Architecture Design for SHA-1 Controller Core

Sruthi Prasad P V, Remya K Manohar, Nandakumar.R
VLSI Design Group,
National Institute of Electronics and Information Technology (NIELIT)
Calicut, Kerala, India
{sruthi94, remyaprasad282008, nanda24x7}@ gmail.com.

Abstract- **An important technique in information security is to verify that, whether the communicating entity is the one that it claims to be and have not been altered, is known as Message authentication. For securing the information against the unspecified attacks Cryptographic hash functions are very important. To compress and encrypt large messages into a smaller message digest cryptographic hash functions are used. SHA-1 (Secure Hash Algorithm – 1) is one such implementation of hash functions. A Controller having four input is used to control the entire functioning of SHA-1. A new area, power efficient and a high throughput SHA-1 Controller Core implementation is proposed in this paper, a deterministic algorithm had been introduced to fulfill this aim. The proposed design was modeled using Verilog HDL and also prototyped on FPFA. The functionality has been simulated and verified using a golden reference.**

Key words – SHA-1 Controller, cryptographic hash, VLSI Cryptosystem, power analysis, resource utilization

1. INTRODUCTION

SHA-1 was designed and published by the United States National Security agency and the NIST respectively as a U.S. Federal Information Processing Standard. SHA means "secure hash algorithm". Slight variation in the hashing algorithm "Merkle-Damgård Construction"[1]. was mainly used in SHA-1 .The Secure Hash Algorithm helps in creating a message digest (condensed of a message) that is infeasible to create without the message itself. SHA-0, SHA-1, SHA-2 and SHA-3 are existing SHA hash functions.SHA-1 is very similar to SHA-0, but certain corrections were made in SHA-0 to rectify its significant weakness. Among these SHA's SHA-1 is the most widely used one and is enrolled in wide spectrum of applications and utilities. SHA-1 is commonly used in several applications and protocols such as TLS, SSL, PGP, SSH, MIME, and IPSec. SHA-1 provides a message digest output of 160 bits when an input of , 2^{64} is given This message digest can be input into Digital Signature Algorithm (DSA) which verifies or generates the signature for the message, signing the message digest, which is smaller in size than the message improves the efficiency of the process. It is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.[2] ,hence SHA is said to be secure. Any change to a message input will, result in a different message digest, and the signature will fail to verify. These all make SHA useful for applications such as digital signatures to verify the authenticity of a message. The main advantage of the SHA -1 is, it's easy implementation on hardware. Hashes are "digests", not "encryption". Encryption is a two-way operation. That is, given a message m, an encryption algorithm 'enc' and its corresponding decryption algorithm 'dec', exists ; where dec (enc (m)) = m..In Hash functions there is no way to "decrypt" a hashed message. That is why it is said that hash functions are one-way operations

2. PRINCIPLE OF OPERATION

The various stages of the Controller can be explained by an FSM. In this controller basically there are three stages (i) IDLE (ii) ACTIVE (iii) TRANSIENT. When rst is in active high the entire controller will be in idle state. The rst in active low leads to the normal working of the controller. When clk, active low rst applied to the controller, if ready is high, it will go to the active state making initial =0 and valid =1,the DATA_IN will be pushed to DATA_OUT of the controller, thereby start functioning the entire circuit .

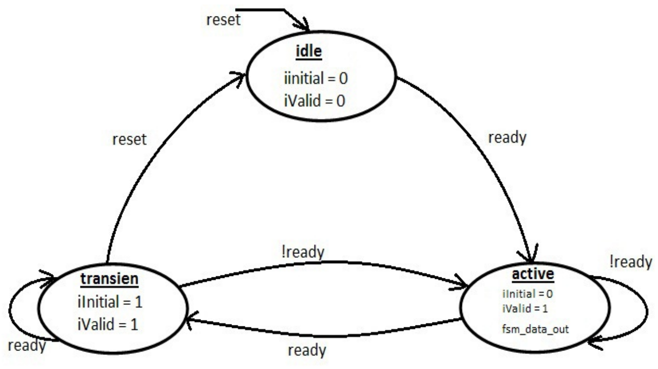


FIGURE 1: CONTROLLER FINITE STATE MACHINE

SHA undergoes five stages (i) initialization (ii) preprocessing (iii) processing (iv) chaining variables update (v) completion. The SHA-1 algorithm operates on 32-bit words, the intermediate calculation (hash) is computed from 512-bit blocks.512 blocks means 16 words. The output (message digest) is 160 bits, which is the condensed representation of the original message. [2]

