Benchmarking SHA-2 and SHA-3 on MicroBlaze

Michael Rosales   
Department of Electrical and Computer Engineering *California State Polytechnic University, Pomona  
Pomona, California, USA*  
mrosales1@cpp.edu

Nathaniel Case  
Department of Electrical and Computer Engineering *California State Polytechnic University, Pomona  
Pomona, California, USA*  
ncase@cpp.edu

Michael Yen   
Department of Electrical and Computer Engineering *California State Polytechnic University, Pomona  
Pomona, California, USA*  
mjyen@cpp.edu

Noah Mendoza   
Department of Electrical and Computer Engineering *California State Polytechnic University, Pomona  
Pomona, California, USA*  
noahgmendoza@cpp.edu

***Abstract* — This project benchmarks the performance of SHA-2 and SHA-3 cryptographic hash functions implemented in software on the MicroBlaze softcore processor. Using the Nexys A7-100T FPGA, functionality, edge case, and performance tests with 1 KB and 1 MB inputs were conducted. Key metrics such as throughput, execution time, and energy consumption were analyzed. Results show SHA-2 achieved higher throughput for smaller inputs (12,288 bytes/s for 1 KB), while SHA-3 excelled for larger inputs (142 bytes/s for 1 MB) and demonstrated better energy efficiency. These findings highlight trade-offs in computational performance and energy consumption, guiding secure embedded system design.**

***Index Terms-* SHA-2, SHA-3, MicroBlaze, FPGA, performance benchmarking**

# Introduction

TheIn today's digital world, cryptographic hash functions are essential for ensuring data integrity, authentication, and security. Two prominent hash function standards, SHA-2 (Secure Hash Algorithm 2) and SHA-3 (Secure Hash Algorithm 3), are widely used in various applications, including digital signatures, secure communication protocols, and data integrity checks. While SHA-2 has been the cornerstone of cryptographic security for years, the emergence of SHA-3, designed as a more secure alternative, has sparked interest in understanding their relative performance and trade-offs.

SHA-2, developed by the National Institute of Standards and Technology (NIST) in 2001, is a family of hash functions known for its computational efficiency and robust security. Its widespread adoption in secure applications has made it a reliable standard. However, with advancements in cryptanalysis and emerging threats, NIST introduced SHA-3 in 2015 as a new standard. Based on the Keccak algorithm, SHA-3 features a fundamentally different design using a sponge construction, offering inherent resistance to certain cryptographic attacks, such as length-extension attacks. This divergence in design philosophy raises the question of how these two algorithms perform in constrained environments, such as embedded systems.

Embedded systems, often characterized by limited resources like processing power, memory, and energy, demand cryptographic solutions that are both secure and efficient. The **MicroBlaze softcore processor**, implemented on FPGAs, serves as an ideal reference for such environments due to its customizable nature and constrained resource profile. By evaluating the performance of SHA-2 and SHA-3 on MicroBlaze, insights can be drawn that apply to a wide range of embedded systems where computational resources are limited.

The motivation for this project lies in the need to provide a comprehensive comparison of SHA-2 and SHA-3 in a resource-constrained environment. While hardware acceleration could yield better performance, it would obscure the fundamental trade-offs between these algorithms by introducing optimization complexities specific to the hardware design. Software implementations, on the other hand, allow for a more direct comparison of computational efficiency and energy consumption, making the results more broadly applicable to other constrained systems.

This project aims to benchmark SHA-2 and SHA-3 on the MicroBlaze processor implemented on a Nexys A7-100T FPGA. Using open-source software implementations of each algorithm, the study evaluates their performance through functionality tests, edge cases, and performance tests with varying input sizes (1 KB and 1 MB). Metrics such as throughput, execution time, and energy consumption are analyzed to understand the trade-offs between the two algorithms. The findings provide valuable insights into the suitability of SHA-2 and SHA-3 for constrained environments, offering a reference point for developers and designers of embedded systems.

By focusing on software-based implementations, this study ensures a fair comparison of the algorithms without the influence of hardware optimizations. This approach highlights the core computational characteristics of SHA-2 and SHA-3, enabling their evaluation in contexts where hardware acceleration is not feasible or desirable. Ultimately, this project contributes to the field by offering guidance for secure and efficient cryptographic design in embedded and resource-constrained systems.

