O HARSHAVARDHAN, et al , International Journal of Computers Electrical and Advanced Communication Engineering [IJCEACE]™
Volume 1 , Issue 6, pp: 23 - 28 , SEPTEMBER 2014.

ANALYSIS OF BLAKE ALGORITHM FOR CACHE PERFORMANCE

O. Harshavardhan 1*, Prof. Ch. Srinivasa Kumar2*
1. II.M.Tech (VLSI), Dept of ECE, AM Reddy Memorial College of Engineering & Technology, Petlurivaripalem.
2. Prof , Dept. of ECE, AM Reddy Memorial College of Engineering & Technology, Petlurivaripalem.

ABSTRACT

This paper presents a novel data security for global competition is currently it taking a hash function it will become a novel standard in the field of cryptography. This competitions was announced by NIST (National Institute of Standard and Technology) is to find previous Secure Hash Algorithms (SHA) standard is SHA1, and SHA2. These submissions have narrowed to their cryptography security, and practical consideration, designing features, and speedup the communication. These five finalized in the BLAKE hash function developed a team in Switzerland. This paper describes the functional details and strengths of given insight implementation and performance. In 2012 BLAKE algorithm may be selected by NIST for published new standard SHA-3.

I. INTRODUCTION

Before the BLAKE algorithm in discussed in detail this section describes hash functions in general as well as their cryptographic applications By definition of hash functions are transformations which produce a numeric "hash" or "digest" of a predefined length from an input message of arbitrary length (both typically measured in bits). These are often used in digital systems for checking the integrity of transmitted data, quickly looking up messages in a database, or mapping messages to table indexes. In the field of cryptography, hashes have further applications, including the creation of authentication codes digitally signing documents, and generation of statistically random data streams. A strong hashing algorithm can generate a hash from data quickly, yet is computationally infeasible for anyone to determine the original input from the hash itself. It should also be infeasible to find two or more messages that result in the same hash, called a collision. Finally, any change in the input message (regardless of significance) should produce a drastic and seemingly-random change in the output hash. The algorithm is considered weak if it does not meet these requirements. Algorithm has been "broken", as researchers found methods to generate different messages which map to identical hashes.) These qualities distinguish cryptographic hash functions from general purpose hash functions, at the cost of complexity, computation time, and output size. For the purposes of this document, any mention of hash functions refers to cryptographic hash functions specifically.

The National Institute of Standards and Technology (NIST) have established a set of standardized hash functions called the Secure Hash Algorithm (SHA). The first

version called SHA-0 was published in 1993. Its design was influenced by cryptographer Ronald L. Rivest and his work on the MD4 and MD5 hashes. Since its 1995 revision as SHA-1 it has generally been accepted as the worldwide standard for digital data authentication and signing. A newer function called SHA- 2 was published in 2001 to address some potential weaknesses, but it never surpassed the original in widespread adoption. Despite attempts, neither has been broken as of 2011. NIST announced a global competition to find a new SHA function in 2007 and submissions were accepted for approximately one year. The algorithms are being analyzed and narrowed down through elimination rounds, based on the security, performance, and design of each function. The final round of candidates was announced in 2010, with public analysis encouraged. In 2012, a winner will be announced and the algorithm will become the official SHA-3 standard for future cryptographic applications. One of the five finalists is the BLAKE function, whose design and performance will be the focus of this paper.

II. IMPLEMENTATION

The BLAKE algorithm already explained and analyzed, this section describes some methods of practical implementation. The focus of this section is hash performance, rather than its security or hardware requirements.

2.1. BLAKE SOFTWARE

The BLAKE designers provide various software implementations of the hash in the C programming language. These range from easy-to-understand versions that closely follow the algorithm description, to highly optimized versions for specific key sizes and architectures. The official BLAKE website also links to various approved ports: Perl, PHP, Java script etc. The "lightweight" 512-bit version will be briefly examined here. Since it is compact and designed for one specific variant, it requires minimal software and hardware resources, and can be run on many platforms. It also includes one Built-In Self Test (BIST) which hashes two hard-coded message and compares the result against the known correct hashes. The hash function is called as blake512_hash (digest, data, length) where digest is a pointer to a 512-bit output buffer, data is a pointer to the input message, and length is the message length (in bytes here, rather than bits). The function creates a 4x4 state matrix and calls three subroutines: one to initialize it, one to process it, and one to finalize it. These correspond to the main steps described the BLAKE-256 description section. The blake512_update function is responsible for padding the input message, segmenting it into 1024-bit blocks, and iteratively executing the compress function. This is implemented as blake512_compress, which executes 16 rounds, each consisting of eight G transformations using arithmetic macros. Note that this software implementation executes all eight sequentially; it does not utilize the parallelization technique described previously. The hash result is written to digest and is typically displayed as a 128 character hexadecimal string. By default, the program hashes a 144 byte (1152 bit) message, but this can easily be adapted for testing different lengths. A timer will be added to the code for measuring hash performance and recording various benchmarks.

