Journal of Southeast University (English Edition) Vol. 30, No. 1, pp. 13 - 16 Mar. 2014 ISSN 1003—7985

Optimized SHA-1 hash function implemented on FPGA

Xue Ye Hu Aiqun

(Research Center of Information Security, Southeast University, Nanjing 211189, China)

Abstract: In order to meet the needs of higher operation speed and lower energy consumption, an optimized SHA-1 algorithm is proposed. It combines two methods: loop-unfolding and pre-processing. In the process, intermediate variables are introduced in the iterations and pre-calculated, so that the original single-threading operation can perform in a multi-threading way. This optimized algorithm exploits parallelism to shorten the critical path for hash operations. And the cycles of the original algorithm are reduced from 80 to 41, which greatly improves the operation speed. Therefore, the shortened iterations of the optimized design require a smaller amount of hardware resource, thus achieving a lower energy consumption. The optimized algorithm is implemented on FPGA (field programmable gate array). It can achieve a throughput rate of 1.2 Gbit/s with the maximum clock frequency of 91 MHz, reaching a fair balance between operation speed and throughput rate. The simulation results show that, compared with other optimized SHA-1 algorithms, this algorithm obtains higher operation speed and throughput rate without compromising the security of the original SHA-1 algorithm.

Key words: SHA-1; hash function; loop unfolding; pre-processing; FPGA

doi: 10.3969/j.issn.1003 - 7985.2014.01.003

As critical components in modern cryptology, hash functions have a wide range of applications in message certification and digital signature[1], because they do not require the processed data to be retrieved. A cryptographic hash function must have the following properties: pre-image resistance (related to that of one-way function), second pre-image resistance, collision resistance, as well as being able to withstand all known types of cryptanalytic attack.

The SHA-1 algorithm is one of the most popular hash algorithms. In 2005, researchers found attacks on SHA-1, suggesting that the algorithm may not be secure enough for ongoing use. Wang et al.[2] announced an attack that can find collisions in the full version of SHA-1, requiring fewer than 2^{69} operations while a brute-force search will require 2^{80} operations. Their analysis is built upon the original differential attack on SHA-0, the near collision attack on SHA-0, the multiblock collision techniques, as well as the message modification techniques used in the collision search attack on MD5. They exploit the following two weaknesses: one is that the file preprocessing step is not complicated enough; the other is that certain math operations in the first 20 rounds have unexpected security problems. Although other data can be found by Wang's collision attack, the content of the data is uncertain, which is most likely to be an unreadable code. In digital signature, people can identify that the data is damaged and there will be no loss. Therefore, in many cases, the SHA-1 algorithm is still secure.

This paper focuses on high-throughput design for the SHA-1. Techniques such as loop unfolding, pre-processing, multi-input adding based on a carry-save adder and pipeline have been proposed[3-7] to achieve high throughput rate, high speed operation, and low cost consumption, but only one or two methods have been applied up to date. Here, the first two methods are integrated and implemented on FPGA to obtain optimized results. The improvement is based on unfolding transformation performing two Hash operations in a cycle. The critical path of a hash operation is short due to the pre-processing of the coefficients and the algorithm parallelism.

In this paper, the conventional SHA-1 algorithm is explained, then the proposed design is presented. Afterwards, the hardware implementation, simulation results, as well as the performance comparison with other works are displayed. Finally, conclusions are made concerning this new design.

1 SHA-1 Algorithm

The SHA-1 algorithm is one of the most popular hash algorithms. It is designed by the U.S. National Security Agency and issued as a federal information processing standard. The input message of the SHA-1 algorithm has a maximum length of less than 2^{64} bits, which generates a 160-bit message digest.

The conventional SHA-1 structure is shown in Fig. 1. The required input message of SHA-1 is a multiple of 512 bits, so it is necessary that the original message is padded. The process of padding is as follows: First, one bit "1" and then $0 \leqslant k < 512$ bits "0" are appended at the end of the massage, so that the length of the message (in bits) is congruent to 448 (mod 512). Afterwards, the

Received 2013-05-06.

Biographies: Xue Ye (1989—), female, graduate; Hu Aiqun (corresponding author), male, doctor, professor, aqhu@seu.edu.cn.

