International Journal of Innovative Research in Engineering
Volume 4, Issue 2 (March-April 2023), PP: 147-155.
www.theijire.com ***ISSN No: 2582-8746***

VLSI Design of Advanced-Features AES Crypt Processor for Cryptography

Titiksha V[1], Surya M[2], Henry Bruno D G[3]
[1,2,3] Electronics and communication engineering , Bannari Amman Institute of Technology, Tamil Nadu, India.

How to cite this paper:
Titiksha V[1], Surya M[2] , Henry Bruno D G[3], " VIsi Design of Advanced-Features Aes Crypt Processor For Cryptography", IJIRE-V4I02-147-155.

Abstract: *Advanced Encryption Standard (AES) is a specification for electronic data encryption. This standard has become one of the most widely used encryption method and has been implemented in both software and hardware.A high-secure symmetric cryptography algorithm, The proposed architecture includes 8-bit data path and five main blocks. We design two specified register banks, Key-Register and State-Register, for storing the plain text, keys, and intermediate data. To reduce the area, Shift-Rows is embedded inside the State- Register. To adapt the Mix-Column to 8-bit datapath, we design an optimized 8-bit block for Mix-Columns with four internal registers, which accept 8-bit and send back 8-bit. Also, shared optimized Sub-Bytes are employed for the key expansion phase and encryption phase. To optimize Sub-Bytes, we merge and simplify some parts of the Sub-Bytes. To reduce power consumption, we apply the clock gating technique to the design. This paper presents an Image Cryptography based 128-bit AES design. This Design is implemented using Verilog HDL and simulated by Modelsim 6.4 c.*

Key Word: *Advanced Encryption Standard , Sub bytes, Mix-column, Shift-rows.*

I.INTRODUCTION

Encryption in other Words is also known as Cryptography, is the method of creating and using a crypto-system or cipher to prevent all the intended recipient from reading or utilizing the information or the application encrypted. A crypto-system is a technique used to encode a message. The recipient can view the encrypted message only by decoding it with the correct algorithm and keys. Cryptography is primarily used for communicating with the sensitive material across the computer networks. The encryption process takes a clear text document and applies it to a key and with the mathematical algorithm to it, that converts it into a crypto-text. In a crypto-text, the document cannot be read unless the reader gives the key that can undo the encryption. The US government's National Institute of Standards and Technology (NIST), a division, began looking for a replacement for the Data Encryption Standard in 1997. (DES). It was well acknowledged that due to improvements in computer processing capacity, DES was not secure. NIST's objective was to propose a DES replacement that US government organizations could adopt for non-military information security applications. Naturally, it was acknowledged that the work of NIST would assist business and other non-government users and that the work would normally be adopted as a commercial standard. The NIST invited experts in data security and cryptography from all over the world to take part in the selection and deliberation process. For research, five encryption techniques were chosen. The encryption technique suggested by Belgian cryptographers Joan Daeman and Vincent Rijmen was chosen through a consensus-building process. Before choosing, Daeman and Rijnmen called the algorithm Pipelined, which is a combination of their names. The Advanced Encryption Standard (AES), which is still widely used today, was given to the encryption algorithm when it was adopted. The AES encryption algorithm was formally adopted by the NIST in 2000 and published as a federal standard under the name FIPS-197. You can access the complete FIPS-197 standard on the NIST website (see the Resources section below). As was to be expected, numerous manufacturers of encryption hardware and software have included AES encryption to their offerings.

II.METHODOLOGY

AES is a symmetric block cipher. This means that it uses the same key for both encryption and decryption. However, AES is quite different from DES in a number of ways. The algorithm Rijindael allows for a variety of block and key sizes and not just the 64 and 56 bits of DES' block and key size. The block and key can in fact be chosen independently from 128, 160, 192, 224, 256 bits and need not be the same. However, the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128, 192, 256 bits. Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES- 256 respectively. As well as these differences AES differs from DES in that it is not a feistel structure. Recall that in a feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped. In this case the entire data block is processed in parallel during each round using substitutions and permutations. The Pipelined algorithm is a symmetric iterated block cipher. The block and key lengths can be 128, 192, or 256 bits. The NIST requested that the AES must implement a symmetric block cipher with a block size of 128 bits. Due to this requirement, variations of Pipelined that can operate on larger block sizes will not be included in the actual standard. Pipelined also has a variable number of iterations or rounds: 10, 12, and 14 when the key lengths are 128, 192, and 256 respectively. The transformations in Pipelined consider the data block as a four-column rectangular array of 4-byte vectors. The key is also considered to be a rectangular array of 4-byte vectors—the number of columns is dependent on key length. Pipelined decryption comprises the inverse of the transformations that encryption uses, performed in reverse order. Decryption

commences with the inverse of the final round, followed by the inverses of the rounds, and finishes with the initial data/key addition, which is its own inverse.

AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix. Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. A number of AES parameters depend on the key length. For example, if the key size used is 128 then the number of rounds is 10 whereas it is 12 and 14 for 192 and 256 bits respectively. At present the most common key size likely to be used is the 128 bit key. This description of the AES algorithm therefore describes this particular implementation.

Rijindael was designed to have the following characteristics:

• Resistance against all known attacks.

• Speed and code compactness on a wide range of platforms.

• Design Simplicity

The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it's counterpart in the encryption algorithm. The four stages are as follows:

1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consist of the following:

1. Inverse Shift rows
2. Inverse Substitute bytes
3. Inverse Add Round Key
4. Inverse Mix Columns

The schematic of AES structure is given in the (figure 5.1)

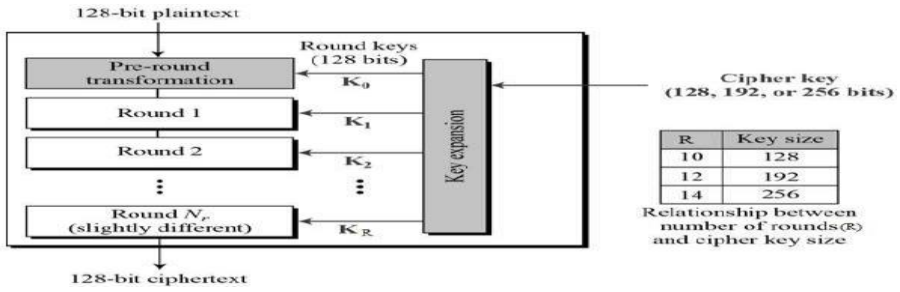

Figure 5.1 AES structure

The AES Encryption and Decryption process is mentioned in bellow(figure 5.2). The process of decryption of an AES cipher text is similar to the encryption process in the reverse order. Each round consists of the four processes

- Add round key
- Mix columns
- Shift rows
- Byte substitution

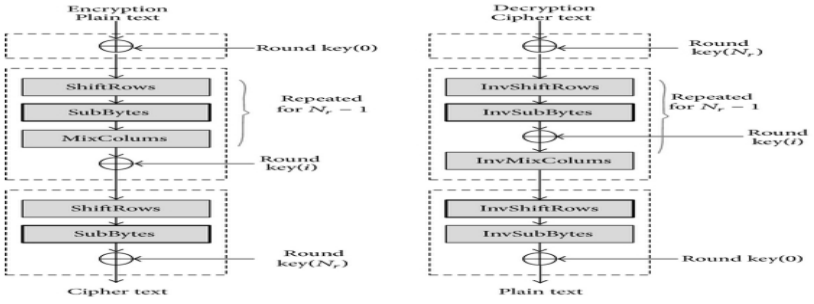

Figure 5.2 AES encryption and decryption process

III.PROPOSED SYSTEM

The AES implementation consists of the masked AES core and Clock gating to generate the encryption masks. The masked AES core performs 128 bit encryption. The process is done in 10cycles, computing 1 round per cycle, with the hardware of each round being reused to save area verses a fully unrolled implementation. The proposed masked AES is shown in bellow Figure. Where the original data (plaintext) is first masked by a random mask. The masked plaintext and the mask are, then, fed through the "Nano AES core" which encrypts the masked data with the secret key. Result masked cipher-text is input into the module to arrive at the intended cipher-text.

1) In order to reduce the required logic, the Shift-Rows are embedded inside the State-Register.
2) We optimize Sub-Bytes block and share it with key expansion phase and encryption phase.
3) We design an optimized 8-bit block for Mix-Columns with 8-bit input and output that is based on the structure of 8-bit data path, which is followed by Add-Round-Key. Thus, the results send to Add-Round- Key byte-by-byte. In comparison to 32-bit Mix-columns, it is not necessary to store the results in the registers or increase the data path for Key-Register to 32-bit.
4) To reduce the power consumption of the design, the clock gating technique is applied in different parts of the design, which leads to reduce the power consumption.

