Multi-Core Crypto-Processor

Arya Tripathi
Department of Electronics and Communications Engineering,
Institute of Technology,
Nirma University.
Ahmedabad, Gujarat, India – 382481

20BEC130@nirmauni.ac.in

Yash Purohit
Department of Electronics and Communications Engineering,
Institute of Technology,
Nirma University.
Ahmedabad, Gujarat, India – 382481

20BEC137@nirmauni.ac.in

***Abstract* – This document presents the design of a Multi-Core Crypto processor that can perform encryption and hashing functions and can be used in conjunction with a regular processor. It can accomplish encryption and decryption of data using two symmetric key cryptographic algorithms 128-bit AES 128 and 64-bit Blowfish along with hashing through SHA-224 and SHA-256 hash algorithms. 32-bit plain input to the processor will be modified to 128- and 64-bit cipher text in AES and Blowfish cryptographic algorithms respectively. Whereas it will be scrambled to 224-bit, and 256-bit digest in the case of SHA-224 and SHA-256 Hashing Algorithms. The goal is to harness the power of encryption and hashing in regular processors without changing their design complexity and performance so as to add a layer of security in the functioning as well as communication between devices. Implementation of the same on Zync-7000 SoC has posed utilization of approximately 13.56% of total available logic elements.**

Keywords: Encryption, Hashing, 128-bit AES, 64-bit Blowfish, SHA-224, SHA-256, Digest.

I. INTRODUCTION

A. *Cryptography:*

Cryptography is the technique to make data secure in the presence of adversarial behavior. It constructs protocols that prevent third parties or any malicious user to access any private data. The fundamental principles of cryptography include confidentiality, integrity, authentication, non-repudiation and key management. A cryptographic algorithm scrambles the plain text and makes it unreadable. The revival of encrypted data depends on the type of algorithm implemented. There are 3 main types of types of Cryptographic algorithms:

1) Secret Key Cryptography:

Also known as private key or symmetric cryptography, uses a single key to encrypt and decrypt data. Data encryption and decryption is done using the same key. The keys may be identical, or there may be a simple transformation to go between the two keys.

2) Public Key Cryptography:

Public key or asymmetric key cryptography uses two separate keys to encrypt and decrypt data. It uses a pair of related keys one public key and one private key for encrypting and decrypting data. Keys are different but mathematically linked.

3) Hash Functions:

Hash functions are one-way, irreversible functions which protect the data, at the cost of not being able to recover the original message. The only way to crack a hash is by trying every input possible, until same Hash is obtained.

B. *Crypto – Processors:*

Crypto-processors are specialized computer processors designed to handle cryptographic operations such as encryption and decryption of data, digital signatures, and key management. They are commonly used in security-sensitive applications where data confidentiality and integrity are critical, such as in banking, e-commerce, and defense applications.

Crypto-processors typically have several features that distinguish them from general-purpose processors. For example, they often have dedicated hardware for performing cryptographic algorithms, which can greatly accelerate the processing of encrypted data. They also have built-in secure key storage and management capabilities, which help to prevent the theft or misuse of encryption keys.

The Crypto-processor design proposed in this paper is capable of performing encryption as well as hashing. Data encryption transforms the data from plain text to ciphertext and hashing scrambles the plain text to a unique digest. In the former case, the encrypted data can be decrypted and made readable again whereas in the latter case, the decryption requires very high computational power making it infeasible.

II. AES ALGORITHM

Advanced Encryption Standard (AES) or Rijndael is a symmetric key cryptographic algorithm developed by National Institute of Standards and Technology in the year 2001 to protect classified information which was implemented on both hardware as well software throughout the world. AES is a symmetric block cipher technique. The size of the plain and the cipher text is the same. Since it is a private key encryption technique, only a single key can encrypt and decrypt the data. AES is very much in application today as it has enhanced strength as compared to both DES and 3DES despite being harder to implement.

A. *Features of AES Algorithm:*

1) Block size : 128, 192, 256 bits
2) Key Size : 128, 192, 256 bits
3) No. of Rounds : 10 (9 with mix columns)
4) No. of S-Boxes : 1 (maps 8-bit input to 8-bit output)

B. *AES Structure and Working:*

First of all, AES converts the block of 128-bit data in the form of a 4x4 matrix of bytes. All the operations are then carried out in matrix form and the intermediate outputs are stored in the form of state matrix.

The plain data is taken as an input and passed through a pre-round transformation block where it is simply XORed with the 128-bit Key. This output now goes to the input of the first round where the data goes through the following for steps:

1) Substitution bytes: A predefined S-Box is used as a look table where the bytes are mapped according to their values. The data stored in the matrix is in Hexadecimal form, The first digit corresponds to the row and next corresponds to the column number. These rows and columns serve as indexes into the S-Box to select a unique byte.

2) Shift Row Operation: There is a cyclic left shift of the row elements. The first row is shifted by 0 bytes, the next row is shifted by 1 byte, the third row is shifted by 2 bytes and the last row is shifted by 3 bytes.

3) Mix Columns: Every Column of the state array is multiplied by a constant matrix and the new column matrix obtained replaces the original one.

4) Add Round Key: The state matrix obtained now is XORed with the 128 bits of the respective round key obtained through key expansion.

These steps remain the same till the 9th round. In the last round, the Mix Column step is discarded.

