An HMAC Processor with Integrated SHA-1 and MD5 Algorithms

Mao-Yin Wang, Chih-Pin Su, Chih-Tsun Huang, and Cheng-Wen Wu
Laboratory for Reliable Computing
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 30013

Abstract—Cryptographic algorithms are prevalent and important in digital communications and storage, e.g., both SHA-1 and MD5 algorithms are widely used hash functions in IPSec and SSL for checking the data integrity. In this paper, we propose a hardware architecture for the standard HMAC function that supports both. Our HMAC design automatically generates the padding words and reuses the key for consecutive HMAC jobs that use the same key. We have also implemented the HMAC design in silicon. Compared with existing designs, our HMAC processor has lower hardware cost—12.5% by sharing of the SHA-1 and MD5 circuitry and a little performance penalty.

I. Introduction

Cryptography is playing an increasingly important role in many digital communications applications, especially those done on the Internet [1]. For instance, the US National Institute of Standards and Technology (NIST) specified the keyed-Hash Message Authentication Code (HMAC), a mechanism for message authentication using cryptographic hash functions [2]. Many hardware implementations of widely used cryptographic algorithms which includes the Advanced Encryption Standard (AES) [3], the conventional Data Encryption Standard (DES) [4], and RSA [5] have been reported in the recent years (see, e.g., [6–8]). Besides, hash algorithms have crucial functions in security systems. Currently, SHA-1 [9] and MD5 [10] are the most popular hash algorithms. In our examination, there are a few works on the implementation of hash algorithms [11–15].

In this paper, we present an HMAC hardware design with the integrated SHA-1 and MD5 hash functions. The advantages of our design are as follows: 1) *reduced hardware complexity*—the number of multiplexers is reduced based on our shift-register approach, and the similarity between SHA-1 and MD5 algorithms makes hardware sharing possible; 2) *similar performance*—our hardware sharing approach leads to a little performance penalty; and 3) *HMAC realization*—instead of the individual SHA-1 and MD5 algorithms, we realize the HMAC processor with SHA-1 and MD5 algorithms, which has automatic word padding and supports key scheduling for consecutive HMAC tasks using the same key (removing key computation time). The proposed HMAC processor is applicable for a wide selection of security systems, both for cost-oriented and performance-oriented applications.

II. Hash Functions

A hash function converts plaintext into a message digest (MD) with fixed length. The HMAC performs keyed-hash operations using existing hash functions and the key input.

SHA-1 and MD5 have common features and basic operations such as 1) word expansion, 2) iterative processing steps, 3) fixed, non-scalable data flow, 3) arithmetic and logical operations, 4) nonlinear subfunctions, and 5) the same block size. Motivated by the observation, we will try to share the hardware of SHA-1 and MD5 in our HMAC design, and to develop an efficient structure to implement the hash functions.

III. HMAC Processor Design

The overall architecture of our HMAC processor core, consisting of three major components—the HMAC controller, SHA-1/MD5 core, and register file. The controller manages the data flow (selecting adequate data among the key words, message words, and hash data words in the register file) among all the blocks, and generates adequate control signals to other blocks. The register file consists of the key register that stores the 512-bit key data (K_0), and the hash register that stores the 160/128-bit hashing data for SHA-1/MD5. The most important SHA-1/MD5 core integrates the two popular hash functions, effectively reducing the area cost.

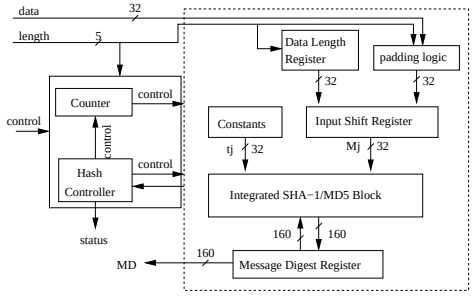

Fig. 1. Block diagrams of the integrated SHA-1/MD5 core.