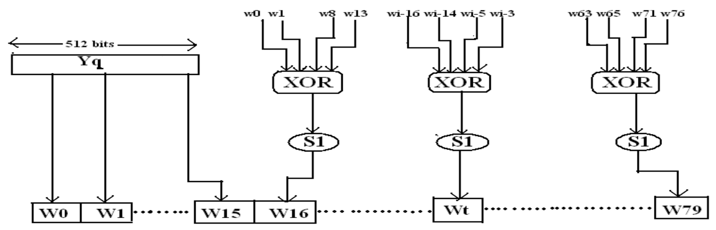


FIGURE 2: CREATION OF 80-WORD INPUT SEQUENCE

Initialization: The constants needed are generated in this stage. The constants include: an additive 32-bit constant, K_t its hexadecimal value is :

	0x5a827999	$0 \leq j \leq 19$
Kt =	0x6ed9eba1	$20 \leq j \leq 39$
	0x8f1bbcdc	$40 \leq j \leq 59$
	0xca62c1d6	$60 \leq j \leq 79$

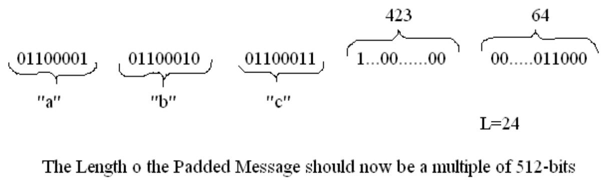
The initial Hash values are as follows:

H0 = 0x67452301
H1 = 0xefcdab89
H3 = 0x98badcfe
H4 = 0x10325476
H5 = 0xc3d2e1f0

The chaining variables H1 to H5, each is 32 bits are used in SHA-1.After each message block processing these chaining variables are updated.

Preprocessing: SHA-1 works on 512-bit blocks, at a time one block will be processed. The length of the message should be a multiple of 512, it is then segmented into 512-bit blocks. Hash computation take place only after preprocessing. Three steps are there in preprocessing: padding the message, parsing the padded message into message blocks and setting the initial hash value, H (0)

(a) Padding: To make the length of the message a multiple of 512 padding has to be done. “1” followed by zero bits array. This is done by appending a single bit of value `1' followed by zero bits, which is equivalent to 448 mod 512. 64 bit representation of the message length is appended to this 448 mod 512 to make it a multiple of 512.



(b) Parsing: Divide the padded message into N 512-bit blocks

Processing: In this each message block is segmented in to 16 words of 32 bits.

Word expansion: The 16 32-bit variables are expanded into 80 32-bit words.

Working variables initialization: The 32 bit working variables A,B,C,D and E are initialized to the current values of the chaining variables:

A = H1
B = H2
C = H3
D = H4
E = H5

Compression step:

T = ROTL5 (A) +f (B, C, D)
E = D
D = C
C = ROTL30 (B)
B = A
A = T

f(B, C, D) is the logical function applied to the 32-bit variables. SHA-1 has 4 rounds, each of 20 compression steps and each round uses the same logical function f .