Foundation items: The Project of Wireless Intelligence Terminal Inspection Services (No. 6704000084), the Special Program of the National Development and Reform Committee.

Citation: Xue Ye, Hu Aiqun. Optimized SHA-1 hash function implemented on FPGA[J]. Journal of Southeast University (English Edition), 2014, 30(1): 13 - 16. [doi: 10.3969/j.issn.1003 - 7985.2014.01.003]

length of the message, in bits, is appended as a 64-bit big-endian integer. After padding, the input message can be divided into one or multiple 512-bit blocks. Then, each block is expanded into 80 32-bit words W_t, one 32-bit word for each round of the SHA-1 processing and each block goes through 4 rounds of hash operations with each round consisting of 20 calculation units. The final result is obtained by adding the performing output with the previous hash code.

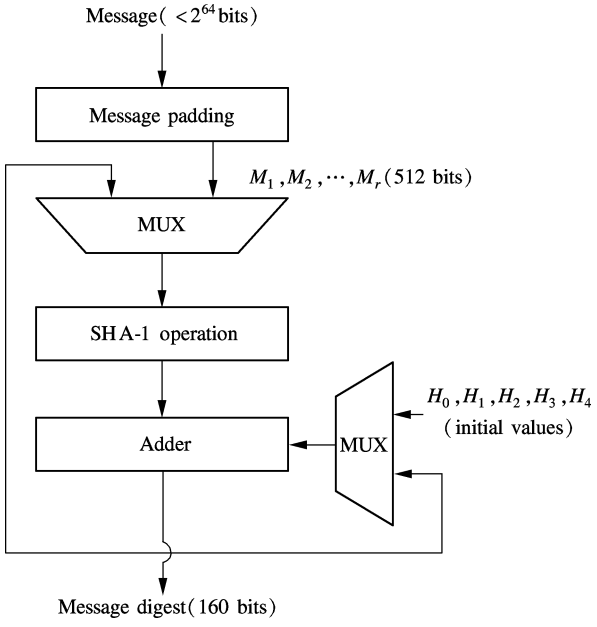

Fig. 1 Conventional SHA-1 structure

The iterative process of the hash operation is shown in Fig. 2. There are both logical operations and additions in each round of calculation, and all the calculations are bitwise, such as rotations to the left ($\ll$) and nonlinear function f_t. The five 32-bit data (A_t, B_t, C_t, D_t, E_t) are calculated from the values obtained in the previous cycle by using the required nonlinear operations. The initial values of the five 160-bit hash variables H_0, H_1, H_2, H_3, H_4 are predefined ($H_0 = 0$x67452301, $H_1 = 0$xEFCDAB89, $H_2 = 0$x98BADCFE, $H_3 = 0$x10325476, $H_4 = 0$xC3D2E1F0), and remain constant throughout the calculations. While at the beginning of each data block calculation, the values of the variables A, B, C, D, E are determined by those hash values. After all the data blocks are computed, the final hash value is the output digest message.

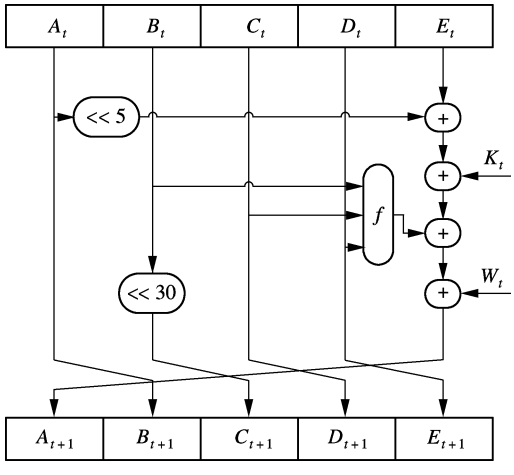

Fig. 2 Computational mode of hash operation

The 32-bit data word W_t ($t = 0, 1, \ldots, 79$) expanding from the input blocks is as

$$W_t = \begin{cases} M_t & 0 \leqslant t \leqslant 15 \\ \mathrm{rot}L^1(W_{t-3} \otimes W_{t-8} \otimes W_{t-14} \otimes W_{t-16}) & 16 \leqslant t \leqslant 79 \end{cases} \tag{1}$$

The value of the constant K_t and the performance of the nonlinear function f are determined by one of the 80 rounds being executed, which are listed in Tab. 1.