Fig. 1 shows the block diagram of our integrated SHA-1/MD5 core. In this design, the input data (padded by the padding logic) and the data length information are the inputs of shift register depicted in Fig. 2. The extra byte-exchange (also done by the padding logic) is mandatory for MD5. The shift register then outputs the adequate message word to compute the temporary hash values (A, B, C, D, and E [9, 10]) by the

integrated SHA-1/MD5 block represented in Fig. 3. The constant word and MD are also required by the block. The Counter counts the round number, determining the hash registers whose contents will be added to the MD.

We perform the input data expansion specified in [9, 10] by using a shift register as shown in Fig. 2. The 16 input words (the message block) are received from the data input first. For SHA-1, the extra words are produced by the XOR of four previous words followed by a 1-bit left rotation, as indicated in the figure. For MD5, a multiplexer (on the top of the figure) is needed. In each round, the data word or one of the words in the shift register will be selected as the message word. In our design, proper control of the shift register is done such that only the shift register words with odd indices are necessary.

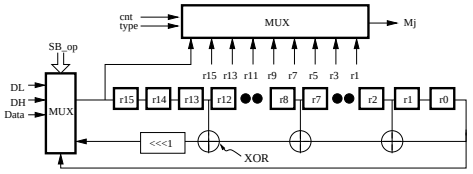

Fig. 2. Design of the input shift register.

The computation of the temporary values for A, B, C, D, and E is listed in Table I. We will propose a shift-register-based design that avoids the selection operation among the temporary registers as done by [13], simplifying the computation.

TABLE I
ROUND FUNCTIONS AND TEMPORARY VALUES.

Reg	Operation	j	Func.
A	$(A \ll 5) + F_j(B,C,D) + E + M_j + t_j$	$j < 80$	SHA-1
B	A	$j < 80$	SHA-1
C	$(B \ll 30)$	$j < 80$	SHA-1
D	C	$j < 80$	SHA-1
E	D	$j < 80$	SHA-1
A	$B + ((A + F_j(B,C,D) + M_j + t_j) \ll s)$	$j\%4 = 0$ & $j < 64$	MD5
B	$C + ((B + F_j(C,D,A) + M_j + t_j) \ll s)$	$j\%4 = 3$ & $j < 64$	MD5
C	$D + ((C + F_j(D,A,B) + M_j + t_j) \ll s)$	$j\%4 = 2$ & $j < 64$	MD5
D	$A + ((D + F_j(A,B,C) + M_j + t_j) \ll s)$	$j\%4 = 1$ & $j < 64$	MD5

Suppose we have a register A for both hash functions.

$$\text{SHA-1}: A = (A \ll 5) + F_j(B,C,D) + E + M_j + t_j, \quad (1)$$

$$\text{MD5}: A = [(A + F_j(B,C,D) + M_j + t_j) \ll s] + B. \quad (2)$$

The proposed implementation of these functions is given in Fig. 3. Its time complexity and hardware cost (in terms of gate count) are as follows:

$$\begin{aligned} T_{max} &= T_{\text{add 4 words}} + T_{\text{add 2 words}} \\ &\quad + Max(T_{\text{mux}}, T_{\text{F(B,C,D)}}, T_{M_j}, T_{t_j}) + T_{\ll s} \\ &\quad + 2T_{\text{mux}} \end{aligned} \quad (3)$$

$$\begin{aligned} Gates &= G_{\text{add 4 words}} + G_{\text{add 2 words}} \\ &\quad + G_{\ll s} + 6G_{\text{mux}} + G_{\text{F(B,C,D)}} + G_{\text{A,B,C,D,E}} \end{aligned} \quad (4)$$

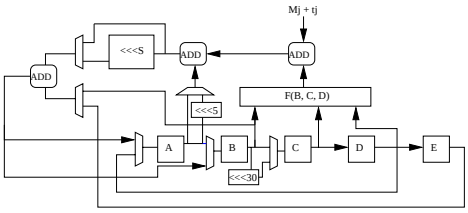

Fig. 3. Integrated implementation for SHA-1/MD5.

