Design of an HMAC Co-Processor Unit Based on SHA-2 Family of Hash Functions

Naisy Mol S.
PG Scholar,VLSI & Embedded Systems
Dept.of Electronics and Communication
TKM Institute of Technology
Kollam, India

Sarath Raj S.
Associate Professor
Dept. of Electronics and Biomedical
TKM Institute Of Technology
Kollam,Kerala,India

***Abstract*— Cryptography entails the design of algorithms for encryption and decryption, intended to ensure the secrecy and authenticity of messages in communication systems. The Secure Hash Algorithm (SHA) is a family of cryptographic hash functions published by NIST. SHA-2 is a family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512.Secure hash algorithms are typically used with other cryptographic algorithms, such as digital signature algorithms and keyed-hash message authentication codes. Hash-based Message Authentication Code (HMAC) is a mechanism for message authentication using cryptographic hash functions. This paper aims to design an efficient HMAC crypto-processor unit for cryptographic applications such as spacecrafts, e-mail, electronic fund transfer, etc. In order to facilitate the computation of message digests, the HMAC module provides a direct interface with the built-in SHA-2 core through input and output ports. The coding is done in VHDL language, synthesized using Xilinx ISE 13.2 and simulated using ModelSim SE 6.2c.**

Keywords*— Cryptographic hash functions, Crypto- processor, SHA-256 , HMAC.

I. Introduction

The growing demand of secure high-speed digital communications has demanded the inclusion of cryptographic primitives into system design. Since cryptographic processing is usually not lightweight, some applications delegate those tasks to specific hardware modules. One of the most important requirements to be satisfied in secure communications is data integrity and data origin authentication. In order to satisfy those security requirements, the U.S. National Institute of Standards and Technology (NIST) recommends the use of the Keyed-Hash Message Authentication Code(HMAC) based on the Secure Hash Standard (SHS).

Several standards and protocols employ SHS and HMAC. For instance, the Digital Signature Standard (DSS) utilizes SHA to compute message digests and generate random numbers. Furthermore, HMAC along with SHS is employed in the Transport Layer Security Protocol (TLS) and Security Architecture for the Internet Protocol (IPsec). More specifically, those algorithms are used to perform data origin authentication and integrity verification for the IPSec Authentication Header(AH), Encapsulating Security Payload (ESP), and Internet Key Exchange Protocol(IKE and IKEv2). The security of the algorithms and protocols depends directly on the cryptographic strength of the underlying hash functions. Due to efficient collision search attacks against SHA-1 and MD5 hash functions, more secure hash algorithms of SHS, such as SHA-2 has gain more importance. To address the ongoing demand of higher levels of security, a high-performance HMAC/SHA-2 processor is introduced. This project aims to design an efficient SHA-256 core for cryptographic applications such as space crafts, e-mail, electronic fund transfer, etc..

This paper is organized as follows. Section 2 focuses on cryptographic hash functions, Secure Hash Standard(SHS) and Algorithm descriptions. In Section 3, proposed method; design of HMAC co-processor unit based on SHA-2 family.ie,SHA-256 is preferred for hash computation in HMAC. In section 4, the simulation results of the proposed techniques were discussed. Conclusions and on-going works are discussed in section 5.

II. Literature Review

A.Cryptography and Hash functions

Cryptography is the science of secret writing with the goal of hiding the meaning of a message. It seems closely linked to modern electronic communication. Cryptography itself splits into three main branches: Symmetric Algorithms, Asymmetric (or Public-Key) Algorithms and Cryptographic Protocols. In Symmetric Algorithms ,two parties have an encryption and decryption method for which they share a secret key. All cryptography from ancient times until 1976 was exclusively based on symmetric methods. Symmetric ciphers are still in wide spread use, especially for data encryption and integrity check of messages. In public-key cryptography, a user possesses a secret key as in symmetric cryptography but also a public key. Asymmetric algorithms can be used for applications such as digital signatures and key establishment, and also for classical data encryption. Cryptographic Protocols deals with the application of cryptographic algorithms, Symmetric and asymmetric algorithms. Cryptanalysis is the science and sometimes art of breaking cryptosystems. It is the only way to assure that a cryptosystem is secure, and it is an integral part of cryptology[6].

