Modified Architecture designing of SHA-256 Algorithm through Verilog HDL approach

A Project Report

Submitted by:

Vaibhav Pal (18-1-3-068)

Satyendra Kumar Trivedi (18-1-3-089)

in partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

ELECTRICAL ENGINEERING

at

DEPARTMENT OF ELECTRICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR,

SILCHAR, ASSAM (INDIA)-788010

MAY 2022

DECLARATION

I hereby declare that the project entitled **"Modified architecture designing of SHA-256 algorithm through Verilog HDL approach."** Submitted for the B. Tech. (EE) degree is my original work and the project has not formed the basis for the award ofany other degree, diploma, fellowship, or any other similar titles.

Signature of the Student:

Vaibhav Pal
(1813068)

Satyendra Kumar Trivedi
(1813089)

Place: Silchar, Assam

Date: 10/05/2022

CERTIFICATE

This is to certify that the project titled **"Modified architecture designing of SHA-256 algorithm through Verilog HDL approach"** is the bonafide work carried out **by Vaibhav Pal(18-1-3-068)** and **Satyendra Kumar Trivedi(18-1-3-089),** students of B.Tech (EE) of National Institute of Technology Silchar (An Institute of National Importance under MHRD, Govt. of India), Silchar, Assam, India during the academic year 2018-22, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Electrical Engineering) and that the project has not formed the basis for the award previously of anyother degree, diploma, fellowship or any other similar title.

Place : Silchar
Date : 10/05/22

Name and Signature of Supervisor

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our supervisor **Dr. Nabanita Adhikary**, Assistant Professor, Department of Electrical Engineering, NIT Silchar for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him from time to time shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the faculty members of the Department of Electrical Engineering, panel members **Prof. Binoy Krishna Roy,Dr. Tapan Pradhan and Dr. Risha Mal**, NIT Silchar for their valuable information and guidance, which helped us in learning the nuances of the project work and in completing this task through various stages.

ABSTRACT

Ongoing utilizations of advanced correspondence frameworks are quickly expanding. Because of this there is an enormous interest for an undeniable degree of safety. In cryptographic calculations, Secure Hash Algorithm (SHA-256) has gotten an indispensable part in numerous applications. The equipment execution of the SHA-256 hash calculation is actually isolated from the fundamental processor and subsequently, it has more security and is better than the product execution. An execution of a hash calculation on FPGAs is advantageous, as it is adaptable and effectively upgradable. Notwithstanding, execution of this calculation on equipment has been testing, because of the interest of high preparing speed. In this project, an improved SHA-256 hash work has been executed in equipment HDL Verilog language and that can be integrated in FPGA. Initially, the basic cryptographic algorithms like data encryption standard (DES) and advanced encryption standard (AES) is simulated using Verilog HDL and their output results is analyzed and studied. In the SHA-256 calculation, the compressorr and expander block of the hash work are altered. The acquired outcome shows a critical improvement in the presentation of the proposed SHA-256 calculation and it is contrasted and existing different structures. Its greatest clock recurrence is 170.75 MHz, throughput of 1344.98 Mbps and an improved proficiency of 2.2 Mbps/cut.

TABLE OF CONTENTS

1. INTRODUCTION

PROBLEM DEFINITION

In this day and age, there is a broad use of digital communication frameworks for performing wide scopes of constant tasks. However, with the increment in use of this framework, potential danger has been expanded. Thus, safety efforts must be considered as a significant part in a digital communication framework. Numerous cryptographic calculations need to guarantee security. In bitcoin mining, large softwares and hardwares are required which consumes a large amount of power. This power consumption needs to be reduced. Whenever a person does a money transaction that also needs more security.

PROBLEM OVERVIEW

In this day and age, there is a broad utilization of computerized correspondence frameworks for performing wide scopes of genuine time tasks. Yet, with the increment in utilization of this framework, potential danger has been expanded. Consequently, safety efforts must be considered as a significant part in an advanced correspondence framework. Numerous cryptographic calculations are created to guarantee security. One of the incredible cryptographic calculations is Hash calculation. Hash capacities take input information of subjective length and convert them into a few fixed information, called as hash work or message digest. Hash work yields the quirky connection between the information message of self-assertive length and hash esteem and henceforth, portrays the more drawn out message in a concise manner. Different Cryptographic algorithms like DES and AES are largely used for the data encryption and decryption purpose.
The Implementation of FPGA based Cryptographer is much faster than the Software based Cryptographer which works on general Processor. Nowadays there is a big race of fast cryptographers due to the Cryptography of Digital Currency like Bitcoin which is based on Secure Hash Algorithm (SHA-256) . FPGA based Cryptographers and Design provides an upper edge. Our design ensures that the user gets cheap and fast hardware for this kind of application. The Importance of SHA-256 can be determined by the fact that it is required to be used by the Law of the United States in some Government Secure Applications.

SOFTWARE SPECIFICATIONS

The software and other hardware descriptive language which we will be using for the simulation and synthesis purpose in project to get the proposed architecture of SHA-256 algorithm are:

1. Verilog HDL
2. Xilinx Vivado

Verilog HDL

Verilog, normalized as IEEE 1364, is an equipment portrayal language (HDL) used to display electronic frameworks. It is most generally utilized in the plan and confirmation of computerized circuits at the register-move level of deliberation. It is additionally utilized in the check of simple circuits and inconsistent message circuits, just as in the plan of hereditary circuits.[1] In 2009, the Verilog standard (IEEE 1364-2005) was converged into the System Verilog standard, making IEEE Standard 1800-2009. From that point forward, Verilog is formally essential for the System Verilog language.

Xilinx Vivado

Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of HDL designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis. Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE). Vivado was introduced in April 2012, and is an integrated design environment (IDE) with system-to-IC level tools built on a shared scalable data model and a common debug environment. Vivado includes electronic system level (ESL) design tools for synthesizing and verifying C-based algorithmic IP; standards based packaging of both algorithmic and RTL IP for reuse; standards based IP stitching and systems integration of all types of system building blocks; and the verification of blocks and systems. Different codes can be simulated and synthesized in it using Verilog HDL.

2. LITERATURE SURVEY

Before starting the project we went through the various papers published by the authors for the implementation of data encryption standard algorithm, advanced standard encryption algorithm and the SHA- 256 algorithm. The work done by them, their output parameters , advantages and disadvantages are analysed. The detailed study of the previous work done by the authors can be seen below:

Finding an Efficient FPGA Implementation of the DES Algorithm to Support theProcessor Chip on Smartcard, 1st Veronica Ernita Kristianti Dept. of Electrical Engineering Gunadarma University Depok, Indonesia, 2018

In this paper, they proposed to get the best DES computation to apply on System on Chip (SoC). The examination was performed by taking a gander at between the 16-round pipelines DES estimations which is the general DES computation with the 8-round pipeline DES estimation which is the outcome of viability. The outcome of the ordinary examination of the overall resources required by all of the DES computations took a gander at was that the 16-round DES requires a typical of 21.2% of the resources, while the 8-round DES requires an ordinary of simply 9.7%. This showed that the 8-round DES pipeline computation is the great gainful DES estimation to apply on SoC as a data and information security system.

