School

of

Electronics and Communication Engineering

SDP Project Report

on

Implementation of 128 bit AES cryptosystem using CBC Mode

By:

1. **T A Varun Ponnappa** USN:01FE19BEC214
2. **Ayushi Negi** USN:01FE19BEC218
3. **Anup Benni** USN:01FE19BEC229
4. **Burhan Darugar** USN:01FE19BEC232

Semester: VII, 2022-2023

Under the Guidance of

Prof Shraddha H

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2022-2023

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled **" Implementation of 128 bit AES cryptosystem using CBC Mode "** is a bonafide work carried out by the team. The project report has been approved as it satisfies the requirements with respect to the Senior Design project work prescribed by the university curriculum for BE (VII Semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2022-2023.

1. **T A Varun Ponnappa**	USN:01FE19BEC214
2. **Ayushi Negi**	USN:01FE19BEC218
3. **Anup Benni**	USN:01FE19BEC229
4. **Burhan Darugar**	USN:01FE19BEC232

Semester: VII, 2022-2023

Shraddha H	**Nalini C. Iyer**	**Dr. Basavaraj S. Anami**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGMENT

We would want to convey our heartfelt gratitude to Prof. Shraddha H, our mentor, for her invaluable advice and assistance in completing our project. She was there to assist us every step of the way, and her motivation is what enabled us to accomplish the task effectively. We would also like to thank all of the other supporting personnel who assisted us by supplying the equipment that was essential and vital, without which we would not have been able to perform efficiently on this project.
We would like to thank KLE Technological University for providing us with the opportunity to work on the project [Implementation of 128 bit AES cryptosystem using CBC Mode]. Last but not least, Dr Nalini C Iyer, Head of School of Electronics and Communication , and we are deeply grateful to everyone who has contributed to the successful completion of this project.

-The project team

ABSTRACT

Due to the rise in the usage of internet applications, the concern over data security has been questioned. To secure the information over the communication channel cryptography is used. There are various cryptographic algorithms used for the encryption and decryption of the data one such algorithm is an Advanced Encryption Standard. In our project, we have designed an Advanced Encryption Standard of key size 128bits using Cipher block chaining (CBC) mode. We have used Spartan-VI family device XC6SLX4-CSG225(device package) in Xilinx 14.7 version. This mode makes use of an initialization vector IV which is an important part of this algorithm. Advanced encryption standard is a block cipher-based symmetric-key cryptography that operates on a 128-bit block of input data. The key size we have used for the encryption and decryption is 128 bits. For a key size of 128 bits, 10 rounds are performed on an input block of data. The advanced encryption standard algorithm uses a round function to perform four different byte-oriented transformations that are Sub Byte: for byte substitution using S-box, Shift Rows: Shifting rows of the state array, Mix Columns: Mixing the data within each column, Add Round Key: for adding the round key to the state. The Cipher Block Chaining (CBC) uses an initialization vector- IV and makes use of a block cipher algorithm. As the plain text that is being sent in the real world is of varying lengths it is divided into blocks and needs to add padding data of 128 bits before encryption or decryption. The plaintext block is then Xor 'ed with the initialization vector IV. Using a block cipher algorithm, it is converted to cipher block text. Encryption result is used to xor with the plaintext block until the last block. When it comes to decrypting the data, it can be decrypted in parallel which is not possible when encrypting the data block. On analyzing the results, it takes 256 clock cycles to shift the plain text and cipher text serially out. The effect of implementing PISO on IO ports reduces the requirement from $642 - 305\%$ to $5 - 2.3\%$

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Significant growth in wireless technologies over the last few years has created a need for high-quality data security, as large amounts of a variety of data are being transferred from the sender to the receiver over an unsafe channel. In this technical era, data authentication and secured communication are becoming very important. Many government sectors, e-commerce websites, and companies are using cryptography to secure their electronic data from third parties. Cryptography is the practice of secure communication intended only for those persons for whom the information is intended to prevent unauthorized access to the data. Cryptography involves two key processes that are encryption and decryption. Encryption is a method in which plaintext is encoded to an unreadable format that is cipher text to prevent a third party from accessing it. Decryption is the opposite of the encryption process in this the encrypted data that is the cipher text is converted back to the original text which is the message sent by the sender to the receiver. Cryptography can be divided into three types that are public key cryptography, secret key cryptography, and hash functions. Public key cryptography uses pair of keys to encrypt and decrypt the information that is private and public keys. Hash functions are used to protect the information in an irreversible way by which an original message cannot be recovered. Secret key cryptography uses a single key for both the encryption and decryption processes. Advanced Encryption Standard is symmetric key cryptography that takes 128-bit input data. Key sizes for the AES encryption and decryption process are 128, 192, and 256 bits. 10 rounds of repetition are performed for 128-bit key size. The advanced encryption standard algorithm uses a round function to perform four different byte-oriented transformations that are Sub Byte: for byte substitution using S-box, Shift Rows: Shifting rows of the state array, Mix Columns: Mixing the data within each column, Add Round Key: for adding the round key to the state. The key expansion algorithm provides eleven 128-bit round keys from the 128-bit cipher key. The first four words of the 128-bit round keys are fixed while for the remaining 40 words three operations are used firstly Rotate: elements in the column are rotated vertically clockwise by 1 byte, secondly being Rotate constant: words in the position of multiple of Nk is xor 'ed with round constant, third S-box: takes 4-byte input data and implies the S-box. NIST approves the use of five modes of operation when using symmetric cryptography and those are Electronic Codebook mode (ECB), Cipher Block Chaining mode (CBC), Cipher Feedback mode (CFB), Output Feedback mode (OFB), Counter mode (CTR). Cipher block chaining mode requires an initialization vector (IV) which is combined with the first block. In the CBC encryption process, the first step is XOR-ing the plaintext with the initialization vector (IV) which then becomes the input for the cipher function. The output of the cipher function is the cipher text block. The cipher text block is then XOR 'ed with the next plain text block and fed to the cipher function resulting in the next cipher text block as depicted in the Fig 1.1. The steps are repeated again to get the further cipher text block. As it goes in the chain, any change in the plain text block or the initialization vector (IV) could affect all the further cipher text blocks.

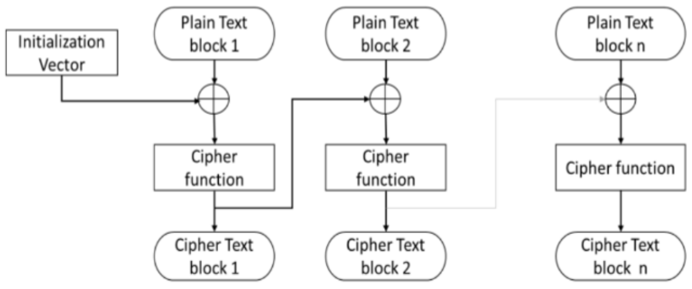


Figure 1.1: AEC CBC Encryption Cycle

In the CBC decryption process, the inverse cipher function is applied to the input cipher text block and the output is then XOR' ed with the initialization vector (IV) to get the first plain text block as depicted in the Fig 1.2. Then again inverse cipher function is applied to the next cipher text block and the output is then XOR' ed with the previous cipher text block resulting in the second plain text block. Further, the steps are repeated again to get the next set of plain text blocks.

