FPGA Implementation of SHA–1 Algorithm

Dai Zibin   Zhou Ning
Institute of Electronic Technology, Information Engineering University
Zhengzhou, 450004, P.R. China
Email: ll7004@sohu.com

Abstract

In information security, message authentication is an essential technique to verify that received messages come from the alleged source and have not been altered. A key element of authentication schemes is the use of a message authentication code (MAC). One technique to produce a MAC is based on using a hash function and is referred to as an HMAC. Secure Hash Algorithm 1 (SHA-1) is one of the algorithms, which has been specified for use in Internet Protocol Security (IPSEC), as the basis for an HMAC. As we shall show in the paper, it is reasonable to construct cryptographic accelerators using hardware implementations based on SHA-1 hash algorithm. Finally, the synthesis results based on the FPGAs are given.

1. Introduction

Hash functions are very common and important cryptographic primitives. Their primary application is their use together with public-key cryptosystems in the digital signature schemes. They are also a basic building block of secret-key Message Authentication. This authentication scheme appears in two currently most widely deployed security protocols, SSL and IPSEC. By far one of the most widely accepted hash functions is SHA-1(Secure Hash Algorithm-1) [3] [4].

Reconfigurable hardware devices such as FPGAs (Field Programmable Gate Arrays) are an appealing alternative for the implementation of cryptographic algorithms [1] [2]. Their advantages combine flexibility and ease of upgrade with the improved physical security and performance that characterize hardware implementations. In addition, the time and cost of FPGA design are smaller than in other hardware approaches.

2. 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-bits message digest. The input is processed in 512-bits blocks. The algorithm processing includes the following steps:

(1). Padding

The purpose of message padding is to make the total length of a padded message congruent to 448 modulo 512(length = 448 mod 512). The number of padding bits should be between 1 and 512. Padding consists of a single 1-bit followed by the necessary number of 0-bits.

(2). Appending Length

A 64-bits binary representation of the original length of the message is appended to the end of the message.

(3). Initialize the SHA-1 buffer

The 160-bits buffer is represented by five four–word buffers (A, B, C, D, E) used to store the middle or finally results of the message digests for SHA-1 functions and they are initialized to the following values in hexadecimal. Low-order bytes are put first.

Word A:	67	45	23	01
Word B:	EF	CD	AB	89
Word C:	98	BA	DC	EF
Word D:	10	32	54	76
Word E:	C3	D2	E1	F0

(4). Process message in 16-word blocks

The heart of the algorithm is a module that consists of four rounds of processing 20 steps each. The four rounds have a similar structure, but each uses a different primitive logical function. These logical functions are defined as follows:

$$f_t(B,C,D)=\begin{cases}(B\wedge C)\vee(\bar{B}\wedge D) & 0\le t\le 19\\ B\oplus C\oplus D & 20\le t\le 39\\ (B\wedge C)\vee(B\wedge D)\vee(C\wedge D) & 40\le t\le 59\\ B\oplus C\oplus D & 60\le t\le 79\end{cases}\qquad(1)$$

These rounds take as input the current 512-bits block and the 160-bits buffer value (A, B, C, D, E), and then update these buffers. Each round also makes use of an additive constant K_t. In hex these are given by:

$$K_t=\begin{cases}5A827999 & 0\le t\le 19\\ 6ED9EBA1 & 20\le t\le 39\\ 8F1BBCDC & 40\le t\le 59\\ CA62C1D6 & 60\le t\le 79\end{cases}\qquad(2)$$

The output of the fourth round is added to the input of the first round, and then the addition is modulo 2^{32} to produce the ABCDE value that calculate next 512-bits block.

(5). Output

After all 512-bits blocks have been processed, the output of the last block is the 160-bits message digest. Figure1 shows the operations involved in a single step.

3. FPGA Implementation

(1). Function Block Diagram

The diagram shown in Figure2, describes the basic architecture of FPGA-based SHA-1 implementation. The controller is a finite state machine (FSM) that iterates the loop sequence and controls input and output data flows by generating the necessary control signals for every block in the architecture. A K_t register latches constant words is used in the algorithm. The padding block is used to load the input data and to append padding bits and the length of the original message.

