Securing and Optimizing Data Flow by Validating Die Level Interconnect Stress with AES XTS 256-bit Encryption and Ring Interconnect Analysis on edge client SoCs

Gowri Shankar Somanjeri
Pankaj Sharma, Ashutosh Mishra, Naveen Urs T R, Aravind Krishnan, Srishti Singh Chauhan
Intel Technology India Pvt Limited, No 23/56-P, Outer Ring Road, Bellandur, Bengaluru – 560103

***Abstract*- This paper presents an innovative approach to enhancing system configuration testing by integrating advanced memory encryption technologies into existing frameworks. The Front-end Tool, central to this study, generates test programs with randomized parameters but faces challenges in incorporating Multi-Key Total Memory Encryption (MKTME) capabilities for comprehensive validation. Current integration within the Die Level Interconnect Stress program offers basic MKTME support but lacks efficient utilization of key aliases. To address this, a solution is proposed to convert front-end tool-generated test lines into MKTME-compatible lines, thereby improving the testing process. The Stress tool is designed to enforce MKTME command assignments to each target, globally applying encryption algorithms and allocating KeyIDs. Enhanced with options for extended MKTME attributes and key aliasing, this tool allows multiple MKTME keys to map to the same memory location, facilitating advanced testing scenarios and ensuring thorough evaluation of encryption features across diverse system configurations.**

Keywords— Total Memory Encryption (TME), Multi-Key Total Memory Encryption (MKTME), Advanced Encryption Standard - XEX-based Tweaked CodeBook mode with ciphertext Stealing (AES-XTS), Processor Reserved Memory Range (PRMRR), System-on-Chip (SoC).

I. Introduction

This paper explores advanced memory encryption and stress testing technologies essential for enhancing system security and performance. It features Total Memory Encryption (TME), which encrypts all system memory using AES-XTS with 256-bit keys [1], [2]. TME is configured through BIOS, ensuring secure data transmission across external memory buses and maintaining compatibility with existing software without requiring changes [3]. Expanding on TME, Total Memory Encryption-Multi-Key (TME-MK) enables page-level encryption with multiple keys, managed by software [4]. This allows for more precise control over data security and supports both CPU-generated and software-provided keys. TME-MK is especially beneficial in virtualized environments, allowing legacy operating systems to leverage modern encryption features and paving the way for future security enhancements. The AES-XTS encryption engine protects data as it moves across external memory buses, keeping CPU-generated keys hidden from software for added security. For systems with Non-Volatile Random-Access Memory (NVRAM), key reuse across power cycles offers flexible and efficient security management. Additionally, the paper highlights the Coherency Stress tool, a robust solution for stress testing system cache coherency [5], [6]. Supporting up to n agents, the Coherency Stress tool validates the effectiveness of memory encryption schemes and integrates with Multi-Key Total Memory Encryption (MKTME) [4]. Overall, the paper demonstrates how these encryptions and testing technologies work together to protect modern computing environments, offering insights into improving both security and performance.

II. Validation Challenges

The integration of Multi-Key Total Memory Encryption (MKTME) into system testing frameworks brings several validation challenges. Ensuring module correctness is vital so that tool-generated tests are accurately converted into

MKTME-compatible formats [4]. Key management must handle unique KeyID assignments, inheritance by threads, and aliasing cases where multiple keys overlap memory. Extended attributes require thorough coverage, including limits on key counts and overlapping ranges. Stress-testing system coherency under encryption scenarios is essential to prevent corruption or race conditions [5], [6]. Robust automation with error detection is needed for aliasing or unsupported setups [6]. Performance must be assessed for scalability and low overhead, while security validation ensures data isolation and prevents key leakage. Finally, interoperability with existing frameworks guarantees smooth adoption [3].

- **Key Management Complexity:** Unique KeyID assignment, inheritance, and aliasing across memory.
- **System Coherency Risks:** Data corruption or race conditions under concurrent, encrypted accesses.
- **Coverage & Compatibility:** Boundary testing, scalability, and seamless integration with legacy frameworks.