Implementation of AES Algorithm Using Verilog Ganesh Gopal Shet , Jamuna V, Shravani S, Nayana H G, Pramod Kumar S J N N College of Engineering, Shimoga, 30 November 2020

The primary goal of this paper was to execute AES calculation utilizing Verilog and to give an ideal circuit with clock recurrence, way delay, time needed to create keys and interpreting the information. For secure correspondence, cryptography is helpful within the sight of outsiders. It for the most part manages dissecting conventions and conquers the impact of data on security. The pupils like math, software engineering and electrical designing are examined by present day cryptography. The equipment execution of AES gives quicker speed than programming execution like secure key in encryption and higher throughput. Their work has been reached out to build the security for more serious assaults since the encryption time has been diminished. There has been further degree to advance the use of assets. The execution can be additionally improved to accomplish more productive use of the assets and increment the most extreme clock recurrence. The key length can be decreased, keeping up a similar security, to enhance their source use.

Designing of AES Algorithm using Verilog, SOUMYA V H VLSI design and Embedded System Visvesvaraya Technological University, PG centre, Belgaum, India, 2018

In this paper they proposed a paper where AES technique was carried out by the utilization of Verilog utilizing Xilinx ISE 14.7, which lessens activity time and clock cycles required for encode furthermore, interpret the message, whenever contrasted and execution utilizing VHDL.

AES has more private contrasts and DES, in light of its key size. It incorporates two fundamental modules, in which every one of the sub modules are called by module call technique. In utilization of inserted frameworks it improves safety efforts.

Design of High-Throughput SHA-256 Hash Function based on FPGA, Shamsiah binti Suhaili, Takahiro Watanabe System LSI Graduate School of IPS, Waseda University Kitakyushu-shi, Fukuoka, 808-0135 Japan

In this paper, another SHA family is created and planned to satisfy the cryptographic calculation execution necessity. SHA-256 plan and SHA-256 unfurling configuration dependent on reconfigurable equipment have been effectively finished utilizing Verilog code. These plans were mimicked and checked utilizing ModelSim. The outcomes from the paper showed that the proposed SHA-256 unfurling configuration gave better execution on Arria II GX as far as throughput. The high throughput of SHA-256 unfurling configuration was obtained at an information move speed of 2429.52 Mbps. The proposed SHA-256 plan can be applied toKeyed-hash Message Authentication Codes (HMAC) in future.

Implementation of SecureHash Algorithm-256 Kirubakaran.T ,Srinivasan.K,Venkatesh.K, Vignesh Jagan.P, Vishnu Varman.A Assistant Professor1 ,student, Department of ECE Sri Shakthi Institute of Engineering and Technology, Coimbatore, India, 2017

In this paper, an effective and minimized FPGA processor for the SHA-256 calculation was planned. The tale processor design depended on a custom datapath that adventures the reusing of modules, having as primary segment a 4-input Arithmetic-Logic Unit not recently announced. This ALU was planned because of considering the kind of tasks in the SHA calculation, their execution succession and the related dataflow. The processor equipment engineering was demonstrated in VHDL and executed in FPGAs. The outcomes acquired from the execution in a Virtex5 gadget showed that the proposed configuration utilizes less assets accomplishing better and effectiveness, outflanking past approaches in the writing zeroed in on reduced plans, saving around 60% FPGA cuts with an expanded throughput (Mbps) and proficiency (Mbps/Slice). The proposed SHA processor was appropriate for applications like Wi-Fi, TMP (Trusted Mobile Platform), and MTM (Mobile Trusted Module), where the information move speed is around 50 Mbps.

Optimisation of the SHA-2 Family of Hash Functions on FPGAs Robert P. McEvoy, Francis M. Crowe, Colin C. Murphy and William P. Marnane Department of Electrical & Electronic Engineering, University College Cork, Ireland

In this paper the creator had explored enhancement methods that had as of late been proposed in the past writing. Another VLSI design for the SHA-256 and SHA-512 hash capacities was introduced, which consolidates two mainstream equipment enhancement methods, to be specific

pipelining and unrolling. The SHA processors were created for execution on FPGAs, along these lines permitting quick prototyping of a few plans. Speed/region results from these processors were dissected and appeared to contrast well with other FPGA-based executions, accomplishing the quickest information throughputs in the writing to date. The new plan was then applied to both the SHA-256 and SHA-512 calculations by building VHDL modules for SHA processors. These full processors incorporate equipment for the SHA center, message scheduler, and message padder, subsequently working with simple reuse in a VLSI SoC climate.

A compact FPGA-based processor for the Secure Hash Algorithm SHA-256 q Rommel García , Ignacio Algredo-Badillo , Miguel Morales-Sandoval , Claudia Feregrino-Uribe , René Cumplido, Universidad del Istmo, Tehuantepec, Oaxaca, México, 2013

In this paper an effective and minimized FPGA processor for the SHA-256 calculation was planned. The tale processor design depended on a custom datapath that adventures the reusing of modules, having as primary segment a 4-input Arithmetic-Logic Unit not recently announced. This ALU was planned because of considering the kind of tasks in the SHA calculation, their execution succession and the related dataflow. The processor equipment engineering was demonstrated in VHDL and executed in FPGAs. The outcomes acquired from the execution in a Virtex 5 gadget showed that the proposed configuration utilizes less assets accomplishing better and effectiveness, outflanking past approaches in the writing zeroed in on reduced plans, saving around 60% FPGA cuts with an expanded throughput (Mbps) and proficiency (Mbps/Slice). The proposed SHA processor was appropriate for applications like Wi-Fi, TMP (Trusted Mobile Platform), and MTM (Mobile Trusted Module), where the information move speed is around 50 Mbps.

Improving SHA-2 hardware implementations, Recardo Chaves, Georgi Kuzmanov, Leonel Sousa, 8th international conference on cryptographic Hardware and Embedded Systems, 2006

In this paper the author proposed a bunch of new procedures to improve the execution of the SHA-2 hashing calculation. These strategies comprised generally in activity rescheduling and equipment reutilization, permitting a huge decrease of the basic way while the necessary territory likewise diminished. Both SHA256 and SHA512 hash capacities have been executed and tried in the VIRTEX II Pro prototyping innovation. Test results recommended upgrades to related SHA256 workmanship above half when contrasted with business centers and 100% with the scholarly community craftsmanship, or more 70% for the SHA512 hash work. The subsequent centers were equipped for accomplishing similar throughput as the quickest unrolled structures with 25% less region occupation than the littlest proposed designs. The proposed centers accomplished a throughput of 1.4 Gbit/s and 1.8 Gbit/s with a cut necessity of 755 and 1667 for SHA256 and SHA512 separately, on a XC2VP30-7 FPGA.

3.DESIGN AND ANALYSIS

Testing simulation of basis encryption algorithms

Firstly we have tried to write the Verilog code for the basic cryptographic algorithms and analyse the output simulation of the algorithms. We have analysed Data Encryption Standard (DES) and Advanced Encryption Standard (AES). After analyzing the output of these algorithms we optimized the SHA- 256 Algorithm.

Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a symmetric-key calculation for the encryption of computerized information. In spite of the fact that its short key length of 56 pieces makes it excessively uncertain for applications, it has been profoundly persuasive in the progression of cryptography. Information encryption is only one weapon in the network protection arms stockpile, yet it's one of the most established and generally utilized. Also, since no conversation about information encryption is finished without discussing DES.DES is a square code, and encodes information in squares of size of 64 digit each, implies 64 pieces of plain content goes as the contribution to DES, which produces 64 pieces of code text. A similar calculation and key are utilized for encryption and decoding, with minor contrasts. The key length is 56 pieces. In this way, the disposing of each eighth piece of the key creates a 56-digit key from the first 64-cycle key.

The algorithm breaks into the accompanying advances:

1. The cycle starts with the 64-bit plain content square getting given over to an underlying stage (IP) work.

2. The underlying change (IP) is then performed on the plain content.

3. Then, the underlying stage (IP) makes two parts of the permuted block, alluded to as Left Plain Text (LPT) and Right Plain Text (RPT).

4. Each LPT and RPT goes through 16 rounds of the encryption interaction.

5. At last, the LPT and RPT are rejoined, and a Final Permutation (FP) is performed on the recently consolidated square.

6. The aftereffect of this cycle creates the ideal 64-digit ciphertext.

7.

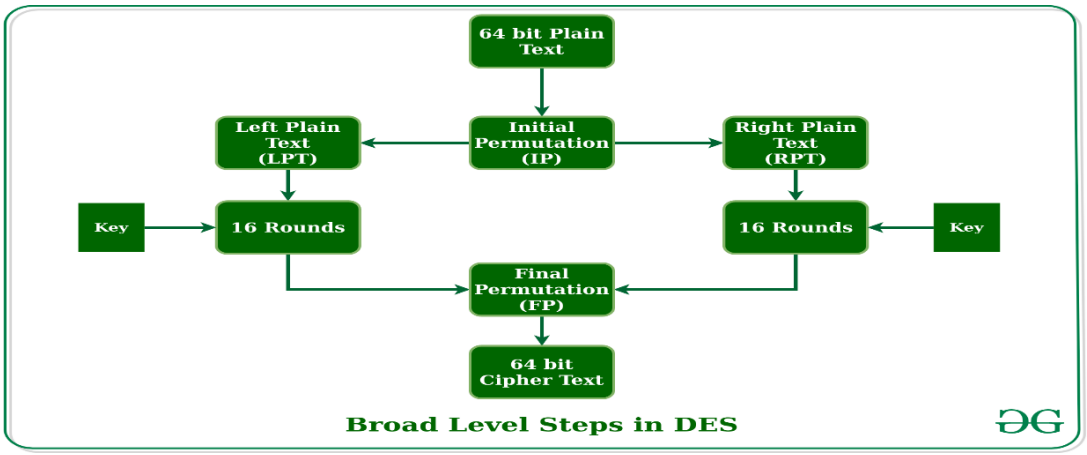
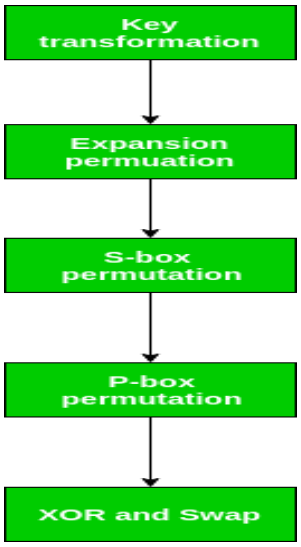


Fig.1 Block diagram of DES

The encryption process can be further broken down into five stages as shown in below figure:

1. Key transformation
2. Expansion permutation
3. S-Box permutation
4. P-Box permutation
5. XOR and swap



Advanced Encryption Standard

- AES represents Advanced Encryption Standard and is a significantly utilized symmetric encryption calculation. It is mostly utilized for encryption and assurance of electronic information. It was utilized as the substitution of DES(Data encryption standard) as it is a lot quicker and better than DES. AES comprises of three square codes and these codes are utilized to give encryption of information.

- It has an data block of 16 bytes and the key size is a variable of 128 bits,192 bits and 256 pieces. In our plan, AES technique is executed by the utilization of Verilog utilizing Xilinx Vivado which diminishes activity time and clock cycles required for encode and disentangle the message, whenever contrasted and execution utilizing VHDL. AES has more private contrasted and DES, in view of its key size.

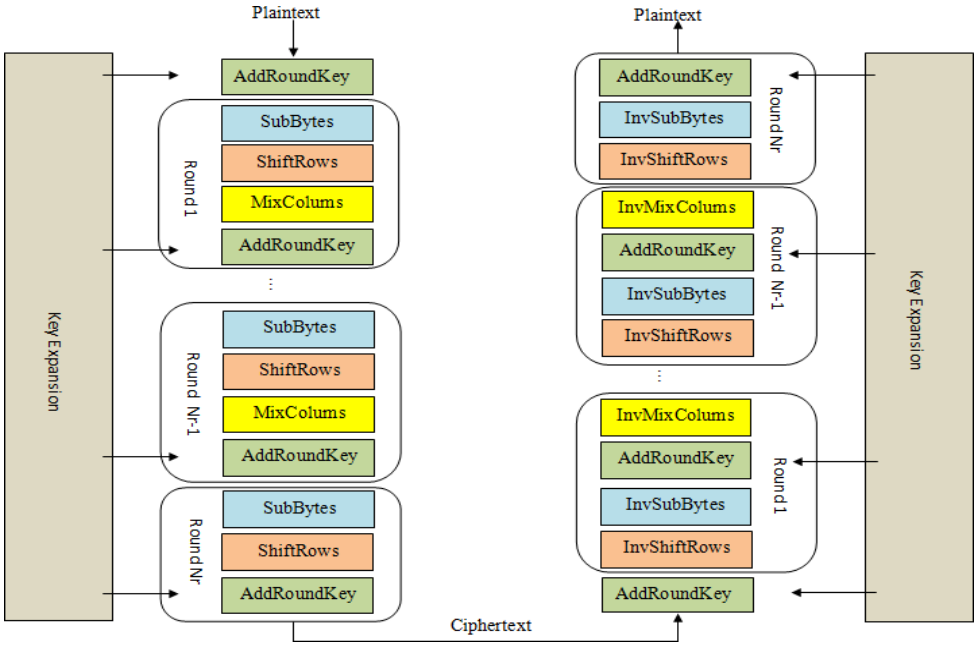


Fig.2 Encryption and Decryption in AES

Encryption in AES

The encryption period of AES can be broken into three stages: the initial round, the principle rounds, and the last round. The entirety of the stages utilize similar sub-activities in various mixes as follows:

- Initial Round: AddRoundKey
- Principle Rounds: SubBytes, ShiftRows, MixColumns, AddRoundKey
- Last Round: SubBytes, ShiftRows, AddRoundKey

The main rounds of AES are repeated a set number of times for each variant of AES. AES-128 uses 9 iterations of the main round, AES-192 uses 11, and AES-256 uses 13.

AddRoundKey

The AddRoundKey activity is the solitary period of AES encryption that straightforwardly works on the AES round key. In this activity, the contribution to the round is selective-ored with the round key.

