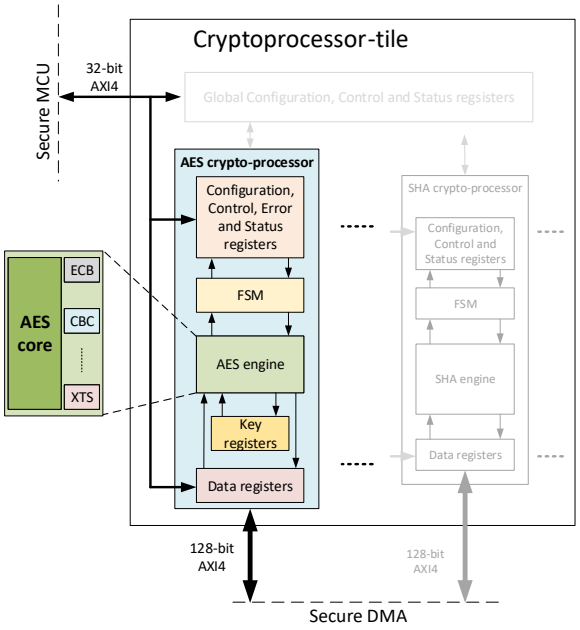

Fig. 2. AES Cryptoprocessor Inside the Crypto-Tile.

Also, the cryptoprocessor embeds control and handling mechanisms that ensure a proper configuration and usage of the module to perform cryptographic operation: a specific state-based flow has to be respected to successfully run an operation, and the evolution from one state to another one is managed by a set of actions that relies on strict control of authorization and privileges. Such mechanisms also, regulate the access to sensitive data, both input and output ones, allowing or forbidding such databasing not only on a Privilege Level base, but also on the state of the AES cryptoprocessor, and limiting the number of access to such data: in case multiple access to the same sensible data is performed, the cryptoprocessor prevent write or read of data and raises an error flag, moving to the error state.

To improve the flexibility and fit real-time application contexts, the AES cryptoprocessor supports also the preemption of cryptographic operations, allowing to halt the execution of a cryptographic operation and to perform the execution of another one with higher priority. The implementation of this feature requires the AES cryptoprocessor to allow access to some internal states of the cryptographic operation that is going to be suspended, to resume it later, once the execution of operation with higher priority has been completed. Halting an operation in progress occurs once the 128-bit block of data has been correctly processed by the cryptoprocessor, and the internal states and data are stored in proper auxiliary registers that can only be accessed according to restriction mechanisms for access to sensitive data. Also in this case the AES cryptoprocessor strictly regulates the access to internal states of cryptographic operations to improve security and counteract unauthorised or illegal usage of assets.

For the same purpose, the cryptoprocessor embeds also panic mechanisms that allow to partially or completely flush the sensible data installed into the cryptoprocessor itself (e.g., the secret keys), offering a specific interface to trigger the panic state (allowed only for the Supervisor Level) and to configure the actions to be performed. This panic mechanism is implemented to react immediately and with minimal assistance of software to manage events that may lead to a critical security breach. Two levels of panic mechanisms are implemented in the AES cryptoprocessor: partial-panic (level 1) and full-panic (level 2). The partial-panic mechanism stops immediately any cryptographic operation and resets all the input and output registers. Each of the key slots can be configured (before being populated with sensitive data) to be sensible to the partial-panic mechanism; in this case, if a partial-panic is triggered all the registers of that key slot are reset immediately. The full-panic mechanism stops and resets any ongoing cryptographic operation and resets all the data and key registers.

Finally, the AES cryptoprocessor embeds also dedicated resources for the local derivation and randomization of clock signal feeding the AES engine, starting from the external clock

source. This feature highly increases the robustness of cryptoprocessor to physical attacks such as Side-Channel Attacks (SCAs), which are based on analysis of power consumption (Power Analysis, PA) and electromagnetic emission (Electromagnetic Analysis, EMA). Randomization and manipulation of clock cycles alter the acquisition pattern an attacker has to gather and observe to retrieve secret data (usually the secret key), significantly increasing the acquisition time and the preprocessing resources an attacker must employ for a successful attack.

We opted for this countermeasure even if there are more secure ones, such as threshold implementation, as it almost does not affect system complexity and throughput, as shown in [23]. A higher level of security can be obtained with various techniques, including threshold implementation, such as described in [24], and can theoretically completely protect the system from side-channel attacks. Unfortunately, the price that needs to be paid for this protection is very high and grows with the desired degree of protection, affecting the system throughput and the dimension of the circuit. The clock randomisation technique proposed does not ensure complete protection against SCA; however, it improves its resistance as demonstrated in [25] [26] [27], with negligible costs in terms of hardware resources, better resistance compared to fixed clock solution. It is possible of course, but out of the scope of this paper, to exploit more complex techniques available in the literature to enrich the resistance level to side-channel attacks.