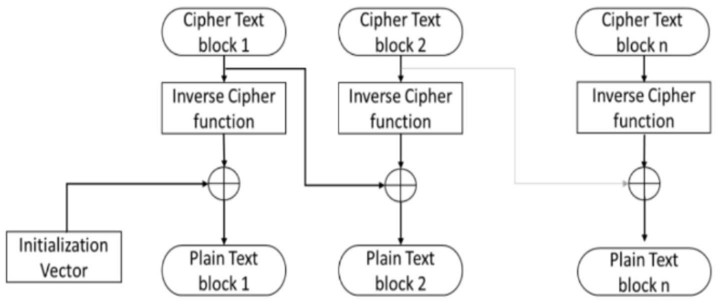


Figure 1.2: AEC CBC Decryption Cycle

1.1 Motivation

In recent years there have been great advancements in the field of wireless technologies which has created a need for secure data transmission over the communication channel from the sender to the receiver end. The main motivation is to protect electronic data from unauthorized access. As users are expecting high-quality security for their electronic data as it consists of all sorts of information regarding their online transactions, banking details, one-time passwords, important notifications, etc. In order to safeguard their messages, there is a need for a strong cryptographic algorithm like an Advanced encryption standard (AES). It is a secret key cryptography as it makes use of a single key which is added in every round to increase the security of the data block. But there can be risks associated with the hacking of the secret key, in order to make it more secure we have applied the cipher block chaining (CBC) algorithm. CBC uses an initialization vector (IV) which is then xor' ed with plain text block in the CBC encryption process in every step. It is a mode of operation in which a sequence of bits is encrypted by a single block with a cipher key applied to it.

1.2 Objectives

1. **Understanding of AES Algorithm**
2. **Implementation of AES 128-bit algorithm –[Encryption, Decryption]**
3. **Implementing AES using CBC mode (Cipher Block Module)**
4. **Implementation of AES Cryptosystem**

1.3 Literature survey

1. AES-CBC Software Execution Optimization

With the advancement in wireless technology, there is an ever-growing need for an efficient and secure method of encryption of electronic data. Cryptography consists of rigorous computational tasks. This paper examines the efficiency and execution of Advanced Encryption Standard (AES)- Rijndael encryption/decryption in Cipher Block Chaining (CBC) mode. AES-CBC is implemented to check its resource utilization and speed. It also explores the working of image encryption on common low-power devices. The detailed result of the performance of the AES cryptosystem on wireless devices is implemented in this paper. After the optimization, the encryption speed performance is increased by 12-30% but at the cost of using double the memory for code size. [6]

2. Security Enhancement of AES-CBC and its Performance Evaluation Using the Avalanche Effect

With the rise in computational calculation, electronic data security over a communication channel has become a significant concern. To protect important information from third-party access high- security cryptographic algorithms are used. This paper examines the security enhancement of advanced encryption standard (AES) Cipher block chaining (CBC) mode and its performance evaluation using the avalanche effect. For CBC mode before the encryption rounds, an Initialization Vector (IV) is needed for which the source has been derived from Unix time. On using the algorithm different cipher text is generated at each execution which decreases the risk of hacking the encryption key. The result generated is examined using the Avalanche effect and tested to check the security of the information. The end results showed that the encryption method succeeds in maintaining the avalanche effect requirement and introducing additional strength to the encryption process by preventing the encryption key update for every new cipher text. [3]

3. Design and analysis of AES-CBC[Verilog] mode for high security applications

In this paper, the design and analysis of AES-CBC mode [Verilog] are presented to find the error during the encryption process. Cryptographic algorithms require intensive computational calculation, so the major challenge involved in securing the communication channel is to find an easier method to solve those computations. Advanced encryption standard (AES) block cipher combines a 128-bit plaintext data block with a 128-bit key to get the cipher text data block of 128 bits. In AES-ECB mode the message to be sent is split into blocks and each block is encrypted separately. If there is an identical plaintext block then the encrypted cipher text block will also be identical this could create vulnerabilities like modification of encrypted data. To solve this problem AES-CBC mode is used. Cipher block chaining makes use of an initialization vector (IV) to make each ciphered text unique. Before encrypting a block, it is XORed with the cipher text of the previous cipher text block. Simulation is done to analyse the chip size reduction. [16]

4. Generation and Evaluation of a New Time-Dependent Dynamic S-Box Algorithm for AES Block Cipher Cryptosystems

Advanced Encryption Standard (AES) block cipher algorithm is an important part of the Communication channel. Nonlinearity produced in the s-box during the encryption process is used to measure the strength of the encryption process. There are two types of s-boxes used in the encryption process one is static that is the content of the s-box does not change and the other is dynamic where the content of the s-box change continuously. This increases the possibility of unauthorized decryption that is identical plaintext data producing identical cipher text data and every time a new s-box is created it requires sharing of the encryption key. This paper uses a dynamic s-box. Inside the transmitter and receiver side, a nonlinear s-box is generated identically. The principal approach is to keep the encryption key constant while the ciphertext is dynamic that is it changes continuously resulting in different encryption results for the identical plaintext. The Avalanche effect method is also used to analyze the strength of the new s-box resulting in a unique cipher text in every encryption process. The results showed that modification in the dynamic s-box produced a better avalanche effect. [2]

5. The Evaluation of Time-Dependent Initialization Vector Advanced Encryption Standard Algorithm for Image Encryption

Sensitive data has to be protected from the public. A strong encryption algorithm is required for protecting images. Computing speed is growing, making algorithms vulnerable to attacks. An enhanced AES-CBC algorithm is evaluated for the encryption proposed. The algorithm satisfied the required standards and passed all tests. The Advanced Encryption Standard (AES) has become an attractive encryption method for high security and fast implementation. The encryption algorithm is approved as a standard for widely-used communication and data processing units. However, the advance in technology and the introduction of quantum computers made the encryption scheme vulnerable to attack. Different attack procedures are continuously developed to attack end decrypted important and sensitive data. This paper evaluated an enhanced Advanced Encryption System operating in Cipher Block Chaining mode, suggesting a promising solution for resisting future attacks. The approach depends on a time-dependent initialization vector that produces the initialization vector block depending on the epoch time without sharing any encryption key. The evaluation process includes correlation analysis, global and local Shannon entropy analysis, chi-square analysis, histogram analysis, and differential analysis. The results showed that the enhanced encryption scheme is reliable and can resist most cyber-attacks without exposing any encrypted data to the public. The results were compared with previously published and tested algorithms and found to satisfy and exceed the minimum requirement. So, the encryption method can be implemented safely in future communication channels or used in the file encryption process[4]

6. Performance Evaluation of Advanced Encryption Standard Algorithm

Nowadays the confidentiality of information is an issue of primary importance. Generally, confidentiality is obtained by encrypting/decrypting the information with a symmetric algorithm.