SubBytes

- The SubBytes phase of AES involves splitting the input into bytes and passing each through a Substitution Box or S-Box. Unlike DES, AES uses the same S-Box for all bytes.

- At the encryption side we interpret the byte as 2 hexadecimal digits. 1st hexadecimal digit- Row. 2nd hexadecimal digit- Column. Transformation is done one byte at a time.

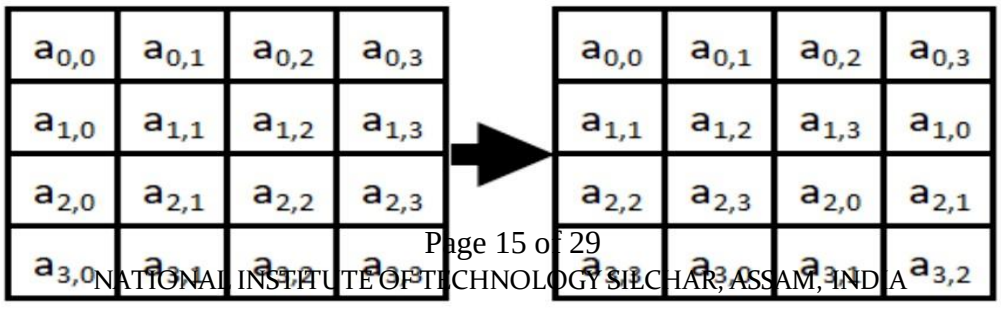
	0	1	2	3	4	5	6	7	8	9	0a	0b	0c	0d	0e	0f
0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
40	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig.3 Substitution Block

- In the ShiftRows phase of AES, each row of the 128-bit internal state of the cipher is shifted.

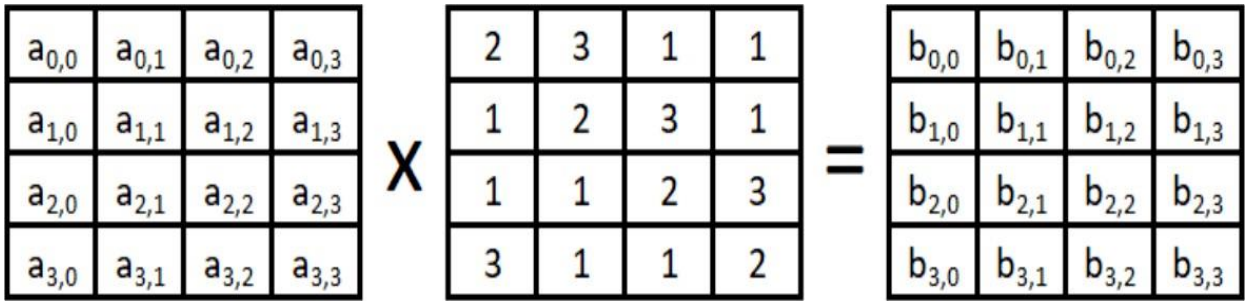
- The rows in this stage refer to the standard representation of the internal state in AES, which is a 4x4 matrix where each cell contains a byte. Bytes of the internal state are placed in the matrix across rows from left to right and down columns.

- In the ShiftRows operation, each of these rows is shifted to the left by a set amount: their row number starting with zero. The top row is not shifted at all, the next row is shifted by one and so on.



MixColumns

- Like the ShiftRows phase of AES, the MixColumns phase provides diffusion by mixing the input around. Unlike ShiftRows, MixColumns performs operations splitting the matrix by columns instead of rows.
- Each word is multiplied with 4*4 matrices and stored in a new column and so on for other words.
- **AddRoundKey:** output matrix is XOR with the 4 words of round key.



Decryption in AES

- To decrypt an AES-encrypted ciphertext, it is necessary to undo each stage of the encryption operation in the reverse order in which they were applied. The three stage of decryption are as follows:

  Inverse Final Round: AddRoundKey, ShiftRows, SubBytes

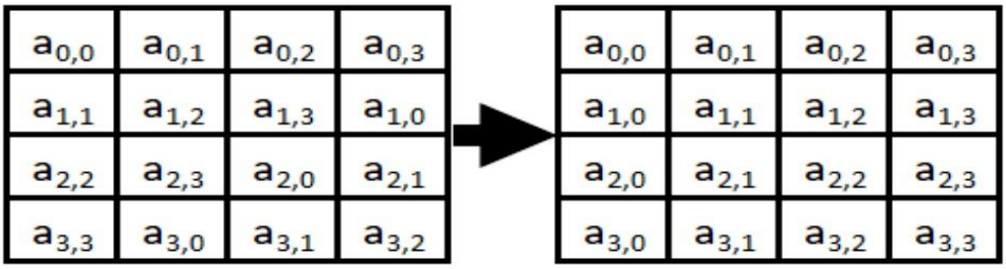
  Inverse Main Round: AddRoundKey, MixColumns, ShiftRows, SubBytes

  Inverse Initial Round: AddRoundKey

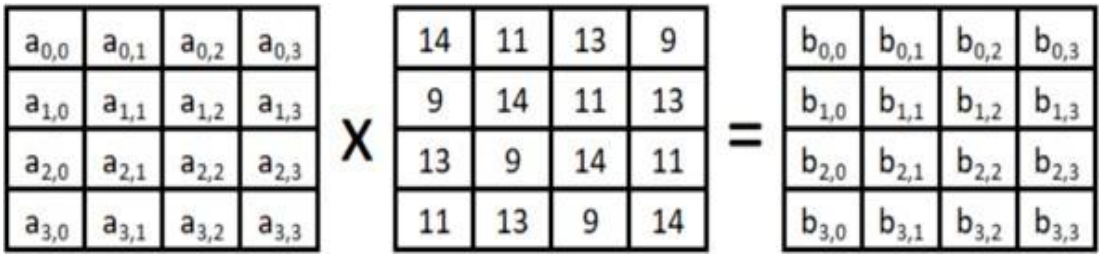
- Of the four operations in AES encryption, only the AddRoundKey operation is its own inverse (since it is an exclusive-or). To undo AddRoundKey, it is only necessary to

expand the entire AES key schedule (identically to encryption) and then use the appropriate key in the exclusive-or.

The other three operations require an inverse operation to be defined and used.The Inverse ShiftRows operation is identical to the ShiftRows operation except that rotations are made to the right instead of to the left. This is illustrated in the Figure below.



The last inverse operation to define is MixColumns. Like MixColumns, Inverse MixColumns can be defined as the matrix multiplication in Galois Field 28. This is illustrated in the Figure below.



Secure Hash Algorithm (SHA - 256)

- No key is utilized in this calculation. A fixed length hash esteem is processed according to the plain content that makes it outlandish for the substance of the plain content to be recuperated.
- Hash functions are also used by many operating systems to encrypt passwords.
- It is a secure hash algorithm.
- It is a mathematical function that is used to convert data to a string of 256 bits hash.

SHA- 256 Working

1. Arbitrary length message string ‘M’
2. Convert it to binary.
3. Preprocessing Stage:
    a. Padding a message M
    b. Parsing the Message M
    c. Setting initial hash values H0…..H7
4. Hash Computation
    a. Prepare the message schedule.
    b. Initialize a,b,c,d,e,f,g and h
    c. Compute the intermediate hash values
5. Append hash values H0, H1, H2, H3, H4, H5, H6, H7
6. 256-bits Message digest.