Tab. 1 Constant K_t and nonlinear function f

Rounds	Function	K_t
0 to 19	$(B \wedge C) \oplus (\bar{B} \wedge D)$	0x5A827999
20 to 39	$B \oplus C \oplus D$	0x6ED9EBA1
40 to 59	$(B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D)$	0x8F1BBCDC
60 to 79	$B \oplus C \oplus D$	0xCA62C1D6

Notes: $\wedge$ represents the bitwise AND operation and $\oplus$ represents the bitwise XOR operation.

2 Proposed SHA-1 Structure

The time consumption in the conventional SHA-1 algorithm is determined by the combined delay in each hash operation. Therefore, the overall consumption can be reduced significantly by cutting down the number of processing rounds.

The proposed method performs two hash operations in one cycle. The coefficient pre-computation of hash function and parallelism are available in two independent hash operation blocks. The deducted hash operations are as

$$\left.\begin{aligned} a_t &= \mathrm{rot}L^5\{\mathrm{Rot}L^5(a_{t-2}) + l_{t-2}\} + \\ &\quad f_t(a_{t-2}, \mathrm{rot}L^{30}(b_{t-2}), c_{t-2}) + m_{t-2} \\ b_t &= \mathrm{rot}L^5(a_{t-2}) + l_{t-2} \\ c_t &= \mathrm{rot}L^{30}(a_{t-2}) \\ e_t &= c_{t-2} \\ l_t &= f_t(b_t, c_t, d_t) + e_t + W_t + K_t \\ m_t &= d_t + W_{t+1} + K_{t+1} \\ n_t &= W_{t+2} + K_{t+2} \end{aligned}\right\} \tag{2}$$

where c_t, d_t, e_t are directly derived from a_{t-2}, b_{t-2}, c_{t-2}, while a_t and b_t require the computational result of a_{t-1}. In order to achieve a higher parallelism in hash operations and reduce the critical path delay in the hash operation block, new parameters l_t, m_t and n_t are defined and pre-computed as important addends to represent the respective computational results of a_{t-1}.

The proposed SHA-1 structure is composed of pre-processing and post-processing parts, as shown in Fig. 3.

The pre-processing part is responsible for the pre-computation of the defined terms l_t, m_t and n_t, while the post-processing part is responsible for the iterative hash operation using the values from a_t to e_t as well as the previous output from the pre-processing part. Because there is no data dependence between the two processing blocks, it is possible for the parallel computation.

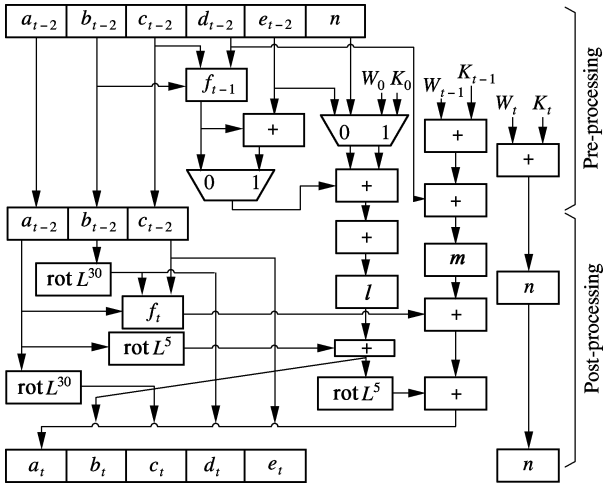

Fig. 3 Optimized SHA-1 architecture

The longest path delay in the two processing blocks exists in a_t and l_t, which are two additions and one nonlinear function f.

Consequently, the critical path delay for the hash operation is the delay of two additions and one nonlinear function f. An additional cycle is needed to initialize the new defined terms l_t, m_t and n_t at the beginning, while the other 40 cycles are required for iterative operation. The proposed SHA-1 structure requires a total of 41 cycles to complete the hash operations. It not only reduces the iterative cycles by half, but also dramatically shortens the critical path delay.

