Hash Function Processor Using Resource Sharing for IPSec Chip

Yong Kyu Kang, Dae Won Kim, Taek Won Kwon and Jun Rim Choi
School of Electronic and Electrical Engineering, Kyungpook National University
1370 Sankyok-Dong, Book-Gu, Daegu, Korea, 702-701
Tel. +82-53-940-8667, Fax.: +82-53-950-5508
e-mail : newer11@palgong.knu.ac.kr

Abstract: This paper presents the implementation of hash functions for IPSEC chip. There is an increasing interest in high-speed cryptographic accelerators for IPSec applications such as VPNs (virtual private networks). Because diverse algorithms are used in Internet, various hash algorithms are required for IPSec chip. Therefore, we implemented SHA-1, HAS-160 and MD5 in one chip. These hash algorithms are designed to reduce the number of gates. SHA-1 module is combined with HAS-160 module. As the result, the required logic elements are reduced by 27%. These hash algorithms have been implemented using Altera's EP20K1000EBC652-3 with PCI bus interface.

1. Introduction

Data integrity assurance and data origin authentication are essential security service in financial transactions, electronic commerce, electronic mail, software distribution, and so on. The broadest definition of authentication within computing systems encompasses identity verification, message origin authentication and message content authentication. In IPSEC, the technique of cryptographic hash functions is utilized to achieve these security services. Hash Functions compress a string of arbitrary length to a string of fixed length. The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. There has been an increased interest in developing a Message Authentication Code (MAC) derived from a hash code. Among the many reasons behind this are that cryptographic hash functions such as SHA-1, MD5 and HAS-160 generally execute faster than symmetric block ciphers such as DES. There are many applications of SHA-1, MD5, HAS-160 and other hash functions to generate MACs. The method to implement the MAC for IPSEC has been chosen as hash-based MAC or HMAC, which uses an existing hash function in conjunction with a secret key.

2. Hash Algorithm

2.1 SHA-1 Algorithm

The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS PUB 180) in 1993; a revised version was issued as FIPS PUB 180-1 in 1995 and is generally referred to as SHA-1. The algorithm takes as input a message with a maximum length of less than 2^{64} bits and produces as output a 160-bit message digest. The input is processed in 512-bit blocks. The heart of the algorithm is a module that consists of four rounds of processing 20 steps each. The logic is illustrated in Figure 1.

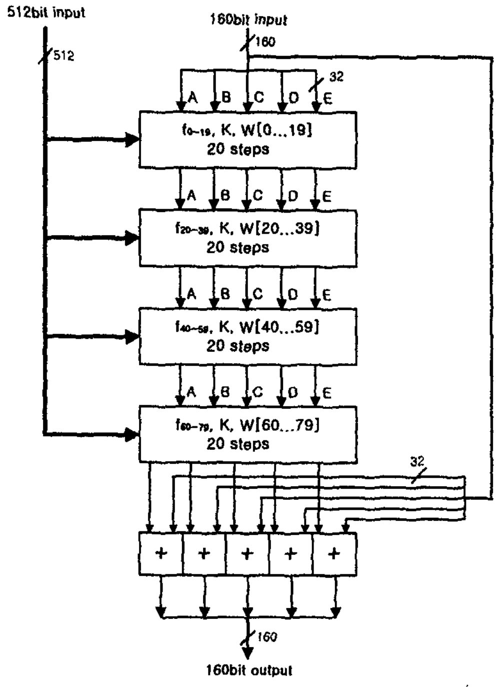


Figure 1. SHA-1 processing of a single 512-bit block

The four rounds have a similar structure, but each uses a different primitive logical function. Each round makes use of an additive constant K_t and W[t], where $0 \le t \le 79$. W[t] is intermediate value.

2.2 HAS-160 Algorithm

The Hash function Algorithm Standard (HAS-160) was developed by Korea Telecommunications Technology Association (KTTA). HAS-160 algorithm takes advantages of SHA-1 and MD5. The structure of HAS-160 is similar to the architecture of SHA-1.

