Araştırma Makalesi / Research Paper
Bayburt Üniversitesi Fen Bilimleri Dergisi
Bayburt University Journal of Science

dergipark.gov.tr/bufbd
ISSN: 2667-579X

Cilt 7, Sayı 2, Yıl 2024, Sayfa 123-132
Vol 7, Issue 2, Year 2024, Pages 123-132
doi.org/10.55117/bufbd.1582809

Geliş Tarihi / Date of Recieve: 11.11.2024
Revizyon Tarihi / Date of Revision: 1.12.2024
Kabul Tarihi / Date of Acceptance: 09.12.2024

Custom TTA Operations for Accelerating the Ascon Encryption Algorithm

Latif AKÇAY[1*]

[1]*Bayburt University, Electrical - Electronics Engineering, Bayburt, Türkiye*

Keywords:

Lightweight cryptography, Instruction set extension, Custom processor design, Application-specific processor design, FPGA implementation

Abstract

Lightweight cryptography is becoming increasingly important in modern applications, especially in resource-constrained environments such as Internet of Things (IoT) devices, embedded systems and mobile platforms. The Ascon encryption algorithm is a modern, secure and efficient cryptographic scheme that meets the demands of low-power devices. However, some steps of the algorithm are computationally intensive, leading to performance issues. In this study, custom operations are proposed to accelerate the Ascon encryption algorithm on Transport-Triggered Architecture (TTA) processors. In order to make more efficient use of hardware resources, the custom operations are designed to have low complexity and high efficiency. The OpenASIP tool was employed to integrate the operations into a general purpose 64-bit TTA processor. The resulting application-specific core was implemented in Hardware Description Language (HDL) and synthesised for FPGA. We analyse the performance gain for different transport bus configurations. The results obtained show that the Ascon-AEAD128 encryption and decryption phases are accelerated by 38% to 50%. When evaluated together with the synthesis results, a significant performance gain was achieved with a very reasonable increase in hardware resources. The study also emphasises that the TTA is a suitable method for accelerating cryptographic applications that require low power consumption and high efficiency.

[*]e-mail: lakcay@bayburt.edu.tr

Bu makaleye atıf yapmak için:
Latif AKÇAY, "Custom TTA Operations for Accelerating the Ascon Encryption Algorithm", Bayburt Üniversitesi Fen Bilimleri Dergisi, C. 7, s 2, ss. 123-132

How to cite this article:
Latif AKÇAY, "Custom TTA Operations for Accelerating the Ascon Encryption Algorithm", Bayburt University Journal of Science, vol. 7, no 2, pp. 123-132

1 INTRODUCTION

In today's digital world, efficient and secure communication protocols are more important than ever, particularly in the context of resource-constrained devices such as embedded systems, IoT devices, and low-power sensors. These devices are typically constrained in computational power, memory, and energy resources. Consequently, there has been a notable increase in research and development efforts focusing on the creation of cryptographic algorithms designed specifically for these kinds of environments. This field of research is commonly referred to as "lightweight cryptography” [1]. The objective of lightweight cryptography is to provide robust security assurances while reducing the computational load and power consumption typically imposed by traditional cryptographic algorithms. It is essential to achieve a balance between efficiency and security, in order to ensure the safe operation of devices in a wide range of applications, including secure communication, authentication, and sensitive data protection.

To address the increasing demand for secure and efficient cryptographic solutions, the National Institute of Standards and Technology (NIST) launched the Lightweight Cryptography competition in 2018 [2]. The objective was to identify cryptographic algorithms that are well-suited for constrained environments. In accordance with the competition conditions, NIST initiated the process with 56 candidate algorithms. Following a three-round evaluation and elimination process, NIST announced the Ascon family as the winner in 2023 [3]. Ascon was designed as a fast, secure and lightweight algorithm that makes it an ideal solution for resource-limited environments. Its algorithmic structure, based on authenticated encryption with associated data (AEAD) and hash functions, provides robust security while maintaining high throughput [4]. However, while Ascon demonstrates excellent results in terms of security and performance, certain aspects of its implementation, especially those requiring numerous bitwise operations, permutations, and large integer multiplications, pose significant challenges for general-purpose processors designed for embedded systems.

Application-specific instruction set processors (ASIPs) are designed to efficiently execute particular applications or application domains [5]. To overcome the limitations of general-purpose processors, ASIPs such as Transport-triggered Architectures (TTAs) can provide an effective solution [6]. TTA is a highly flexible and efficient core design method that enables designer to map complex computational tasks to hardware operations in a way that traditional processors cannot achieve. In a TTA processor, instructions are encoded as data transport operations. The architecture's flexibility allows the design of custom units for the high-performance and low energy execution of cryptographic algorithms, such as those required by Ascon.

The objective of this paper is to investigate the design and implementation of custom TTA operations to accelerate the performance of the Ascon encryption algorithm. The designed operations are focused on accelerating the linear and substitution layers of the algorithm for improving the overall performance. The custom operations were implemented in HDL and the entire ASIP synthesized for an FPGA platform. The performance analyses were conducted for various bus configurations. By leveraging the inherent parallelism and low-latency characteristics of TTA-based hardware, we demonstrate how these custom operations can significantly enhance Ascon's throughput and energy efficiency performance, particularly for use cases in embedded and IoT environments where power and resource constraints are critical. The rest of the paper is organized as follows: This section provides a concise overview of related studies in the literature. The second section introduces the Ascon family and explains the tools found in the TTA and unified processor design environment. The third section presents a detailed analysis of the developed operations and ASIP design and implementation phases. The fourth section offers a comprehensive summary of the study and suggests avenues for future research.

