End-to-end Integration of OpenTitan Security Features in a Pure RISC-V SoC

End-to-end Integration of OpenTitan Security Features in a Pure RISC-V SoC

Alberto Musa*, Emanuele Parisi*, Luca Barbierato†, Andrea Acquaviva*, and Francesco Barchi*

* Università di Bologna
Bologna, Italy
Email: francesco.barchi@unibo.it

† Politecnico di Torino
Torino, Italy
Email: luca.barbierato@polito.it

***Abstract*—The RISC-V open-source instruction set architecture (ISA) is gaining popularity in the realm of new processor development. Its maturity allows implementation even in scenarios where security plays a key role. OpenTitan (OT), an open-source silicon Root of Trust (RoT) designed specifically for secure embedded environments is a significant example of this adoption. This study focuses on integrating OT into a System-on-Chip (SoC) exclusively featuring RISC-V architectures. In this configuration, OT serves as a secure co-processor, leveraging its cryptographic accelerators to enhance the performance of cryptographic workloads. A noteworthy aspect of this implementation is the intentional preservation of strong isolation in computation and memory, an essential feature to protect sensitive data, including cryptographic keys and intermediate results of cryptographic tasks. The integration of OT into a complete RISC-V SoC was executed and characterized on an FPGA. The FPGA runs the entire SoC which leverages a specialized communication system between two domains: a Host domain and a Safe domain. The Host domain features a CVA6 processor running Linux, while the Safe domain houses the OT system. During system characterization, the delays introduced by the communication and synchronization system between the Host and Safe domains were measured, along with the performance of cryptographic operations conducted in the Safe domain. Results demonstrate the effectiveness of OT HW/SW integration, compensating for the overhead introduced by the communication and synchronization system between the two domains. This makes the proposed implementation sustainable for various application cases and facilitates its integration into embedded and cyber-physical systems based on secure open-hardware architectures.**

I. Introduction

The research community is exploring the use of open-hardware RISC-V platforms in security-sensitive environments, as highlighted in studies such as [1], [2]. To enable secure computation without compromising computational efficiency, these systems often incorporate dedicated hardware extensions, such as ISA extensions, cryptographic accelerators, secure enclaves, and on-chip Root-of-Trust (RoT) [3], [4].

OpenTitan represents the first open-source project to use RISC-V for the implementation of silicon RoT. It consists of an Ibex RISC-V microcontroller and incorporates cryptographic hardware accelerators to perform cryptographic hash functions (such as SHA-256), message authentication codes (HMAC), block ciphers (AES), and accelerators for asymmetric cryptography operations (such as the OpenTitan

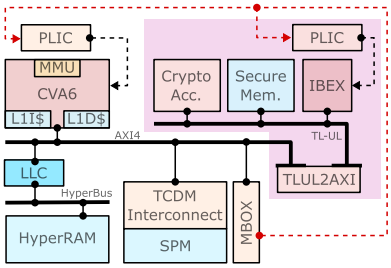

Fig. 1: System-on-Chip architecture.

Big Number accelerator, aka OTBN). These operations are essential for establishing secure communication channels and ensuring the authenticity and integrity of data. OpenTitan also provides secure memory for storing cryptographic keys and secrets in general. Researchers have utilized OpenTitan to introduce advanced security features such as Secure Boot [5], the implementation of execution integrity schemes [6], and the incorporation of unconventional communication channels in a SoC [1]. Despite the growing interest in the security validation of OpenTitan [7], and the analysis of its architectural bottlenecks [8], there is a lack of a feasibility analysis regarding the performance of OpenTitan when used as a security co-processor, considering the complex system interactions with the memory hierarchy of the integrated System-on-Chip (SoC). Two key characteristics of OpenTitan, when used as a security co-processor, limit the potential performance gains ideally achievable from its cryptographic accelerators. i) The physical isolation of OpenTitan from the main system-on-chip core requires the secure microcontroller to retrieve data through the system-on-chip memory hierarchy before forwarding them to the cryptographic accelerators. Depending on the organization and performance of the target system memory hierarchy, the ideal throughput of the accelerator may be significantly reduced. ii) The absence of a transparent data movement mechanism (such as DMA) means that the entire responsibility for data transfer lies with the OpenTitan secure microcontroller, further limiting the effectiveness of the cryptographic accelerators. This study presents two main contributions: **Integration**: Starting with a SoC, depicted in Figure 1, composed of CVA6, a modern RISC-V application-class core widely used in RISC-V embedded platforms,