# Related works

The performance of SHA-2 and SHA-3 has been studied extensively, often focusing on hardware-accelerated implementations or optimized environments. A study published in the *Journal of Cryptographic Engineering* demonstrated that SHA-3 achieves superior throughput compared to SHA-2 in hardware-optimized environments. The researchers highlighted the benefits of SHA-3’s sponge construction for hardware acceleration, achieving significant performance improvements [1]. However, these results are less applicable to software-only implementations in resource-constrained environments, where hardware optimizations are not feasible.

Another relevant work by the Keccak team analyzed the performance of SHA-3 in various platforms, including software environments. The study pointed out that while SHA-3 is generally slower than SHA-2 in software due to its more complex design, its performance in hardware and energy efficiency often compensates for this drawback [2]. This highlights the importance of evaluating both algorithms in constrained environments to understand their trade-offs without the influence of hardware-specific optimizations.

## Comparisons

Unlike these studies, this project evaluates SHA-2 and SHA-3 in a software-only context using the MicroBlaze softcore processor, representing a constrained embedded system. The results reveal:

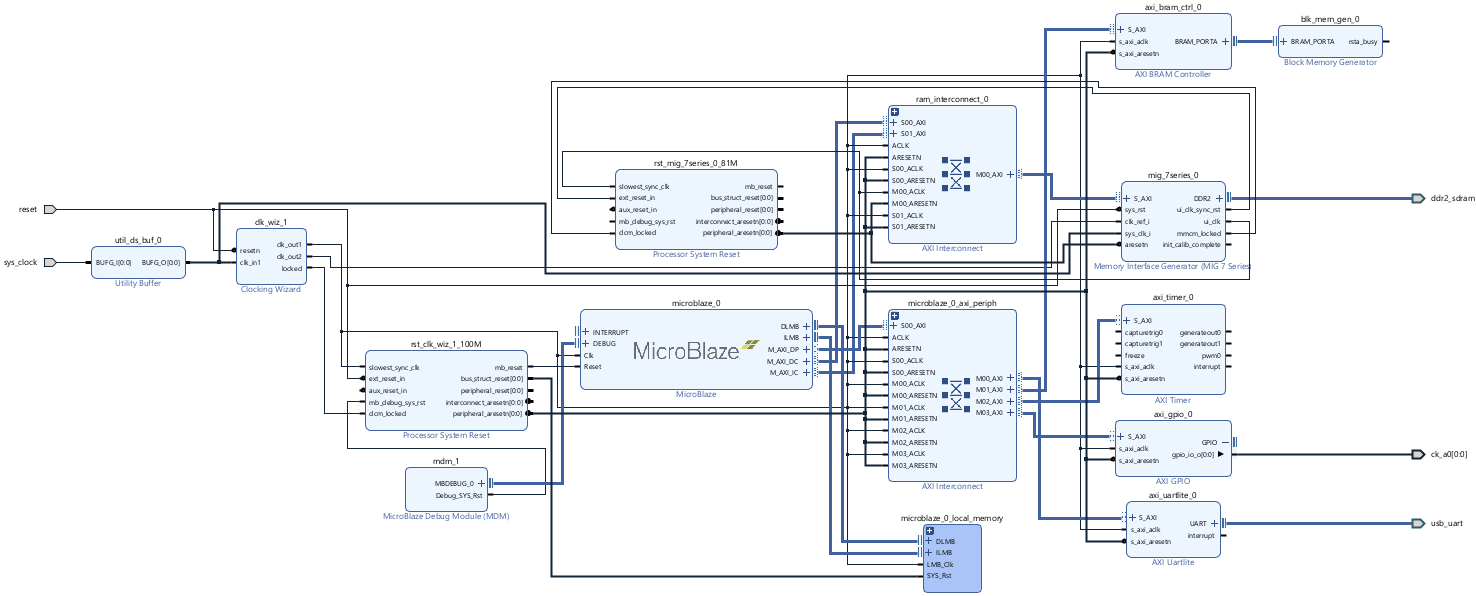
* **Throughput:** SHA-2 outperformed SHA-3 for smaller inputs (e.g., 12,288 bytes/s for 1 KB using SHA-256), while SHA-3 excelled in energy efficiency for larger inputs, such as 1 MB.
* **Energy Consumption: SHA**-3 showed better energy efficiency overall, making it more suitable for applications prioritizing power savings.