1.1 Literature review

A number of comprehensive studies have been carried out on the implementation of cryptography algorithms on TTA processorrs. In [7], the authors designed a TTA-based ASIPs for the RC4 and Advanced Encryption Standard (AES) encryption algorithms. They also implemented their processor design as an Application-Specific Integrated Circuit (ASIC) and shared the resource utilization and throughput in detail. A similar study was conducted for the Triple Data Encyrption Standard (3DES), RC4, and AES in [8]. The authors used a design space exploration to automatically find the most proper configurations for different requirements. In another study, the authors proposed an efficient method for design space exploration and developed TTA cores for the Blowfish and Secure Hash Algorithm (SHA) [9]. Similar processor designs also developed in [10, 11].

In addition to symmetri- key encryption algorithms, public-key cryptography algorithms have been evaluated on TTA processors. A parallel architecture for efficient hardware implementation of Rivest Shamir Adleman (RSA) cryptography is proposed in [12, 13]. Similarly, a unified crypto-processor with coarse-grained reconfigurable datapath to perform either RSA or elliptic curve cryptosystems (ECC) is also proposed in [14]. In the majority of RSA-based applications, the most significant factor limiting the performance of the encryption process is the key generation speed. In [15], a Residue Number System (RNS) and custom TTA operations have been employed to significantly accelerate the generation of RSA keys. Another area of cryptography where TTA processors can be used efficiently is in the Post-quantum cryptography (PQC) methods that researchers have focused on in recent years. These PQC algorithms, which must be resistant to both classical and quantum computer attacks, require intensive computational resources. This situation poses a serious challenge to the development of PQC applications in embedded systems. Researchers have been able to achieve remarkable performance gains and low energy consumption levels by designing custom TTA operations for the Key Encapsulation Mechanisms (KEMs) [16-19]. On the other hand, apart from the present study, no other research in the literature has yet employed the TTA architecture to accelerate the Ascon encryption algorithms.

2 MATERIALS AND METHODS

This section briefly introduces the encryption algorithm targeted to be accelerated in the study, the preferred ASIP architecture, and the integrated design environment. In addition, the design of custom operations and their integration into the template architecture are explained.

2.1 Ascon encryption algorithm

Ascon is a family of encryption algorithms that includes authenticated ciphers (Ascon-AEAD128, Ascon-AEAD128a), cryptographic hash (Ascon-Hash256), and extendable output functions (Ascon-XOF128 and Ascon-CXOF128.). It is designed to provide high security, efficiency and suitability for a wide range of cryptographic applications. It is primarily targeted at lightweight environments such as embedded systems and low power devices, but is also suitable for general purpose use. Ascon was submitted to the CAESAR (Competition for Authenticated Encryption: Security, Applicability and Robustness) competition and was selected as a finalist for its advanced efficiency and security features [20]. It was then selected as the winner of the NIST Lightweight Cryptography competition [2]. Ascon is a symmetric-key encryption algorithm that provides both encryption and authentication to protect against a variety of cryptographic attacks.

Ascon is a stream-based authenticated encryption algorithm that combines encryption and authentication. Ascon consists of three main variants: Ascon-AEAD128, Ascon-AEAD128a, and Ascon-80pq which differ in certain operational details but have a similar structure. The Ascon design is based on a sponge construction and uses a permutation-based core. The sponge construction absorbs the input data (key, plaintext, and associated data (AD)) and then squeezes out the output (ciphertext and authentication tag). Ascon uses a 128-bit state, divided into two main parts: one for the message and one for the AD. Fig. 1 represents the inner stages of the encryption and decryption operations of the cipher.

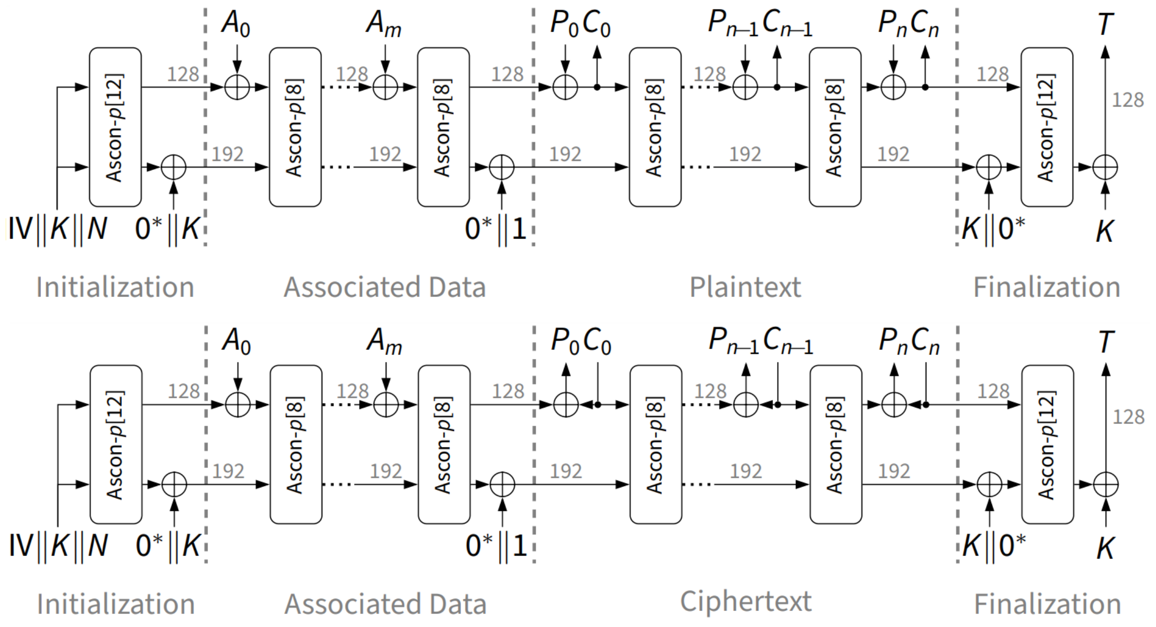


Figure 1. Ascon-AEAD128 encryption (top) and decryption (bottom) stages [4]