Figure 3 shows a functional block scheme of the clock randomization circuitry: for simplicity, control logic, reset, and exceptions signals are not reported. Such sub-module of AES cryptoprocessor is mainly composed of a multiplexer to select the clock source (i.e. from PLL or an on-board internal oscillator), a clock division stage, and finally the pure clock randomization stage. As depicted by the detail box at the bottom of the Figure 3, this last stage counts a pseudo-random number generator (PRNG, based on linear feedback shift register, LFSR, approach) that generates a random number (*rnd_num*) and a clock cycles counter that reports the actual number of elapsed clock cycles (*cc_cnt*) and raise the *update* flag when it reaches the total number of clock cycles (this parameter is configured employing AES cryptoprocessor configuration registers). When the random number and the number of elapsed clock cycles match, the *skip_cc* flag is asserted and the clock gating logic mask the clock cycle of clock signal *clk_div_x_gmux*; when the *update* flag is asserted, the clock cycles counter is cleared, while the PRNG block generates a new random number. In a complete Cryptoprocessor system, the PRNG resistance to SCA is fundamental. In this work, we focused on the AES core of a much wider Cryptoprocessor, as mentioned in Section I. However, in our vision we are aware of the importance of the topic: indeed, we designed also a TRNG to be exploited in our final Cryptoprocessor system [6] [7]. The PRNG used for clock randomisation is re-seeded periodically with TRNG values, to improve both entropy and SCA resistance.

For what concerns the compliance with standards defining security requirements for cryptographic modules, the definition of architecture and security features of both the AES

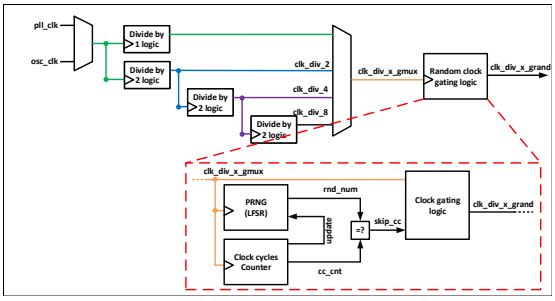

Fig. 3. Clock Division and Randomization Circuit Architecture.

cryptoprocessor and the whole Cryptoprocessor-Tile has been designed following the guideline of FIPS 140-2 [28]. The specifications of [28] cover different areas such as module ports and interfaces, finite state model, cryptographic key management, self-tests, and other aspect related also to physical security. The Cryptoprocessor-Tile employs only NIST-approved algorithms and modes of operations and is strictly physically isolated to the rest of the EPI chip. Authentication mechanisms are included to permit the usage of the entire Cryptoprocessor-Tile and of defined key slots of the AES cryptoprocessor adopting a seal/unseal procedure that requires secret tag values. All the possible operations and states of the AES cryptoprocessor are specified with a finite state model. Cryptographic key storage and management are included in the AES cryptoprocessor, which is in line with the indication provided by NIST. The connectivity with a RISC-V processor allows implementing Self-Test and Power-Up test through hardware-software cooperation. The security considerations related to physical and environmental aspects are out of the scope of this work.

B. Secure Keys Storage and Management

The AES cryptoprocessor embeds a dedicated set of registers to contain and manage the cypher keys through strict usage rules, detection, and management of error conditions and access restrictions. The population and usage of a key in a cryptoprocessor is a very sensitive operation. Key slot access and usage shall be protected to prevent any misusage due to hardware failures, software bugs, or attacks.

The key management registers are composed of the following set of registers implementing different security features:

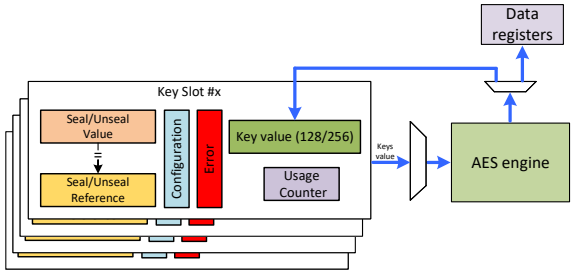

Fig. 4. AES Key Storage and Management Registers.

- Seal/unseal registers: store an access tag required to lock/unlock the key slot;
- Configuration registers: allow to configure the key slot;
- Errors registers: flag errors due to misusages of the key slots;
- Usage counter: allows to keep track of the number of cryptographic operations executed with the key;
- Key-value register: contains the value of the key or the value of the encrypted key;

Six independent key slots are supported to allow key management fully done inside the cryptoprocessor tile and allow population by different isolated parts of software upon specific chip lifecycles (e.g. some key slots are populated during chip boot, other during applicative context switch). The proposed key-management registers aim to handle errors and threats, offering different levels of protection.

1) Key slot protection: The access of each key slot can be reserved to a given Privilege Level. If a key is loaded by a given Privilege Level, it is aimed and shall be used only by that Privilege Level, until it is released by being cleared. Also, key slots access is protected by an access tag to prevent access or usage from an unexpected part of the software not knowing the tag value (it lowers the attack surface at the software level by requiring more than a given Privilege Level). Such an access tag is intended to allow access (set of configuration, usage rules, and value) to the key slot only to a specific, identified part of the software that knows the tag. The key slot supports the possibility to lock his configuration (including his value) until the next reset. This allow, for instance, to protect master keys of the chip that can be set at an early stage of boot. Every misusage of each key slot flags an error on the corresponding error register that must be cleared before using again the slot.