## Significance of this work

This study provides a unique perspective by focusing on software-only implementations of SHA-2 and SHA-3 on a resource-constrained platform. The findings are directly applicable to embedded systems, offering insights for algorithm selection in scenarios where hardware acceleration is impractical. By emphasizing the core computational trade-offs, this work complements existing studies and serves as a valuable reference for secure and efficient embedded system design.

# Hardware configuration and setup

The hardware setup for this project is centered on the MicroBlaze softcore processor, configured in Vivado and implemented on the Nexys A7-100T FPGA. This section provides a detailed technical overview of the processor configuration and supporting peripherals to explain the methodology of benchmarking SHA-2 and SHA-3 cryptographic hash functions.



1. *MicroBlaze Block Diagram*

## MicroBlaze Processor Configuration

A 32-bit MicroBlaze processor implementation was used, configured for optimal performance with the following specifications:

* **Performance Optimization:** The processor was tuned with performance implementation optimizations enabled, allowing for efficient execution of cryptographic algorithms.
* **System Clocking:** The DDR2 memory controller ran at 200 MHz to match the required data rates for handling large test cases.The remainder of the system, including the MicroBlaze processor and peripherals, was clocked at 100 MHz to ensure stable operation while balancing resource usage.
* **Instruction and Data Caches:** Both instruction and data caches were enabled to improve memory access speeds, reducing delays caused by frequent memory fetches during the processing of hash functions.
* **Floating-Point Unit (FPU):** The FPU was disabled to avoid introducing extraneous computation overhead. Since SHA algorithms primarily rely on integer operations, disabling the FPU ensured more accurate benchmarking results by focusing solely on the cryptographic workloads.

## Memory Subsystem

The memory system below outlines the necessary configurations for the MicroBlaze to support the testing requirements.

* **Initial BRAM Configuration:** Initially, Block RAM (BRAM) was used as the sole memory resource due to its proximity to the processor and high-speed access.However, BRAM's limited size was insufficient for handling large test cases, such as the 1 MB input for performance testing.
* **DDR2 SDRAM Integration:** To accommodate larger input sizes, DDR2 SDRAM was added to the system. The DDR2 memory was configured to run at 200 MHz and allocated its full capacity of 0x8000000 bytes (128 MB), ensuring enough space for the largest test inputs.

Although DDR2 provided the necessary storage for large inputs, its latency and access time negatively impacted overall performance, which will be discussed in the results section.

## Peripherals and Interfaces

The following peripherals were essential for obtaining benchmark metrics, as the MicroBlaze processor operates without an operating system and therefore lacks built-in tools for measuring performance and facilitating software interaction. These peripherals provided the necessary functionality to measure execution times, manage data inputs and outputs, and display results effectively.

* **AXI UARTLite:** The UARTLite IP core was included to display benchmarking results, including execution time, throughput, and hash outputs. UART also served as the interface for functionality tests, allowing user input and real-time observation of system behavior.
* **AXI Timer:** The AXI Timer was used to accurately measure execution time for each hash function, capturing the number of clock cycles required to process test inputs.
* **GPIO (General Purpose Input/Output):** GPIO peripherals were configured for debugging purposes, facilitating real-time status monitoring during testing.

# Software development and testing

The software implementation of this project was developed using Xilinx Vitis, a unified platform for embedded system design. The hardware configuration, exported from Vivado, served as the foundation for creating a platform to execute and benchmark the SHA-2 and SHA-3 cryptographic algorithms. The process involved testing, optimizing, and configuring the system for performance while adhering to the constraints of an embedded environment

## Open-Source Algorithm Selection

The SHA-2 and SHA-3 implementations used in this project were sourced from open-source GitHub repositories:

* SHA-2: [SHA-2 GitHub Repository](https://github.com/ogay/sha2) [3]
* SHA-3: [SHA3IUF GitHub Repository](https://github.com/brainhub/SHA3IUF) [4]

Open-source software was chosen for several reasons:

* **Proven Functionality**: These repositories provided tested and reliable implementations, ensuring correctness while minimizing development time.
* **Adaptability**: The open-source nature allowed modifications to adapt the algorithms to the MicroBlaze processor and Vitis environment.
* **Transparency**: Using open-source code ensures reproducibility, enabling others to replicate or build upon the results of this study.