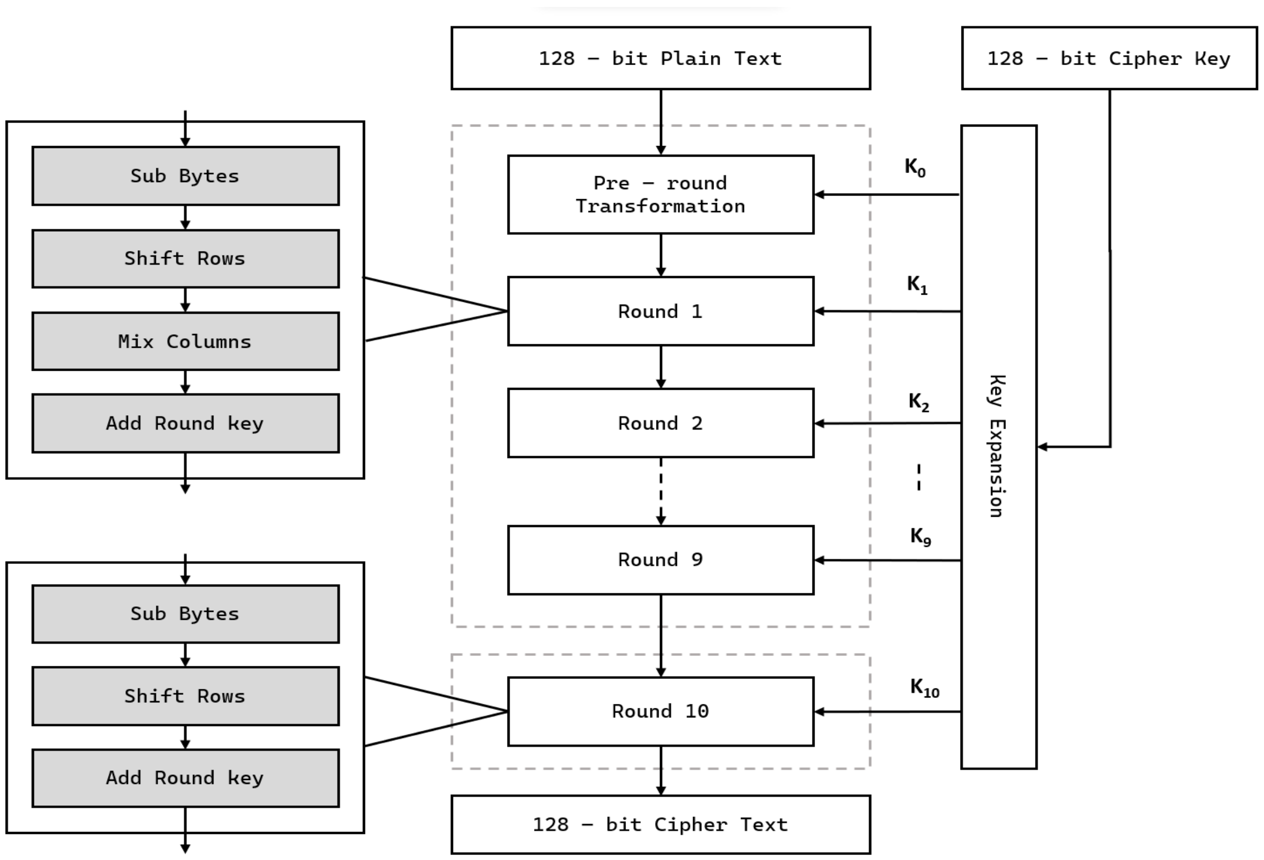

Fig. 1 AES - 128 : Flow of Algorithm

C. Key Expansion:

There are a total of 11 sub keys, one for the pre round transformation and the remaining ones for the next 10 rounds. The first key K0 is expanded through a series of steps and the key for next round K1 is obtained. This follows till the next round.

The last Column of the previous key is cyclically shifted by 1 byte (RotWord), bytes of which are then substituted by mapping with the AES S-Box (SubWord) and finally the SubWord is XORed with a round constant.

Column 1 of the new round is obtained by XORing the obtained column with the first column of the previous round. Column 2 is obtained by XORing column 1 with the second column of the previous round.

Column 3 obtained by XORing column 2 with the third column of the previous round and column 4 of the new round is obtained by XORing column 3 with the fourth column of the previous round. Thus, each round key has its new key obtained from the previous key.

Decryption of AES-128 cipher text is carries out by applying the rounds in reverse order.

C. Blowfish Algorithm

Blowfish encryption and decryption technique, developed by Bruce Schneier in 1993 is a 64-bit block size and variable key length ciphering method, which happens to be an alternative to the DES (Data Encryption Standard) technique as it provides enhanced encryption rate and improved speed as compared to DES and 3DES. Its variable key length of maximum 446 bits enables high security layers and hence it becomes difficult to crack. Till date no deciphering or cryptanalysis method for Blowfish has been found.

A. Features of Blowfish Algorithm:

1) Block size : 64 bits
2) Key Size : 32 – 446 bits (variable size).
3) No. of Subkeys : 18 (16 for each round and 2 for post processing)
4) No. of Rounds : 16
5) No. of S-Boxes : 4 (each with 256 entries of 32 bits)
6) Structure : Feistel Network

B. Blowfish Structure:

Blowfish Algorithm encrypts the data in the series iterations of 16 rounds and each requires a sub-key (here, 32-bit) which is used in both encryption and decryption. After the 16 rounds, the final post processing generates the 64-bit cipher text utilizing 2 sub-keys.

