A VHDL Model for Implementation of MD5 Hash Algorithm

Mohammed A. Noaman
Electrical Engineering Department, University of Technology /Baghdad
Email:hayanni75@yahoo.com

Received on: 14/3/2012 & Accepted on: 8/11/2012

ABSTRACT

With the increase of the amount of data and users in the information systems, the requirement of data integrity is needed to be improved as well, so the work has become necessary independently. One important element in the information system is a key of authentication schemes, which is used as a message authentication code (MAC). One technique to produce a MAC is based on using a hash function and is referred to as a HMAC.MD5 represents one efficient algorithms for hashing the data, then, the purpose of implementation and used this algorithm is to give them some privacy in the application. Where they become independent work accessories as much as possible, but what is necessary, such as RAM and the pulse generator. Therefore, we focus on the application of VHDL for implement and computing to MD5 for data integrity checking method and to ensure that the data of an information system is in a correct state. The implementation of MD5 algorithm by using Xilinx-spartan-3A XCS1400AFPGA, with 50 MHz internal clock is helping for satisfies the above requirements.

Keywords: FPGA, MD5, MAC, Hash function and password.

نموذج VHDL لبناء خوارزمية MD5

الخلاصة

مع زيادة كمية البيانات والمستخدمين في انظمة المعلومات, لذا فانه من الضروري تحسين نظام سلامة وتكامل البيانات وبالتالي زيادة استقلاليته. واحد من العناصر المهمة في انظمة المعلومات هو مفاتيح المصادقة والذي يستخدم كرمز لمصادقة الرسائل (MAC) . واحدة من التقنيات الاساسية لتوليد (MAC) هو باستخدام دالة الفرم (Hash function) والتي يشار لها ب (HMAC). تعدMD5 من اكفأ الخوارزميات المعدة لهذا الغرض لذا الغرض من بنائها واستخدامها لاستخدام خصائصها في التطبيق. لذا يصبح عملها في التطبيقات مستقلا عدا بعض الملحقات المهمة كالذاكرةاو مولد النبضات عند الحاجة. لذلك ركزنا على تطبيق VHDL لبناء وحساب ال MD5 لتحقق خصائصها. تم البناء لخوارزمية MD5 باستخدام Xilinx-spartan-

https://doi.org/10.30684/etj.31.6A7
2412-0758/University of Technology-Iraq, Baghdad, Iraq

XCS1400AFPGA 3A مع نبضة داخلية ٥٠ ميكا هيرتز والتي ساعدت على تحقيق المتطلبات اعلاه.

INTRODUCTION

With the increase of the amount of data and users in the information system, the requirement of data integrity is needed to be improved as well. So it was imperative for the designers of information security to take many measures to protect private data and systems from the early days [1].While the information security and the message authentication is an essential technique to verify that received message 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 a HMAC. Message Digest 5 (MD5) is one of its algorithms[2].

RELATED WORK

In the existing technical literature, many related, studies from the implementation point of view, the main are focused in three directions these are; Pure Hardware by Xilinx, Mixed Hardware with Software by use GPU and pure software; these are;

In hardware direction; [3] present a FPGA implementation of MD5 modules, besides the internal parallel easily to duplicate and connected to Ethernet LAN. His built on Cyclone II EP2C35F672C6, for a single board, a throughput of 4.3Gbps was achieved with 30,134 logic elements are used. An extensive study of effects of pipelining on delay, area requirements and throughput is performed. The design was carried out on a Xilinx Virtex-II XC2V4000-6 FPGA, and a throughput of 586 Mbps was achieved with logic requirements of only 647 slices and two Block RAMs was made by[4].