2.2. PC PERFORMANCE

The BLAKE specification includes some performance tests of software implementations on microcontrollers and consumer Central Processing Units (CPUs). However these tests were run on the BLAKE-32 and BLAKE-64 algorithms earlier versions of BLAKE-256 and BLAKE-512 which include fewer rounds per compression. (The recommended round counts have been increased since the initial 2008 submission.) A summary of the early benchmarks on an Intel 64-bit 2.4 GHz Core 2 Duo processor are shown in Table I. Note that the 64-bit version is natively faster than the 32-bit version, the number of hash clock cycles per message byte decreases dramatically as the input grows and the results for small message lengths (here, 10 bytes) are approximations since small execution variances affected the cycles/byte ratio significantly.

More recent experiments use the updated BLAKE variants. The third-party ECRYPT benchmarking project compared the five SHA-3 finalists on a large number of computer systems across a wide range of message sizes. Their results are released into the public domain and are available at the project's website. The data is also shown graphically in Fig. 1 illustrating the increase in efficiency (decrease in cycles/byte) as message size increases. The ECRYPT benchmarks also serve as an informative comparison between the five SHA-3 finalists: BLAKE, JH, Skein, Keccak, and Grøstl. In general, BLAKE was typically one of the fastest algorithms (perhaps due to its small number of rounds), comparable with Skein.

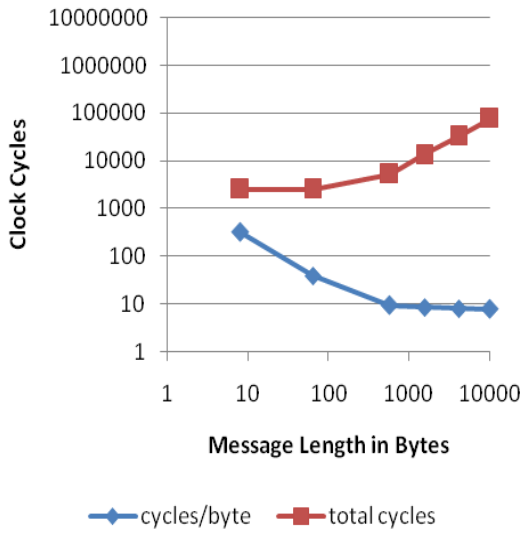

Fig. 1. ECRYPT benchmarks for BLAKE-512 on the Core 2 Duo.

The SHA-3 candidates have also been analyzed on "exotic" architectures. Researchers at the Laboratory for Cryptographic Algorithms have tested the performance of all second round candidates on two specialized processors: the Cell Broadband Engine developed by Sony, Toshiba, and IBM, and graphical processing units (GPUs) developed by NVIDIA. Both are high-performance, multi-core processors that utilize Single Instruction Multiple Data (SIMD) and Single Instruction Multiple Threads (SIMT) paradigms respectively. The GPU is typically used for heavy graphics processing, while the Cell processor is currently used, for example, in the high-end PlayStation 3 video game system.

These tests used different types of metrics for measuring performance. The candidates were divided into AES inspired (Advanced Encryption Standard, NIST's current block cipher standard) and non-AES inspired hashes (under which BLAKE is categorized). The tests of the latter group were focused on their compression

functions, rather than the entire algorithms. This does not account for setup time finalization time, or memory copying time, except for the copying of chain values (such as hi, for BLAKE). The researchers also focused on the number of times instructions were called, rather than explicit clock cycles. While some clock-cycles-per-message-byte estimations are extrapolated from this data, the relationship is not directly proportional. The variant of BLAKE tested was BLAKE-32, the earlier version of BLAKE-256 which executes only ten compression rounds. Therefore the performance is expected to be greater than that of the BLAKE-512 tests above. With appropriate optimizations and multi-threading, the compression algorithms were executed on the two platforms and their speeds were monitored. On the Cell processor, BLAKE-32 was the third fastest of the eleven candidates, measured at 5.0 cycles/byte. On the NVIDIA GPU, BLAKE-32 was tied with BMW-256 as the fastest, measured at 0.27 cycles/byte with a theoretical peak rate of 0.13 cycles per byte. These findings are important as the winning SHA-3 hash will surely be implemented on a variety of modern, specialized platforms.

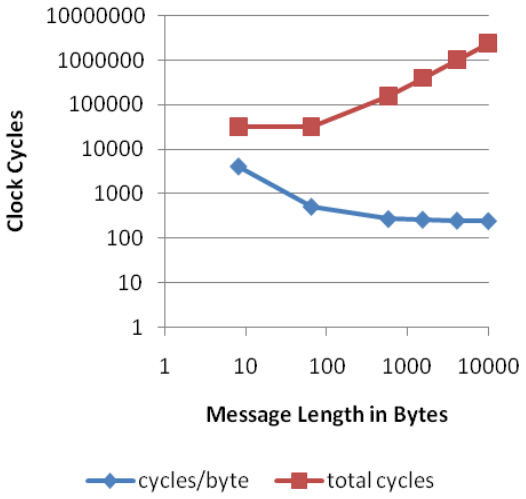

Fig. 2. Benchmarks for BLAKE-512 on the FPGA's PowerPC.