2) Key-value protection: A lot of attacks conducted through malicious software aim at trying to steal secret keys or to use such keys without authorization. The restrictions based on the privilege levels are clearly not sufficient to enforce proper protection. A solution proposed inside the AES cryptoprocessor is to have the application only manipulating half secret, i.e. encrypted keys, which are then deciphered only inside the cryptoprocessor and associated with needed usage rules. Each AES secret key slot can be directly populated with a specific configuration from the output of an AES cryptographic operation performed by the cryptoprocessor. This internal mechanism enforces that the application is never manipulating the plaintext key value. For this reason, different key slots are included to have some key slots reserved for this usage: as an example, an AES key slot may be dedicated to storing a dedicated chip unique key, and another slot can contain a key specific to the application process context (that is changed upon each context switch of the OS, thus enforcing strong isolation and data confidentiality between each application).

3) Key-value usage: To mitigate improper usage of a loaded key, each key slot integrates configuration elements that allow restricting its usage to specific algorithms and modes. There is a counter that reports the usage count accessed by the software, which allows it to effectively detect misuse or illegal access to the key, since at the end of the cryptographic operation it can check that the counter matches the expected value it calculated.

III. AES ENGINE

A. AES core

Only one AES core is instantiated and shared among all the implemented block cypher modes, thus only one mode at a time can be driven from the processor. The AES core is implemented with the most convenient trade-off between performance and complexity: only the functions belonging to one round are implemented and used iteratively for several rounds ranging from 10 to 14 depending on the key size. To save logic resources, only the 128 and 256-bit key lengths are supported. It is a remarkable fact that the AES-256 is already considered secure against post-quantum attacks [29]. The AES core includes the key schedule that provides the round-keys from the input key.
The implemented architecture of the AES core is based on the one presented in [12]: this architecture uses a merged S-box approach [30], to perform both encryption and decryption by executing one round of AES per clock cycle.

B. ECB, CBC, OFB, CFB, CTR modes of operations

The AES engine supports the basic AES modes of operations (i.e. ECB, CBC, OFB, CFB, and CTR algorithms) described in National Institute of Standards and Technology NIST special publication 800-38A [31]. To reduce the logic resources consumption, the proposed architecture shares the AES core and performs reshaping of data flow according to the mode of operation, employing multiplexers, XOR gates, and registers. Figure 5 shows data flow in case of encryption process, while Figure 6 shows data flow in case of decryption process. Referring to Figure 5 and Figure 6, *IV* corresponds to the Initialization Vector foreseen by the various AES modes, and *D_in* and *D_out* are, respectively, the input data and the output data; thus, in case of encryption (Figure 5) *D_in* port is fed with a data block of plaintext, and *D_out* corresponds to a ciphertext block, viceversa in case of decryption.

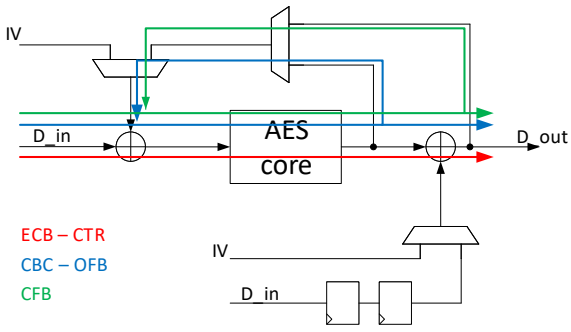

Fig. 5. ECB, CBC, OFB, CFB and CTR Modes With a Single AES Core Architecture: Encryption Data Flow.

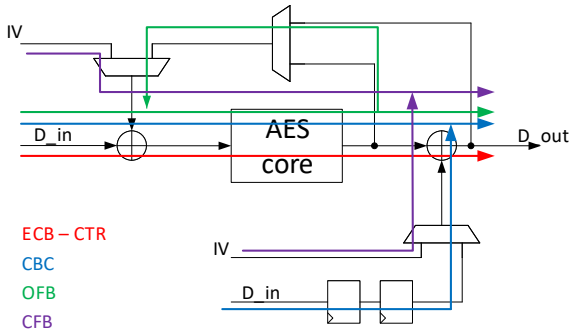

Fig. 6. ECB, CBC, OFB, CFB and CTR Modes With a Single AES Core Architecture: Decryption Data Flow.

C. CMAC core

The Cipher-based MAC (CMAC) mode described in the NIST special publication 800-38B [32] is supported by the AES engine. Figure 7 shows the high-level block scheme of the logic resources employed by the AES engine when supporting the CMAC mode: other than the AES core in CBC (encryption) mode, an additional module for derivation of sub-keys *K1* and *K2* is instantiated. In this case, after the processing of the sequence of all the M_i blocks composing the message *M* a unique output data block is produced, which is the tag of the message, *T*.

On the receiver side, the recipient can easily verify the integrity and authenticity of the message by locally recomputing the message tag and comparing it with the received one.

D. CCM and GCM cores

The AES engine supports CCM and GCM modes, described respectively in the NIST special publications 800-38C [33] and 800-38D [34]. They are AES modes of operations providing confidentiality, integrity and authenticity services. For this reason, the processes they perform are named *encryption-generation* and *decryption-verification* (being the terms *encryption/decryption* referring to plaintext/ciphertext and *generation/verification* to the tag produced for integrity check).