2.2 Transport-triggered architecture

TTA is an exposed datapath architecture, similar to the Very Long Instruction Word (VLIW) [21] methodology, that benefits from programmable datapath interconnect. Through the effective use of instruction-level parallelism (ILP), TTA aims to achieve high-performance program execution. Extensive use of register file (RF) bypass is one of the key benefits of TTA. During the programme flow, intermediate results are not transferred to the register file, but to the input of the functional unit (FU) where the next operation is located, unless necessary. In TTA processors, arithmetic or memory operations can be performed simultaneously after the fetch and decode phase of the long instruction word, a significant advantage over the traditional Reduced Instruction Set Computing (RISC) approach. A typical TTA processor consists of RF, load-store unit (LSU), global control unit (GCU), immediate unit (IMU), and other FUs for various arithmetic and logic operations. These units are vertically connected to the transport buses via input and output sockets. Fig. 2 depicts the structure of a TTA processor.

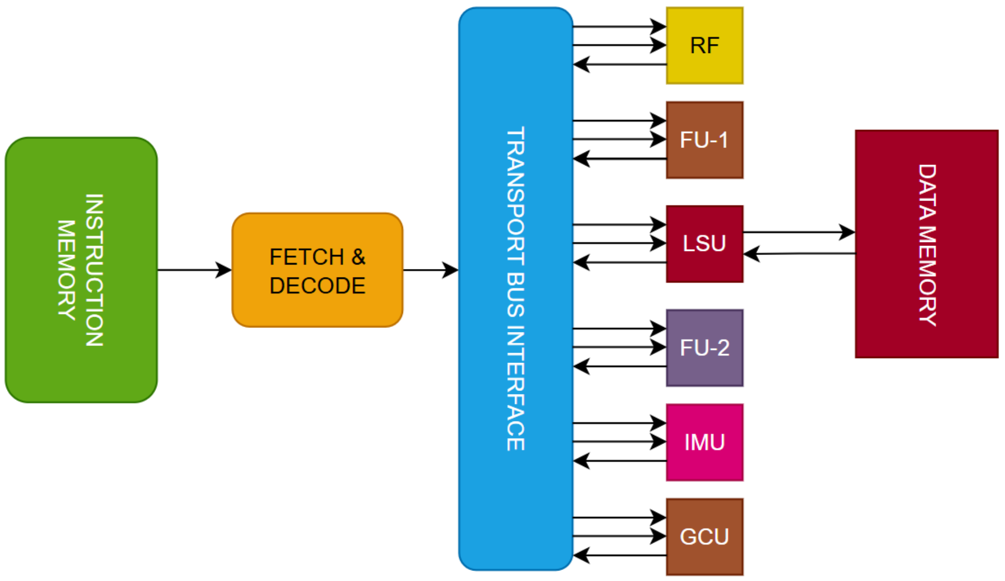


Figure 2. A typical TTA processor architecture

2.3 OpenASIP

OpenASIP, formerly know as TTA-based Co-Design Environment (TCE), is a free and fully open-source toolset developed for designing TTA-based custom ASIPs [22]. It consists of an LLVM-based compiler (TCECC), a processor model designer (ProDe), a cycle-accurate simulator (Proxim), and an RTL code generator (ProGe). Fig. 3 shows how a designer uses OpenASIP tools to develop an ASIP for a target application. A template processor format and high-level application code are developed step-by-step in design cycles involving simulation and profiling analysis. When a satisfactory design is achieved, RTL code is generated and then passed to third party synthesize tools.