Currently, the most used and standardized algorithm is the Advanced Encryption Standard (AES), but the encryption and decryption usually cause undesired delays in the access to information. As users must necessarily use either AES or another encryption/decryption algorithm to guarantee confidentiality, the implications on performance must always be evaluated. It is very interesting to evaluate the influence of the configuration parameters of AES on performance, in order to select an appropriate configuration. This work provides a performance evaluation methodology to estimate how the configuration of any encryption/decryption algorithm affects the performance. The methodology has been applied to the AES algorithm in five different execution platforms, obtaining useful results for any user of the AES algorithm. [8]

7. A High Throughput/Gate AES Hardware Architecture by Compressing Encryption and Decryption Datapaths - Toward Efficient CBC-Mode Implementation

This paper proposes a highly efficient AES hardware architecture that supports both encryption and decryption for the CBC mode. Some conventional AES architectures employ pipelining techniques to enhance the throughput and efficiency. However, such pipelined architectures are frequently unfit because many practical cryptographic applications work in the CBC mode, where block-wise parallelism is not available for encryption. In this paper, we present an efficient AES encryption/decryption hardware design suitable for such block-chaining modes. In particular, new operation-reordering and register-retiming techniques allow us to unify the inversion circuits for encryption and decryption without any delay overhead. A new unification technique for linear mappings further reduces both the area and critical delay in total. Our design employs a common loop architecture and can therefore efficiently perform even in the CBC mode. We also present a shared key scheduling data path that can work on-the-fly in the proposed architecture. To the best of our knowledge, the proposed architecture has the shortest critical path delay and is the most efficient in terms of throughput per area among conventional AES encryption/decryption architectures with tower-field S-boxes. We evaluate the performance of the proposed and some conventional data path by logic synthesis results with the TSMC 65-nm standard-cell library and Nan Gate 45- and 15-nm open-cell libraries. As a result, we confirm that our proposed architecture achieves approximately 53–72% higher efficiency than any other conventional counterpart. [13]

1.4 Problem statement

To design and implement Advanced Encryption Standard of key size 128bits using CBC mode

1.5 Application in Societal Context

Protecting our Digital data is important as it contains all sorts of major data regarding our online banking transactions, one-time passwords, password reset messages, important notifications, etc. Cryptography plays a major role in safeguarding our confidentiality and authentication between communicators. It has applications in cyber security, protecting electronic data, and providing security to government computers. It is also used in encrypting large chunks of data for communication devices, mobiles, laptops, smart phones, etc. Digital wallet platforms and e-commerce websites make use of cryptographic algorithms such as Pay tm makes use of the AES128 encryption algorithm. Healthcare departments also make use of such algorithms to keep electronic health information confidential from third parties. Moreover, AES accounts for the major part of mobile applications, Wi-Fi, VPN, etc.

1.6 Organization of the report

Chapter 1: Introduction
It includes motivation towards the project, objectives of project, literature survey done towards the problem statement, defining the problem statement and looking at the application in a societal context, followed by project planning.project planning.

Chapter 2: System design
This chapter includes the High-level functional block diagram and description of the functional blocks.

Chapter 3: Implementation details
This chapter contains detailed information about implementation. It also Includes specifications used to design and propose the final system architecture. The algorithm and flowchart give the birds-eye view of the functionalities achieved

Chapter 4: Results and Discussions
This chapter includes the conclusion and results of implementation.

Chapter 5: Conclusion
This is the concluding chapter that includes project closure and epilogue along with the future scope of our project.

Chapter 2

System design

This chapter discusses the functional block diagram AES Encryption and Decryption using CBC Mode, design alternatives and final design.

2.1 Functional block diagram

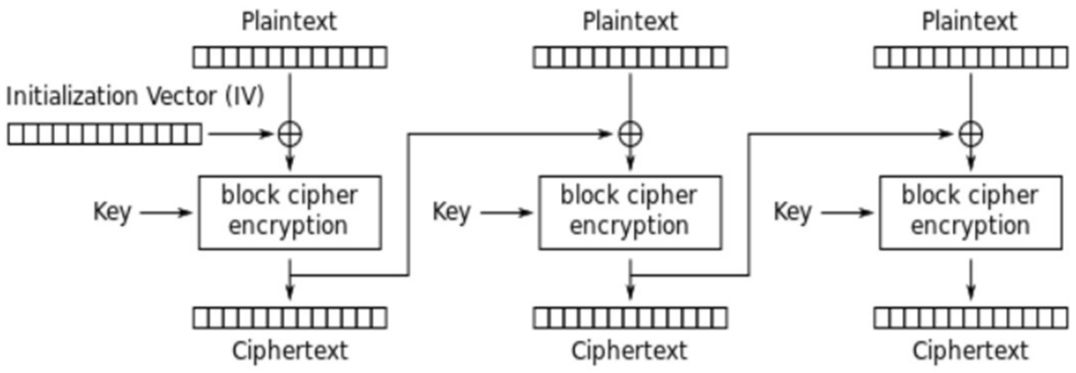


Figure 2.1: Functional Block Diagram

As per the figure above the system should be capable of producing cipher text from plaintext using CBC mode and encryption and decryption is done by AES Algorithm.

2.2 Design alternatives

- **Modes**
  - **ECB Mode:**The easiest way is ECB (Electronic Code Book). It is typically not advised due to its evident flaws. Blocks of the plaintext are separated according to the 128-bit AES block size. As a result, the ECB mode must pad the data until it is the same length as the block. Then, each block will be encrypted using the same method and key. We will thus obtain the same ciphertext if we encrypt the same plaintext. As a result, this mode carries a considerable risk. Additionally, there is a 1:1 correlation between the plaintext and ciphertext blocks. We can encrypt

and decode the data simultaneously since encryption and decryption are separate processes. Additionally, a broken block of plaintext or ciphertext won't have an impact on other blocks. The Mallory can launch an attack even if they are unable to obtain the plaintext because to an ECB feature. For instance, the Mallory may replicate the information in C1 to C2 if we encrypt the information about our bank account as follows: The ciphertext: C1: 21 33 4e 5a 35 44 90 4b (the account) C2: 67 78 45 22 aa cb d1 e5. Then, using the account as the password, which is simpler to remember, he may get into the system.[15]