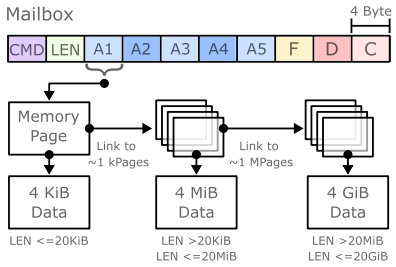

Fig. 2: Scheme of the Mailbox structure and its fields.

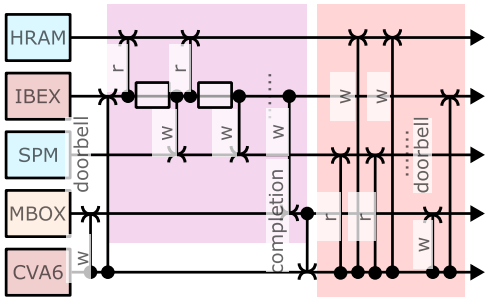

Fig. 3: Scheme of the Offloading communication protocol.

and OpenTitan, we developed the necessary software to use OpenTitan as a secure co-processor. The interaction between the application domain and the secure domain is supported by a dedicated communication system, leveraging firmware running on the secure microcontroller inside OpenTitan and a Linux driver running on the CVA6 application processor side. **Characterization**: By running the system on an FPGA, we assess the effectiveness of three cryptographic accelerators in OpenTitan: SHA, and AES, using different workloads. We measure the end-to-end latencies of the HW/SW implementation, characterizing also the impact of the memory hierarchy involved in payload communication. We compare the results obtained with a software implementation. Our analysis reveals that the adoption of OpenTitan results, using a payload of 4 KiB, in a performance improvement of 13x in SHA and 5x in AES compared to a pure software implementation. We observe that, due to non-negligible memory access latencies and the absence of a transparent data transfer mechanism in the current implementation, the overhead introduced by the communication protocol in the System-on-Chip (SoC) ranges from 5% to 42% of the operating cost of the cryptographic accelerators. Thus, the present study opens the way to more effective integration of Host and Secure domains through the optimization of the communication subsystem.

II. Background

OpenTitan is an open-source project by lowRISC for the development of a hardware RoT used to protect sensitive systems from hardware attacks and tampering. There are different versions of the OpenTitan architecture, each tailored to a different use case: data centers, cloud systems, and embedded systems. In this study, we use the `top_earlgrey` version, which includes a RISC-V microcontroller, private memory, cryptographic hardware accelerators, and I/O interfaces.

The microcontroller is the Ibex core, a 32-bit low-power RISC-V CPU designed for embedded applications [9]. The TileLink Uncached Lightweight (TL-UL) protocol is used for component interconnection. The system incorporates a key manager to manage hardware identities and root keys, safeguarding critical resources from external threats and enabling Secure Boot of the system, and dedicated hardware accelerators to efficiently handle cryptographic calculations. An HMAC accelerator allows the acceleration of Secure Hash Functions (SHA-256 and SHA-3) and Message Authentication Codes (HMAC and KMAC). An AES accelerator enables the acceleration of the Advanced Encryption Standard (AES), a symmetric block cipher algorithm with a block size of 16 bytes, supporting key lengths of 128, 192, and 256 bits. In this study, we will use the default implementation of the AES accelerator in CBC mode with a 256-bit key. Under these conditions, encrypting a block requires 72 cycles.

To assess the effectiveness of OpenTitan as a secure co-processor, we implemented a System-on-Chip (SoC) on an FPGA based on the design outlined in [1] and illustrated in Figure 1. This SoC comprises an application subsystem and a secure subsystem. The application subsystem consists of a CVA6 core [10] and a multicore cluster designed to efficiently handle signal processing and machine learning tasks [11]. The secure subsystem was constructed by integrating OpenTitan as a bus-master with unlimited access to all levels of memory within the SoC. The host's memory hierarchy includes three levels: L1) private data and instruction cache memories for CVA6, L2) an integrated Scratchpad Memory (SPM), and L3) an external HyperRAM serving as the main memory. Additionally, the system features a Last-Level Cache (LLC) to enhance the performance of HyperRAM accesses.