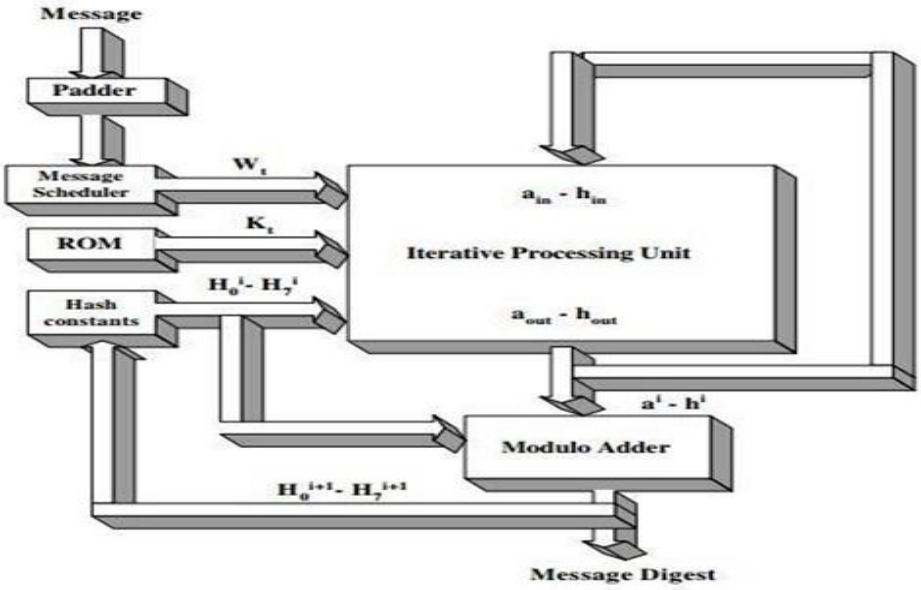


Fig.4 Block Diagram of SHA- 256

First Module in SHA-256 is the Padding and Parser Module.

1. **Padding and Parser:**

- First message is written in binary form then it is padded by adding 1 to it. Then 423 zeros is added and length becomes 448 bits.
- This module includes the extra data after the original data has stopped to make total data for SHA-256 ‘s next modules a 1_Block (512 bits Block).
- The Parser in this module stores all of this data after adding a 64-bit chunk which represents the Size of the Input data. This module also issues the “padding_done” to signal the other modules that padding and parsing has been done. Total length becomes 512 bits.

2. **Message Scheduler :**

The cushioned message in hex representation is presently parsed into 16 blocks,In message scheduler we execute the 0-15 cycles for 32bit 'w' words which is all the 1_Block information from cushioning (32*16=512 pieces). After 16 'w' words we perform cycles from effectively got 16 words to make further 48 'w' words.

For all out 64 rounds:

Message plan, {Wt}: for $0 \leq t \leq 15$, Wt =Mt

for $16 \leq t \leq 63$, $Wt = \sigma (Wt-2) + Wt-7 + \sigma (Wt-15) + Wt-16$

3. Iterative Processing :

Hash values are initialised here.In iterative processing we create 8 registers each 32bit: a, b, c, d, e, f, g, h with initial values equal to 8 intermediate initial values

H-0 = 32'h6b09d667

H-1 = 32'hbc67be85

H-2 = 32'b3e6cf372

H-3 = 32'ha54ff53a

H-4 = 32'h510f527e

H-5 = 32'h9a05688b

H-6 = 32'h1e83d9ab

H-7 = 32'h5bf0cd19

After this the compression is applied

seg_1 <= (e ROTR 6) ^ (e ROTR 11) ^ (e ROTR 25)

ch <= (e and f) ^ ((~ e) and g)

Maj<= a and b and c

T1 <= h +semation_1 + ch + k[n] + w[n]

T2 = semation_0 + maj

h <= g g <= f f <= e e <= d + T1 d <= c c <= b b <= a a <= T1 + T2

Where K's values are pre-defined by FIPS (Federal Information & Processing Standards)

4. Message Digest :

In message digest we perform addition of Initial values of "H" and values of 8 H after last round.After addition we concatenate values of 8 'H' each 32bit giving the out of 256 bits Hash. This is the final output.

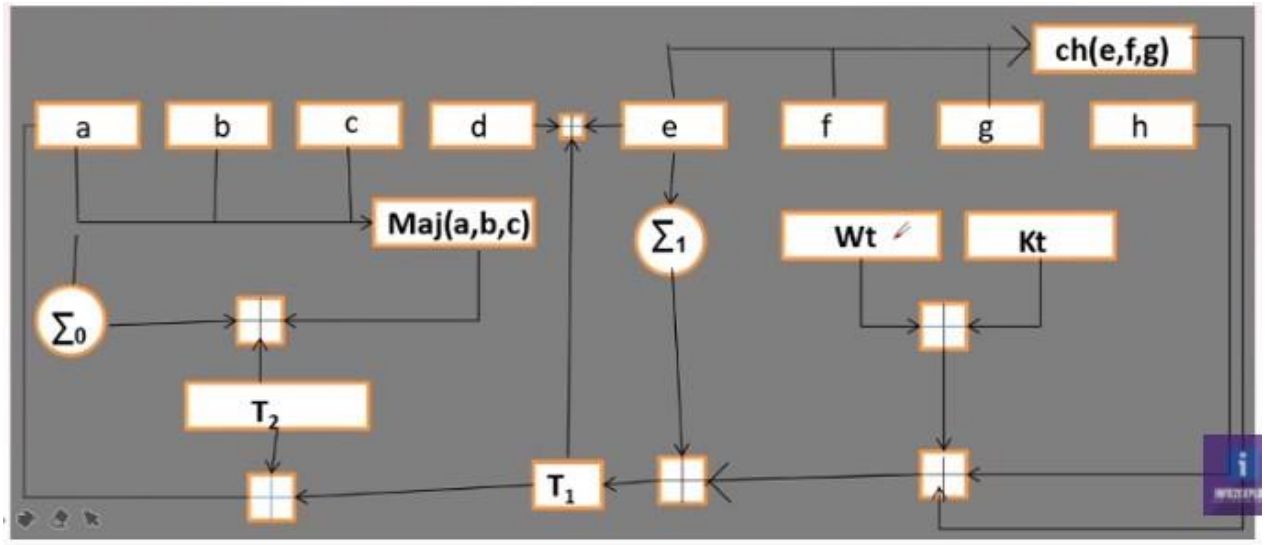


Fig 5. Block Diagram of equation used in algorithm

Modified Algorithm

The expander and the compressor are the two important blocks of the proposed SHA-256 algorithm.
We have modified the two blocks so that speed, frequency, efficiency and throughput of the output can be improved.