IV. EXPERIMENTAL RESULTS

The proposed HMAC processor core has been implemented by using a standard 0.25μm CMOS technology and a cell-based design flow. The total core power is 14.9 mW at a clock rate of 50 MHz. The throughput for performing hashing using the proposed design is

$$\text{Throughput}_{hash} = \frac{\text{Block size} \times \text{Max frequency}}{\text{Clock cycles per block}} \quad (5)$$

Performing the HMAC-SHA-1 and HMAC-MD5 operations for modulo-512 values, the throughput is then

$$\text{Throughput}_{hmac} = \frac{m}{3+k+m} \cdot \text{Throughput}_{hash}, \quad (6)$$

where k and m are the numbers of hash blocks for the key data and message data, respectively. In our design, the SHA-1 hashing of one 512-bit block takes 81 clock cycles, and for MD5 it is 65 cycles. We compare the proposed HMAC design with some commercial products and previous works, as shown in Table II.

From the table, we can see that our integrated SHA-1/MD5 core has only 21K gates, which is smaller than that reported in Cadence by 12.5%, and the two circuits combined proposed in SafeNet by 36%. We also compare the designs implemented by FPGA, and our design requires the fewest logic elements (with a similar number of memory bits) as compared with ALMA and CAST companies, and [12], when both SHA-1 and MD5 are considered. Our designs also outperform others, except those shown in Cadence and SafeNet companies.

However, the performance is comparable if the process technology is normalized. The hardware efficiency of our design makes the integrated SHA-1/MD5 core an attractive design for a wide range of applications. With about 29K gates, the HMAC processor core that performs both HMAC-SHA1 and HMAC-MD5 can easily be integrated on a system chip.

V. CONCLUSIONS

We have designed and implemented an area-efficient integrated SHA-1/MD5 core that supports both the SHA-1 and MD5 hash functions. The word expansion for both hash functions has been implemented by a novel shift-register structure, effectively reducing the area complexity. The hash operations has been rearranged to reduce the critical path delay.

TABLE II
COMPARISON WITH PREVIOUS WORKS.

	Hash func.	Technology	α	f (MHz)	Mbps	γ	λ	β
ALMA Inc.	MD5	EP20K100EQC208-1	512	44	341		2343	2240
ALMA Inc.	SHA-1	APEX20KE	512	64	404		1329	
SafeNet Inc.	MD5	ASIC 0.18μ	512	50-200	400-1600	14K		
SafeNet Inc.	SHA-1	ASIC 0.18μ	512	50-200	640-2560	19K		
CAST Inc.	MD5	EP20K60E	512	39	307		2285	1984
CAST Inc.	SHA-1	EP20K100E	512	50	316		1916	
Cadence Inc.	SHA-1 with MD5	ASIC 0.25μ	512	75	474 for SHA-1 590 for MD5	24K		
ALDEC Inc.	SHA-1	APEX20K100-1	512	64	404		1413 LCs	
McLoone [15]	SHA-384 and SHA-512	Virtex-E XCV600E-8	1024	38	479		2914 slices	
McLoone [14]	HMAC-SHA-1	Virtex XCV1000E	512	50	62.4-78			
Deepakumara	MD5 (iterative method)	Virtex V1000FG680-6	512	21	165		880 slices	
[12]	MD5 (unrolling method)	Virtex V1000FG680-6	512	71.4	354		4763 slices	
Kang [13]	MD5	EP20K1000EBC652-3	512	18	142			
	SHA-1	EP20K1000EBC652-3	512	18	114			
	HAS-160	EP20K1000EBC652-3	512	30	160			
	SHA-1/HAS-160/MD5	EP20K1000EBC652-3	512				14604	
Ours	SHA-1 and MD5	ASIC 0.25μ	512	66	417 for SHA-1 520 for MD5	21K		
	SHA-1 and MD5	EP20K1000EBC652-1X	512 512	22.67	143.3 for SHA-1 178.6 for MD5		3040	2048
	HMAC with SHA-1/MD5	ASIC 0.25μ	512	66	83.4-104.3 for SHA-1 104-130 for MD5	29.2K		
	HMAC with SHA-1/MD5	EP20K1000EBC652-1X	512	21.96	27.7-34.7 for SHA-1 34.6-43.2 for MD5		5329	2048