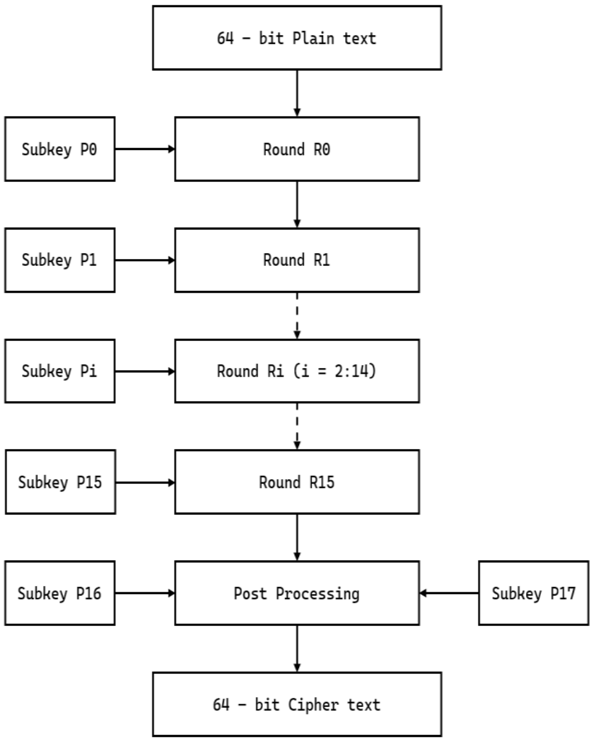

Fig. 2 Blowfish : Flow of Algorithm

C. Rounds:

Each round in the Blowfish Structure divides the input into 32-bit higher and lower double words. The higher double-word (X_L) is XORed with the subkey of the respective round and the result is fed into the Feistel function, the output of which is again XORed with the lower double-word (X_R) and is fed into the higher 32-bit double word of the round's output. The lower 32-bit double word of the output is simply equal to the higher double-word (X_L) of input.

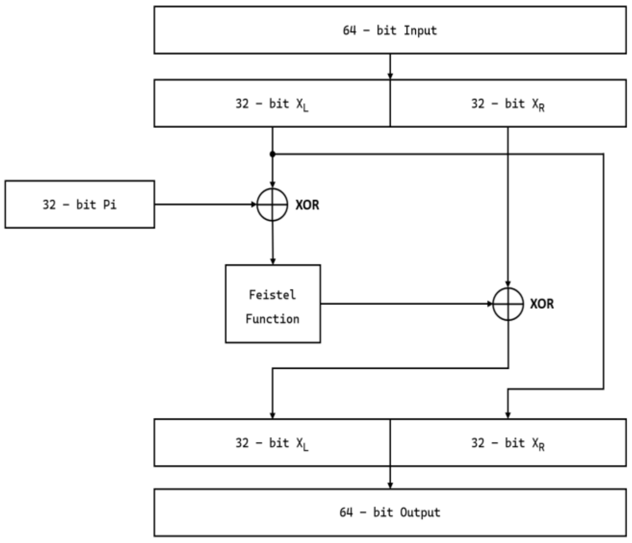

Fig. 3 Blowfish : Flow Diagram of Round R_i

D. *Feistel Function:*

The 32-bit input is divided into four 8-bit quarters by the Feistel function, which then feeds the quarters into the S-boxes. The S-boxes generate 32-bit output from 8-bit input. The final 32-bit output is created by adding the outputs and XORing them.

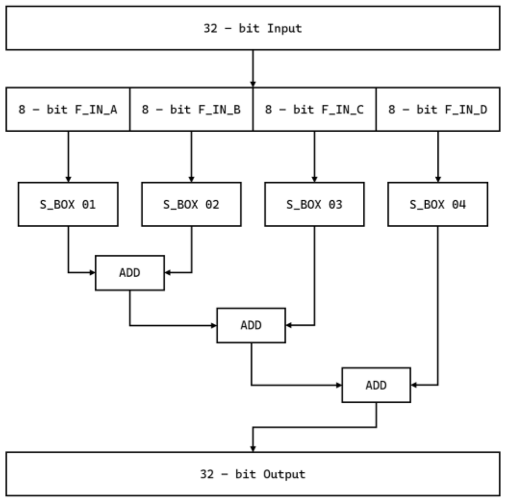

Fig. 4 Blowfish : Feistel Function

E. *Key Schedule:*

The key schedule of Blowfish starts by initializing the P-Array and the S-Boxes.

1) P – Array:

The P-Array is an 18-element array consisting of the 18 symmetric (private) subkeys, each of 32 – bits, initialized with digits of π, that are required in the processing and the post-processing rounds. The same array of 18 subkeys is utilized in the decryption process as well.

2) S – Boxes:

Both the encryption and decryption processes require 4 substitution boxes (S-boxes), each of which has 256 entries with a 32-bit length. These are initialized after the P-Array with digits of π. These boxes accept the 8-bit quarter and return a 32-bit double word which is then processed in the Feistel function.

F. *Post Processing:*

The final post processing block accepts input from the 16th round and swaps the higher and lower 32-bit double words XORing with subkey P17 and P18 respectively, to generate the 64-bit Cipher text.