Hash functions are an important cryptographic primitive and are widely used in protocols. They compute a digest of a message which is a short, fixed-length bit string. For a particular message, the message digest, or hash value, can be seen as the finger print of a message, i.e., a unique representation of a message. Unlike all other crypto algorithms introduced, hash functions do not have a key. The use of hash functions in cryptography is manifold: Hash functions are an essential part of digital signature schemes and message authentication codes. Hash functions are also widely used for other cryptographic applications, e.g., for storing of password hashes or key derivation[6].

B. Secure Hash Standard (SHS)

This Standard specifies five secure hash algorithms, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. All five of the algorithms are iterative, one-way hash functions that can process a message to produce a condensed representation called a message digest. These algorithms enable the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest. This property is useful in the generation and verification of digital signatures and message authentication codes, and in the generation of random numbers or bits[2].

Each algorithm can be described in two stages: preprocessing and hash computation. Preprocessing involves padding a message, parsing the padded message into m-bit blocks, and setting initialization values to be used in the hash computation. The hash computation generates a message schedule from the padded message and uses that schedule, along with functions, constants, and word operations to iteratively generate a series of hash values. The final hash value generated by the hash computation is used to determine the message digest.

The five algorithms differ most significantly in the security strengths that are provided for the data being hashed. The security strengths of these five hash functions depends on cryptographic algorithms, such as digital signature algorithms and keyed-hash message authentication codes. Additionally, the five algorithms differ in terms of the size of the blocks and words of data that are used during hashing. Table 2.1 presents the basic properties of these hash algorithms.

The five hash algorithms specified in this Standard are called secure because, for a given algorithm, it is computationally infeasible to find a message that corresponds to a given message digest and also infeasible to find two different messages that produce the same message digest. Any change to a message will result in a different message digest. This will result in a verification failure when the secure hash algorithm is used with a digital signature algorithm or a keyed-hash message authentication algorithm.

Table 2.1 : Secure Hash Algorithm Properties

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message digest size (bits)
SHA-1	$<2^{64}$	512	32	160
SHA-224	$<2^{64}$	512	32	224
SHA-256	$<2^{64}$	512	32	256
SHA-384	$<2^{64}$	1024	64	384
SHA-512	$<2^{64}$	1024	64	512

C. Algorithms and Descriptions

One of the goals of the HMAC algorithm is to be independent of a given hash function, so that the latter can be easily replaced with faster and more secure algorithms. Besides, the underlying hash function constitutes the core of the HMAC algorithm, and dictates its security level. Since this paper proposes an HMAC crypto-processor unit based on SHA-2, both algorithms are presented in this section.

SHA-2 Algorithm

The SHA-2 family of hash functions comprises four algorithms, namely, SHA-224, SHA-256, SHA-384, and SHA-512.The execution of SHA-2 algorithms is divided into two parts: Pre-processing and Hash computation. A list of SHA-2 parameters is presented in table 2.2, whereas SHA-2 symbols are listed below:

B : SHA-2 input block size (in bits);
L : SHA-2 message digest size (in bits);
D :Data path width (in bits);
N : Number of message blocks;
i : Message block index, where $1 \leq i \leq N$;
j : Number of algorithm iterations;
t : Iteration index, where $0 \leq t \leq j -1$;
$M^{(1)},\ldots,M^{(N)}$: Message blocks;
$H_0^{(0)},\ldots,H_7^{(0)}$: Initial hash values;
$H_0^{(i)},\ldots,H_7^{(i)}$: Intermediate hash values;
$W_0,\ldots,W_j$: Message schedule words;
A,…, h : Working variables;
$K_0,\ldots,K_j$: Constants.

Table 2.2 : SHA-2 algorithm parameters

parameter	SHA-224	SHA-256	SHA-384	SHA-512
B	512	512	1024	1024
L	224	256	384	512
D	32	32	64	64
J	64	64	80	80

Symbols and Operations

The following symbols are used in the secure hash algorithm specifications; each operates on *w*-bit words.