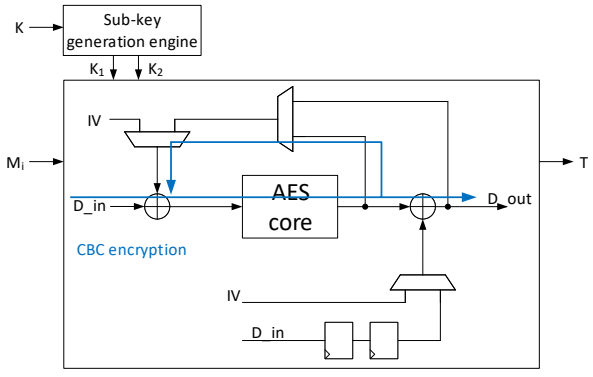

Fig. 7. Logic Resources of AES Engine When Supporting CMAC Mode.

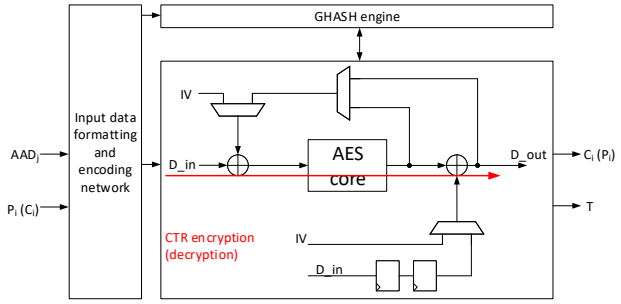

Fig. 8. GCM Core.

The two modes are equivalent in terms of security services offered, the only difference consists in the methodology for the generation of the tag. CCM mode employs the MAC-then-Encrypt (MtE) paradigm, i.e. the tag is computed before converting the plaintext into ciphertext, and this implies that on the recipient side, the verification of tag can be performed only once the whole ciphertext has been completely decrypted; the GCM employs the Encrypt-then-MAC (EtM) paradigm, that calculates the integrity tag using the ciphertext: in this case, the recipient can verify the tag without decrypting the ciphertext, allowing to save resources dedicated to the decryption in case the tag verification fails.

Both cores embeds the module described in section III-B and a module dedicated to the encoding and the formatting of input data, i.e. associated data, that are not encrypted and usually termed as Additional Authenticated Data (AAD), and the plaintext (or ciphertext, in case of *decryption-verification*). For encryption/decryption, both the cores exploit the CTR mode of AES, while for tag generation the CCM core employs the CBC mode (like the CMAC algorithm); instead, the GCM core requires a multiply-and-accumulate operation using a multiplication over the Galois field $GF(2^{128})$ and using as modulo the polynomial $M(x) = x^{128} + x^7 + x^2 + x + 1$: such operation is typically termed GHASH (Galois-HASH) and for this reason, the GCM core integrates also a module dedicated to it.

Figure 8 illustrates the GCM core, for which AAD_j is the j^{th} data block of AAD, P_i (or C_i) is the i^{th} block of plaintext (or ciphertext) and T is the integrity tag.

E. XTS core

AES XTS described in NIST special publication 800-38E [35] is a mode of operation aimed at encryption of sector-based storage, guaranteeing random access to encrypted data. To perform this operation two different keys are required and an initialization vector that must be updated for each new sector to be encrypted: the initialization vector is computed using arithmetic operation over the Galois field defined by the polynomial $M(x) = x^{128} + x^7 + x^2 + x + 1$. Thus, the XTS core embeds the ECB mode logic described in section III-B, a module dedicated to the computation of sector-based initialization vector and a block dedicated to formatting of input and output data.

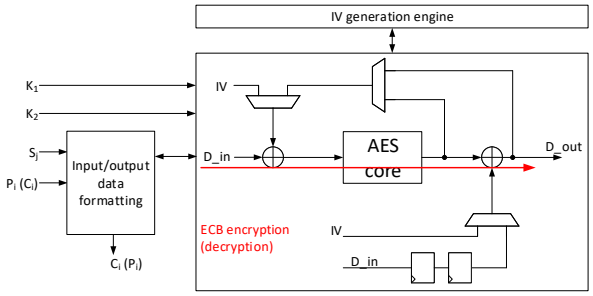

Fig. 9. XTS Core.

Figure 9 shows the XTS core, for which S_j is the j^{th} sector number, P_i (or C_i) is the i^{th} data block to be encrypted (or ciphertext) and K_1 and K_2 are the keys.

IV. Cryptoprocessor technology validation on FPGA-based SoC prototype

To validate the AES cryptoprocessor system emulating the EPI cryptoprocessor behaviour, we designed an FPGA prototype system. The designed demo application employs the CVA6 RISC-V core [36] and exploits an AXI4 Interconnect structure for securely performing AES cryptographic operations. Using the UART module, the application provides to the user a command-line interface, with which he can perform encryption and decryption operations of strings.

A. Unipi Cryptoprocessor Demoboard

The chosen board, ZCU106 from Xilinx, features a Zynq UltraScale+ MPSoC EV device and supports all major peripherals and interfaces, enabling development for a wide range of applications.