– **CBC Mode:**This is made possible by the initialization vector IV used in the CBC (Cipher Block Chaining) mode of Figure 2 (Fig. The IV is the same size as the encrypted block. The IV is often a random number, not a nonce, in general. There has to be padding information added to the blocks of the plaintext. The plaintext block xor with the IV will be used first. The output is then encrypted using CBC to create a block of ciphertext. The encryption result will be combined with the plaintext block until the final block in the next block. Even if we use this mode to encrypt the same plaintext block, the ciphertext block we receive will be different. Decrypting data in parallel is conceivable, but encrypting data is not. A Mallory can modify the IV to attack the system. If a plaintext or ciphertext block is compromised, it will impact all blocks that follow. All data is corrupt even if only one bit in the IV is incorrect. Mallory is also capable to launching a padding oracle attack. To increase the size of the ciphertext block, they might employ some ciphertext. Several messages pertaining to the plaintext will be returned. The key must be changed while encrypting 2((n+1)/2) to maintain security. It is resistant to CPA, but easily susceptible to CCA and PA (n is the length of a block). [16]

– **CFB Mode:** The block encryptor may be used as a stream cypher by switching to the CFB (Cipher FeedBack) mode of operation. An IV is also required. CFB will first encrypt the IV before xoring it with a plaintext block to produce ciphertext. The plaintext will then be xor encrypted using the encryption result. This mode just utilises the ciphertext to xor with the plaintext to produce the ciphertext because it won't directly encrypt plaintext. This mode eliminates the requirement for data padding and allows for concurrent data decryption rather than encryption. Since this mode is comparable to the CBC, any broken blocks will have an impact on all subsequent blocks. This mode is vulnerable to replay attacks.[5]

MODE	ADVATNAGES	DISADVATNAGES
ECB	• Simple • Fast	• Duplicate data in plaintext will be reflected in the ciphertext • Can't resist replay attacks • The plaintext can be operated by deleting or replacing the cipher text.
CBC	• Ability to decrypt any cipher text packet. • Support for parallel computing.	• Wrong blocks affect all following blocks. • Do not support for parallel computing(encryption).
CFB	• No padding • Can be prepared for encryption and decryption first.	• Do not support for parallel computing(encryption). • Can't resist replay attacks. • Wrong blocks affect all following blocks.

2.3 Final design

We select one of the optimal solutions based on its working and ease of the implementation.

Modes

CBC Mode: We selected CBC Mode of AES encryption and Decryption because it supports for parallel Computing, It has ability to decrypt any cipher text packet, And duplicate data in plaintext will not be reflected in cipher text packet.

Chapter 3

Implementation details

This Chapter concludes about system Architechture for AES CBC Block. And algorithm in designing the CBC Unit for a File.

3.1 Specifications and final system architecture

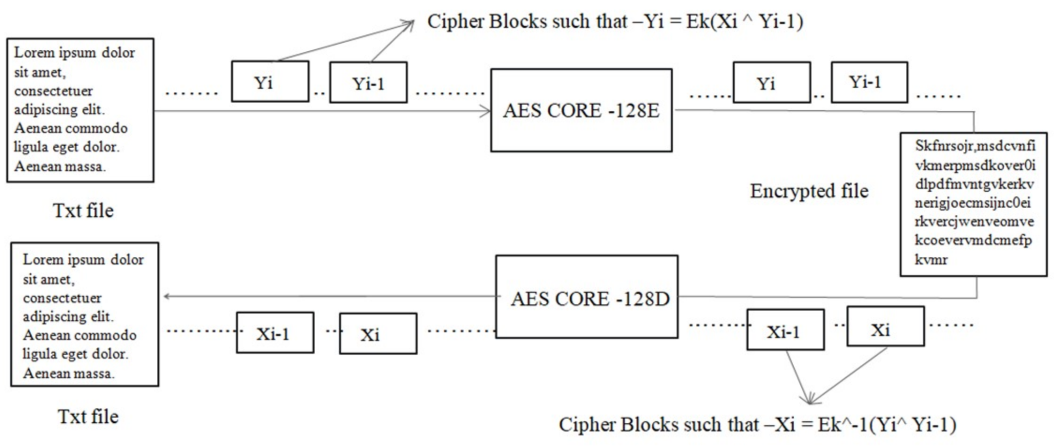

Cipher Blocks such that –Yi = Ek(Xi ^ Yi-1)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.
Txt file
Yi
Yi-1
AES CORE -128E
Yi
Yi-1
Skfnrsojr,msdcvnfi vkmerpmsdkover0i dlpdfmvntgvkerkv nerigjoecmsijnc0ei rkvercjwenveomve kcoevervmdcmefp kvmr
Encrypted file
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.
Txt file
AES CORE -128D
Xi-1
Xi
Xi-1
Xi
Cipher Blocks such that –Xi = Ek^-1(Yi^ Yi-1)

Figure 3.1: System architechture

The above is the Final system architecture of AES CBC Crypto system. The idea in the above system is to divide the content of the files into set of blocks link these blocks using CBC mode and pass one block at a time to the AES_128bit_Encryption core. And then generate the encrypted file from the AES Core. The same is being done at the decryption side .Encrypted file acts as the Input to the AES_128bit_Decryption core we divide the file content into 128 bit block and give it to the core to get back the text file. [8]

Cipher Block Chaining Encryption- Equation 3.1 depicts the relation of one cipher block to the other during the Encryption side. Consider a file with n no of blocks using a single 128 bit key we connect the blocks together. We take the current block and XOR it with the previous cipher text. For the first block IV acts as previous cipher text. And the same is being given to the encryption core.

$$C_i = E_k(M_i \oplus C_{i-1}) with C_0 = IV \tag{3.1}$$

Cipher Block Chaining Decryption- Equation 3.2 depicts the relation of one cipher block to the other with respect to the decryption side. Consider a file with n no of blocks. We take the Current cipher text block and give it to the Decryption core. The result from the decryption core is then XORed with the previous cipher block. For the first block we take the previous one as IV vector.

$$M_i = C_{i-1} \oplus D_k(C_i) with C_0 = IV \tag{3.2}$$

Cipher Block Chaining Representation. The Fig 3.2 Depicts the Input to the Encryption core as the XOR of IV with plain text for the first particular block. XOR of current plaintext with the cipher text of previous block for further blocks. The Reverse of this procedure is done at the Decryption side to obtain the plaintext from the Encrypted file. [9]

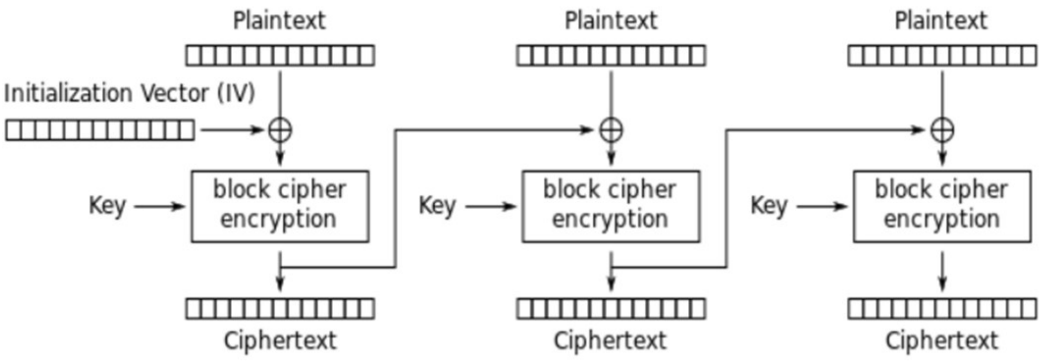


Figure 3.2: CBC Block

3.2 Step By Step Procedure

Key Expansion in AES

1. AES-CBC core. It takes a 128-bit (16-byte) key and a 4-word key as input and creates a 44-word linear array.

2. A 4-word round key is available in the initial phase. In the initial stage, we add round key and 10 rounds of encryption each. So a total of 11 rounds = 44 words.