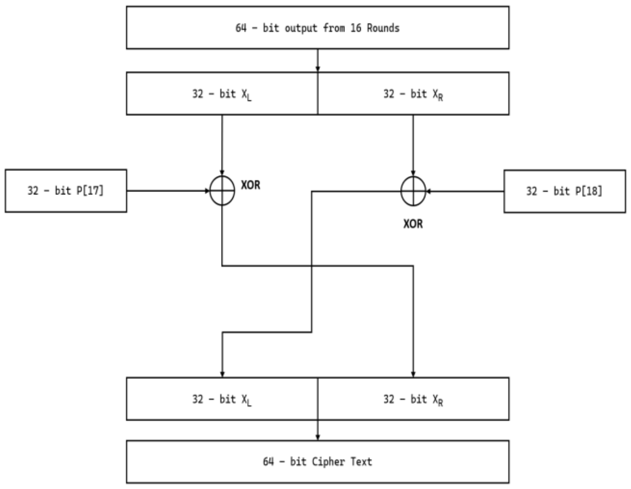

Fig. 5 Blowfish : Post – Processing Block

Blowfish decryption uses the exact same structure and operations. Only difference is that the subkeys are used in the reverse order in the deciphering process.

D. SHA HASHING ALGORITHM

SHA or Secure Hash Algorithm is a part of SHA 2 family of Hashing Algorithms that was developed in 2001 jointly by NA and NIST as a successor of SHA 1 family. Hashing can be understood as a process of scrambling raw information upto an extent that it cannot be reproduced back to its original form, with the help of certain mathematical operations. The function performing the hashing process on the variable length input is known as a hash function and the fixed length output is known as a hash/digest value.

A. Characteristics of Hash Function:

1) Deterministic: Given the same input, it will always produce the same output.

2) Uniformity: The output produced is always unique for a given input. No two separate input texts can produce the same output hash value.

3) Sensitivity to Input Changes: A small change in input results in large variations in the hash value which ensures data tampering detection as well as protection of the system integrity.

4) Computationally Efficient: A hash function produces the has values with minimal usage of time as well as systems resources.

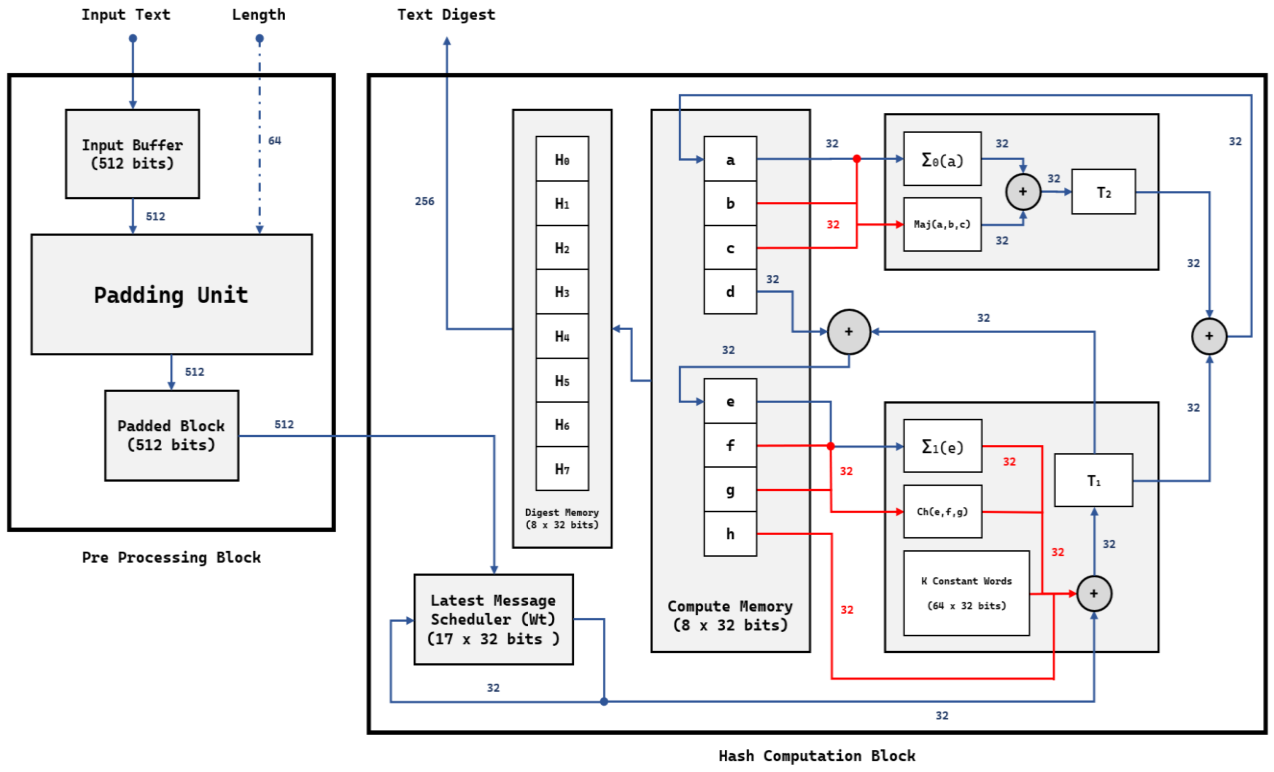

Fig. 6 SHA-256 : Algorithm Structure

B. Features of SHA-256 Hashing Algorithm :

1) Input text size : Variable
2) Output Hash size : 256 bits (32 bytes)
3) Number of Hard Coded Constants : 8 (32 - bits)
4) Number of K Constants : 64 (32 - bits)
5) Rounds : 64