## Memory Configuration

The SHA algorithms were first tested using only Block RAM (BRAM), a fast, low-latency memory resource directly connected to the MicroBlaze processor. Compiler optimization for size (-oS) was enabled during these tests to minimize memory usage. This configuration worked effectively with smaller datasets (e.g., functional tests and 1 KB performance tests). However, when testing with larger inputs (1 MB), the program froze, indicating insufficient memory for the processing and storage requirements. To handle larger datasets, the hardware design was updated to include DDR2 SDRAM. This required adjustments to the memory allocation in the Vitis linker script:

* **Heap Memory:** The heap size was significantly increased and mapped to the DDR2 region to provide sufficient space for dynamic memory allocation required by large inputs.
* **Stack Memory:** The stack remained in BRAM, where its proximity to the processor ensured fast access for temporary storage and function calls.
* **Program Code:** The program's code and data sections were also placed in BRAM to leverage its speed for critical operations.

This memory configuration balanced the need for speed (via BRAM) and capacity (via DDR2) while ensuring the system could handle all test cases without freezing.  
Open-Source Algorithm Selection

## Program Structure

The software implementation for testing SHA-2 and SHA-3 was developed using separate programs with similar code structures and lengths. This approach ensured that both algorithms were evaluated under consistent conditions while accommodating differences in their implementation details. The programs were designed to benchmark the 256-bit, 384-bit, and 512-bit variants of each algorithm, providing a direct comparison of their performance.

The AXI Timer was configured to accurately measure execution time for hash computations. This measurement formed the basis for calculating key performance metrics:

* **Execution Time**: The total time taken to compute the hash for each input size.
* **Throughput**: The rate of data processed, calculated in bytes per second.
* **Energy Consumption**: Estimated using the execution time and the power consumption of the system.

Both programs implemented three variants for their respective algorithms:

* SHA-2: SHA-256, SHA-384, and SHA-512.
* SHA-3: SHA3-256, SHA3-384, and SHA3-512.

Each variant was implemented as a separate function, using the GitHub repositories as a foundation:

* Input data was processed and converted into a readable hexadecimal hash output, displayed via the UART interface for verification and logging.

The program includes a testing framework with three distinct test types:

1. **Functional Tests**:

* Verifies the correctness of the hash output for user-provided inputs.
* Allows interactive testing via the UART interface.

1. **Edge Case Tests**:

* Handles boundary conditions, such as empty input, ensuring robustness and correctness.

1. **Performance Tests**:

* Benchmarks the algorithms using fixed input sizes of 1 KB and 1 MB to evaluate execution time, throughput, and energy consumption.

The correctness of the SHA-2 and SHA-3 implementations was validated by cross-checking the hash outputs with an online hashing tool [5]. This ensured that the results produced by the software matched the expected hash values for various input cases, including functional and edge case tests.

Dynamic memory allocation was employed to handle varying input sizes.

* The heap was specifically configured in DDR2 SDRAM to provide sufficient space for large inputs, such as the 1 MB test case.
* The stack and program code were kept in BRAM to leverage its high-speed access for temporary storage and critical operations.

By using identical programs for both SHA-2 and SHA-3 testing, the project ensured a consistent methodology:

* The testing framework dynamically adjusted to call the appropriate functions for each algorithm.
* The same performance metrics were collected and calculated for direct comparisons between SHA-2 and SHA-3 across their 256-bit, 384-bit, and 512-bit variants.

Despite being separate implementations, the programs shared a consistent structure, which includes the benefits.

* Similar modular design for maintainability and clarity.
* Identical metrics collection methods for fair and direct comparisons.
* Similar handling of inputs, outputs, and memory management to ensure parity between the two testing setups.

# Performance Results

This section presents and analyzes the performance results for SHA-2 and SHA-3 algorithms based on their 256-bit, 384-bit, and 512-bit variants. Key performance metrics, including execution time, throughput, and energy consumption, were obtained by running edge case, functional, and performance tests (1 KB and 1 MB inputs). The results highlight the trade-offs between the two cryptographic standards.