In the GPU(mixed) direction;[5] was focused his work on the application of GPU (Graphical Processor Unit) for speedup the processing time. His implementation was based on 64 bytes with shift 112 left and 64 rights, then gained 16 times speedup over CPU but loses the independence. The IPSec protocol stocks for micro server are proposed by [6]. Protocol authentication was worked by MD5 with 2.144 Kbyte in 2.704 Kbyte IPSec Protocol. The good-point in his work is the mixing between the commercial and security requirements.[7]was built the cloud data server with session controller architecture to satisfy MD5 algorithm.[8]puts forward a CUDA-based design of the MD5 hash algorithm on GPU according to the specific application needs and its implementation, which based on C-language, is comprehensive optimization in terms of the characteristics of GPU and CUDA.[9]was presented an efficient implementation for MD5-RC4 encryption using NVIDIA GPU with a novel CUDA programming framework. The MD5-RC4 encryption algorithm built on NVIDIA GeForce 9800GTX GPU with 580 MHz core clock and 512 MB of GDDR. The performance of his solution was compared with the implementation running on an AMD Sempron Processor LE-1200 CPU. The results show that his GPU-based implementation exhibits a performance gain of about 3-5 times speedup for the MD5-RC4 encryption algorithm.

In software direction;[1]was presented the data integrity checking method based on MD5 to ensure that the data of an information system is in a correct state, his checking system was built as software by C^{++} .[10] suggests a software implementation of a digital envelope for a secure e-commerce channel that combined the hashing algorithm of MD5, the symmetric key algorithm of AES and the asymmetric key algorithm of Hyper elliptic Curve Cryptography (HECC) by use Java. The result illustrates that HECC is the best alternative asymmetric key technique rather than ECC and RSA in the digital envelope hybrid cryptosystem.

MESSAGE DIGESTS (MD)

Message Digest (MD) algorithms, also called as Hash algorithms, which generate a unique message digest for an arbitrary message. Also, it's used widely in cryptographic protocols and Internet communication [3]. The core of MD algorithm is a hash function, which compress a string of arbitrary length for a string of fixed length. They provide a unique relationship between the input and the hash value and replace the authenticity of a large amount of information (message) by the authenticity of much smaller hash value (authenticator) [2].

One of the most famous algorithms of MD is the MD5 message digest's algorithm developed by Ronald Rivest [4].The message digests to be generated by MD5 algorithm has the irreversible and non-counterfeit features, so MD5 algorithm is superior in anti-tamper capability. Also, it can be considered as a fingerprint of the message, and it must have the following properties:

First - must be easy to compute,
Second - it must be very hard to compute the message from the digest and,
Third - it must be hard to find another message which has the same message digest as the first one [4].

IMPLEMENTATION STEPS OF MD5 ALGORITHM

MD5 algorithm flowchart shown in Figure (1) consist of the following steps;

1. Check the no. of bits (length) of the original massage (NOM).
2. Add No. of bits to the input massage (IM) so total length of the result data equal to the multiple of 512-64(the add bit are 1 0 0 ……….0).
3. Add 64-bit which represent the length of the (IM) to the result of point 2 the final will represent by M. where M (multiple of 512 bits).
4. Divided M to blocks (B) each one have 512 bits.
5. Divided (B) to 16 blocks (X) each one have 32-bits.
6. The algorithm steps have 4 rounds and each round have 16 steps. Then total steps are 64 steps.
7. There are four 32-bit shift registers, each one have initial (Hex.) values as follow;
   a. Reg. A=[6 7 4 5 2 3 0 1] 32-bits [A] = [D]'
   b. Reg. B=[e f c d a b 8 9] 32-bits [B] = [C]'
   c. Reg. C=[9 8 b a d c f e] 32-bits [C] = [B]'
   d. Reg. D=[1 0 3 2 5 4 7 6] 32-bits [D] = [A]'
8. The values of A, B, C and D are stored temporary in AA, BB, CC, and DD respectively.
9. Then;

This is the algorithm core, which includes four "rounds" of processing. Each round consist of 16 steps and each step uses a 64 element table T [0…63] for T[i], i is the index as shown in table 1,the four rounds have similar structure but each uses different functions *F, G, H* and *I.*

$F(B,C,D)=[(B \wedge C) \vee (\neg B \wedge D)]$, *T[0-15], 16 steps*

$G(B,C,D)=[(B \wedge D) \vee (B \wedge \neg D)]$,*T[16-31], 16 steps*

$H(B,C,D)=[B \oplus C \oplus D]$, *T[32-47], 16 steps*

$I(B,C,D)=[C \oplus (B \vee \neg D),]$*T[48-63], 16 steps*

Where: v, ʌ, ⊕, and ¬ represent the logical OR, AND,XOR and NOT operations, respectively.