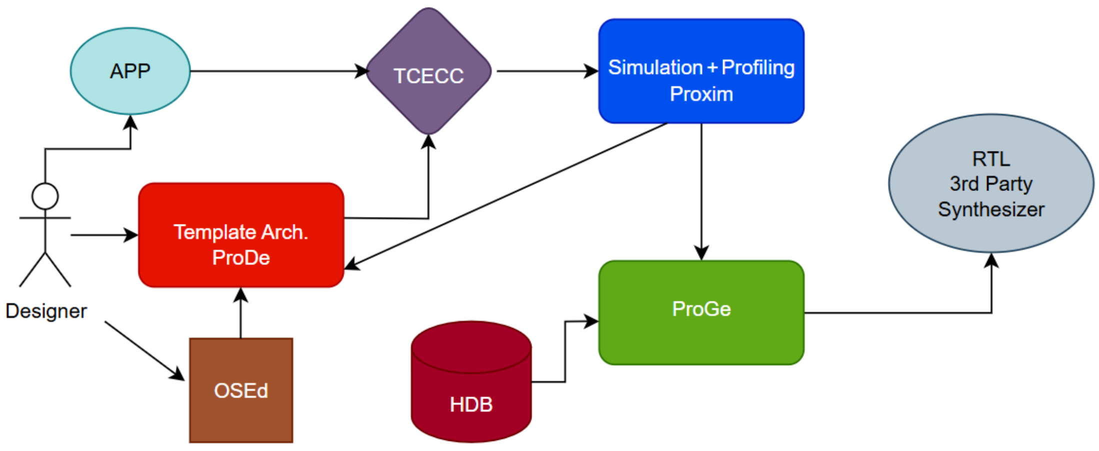


Figure 3. ASIP design flow with OpenASIP

OpenASIP contains many template processor architectures, pre-written HDL codes, and sample custom designs. This allows users to take advantage of a very rich development environment. They can use the Operation Set Editor (OSEd) to introduce their own application-specific instruction designs into the compiler and simulator. They can also add their own RTL code to the Hardware Database Editor (HDBEditor). This enables simulation, profiling and HDL code generation for the entire ASIP design.

2.4 Template architecture

A typical TTA-based ASIP design phase is initiated with a template architecture. Ascon performs operations using 64-bit values. Specifically, its internal state consists of five 64-bit words, making a total of 320 bits. The algorithm processes inputs in 64-bit blocks and applies a permutation on this 320-bit state, involving bitwise operations like XOR, rotations, and shifts. Therefore, a 64-bit general purpose TTA core is used in this study as a template which was designed during our previous studies [16-18]. The template processor is named as TTA64 and its ProDe model is given in Fig. 4. TTA64 contains general purpose operations *such as add*, *or*, *xor*, *shift*, *mul*, etc. It has a dual arithmetic logic FU (ALU64 and ALU64_1) where frequently repeated arithmetic operations are also put into a second smaller FU (ALU64_1). Apart from this, the processor has a simple structure consisting of RFs with 32x64-bit size registers and 2x1-bit Boolean, an FU for *printf* function, a classic LSU and GCU. The reference software implementation of the Ascon-128AEAD [23] was first run on TTA64 in this study.

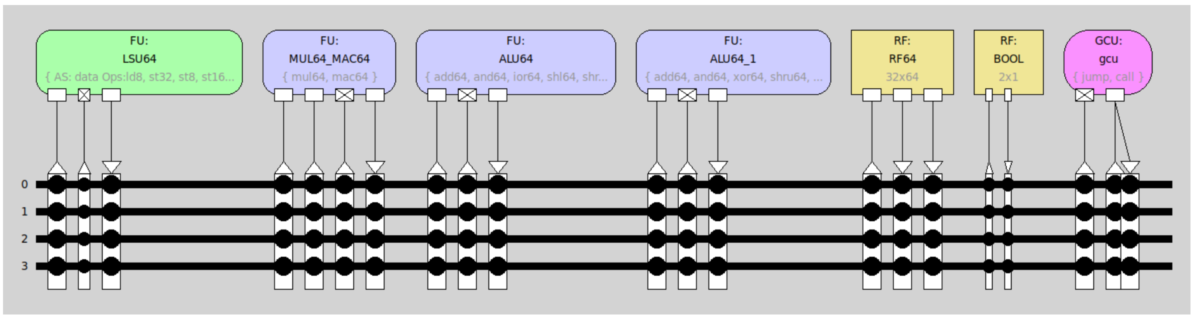
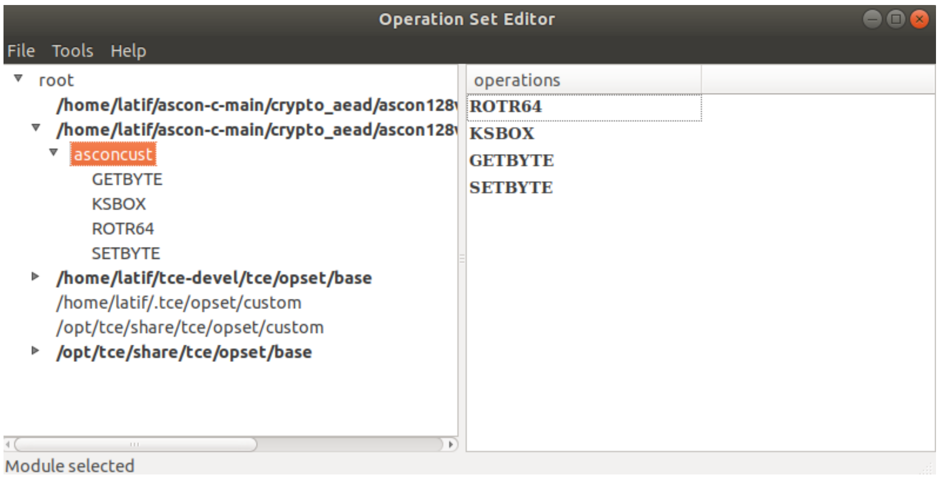

FU:
LSU64
{ AS: data Ops:ld8, st32, st8, st16...
FU:
MUL64_MAC64
{ mul64, mac64 }
FU:
ALU64
{ add64, and64, ior64, shl64, shr...
FU:
ALU64_1
{ add64, and64, xor64, shru64, ...
RF:
RF64
32x64
RF:
BOOL
2x1
GCU:
gcu
{ jump, call }
0
1
2
3

Figure 4. ProDe model of the template 64-bit TTA core (TTA64)

2.5 Design of custom operations

The most effective way to increase the performance of TTA processors is to design custom operations for the most compute-intensive parts of the application algorithm to be executed and integrate them into the processor architecture. To this end, the Ascon-AEAD128 reference C implementation was analysed using the Proxim tool and the code was examined in detail. As expected, the linear and substitution layers of the algorithm were identified as the most computationally intensive parts as a result of these analyses.

