Implementation of Efficient SHA-256 Hash Algorithm for Secure Vehicle Communication using FPGA

Chanbok Jeong and Youngmin Kim

School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology (UNIST)
UNIST-gil 50, Ulsan 689-798, Republic of Korea
+82-52-217-2182, {jcbcom, youngmin}@unist.ac.kr

Abstract

In this paper, we implement the SHA-256 FPGA hardware module for the security protocol of the IEEE 1609.2 vehicle communication (VC). VC requires high-throughput and low-latency hardware architectures. For fast and efficient design, we exploit parallel structures for preprocessing and hash computation in SHA-256. The proposed design is implemented in Vertex-5 and verified for correct hash operation. As a result, we can achieve up to 179.08 MHz with 2796 slices.

Keywords-SHA-256, FPGA, Security for Vehicle Communication, Hash Algorithm, IEEE 1609.2, FIPS-180-3

Introduction

Recently, vehicle communication (VC) has become the most important electrical function for intelligent vehicles and systems. IEEE 1609.2 Wireless Access in Vehicular Environments (WAVE) standard method is used for secure VC. To implement WAVE protocol, hash module is required to process both Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Integrated Encryption Scheme (ECIES) security algorithms. In addition, the hash algorithm is vital and widely used in many cryptography processes [1].

In [2], authors have implemented the hash function in SHA-256 module for HMAC (Hash-based Message Authentication Code). To improve the performance, they use four pipeline stages for hash computation of four different input messages, and a carry save adder. In [3], a compact FPGA processor for the SHA-256 algorithm is implemented without preprocessing unit. To optimize the SHA-256 hash function, they have proposed several techniques, such as minimization of the critical path, reducing of the memory access by using data reuse, and a specific 4-input arithmetic logic unit (ALU).

In this paper, we implement an entire SHA-256 algorithm using parallel architecture of the preprocessing and the hash computation, a proposed Latest Message Scheduler (LMS) block, and Xilinx adder IP for WAVE protocol to support fast and efficient cryptography using FPGA. In contrast to other research, we implement the preprocessing function because we use the SHA-256 module for embedded environment such as vehicle communication. By aforementioned efficient methods, we can achieve SHA-256 FPGA hardware with 179.08 MHz speed.

SHA-256 Hash Function

The SHA-256 is an one-way hash function that has been published as FIPS 180-3 by the National Institute of Standards and Technology (NIST) and included in SHA-2 family. Input data of SHA-256 is processed by preprocessing and hash computation. The preprocessing conducts padding a message first, then parses the padded message into N x 512-bit blocks and sets initialization values. After that, hash computation follows to generate message schedule blocks using the padded message and a series of the hash value using the message schedule blocks. Finally, the message digest, which is the output of the SHA-256, is generated as the output. The size of the message digest of SHA-256 is 256 bits. The message digest always contains unique values depending on the input message [4].

In SHA-256, all operations are processed based on 32-bit unit. Thus, all arithmetic operations are implemented based on 2^{32}. The SHA-256 has six logical functions (i.e., *Ch*, *Maj*, $\sum_0^{\{256\}}(x)$, $\sum_1^{\{256\}}(x)$, $\sigma_0^{\{256\}}(x)$, and $\sigma_1^{\{256\}}(x)$), eight working variables (a, b, c, d, e, f, g, and h), and two temporary variables (T_1 and T_2). These logical functions, working variables, and temporary variables are combined and mixed together to process the hash computation stage.

Hardware Implementation

To construct a fast and efficient SHA-256, we apply a parallel architecture for the preprocessing block and the hash computation block in the proposed SHA-256 architecture as shown in Fig. 1. In our design, the preprocessing block for new message and hash computation block are pipelined for higher throughput. At the preprocessing block, the input data (Input_Text) is parsed in the 512 bits padded blocks. The last block consists of the end of message bits, one bit '1' value, sequence of '0' bits, and the 64-bit value of the length of message. After that, the block scheduler prepares the message schedule blocks (W_t). The W_t are used to calculate the eight working variables in the Compute Memory block and the two temporary variables (T_1 and T_2). The W_t from 0 to 15 are equal to the corresponding padded blocks stored in separate 16 registers, but the W_t from 16 to 63 require additional calculations using the previous W_t in a single 32 bits register. To calculate the W_t from 16 to 63, we propose the LMS module for reducing total number of registers. The LMS stores current padded message and schedules to issue a proper message for each hash computation round. At the final, in 63th round,