The operation in single step involve the functions

$$A=B+((A+F(B,C,D)+X_J[k]+T[i])<<<s) \quad \ldots(1)$$

$$D=C;C=B;B=A;A=D \quad \ldots(2)$$

Where:

$X_J[k]$-kth X (divided of B) and Jth no. of blocks (B-512 bits).

$<<<s$ – circular shift left by s-bits.

The value of *k*, *s*, and *T[i]* from the table (1)[4].

In the end of four rounds, the output is added to the input of the first round

$$A=A+AA;\ B=B+BB;\ C=C+CC;\ D=D+DD \quad \ldots(3)$$

10. Then, the output are 128-bits will arrange as follows;

(most) *D C B A* *(lest)*

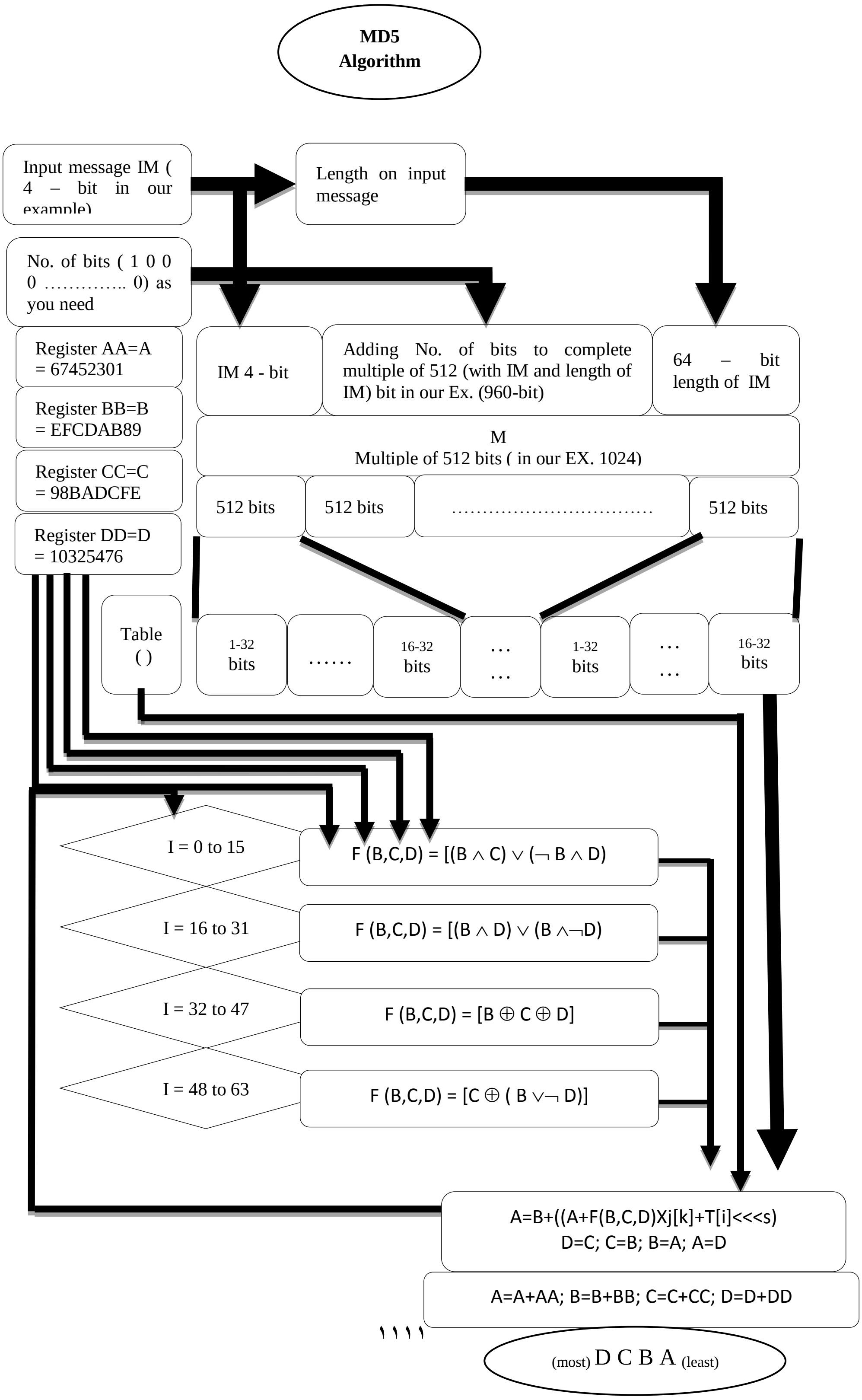

MD5 Algorithm
Input message IM (4 – bit in our example)
Length on input message
No. of bits (1 0 0 0 ………….. 0) as you need
Register AA=A = 67452301
Register BB=B = EFCDAB89
Register CC=C = 98BADCFE
Register DD=D = 10325476
IM 4 - bit
Adding No. of bits to complete multiple of 512 (with IM and length of IM) bit in our Ex. (960-bit)
64 – bit length of IM
M
Multiple of 512 bits (in our EX. 1024)
512 bits
512 bits
……………………………
512 bits
Table ( )
1-32 bits
……
16-32 bits
… …
1-32 bits
… …
16-32 bits
I = 0 to 15
F (B,C,D) = [(B ∧ C) ∨ (¬ B ∧ D)
I = 16 to 31
F (B,C,D) = [(B ∧ D) ∨ (B ∧¬D)
I = 32 to 47
F (B,C,D) = [B ⊕ C ⊕ D]
I = 48 to 63
F (B,C,D) = [C ⊕ (B ∨¬ D)]
A=B+((A+F(B,C,D)Xj[k]+T[i]<<<s)
D=C; C=B; B=A; A=D
A=A+AA; B=B+BB; C=C+CC; D=D+DD
١١١١
(most) D C B A (least)

Figure (1) MD5 algorithm flowchart

Table(1) values of k, s and T[i].

I	k	s	T[i]	i	k	s	T[i]	i	k	s	T[i]	i	k	s	T[i]
0	0	7	d76aa478	1	1	12	e8c7b756	2	2	17	242070db	3	3	22	c1bdceee
4	4	7	f57c0faf	5	5	12	4787c62a	6	6	17	a8304613	7	7	22	fd469501