In OpenASIP environment, the operation set is managed by the OSEd tool. Thus, 4 custom operations were created using this tool. The OSEd tool was used for all processes, such as designing custom operations, determining the input/output port numbers, and defining the functions they require in a C++-like fashion. Fig. 5 shows the definition of the custom TTA operations developed for Ascon-AEAD128 in the OSEd interface.



```
▾ root
    /home/latif/ascon-c-main/crypto_aead/ascon128
  ▾ /home/latif/ascon-c-main/crypto_aead/ascon128
    ▾ asconcust
        GETBYTE
        KSBOX
        ROTR64
        SETBYTE
  ▸ /home/latif/tce-devel/tce/opset/base
    /home/latif/.tce/opset/custom
    /opt/tce/share/tce/opset/custom
  ▸ /opt/tce/share/tce/opset/base
```

operations
ROTR64
KSBOX
GETBYTE
SETBYTE

Module selected

Figure 5. Identification of the custom TTA operations in OSEd

In the process of designing a custom operation, two key considerations were identified. Firstly, the complexity of the operation should be kept to a minimum to prevent the maximum clock frequency from being reduced. Secondly, the number of operations should be kept to a reasonable level to avoid an excessive increase in the chip area. In consideration of the circumstances outlined above, the operations detailed in Table 1 were developed.

Table 1. Custom Ascon operations and their functionalities

Operation	Function	Related Code	Latency (cycle)
ROTR64	O = I1 >> I2 | I1 << (-I2 & 63)	Linear Layer	1
KSBOX	O = I1 ^ (~I2 & I3);	Substitution Layer	1
GETBYTE	O = (UI8) (I1 >> (56 - 8 * I2))	Load, Store, Clear bytes	1
SETBYTE	O = (UI64) (I1 << (56 - 8 * I2))	Load, Store, Clear bytes	1

2.6 ASIP design

ASIP design phase was initiated after the integration of custom operation to the OpenASIP compilation framework. As previously stated, Ascon-AEAD128 software was simulated on TTA64, and comprehensive profiling analyses were conducted. TTA is a flexible architecture and allows designer to add, remove, and chage the FUs. Leveraging this customization capability, the MUL64_MAC64 FU, which contains the *mul* and *multiply-and-add (mac)* operations that are seldom utilized when the Ascon algorithm operates on TTA64, was eliminated from the processor design and substituted with a new FU called ASCON, which contains the custom operations. The final ASIP, which is renamed as Ascon-TTA64, design is illustrated in Fig. 6. On the other hand, design alternatives that could provide higher performance, where custom operations are located in separate FUs, were not preferred because they would complicate the interconnection structure and thus increase the chip area and harden the routing during synthesis.

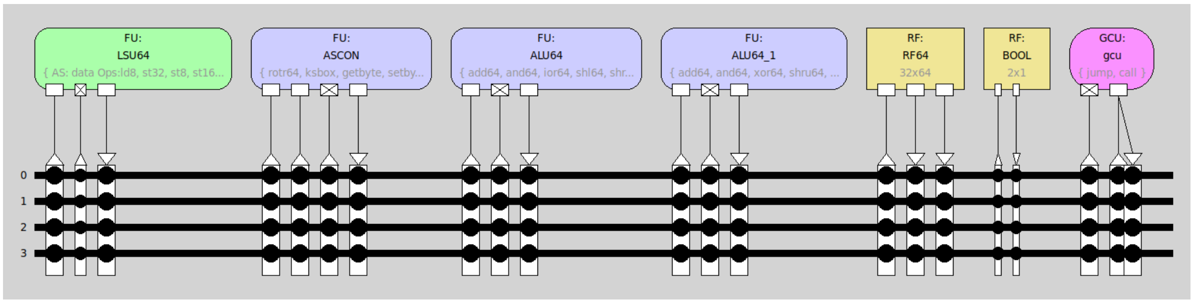

FU:
LSU64
{ AS: data Ops:ld8, st32, st8, st16...
FU:
ASCON
{ rotr64, ksbox, getbyte, setby...
FU:
ALU64
{ add64, and64, ior64, shl64, shr...
FU:
ALU64_1
{ add64, and64, xor64, shru64, ...
RF:
RF64
32x64
RF:
BOOL
2x1
GCU:
gcu
{ jump, call }
0
1
2
3

Figure 6. The ProDe model of the TTA64 (Ascon-TTA64) core after integrating the custom Ascon operations