2.3 MD5 Algorithm

The MD5 message-digest algorithm, specified in RFC 1321, was developed by Ron Rivest at MIT. The algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest. The input is processed in 512-bit blocks. This algorithm is simpler than the previous two algorithms.

3. FPGA Implementation

3.1 SHA-1 Structure

The SHA-1 architecture is illustrated in Figure 2. This architecture uses 32-bit data bus and its flow is controlled by multiplexers. The right side of Figure 2 is the core operation blocks and the left side of Figure 2 is the circuit for intermediate value. The circuit for intermediate value generates intermediate value W[t], where $0 \le t \le 79$. The intermediate values for each of the step operation are generated from the previous intermediate value. The circuit for intermediate value designed on-the-fly method. This method can reduce the number of register. We design it with 512-bit registers and two multiplexers comparing with the method of storing all intermediate values using 2560-bit registers.

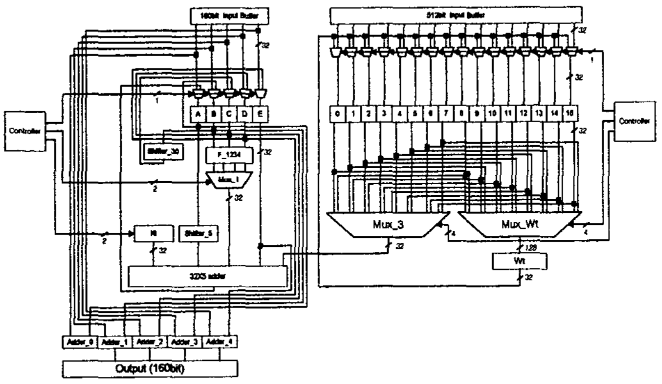


Figure 2. The architecture of SHA-1

The 512-bit input is loaded to the 512-bit input buffer. The 160-bit input buffer is used to hold the 160-bit initial value. F_1234 block in Figure 2 is four Boolean operators for parallel processing. SHA-1 algorithm consists of four rounds of processing of 20 steps each. Each round needs a different Boolean operator. Kt block is the ROM that stores constants Kt, where $0 \le t \le 79$. The 32×5 adder block calculates each five 32-bit inputs. To minimize the adder delay, we implemented high speed adder using 8-bit CLA (carry look-ahead adder), CSA (carry select adder) and CSA (carry save adder).

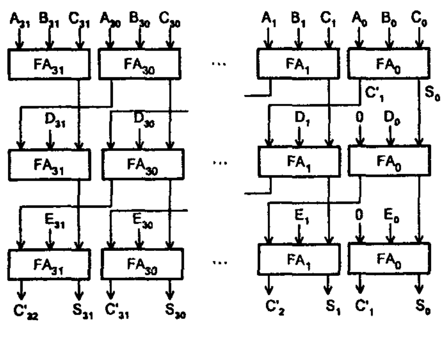

A_{31} B_{31} C_{31} A_{30} B_{30} C_{30}
A_1 B_1 C_1 A_0 B_0 C_0
FA_{31} FA_{30} FA_1 FA_0
D_{31} D_{30} D_1 D_0
E_{31} E_{30} E_1 E_0
C'_1 S_0
C'_{32} S_{31} C'_{31} S_{30}
C'_2 S_1 C'_1 S_0

Figure 3. Extension of carry save adder

Extension of carry save adder is illustrated in Figure 3. We designed 32×5 adder using a three-column carry save adder and high speed 32-bit$\times$2 adders.

3.2 HAS-160 Structure

The HAS-160 architecture is illustrated in Figure 4. This architecture is similar to the SHA-1 architecture except for the circuit of intermediate value. Therefore the implemented architecture holds the core operation blocks of the architecture of SHA-1 in common. Controller and the circuit for intermediate value are made separately and these blocks combine with the architecture of SHA-1.