In the different parts of the design, we apply the clock gating technique to reduce the dynamic power consumption. The clock gating is separately applied on State-Register, the internal registers of Mix-Columns, Key-Register, and RCON. For instance, the most power consumption is saved during the key expansion phase; the clock of State-Register and Mix- Columns is disabled to save power because these two blocks are not used in the key expansion phase.

Clock Gating:

Clock gating is a well-known technique to reduce chip dynamic power. Recent clock gating techniques based on ACG(Adaptive Clock Gating) and instruction level clock gating. clock gating technique reduces not only switching activity of functional blocks in IDLE state but also dynamic power in running state. Our modified ACG (Adaptive Clock Gating) can automatically enable or disable the clock of the functional block. Clock gating is a popular technique used in many synchronous circuits for reducing dynamic power dissipation. Clock gating saves power by adding more logic to a circuit to prune the clock tree.

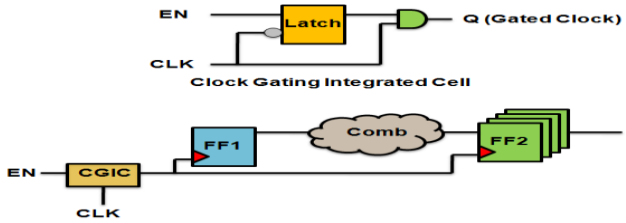

Figure 3.1 Clock gating

Everywhere acceptance of portable devices such as cell phones, PDAs and mp3 players has much research in the development of technique for low-power design. The continuous decrease in the minimum feature size of transistors which increase of both device density and design complexity. The overall power dissipation on a chip is due to clock and data-path.

The clock -gating is one of the effective logic in RTL and architectural power reduction.

Clock gating is an effective technique to reduce dynamic power, because individual IP usage varies across applications, not all IP cores are used all the time, giving rise to opportunity for reducing the unused IP cores' power. By combining(AND gate) the clock with a gate-control signal, clock gating essentially disables the clock to an IP core when that IP is not used, avoiding power dissipation due to unnecessary charging and discharging of the unused circuits.

The clock -gating is one of the effective logic in RTL and architectural power reduction. Clock gating is an effective technique to reduce dynamic power.

Proposed System Block Diagram:

Below figure 3.2 represents the proposed system block scheme

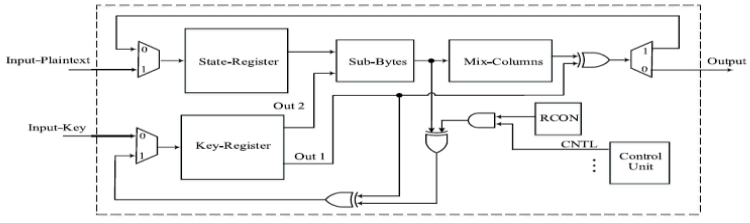

Figure 3.2 Block scheme of the proposed

Proposed Application Block Diagram:

We will design a Decryption Scheme based on Proposed Scheme from below (figure 3.3). We are going to Make Encryption & Decryption Based on Images. We are going to merge the Matlab& VLSI for this Process.

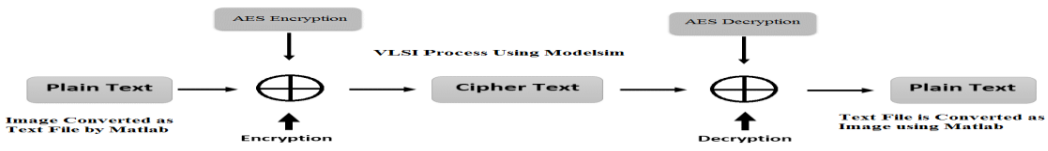

Figure 3.3 Proposed application block diagram

Proposed Algorithm:

Below (figure 3.4) represents the proposed algorithm

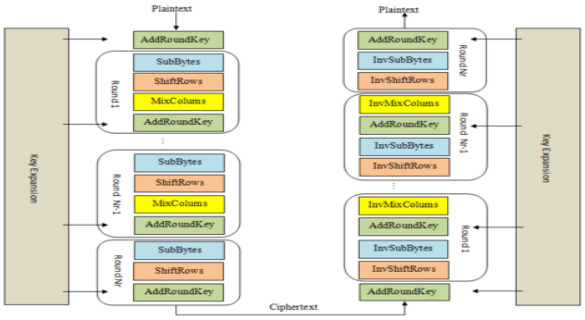

Figure 3.4 Proposed algorithm

IV.DESIGN APPROCH

Substitution Box (S- Box):

A nonlinear byte replacement is what the Sub Bytes operation does. According to the replacement box, each byte from the input state is replaced with a different byte (called the S-box). The S-box is calculated via a bitwise affine translation and a multiplicative inverse in the finite field GF (28).