III. Background and Related Work

Validating interconnects under stress is crucial because it ensures that the communication pathways within the SoC can reliably handle high traffic, voltage, temperature, and frequency conditions that may occur during real-world operation [5], [6]. Stress testing helps uncover issues such as data corruption, timing violations, or bottlenecks that might not appear under normal conditions [6]. By subjecting interconnects to extreme scenarios, designers can identify and address weaknesses, ensuring robust data integrity, system stability, and long-term reliability [5]. This process is essential for meeting quality, reliability, and performance goals, and helps prevent costly failures or erratic behavior in the field [3].

The ring interconnect is responsible for moving data between cores, caches, and memory controllers. If data is not encrypted before it leaves the CPU core or cache, it could be intercepted by a malicious agent with physical access to the ring or memory interface [1]. Multiple agents (cores, I/O, accelerators) share the ring. Ensuring that one agent cannot snoop or tamper with another's data is critical, especially in virtualized environments or when running secure enclaves [4]. Attackers may attempt to replay old data or inject modified data into the ring. Without integrity and replay protection, these attacks could compromise system security. Multi-Key Total Memory Encryption (MKTME) allows different memory pages to be encrypted with different keys. The challenge is ensuring that keys are securely managed and only accessible to trusted hardware/firmware. Encryption and decryption add latency to memory accesses. The ring and memory controller must be designed to handle these delays without causing bottlenecks or back pressure [2], [3]. Not all memory regions may be encrypted. Ensuring that sensitive data is always encrypted and that exclusion policies are correctly enforced is crucial [4].

TME/MKTME encrypt all data leaving the SoC, so even if DRAM is physically removed, the data is unreadable without the keys [1]. MKTME allows each VM or secure enclave to have its own encryption key, preventing one VM from reading another's memory even if pages are reassigned [4]. PRMRR regions use counters and MACs to ensure that only fresh, authentic data is used and replayed or tampered data is detected and rejected. Encryption engines (AES-XTS) and MAC generation add cycles to memory transactions [2]. The ring and memory controller must be provisioned to handle these delays, especially for high-bandwidth scenarios [3], [1]. Intermediate buffers in the Crypto Engine and memory controller may hold plain text data. These are protected by hardware access controls and are not accessible to unrelated transactions or debug interfaces.

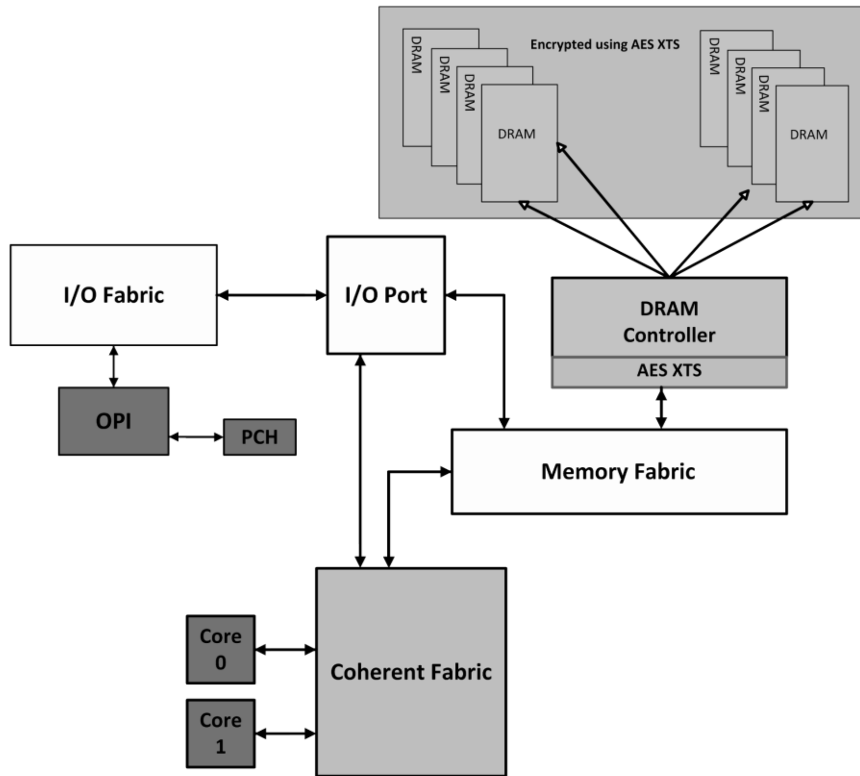

Figure 1: Coherent Fabric with multiple agents.