8	8	7	698098d8	9	9	12	8b44f7af	10	10	17	ffff5bb1	11	11	22	895cd7be
12	12	7	6b901122	13	13	12	fd987193	14	14	17	a679438e	15	15	22	49b40821
16	1	5	f61e2562	17	6	9	c040b340	18	11	14	265e5a51	19	0	20	e9b6c7aa
20	5	5	d62f105d	21	10	9	02441453	22	15	14	d8a1e681	23	4	20	e7d3fbc8
24	9	5	21e1cde6	25	14	9	c33707d6	26	3	14	f4d50d87	27	8	20	455a14ed
28	13	5	a9e3e905	29	2	9	fcefa3f8	30	7	14	676f02d9	31	12	20	8d2a4c8a
32	5	4	fffa3942	33	8	11	8771f681	34	11	16	6d9d6122	35	14	23	fde5380c
36	1	4	a4beea44	37	4	11	4bdecfa9	38	7	16	f6bb4b60	39	10	23	bebfbc70
40	13	4	289b7ec6	41	0	11	eaa127fa	42	3	16	d4ef3085	43	6	23	04881d05
44	9	4	d9d4d039	45	12	11	e6db99e5	46	15	16	1fa27cf8	47	2	23	c4ac5665
48	0	6	f4292244	49	7	10	432aff97	50	14	15	ab9423a7	51	5	21	fc93a039
52	12	6	655b59c3	53	3	10	8f0ccc92	54	10	15	ffeff47d	55	1	21	85845dd1
56	8	6	6fa87e4f	57	15	10	fe2ce6e0	58	6	15	a3014314	59	13	21	4e0811a1
60	4	6	f7537e82	61	11	10	bd3af235	62	2	15	2ad7d2bb	63	9	21	eb86d391

From equation (1), it's clear there are prime five variables this is *A*, *B*, *F*, *T*, and *X* each of them has been 32-bits. Then *A* have an initial value as showed above and then take its value from equation (1), and *B* also have initial value and take its other value from equation (2). *F* takes its value from expressions above inside the loop. *T* has values from Table (1). Then the important variable *X* also 32-bit and it represents the values from the IM-input massage, and by the loop will process all data. Then the output as in point 10 will be;

[$_{\text{(most)}}D\ \ C\ \ B\ \ A_{\text{(lest)}}$] 128 bits.