In the example below (figure 4.1), combinational logic circuits are used to implement the composite field S-BOX rather than using S-BOX values that have already been recorded.

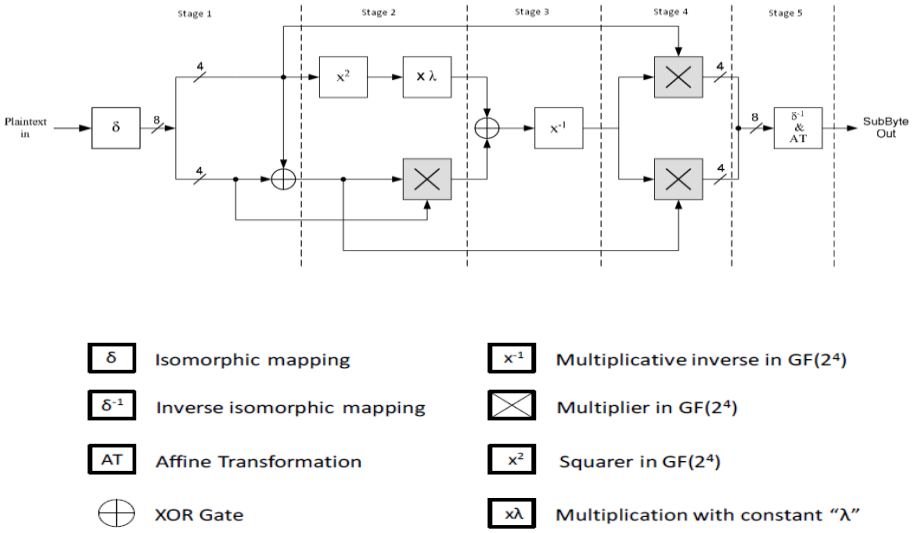

Figure 4.1 S-box implementation

Addition in GF (2^4):

Galois Field addition of two elements can be expressed as a straightforward bitwise XOR operation. Galois Field addition of two elements translates to a straightforward bitwise XOR operation.

GF (2^4) Multiplier

A nonlinear transformation called Sub Bytes makes use of 16-byte replacement tables (S-Boxes). An S-Box is the affine transformation of the multiplicative inverse of a Galois field GF (24) from (figure 4.2). Despite the fact that two Galois Fields of the same order are isomorphic, the complexity of the field operations may be greatly influenced by the way the field elements are represented. To simplify the hardware, composite field arithmetic might be used.

Finding the multiplicative inverse in GF (28) requires three multipliers in GF (24). The GF (24) multiplier circuit is depicted in Fig. The GF (24) multipliers, as seen in the picture, are made up of 3 GF (22) multipliers with 4 XOR Gates and a constant multiplier of. The higher output bit will be the outcome of an XOR operation between the two input bits in this constant multiplier, which takes two input bits and extracts the lower bit output as the higher bit input. You may get the whole derivation of this multiplier circuit.

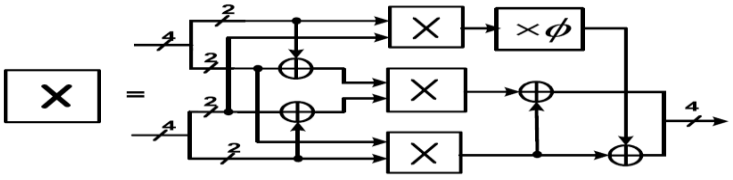

Figure 4.2 GF (24) Multiplier

GF (2^2) Multiplier

There exist an infinite number of possible finite fields, however each one is not infinite in and of itself. Each finite field's element count, or cardinality, must take the form pn, where p is a prime number and n is a positive integer. The Implementation Of The GF (22) Multiplier is seen in (figure 4.3)..