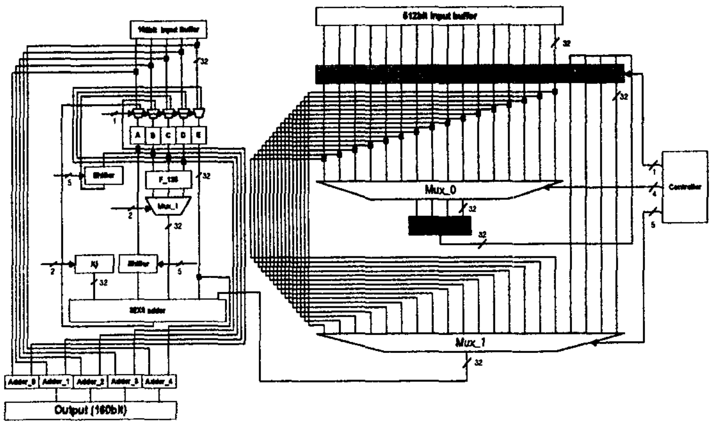


Figure 4. The architecture of HAS-160

3.3 MD5 Structure

The MD5 architecture is illustrated in Figure 5.

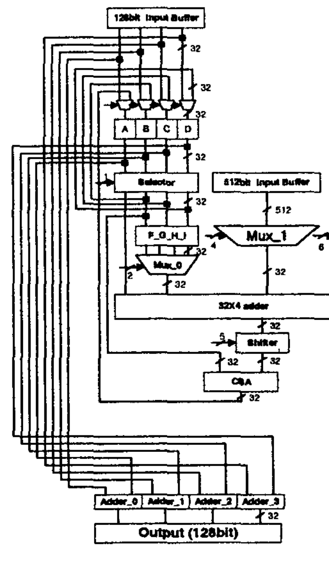


Figure 5. The architecture of MD5

This architecture doesn't need the complicated circuit for intermediate value. Input in 512-bit buffer is multiplexed only with wiring to generate intermediate value. The core operation blocks are similar to the previous two algorithms. But the 128-bit output gives a simple hardware design.

3.4 The structure of non-resource sharing hash function

The global structure of non-resource sharing processor is illustrated in Figure 6.

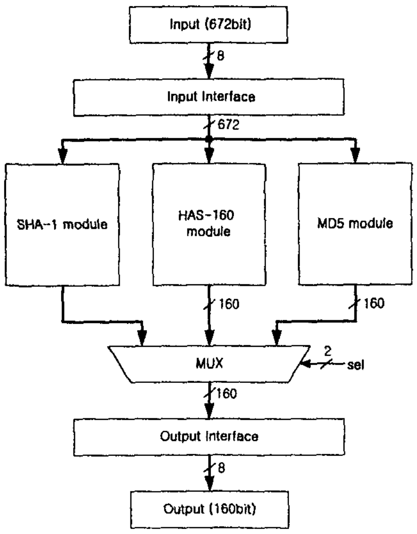


Figure 6. The global structure of non-resource sharing processor

This architecture has each algorithm module such as SHA-1 module, HAS-160 module and MD5 module. This architecture is implemented with all algorithms in one chip. The separated modules are easily designed for IPSec chip but it is so inefficient in size.

3.5 The structure of resource sharing processor

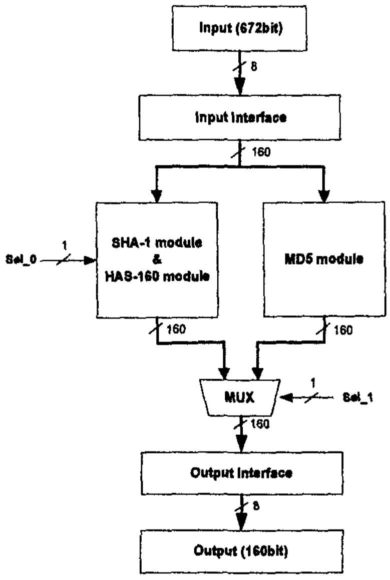


Figure 7. The global structure of resource sharing processor

The global structure of resource sharing processor is illustrated in Figure 7. This architecture unites SHA-1 module and HAS-160 module. As mentioned before, it is necessary for the efficient design in one chip.