$\wedge$	Bitwise AND operation.
$\vee$	Bitwise OR ("inclusive-OR") operation.
$\oplus$	Bitwise XOR ("exclusive-OR") operation.
$\neg$	Bitwise complement operation.
+	Addition modulo 2*w*.
<<	Left-shift operation, where *x* <<*n* is obtained by discarding the left-most *n* bits of the word *x* and then padding the result with *n* zeroes on the right.
>>	Right-shift operation, where *x* >>*n* is obtained by discarding the right-most *n* bits of the word *x* and then padding the result with *n* zeroes on the left.

The following operations are used in the secure hash algorithm specifications:

ROTL^n(x) The rotate left (circular left shift) operation, where x is a w-bit word and n is an integer with $0 \le n < w$, is defined by ROTL^n(x)=(x <<n) $\vee$ (x >>w - n).
ROTR^n(x) The rotate right (circular right shift) operation, where x is a w-bit word and n is an integer with $0 \le n < w$, is defined by ROTR^n(x)=(x >>n) $\vee$ (x <<w - n).
SHR^n(x) The right shift operation, where x is a w-bit word and n is an integer with $0 \le n < w$, is defined by SHR^n(x)=x >>n.

Hash Computation

The hash computation is based on operations over D-bit words. The number of iterations performed by the algorithm is given by 'j'. For SHA-224/256, j = 64, whereas for SHA-384/512, j = 80. Actually, 'j' can be considered to represent the number of D-bit words processed by the algorithm. More specifically, the SHA-2 algorithms comprise 'j' message schedule words (W_0,..,W_j), eight working variables (a, b, c, d, e, f, g, h), and eight intermediate hash values ($H_0^i,\ldots,H_7^i$). Additionally, six logical functions are employed.ROR^n(x) and SHR^n(x) correspond to, respectively, a rotation and a shift of 'x' by 'n' bits to the right. Besides,'$\oplus$' represents the bitwise XOR operation, '$\wedge$' the bitwise AND operation, and '$\neg$x' the bitwise complement of x[2].

After preprocessing is completed, each message block, $M^{(1)}, M^{(2)},\ldots,M^{(N)}$, is processed in order, using the following steps:

```
For i=1 to N
{
1. Prepare the message schedule
$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_0^{256}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$

2. Initialize the eight working variables
a = $H_0^{i-1}$
b = $H_1^{i-1}$
c = $H_2^{i-1}$
d = $H_3^{i-1}$
e = $H_4^{i-1}$
f = $H_5^{i-1}$
g = $H_6^{i-1}$
h = $H_7^{i-1}$

3. For t=0 to 63
{
T1 = h+$\Sigma_1^{256}(e)$ + Ch(e,f,g) + $k_t^{256}$ + $W_t$
T2 = $\Sigma_0^{256}(a)$ + Maj(a,b,c)
h=g
g=f
f=e
e=d+ T1
d=c
c=b
b=a
a= T1 + T2
}

4.Compute thei^th intermediate hash value $H^{(i)}$:
    $H_0^i$= a + $H_0^{i-1}$
    $H_1^i$= b + $H_1^{i-1}$
    $H_2^i$= c + $H_2^{i-1}$
    $H_3^i$= d + $H_3^{i-1}$
    $H_4^i$=e + $H_4^{i-1}$
    $H_5^i$= f + $H_5^{i-1}$
    $H_6^i$= g + $H_6^{i-1}$
    $H_7^i$= h + $H_7^{i-1}$
}
```

After repeating steps one through four a total of *N* times (i.e., after processing $M^{(N)}$), the resulting 256-bit message digest of the message, *M*, is

$$H_0^N \| H_1^N \| H_2^N \| H_3^N \| H_4^N \| H_5^N \| H_6^N \| H_7^N$$

HMAC Algorithm

The HMAC algorithm processes two inputs, a cryptographic key and a message, to produce message authentication code(MAC). The combination of HMAC with SHA-2 is denoted as HMAC/SHA-2. Individual combinations of HMAC with the four SHA-2 algorithms are denoted as HMAC/SHA-224, HMAC/SHA-256,HMAC/SHA-384,and HMAC/SHA-512. Due to certain commonalities between SHA-224 and SHA-256, as well as between SHA-384 and SHA-512, the algorithms can also be referred to as HMAC/SHA-224/256 and HMAC/SHA-384/512[5].