2.3. FPGA PERFORMANCE

For further insight, the lightweight version of BLAKE-512 was tested on a Field Programmable Gate Array (FPGA) system. The FPGA used is a Xilinx Virtex-5, with an integrated PowerPC microprocessor. The hardware platform (designed in Xilinx Platform Studio [XPS] for the ML507 evaluation board) is kept to a minimum: a single processor, one hardware timer, one Universal Asynchronous Receiver/Transmitter (UART) port for output. A 32 KB data/instruction cache is also enabled for greater performance. Performance tests were run on the embedded processor, similar to the PC tests run by ECRYPT. There are some clear reasons for this: the ECRYPT software was compiled with loop unrolling and similar optimizations, and executed on a multi-core processor that natively operates on 64-bit words. The single-core embedded processor operates on 32-bit words, so handling 64-bit values is slower, and its code was compiled with no explicit optimization. There are also some ambiguous differences between tests: the ECRYPT team's memory access/caching system is not known, and they likely used different source code, perhaps a platform optimized version. There are some optimizations suggested for Intel platforms specifically, such as using the Core 2's native bit-rotate instruction in place of four previous instructions. Finally, it should be mentioned that the content of the message being hashed has no effect on performance, since the transformations consist of primitive bitwise operations. All input words are processed without computational bias of any nature.

III. SOFTWARE PERFORMANCE AND HARDWARE ACCELERATION

The FPGA performance of BLAKE could be greatly increased by implementing some of the entire algorithm as hardware components. The main improvement would be changing the eight G transformations to the two parallelized "column step" and "diagonal step" operations described in the Algorithm section. This would have a major effect on total execution time, since it is computed 16 (or 14) times per compression, which itself is computed for every segment of the input stream. Furthermore, the accessing and processing of data is naturally faster on hardware than in software. Many words can loaded into parallel registers simultaneously, as opposed to the sequential loading of software words. Arithmetic operations such as XOR and addition can be executed in a single clock cycle, whereas software might require four or more. Diffusion operations such as permutation and bit-rotation can be directly implemented by the routing of data wires. The BLAKE website offers a few VHSIC Hardware Description Language (VHDL) frameworks for hardware implementation. Research into their performance is currently underway, and some tests predict ideal throughput of about 2 clock cycles per message byte when hashing very long messages. This is feasible since a single compression processes 64 (or 128) bytes, and since the cycles required for initialization and finalization become negligible as the input message grows.

IV. CONCLUSION

This paper introduced a BLAKE hash function of finalists in the SHA-3 hash completion. The cryptographic structure algorithms have been examined and leading general thing for communication security. The simulation of both software and hardware tests on real time systems have been performed optimal performance in practical applications, compared to existing hash functions. In future you can the BLAKE algorithm and its potential worldwide adoption will be determined in 2012, and the NIST announced new SHA-3 standard. This technique will ensure an optimal balance between resources, performance, security, and efficiency.

REFERENCES

[1] ARM. ARM-7TDMI Data Sheet. Advanced RISC Machines Ltd., August 1995.

[2] D. Burger; T. M. Austin, "The Simplescalar Tool Set, version 2.0". ACM SIGARCH Computer Architecture News, 25(3):13-25, June 1997.

[3] J. L. Hennessy; D. A. Patterson, "Computer Architecture: A Quantitative Approach". Morgan Kaufmann Publishers, San Francisco, 4th edition, 2006, 704p.

[4] W. R. A. Dias. "Arquitetura PDCCM em Hardware para Compressão/Descompressão de Instruções em Sistemas Embarcados". Dissertação de Mestrado, Departamento de Ciência da Computação da UFAM, Brasil Abril de 2009. 152 p.

[5] J. Yiu. "The Definitive Guide to the ARM Cortex-M3". Newnes, 2nd edition, 2009, 479p.

[6] M. Hai-Feng, Y. Nian-Min, F. Hong-Bo, "Cache Performance Simulation and Analysis under SimpleScalar Platform", International Conference on New Trends

in Information and Service Science (NISS 2009). Beijing, China, July 2009.
[7] J. W. Bos and D. Stefan, "Performance analysis of the SHA-3 candidates on exotic multi-core architectures," 2010.
[8] S. Neves, "ChaCha implementation," 2009, http://eden.dei.uc.pt/ sneves/ chacha/chacha.html.
[9] M. Knezevic, K. Kobayashi, J. Ikegami, S. Matsuo, A. Satoh, U. Kocabas, J. Fan, T. Katashita, T. Sugawara, K. Sakiyama, I. Verbauwhede, K. Ohta, N. Homma, and T. Aoki, "Fair and consistent hardware evaluation of fourteen round two SHA-3 candidates," April 2011.