4. Performance Analysis

4.1 Operating time

The performance of our design architecture of hash processor is shown in Table 1. This performance is the same general architecture as modified architecture. All designs were synthesized and placed and routed on the Altera's EP20K1000EBC652-3 target device with speed grade 3. The obtained results depend on speed grade of FPGA devices. Figure 8 shows that test-board connects with PC through PCI interface. Test-board control program executes each hash algorithm. Figure 9 is captured practical working screen.

Table 1. The feature of the hash processor

	SHA-1	HAS-160	MD5
Clock Cycles	81	96	65
Frequency	18MHz	30MHz	18MHz
Throughput	114Mbit/s	160Mbit/s	142Mbit/s

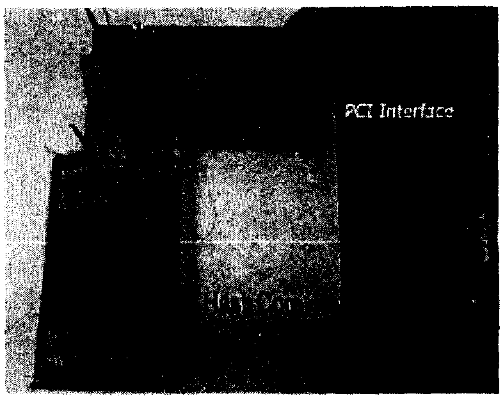
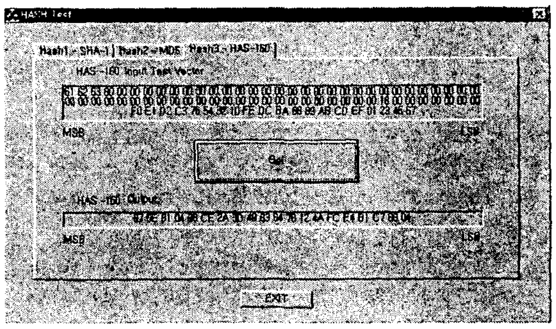


Figure 8. Test-board



```
[illegible]
F0 E1 D2 C3 76 54 32 10 FE DC BA 98 89 AB CD EF 01 23 45 67
```

```
[illegible]
```

Figure 9. Test-board control program

4.2 Hardware resources and Comparison

In the case of the general architecture design, presented in Figure 6, the utilization of logic elements was 14604. This design occupies 38% chip area of Altera's EP20K1000EBC652-3. For the modified architecture design, presented in Figure 7, the utilization of logic design, presented in Figure 7, the utilization of logic elements was 10573. The summary is given in Table 2. As a result, the required logic elements reduce 27%.

Table 2. The logic element counts of each architecture

	Non-resource sharing processor	Resource sharing processor
Logic element counts	14604	10573

5. Conclusions

In this paper, hash processor was designed and mapped in FPGA. For IPSEC chip, various hash algorithms is requested. Therefore, we implemented SHA-1, HAS-160 and MD5 in one chip. For the purpose of implementing one chip, each algorithm module such as SHA-1 module, HAS-160 module and MD5 module is separately designed. In result, we find that SHA-1 module is similar to HAS-160 module. SHA-1 module is combined with HAS-160. We present a new architecture to reduce the chip size by using the reusable blocks. And performance is same. The proposed architecture provides a good solution to the practical IPSEC chip implementation.

References

[1] National Institute of Standards and Technology, "Secure Hash Standard (SHA-1)," Federal Information Processing Standards Publication #180-1, 1993.

[2] Korea Telecommunications Technology Association (KTTA), "Hash Function Standard (HAS-160)," TTAS. KO-12.0011, 2000.

[3] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, MIT LCS & RSA Data Security, Inc., April 1992.

[4] W. Stallings, "Data and Computer Communications Fifth Edition," Prentice-Hall, NJ, 1997.

[5] J. Deepakumara, Howard M. Heys and R. Venkatesan, "FPGA Implementation of MD5 Hash Algorithm," Electrical and Computer Engineering Canadian Conference on, vol. 2, pp 919-924, 2001.

[6] B. Parhami, "Computer Arithmetic Algorithms and Hardware Designs," Oxford University Press, 2000.