HMAC Parameters and Symbols

B : SHA-2 input block size (in bits);
L : SHA-2 message digest size (in bits);
Key : Secret key of the communicating parties;
K : Size of the Key (in bits);
K_0 :Key after any necessary pre-processing;
Ipad: (Inner pad) Byte 0x36 repeated B=8 times;
Opad: (Outer pad) Byte 0x5C repeated B=8 times;
Text : The data on which the HMAC is calculated;
Hash(V) : Hash of variable/value V ;
$\|$(0..)z: Padding with 'z' zeros.

Parameters ‘B’ and ‘L’ are inherited from SHA-2. The Text can be ‘n’ bits long, where $0 \leq n < 2^{B-8} - B$, whereas the size of a MAC is ‘L’ bits long. The HMAC algorithm is consisted of seven steps as shown below. Notice that the hashes computed in Step IV and Steps VII can be split into two parts to facilitate its implementation.

HMAC Algorithm Steps

Step I: Pre-process key as follows

$$K_0 = \begin{cases} \text{Key}, & k=B \\ \text{Key} \| (0..)_{B-K} & k<B \\ \text{Hash(key)} \| (0..)_{B-L} & k>B \end{cases}$$

Step II: Compute $(K_0 \oplus \text{Ipad})$.
Step III: Do $(K_0 \oplus \text{Ipad}) \| \text{Text}$
Step IV: $\text{Hash}((K_0 \oplus \text{Ipad}) \| \text{Text})$
StepV : Compute $(K_0 \oplus \text{Opad})$
StepVI : Do $(K_0 \oplus \text{Opad}) \| \text{Hash}((K_0 \oplus \text{Ipad}) \| \text{Text})$
StepVII: $\text{Hash}((K_0 \oplus \text{Opad}) \| \text{Hash}((K_0 \oplus \text{Ipad}) \| \text{Text}))$

Step VII produces the MAC of the message digest.A common practice is to truncate the MAC by using only its ‘t’leftmost bits. Here, MAC is ‘L’ bits long, and any truncation is performed at the user level. To compute a MAC over the data ‘text’ using the HMAC function, the following operation is performed:

$$\mathbf{MAC(text)_t = HMAC(K, text)_t = H((K_0 \oplus Opad) \| H((K_0 \oplus Ipad) \| text))_t}$$

D. Message Authentication Codes

A Message Authentication Code (MAC), also known as a cryptographic checksum or a keyed hash function, is widely used in practice. In terms of security functionality, MACs share some properties with digital signatures, since they also provide message integrity and message authentication. However, unlike digital signatures, MACs are symmetric-key schemes and they do not provide nonrepudiation. One advantage of MACs is that they are much faster than digital signatures since they are based on either block ciphers or hash functions. Similar to digital signatures, MACs append an authentication tag to a message. The crucial difference between MACs and digital signatures is that MACs use a symmetric key ‘k’ for both generating the authentication tag and verifying it. A MAC is a function of the symmetric key ‘k’ and the message ‘x’

i.e., $m = MAC_k(x)$

The principle of the MAC calculation and verification is shown in Figure 1.The motivation for using MACs is typically that sender and receiver want to be assured that any manipulations of a message in transit are detected. For this, sender computes the MAC as a function of the message and the shared secret key ‘key’. Sender sends both the message and the authentication tag MAC to receiver. Upon receiving the message and MAC, receiver verifies both. Since this is a symmetric set-up, receiver simply repeats the steps that sender conducted when sending the message. Receiver merely re-computes the authentication tag with the received message and the symmetric key.