3 Implementations on FPGA

The optimized SHA-1 architecture is implemented on FPGA EP1S25F1020C5, which is one of the Stratix families from Altera.

After successful compiling, an ASCII string text "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq" is chosen as an input of the program, and the execution result, which is a 160-bit hexadecimal string text, is shown in Fig. 4. Fig. 4 also depicts the required 41 clock cycles to complete the whole execution.

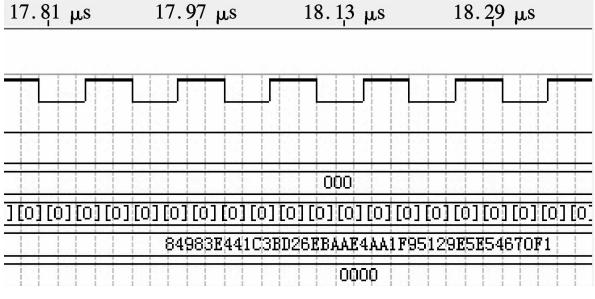

Fig. 4 Program execution result

The validity of the proposed algorithm is tested using the SHA-1 checking software HashCalc. As shown in Fig. 5, the testing result is consistent with that of the program.

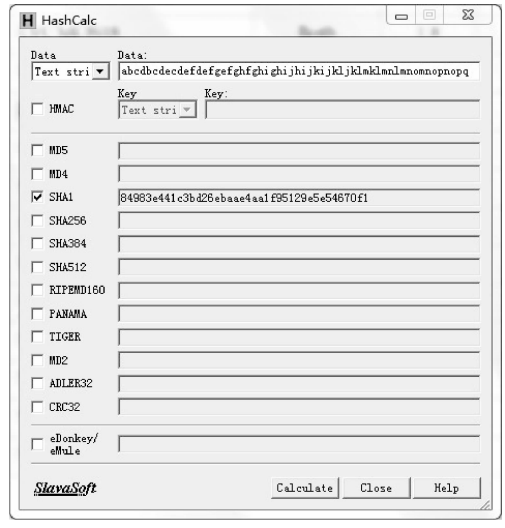

Fig. 5 Testing result

4 Performance Analysis

Comparisons with other SHA-1 designs mainly focus on four factors: frequency, throughput, the number of slices and cycles. The greater the number of slices, the higher the energy consumption of the FPGA. And the more the number of cycles, the more slowly the program will run on the same hardware. Tab. 2 shows performance comparisons with other optimized structures. The throughput is calculated by

$$T = p \frac{f}{n} \times 512 \text{ bits}$$

where T is the throughput rate; p is the stage of pipeline; f is the frequency of the system; n is the number of cycles.

Tab. 2 Performance comparisons with other optimized structures

Design	Frequency/ MHz	Number of cycles	Number of slices	Throughput/ (Gbit · s^{-1})
Ref. [1]	162	80	854	1.0
Ref. [2]	91	40	4 848	4.7
Ref. [3]	42	24	4 258	3.5
Ref. [4]	118	41	2 894	5.9
Proposed design	91	41	2 444	1.2

Ref. [1] uses the original SHA-1 architecture without modification. Refs. [2] and [3] employ both unfolding and pipeline with 4 stages, while Ref. [4] combines loop unfolding, pre-processing and pipelining, but implemented on a different device (Virtex-2).

Tradeoffs among the four evaluated factors are apparently laid out. Though not having the highest throughput rate, the proposed design has reached a fair balance among frequency, the number of slices and cycles.

5 Conclusion

This paper analyzes a high-speed SHA-1 design. By using loop unfolding and pre-processing techniques, the original 80 performing cycles are reduced to 41, and the implementations on FPGA show the proposed design gives the maximum throughput of 1.2 Gbit/s when working at 91 MHz clock frequency. Compared with other designs, the proposed design reaches a fair balance among the four evaluation factors without compromising the security of the original algorithm.

Further research reveals that methods such as pipeline and multi-input adding based on a carry-save adder[8-11] can be used to improve the throughput rate. Our future efforts will be concentrated on the comparisons between different hash algorithms[12], including the comparison of collision rate, execution speed, cracking methods and so on. Furthermore, we will also focus on the actual application scenarios[13], such as in a specific security system.