In Figure 10 a simplified block diagram of the prototyped system is shown. The interconnection of the demo board is composed of several AXI peripheral masters and slaves due to the presence of the DMA. For this reason there are three different AXI Crossbars that connect the main components of the Block Design, reported in the following together with the other building blocks:

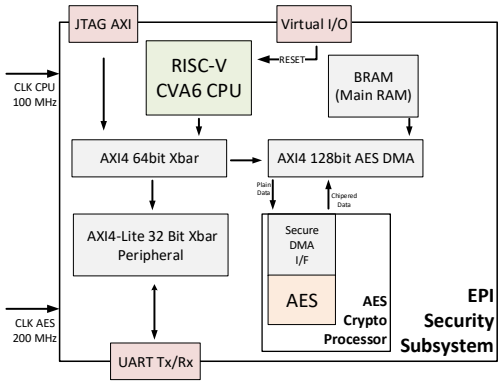

Fig. 10. Architecture of the Demonstrative System On Chip.

- *RISC-V CVA6 CPU*: it has a master AXI4 interface that is connected to the main memory Crossbar and the peripheral Crossbar. It can access all the available resources in the design. To save resources and power we did not include the Floating Point Unit (FPU).
- *UART*: provided by Xilinx; it has a slave AXI4-Lite interface which is connected to the peripheral Crossbar, from which it can be configured and used.
- *AES Cryptoprocessor*: it has a slave AXI4-Lite interface connected to the peripheral Crossbar from which it can be configured and a couple of AXI Stream interfaces connected to the AES DMA with which the data can be sent and the results can be taken.
- *AXI4 128bit AES DMA*: provided by Xilinx; in addition to the AXI Stream interfaces connected to the AES peripheral, it has a slave AXI4-Lite interface connected to the peripheral Crossbar for configuration and a master AXI4 interface connected to the main memory Crossbar for accessing data.
- *Block RAMs (BRAM)*: provided by Xilinx; are on-chip memories implemented with Block RAMs and are used to store the program binary and data. Its size is 128 KB.
- *AXI4 64Bit Xbar SmartConnect*: provided by Xilinx; connect the masters with the peripherals and the memory using *speed* optimization strategy for CPU and memory interconnects and *area* optimization strategy for the peripherals interconnect.
- *AXI4-Lite 32Bit Xbar Peripheral*: provided by Xilinx; bridges the AXI4-Lite peripherals to the main AXI interconnect.
- *JTAG AXI*: provided by Xilinx; it is used to load the program binary into the main memory at run time.
- *Virtual I/O*: provided by Xilinx; it is used to assert the reset signal to the SoC from the *Hardware Manager* in Vivado.

We opted for a loosely coupled accelerator approach to maximise system performance, at the price of a more complex

Clock Domain	WNS	WHS
main (100 MHz)	0.067 ns	0.011 ns
AES Engine (200 MHz)	0.160 ns	0.019 ns

TABLE II
Timing Report for the Two Clock Domains.

Module	LUT	FF	BRAM
SoC	71'947 (31.23%)	60'040 (13.03%)	77.5 (24.84%)
CVA6	32'764 (45.54%)	19'073 (33.77%)	37 (47.74%)
AES CP	15'694 (21.81%)	9'925 (16.53%)	0
AXI Bus	17'680 (24.57%)	10'154 (33.57%)	0
DMA	3'485 (4.84%)	6'967 (11.60%)	5 (6.45%)
RAM	164 (0.23%)	1 (0%)	32 (41.29%)
JTAG AXI	783 (1.09%)	2'021 (3.37%)	3.5 (4.52%)

TABLE III
Absolute and Relative Resource utilization on the ZCU106 board, XCZU7EV FPGA.

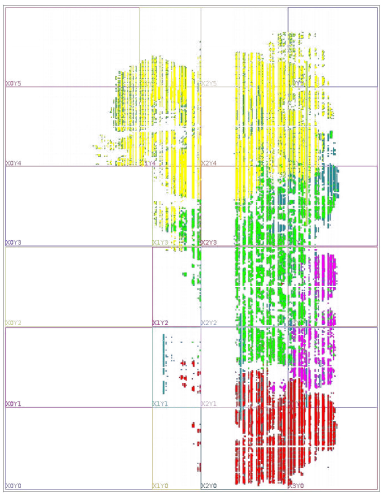

Fig. 11. Visual Resource Placement on the FPGA of CVA6 no FPU (*yellow*), AES Cryptoprocessor (*red*), AXI SmartConnect (*green*) and DMAs (*purple*).

hardware accelerator. Indeed, a tightly coupled accelerator requires a negligible amount of resources but has several disadvantages: even if it does not have any AXI communication delay, it has a memory communication overhead, especially heavy in case of large encryptions. Our solution to the problem, optimised for larger blocks encryption, is to mitigate the delay by exploiting an independent DMA to deliver plain data to the engine and collect the encrypted messages. A second problem is that it would be problematic to enable hardware acceleration of all the supported AES operational modes inside the processor pipeline, and the overall clock frequency would be heavily affected by the resulted complexity. This is observable in work such as [22], where the system is less complex than our proposed system (about 6.5kGE against 131.45kGE), but the performances are much lower (about 6 Mbps against 30 Gbps), resulting in the worst throughput on complexity ratio.

Please note that the blocks performing the conversion of the AXI protocol and the data width are omitted in Figure 10.

The demo application takes the input from the user using the UART interface. Then it configures the AES DMA to gather the data, that must be encrypted or decrypted, from the memory and send it to the AES peripheral which performs the cryptographic operation. The data is processed and the result is scattered back in the main memory. After that, the application prints the result on the terminal using the UART.