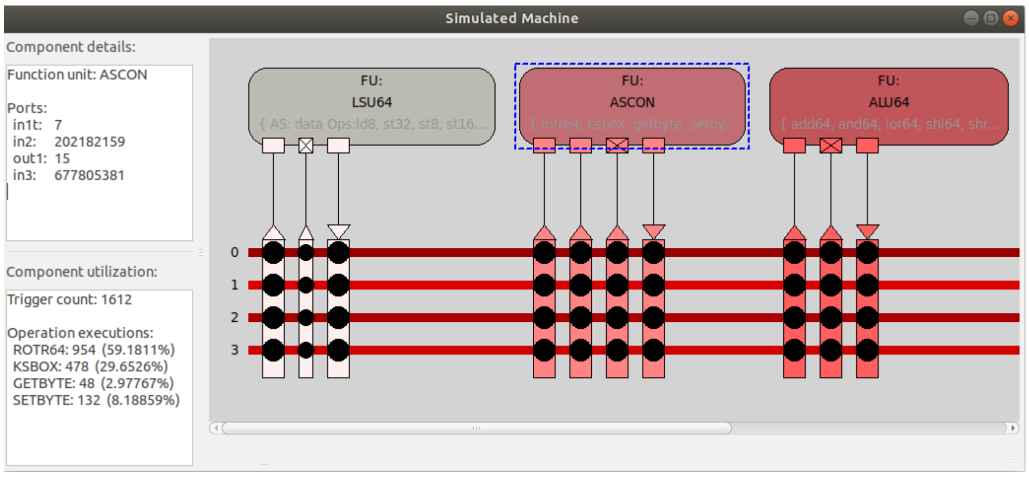
3 RESULTS AND DISCUSSION

This section presents the results of tests and performance analyses conducted with the reference Ascon-AEAD128 software on a range of configurations of the ASIP model and general-purpose TTA64. Furthermore, it offers an evaluation of the hardware resource utilizations of the designed processors following the HDL implementation and the synthesis process.

3.1 Cycle-accurate simulations

The OpenASIP toolkit employs the Proxim tool for simulation and profiling operations. Proxim is a cycle-accurate simulation tool that offers a range of practical and advanced features. Designers can utilise Proxim to run their applications directly on the ProDe model, thereby gaining insight into the operations, intermediate values, and memory operations performed on all units of the processor in a step-by-step manner.

In order to analyse the effects of the custom operations developed for the Ascon encryption algorithm, simulations were initially performed with the Ascon-128AEAD software for the TTA64 processor, which only includes general-purpose operations. The simulation was conducted again with alterations to the number of transport buses within the TTA64 architectural framework. Subsequently, similar operations were conducted on the Ascon-TTA64 processor, which had been adapted with custom operations, thus enabling an evaluation of its performance. Fig. 7 presents a visual representation of the analyses conducted in Proxim.


Component details:

Function unit: ASCON

Ports:
 in1t: 7
 in2: 202182159
 out1: 15
 in3: 677805381

Component utilization:

Trigger count: 1612

Operation executions:
 ROTR64: 954 (59.1811%)
 KSBOX: 478 (29.6526%)
 GETBYTE: 48 (2.97767%)
 SETBYTE: 132 (8.18859%)

Simulated Machine
FU:
LSU64
{ AS: data Ops:ld8, st32, st8, st16...
FU:
ASCON
{ rotr64, ksbox, getbyte, setby...
FU:
ALU64
{ add64, and64, ior64, shl64, shr...
0
1
2
3

Figure 7. Cycle-accurate simulation and profiling of the Ascon-TTA64 core using the Proxim tool

3.2 Performance Analysis

Performance analyses were conducted for both TTA64 and Ascon-TTA64, based on two distinct cases. Firstly, in order to analyse the impact of the transport bus number on performance, the Ascon-128AEAD encryption and decryption code was executed for all configurations with bus numbers ranging from 1 to 6. The same configurations were performed on the Ascon-TTA64 processor. The results obtained in terms of clock cycle are presented in Table 2 for TTA64 and Table 3 for Ascon-TTA64, respectively. Additionally, the instruction word length (IWL) information for the processors is provided in both tables.

As previously stated in Section 2.4, the TTA64 has two ALU FUs. The second ALU (ALU64_1) has a reduced number of operations, including those that are repeated frequently in ALU_64. In order to evaluate the impact of the aforementioned dual ALU structure on performance, ALU64_1 was removed from both the TTA64 and Ascon-TTA64 designs. Then, all simulations conducted in the first section were repeated. The test results are presented in Table 2 and Table 3, with columns designated with the suffixes -SA for the single ALU configuration and -DA for the dual ALU configuration. The Ascon-TTA64 model with a single ALU is shown in Fig. 8.

Table 2. Performance analysis of the TTA64 core for different core and FU configurations

Processor	Transport Buses	IWL	NoC-SA	NoC-DA
TTA64-R1	1	75	17652	16814
TTA64-R2	2	150	10947	9035
TTA64-R3	3	225	6627	6146
TTA64-R4	4	300	6597	5243
TTA64-R5	5	375	6595	4347
TTA64-R6	6	450	6595	4339

Table 3. Performance analysis of the Ascon-TTA64 ASIP for different core and FU configurations

Processor	Transport Buses	IWL	NoC-SA	NoC-DA
Ascon-TTA6-R1	1	75	10284	9942
Ascon-TTA6-R2	2	150	5722	5375
Ascon-TTA6-R3	3	225	3967	3659
Ascon-TTA6-R4	4	300	3332	3035
Ascon-TTA6-R5	5	375	2961	2716
Ascon-TTA6-R6	6	450	2953	2570

When Table 2 and Table 3 are evaluated together, it is clear that the proposed custom operations significantly increase performance. Comparing the single ALU versions, a performance gain of approximately 42% to 50% is observed. This value is between 38% and 41% for the dual ALU configurations. For TTA64 core, the contribution of the transport bus count to performance becomes ineffective after 3 buses in the single ALU versions. This situation is observed from 4 busses in the dual ALU versions. For this reason, configurations with more than 4 buses were not included in the Ascon-TTA64 comparison. On the other hand, the parallelization is much better in the Ascon-TTA64 core: Although the performance increase gradually slows down, the positive contribution continues up to 5 buses in the single ALU configurations and up to 6 buses in the dual ALU versions.