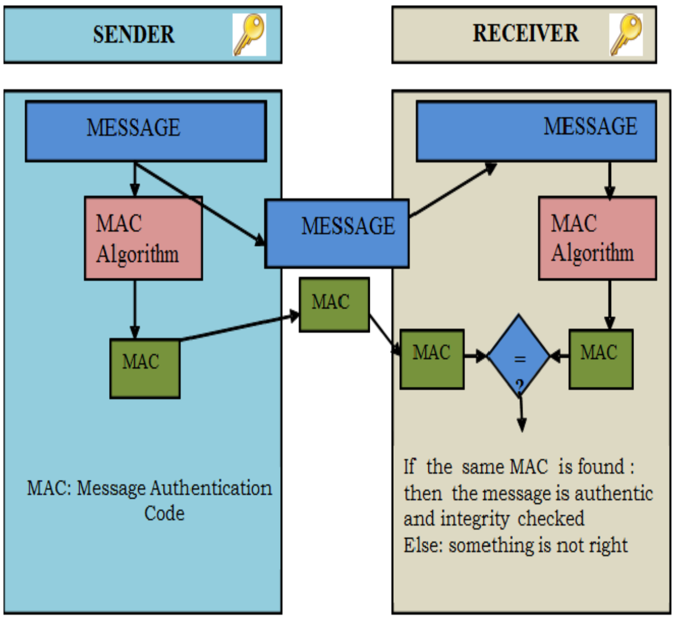
SENDER
RECEIVER
MESSAGE
MAC Algorithm
MESSAGE
MAC
MAC
MESSAGE
MAC Algorithm
MAC
=?
MAC
MAC: Message Authentication Code
If the same MAC is found : then the message is authentic and integrity checked
Else: something is not right

Fig. 1. principle of MACs

The underlying assumption of this system is that the MAC computation will yield an incorrect result if the message was altered in transit. Hence, message integrity is provided as a security service. Furthermore, receiver is now assured that sender was the originator of the message since only the two parties with the same secret key have the possibility to compute the MAC. If an adversary, intruder, changes the message during transmission, he cannot simply compute a valid MAC since he lacks the secret key. Any malicious or accidental (e.g., due to transmission errors) forgery of the message will be detected by the receiver due to a failed verification of the MAC.

III. PROPOSED METHOD

In practice, message authentication codes are constructed in essentially two different ways from block ciphers or from hash functions. One possible construction, named HMAC, has become very popular in practice over the last decade. This section presents the HMAC co-processor unit based on the SHS, and comprising the entire SHA-2 family of hash functions. Therefore, the proposed co-processor unit provides applications with much higher levels of security than previous works. In hash computation, the HMAC module allows the user to interface directly with the built-in SHA-2 core through the ports Key_Text_In and Msg_Digest_MAC_Out. The proposed HMAC architecture consists of a SHA-2core, multiplexors, logical operations, and registers as shown in Figure 2. Three registers, namely K_0 Ipad_ Hash, K_0 Ipad_ Text_ Hash, and K_0 Opad_ Hash are also used in hash computation.