As discussed earlier, BRAM was initially used for its proximity to the MicroBlaze processor, with the expectation of achieving faster and more efficient results. However, testing with various compiler optimization flags revealed that BRAM’s limited capacity could not accommodate the larger input buffers required to fully demonstrate the performance of the SHA-2 and SHA-3 algorithms. This limitation restricted the ability to showcase their true effectiveness, particularly for larger datasets.

The results presented in the following tables were obtained using a MicroBlaze platform configured with DDR2 SRAM, which provided the necessary memory support for handling large inputs. Both SHA-2 and SHA-3 applications shared consistent code structure, sizes, and memory allocations to ensure a fair and accurate comparison of their performance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SHA 2-256 | | | |
| **Test Type** | **Elapsed Cycles** | **Execution Time (µs)** | **Throughput (bytes/s)** | **Energy Consumed (µJ)** |
| Edge Test (Empty Input) | 8332655 | 83326 | 0 | 92575 |
| Performance Test (1KB) | 8332791 | 83327 | 12288 | 92576 |
| Performance Test (1MB) | 550589815 | 5505898 | 109 | 6117052 |
| Functional Test (1234567890) | 8332615 | 83326 | 120 | 92575 |
|  | **SHA 2-384** | | | |
| **Test Type** | **Elapsed Cycles** | **Execution Time (µs)** | **Throughput (bytes/s)** | **Energy Consumed (µJ)** |
| Edge Test (Empty Input) | 11665804 | 116658 | 0 | 116658 |
| Performance Test (1KB) | 11665920 | 116659 | 8777 | 129608 |
| Performance Test (1MB) | 686904071 | 6869040 | 87 | 7631503 |
| Functional Test (1234567890) | 11665743 | 116657 | 85 | 129605 |
|  | **SHA 2-512** | | | |
| **Test Type** | **Elapsed Cycles** | **Execution Time (µs)** | **Throughput (bytes/s)** | **Energy Consumed (µJ)** |
| Edge Test (Empty Input) | 14998762 | 149987 | 0 | 166635 |
| Performance Test (1KB) | 14998927 | 149989 | 6827 | 166637 |
| Performance Test (1MB) | 690268252 | 6902682 | 87 | 7668879 |
| Functional Test (1234567890) | 14998773 | 149987 | 66 | 166635 |

1. *SHA 2 Performance Results*

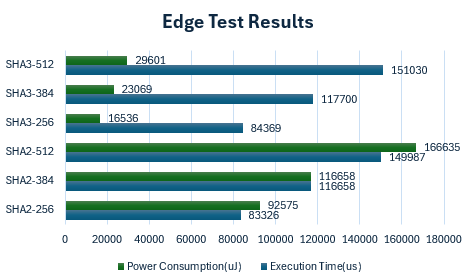
SHA-2 demonstrated strong performance across all tests, particularly for smaller datasets. For instance, SHA-256 required only 83,327 µs to process 1 KB of data and maintained a throughput of 12,288 bytes/second. As expected, the execution time increased significantly for larger inputs, with 1 MB requiring 550,5898 µs. SHA-384 and SHA-512 variants exhibited slightly higher execution times and lower throughput due to their more complex internal operations and larger digest sizes. Energy consumption mirrored the execution times, with the 1 MB performance test for SHA-256 consuming 6,117,052 µJ. Overall, SHA-2's lower computational complexity and efficient memory usage make it well-suited for constrained systems, especially for small to moderate input sizes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SHA 3-256 | | | |
| **Test Type** | **Elapsed Cycles** | **Execution Time (µs)** | **Throughput (bytes/s)** | **Energy Consumed (µJ)** |
| Edge Test (Empty Input) | 8436960 | 84369 | 0 | 16536 |
| Performance Test (1KB) | 11801316 | 118013 | 8677 | 23130 |
| Performance Test (1MB) | 509016809 | 5090168 | 118 | 997672 |
| Functional Test (1234567890) | 8436959 | 84369 | 118 | 16536 |
|  | **SHA 3-384** | | | |
| **Test Type** | **Elapsed Cycles** | **Execution Time (µs)** | **Throughput (bytes/s)** | **Energy Consumed (µJ)** |
| Edge Test (Empty Input) | 11770011 | 117700 | 0 | 23069 |
| Performance Test (1KB) | 16394718 | 163947 | 6245 | 32133 |
| Performance Test (1MB) | 1966725523 | 19667255 | 30 | 3854781 |
| Functional Test (1234567890) | 11770032 | 117700 | 84 | 23069 |
|  | **SHA 3-512** | | | |
| **Test Type** | **Elapsed Cycles** | **Execution Time (µs)** | **Throughput (bytes/s)** | **Energy Consumed (µJ)** |
| Edge Test (Empty Input) | 15103082 | 151030 | 0 | 29601 |
| Performance Test (1KB) | 22810932 | 228109 | 4489 | 44709 |
| Performance Test (1MB) | 422571709 | 4225717 | 142 | 828240 |
| Functional Test (1234567890) | 15103077 | 151030 | 66 | 29601 |