	(B & C) ^ (~B & D)	$0 \leq j \leq 1$
f (B,C,D)=	B ^ C ^ D	$20 \leq j \leq 39$
	(B & C) ^ (C & D) ^ (B & D)	$40 \leq j \leq 59$
	B ^ C ^ D	$60 \leq j \leq 79$

Chaining variables update**:** chaining variables are updated such that

H1 = A + H1
H2 = B + H2
H3 = C + H3
H4 = D + H4
H5 = E + H5

Completion: A 160 bit message digest is produced by SHA-1 and is denoted by Hash,it is computed by concatenating the chaining variables after the last message block processing

Hash = H1 @ H2 @ H3 @ H4 @ H5

3. ARCHITECTURAL DESCRIPTION

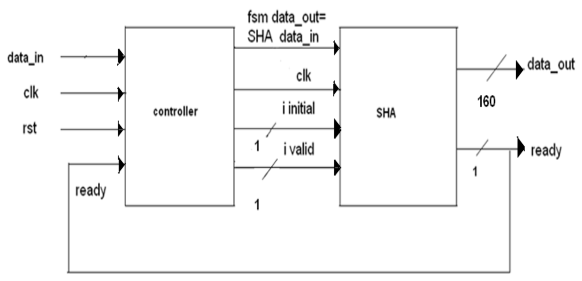


FIGURE 3: SHA-1 WITH CUSTOM CONTROLLER

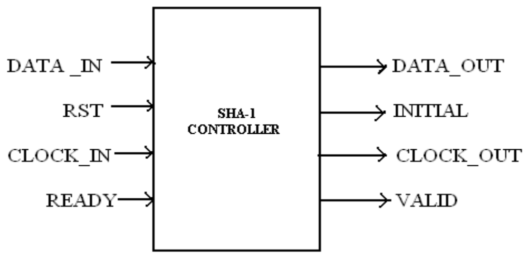


FIGURE 4: SHA-1 CONTROLLER-INPUT-OUTPUT DIAGRAM

For the purposes of this illustration, Controller part alone is taken into consideration, in figure 4. The SHA-1 Controller has 4 inputs and 4 outputs. The input pins clk; rst and ready will control the functioning of SHA-1. Idle, Active, Transient are the different states of the controller fsm. When rst is high the entire SHA will be in idle state. It starts functioning on active low reset.DATA_IN will be outputted only when ready is 1 thereby making initial=0 and valid=1, which is the working state of SHA-1.

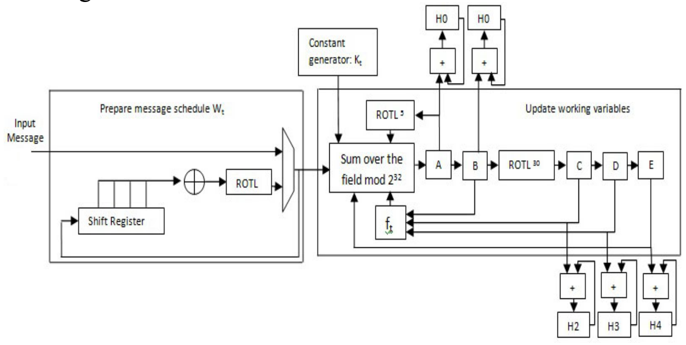


FIGURE 5: SHA1- BASIC BLOCK DIAGRAM

In sha-1 basic block, shown in figure 5, the initial message is properly padded and parsed. The 160-bit hash is initialized with five 32-bit words H0, H1, H2, H3 and H4. The message schedule is represented as 80 words W0, W1….. W79, the five working variable are represented by registers A–E, and T is temporary word. The working variables are updated with the current hash values H0–H4 where the initial hash values themselves are defined as constants in the SHA specification. For a total of 80 loops, the message schedule must generate a unique Wt, which is added to a function of the working variables over the finite field mod 2^{32}. The constant generator and the function f of B, C, and D are defined above. After 80 loops of the working variables, A–E is added to H0–H4, respectively, for the final hash value. Both the message schedule and the working variable updates operate in an iterative manner as shown in the block diagram. It is not until the hash is complete that a new hash can begin.. The functions f (B, C, and D) and Kt are implemented as muxes with appropriate outputs which are selected on iteration basis. The constant generator simply selects the constants that are predefined based on iteration. Modular addition is being handled automatically with a 32 bit register