C. SHA-256 Algorithm Structure :

1) Preprocessing :

The input data is preprocessed to make it compatible with the hash function which is divided into two main sub-steps :

a) Padding bits :

The message data is appended with 1 followed by all 0s at the end so that the final length becomes 64 bits less than a multiple of 512, i.e.,

Message length + Padded bits = (n * 512) – 64

b) Length bits :

The modules of the original message is taken with 2^{32} to get 64 bits of data. These 64 bits are then appended at the end of the padded message to get the processed message of length, which is a multiple of 512.

2) Buffer Initialization :

The default buffer initialized contains 8 32-bit hard coded constants which are known as hash values denoted by the expression W(i), where i denotes the index of the constant. The constants, W(i) are derived by breaking each of the 512-bit chunks into blocks of 32 bits for the first 16 rounds. For the subsequent rounds, the W(i) is calculated using certain mathematical operations as shown below :

S^0 = *(W(i - 15) R_Rotate 7)* ***XOR*** *(W(i - 15) R_Rotate 18)* ***XOR*** *(W(i - 15) R_Shift 3)*

S^1 = *(W(i - 2) R_Rotate 17)* ***XOR*** *(W(i - 2) R_Rotate 19)* ***XOR*** *(W(i - 2) R_Shift 10)*

$W(i) = W(i - 16) + s^0 + W(9 - 7) + s^1$

The above process is a sub-step in the compression step which is known as message schedule determination.

3) Compression:

The 512*n bits long input processed message is now divided into n chunks of 512 bits and each of these is processed through 52 rounds of operations. The intermediate values generated and utilized in computations are determined as follows :

$$Ch(e, f, g) = (e \text{ AND } f) \textbf{ XOR } ((\textbf{ NOT } e) \textbf{ AND } G)$$

$$Ma(a, b, c) = (a \textbf{ AND } b) \textbf{ XOR } (a \textbf{ AND } c) \textbf{ XOR } (b \textbf{ AND } c)$$

$$\Sigma a = (a >>> 2) \textbf{ XOR } (a >>> 13) \textbf{ XOR } (a >>> 22)$$

$$\Sigma e = (e >>> 6) \textbf{ XOR } (e >>> 11) \textbf{ XOR } (e >>> 25)$$

The output from the last block is the hash digest value of length 256 bits.

The SHA 224 algorithm is the truncated version of SHA 256 algorithm that is computed with different set of initial hash constant values. The output hash here is of 224 bits.

E. Crypto – Processor Architecture

The proposed design of the Crypto Processor is enabled to perform four cryptographic operations on the given input text. The processor can serve as a clock-driven, stand-alone unit to encrypt and provide the processed output as well as can be used in conjunction with another master processing unit as a coprocessor unit in securing the required data.

The input and output data ports of the processor are of 32 – bits that connect with the input and output queues. The data is latched in and out of the queues, with the help of the *Latch_in* and *Latch_out* signals. The queues have been used to support the varying lengths of input data as well as the encrypted outputs.

The function select input determines the type of cryptographic algorithm to be run on the input data, which is in accordance with the following table :

TABLE I
Function Selection

Function Selection	
00	AES - 128
01	Blowfish - 64
10	SHA – 256
11	SHA - 224

As per the function selection, the core is activated and the input data is fed into the input buffer of the respective core. After the data is available in the input buffer of the selected core, the start signal is sent to the core to begin encryption / compression of the input text. The cores available in the design here are all synchronized with a central clock. However, each requires a different amount of clock cycles to process the data and return the crypted cipher. AES requires 2 cycles, Blowfish requires 16 clock cycles in totality, i.e. 1 for each round of computation. The SHA Core utilizes 64 clock cycles to process the 64 rounds of digest.

After the completion of the process, the completion flag for the respective core is set indicating that the processed cipher can be extracted.