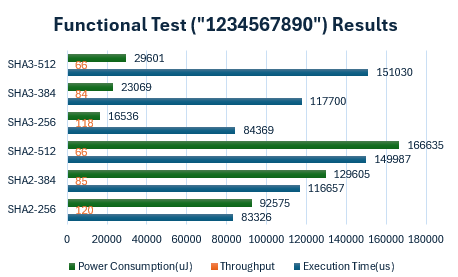
1. *SHA 3 Performance Results*

In comparison, SHA-3 exhibited longer execution times and lower throughput across all test cases, owing to its sponge construction, which is inherently more computationally intensive. For example, SHA3-256 required 11,801,316 µs to process 1 KB of data, achieving a throughput of only 8,677 bytes/second. For larger inputs (1 MB), SHA3-256 required over 509,016,809 µs, with throughput dropping to 118 bytes/second. SHA3-384 and SHA3-512 showed similar trends, with slightly worse performance metrics due to their larger sizes. Energy consumption for SHA-3 was also higher across all tests, with SHA3-512 consuming 828,240 µJ for the 1 MB test. These results highlight that while SHA-3 provides enhanced security, it does so at the cost of efficiency, making it less suitable for highly constrained environments.

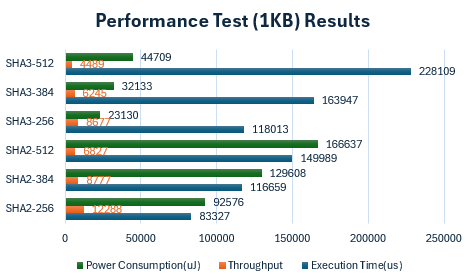
## Comparing SHA2 and SHA3



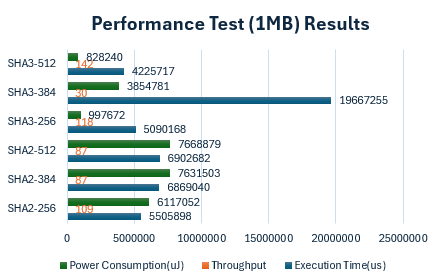
1. *Edge Test Comparisons*



1. *Functional Test Comparisons*



1. *Performance Test, 1KB*



1. Performance Test, 1MB

When comparing SHA-2 and SHA-3, SHA-2 consistently outperformed SHA-3 across all metrics. For smaller datasets (1 KB), SHA-2 demonstrated faster execution times and significantly higher throughput. For example, SHA-256 achieved 12,288 bytes/second, while SHA3-256 managed only 8,677 bytes/second. The energy consumption for SHA-2 was also lower across the board, making it more suitable for power-constrained systems.

For larger datasets (1 MB), both algorithms experienced reduced throughput due to memory access overhead and computational complexity. However, SHA-2 retained a significant advantage, with SHA-256 maintaining a throughput of 109 bytes/second, compared to SHA3-256’s 118 bytes/second, which was achieved at much higher energy consumption levels. These findings reinforce SHA-2’s efficiency in constrained systems, while SHA-3's performance lag reflects the cost of its enhanced security features.

## Comparison to Related Works

The results of this study align partially with findings in related works but also highlight unique insights. For instance, studies focusing on hardware-accelerated implementations, such as those in the *Journal of Cryptographic Engineering*, report that SHA-3 often outperforms SHA-2 in throughput due to its suitability for hardware optimization [1]. However, this study’s software-only approach reveals the opposite trend, where SHA-2 consistently outperforms SHA-3 due to the computational overhead introduced by the sponge construction.