3. First, the key is inserted into the first four words of the extended key.

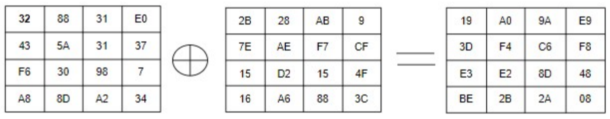
4. The function g defines a complex number and this function consists of the following sub functions.

- Perform a left circular shift by 1 byte
- When using S-Box, each partial word performs byte replacement.
- The final result of the word is XORed with the round constant

Initial Round

The Add round key adds every column of the state matrix with the round key word. The Add round key precedes one column at a time. In this transformation the cipher text key will be included .The final state matrix will be generated by doing XOR operation among column of the state and the key matrix which is generated by key generator.

- Add Round key operation is performed in Initial stage.
- The 16 bytes of the matrix are now considered as 12 bits and are XORed to the 128 bits of the round key.



$$\begin{array}{|c|c|c|c|}\hline 32 & 88 & 31 & E0 \\ \hline 43 & 5A & 31 & 37 \\ \hline F6 & 30 & 98 & 7 \\ \hline A8 & 8D & A2 & 34 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline 2B & 28 & AB & 9 \\ \hline 7E & AE & F7 & CF \\ \hline 15 & D2 & 15 & 4F \\ \hline 16 & A6 & 88 & 3C \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 19 & A0 & 9A & E9 \\ \hline 3D & F4 & C6 & F8 \\ \hline E3 & E2 & 8D & 48 \\ \hline BE & 2B & 2A & 08 \\ \hline \end{array}$$

Figure 3.3: Add Round Key

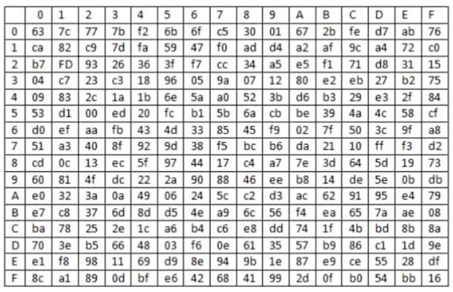
Main Round Operations

Based on the key size we decide the no of rounds. As the key size here is 128 bit key hence the no of rounds are 10 rounds. After completing the Initial round we computer four main steps .The last round doesn't have Mix Columns round. The Sub bytes does the substitution and shift rows and Mix columns performs the permutation in the algorithm .

STEP 1 – Sub Bytes Operation

- Substitutions are performed in this step.
- This step replaces each byte with another byte. This is done using a lookup table, also called an S-Box. This replacement is done in such a way that the byte is neither replaced by itself nor by another byte that is the complement of the current byte.
- Galois Field (GF(28)) and irreducible polynomial (as given in eq 3.3) are used in the Byte Substitution Transformation of AES to produce the S-Box. [11]

$$x^8 + x^4 + x^3 + x^1 + 1 \tag{3.3}$$



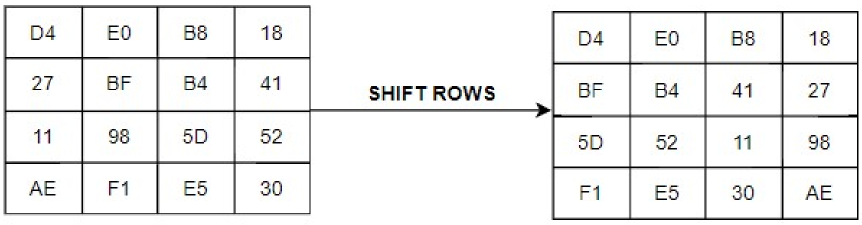
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	FD	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
A	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
B	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
C	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
D	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
E	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
F	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 3.4: S Box

STEP 2 – Shift Rows

Shift Rows operation that cyclically shifts last three rows of 16 byte block as mentioned below. a circular shift is being performed hift Rows operation that cyclically shifts last three rows of 16 byte block as mentioned below. a circular shift is being performed

- No Shift in First Row
- 2^{nd} row is shifted left once
- 3^{rd} row is shifted left twice
- 4^{th} row is shifted left thrice



D4	E0	B8	18
27	BF	B4	41
11	98	5D	52
AE	F1	E5	30

D4	E0	B8	18
BF	B4	41	27
5D	52	11	98
F1	E5	30	AE

Figure 3.5: Shift Row Operation

STEP 3 – Mix Columns

This step is basically a matrix multiplication. Each column is multiplied by a specific matrix, resulting in a change in the position of each byte within the column.

Mix Column multiplies each byte in the row of the matrix transformation by each value (byte) in the status column. That is, each row of the matrix transformation must be multiplied by each column of status. The results of these multiplications are XORed to produce 4 new bytes with the following states: This procedure does not resize the stand, it remains at his original 4x4 size.

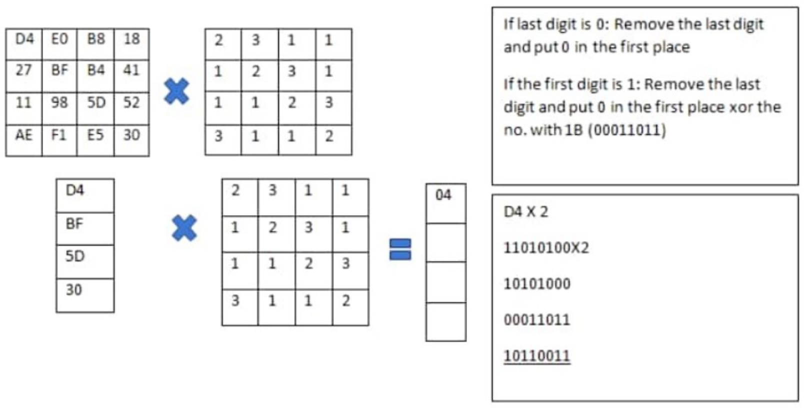

D4 E0 B8 18
27 BF B4 41
11 98 5D 52
AE F1 E5 30
2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
D4
BF
5D
30
04
If last digit is 0: Remove the last digit and put 0 in the first place
If the first digit is 1: Remove the last digit and put 0 in the first place xor the no. with 1B (00011011)
D4 X 2
11010100X2
10101000
00011011
10110011

Figure 3.6: Mix Column

STEP 4 – Add round key

This operation is the same as Initial round Add Round key where XOR operation is performed between 128-bit length text and 128-bit length key.