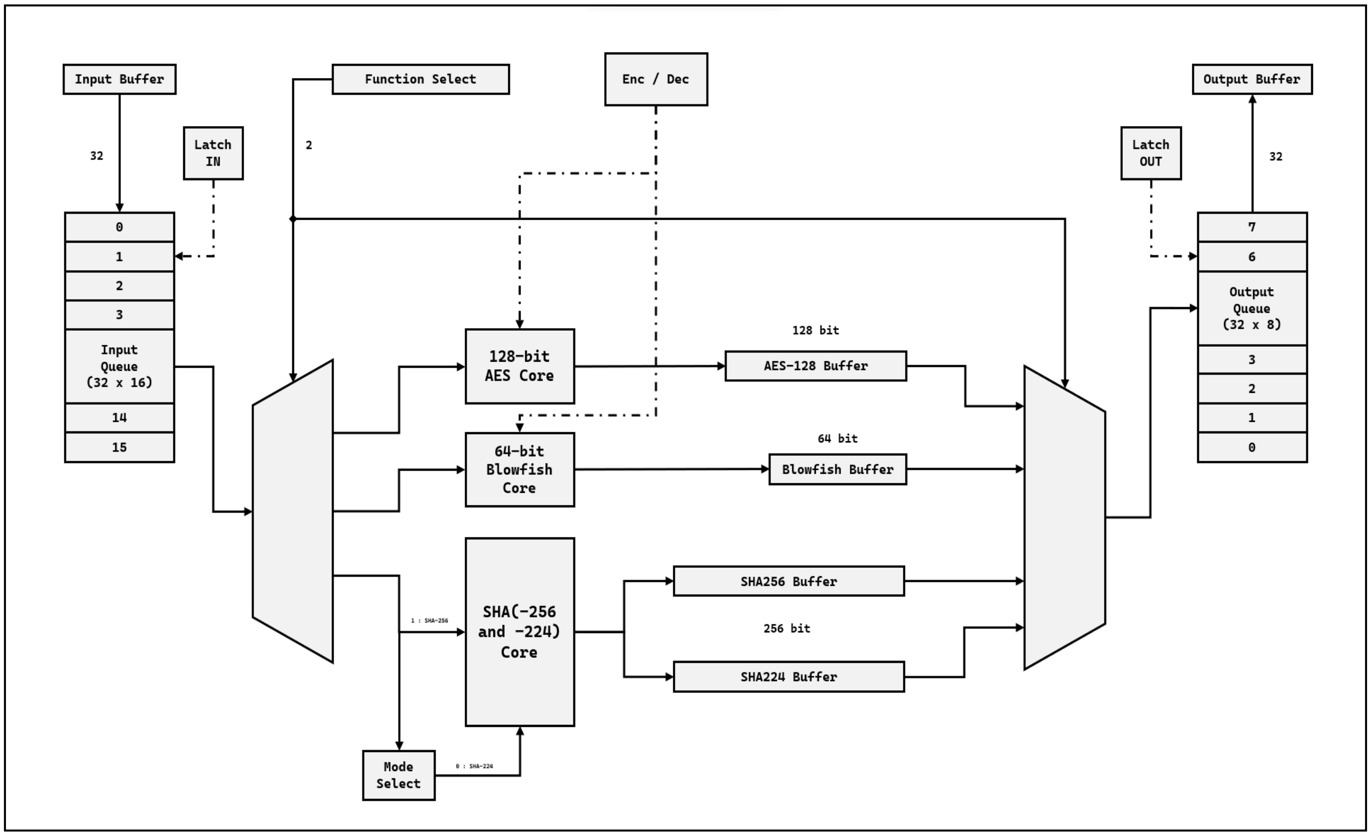

Fig. 7 Crypto-processor Architecture

After the processing, the output data is available in the output buffer of the respective core. Upon detection of the positive edges of the *Latch_out* signal, the data from the output buffer is made available in the output queue and thus is pushed out in blocks of 32 – bits.

F. Design and Implementation

The cores for the various cryptographic algorithms used here have been designed using *Verilog HDL* for implementation as well as testing on various FPGA platforms. Currently, DE2 board by Altera and ZedBoard Zynq Evaluation and Development Kit have been used for hardware implementation.

The DE2 board is a high-density FPGA from Altera designed for prototyping and testing of digital designs based on the Altera Cyclone II FPGA chip. The board includes a wide range of features, such as an on-board 50MHz oscillator, 8-channel analog-to-digital converter (ADC), 18-bit VGA output, 2-line by 16-character LCD display, and a variety of input and output interfaces.

The Cyclone II FPGA chip on the board includes 33,216 logic elements (LEs), 475 pins, and a total of 33,216 dedicated logic registers making the implementation of complex digital designs and algorithms possible.

The ZedBoard is a Xilinx Zync-7000 SoC based development board which provides a highly integrated platform. The programmable logic fabric on the Zynq-7000 SoC includes up to 85,000 logic cells, which can be used to implement complex digital designs and algorithms. It has 53200 logic cells (LUTs) fabricated over the entire area of the chip thereby making it suitable for implementing the prototype.

TABLE II
DEVICE AND TOOL SPECIFICATIONS

Device and Tool Specifications		
Device	Cyclone II	Zynq 7000
Chip Name	EP2C35F672C6	xc7z020clg484-1
HDL Used	Verilog HDL	
Design Tool	Quartus II	Xilinx Vivado
Simulation Tool	Modelsim Altera	Xilinx Vivado
FPGA Kit	Altera DE2 Board	ZedBoard Zynq Evaluation and Development Kit

A. Modules' Description:

1) AES_EC :

Accepts the 128-bit data as input along with a 128-bit key (currently initialized to 0), and returns the processed encrypted/decrypted text at the output depending upon the selection signal *E_Db.*

2) BF_EC :

Accepts a 64-bit data as input (64-bit key currently initialized to 0), and returns the processed encrypted/decrypted text at the output as per the selection signal *E_Db.*

3) SHA_256_Core :

Accepts input of 512-bits as well as the mode of SHA Algorithm to be used for Hashing (0 – SHA-224, 1 – SHA-256). The 256-bit output digest is then obtained at the output after 64 clock cycles.

4) Crypt_Proc :

This is the top module combining all the different cores. It defines the input and the output queue structure, along with the input and output buffers for the cryptographic cores, as well as the logic for core selection and input and output data handling to and from the cores.