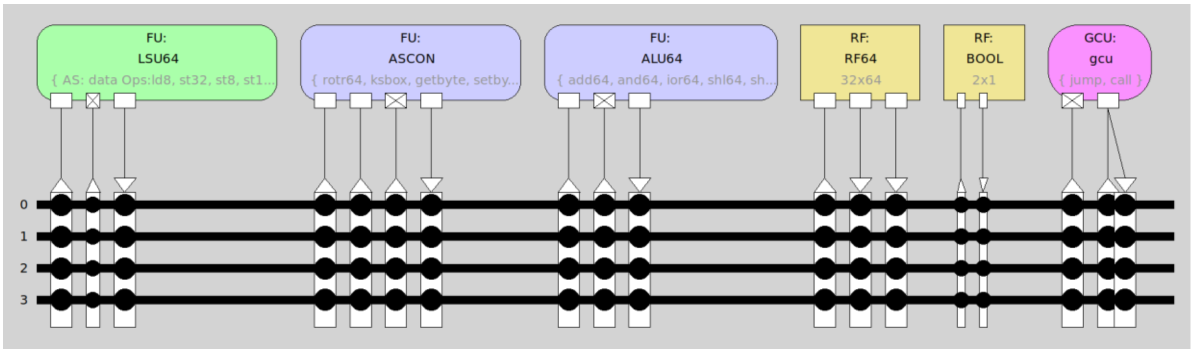

FU:
LSU64
{ AS: data Ops:ld8, st32, st8, st1...
FU:
ASCON
{ rotr64, ksbox, getbyte, setby...
FU:
ALU64
{ add64, and64, ior64, shl64, sh...
RF:
RF64
32x64
RF:
BOOL
2x1
GCU:
gcu
{ jump, call }
0
1
2
3

Figure 8. The ProDe model of the Ascon-TTA64-SA configuration

3.3 FPGA implementation

Following the completion of the performance analysis, the HDL implementation development phase was initiated. HDBEditor in the OpenASIP toolkit allows storing and managing HDL codes. The OpenASIP toolkit provides a multitude of pre-installed hardware designs, which serve as a reference for designers undertaking their own customizations. HDBEditor was employed to integrate the HDL codes of the ASCON FU into the TTA64 HDL implementation, which was previously developed in the prior studies [14-16]. Consequently, the Ascon-TTA64 ASIP has become now fully RTL-producible.

ProGe is a tool that facilitates the generation of RTL code for an entire processor through the parametrically combined use of FUs whose HDL designs were previously registered in the HDBEditor. HDL implementations of the TTA64 and Ascon-TTA64 cores were generated via ProGe. It is important to note here that the MUL_MAC64 FU has been removed from the TTA64 cores for a fair comparison. Subsequently, the processors were synthesised for an FPGA (part number xc7a100t [24]) using Xilinx Vivado IDE [25]. Table 4 illustrates the post-synthesis hardware resources and maximum clock frequency values for single (-SA) and dual (-DA) ALU64 configurations. As expected, integrating custom operations into the processor architecture resulted in a slight increase in the total chip area. However, this increase is modest when considered together with the performance gain.

Table 4. Required FPGA resources for template and custom cores

Processor	Transport Buses	LUT	FF	f_{max} (MHz)
TTA64-SA	2	3054	2123	108
TTA64-DA	4	4348	3125	104
Ascon-TTA64-SA	2	3845	2627	103
Ascon-TTA64-DA	4	5336	3682	101

4 CONCLUSION

Lightweight cryptography is a fundamental requirement for security, especially in resource-constrained devices. In this study, application-specific TTA operations have been developed for the Ascon encryption algorithm, which has become a standard in recent years, and these operations have been integrated into a 64-bit TTA processor. The analysis of the algorithm, the design and the integration of the operations are explained in detail. Comparative analyses have been carried out on how the custom operations can increase the processing speed of the Ascon-128AEAD algorithm. The results obtained show that the performance increase can vary between 38% and 50% depending on the core configuration. HDL code was developed for the designed ASIP and synthesised on FPGA. This research shows that the development of custom TTA operations offers significant potential for efficient acceleration of the Ascon encryption algorithm. In future studies, research such as adapting the custom operations to all Ascon parameter sets and performing transport bus optimisation will provide more useful results in this area.

Author Contributions

Latif AKÇAY: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - Original Draft, Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition

All authors read and approved the final manuscript.

Conflict of interest

No conflict of interest was declared by the author.

References

[1] W. J. Buchanan, S. Li, and R. Asif, “Lightweight cryptography methods,” *J. Cyber Secur. Technol.*, vol. 1, no. 3–4, pp. 187–201, 2017. https://doi.org/10.1080/23742917.2017.1384917

[2] K. Mohajerani, L. Beckwith, A. Abdulgadir, J.-P. Kaps, and K. Gaj, “Lightweight champions of the world: Side-channel resistant open hardware for finalists in the NIST Lightweight Cryptography Standardization Process,” *ACM Trans. Embed. Comput. Syst.*, 2024. https://doi.org/10.1145/3677320

[3] Sonmez Turan M, McKay K, Chang D, Bassham LE, Kang J, Waller ND, Kelsey JM, Hong D, “Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process,” (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) NIST IR 8454., 2023. https://doi.org/10.6028/NIST.IR.8454.

[4] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, “Ascon v1.2: Lightweight authenticated encryption and hashing” *J. Cryptology*, vol. 34, no. 3, 2021. https://doi.org/10.1007/s00145-021-09398-9

[5] C. Shekhar, Raj Singh, A. S. Mandal, S. C. Bose, R. Saini and P. Tanwar, "Application Specific Instruction Set Processors: redefining hardware-software boundary," *17th International Conference on VLSI Design. Proceedings.*, Mumbai, India, 2004, pp. 915-918, doi: 10.1109/ICVD.2004.1261047.