3.3 Algorithm

To implement an AES CBC (Encryption, Decryption) algorithm is

Step 1 - START

Step 2 - Derive the set of round keys from the 128 bit cipher key.

Step 3 -Initialize the state array with a given block Data –plaintext. XOR plaintext with IV for a First Block and XOR with cipher text for the next consecutive blocks.

Step 4 - Add the Initial round key to the starting state array.

Step 5 - Perform all the ten rounds of state manipulation.

Step 6 - Increment the given block. Repeat step 3 to 5 till all blocks lasts.

Step 7 - STOP

Based on the key size we decide the no of rounds. As the key size here is 128 bit key hence the no of rounds are 10 rounds. After completing the Initial round we computer four main steps .The last round doesn't have Mix Columns round. The Sub bytes does the substitution and shift rows and Mix columns performs the permutation in the algorithm.

- Initial Round
  - Add Round key
- Main Round
  - Sub Bytes

  - Shift rows
  - Mix columns
  - Add Round key
- Last Round
  - Sub Bytes
  - Shift rows
  - Add Round key

The Algorithm is to divide the block. Chain the blocks according to the equation 3.1 and 3.2.The flowchart of the same implementation is in fig 3.7.

Consider a file of length 1280 bits .the file is divided into 10 blocks and then the each block is given to the AES core one by one. First block is XORed with IV and then further blocks is given as the fig 3.7.The reverse of the same operation is done at the Decryption side to get back the text file. [12]

3.4 Flowchart

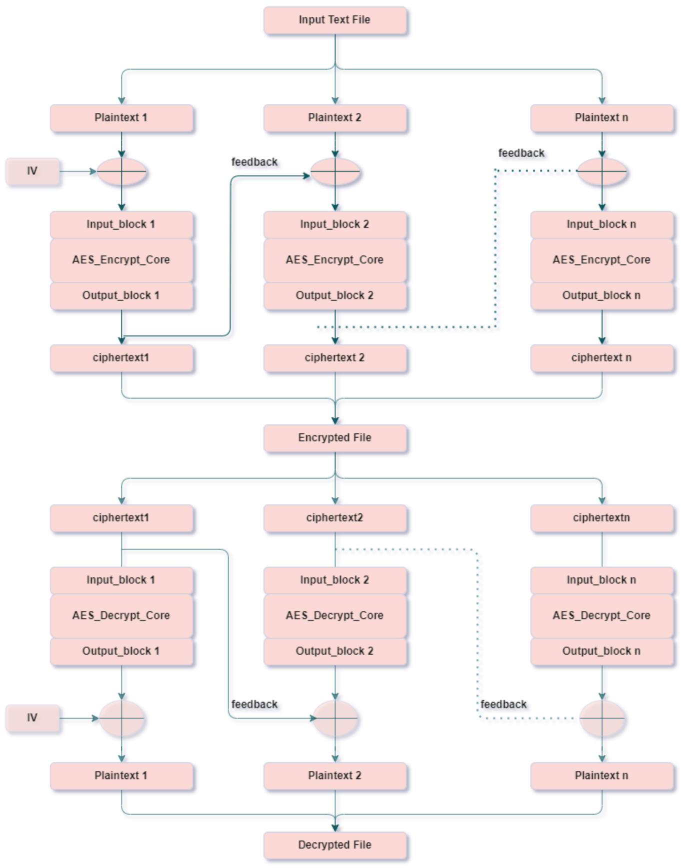


Figure 3.7: Flowchart

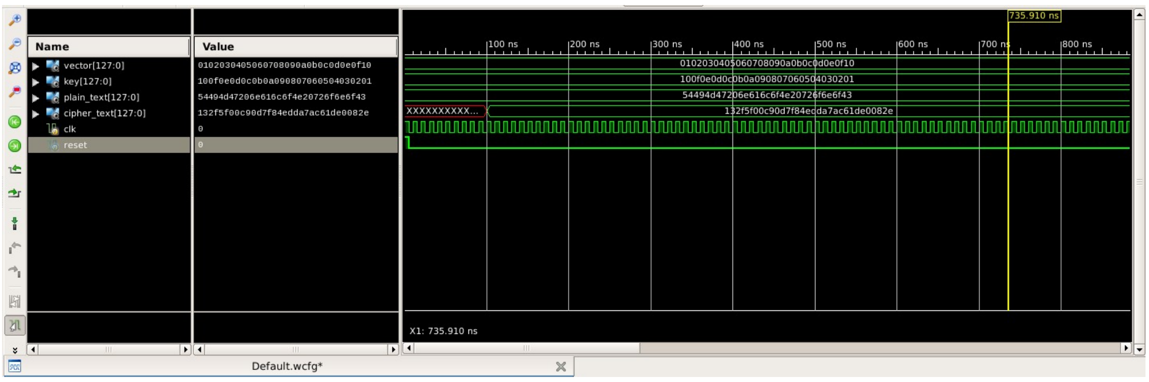
Chapter 4

Results and discussions

This Chapter concludes for the result of AES CBC mode Encryption, Decryption for a block of data.

4.1 Result Analysis

Output of the AES CBC Encryption Implementation. Fig 4.1 depicts results for a single block of data with 128 bit key as key=0x100f0e0d0c0b0a090807060504030201 with the Initial Vector as =0x0120304050607080090a0b0c0d0e0f10.
The system generates the cipher text as 0x132f5f00c90d7ff84edda7ac61de0082e for a given block plain text=0x54494d47206e616c6f4e20726f6e6f43. This output is generated at the sender side for a given block of data. Internally the block 128 bits is divided into a word block [32 bits].

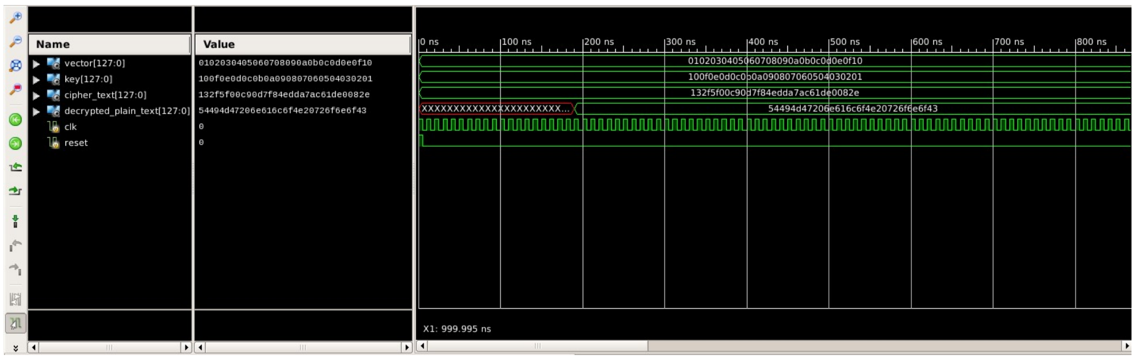


Name	Value
vector[127:0]	0102030405060708090a0b0c0d0e0f10
key[127:0]	100f0e0d0c0b0a090807060504030201
plain_text[127:0]	54494d47206e616c6f4e20726f6e6f43
cipher_text[127:0]	132f5f00c90d7f84edda7ac61de0082e
clk	0
reset	0