B. Implementation Results

The presented SoC has been synthesised and implemented on the ZCU106 board. The timing requirements are met with the margins listed in Table II; please note that within the AES cryptoprocessor, the interface registers (i.e. Configuration, Control, Error, and Status registers), the state machine, and the data and key registers are clocked with the *main* clock. While the resource utilization is presented in Table III, both with the absolute values and the percentage of resources occupied on the FPGA. The resulting placement is illustrated in Figure 11.

Module	**Gate [kGE]**	**Power [mW]**
Cryptoprocessor support (Key regs, FSM, ...)	73.23	32.14
Engine support (ECB, CBC, ...)	36.49	10.90
AES core	21.73	6.46
Tot AES Cryptoprocessor	**131.45**	**49.50**

TABLE IV
7NM STANDARD-CELL TECHNOLOGY SYNTHESIS AT 2.55 GHZ.

V. VLSI TECHNOLOGY MAPPING

This section presents the synthesis and implementation results obtained in different platforms and technologies.

A. Synthesis on 7nm Standard-Cell technology

The proposed AES cryptoprocessor has been synthesised with Design Compiler by Synopsys on the Artisan 7nm TSMC Standard-Cell technology in the typical case (0.75V 85 °C). Table IV reports the synthesis results for the AES cryptoprocessor, the AES engine, and the AES core. Note that the AES engine includes the AES core plus the cores dedicated to the supported modes of operations (i.e. ECB, CBC, OFB, CFB, CTR, CMAC, CCM, GCM, and XTS), while the AES cryptoprocessor includes the AES engine plus the interface registers (i.e. Configuration, Control, Error and Status registers), the state machine, the data, and key registers, all synthesised at 2.55 GHz as reported in Table IV. Also, the AES cryptoprocessor includes a dedicated synchronization logic to manage clock domain crossing. In Table V we present the system throughputs for each operating mode. The measurements have been carried out exploiting the system presented in [41], where an emulation environment capable of performing a low-level post-synthesis simulation of code execution on a RISC-V based system is carefully described. The test has been performed simulating the entire system, loading the memory with bare-metal code, aiming at measuring the maximum capabilities of the hardware system.

B. Comparison to the State of the Art

To the best of our knowledge, no contribution related to hardware accelerators for AES implemented in 7nm technology can be found in the literature. In addition, few implementations can be directly compared to the proposed AES

	Throughput [Gbps]			
	Encryption		**Decryption**	
	128	**256**	**128**	**256**
EBC	32.64	23.31	15.54	11.26
CBC	29.45	21.76	15.54	11.26
CFB	29.45	21.76	32.64	23.31
OFB	29.45	21.76	29.45	21.76
CTR, CMAC, CCM, GCM, XTS	32.64	23.31	32.64	23.31

TABLE V
7NM STANDARD-CELL TECHNOLOGY SYNTHESIS AT 2.55 GHZ.

	This work	[37]	[38]	[39]	[40]
Key length [Min-Max]	128-256	128-256	128-256	128-256	128
Access Control Mechanism	✓	✗	✗	✗	✗
Keys storage and management	✓	✓	✗	✗	✗
Clock Randomization	✓	✗	✗	✗	✗
DMA Interface	✓	✓	✗	✗	✗
ECB, CBC Modes	✓	✓	✓	✓	✓
OFB, CFB, CTR Modes	✓	✗	only CTR	✗	✓
CCM, GCM Modes	✓	✓	only GCM	✗	✗
CMAC Mode	✓	✓	✗	✗	✗
XTS Mode	✓	✗	✗	✗	✗
Technology	7nm	–	–	45nm	45nm
Area [kGE]	131.45	49-53	–	187.67	7925.31
Throughput [Gbps]	29.45 (CBC)	–	–	16.74 (CBC)	6.40 (CBC)

TABLE VI
COMPARISON AMONG DIFFERENT AES CRYPTOPROCESSOR IMPLEMENTATIONS.

cryptoprocessor due to the major differences that can be found in the overall architecture of the cryptoprocessor (e.g. interface registers, communication interfaces, supported modes of operations, key storage, key length, etc). Highly optimised implementations are available but unfair to be compared since they focus just on one of the figures of merit (e.g. area [42], [43]). Also, new emerging research topics tend to involve AES-like processing, such as the use of memristors to empower side-channel attack resistance [44]. Table VI reports the results in terms of area, throughput, and supported functions of different AES crypto-processors which have been extracted in both research papers and commercial products. Since the most relevant work in the literature provided significant results only in CBC mode, we decide to carry out the comparison based on that operational mode. The works in [37] and [38] are both commercial AES cryptographic IPs, while the works in [39] and [40] are research papers. The throughput refers to the 128-bits CBC encryption mode as it was the one with more reports in the referenced works. The CRYPT-IP-120b [37] is produced by Rambus and supports different AES modes of operations (i.e. ECB, CBC, CTR, CMAC, CCM, and GCM). It includes key storage and management service and a DMA controller. The area consumption is reported around 49 to 53 kGE while the throughput cannot be extracted. The IP in [38] is designed by EnSilica and is all-in-one encryption, decryption, key expansion, and chaining modes ECB, CBC, CTR, and GCM. It can support three different key sizes (i.e. 128, 192, and 256) but does not have any support for secure storage and DMA. No performance in terms of throughput and area is reported by the company. In [40] a flexible cryptographic processor named Cryptoraptor is described. It supports a wide range of symmetric key algorithms in addition to the AES and showed good performance in terms of throughput for the AES algorithm. Concerning our work, it offers more flexibility but no dedicated features for security. The work in [39] is an AES hardware accelerator fabricated in 45nm CMOS, for content protection in a high-performance microprocessor. It supports three key lengths and only ECB and CBC modes of operations. Although it is hard to perform a full comparison among these implementations due to the architectural differences they have, we can claim that our proposed solution is the more complete on the hardware side, supporting all the available modes of operation at the same time, with complexity and throughputs overcoming less-featured cryptoprocessors.