The Coherent Fabric is a scalable and configurable interconnect and caching solution used in SoC architecture [3], [5]. It is designed to efficiently connect multiple cores, cache slices, and system agents within a single package, supporting a wide range of products from mobile devices to servers [3]. Coherent Fabric manages data movement and coherence between compute elements through a ring-based topology, incorporating components that work together to handle memory requests, maintain cache coherence, and arbitrate access to shared resources [5], [6]. Its modular design allows flexibility and scalability, making it a key enabler for the performance and efficiency goals of modern SoC platforms [3].

IV. Validation Methodology

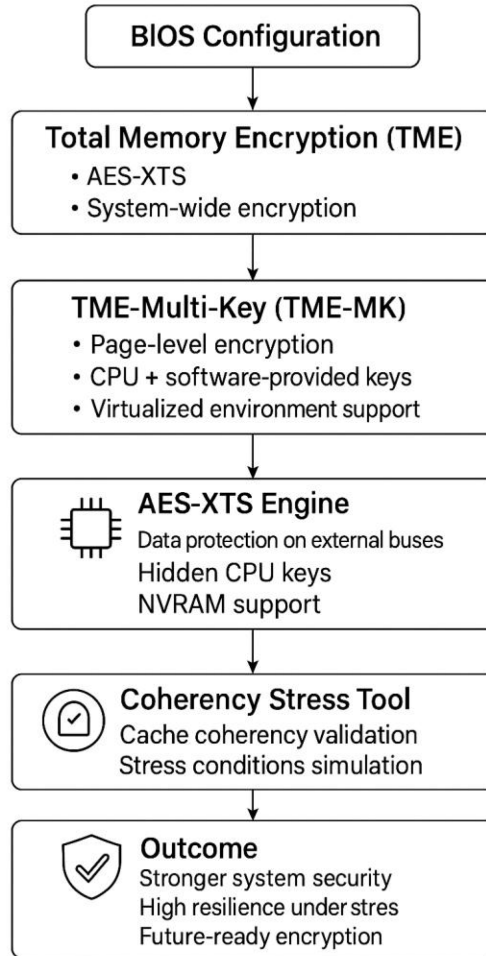

Figure 2: Implementation Details with Flow Chart

Existing test frameworks require integration of memory encryption technologies for comprehensive validation [1], [4]. The Front-end Tool generates test programs with randomized parameters but lacks direct MKTME support. Incorporating Multi-Key Total Memory Encryption (MKTME) into tests is necessary to validate encryption features [4]. A module around the front-end tool can convert generated test lines into MKTME-compatible test lines, enabling advanced testing scenarios and ensuring encryption features are evaluated across various configurations [6]. The tool enforces assignment of an MKTME command to each memory target. A system-managed process ensures that encryption keys are uniquely assigned and consistently applied across targets, while also supporting controlled scenarios where multiple keys may reference shared memory regions [1], [4]. This tests the system's ability to handle cache line eviction and coherency when different keys access the same address [5], [6]. It streamlines the process of integrating MKTME into existing test frameworks, enhances the thoroughness and flexibility of encryption feature validation, and facilitates advanced and realistic testing scenarios, including key aliasing and inheritance [4].

V. Results

A proposed solution around the Front-end Tool converts test lines into MKTME-compatible lines, enhancing the testing process and ensuring comprehensive validation of encryption features [4], [6]. The system Coherency Stress tool automates the assignment of MKTME commands, globally applying encryption algorithms and allocating KeyIDs, reducing manual intervention and potential errors [5], [6]. It also enables key aliasing for complex testing scenarios [4].