B. Implementation Results:

Following table summarizes the implementation results for the selected device :

TABLE III
IMPLEMENTATION RESULTS ON DE2 BOARD

Implementation Results : Cyclone II	
Logic Elements Used	80,302 / 33,216
Combinational Functions	78,698 / 33,216
Dedicated Logic Registers	3,617 / 33,216
Pins Used	71 / 475
Registers Used	3617
Build Time	2 Hours 51 Minutes

The number of logic elements required for implementation of proposed crypto-processor block on Cyclone II exceeds the number of total logic elements available. This is a consequence of the design complexity at the hardware level, which implies that a high-density FPGA is required for the successful hardware synthesis of the design. Hence, ZedBoard is chosen to compensate for the insufficiency.

TABLE IV
IMPLEMENTATION RESULTS ON ZEDBOARD

IMPLEMENTATION RESULTS : ZYNQ - 7000	
LUTs	19138 / 53200
FFs	2477 / 106400
IOs	71 / 200
BUFGs	6 / 32
BUILD TIME	18 MINUTES 05 SECONDS

The diagram below illustrates the flow of data through the processor and its sub modules.

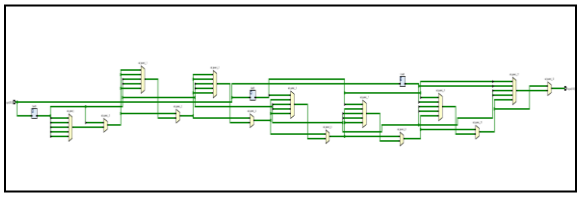

Fig. 8 Data Flow Diagram

C. *Simulation Results:*

Simulation of the designed processor module has been carried out for the different available cryptographic algorithms on simulation tool available with Xilinx Vivado and following are the results of the same :

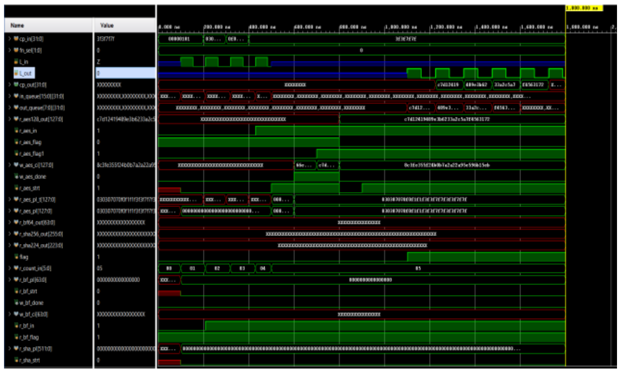

Fig. 9 AES-128 Encryption

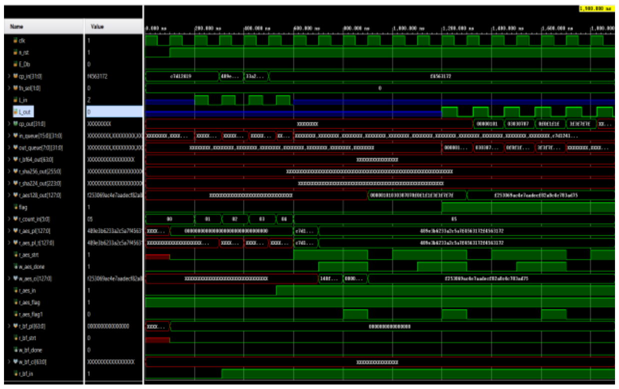

Fig. 10 AES-128 Decryption

TABLE V
AES-128 SIMULATION RESULTS

SIMULATION RESULTS	
INPUT DATA	00000101_03030707_0F0F1F1F_3F3F7F7F
INPUT KEY	00000000_00000000_00000000_00000000
AES OUTPUT	C7D12419 489E3B62 33A2C5A7 F4563172