A. Architecture of HMAC co-processor unit

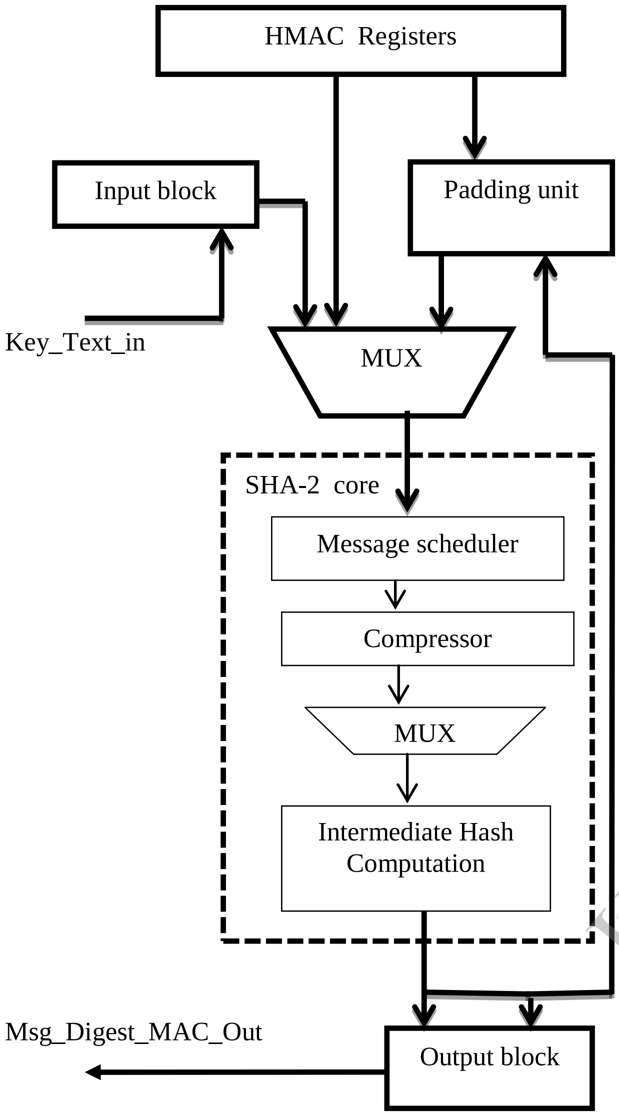

Fig. 2. Architecture of HMAC unit

The HMAC hardware module to be designed needs to cope with both message digest and MAC computations, to process long messages and keys of different sizes (K · B and K > B),as well as to perform pre-processing of long keys. Actually, key reuse is supported the utilizing registers K_0_Ipad_Hash and K_0_Opad_Hash in further computations. Moreover, register K_0_Opad_ Hash can also works as a temporary key storage during the HMAC processing. The HMAC processing is divided into five stages, such as NewKeyHash, KeyIpadHash,TextHash, KeyOpadHash, and MACHash. The stages are executed in aboveorder, but not all stages are necessarily used. The size of the key, the size of the text, and whether a key is being reused will determine the number of stages needed[1].

B. Architecture of SHA-2 core

The hardware design of the SHA-2 algorithm consists of shift-registers, logical operations, D-bit adders, and a memory to store the algorithm constants. Figure 3 shows the SHA-2 architecture. The SHA-2 architecture can be divided into four main blocks: Intermediate Hash Computation, Compressor, Message Scheduler, and Constants Memory.

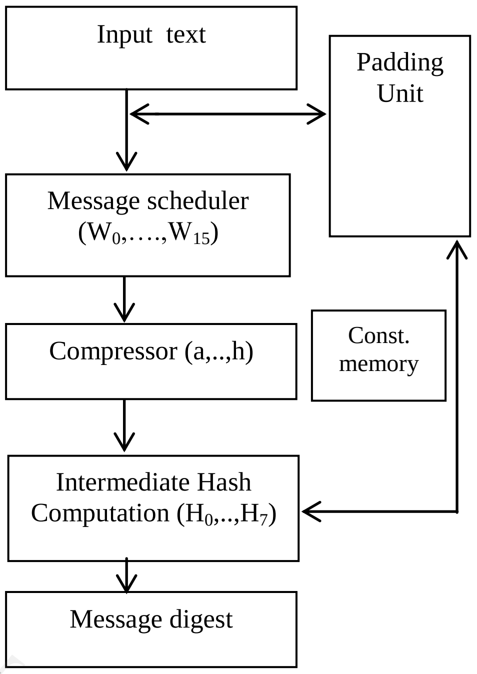
Input text
Padding Unit
Message scheduler (W_0,….,W_{15})
Compressor (a,..,h)
Const. memory
Intermediate Hash Computation (H_0,..,H_7)
Message digest

Fig. 2. Architecture of SHA-2 core

The message scheduler's registers W_0,...,W_{15} can be initialized either in serial or in parallel. Serial initialization requires the first 16 words M_t of the message M to be shifted into the module. For this processing it takes 16 clock cycles. It is utilized when writing a new message or key to the SHA-2core through the HMAC module. For example, it can be a message being hashed, a long key being pre-processed, or a text being processed as part of a MAC computation.

Parallel initialization is used by the HMAC module to load internally stored values directly into registers W_0,...,W_{15}. Simultaneously, the constants memory provides the initialization values for the working variables (a,..., h). Initial hashes (H_0,...,H_7) are also set within this period of time. The initialization of working variables and initial hashes takes one clock cycle. When initialization is complete, the compressor employs registers a,...,h, as well as W_t and K_t to determine the new values of a,..., h. The algorithm takes 't' iterations and is controlled internally by an iteration counter, which is described using step 3 of SHA-2. Precisely, SHA-224/256 and SHA-384/512 utilizes 64 and 80 iterations respectively,. In each of these iterations, registers W_0,...,W_{15} and a,..., h are shifted in the direction of the arrows shown in Figure 3. The register requirements are 1024 bits for SHA-224/256, and 2048 bits for SHA-384/512.