Figure (2) represents the VHDL program for MD5 implementation.

```
begin
foriin 0 to 31 loop
X(i):= BL((i*32)+31 downto(i*32));
end loop;
forj in 0 to 1 loop
AA:=A;BB:=B;CC:=C;DD:=D;
foriin 0 to 63 loop
if(i>=0) and (i<=15)then
F:=((B and C)or((not B) and D));
elsif(i>=16) and (i<=31)then
F:=((B and D)or(B and (not D)));
elsif(i>=32) and (i<=47)then
F:=(B xorC xorD);
elsif(i>=48) and (i<=63)then
F:=(C xor(B and (not D)));
end if;
TS(31-S(i)downto0):=T(i)(31 downtoS(i));
TS(31         downto(31-S(i)+1)):=T(i)((S(i)-1)
downto0);
A:=B+((A+F)+X(K(i)+j*16)+TS);
D:=C;C:=B;B:=A;A:=D;
end loop;
A:=A+AA;B:=B+BB;C:=C+CC;D:=D+DD;
end loop;
end Behavioral;
```

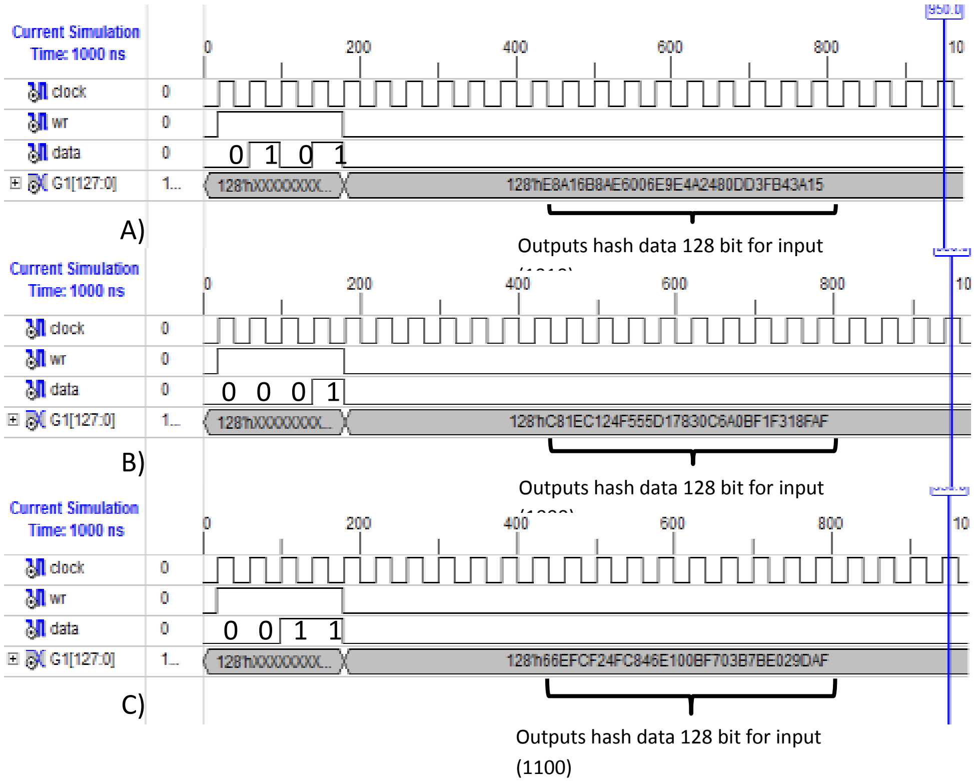
Figure (2) VHDL program for MD5 implementation.

Figure (3) represents the waveform output of the VHDL implementation. Sixteen examples for test the implementation and have been taken, nine of them have been recorded as in Figure (3) and Table (2).

Table (2) I/O of MD5 algorithm.