References

[1] Yiakoumis I I, Papadonikolakis M E, Michail H E, et al. Maximizing the hash function of authentication codes [J]. *IEEE Potentials*, 2006, **25**(2): 9 - 12.

[2] Wang Xiaoyun, Yin Yiqun Lisa, Yu Hongbo. Finding collisions in the full SHA-1 [C]//*25th Annual International Cryptology Conference*. Santa Barbara, CA, USA, 2005: 1 - 16.

[3] Michail H, Goutis C. Holistic methodology for designing ultra high-speed SHA-1 hashing cryptographic module in hardware [C]//*IEEE International Conference on Electron Devices and Solid-State Circuits*. Hong Kong, China, 2008.

[4] Lee Y K, Chan H, Verbauwhede I. Throughput optimized SHA-1 architecture using unfolding transformation [C]//*IEEE 17th International Conference on Application-specific Systems, Architectures and Processors*. Steamboat Springs, CO, USA, 2006: 354 - 359

[5] Lee E H, Lee J H, Park Ⅱ H, et al. Implementation of high-speed SHA-1 architecture [J]. *IEICE Electronics Express*, 2009, **6**(16): 1174 - 1179.

[6] Jung E G, Han D, Lee J G. Low area and high speed SHA-1 implementation [C]//*SoC Design Conference*. Jeju, Republic of Korea, 2011: 365 - 367.

[7] Kim J-W, Lee H-U, Won Y. Design for high throughput SHA-1 hash function on FPGA [C]//*Fourth International Conference on Ubiquitous and Future Networks*. Phuket, Thailand, 2012: 403 - 404.

[8] Michail H E, Kakarountas A P, Milidonis A S, et al. A top-down design methodology for ultrahigh-performance hashing cores [J]. *IEEE Transactions on Dependable and Secure Computing*, 2009, **6**(4): 255 - 268.

[9] Jiang L H, Wang Y L, Zhao Q X, et al. Ultra high throughput architectures for SHA-1 hash algorithm on FPGA [C]//*Computational Intelligence and Software Engineering*. Wuhan, China, 2009.

[10] Nakajima J, Matsui M. Performance analysis and parallel implementation of dedicated hash functions [J]. *IEICE Trans Fund Electron Commun Comput Sci*, 2003, **E86-A**(1): 54 - 63.

[11] Hodjat A, Verbauwhede I. A 21.54 Gbits/s fully pipelined AES processor on FPGA [C]//*Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. Napa, CA, USA, 2004: 308 - 309.

[12] Wang Z Q, Cao L S. Implementation and comparison of two hash algorithms [C]//*Fifth International Conference on Digital Object Identifier*. Shiyan, China, 2013: 721 - 725.

[13] Ratna P, Agung A, Purnamasari P D, et al. QiR (quality in research), analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system [C]//*International Conference on Digital Object Identifier*. Yogyakarta, Indonesia, 2013: 99 - 104.

FPGA 实现的一种 SHA-1 优化杂凑算法

薛 也 胡爱群

(东南大学信息安全研究中心,南京 211189)

摘要:为了满足实际应用中对算法速度以及能耗的需要,提出了一种优化的 SHA-1 算法. 该算法将环展开与预处理 2 种方法相结合,通过在迭代过程中引入中间变量,并且对中间变量进行预先计算,使原本单线程的运算能够多线程地并行运行. 这种并行性缩短了散列函数操作的关键路径,将循环周期从原来的 80 缩减到了 41,运算速率得到了提高,运算时所需的芯片面积也得以减少,从而降低了能耗. 该算法在 FPGA 中硬件实现时的吞吐率高达 1.2 Gbit/s,时钟频率最高为 91 MHz,在吞吐率与时钟频率方面取得了较好的平衡. 仿真结果表明,与其他 SHA-1 的改进算法相比,该优化算法在没有影响经典算法安全性的基础上,获得了较高的吞吐率和较快的速率.

关键词:SHA-1;杂凑算法;环展开;预处理;FPGA

中图分类号:TP30