Expander Block

The expander is nothing but the message schedule preparation stage as shown in Eq.

$$W_t = \begin{cases} D_t^i & 0 \le t \le 15 \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

It comprises 16 shift enlists, every one of size 32 cycle. At each clock cycle, it produces Wt which is taken care of to the blower block. For the initial 16 clock cycles, Wt is identical to message Dt of size 512 pieces and for the accompanying 48 clock cycles Wt is registered utilizing the value appeared in Eq above. The message Dt is isolated to 16 sections, addressed as M with every one of size 32 pieces As demonstrated in Fig 6, for the initial 16 rounds, the multiplexer chooses the message M and toward the finish of the sixteenth round, every one of the registers are stacked. For the following 48 rounds, other input is selected by multiplexer which

$$sig0 = ROTR(R14,7) \oplus ROTR(R14,18) \oplus SHR(R14,3)$$
$$sig1 = ROTR(R1,17) \oplus ROTR(R1,19) \oplus SHR(R1,10)$$

is calculated as shown in previous eqn. The functions sig1 and sig0 must be worked on register

R2 and R15 respectively.The register R2 and R15 take the past worth of registers R1 and R14 separately. Subsequently, the functions sig1 and sig0 are worked on R1 and R14 one round ahead of time, which appears beneath eqn. The yields of these capacities are added and its worth sum2 is put away in transitory register Reg2. Every one of these tasks are done one cycle ahead of time, and they are included in such a way that, it requires less number of clock cycles to get the yield, in order to save the time in the present round and subsequently, speed up.

The message schedule Wt is taken from register R1 and fed to the compressor block.

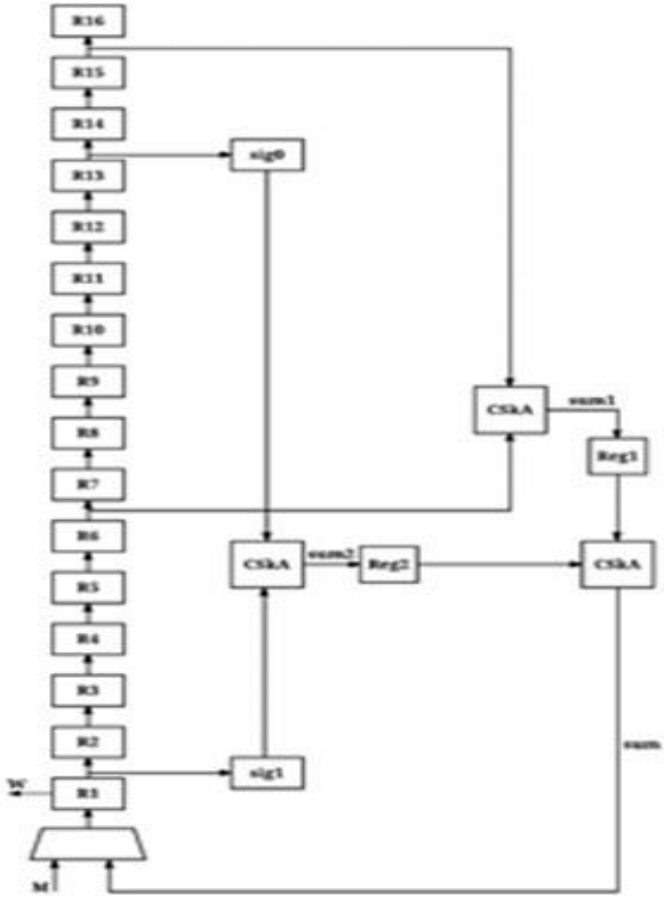


Fig.6 Optimised Expander Block

Compressor Block

The compressor block appeared in Fig.7 , is a vital part in this calculation, since it incorporates some incredible activities, which makes the calculation strong. A relatively enormous deferral is experienced because of different comprehensive estimations in the blower block, . Each round of the estimation is dependent upon its past round. In each round, new characteristics for the components A and E are controlled by performing distinctive complex undertakings and the extra characteristics are only the potential gains of past changes. The potential gains of elements A and E can be resolved as exhibited in underneath equations.

$$A_{t+1} = \begin{pmatrix} \sum_1 E_t + CH(E_t, F_t, G_t) + MAJ(A_t, B_t, C_t) + \\ \sum_0 A_t + H_t + K_t + W_t \end{pmatrix}$$

$$E_{t+1} = \sum_1 E_t + CH\left(E_t, F_t, G_t\right) + D_t + H_t + K_t + W_t$$

In above two eqn, reordering of operations and pipelining is introduced to minimize overall path delay.

$$U(t) = K_t + W_t + H_t$$
$$V(t) = K_t + W_t + D_t + H_t$$
$$\text{Re}\,g3 = B_{t-2} + F_{t-2}$$

The Fig.6 shows the design that is obtained from above conditions ,in which every one of the blocks are timed simultaneously. In the primary clock cycle, registers A,..., H are stacked with introductory hash esteems H0,H1 , ,.., . H7. In this clock cycle, multiplexers M1 and M2 select the worth put away in registers H and D separately and henceforth, Reg1=D(0), Reg2=H(0). Substance of B and F registers are additionally included this clock from past upsides of An and E individually. Expansion of registers B and F is done in this check cycle and put away in brief registers, to lessen the estimation time in the following clock cycle. Qualities K0 and W0 are likewise accessible from the expander block and table of constants at this clock cycle and new upsides of registers A H ,..., are processed. In the subsequent clock cycle, new upsides of the registers A H ,..., and Sum3, which were processed in the past round, are put away in registers A H ,..., and Reg3 individually. In this cycle, multiplexers M1 and M2 select the worth put away in registers G and C separately and consequently, Reg1=C(0), Reg2=G(0). As the worth of register Reg4 is stacked in the third clock cycle, viper yield Sum2 is accessible from the third clock cycle onwards. For the initial two clock cycles, multiplexer M3 chooses Sum1 as information, and for residual clock cycles Sum2 is chosen. Qualities K1 and W1 which are the subsequent information coming from the expander square and table of constants are utilized. New upsides of registers A H ,..., are determined and stacked into their separate registers in the following clock cycle. After the finish of every one of the 64 rounds, transitory hash esteem Htemp is produced by attaching every one of the upsides of registers A H ,..., as referenced in area II. The transitory hash esteem Htemp is added with gave beginning hash esteems to acquire hash an incentive for the first round, which goes about as starting hash esteems for the following round. Hash esteem Hi is registered for message block Di and in the wake of preparing the last message block Dn , last hash worth can be determined. For extra reason in both expander and blower blocks Carry Skip Adder (CSkA) is utilized which accomplishes least deferral as it sidesteps middle convey and subsequently, upgrades the general speed.