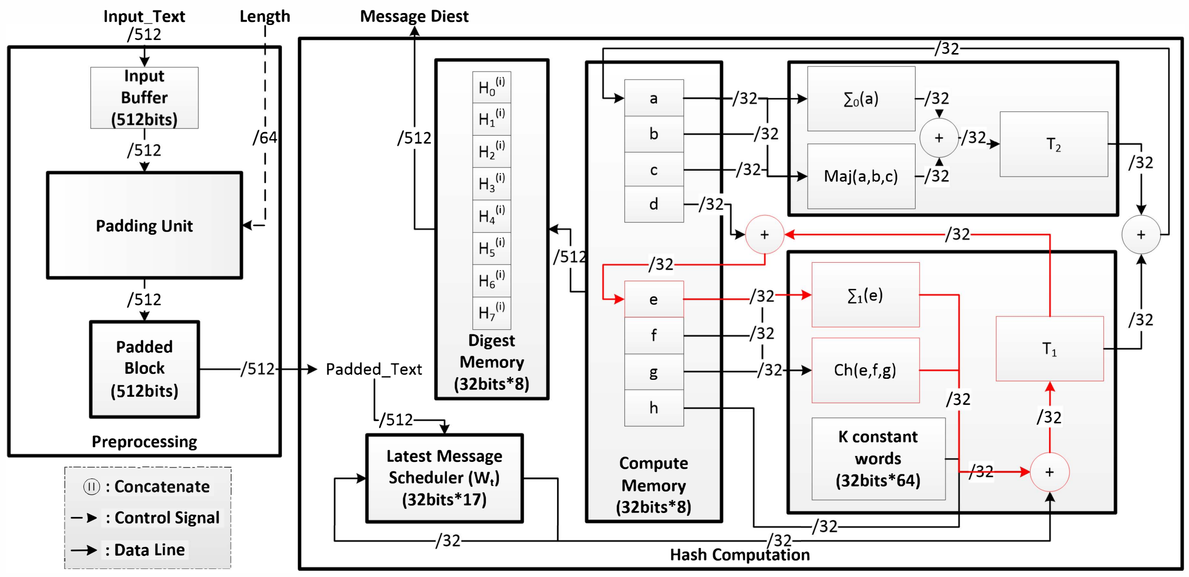
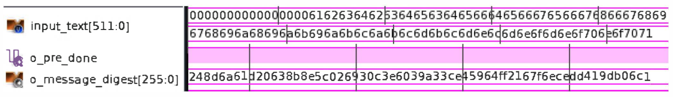


Fig. 1. The proposed parallel structure of the SHA-256.

TABLE I
FPGA Implementation result and comparison with previous works

Device	FPGA slice	Clock Freq. (MHz)	Implementation of Preprocessing
Virtex-5 [2]	1885	169.00	Software
Virtex-5 [3]	139	64.45	No
Virtex-5 [this work]	2796	179.08	Yes (Hardware)



```
input_text[511:0]           00000000000000006162636462636465636465666465666765666768666768696768696a68696a6b696a6b6c6a6b6c6d6b6c6d6e6c6d6e6f6d6e6f706e6f7071
o_pre_done
o_message_digest[255:0]     248d6a61d20638b8e5c026930c3e6039a33ce45964ff2167f6ecedd419db06c1
```

Fig. 2. The result of timing simulation showing input text and output hased message ($o\_message\_digest$).

eight working variables are added with previous hash values to generate the message digest of the input message. In Fig. 1, the critical path of the hash computation block is drawn in red line. As shown, the path has data dependency to calculate the variable itself. And many 32-bit add operations affect the performance of the path. To reduce the delay of this critical path and improve the overall performance, we exploit the adder and subtracter IP logics provided by Xilinx ISE tool [5].

Result

The proposed architecture is implemented in the Virtex-5 and tested in various cases. The FPGA hardware implementation is summarized and compared with other designs in Table I. As shown, we can achieve the best clock speed (i.e., up to 179.08 MHz) with a hardware preprocessing unit by using of 2796 slices in the FPGA hardware. Fig. 2 is timing simulation result of the synthesized SHA-256 FPGA hardware. For the timing simulation, the NIST SHA-256 test vector is used [6].

Conclusion

We have successfully implanted the SHA-256 module in Virtex-5 FPGA chip for use in the security protocol of the IEEE 1609.2 VC. To meet the performance requirement of the VC systems, several techniques for fast and efficient design of the preprocessing and the hash function are proposed and exploited. As a result, the implemented hardware operates correctly in a 179.08 MHz system clock by using of 2796 slices.

Acknowledgments

This research was financially supported by the "R&D Infrastructure for Green Electric Vehicle (RE-EV)" through the Ministry of Trade Industry & Energy (MOTIE) and Korea Institute for Advancement of Technology (KIAT).

References

[1] IEEE Std 1609.2-2013. (April, 2013.) IEEE Standard for WAVE–Security Services for Applications and Management Messages. [Online]. Available : `http://standards.ieee.org/findstds/standard/1609.2-2013.html`

[2] H. E. Michail, *et al.*, "On the Exploitation of a High-Throughput SHA-256 FPGA Design for HMAC," *ACM Trans. on Reconfigurable Tech. and Sys.*, vol. 5 no. 1, pp. 1–28, 2012.

[3] R. Garcïaa, *et al.*, "A compact FPGA-based processor for the Secure Hash Algorithm SHA-256," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 194–202, 2014.

[4] NIST FIPS PUB 180-3. (October, 2008), "Secure Hash Standard (SHS)," [Online]. Available : `http://csrc.nist.gov/publications/fips/fips180-3/fips180-3\_final.pdf`

[5] Xilinx LogiCORE IP Adder/Subtracter v11.0. (March, 2011), [Online]. Available : `http://www.xilinx.com/support/documentation/ip\_documentation/addsub\_ds214.pdf`

[6] SHA-256. (October, 2007), [Online]. Available : `http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/SHA256.pdf`