The application subsystem uses an AXI4 interconnect, while the secure subsystem relies on a TileLink interconnect. To enable the interaction between the Ibex core and the CVA6 core, a TileLink-AXI4 bridge, named TLUL2AXI, has been implemented as shown in Figure 1. This module acts as a slave of the TL-UL interconnect, exposing AXI4 master interfaces to the OpenTitan input-output interface. This configuration allows OpenTitan to exert control over the primary interconnect through a dedicated master port, providing visibility into the entire memory map of the SoC. The communication between the CVA6 core and the Ibex core is limited to the exchange of messages through a mailbox system (MBOX in Figure 1) managed via interrupts. The mailbox, along with the TLUL2AXI bridge, allows the main processor to delegate security-sensitive responsibilities to the secure subsystem.

III. Software and Benchmarking Design

This section presents the communication and signaling protocol we designed to exchange data between the host core and the OpenTitan RoT. When transmitting data from the CVA6 to OpenTitan, metadata concerning the cryptographic operation to be performed, along with its payload, are passed through the mailbox. The payload is transmitted in the form of a sequence of physical memory addresses. Leveraging the fact

that OpenTitan operates as a bus master in our architecture, it can freely access every location in the SoC address space using the physical memory addresses provided. This setup enables efficient memory access for the Ibex microcontroller within OpenTitan. Two key features, Secure Boot and Memory protection, ensure the integrity and confidentiality of data exchanged through the mailbox, solving the problems of potential manipulation during communication offloading: i) Secure Boot: OpenTitan verifies and authenticates all firmware and software components executed in CVA6 and OpenTitan itself. This process ensures that the code to be executed remains unaltered. ii) Memory Protection: Write accesses to the mailbox are allowed only when the processor is in S-mode. This execution mode, compliant with the RISC-V supervisor specification, is activated only when internal Linux kernel functionalities are running.

Upon completion of the cryptographic operation, OpenTitan must write back the results to the host memory of the CVA6. However, the lack of coherence between the RoT in our SoC and the CVA6 host core makes it impossible to simply copy the output data to a given address in memory. Specifically, OpenTitan and CVA6 are not coherent, and OpenTitan lacks the capability to invalidate the L1 data cache of the CVA6 during its memory accesses. Consequently, there exists a risk that the CVA6 may not utilize the most recent results of the OpenTitan computation, especially if a memory location written by OpenTitan is already present and valid in the CVA6 data cache. To address this challenge, we utilize the SoC SPM, an uncacheable memory region accessible by both OpenTitan and CVA6, as a temporary storage space for the output data produced by OpenTitan. After OpenTitan has copied the operation output to the SPM, the software running on CVA6 is responsible for transferring this data to the main memory hierarchy, ensuring that the most recent results are available to any software reading such data.

The communication protocol between the CVA6 and OpenTitan is illustrated in Figure 3. Initially, the CVA6 specifies the command to be executed and the payload metadata via the mailbox and asserts the doorbell register to awaken the Ibex core within OpenTitan. Subsequently, OpenTitan performs the requested cryptographic operation, loading data from the system RAM memory into its private local memory, processing the data using cryptographic accelerators, and ultimately copying the operation results into the SoC SPM. Upon filling the SPM or completing the communication, OpenTitan signals completion by asserting the Completion register in the mailbox, triggering an interrupt request to the CVA6 core. If the requested operation is completed, the CVA6 driver responsible for managing OpenTitan operations transfers the data produced by OpenTitan from the SPM to the main memory. This sequence of operations may be iterated multiple times, if the SPM becomes full before the operation is completed (e.g., when OpenTitan is working with a payload larger than the SPM). Such scenarios are managed using special flag bits in the mailbox whose functioning details will now be described.