VI. CONCLUSIONS

This work presented an AES-based cryptoprocessor implementation, suitable to be employed both in embedded SoC applications and already adopted by the upcoming EPI processor. The accelerator was implemented on both a Xilinx Ultrascale+ FPGA device and 7nm standard-cell technologies. In particular, the Cryptoprocessor has been embedded with a RISC-V CVA6 processor and the proposed SoC has been prototyped on a ZCU106 board. Complexity and performance have been thoroughly characterised for all the AES modes of operations in all the above-mentioned technologies. The achieved results show that our implementation offers innovative hardware support to advanced cryptoprocessor features such as access control, IP configuration, key management, and clock randomization mechanisms. On top of that, we extensively support up to 9 modes of operations, contemporaneously on the same hardware accelerator, obtaining results that are aligned with the state-of-the-art solution, where only a few modes of operations were included and no advanced features were supported. To the best of the authors' knowledge, this work is the first contribution available in the literature on AES implementation results on 7nm technology.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826647 (European Processor Initiative).

REFERENCES

[1] F. Rahman, M. Farmani, M. Tehranipoor, and Y. Jin, "Hardware-Assisted Cybersecurity for IoT Devices," *IEEE 18th International Workshop on Microprocessor and SOC Test and Verification*, 2017.
[2] E. Consortium, "EPI website." [Online]. Available: https://www.european-processor-initiative.eu/
[3] L. Baldanzi, L. Crocetti, S. DI MATTEO, L. Fanucci, S. Saponara, and H. Patrice, "Crypto accelerators for power-efficient and realtime on-chip implementation of secure algorithms," in *IEEE ICECS 2019*. IEEE, 2019, pp. 1–4.

[4] P. Nannipieri, M. Bertolucci, L. Baldanzi, L. Crocetti, S. Di Matteo, F. Falaschi, L. Fanucci, and S. Saponara, "Sha2 and sha-3 accelerator design in a 7 nm technology within the european processor initiative," *Microprocessors and Microsystems*, p. 103444, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933120305986

[5] S. Di Matteo, L. Baldanzi, L. Crocetti, P. Nannipieri, L. Fanucci, and S. Saponara, "Secure elliptic curve crypto-processor for real-time iot applications," *Energies*, vol. 14, no. 15, p. 4676, 2021.

[6] P. Nannipieri, S. Di Matteo, L. Baldanzi, L. Crocetti, J. Belli, L. Fanucci, and S. Saponara, "True random number generator based on fibonacci-galois ring oscillators for fpga," *Applied Sciences*, vol. 11, no. 8, 2021.

[7] L. Baldanzi, L. Crocetti, F. Falaschi, M. Bertolucci, J. Belli, L. Fanucci, and S. Saponara, "Cryptographically secure pseudo-random number generator ip-core based on sha2 algorithm," *Sensors*, vol. 20, no. 7, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/7/1869

[8] NIST, "FIPS 197: Advanced Encryption Standard (AES)," *Federal Information Processing Standards Publication*, vol. 197, no. 441, p. 0311, 2001.

[9] L. Baldanzi, L. Crocetti, S. Di Matteo, L. Fanucci, S. Saponara, and P. Hameau, "Crypto accelerators for power-efficient and real-time on-chip implementation of secure algorithms," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2019, pp. 775–778.

[10] D.-e.-S. Kundi, A. Khalid, A. Aziz, C. Wang, M. O'Neill, and W. Liu, "Resource-shared crypto-coprocessor of aes enc/dec with sha-3," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4869–4882, 2020.

[11] S. Sawataishi, R. Ueno, and N. Homma, "Unified hardware for high-throughput aes-based authenticated encryptions," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 9, pp. 1604–1608, 2020.

[12] R. Ueno, S. Morioka, N. Homma, and T. Aoki, "A High Throughput/Gate AES Hardware Architecture by Compressing Encryption and Decryption Datapaths - Toward Efficient CBC-Mode Implementation," *Cryptology ePrint Archive, Report 2016/595*, 2016.

[13] J. Resende and R. Chaves, "Compact dual block AES core on FPGA for CCM protocol," *25th International Conference on Field Programmable Logic and Applications (FPL), IEEE*, 2015.

[14] X. Tao, D. Zhang, and Y. Song, "An implementation of configurable and small-area aes ip core oriented avalon bus," *2015 International Conference on Artificial Intelligence and Industrial Engineering*, 2015.

[15] "Get started with k210: Hardware and programming environment," https://www.seeedstudio.com/blog/2019/09/12/get-started-with-k210-hardware-and-programming-environment/, accessed: 2021-10-26.