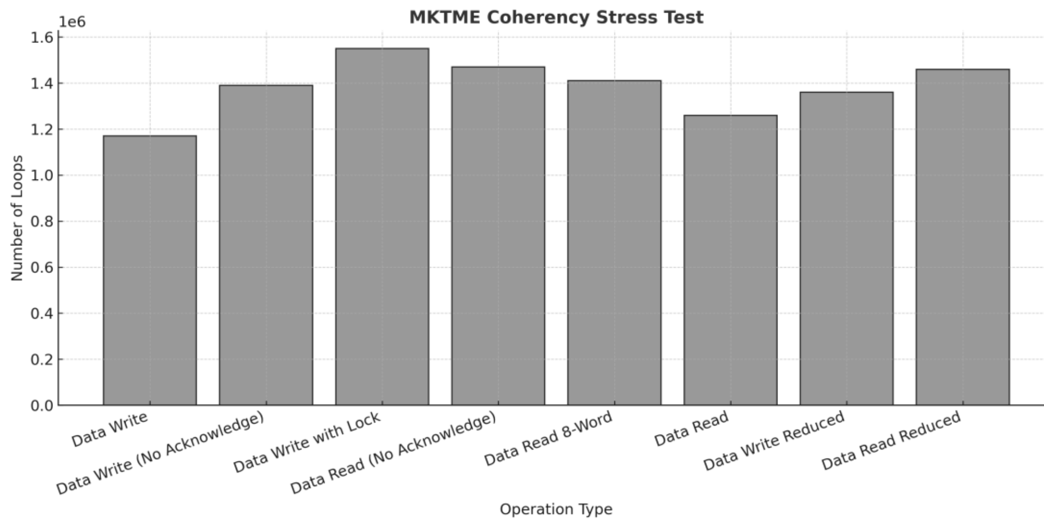

Figure 3: Sample Results generated by Coherency Stress Tool with MKTME enabled

VI. Conclusion And Impact

This paper addresses the critical need for integrating advanced memory encryption technologies into system configuration testing frameworks [1], [4]. By developing a module around the Front-end Tool, we have successfully enhanced the testing process to incorporate Multi-Key Total Memory Encryption (MKTME) capabilities. This approach allows for seamless conversion of test lines into MKTME-compatible formats, ensuring comprehensive validation of encryption features across diverse system configurations [6]. The Coherency Stress tool's automated assignment of MKTME commands and global algorithm application streamlines key management, reducing manual intervention and potential errors.

Overall, this solution represents a significant advancement in memory encryption testing, offering improved flexibility, scalability, and thoroughness compared to previous methods [3], [4]. As industries increasingly prioritize data security, the innovations presented in this paper provide valuable tools and insights for achieving robust and efficient encryption validation, paving the way for secure system deployment in complex digital environments [1], [4].

REFERENCES

[1] Intel Corporation, "Intel® Architecture Memory Encryption Technologies Specification," Rev. 1.0, Sept. 2019. [Online]. Available: https://www.intel.com

[2] S. Gueron and R. Johnson, "AES-XTS: Counter-Mode Encryption for Secure Memory," IACR Cryptology ePrint Archive, vol. 2010, pp. 1–24, 2010.

[3] Intel Corporation, Intel® 64 and IA-32 Architectures Software Developer's Manual: Vols. 1–3, Santa Clara, CA, USA: Intel, 2022.

[4] Intel Corporation, "Intel® Multi-Key Total Memory Encryption (MKTME)," White Paper, 2020.

[5] R. A. Hankins, J. D. Collins, and D. A. Wood, "Cache coherence and memory consistency: A programmer's perspective," IEEE Micro, vol. 22, no. 3, pp. 12–18, May/Jun. 2002.

[6] J. C. Mogul and A. Gupta, "Memory performance tests and stress tools for cache-coherent multiprocessors," in Proc. ACM SIGPLAN Workshop on Performance Evaluation of Software and Systems, Las Vegas, NV, USA, 1997, pp. 61–71.