735.910 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
0102030405060708090a0b0c0d0e0f10
100f0e0d0c0b0a090807060504030201
54494d47206e616c6f4e20726f6e6f43
XXXXXXXXXX...
132f5f00c90d7f84edda7ac61de0082e
X1: 735.910 ns
Default.wcfg*

Figure 4.1: AES_Encryption

Output of the AES CBC Decryption Implementation. Fig 4.2 depicts results for a single block of data with 128 bit key as key=0x100f0e0d0c0b0a090807060504030201 with the Initial Vector as =0x0120304050607080090a0b0c0d0e0f10.The system generates the plain text as 0x54494d47206e616c6f4e20726f6e6f43.
For a given block cipher text= 0x132f5f00c90d7ff84edda7ac61de0082e. This output is generated at the Receiver side for a given block of data. Internally the block 128 bits is divided into a word block [32 bits]. [14]

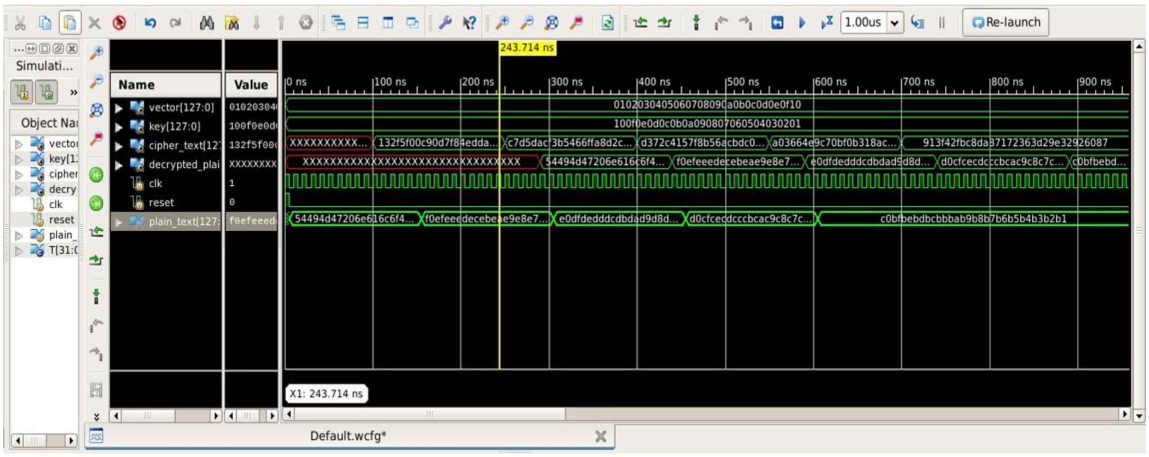


Name	Value
vector[127:0]	0102030405060708090a0b0c0d0e0f10
key[127:0]	100f0e0d0c0b0a090807060504030201
cipher_text[127:0]	132f5f00c90d7f84edda7ac61de0082e
decrypted_plain_text[127:0]	54494d47206e616c6f4e20726f6e6f43
clk	0
reset	0

0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
0102030405060708090a0b0c0d0e0f10
100f0e0d0c0b0a090807060504030201
132f5f00c90d7f84edda7ac61de0082e
XXXXXXXXXXXXXXXXXXXXXX...
54494d47206e616c6f4e20726f6e6f43
X1: 999.995 ns

Figure 4.2: AES_Decryption

Output of the AES CBC Full Implementation. Fig 4.3 depicts results for N block of data with 128 bit key as key=0x100f0e0d0c0b0a090807060504030201 with the Initial Vector as =0x0102030405060708090a0b0c0d0e0f10.The system generates the cipher text for a given block of plain data and in the same implementation the plain text is obtained from the same block of the Cipher text. This output is generated at a single machine side for multiple blocks of data. Internally the block 128 bits is divided into a word block [32 bits]. And each block of data is connected to one another using Cipher Block Chaining. [10]



Name	Value
vector[127:0]	01020304
key[127:0]	100f0e0d
cipher_text[12	132f5f00
decrypted_plai	XXXXXXXX
clk	1
reset	0
plain_text[127:	f0efeeed

Figure 4.3: AES_Multiple Blocks

Slice Logic Utilization and Distribution

Table 4.1: Synthesis of AEC CBC 128 bit Cryptosystem

Slice Logic	LUT Utilization
Number of Slice Registers	5136 out of 126800 4%
Number of Slice LUTs	14480 out of 63400 22%
Number used as Logic	14480 out of 63400 22%
Number with an unused Flip Flop	9789 out of 14925 65%
Number with an unused LUT	445 out of 14925 2%
Number of fully used LUT-FF pairs	4691 out of 14925 31%
Number of unique control sets	1

IO and specific feature Utilization

Table 4.2: IO and specific feature Utilization

Slice Logic	Utilization
Number of IOs	642
Number of bonded IOBs	642 out of 210 305%(*)
Number of Block RAM/FIFO	642 out of 135 47%
Number using Block RAM only	64
Number of BUFG/BUFGCTRLs	1 out of 32 3%

4.2 Discussion on optimization

With reference to Table 4.2 - The IO ports required for implementation of the given logic is 642, whereas available ports are 210. Most of the Consumption here is due to the key and vector as all other parameters can be given in the form of memory. Here each bit output comes in parallel. For both the key and vector alone we require 256 IO ports but we can Implementation the vector as well as key using PISO Implementation. Which in turn helps use reduce the IO ports from 256– 2. And further we can take the plaintext from the ram memory created in the test bench. Where we can read the plain text from file directly and write to it. It takes an additional 256 clock cycles to shift the plaintext and cipher text serially out. [1]

Table 4.3: IO port description

Input/output	IO Required [before optimization]	IO Required [after optimization]
Reset	1	1
Clock	1	1
Plain Text [Blocks]	128	–
Cipher Text [Blocks]	128	1
Decrypted Plain Text Blocks	128	1
Vector	1	1
Private key	128	–

Effect of implementing PISO on IO ports reduces the requirement from 642 – 305% to 5 – 2.3% We use D flip flops to shift the data serially. After a particular amount of time delay,

Table 4.4: IO ports Comparison

Slice Logic	**LUT Utilization**
Number of bonded IOBs	642 out of 210 305% (*)
Number of bonded IOBs	5 out of 210 2.3%

The system generates the cipher text for the individual blocks and then these block written to memory and hence generating the file . Hence we use a High frequency clock to reduce the delay. With which we need not comprise on hardware or on delay. Fig shows the implementation of a 4 bit PISO register -Using 4 D flip flops. [7]