The mailbox serves as a SoC shared memory space enabling the communication between the CVA6 host core and OpenTitan. It comprises 8 32-bit general-purpose registers and two reserved to trigger interrupt requests toward the CVA6 and Ibex cores, namely the Doorbell register `D`, and the Completion register `C`. Our communication protocol assigns a precise meaning to each general-purpose register to facilitate the exchange of metadata essential for communication. For simplicity, we refer to these 8 mailbox fields as indicated in Figure 2: i) `CMD`, ii) `LEN`, iii-vii) `A1`−`A5`, and viii) `F`. The `CMD` and `LEN` registers specify the cryptographic operation to be executed and the length in bytes of the payload. Address fields, denoted from `A1` to `A5`, encode the physical addresses of the data to be processed by the OpenTitan security accelerators. Our protocol supports three addressing schemes, namely i) *direct*, ii) *indirect-L1*, and iii) *indirect-L2*.

In the *direct* addressing scheme, the `A*` fields in the mailbox represent the physical addresses of 4 KiB data chunks which is the typical virtual memory page physical dimensions in modern operating systems, such as Linux, ensuring that the data are physically contiguous in memory. The *direct* addressing scheme supports payloads up to 20 KiB in size, while *indirect-L1* and *indirect-L2* addressing is meant to handle larger payloads. In the *indirect-L1* addressing scheme, each address corresponds to a physical page of host memory populated with physical addresses to pages of data, extending the addressable data range to 20 MiB. Similarly, the *indirect-L2* methodology introduces a second level of indirection, further expanding the addressable address space to 20 GiB. Our protocol does not support further levels of indirection under the assumption that embedded systems seldom, if ever, necessitate processing payloads larger than 20 GiB, a capacity already surpassing typical memory sizes available in such systems.

Lastly, the `F` field comprises two flag bits, namely `F_STATUS` and `F_FINISH`, serving as control flags essential for managing the communication protocol state machine. Initially, when the first transaction is initiated from the host core towards OpenTitan, both bits are cleared. The `F_FINISH` bit indicates the completion of the requested operation. In this scenario, OpenTitan sets the Completion bit with the `F_FINISH` and `F_STATUS` flags set. Subsequently, the CVA6 retrieves data from the SPM and completes the operation by setting the doorbell bit with the `F_FINISH` bit set and the `F_STATUS` bit cleared. Conversely, if the `F_FINISH` bit is cleared, and the protocol state machine is not in the initial state, it means that the SPM is full, yet the requested operation has not been completed. In such cases, CVA6 reads available data from the SPM and triggers a new doorbell IRQ with the `F_STATUS` flag cleared, indicating that the SPM has been drained and OpenTitan may resume its operations.

We implemented the communication protocol into the Linux operating system, and instrumented the developed software to measure the execution cost in cycles of various protocol steps and the cryptographic operations executed on the OpenTitan platform. To quantify the execution cost in cycles, we utilize native performance monitoring registers on both the CVA6 and OpenTitan sides. On the CVA6 side, we utilize the `CYCLE`

Block Size	SW/FW		OpenTitan		SW/FW	Pure	
(Bytes)	CVA6	IBEX	Operation	Total	Overhead	Software	Speedup
SHA-256							
1024	1347	801	10454	12602	17.0%	144231	11.4
2048	1270	801	20416	22487	9.2%	276753	12.3
4096	1263	801	40394	42458	4.9%	541516	12.8
AES-CBC-256							
1024	7436	8271	29944	45651	34.4%	182039	4.0
2048	14730	15925	46600	77255	39.7%	364078	4.7
4096	26334	31281	79939	137554	41.9%	724638	5.3

TABLE I: Comparison (# cycles) of the proposed offloading mechanism with respect to a pure software implementation (column: "Pure Software"). Columns related to CVA6 software and IBEX firmware overheads (SW/FW) are also indicated.

CSR to measure cycle counts, while on the OpenTitan side, we employ the `MCYCLEH` and `MCYCLE` registers for the same purpose. Within the CVA6 Linux device driver, we measure the cycles required for acquiring memory-mapped addresses corresponding to the mailbox address space using the `ioremap` system call. Subsequently, we gauge the overhead of populating the mailbox with relevant metadata and asserting the doorbell interrupt to trigger communication with OpenTitan. Afterward, we measure the cycles involved in resuming driver operations following the completion interrupt asserted by OpenTitan. This includes reading the contents of the mailbox and moving data from the SPM into the host core memory hierarchy. On the OpenTitan side, we benchmark the execution cost of the requested cryptographic operations and the subsequent copying of operation results into the SPM. This comprehensive analysis provides insights into the performance characteristics of both the communication protocol and the cryptographic operations executed by OpenTitan.