After 't' iterations, the intermediate hash computation is performed. It would be possible to execute this operation in a single clock cycle, eight adders are to be operated in parallel. To save implementation area, only two adders are utilized. Therefore, the computation of the intermediate hash is spread over the last 4 iterations by computing two additions per clock cycle. The additions are performed when t = 60,..., 63 for SHA-224/256.For instance, in SHA-224/256, H3 and H7 are computed when t = 60, H2 and H6 are computed when t = 61, and so on.

In the case of hashing a multi-block message, a new execution cycle initiates. If serial load is used, 16 more words Mt must be shifted into the module. If parallel load is used, the HMAC module reloads registers $W_0,...,W_{15}$ with the appropriate value. After that, the same procedure described above is re-executed. When the hash is complete, the message digest is available in registers $H_0,...,H_7$. Again, the message digest may correspond to either a single hash computation or to any of the hashes computed in the HMAC algorithm. In the case of a single hash computation, the result is read serially. Since a D-bit word is output per read operation, L=D iterations are necessary. Specifically, SHA-224, SHA-256, SHA-384 and SHA-512 require 7, 8, 6, and 8 read operations respectively. In the case of the HMAC processing, registers $H_0,...,H_7$ are read in parallel by the HMAC core.

The number of iterations involved in a hash computation, makes it clear that a single bit-flip in the constants memory, input blocks, or registers can easily compromise the computation of the hash function and the associated upper level application (e.g., data integrity checking and MACs). For making hash function designs appropriate for space applications, error detection and correction schemes must be incorporated so that SEU (Single Event Upsets)s do not compromise its normal operation.

IV. RESULTS AND DISCUSSION

The design entry is modelled using VHDL in Xilinx ISE Design Suite 13.2 to obtain the synthesis report and the simulation of the design is performed using Modelsim6.2c to validate the functionality of the design. The simulation result of SHA-256 algorithm is shown in figure 4.

For SHA-256 algorithm each input message block has 512 bits, which are represented as a sequence of sixteen 32-bit words. The output or message digest are of 256 bits. SHA-2 algorithm uses 64 iterations to produce 256-bit message digest. The clk(clock), rst(reset), m(input message) are the inputs and v(valid signal) , md(output message digest) are the outputs of SHA-256 algorithm. After forcing rst =0 and the binary value of input message to 'm' , output is obtained as per the algorithm described in above sections. The six bit counter and ROM module also designed to get the functionality of SHA-2 core.

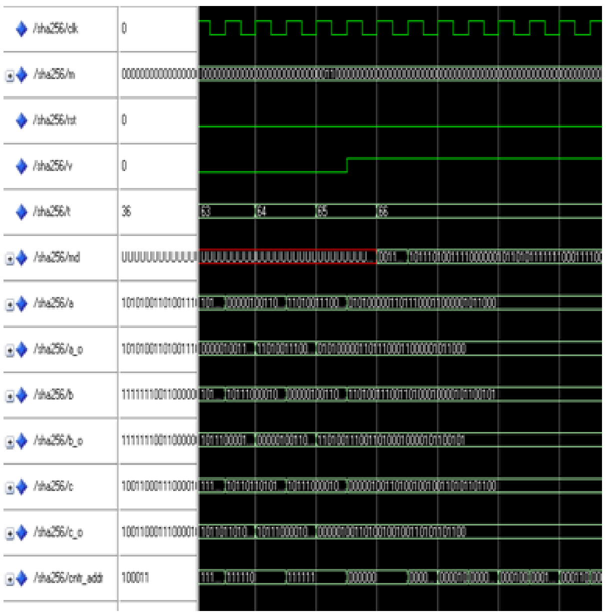

Fig. 4. simulation result of SHA-256

The simulation result of HMAC algorithm is shown in figure 5. The HMAC algorithm processes two inputs, a cryptographic key and a message, to produce message authentication code(MAC). A fully generic design of the HMAC co-processor unit had been developed in the hardware description language VHDL in order to achieve better security in communication systems.