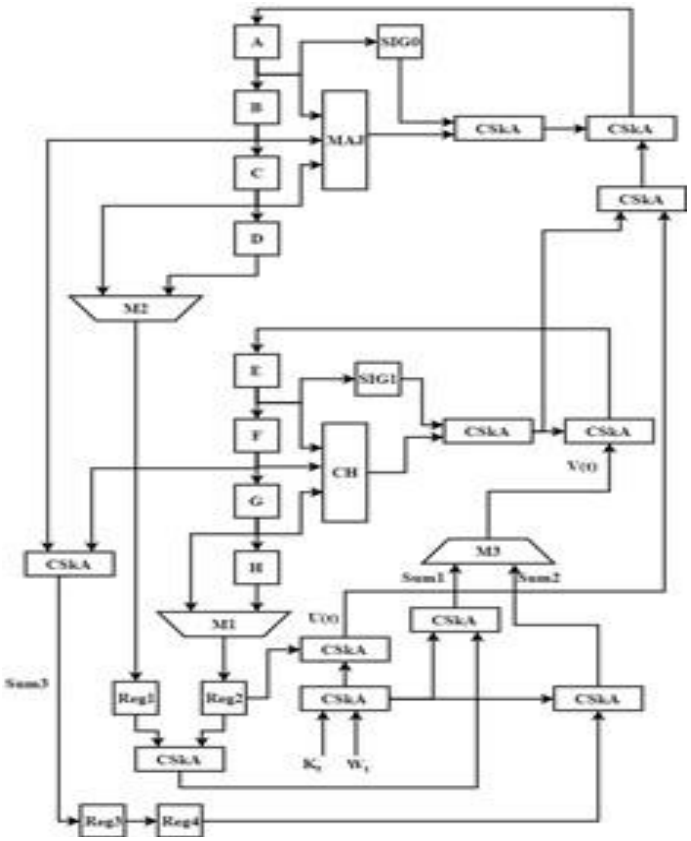


Fig.6 Optimised Compressor Block

Table with output of different algorithms.

The Xilinx Vivado device is utilized for carrying out the proposed Verilog plan. A reenactment of Verilog plan of proposed design is performed utilizing Vivado Tools for dissecting the information flow.Device use rundown of proposed engineering and execution examination of proposed strategy with existing models is appeared in Table. Throughput and Efficiency can be acquired by utilizing the formula below,.

$$Throuhput = \frac{Data\ block\ size}{Clock\ time \times Clock\ cycle}$$

$$Efficiency = \frac{Throughput}{Number\ of\ slices}$$

In this work, total of 65 clock cycles are required. To load the initial hash values and the pre-computed part in the temporary register one clock cycle is required. The remaining 64 clock cycles are used to carry out 64 rounds. Higher frequency is provided by proposed architectures of both expander and compressor blocks. As shown in table , the proposed comprehensive pipelined architecture of SHA-256 algorithm delivers a maximum frequency of nearly 170 MHz which is

higher as compared to existing techniques of [5], [6], [7], resulting into the high throughput of 1344 Mbps. The proposed architecture gives efficiency of 2.2 Mbps/slice which is very high as compared to [5], [6], [7].

Design	Proposed Method	[5]	[6]	[7]
Max Freq(MHz)	170.75	35.5	133.06	150
Clock Cycle	65	280	68	65
Throughput(Mbps)	1344.98	64.91	1009	1184
Slices	610	431	1373	797
Efficiency(Mbps/slice)	2.2	0.15	0.735	1.49

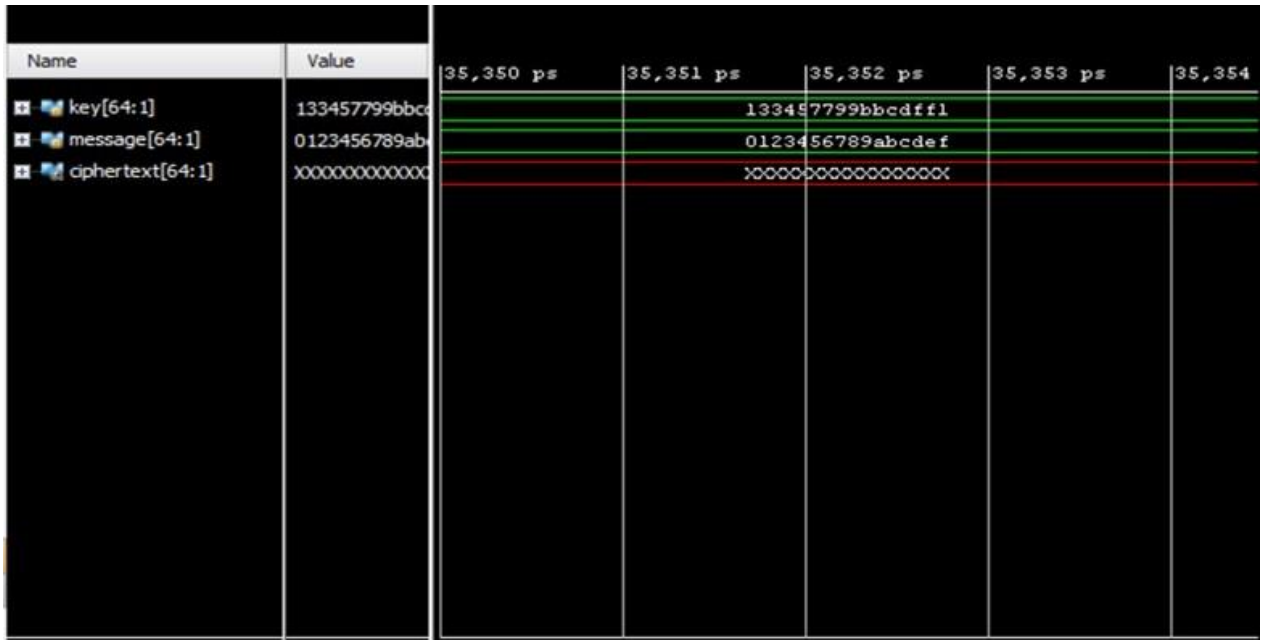
Applications of SHA 256 Algorithm

- In popular authentication and encryption protocol.
- For secure hashing passwords, Cryptocurrencies such as bitcoin use SHA-256 for verifying transactions.
- Used in blockchain.
- For Cryptographic security, Cryptographic hash algorithms produce irreversible and unique hashes, the smaller the chance that the two hash values will create the same hash.
- Software File Authentication
- Data Transmission Authentication
- Secure Code Transmission