Note: 1) α is the block size; 2) β is the number of ROM bits or ESB bits for FPGA—we store 64 32-bit constants in memory (MD5); 3) γ is the gate count; 4) λ is the number of building blocks of the Altera FPGA (LC—logic cell—is that of the Xilinx FPGA; 5) our HMAC assumes similar conditions as in [14]; and 5) In [12], 1 CLB contains 2 slices.

An HMAC processor core has also been implemented based on the SHA-1/MD5 design, with automatic word padding. The hardware also supports key scheduling for consecutive HMAC tasks using the same key. As a result, the proposed HMAC processor core is applicable for a wide range of security systems, for both cost-oriented and performance-oriented applications. Compared with existing designs, our HMAC processor has a similar performance, with a lower hardware cost.

REFERENCES

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, Oct. 1996.

[2] National Institute of Standards and Technology (NIST), *The Keyed-Hash Message Authentication Code (HMAC)*, National Technical Information Service, Springfield, VA 22161, Mar. 2002.

[3] National Institute of Standards and Technology (NIST), *Advanced Encryption Standard (AES)*, National Technical Information Service, Springfield, VA 22161, Nov. 2001.

[4] National Institute of Standards and Technology (NIST), *Data Encryption Standard (DES)*, National Technical Information Service, Springfield, VA 22161, Oct. 1999.

[5] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[6] C. Patterson, "High performance DES encryption in virtex FPGAs using Jbits", in *Proc. 8th IEEE Symp. Field-Programmable Computing Machines*, Napa Valley, Apr. 2000, pp. 113–121.

[7] T.-F. Lin, C.-P. Su, C.-T. Huang, and C.-W. Wu, "A high-throughput low-cost AES cipher chip", in *Proc. 3rd IEEE Asia-Pacific Conf. ASIC*, Taipei, Aug. 2002, pp. 85–88.

[8] J.-H. Hong and C.-W. Wu, "Cellular array modular multiplier for the RSA public-key cryptosystem based on modified Booth's algorithm", *IEEE Trans. VLSI Systems*, vol. 11, no. 3, pp. 474–484, June 2003.

[9] National Institute of Standards and Technology (NIST), *Secure Hash Standard (SHS)*, National Technical Information Service, Springfield, VA 22161, Aug. 2002.

[10] R. L. Rivest, "The MD5 message-digest algorithm", RFC 1321, the Internet Society, Apr. 1992.

[11] N. Sklavos, P. Kitsos, K. Papadomanolakis, and O. Koufopavlou, "Random number generator architecture and VLSI implementation", in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Scottsdale, Arizona, May 2002, vol. IV, pp. 854–857.

[12] J. Deepakumara, Howard M., and R. Venkatesan, "FPGA implementation of MD5 hash algorithm", in *Proc. Canadian Conf. Electrical and Computer Engineering*, Toronto, May 2001, vol. 2, pp. 919–924.

[13] Y.-K. Kang, D.-W. Kim, T.-W. Kwon, and J.-R. Choi, "An efficient implementation of hash function processor for IPSEC", in *Proc. 3rd IEEE Asia-Pacific Conf. ASIC*, Taipei, Aug. 2002, pp. 93–96.

[14] M. McLoone and J. V. McCanny, "A single-chip IPSEC cryptographic processor", in *Proc. IEEE Workshop on Singal Processing Systems (SIPS)*, San Diego, Oct. 2002, pp. 133–138.

[15] M. McLoone and J. V. McCanny, "Efficient single-chip implementation of SHA-384 & SHA-512", in *Proc. IEEE Int. Conf. Field-Programmable Technology (FPT)*, Hong Kong, Dec. 2002, pp. 311–314.