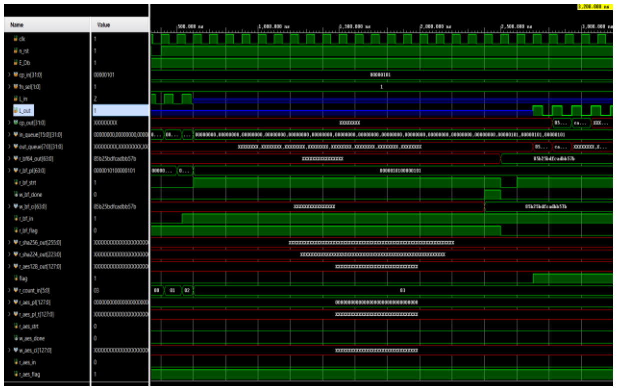

Fig. 11 Blowfish-64 Encryption

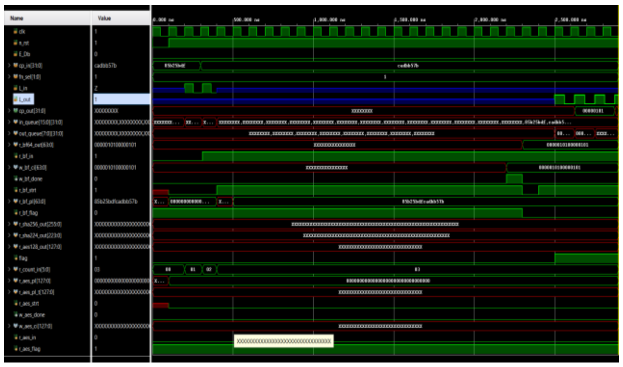

Fig. 12 Blowfish-64 Decryption

TABLE VI
BLOWFISH-64 SIMULATION RESULTS

SIMULATION RESULTS	
INPUT DATA	00000101_00000101
INPUT KEY	00000000_00000000
BLOWFISH OUTPUT	85B25BDF_CADBB57B

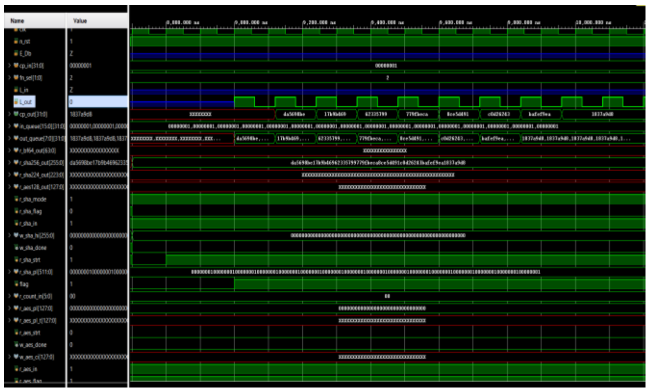

Fig. 13 SHA-256 Hashing

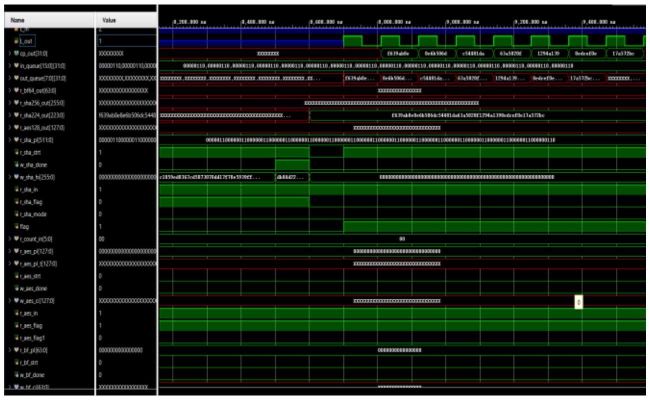

Fig. 14 SHA-224 Hashing

G. CONCLUSION

This paper elaborates the design and hardware implementation of a crypto-processor intended for the execution of multiple encryption and hashing algorithms. Incorporated with 128-bit, 10 round AES encryption and 64-bit, 16 round Blowfish, 256 bit and 224 bit SHA Hashing algorithms, the design is implemented using Verilog HDL, and has been simulated to verify the functionality as well as test it's effectiveness in the domain. The proposed unit can act as a stand alone block to convert the provided input at the respective ports to the selected cipher as well as can be used alongside a main processor unit to carry out the encryption, decryption and hashing operations for enabling an additional security layer in the simple as well as crucial operations of hardware blocks, without compromising with the performance of the main unit.

Due to the limitation of logic elements available with Cyclone II, the design ceased to be implemented on the same. Therefore, implementation has been carried out on the Zynq 7000 SoC powered ZedBoard. The design utilized almost 13.56% of the available resources which mainly have been consumed by the repeating instances of the substitution boxes required for AES encryption and decryption due to use of pure structural modelling in implementing the same.

The internal design of the AES core as well as other cores here can be improved to optimize the number of resource elements used here as well as to complete the entire process in least required clock cycles. The clock cycle requirement of SHA core typically can be reduced to bring synchronization in the operations of the crypto-processor when used as a co-unit with another processor.

To improve the versatility of the block, cores of other cryptographic algorithms can also be incorporated in the design. However, it would certainly be in trade off with the hardware elements required to implement the same and would require optimizations in the accompanying cores in order to provide faster processing and operations.

REFERENCES

[1] Advanced encryption standard (2022) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard (Accessed: November 25, 2022).

[2] Blowfish algorithm with examples (2021) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/blowfish-algorithm-with-examples/ (Accessed: November 27, 2022).

[3] Saini, G. (2021, December 15). AES algorithm and its Hardware Implementation on FPGA- A step by step guide. Medium. https://medium.com/@imgouravsaini/aes-algorithm-and-its-hardware-implementation-on-fpga-a-step-by-step-guide-2bef178db736

[4] C. Jeong and Y. Kim, "Implementation of efficient SHA-256 hash algorithm for secure vehicle communication using FPGA," 2014 International SoC Design Conference (ISOCC), Jeju, Korea (South), 2014, pp. 224-225, doi: 10.1109/ISOCC.2014.7087617.

[5] Michael Grand, Lilian Bossuet, Guy Gogniat, Bertrand Le Gal, Jean-Philippe Delahaye, et al.. A Reconfigurable Multi-core cryptoprocessor for Multi-channel Communication Systems. IPDPS - 25th IEEE International Parallel & Distributed Processing Symposium, May 2011, Anchorage, United States. pp.199-206. ffhal-00595998ff

[6] S. U. Jonwal and P. P. Shingare, "Advanced Encryption Standard (AES) implementation on FPGA with hardware in loop," *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, 2017, pp. 64-67, doi: 10.1109/ICOEI.2017.8300776.

[7] Crypto processors - Semiconductor Engineering. (2019, July 25). Semiconductor Engineering. https://semiengineering.com/knowledge_centers/semiconductor-security/crypto-processors/