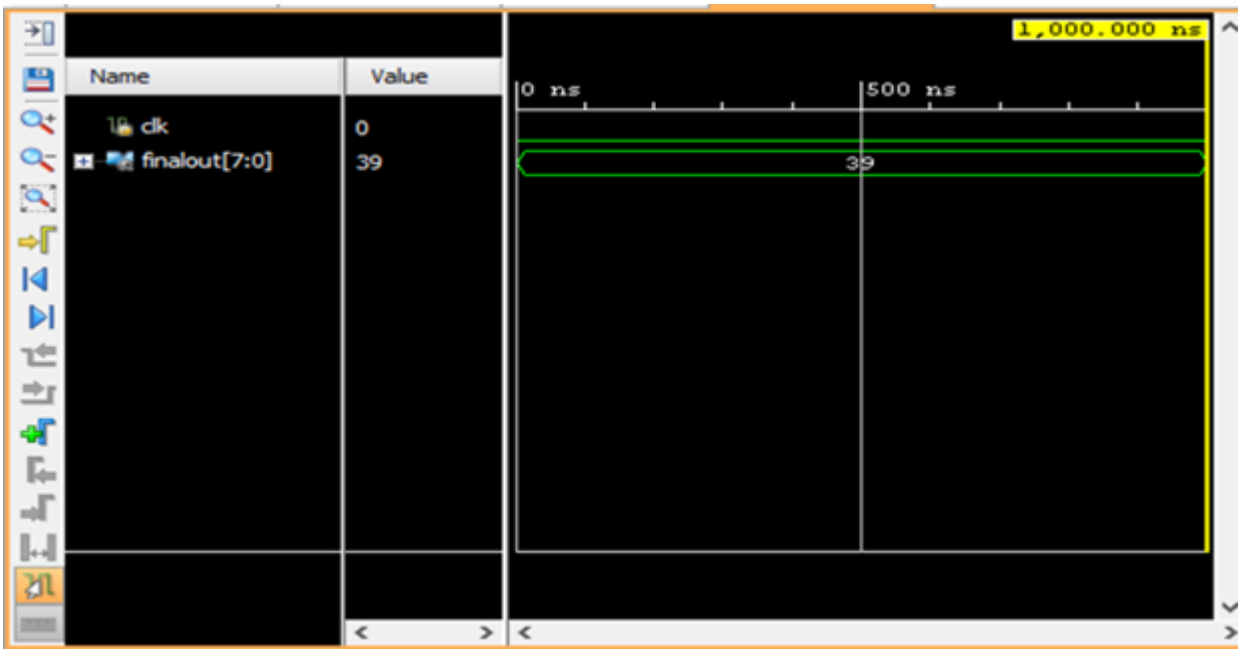
4. OUTPUT RESULTS

We have implemented the verilog code of various cryptographic algorithms in Xilinx Vivado and got the output waveforms and output results. From these output results we got an idea about the various parameters of the algorithms. From the output result of DES and AES we got an idea how to optimise the SHA- 256 algorithm. We modified the expander and compressor block of the algorithm. After implementing the verilog code we got the output simulation of tha secure hash algorithm(SHA 256) from where we can calculated the throughput, frequency, efficiency and speed of the algorithm. These results were the optimised results and more efficient hardware could be made from that.

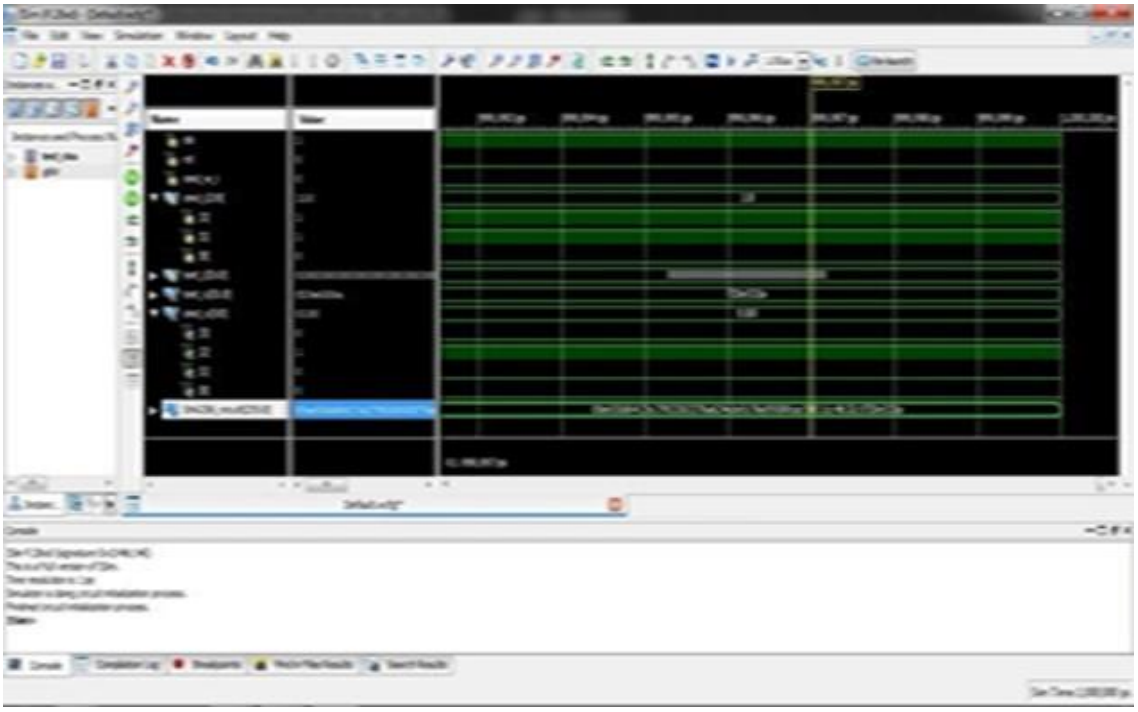
DES Simulation Output



AES Simulation Output



SHA- 256 Simulation Output



5. CONCLUSION

In this paper, at first we have carried out different cryptographic calculations in verilog like DES and AES. Through them we saw how we can execute and adjust the SHA-256 calculation. A proposed rapid FPGA execution of the SHA-256 calculation is introduced. Expander and compressor blocks of this Secure HAsh Algorithms were altered to improve the exhibition of the equipment execution. The improved equipment plans depend on a quick pipelined plan for pre-registering lively elements of SHA-256 calculation and thus, execution accelerates. This design shows an elite as far as greatest recurrence, throughput and productivity. This calculation with better proficiency can be utilized in different applications like security purposes , in bitcoin mining where more speed with less power usage is required. We have successfully optimised the algorithm. In future, the various blocks of the algorithm can be modified to further increase the throughput, efficiency and frequency of the algorithm.

6. REFERENCES

[1] Implementation of Secure Hash Algorithm-256 Kirubakaran.T1 , Srinivasan.K2 , Vengatesh.K3 , Vignesh Jagan.P4 , Vishnu Varman.A5 Assistant Professor1 , Student2,3, 4, 5 Department of ECE Sri Shakthi Institute of Engineering and Technology, Coimbatore, India, 2017

[2] Finding an Efficient FPGA Implementation of the DES Algorithm to Support the Processor Chip on Smartcard , The 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT) 2018

[3] Designing of AES Algorithm using Verilog, 2018 4th International Conference for Convergence in Technology (I2CT)

[4] Design of High-Throughput SHA-256 Hash Function based on FPGA, Shamsiah binti Suhaili Department Electrical and Electronic Engineering Faculty of Engineering, Universiti Malaysia Sarawak 94300 Kota Samarahan, Sarawak, Malaysia sushamsiah@unimas.my

[5] R. García, I. Algredo-Badillo, M. Morales-Sandoval, C. FeregrinoUribe, R. Cumplido, "A compact FPGA-based processor for the Secure Hash Algorithm SHA-256", in Computers & Electrical Engineering, 40(1), pp. 194-202, Jan 2014.

[6] R.P. McEvoy, F.M. Crowe, C.C. Murphy, W.P. Marnane, "Optimisation of the SHA-2 family of hash functions on FPGAs", in IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures, pp. 6, 2006.

[7] R. Chaves, G. Kuzmanov, L. Sousa, S. Vassiliadis, "Improving SHA-2 hardware implementations", in International Workshop on Cryptographic Hardware and Embedded Systems, Oct 2006, Springer, Berlin, Heidelberg.

[8] An FPGA based SHA-256 processor Conference Paper in Lecture Notes in Computer Science · August 2002