IV. Experimental Results

Table I presents the outcomes achieved through our benchmark setup, comprising an FPGA operating a System on Chip (SoC) with Linux running on CVA6, OpenTitan, and the communication mailbox, as detailed in previous sections. The "SW" column in the table denotes the number of cycles required to execute SHA or AES implemented directly in software. This metric is determined using the `openssl speed` command for each considered payload size and serves as our reference point for comparing the cycles of our solution. Cycles needed by our solution are divided between cycles executed by CVA6 and IBEX. The cycles attributed to CVA6 consist solely of overhead cycles resulting from communication between the two environments, facilitated by MBOX and SPM. On the other hand, the cycles associated with IBEX are further categorized into overhead cycles dedicated to managing the communication protocol and operational cycles during which cryptographic accelerators are actively utilized. The last two columns illustrate the overall overhead arising from the communication protocol and the speedup achieved in comparison to the pure software version. Analyzing the SHA results, it becomes evident that the suggested solution significantly enhances system performance, exhibiting a speedup ranging from 11x to approximately 13x. This improvement varies depending on the volume of data being processed when the SHA accelerator is employed. For SHA, the protocol overhead tends to decrease as the payload dimension increases. This is attributed to the fact that the output of the algorithm consistently remains at 256 bits, regardless of the input size. Consequently, the substantial data transfer occurs solely in IBEX, reading from HRAM. In this scenario, the Scratchpad Memory (SPM) is employed to store only 32 bytes. Analyzing the results of AES, we also notice a significant improvement in performance in this case when using the security co-processor, with a speedup between 4x and 5x. In this scenario, the accelerator's usage is limited by the memory transfers required for the operation of the communication protocol via MBOX and SPM. The overhead increases instead of decreasing as the payload size grows, going from 34% to 42%. This is due to the lack of tools to optimize and mask data movement, which occurs in the output of the algorithm when heavily utilizing the SPM in this case. Despite this, the speedup is remarkable, and it will be further enhanced in the future.

V. Conclusions

In this work, we have shown an end-to-end integration of Opentitan into a SoC to serve as a security co-processor. We evaluated its performance through a complete FPGA implementation and we characterized the communication overheads between Host and Secure domains. Future work will focus on reducing communication overhead, for instance by introducing specialized hardware to mask memory transfers.

References

[1] M. Ciani *et al.*, "Cyber security aboard micro aerial vehicles: An opentitan-based visual communication use case," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[2] P. Schönle *et al.*, "A multi-sensor and parallel processing soc for miniaturized medical instrumentation," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2076–2087, 2018.

[3] D. Lee *et al.*, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020.

[4] P. Nasahl *et al.*, "Hector-v: A heterogeneous cpu architecture for a secure risc-v execution environment," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, p. 187–199.

[5] A. Wagner *et al.*, "To be, or not to be stateful: Post-quantum secure boot using hash-based signatures," in *Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security*, 2022, p. 85–94.

[6] E. Parisi *et al.*, "Titancfi: Toward enforcing control-flow integrity in the root-of-trust," *arXiv preprint arXiv:2401.02567*, 2024.

[7] A. Meza *et al.*, "Security verification of the opentitan hardware root of trust," *IEEE Security & Privacy*, vol. 21, no. 3, pp. 27–36, 2023.

[8] E. Parisi *et al.*, "Assessing the performance of opentitan as cryptographic accelerator in secure open-hardware system-on-chips," *To appear at 21th ACM International Conference on Computing Frontiers*, 2024.

[9] P. Davide Schiavone *et al.*, "Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[10] F. Zaruba *et al.*, "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, 2019.

[11] D. Rossi *et al.*, "Pulp: A parallel ultra low power platform for next generation iot applications," in *2015 IEEE Hot Chips 27 Symposium (HCS)*, 2015, pp. 1–39.