IM (4-bit) bi (hex.)	Fig.(3) part	Output (128-bit) hex. D C B A
1010 (A)	A	E8A16B8AE6006E9E4A2480DD3FB43A15
1000 (8)	B	C81EC124F555D17830C6A0BF1F318FAF
1100 (C)	C	66EFCF24FC846E100BF703B7BE029DAF
0101 (5)	D	BFD6451549DA235CF1AFAA5616E913A0
0111 (7)	E	C6B0279580B730FC99A3DFBE1DC2F620
0001 (1)	F	46948A71C95142044BBBD88A9DA758FC
0110 (6)	G	AAD981B6B55BA36ED7B4D35101EC5041
1110 (E)	H	7E354E768C5CE8AE4A05CB91D5481D01
1111 (F)	I	DFA9986D69A5B0046446C01636BC66F8

Note: input massages (IM) in Figure (3) from lest to most and in Table (2) from most to lest.


950.0
Current Simulation Time: 1000 ns
0
200
400
600
800
10
clock 0
wr 0
data 0
0 1 0 1
G1[127:0] 1...
128'hXXXXXXXX...
128'hE8A16B8AE6006E9E4A2480DD3FB43A15
A)
Outputs hash data 128 bit for input (1010)
Current Simulation Time: 1000 ns
0 0 0 1
128'hC81EC124F555D17830C6A0BF1F318FAF
B)
Outputs hash data 128 bit for input (1000)
Current Simulation Time: 1000 ns
0 0 1 1
128'h66EFCF24FC846E100BF703B7BE029DAF
C)
Outputs hash data 128 bit for input (1100)