[6] H. Corporaal and M. Arnold, “Using transport triggered architectures for embedded processor design,” *Integr. Comput. Aided Eng.*, vol. 5, no. 1, pp. 19–38, 1998. doi: 10.3233/ICA-1998-5103

[7] P. Hamalainen, J. Heikkinen, M. Hannikainen and T. D. Hamalainen, "Design of transport triggered architecture processors for wireless encryption," *8th Euromicro Conference on Digital System Design (DSD'05)*, Porto, Portugal, 2005, pp. 144-152, doi: 10.1109/DSD.2005.33.

[8] P. Hamalainen, M. Hannikainen, T. Hamalainen, H. Corporaal and J. Saarvinen, "Implementation of encryption algorithms on transport triggered architectures," *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, Sydney, NSW, Australia, 2001, pp. 726-729 vol. 4, doi: 10.1109/ISCAS.2001.922340.

[9] T. Viitanen, H. Kultala, P. Jääskeläinen, and J. Takala, “Heuristics for greedy transport triggered architecture interconnect exploration,” in *Proceedings of the 2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2014. https://doi.org/10.1145/2656106.265612

[10] El-Hadidi, M. T., Elsayed, H. M., Osama, K., Bakr, M., & Aslan, H. K. (2018). Optimization of a novel programmable data-flow crypto processor using NSGA-II algorithm. Journal of Advanced Research, 12, 67–78. https://doi.org/10.1016/j.jare.2017.11.002

[11] Multanen, J., Kultala, H., Jaaskelainen, P., Viitanen, T., Tervo, A., & Takala, J. (2018). LoTTA: Energy-efficient processor for always-on applications. *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*. https://doi.org/10.1109/SiPS.2018.8598408.

[12] W. Guo, Y. Liu, S. Bai, J. Wei, and D. Sun, “Hardware architecture for RSA cryptography based on residue number system,”*Trans. Tianjin Univ.*, vol. 18, no. 4, pp. 237–242, 2012. doi:10.1007/s12209-012-1902-7

[13] A. Hakkala, J. Isoaho, and S. Virtanen, “Towards adaptive cryptography and security with software defined platforms,” in *Computing Platforms for Software-Defined Radio*, Cham: Springer International Publishing, 2017, pp. 209–236. https://doi.org/10.1007/978-3-319-49679-5_11

[14] J. Wei, W. Guo, H. Liu, and Y. Tan, “A Unified Cryptographic Processor for RSA and ECC in RNS,” in *Communications in Computer and Information Science*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 19–32. https://doi.org/10.1007/978-3-642-41635-4_3

[15] J. Hu, W. Guo, J. Wei, Y. Chang and D. Sun, "A Novel Architecture for Fast RSA Key Generation Based on RNS," *2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, Tianjin, China, 2011, pp. 345-349, doi: 10.1109/PAAP.2011.75.

[16] L. Akçay and B. Ö. Yalçın, "Analysing the potential of transport triggered architecture for lattice-based cryptography algorithms," *International Journal of Embedeed Systems*, vol. 15, no. 5, p. 404, 2022. https://doi.org/10.1504/IJES.2022.127164

[17] L. Akcay and B. Ors, "Custom TTA operations for accelerating kyber algorithm," in *2021 13th International Conference on Electrical and Electronics Engineering (ELECO)*, 2021. doi: 10.23919/ELECO54474.2021.9677863.

[18] L. Akçay and B. Ö. Yalçın, "Lightweight ASIP Design for Lattice-Based Post-quantum Cryptography Algorithms," *Arabian Journal for Science and Engineering*, 1-15. https://doi.org/10.1007/s13369-024-08976-w

[19] Akçay, L., & Örs, B. (2021). Comparison of RISC-V and transport triggered architectures for a postquantum cryptography application. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, *29*(1), 321–333. https://doi.org/10.3906/elk-2003-27

[20] F. Abed, C. Forler, and S. Lucks, "General classification of the authenticated encryption schemes for the CAESAR competition," *Computer Science Review.*, volume 22, pp. 13–26, 2016. https://doi.org/10.1016/j.cosrev.2016.07.002

[21] J. A. Fisher, "Very Long Instruction Word architectures and the ELI-512," in *Proceedings of the 10th annual international symposium on Computer architecture - ISCA '83*, 1983. https://doi.org/10.1145/800046.801649

[22] K. Hepola, J. Multanen and P. Jääskeläinen, "OpenASIP 2.0: Co-Design Toolset for RISC-V Application-Specific Instruction-Set Processors," *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Gothenburg, Sweden, 2022, pp. 161-165, doi: 10.1109/ASAP54787.2022.00034.

[23] Ascon Team, "ascon/ascon_collection: A collection of Ascon implementations & documents (as submodules)," *GitHub*, 2014. https://github.com/ascon/ascon_collection (accessed Nov. 10, 2024).

[24] H. Modi and P. Athanas, "In-system testing of Xilinx 7-Series FPGAs: Part 1-logic," in MILCOM 2015 - 2015 IEEE Military Communications Conference, 2015.

[25] S. Chakraborty, "Vivado Design Tools," Designing with Xilinx® FPGAs, pp. 17–21, Oct. 2016, doi: 10.1007/978-3-319-42438-5_2.