Furthermore, the Keccak team notes that SHA-3 is slower in software environments, especially without optimization [2]. This observation aligns with our results, where SHA-3 demonstrated lower throughput and higher energy consumption compared to SHA-2. Unlike previous works focusing on hardware acceleration or software optimizations, this study highlights the trade-offs in a software-only, resource-constrained environment, offering a realistic reference for embedded system designers.

The execution time for the 1 MB test of SHA3-384 appears anomalous when compared to other SHA-3 and SHA-2 variants. Specifically, SHA3-384 recorded an execution time of 19667255 µs, which is significantly longer than SHA3-256 (5090168 µs) and SHA3-512 (4225717 µs). This is unexpected, as SHA3-384 is generally more computationally intensive than SHA3-256 but less than SHA3-512, suggesting its execution time should fall between the two. For comparison, the SHA-2 family exhibits a more consistent trend, with SHA2-256 executing in 5505898 µs, SHA2-384 in 6869040 µs, and SHA2-512 in 6902682 µs. The large discrepancy in SHA3-384's execution time for the 1 MB test could indicate an anomaly, possibly caused by measurement inaccuracies, memory access issues, or differences in algorithm implementation. Further investigation is needed to confirm whether this result is genuinely anomalous or reflective of specific processing characteristics of SHA3-384.

# Insight and next steps

This project benchmarked the performance of SHA-2 and SHA-3 cryptographic hash functions in a constrained software-only environment using the MicroBlaze softcore processor. The results highlighted significant trade-offs between computational efficiency and cryptographic strength, providing a valuable reference for embedded system designers:

* **SHA-2** demonstrated superior performance in terms of execution time, throughput, and energy consumption, making it the preferred choice for resource-constrained environments.
* **SHA-3**, while slower and more computationally intensive, offers enhanced cryptographic resilience, making it suitable for high-security applications where performance is secondary.

Although the project successfully achieved its goals, there are multiple avenues for further improvements and extended research.

## Future Improvements and Extensions

While this project provided valuable insights into the performance of SHA-2 and SHA-3 in a constrained, software-only environment, several opportunities for further exploration could enhance the findings. Testing the algorithms on alternative softcore processors, such as Intel Nios II or an open-source RISC-V core, could provide a broader perspective on how architectural differences influence performance. Extending the range of input sizes and testing with randomized or real-world datasets, such as logs or sensor data, would simulate practical applications and offer a deeper understanding of how the algorithms scale across different scenarios. Additionally, analyzing the effects of compiler optimizations, such as speed-focused flags like -O3, could provide actionable recommendations for developers working with constrained systems.

Another significant extension involves incorporating hardware-accelerated implementations of SHA-2 and SHA-3 using FPGA logic. Comparing software-only and hardware-optimized versions would help quantify the potential performance gains achievable with hardware acceleration. Finally, more precise power profiling using hardware tools or on-chip power monitors would refine energy consumption estimates, allowing for a more detailed analysis of algorithm efficiency in resource-limited environments. These improvements would expand the applicability of this study and provide a comprehensive reference for selecting and optimizing cryptographic algorithms in embedded systems.

##### References

1. X. Zhang and Y. Wang, "Efficient Hardware Acceleration for SHA-3 on FPGAs," *Journal of Cryptographic Engineering*, vol. 13, no. 2, pp. 105–116, Aug. 2023. [Online]. Available: [link.springer.com](https://link.springer.com/article/10.1007/s13389-023-00334-0)
2. Keccak Team, "Is SHA-3 slow?" *Keccak Team Blog*. [Online]. Available: keccak.team
3. Ogay, "SHA-2 Implementation," GitHub, [Online]. Available: https://github.com/ogay/sha2/tree/master. [Accessed: Dec. 2, 2024].
4. Brainhub, "SHA3IUF: SHA-3 Incremental Update Function," GitHub, [Online]. Available: <https://github.com/brainhub/SHA3IUF/tree/master>. [Accessed: Dec. 2,
5. emn178, "Online Tools for Cryptographic Algorithms," [Online]. Available: <https://emn178.github.io/online-tools/>. [Accessed: Dec. 2, 2024]