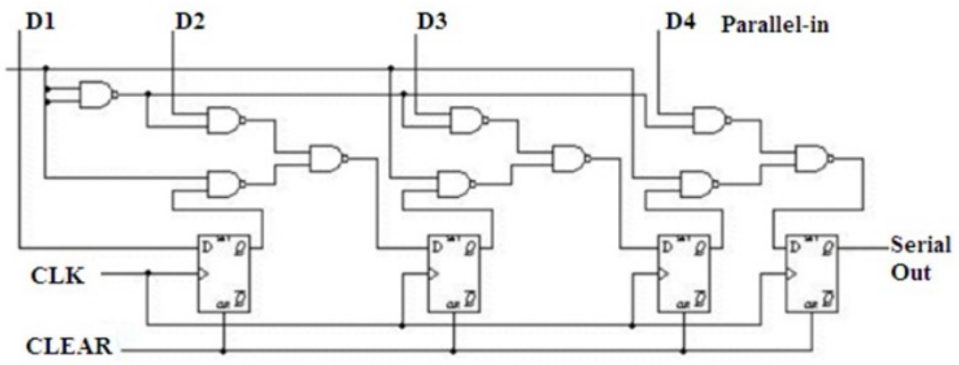


Figure 4.4: PISO Implementation using D latch

Chapter 5

Conclusions and future scope

5.1 Conclusion

To secure electronic data we make use of various cryptography algorithms one such algorithm is the advanced encryption standard. In this paper, we successfully implemented the design of the Efficient Advanced Encryption Standard using Cipher Block Mode. The Cipher Block Chaining (CBC) uses an initialization vector- IV and makes use of a block cipher algorithm. In this chaining of the plain data block with the prior cipher, the data block is done. The initialization vector and the block to be encrypted are of the same size. The initialization vector IV chosen is an arbitrary number. As the plain text that is being sent in the real world is of varying lengths it is divided into blocks and needs to add padding data of 128 bits before encryption or decryption. The plaintext block is then Xor 'ed with the initialization vector IV. Using a block cipher algorithm, it is converted to cipher block text. Encryption result is used to xor with the plaintext block until the last block. When it comes to decrypting the data, it can be decrypted in parallel which is not possible when encrypting the data block. As it follows chaining a one-bit change in plaintext or cyphertext block will affect all the following blocks. On analyzing the results, it takes 256 clock cycles to shift the plain text and cipher text serially out. The effect of implementing PISO on IO ports reduces the requirement from 642 – 305% to 5 – 2.3%.

5.2 Future scope

In the future, work can be done in generating more secure keys to enhance the encryption speed to achieve the protection of classified data. Furthermore, working on these two things which are the optimization of memory and utilization of resources can help in designing a small and fast hardware design. The optimization can be done by using the pre-calculated tables, exploiting the resemblance between the steps used in encryption and decryption. As the concept of the AES algorithm is becoming popular in the field of both software and hardware design, we can further work on improving the chip area and power consumption. In some cases, security is valued more than the efficiency of the design. It can be further used for image communication, IoT based projects, defense communication, government agencies, etc. Due to the advancement in cloud computing technology, it can be used in storing the data at the storage level, where data can be encrypted using the AES algorithm and then uploaded to the cloud.

Bibliography

[1] Kazumaro Aoki and Helger Lipmaa. Fast implementations of aes candidates. In *AES Candidate Conference*, pages 106–120. Citeseer, 2000.

[2] Hayder T Assafli and Ivan A Hashim. Generation and evaluation of a new time-dependent dynamic s-box algorithm for aes block cipher cryptosystems. In *IOP Conference Series: Materials Science and Engineering*, volume 978, page 012042. IOP Publishing, 2020.

[3] Hayder T Assafli and Ivan A Hashim. Security enhancement of aes-cbc and its performance evaluation using the avalanche effect. In *2020 3rd International Conference on Engineering Technology and its Applications (IICETA)*, pages 7–11. IEEE, 2020.

[4] Hayder T Assafli, Ivan A Hashim, and Ahmed A Naser. The evaluation of time-dependent initialization vector advanced encryption standard algorithm for image encryption. *Engineering and Technology Journal*, 40:08, 2022.

[5] M Razvi Doomun, KM Sunjiv Soyjaudah, and Devesh Bundhoo. Energy consumption and computational analysis of rijndael-aes. In *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*, pages 1–6. IEEE, 2007.

[6] Razvi Doomun, Jayramsingh Doma, and Sundeep Tengur. Aes-cbc software execution optimization. In *2008 International Symposium on Information Technology*, volume 1, pages 1–8. IEEE, 2008.

[7] Ahmed Fathy, Ibrahim F. Tarrad, Hesham F. A. Hamed, and Ali Ismail Awad. Advanced encryption standard algorithm: Issues and implementation aspects. In *AMLTA*, 2012.

[8] Daniel F García. Performance evaluation of advanced encryption standard algorithm. In *2015 Second International Conference on Mathematics and Computers in Sciences and in Industry (MCSI)*, pages 247–252. IEEE, 2015.

[9] Sungha Kim and Ingrid Verbauwhede. Aes implementation on 8-bit microcontroller. *Department of Electrical Engineering, University of California, Los Angeles, USA*, 2002.

[10] Hyeopgeon Lee, Kyounghwa Lee, and Yongtae Shin. Implementation and performance analysis of aes-128 cbc algorithm in wsns. In *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, volume 1, pages 243–248. IEEE, 2010.

[11] G Manjula and HS Mohan. Improved dynamic s-box generation using hash function for aes and its performance analysis. In *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 109–115. IEEE, 2018.

[12] Ganesh Gopal Shet, Jamuna, S. Shravani, G NayanaH, and S PramodKumar. Implementation of aes algorithm using verilog. *JNNCE Journal of Engineering and Management*, 2020.

[13] Rei Ueno, Sumio Morioka, Naofumi Homma, and Takafumi Aoki. A high throughput/gate aes hardware architecture by compressing encryption and decryption datapaths. In *International conference on cryptographic hardware and embedded systems*, pages 538–558. Springer, 2016.

[14] Rei Ueno, Sumio Morioka, Noriyuki Miura, Kohei Matsuda, Makoto Nagata, Shivam Bhasin, Yves Mathieu, Tarik Graba, Jean-Luc Danger, and Naofumi Homma. High throughput/gate aes hardware architectures based on datapath compression. *IEEE Transactions on Computers*, 69(4):534–548, 2019.

[15] Drashti O Vadaviya and P Tandel. Study of avalanche effect in aes. In *National Conference on Recent Advances in Engineering for Sustainability*, pages 1–4, 2015.

[16] M Vaidehi and B Justus Rabi. Design and analysis of aes-cbc mode for high security applications. In *Second International Conference on Current Trends In Engineering and Technology-ICCTET 2014*, pages 499–502. IEEE, 2014.