FSM data_in	READY	PRESENT	NEXT	i initial	i valid	FSM data_out
(if rst=1)						
32'b1	0	00	00	x	x	32'bx
(if rst=0)						
32'b1	1	00	01	0	0	32'bx
32'b1	0	00	00	0	0	32'bx
32'b1	1	01	10	0	1	32'b1
32'b1	0	01	01	0	1	32'b1
32'b1	1	10	10	1	1	32'b1
32'b1	0	10	01	1	1	32'b1

TABLE 1: CONTROLLER STATE TABLE

READY	PRESENT STATE		NEXT STATE		OUTPUT		D0	D1
R	X0	X1	X0*	X1*	I	V		
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
0	0	1	0	1	0	1	0	1
1	0	1	1	0	0	1	1	0
1	1	0	1	0	1	1	1	0
0	1	0	0	1	1	1	0	1

TABLE 2: EXCITATION TABLE FOR CONTROLLER

From the excitation table, given above, equations for $i(q_0)$,$v(q_1)$, D_0 and D_1 can be calculated by doing it on truth table. The equations and the circuit structure are given below:

$$i = \overline{x1}\ x0$$

$$v = x1 + x0\ \overline{x1}$$

$$D0 = r\ x1 + r\ x0\ \overline{x1}$$

$$D1 = r\ \overline{x0}\ \overline{x1} + \overline{r}\ (x0 \oplus x1)$$

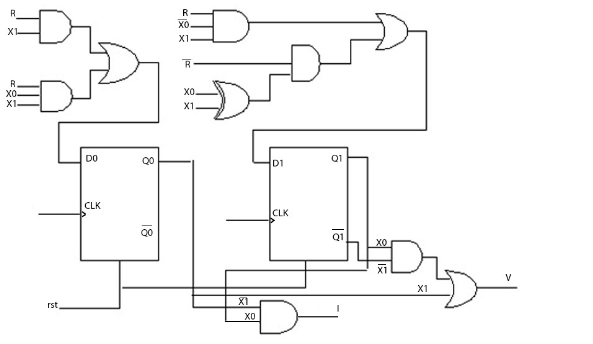


FIGURE 6: SHA1- CONTROLLER STRUCTURAL MODEL

4. FUNCTIONAL SIMULATION RESULTS

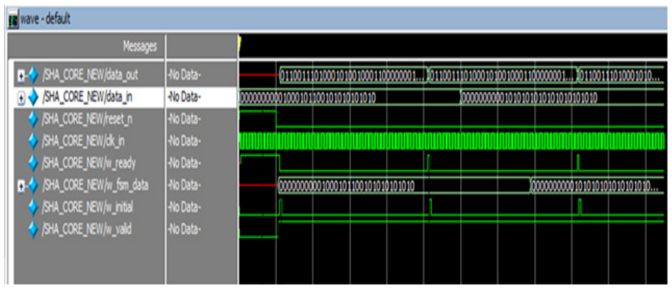

wave - default
Messages
/SHA_CORE_NEW/data_out -No Data-
/SHA_CORE_NEW/data_in -No Data-
/SHA_CORE_NEW/reset_n -No Data-
/SHA_CORE_NEW/clk_in -No Data-
/SHA_CORE_NEW/w_ready -No Data-
/SHA_CORE_NEW/w_fsm_data -No Data-
/SHA_CORE_NEW/w_initial -No Data-
/SHA_CORE_NEW/w_valid -No Data-
01100111010001010010001100000001...
0000000000100010110010101010101010
00000000001010101010101010101010
00000000001000101100101010101010
0000000000101010101010101010...