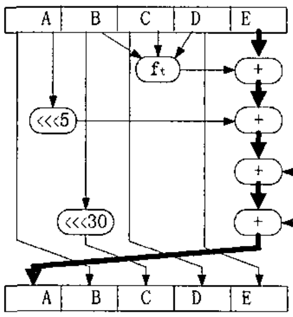


Figure1. Operations in a single step of SHA-1

(2). Expansion Block

According to SHA-1 algorithm, the message digest is computed using the message padded as the above described. The computation is described using W_t, a sequence of eighty 32-bits words. The first 16 of these words, W_0, W_1,…,W_{15}, is simply the first 16 words of the input message block, M_0, M_1,…, M_{15}, the remaining words are computed using a simple feedback function, based on rotations, shifts, and XOR operations. Wt can be calculated by following formula:

$$W_t=\begin{cases}M_t & 0\le t\le 15\\ (W_{t-3}\oplus W_{t-8}\oplus W_{t-14}\oplus W_{t-16})<<<1 & 16\le t\le 79\end{cases}\qquad(3)$$

The actual implementation of the functions is given in Figure 3.

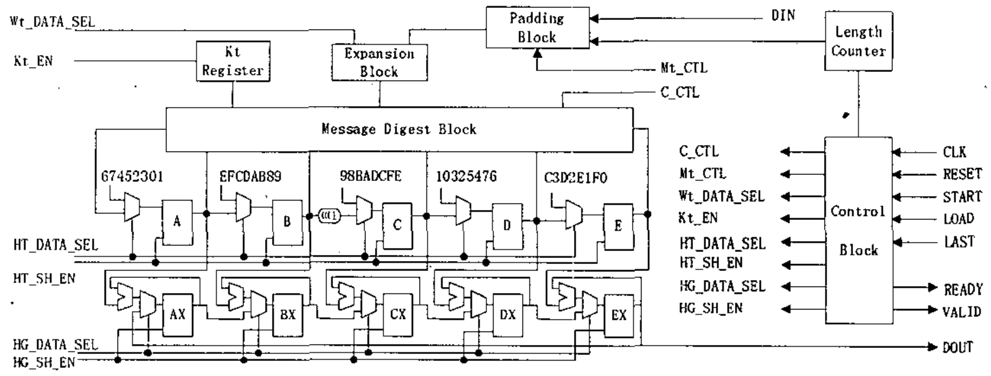


Figure2. Block Diagram of FPGA-Based SHA-1 Implementation

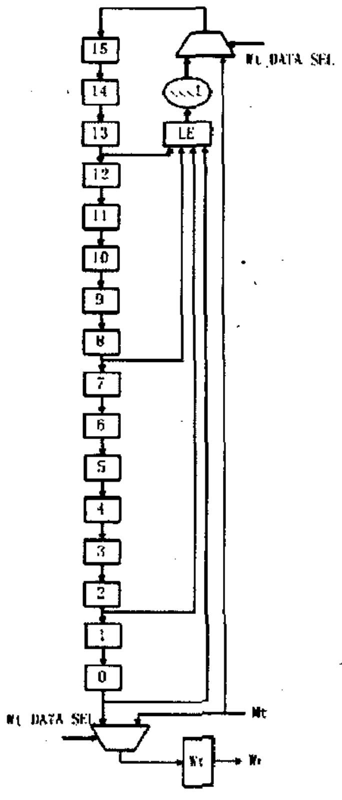


Figure3. Expansion Block

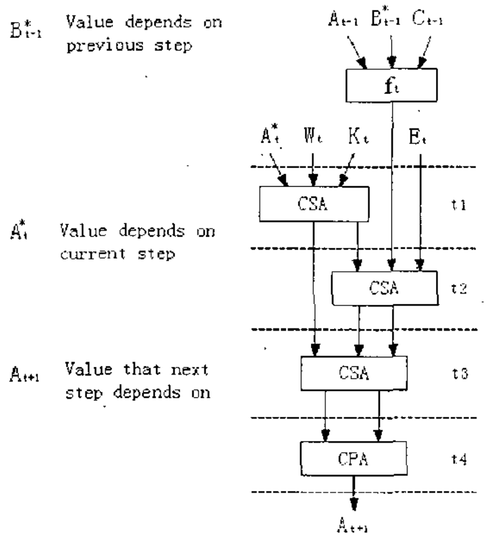
(3). Data Path

In SHA-1 algorithm, four out of five words (A, B, C, D) remain almost unchanged by a single round. These words are only shifted by one position down. The last word, E, undergoes a complicated transformation equivalent to multi-operand addition modulo 2^{32}, with five 32-bits operands dependent on all input words, the round dependent constant K_t, and the message dependent word W_t. The principles of updating these registers are described as follows:

$$\begin{cases} A_{t+1} = (E_t + f(t, B_t, C_t, D_t) + A_t(<<<5) + W_t + K_t) \bmod 2^{32} \\ B_{t+1} = A_t \\ C_{t+1} = B_t(<<<30) \\ D_{t+1} = C_t \\ E_{t+1} = D_t \end{cases} \tag{4}$$

In our implementations, the five words of the message digests are stored in five 32-bits registers (A, B, C, D, E). These registers are initialized with the constant initialization vector when START signal is enabled, and are updated with the new value in each round during shift-enable signal is asserted high.

The critical path is marked with a thick line in Figure1. The circuit uses the carry save representation of numbers to speed-up the multi-operand addition, and minimizes delays associated with carry propagation [5]. The straightforward use of carry save adders in case of five operand addition would lead to three levels of 3-to-2 carry save adders, followed by a carry propagate adder as shown in Figure 4.


B*t-1 Value depends on previous step
A*t Value depends on current step
At+1 Value that next step depends on
At-1 B*t-1 Ct-1
ft
A*t Wt Kt Et
CSA
CSA
CSA
CPA
t1
t2
t3
t4
At+1
Figure4. Data Path

4. Performance Evaluation

To process one 512-bits block operation of SHA-1 algorithm requires 82 system clock cycles in our design. The first cycle is data reading cycle, the following 80 cycles are operation cycles and the last cycle is writing output register cycle. Its operation rate can be calculated by following formula:

Throughput= System clock frequency $\times$Bits of message block / Number of rounds

Our design were synthesized and placed and routed on the EP1K100QC208 target device [6]. The performances of SHA-1 IP Core based on these different speed grade devices are shown in Table1.

Table1. Performances of SHA-1 IP-CORE

Target Device	System Clock f_{max}(MHz)	Area (LEs)	Throughput (Mbps)
EP1K100QC208-1	43.08	1622	268.99
EP1K100QC208-2	32.79	1622	204.74
EP1K100QC208-3	23.15	1622	144.55

In the case of choosing Altera's EP1K100QC208-1 to implement this design, the utilization of logic elements was 1622. Its maximum frequency is 43 MHz. To compare with ALDEC Inc. [7], this design throughput increases 59%. The required logic elements reduce 17%.

5. Conclusions

The significance of the hardware implementation of the SHA-1 algorithm has been examined. The architecture has been studied for both area utilization and speed with FPGAs as the target device. It is clear that the architectures can be easily fitted to a single device. Although the inherent nature of the SHA-1 structure does not allow parallel hash operations of blocks, hardware implementations can obtain a significant throughput to cater to some of currently available IP bandwidths.

The obtained results can be further improved by using the latest FPGA devices such as Cyclone family. The Cyclone devices provide better performance than the previous generation of FPGAs. The proposed architecture provides a good solution to the practical IPSEC chip implementation. FPGA implementations would therefore be suitable as components in cryptographic accelerators.

References

Journals

[1] J. Deepakumara, H.M. Heys, and R. Venkatesan, Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, (Toronto, Ontario, May 2001), p.176

[2] J. Elbirt, W. Yip, B. Chetwynd and C. Paar, IEEE Transactions on VLSI, Volume 9, Issue 4, 2001, p.547

Books

[3] FIPS 180-1, Secure hash standard, NIST, US Department of Commerce, Washington D. C., April 1995

[4] Schneier. B, Applied Cryptography, algorithms, and source code in C, (John Wiley and Sons, Inc., 1996), 2nd ed., p316

[5] B. Parhami, Computer Arithmetic Algorithms and Hardware Designs, (Oxford University Press, 2000), p274

[6] Altera Corporation, ACEX 1K Programmable Logic Device Family Datasheet, p.17

[7] Aldec Inc., Communication IP Cores-SHA, http://www.aldec.com