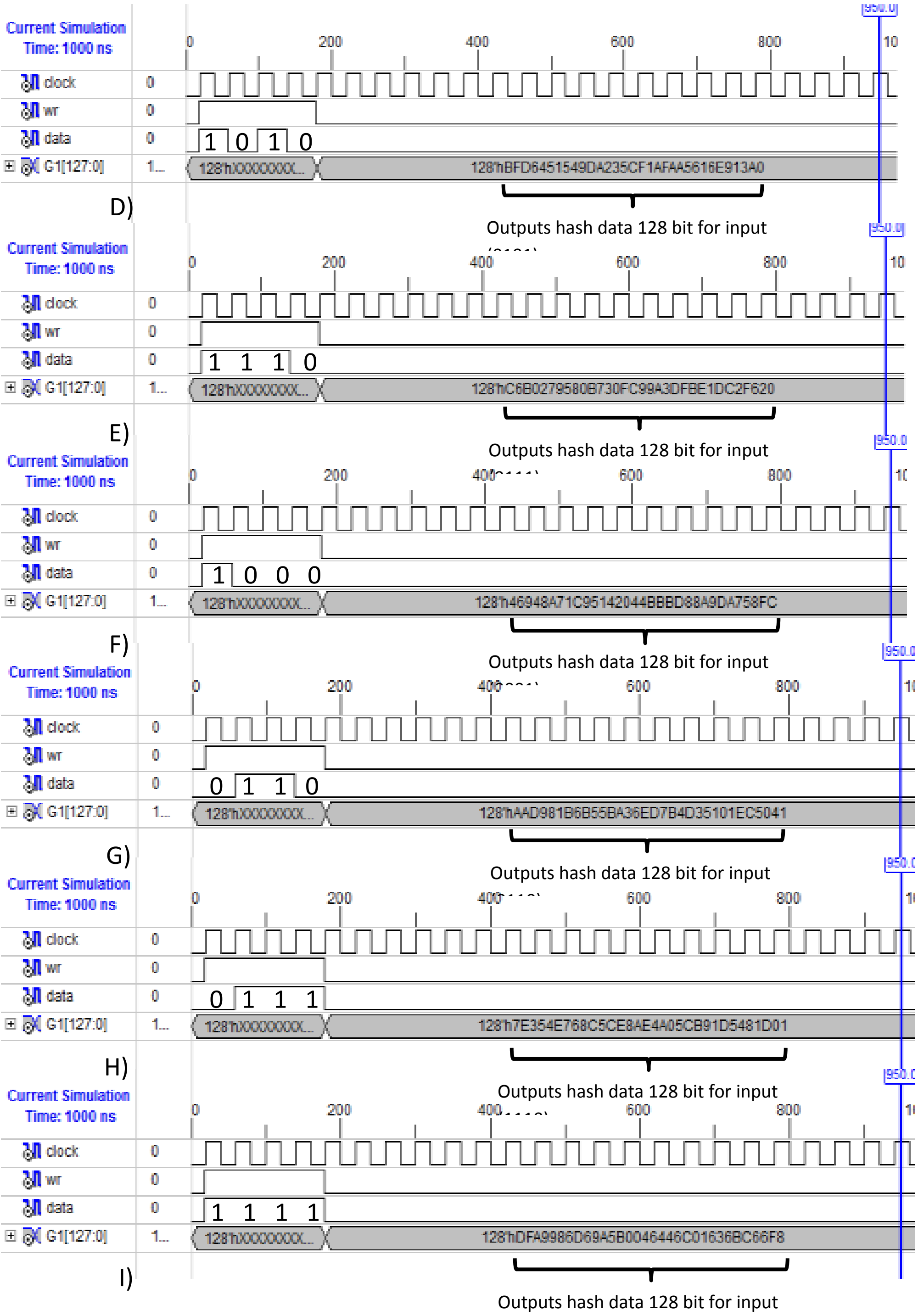
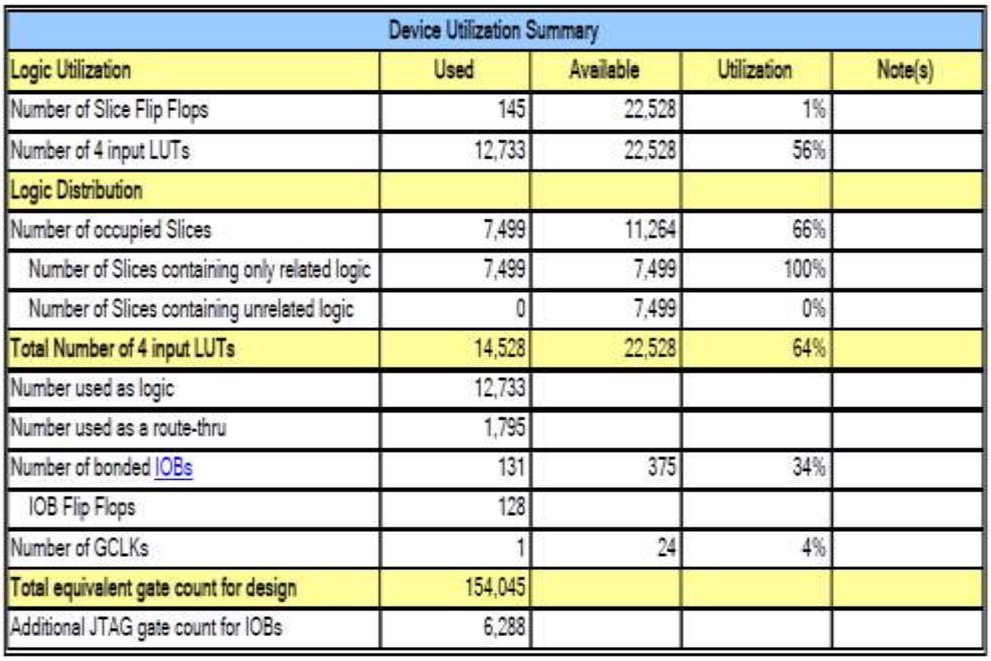

Current Simulation Time: 1000 ns
950.0
0
200
400
600
800
10
clock 0
wr 0
data 0
G1[127:0] 1...
1 0 1 0
128'hXXXXXXXX...
128'hBFD6451549DA235CF1AFAA5616E913A0
D)
Outputs hash data 128 bit for input
Current Simulation Time: 1000 ns
clock 0
wr 0
data 0
G1[127:0] 1...
1 1 1 0
128'hXXXXXXXX...
128'hC6B0279580B730FC99A3DFBE1DC2F620
E)
Outputs hash data 128 bit for input
Current Simulation Time: 1000 ns
clock 0
wr 0
data 0
G1[127:0] 1...
1 0 0 0
128'hXXXXXXXX...
128'h46948A71C95142044BBBD88A9DA758FC
F)
Outputs hash data 128 bit for input
Current Simulation Time: 1000 ns
clock 0
wr 0
data 0
G1[127:0] 1...
0 1 1 0
128'hXXXXXXXX...
128'hAAD981B6B55BA36ED7B4D35101EC5041
G)
Outputs hash data 128 bit for input
Current Simulation Time: 1000 ns
clock 0
wr 0
data 0
G1[127:0] 1...
0 1 1 1
128'hXXXXXXXX...
128'h7E354E768C5CE8AE4A05CB91D5481D01
H)
Outputs hash data 128 bit for input
Current Simulation Time: 1000 ns
clock 0
wr 0
data 0
G1[127:0] 1...
1 1 1 1
128'hXXXXXXXX...
128'hDFA9986D69A5B0046446C01636BC66F8
I)
Outputs hash data 128 bit for input

Figure (3) output results of MD5 algorithm with input massages.