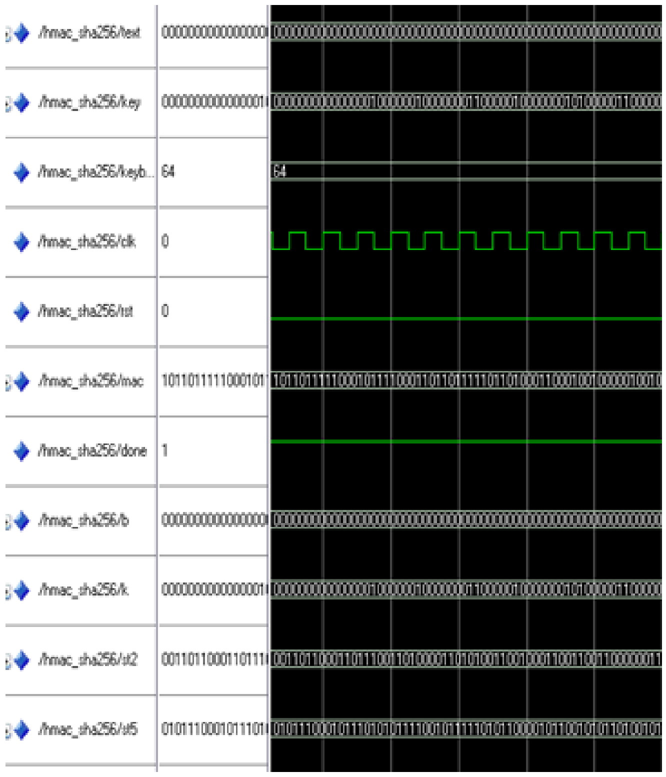

Fig. 5. simulation result of HMAC

V. Conclusion

This paper explains a high-performance HMAC co-processor based on SHA-2 family of hash functions. The utilization of hash functions and Keyed-Hash Message Authentication Codes (HMAC) are of utmost importance to ensure data integrity and data origin authentication in digital communications. Hash-based Message Authentication Code (HMAC) is a mechanism for message authentication using cryptographic hash functions. The Secure Hash Algorithm (SHA) is a family of cryptographic hash functions published by NIST.SHA-2 with HMAC will not necessarily lead to slower implementations or higher energy consumption. SHA-2 with HMAC is completely feasible to efficiently replace SHA-1 and MD5 with SHA-2 in hardware implementations of HMAC. These results are fundamental to support the implementation of higher levels of security in high-end and constrained applications.

References

[1] MarcioJuliato and Catherine Gebotys, "A Quantitative Analysis of a Novel SEU-Resistant SHA-2 and HMAC Architecture for Space Missions Security" *IEEE transactions on aerospace and electronic systems*, vol. 49, no.3,pp.1536-1554 ,july 2013.

[2] NIST. Secure hash standard (SHS). Federal Information Processing Standards Publication FIPS PUB 180-3, October 2008.

[3] Sklavos, N. and Koufopavlou, O, "On the hardware implementations of the SHA-2 (256, 384, 512) hash functions", *Proceedings of the 2003 International Symposium on Circuits and Systems* (ISCAS '03), vol. 5, Bangkok, Thailand, May 2003, pp. 153—-156.

[4] NIST. Advanced encryption standard(AES).Federal Information Processing Standards Publication FIPS PUB 197, November 2001.

[5] NIST. The keyed-hash message authentication code (HMAC).Federal Information Processing Standards Publication FIPS PUB 198, March 2002.

[6] Christ of Paarand JanPelzl, "Understanding Cryptography: A Textbook for Students and Practitioners" Chapters 10,11&12;page nos: 259-325.

[7] Grembowski, T and et al. "Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512".*Information Security: Proceedings of the 5th InternationalConference on Information Security* (ISC'02), vol. 2433, Sao Paulo, Brazil, Sept. 30—Oct. 2, 2002, pp. 75—89, NewYork: Springer-Verlag, 2002.

[8] Yiakoumis I, Papadonikolakis M, and Michail H, "Efficient small-sized implementation of the keyed-hash message authentication code" *Proceedings of theIEEE Eurocon Conference 2005 Computer as a Tool*, volume 2, pages 1875-1878, November 2005.

[9] Kim M, Kim Y, Ryou J, and Jun S, " Efficient implementation of the keyed-hash message authentication code based on SHA-1 algorithm for mobile trusted computing", *Proceedings of the 4thInternational Conference on Autonomic and Trusted Computing (ATC 2007)*, pages 410-419, 2007.

[10] NIST.Digital signature standard(DSS), Federal Information Processing Standards Publication Draft FIPS PUB 186-3, November 2008