[16] V. B. Y. Kumar, A. Chattopadhyay, J. Haj-Yahya, and A. Mendelson, "Itus: A secure risc-v system-on-chip," in *2019 32nd IEEE International System-on-Chip Conference (SOCC)*, 2019, pp. 418–423.

[17] "Sifive shield: An open, scalable platform architecture for security," https://www.sifive.com/blog/sifive-shield-an-open-scalable-platform-architecture, accessed: 2021-10-26.

[18] "Using risc-v as a security processor for darpa chips and commercial iot," https://riscv.org/wp-content/uploads/2017/12/Wed-1430-RISCV-MarkBealV2.pdf, accessed: 2021-10-26.

[19] W. Wang, J. Han, X. Cheng, and X. Zeng, "An energy-efficient crypto-extension design for risc-v," *Microelectronics Journal*, vol. 115, p. 105165, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026269221001749

[20] B. Marshall, G. R. Newell, D. Page, M.-J. O. Saarinen, and C. Wolf, "The design of scalar aes instruction set extensions for risc-v," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 1, p. 109–136, Dec. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8729

[21] K. Stoffelen, "Efficient cryptography on the risc-v architecture," in *Progress in Cryptology – LATINCRYPT 2019*, P. Schwabe and N. Thériault, Eds. Cham: Springer International Publishing, 2019, pp. 323–340.

[22] C. Duran, H. Gomez, and E. Roa, "Aes sbox acceleration schemes for low-cost socs," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[23] B. Hettwer, K. Das, S. Leger, S. Gehrer, and T. Güneysu, "Lightweight side-channel protection using dynamic clock randomization," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, 2020, pp. 200–207.

[24] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of aes," in *Advances in Cryptology – EUROCRYPT 2011*, K. G. Paterson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 69–88.

[25] A. G. Bayrak, N. Velickovic, F. Regazzoni, D. Novo, P. Brisk, and P. Ienne, "An eda-friendly protection scheme against side-channel attacks," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 410–415.

[26] A. A. El-Moursy, A. M. Darya, A. S. Elwakil, A. Jha, and S. Majzoub, "Chaotic clock driven cryptographic chip: Towards a dpa resistant aes processor," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2020.

[27] R. Menicocci, A. Trifiletti, and F. Trotta, "Experiments on two clock countermeasures against power analysis attacks," in *2014 Proceedings of the 21st International Conference Mixed Design of Integrated Circuits and Systems (MIXDES)*, 2014, pp. 215–219.

[28] NIST, "FIPS 140-2: Security Requirements for Cryptographic Modules," *Federal Information Processing Standards Publication*, 2001.

[29] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The Impact of Quantum Computing on Present Cryptography," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, 2018.

[30] R. Ueno, N. Homma, Y. Sugawara, Y. Nogami, and T. Aoki, "Highly Efficient $GF(2^8)$ Inversion Circuit Based on Redundant GF Arithmetic and Its Application to AES Design," *IACR Cryptology ePrint Archive*, vol. 2015, p. 763, 2015.

[31] Morris Dworkin, "NIST Special Publication 800-38A," National Institute of Standards and Technology, Tech. Rep., 2001.

[32] ——, "NIST Special Publication 800-38B," National Institute of Standards and Technology, Tech. Rep., 2005.

[33] ——, "NIST Special Publication 800-38C," National Institute of Standards and Technology, Tech. Rep., 2004.

[34] ——, "NIST Special Publication 800-38D," National Institute of Standards and Technology, Tech. Rep., 2007.

[35] ——, "NIST Special Publication 800-38E," National Institute of Standards and Technology, Tech. Rep., 2008.

[36] F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2629–2640, 11 2019.

[37] "Crypt-ip-120 aes crypto, rambus," https://www.rambus.com/security/crypto-accelerator-hardware-cores/basic-crypto-blocks/crypt-ip-120/, accessed: 2021-06-04.

[38] "Aes cryptographic ip, ensilica," https://www.ensilica.com/ip/esi-crypto/aes/, accessed: 2021-06-04.

[39] S. K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S. K. Hsu, H. Kaul, M. A. Anders, and R. K. Krishnamurthy, "53 gbps native $gf(2^4)^2$ composite-field aes-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 767–776, 2011.

[40] G. Sayilar and D. Chiou, "Cryptoraptor: High throughput reconfigurable cryptographic processor," *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 155–161, 2014.

[41] L. Zulberti, P. Nannipieri, and L. Fanucci, "A script-based cycle-true verification framework to speed-up hardware and software co-design of system-on-chip exploiting risc-v architecture," in *2021 16th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2021, pp. 1–6.

[42] K. Shahbazi and S.-B. Ko, "Area-efficient nano-aes implementation for internet-of-things devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 136–148, 2021.

[43] S. N. Dhanuskodi, S. Allen, and D. E. Holcomb, "Efficient register renaming architectures for 8-bit aes datapath at 0.55 pj/bit in 16-nm finfet," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 8, pp. 1807–1820, 2020.

[44] M. Masoumi, "Novel hybrid cmos/memristor implementation of the aes algorithm robust against differential power analysis attack," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 7, pp. 1314–1318, 2020.