We can note the satisfaction of the properties which stated in Message digest's section, where the implementation made for easy computes, and just now nearly impossible to compute the input messages from the digest output. Also, never to find the similarities between the all messages digest.

Another point which we can consider as a power point of our implementation, this is the use of the Xilinx IC XC3S1400A –FPGA. Where, in comparison with [3] and [4] the use of this IC represents best selection as in Table (3) which represents the devise utilization.

Table (3) Device Utilization.



Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	145	22,528	1%	
Number of 4 input LUTs	12,733	22,528	56%	
Logic Distribution				
Number of occupied Slices	7,499	11,264	66%	
Number of Slices containing only related logic	7,499	7,499	100%	
Number of Slices containing unrelated logic	0	7,499	0%	
Total Number of 4 input LUTs	14,528	22,528	64%	
Number used as logic	12,733			
Number used as a route-thru	1,795			
Number of bonded IOBs	131	375	34%	
IOB Flip Flops	128			
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	154,045			
Additional JTAG gate count for IOBs	6,288			

CONCLUSIONS

Compare with the normal execution using CPU and GPU, the FPGA Xilinx technology represents the best means to achieve the propose of MD5, for several reasons such as low power consumption, low cost, and the important factor to achieve the independence of action. Also, the ability to develop the design while maintaining the components and incorporating an improvement to the low level. Also, the select of the Xilinx IC XC3S1400A compared with other in Xilinx family represent good selection for this purpose.

REFERENCES

[1].Danyang C. andBingru Y., "Design and implementation forMD5-based data integrity checking system", IEEE 2nd International Conference on Information Management and Engineering, Pag. 608-611, 2010.

[2].Janaka D., Howard M. H. andVenkatesan R., " FPGA Implementation of MD5 Hash Algorithm", IEEE, Electrical and Computer Engineering Canadian Conference, 2001.

[3].Dongjing H. and Zhi X., " Multi-parallel Architecture for MD5 Implementations on FPGA with Gigabit-level Throughput", 2010 International Symposium on Intelligence Information Processing and Trusted Computing.

[4].Kimmo J., Matti T. and Jorma S., " Hardware Implementation Analysis of the MD5 Hash Algorithm", Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

[5].Hongwei W., Xiangnan L. and Weibin T., " A fast GPU –Based Implementation for MD5 Hash Reverse", IEEE, International Conference on Anti-Counterfeiting, Security and Identification, Pag. 13-16, 2011.

[6].Nguyen T. H., Kensuke N. and Yoshiyasu T., "Simpli¯edIPSec Protocol Stack for Micro Server", International Journal of Network Security, Vol.11, No.1, PP.46-54, July 2010.

[7].D. Kesavaraja , R. Balasubramanian and D. Sasireka, "Implementation of A Cloud Data Server (CDS) for Providing Secure Service in E-Business". IJDMS, Vol. 2, No. 2, May 2010.

[8].Guang H., Jianhua M., and Benxiong H., " High Throughput Implementation of MD5 Algorithm on GPU" IEEE Proceedings of the 4th International Conference on Ubiquitous Information Technologies and amp; Applications, Pag.1-5, 2009.

[9].Changxin L., Hongwei W., Shifeng C., Xiaochao L. and Donghui G., " Efficient Implementation for MD5-RC4 Encryption Using GPU with CUDA", IEEE 3rd International Conference on Anti-Counterfeiting, Security, and Identification in Communication, Pag. 167-170, 2009

[10].Ganesan R. and Vivekanandan K., " A Novel Hybrid Security Model for E-Commerce Channel", IEEE International Conference on Advances in Recent Technologies in Communication and Computing, 2009.