FIGURE 7: MODELSIM® SIMULATED OUTPUT

Note about simulation:

Controller has only 4 inputs.
When rst=1, the circuit will be in idle state itself
Clk,rst=0,for the fist clk ready will be 0 then ready =1,data_in will be sent to data_out making initial=0 , valid=1 which makes the SHA -1 to produce the corresponding hash code. Next input of the controller will be outputted only when ready=0 again this process continues

5. HARDWARE TEST RESULTS

Power dissipation and Resource utilization obtain on synthesizing SHA -1 in ALTERA®; Family: CYCLONE® II; Device: EP2C20F484C7, are given below

RESOURCE	USAGE
ESTIMATED TOTAL LOGIC ELEMENTS	1359
TOTAL COMBINATIONAL FUNCTIONS	911
LOGIC ELEMENTS USAGE BY NUMBER OF LUTs INPUTS	
4 INPUT FUNCTIONS	287
3 INPUT FUNCTIONS	385
2 INPUT FUNCTIONS	239
LOGIC ELEMENTS BY MODE	
NORMAL MODE	601
ARITHMETIC MODE	310
TOTAL REGISTERS	875
DEDICATED LOGIC REGISTERS	875
I/O REGISTERS	0
I/O PINS	194
MAXIMUM FAN-OUT NODE	clk_in
MAXIMUM FAN-OUT	875
TOTAL FAN-OUT	5811
AVERAGE FAN-OUT	2.93

TABLE 3: RESOURCE UTILIZATION

PARAMETERS	SHA CORE
TOTAL LOGIC ELEMENTS	1359
TOTAL COMBINATION FUNCTIONS	911
DEDICATED LOGIC REGISTERS	875
TOTAL REGISTERS	875
TOTAL PINS	194
TOTAL VIRTUAL PINS	0
TOTAL MEMORY BITS	0
TOTAL PLLs	0

TABLE 4: EDA COMPILATION REPORT

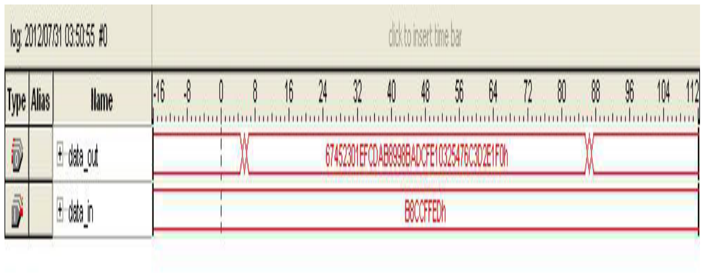


FIGURE 8:PROTOTYPE TEST RESULT

6. CONCLUSION

Cryptographic hash functions are an interesting and challenging class, in the area of communication security. SHA-1 is the second most common data integrity check standard used throughout the world today. SHA-1 is faster than its successor and is widely used in file transfer technology. From the data obtained experimentally and the subsequent analysis of the performance parameters that are recorded [4], it can be conclusively said that this implementation shows its better characteristics as compared to the other Hashing Algorithm.

7. REFERENCES

[1] FIPS PUB 180-1, Secure Hash Standard (SHA-1),National Institute of Standards and Technology (NIST), 1995

[2] Y.K. Kang, D.W. Kim, T.W. Kwon, and J.R. Choi. An Efficient Implementation of Hash Function Processor for IPSEC. Proceedings of **the IEEE Asia-Pacific Conference** on **ASIC**, pages 93 – 96, August 6 – 8, 2002.

[3] N. Sklavos, G. Dimitroulakos, and O. Koufopavlou. An Ultra High Speed Architecture for VLSI Implementation of Hash Functions. Proceedings of the 10th **IEEE International Conference on Electronics, Circuits and Systems** (ICECS 2003), Sharjah, United Arab Emirates, 3:990 – 993, December 14 – 17, 2003.

[4] Dan Cao, Jun Han*, Xiao-Yang Zeng "A Reconfigurable and Ultra Low_ cost VLSI Implementation Of SHA-1 and MD5 Functions, Proceedings of the 7th International Conference ASICON, 2007, Page(s): 862 – 865 .DOI: 10.1109/ICASIC.2007.4415767