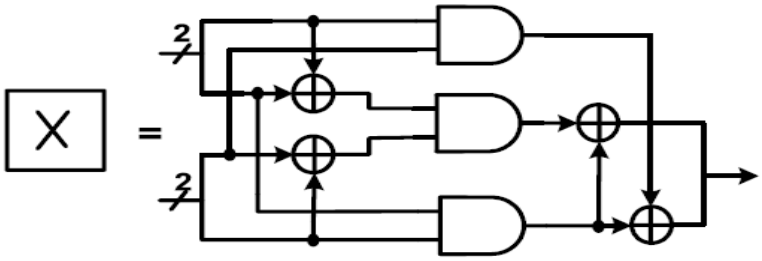

Figure 4.3 GF (2^2) Multiplier

GF (2^4) SQUARER:

The GF (24) Squarer is depicted in the figure 4.4 below. Bitwise xor operation is what it consists of. One or more bit patterns or binary numbers are subject to a bitwise operation at the level of the individual bits. It is a quick, simple action that is supported directly by the processor and is used to change data in order to make comparisons and calculations. Bitwise operations are frequently noticeably faster than addition, many times faster than multiplication, and much faster than division on basic low-cost processors. Bitwise operations may still be preferable for overall power/performance due to their reduced resource usage, even though addition and multiplication are typically performed as quickly as bitwise operations by modern high-performance processors.

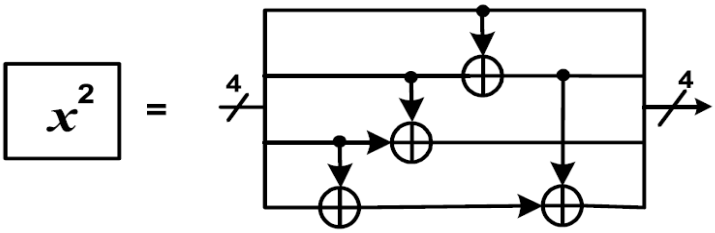

Figure 4.4 Hardware diagram for Squarer in GF (2^4)

Constant Multiplier (Xϕ):

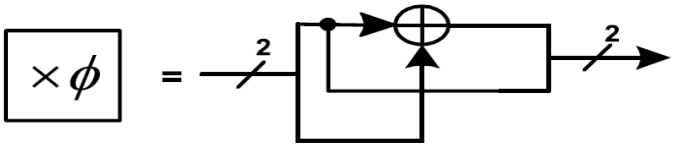

Figure 4.5 Constant multiplier

From (figure 4.5) We have done Multiplication Of Hardware Implementation using Constant Φ

Constant Multiplier (Lambda):

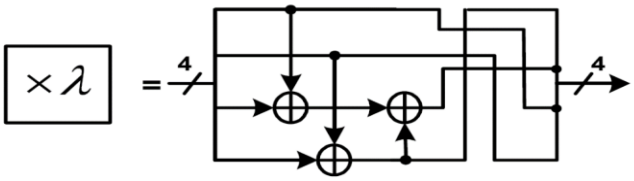

figure 4.6 Hardware Diagram For Multiplication With Constant

Add round key Transformation:

By using Bitwise Exclusive-OR (XOR), a round key is added to the state in the Add Round Key transformation. The Add Round Key is seen in (Figure 4.7) below. Both encryption and decryption use the same transformation.

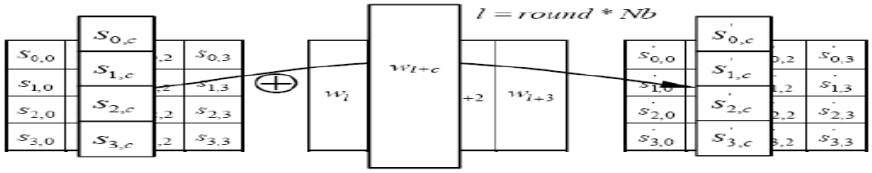

Figure 4.7 Add Round Key

Shift Rows Transformation (Inv Shift Rows):

Each row of the State undergoes a cyclical shift operation called Shift Rows. The state's first row of bytes do not change during this operation. As shown in, the second, third, and fourth rows cyclically move to the left by one byte, two bytes, and three bytes, respectively (Figure 4.8). Shift Rows are processed in reverse order by the reverse procedure, inv Shift Row..

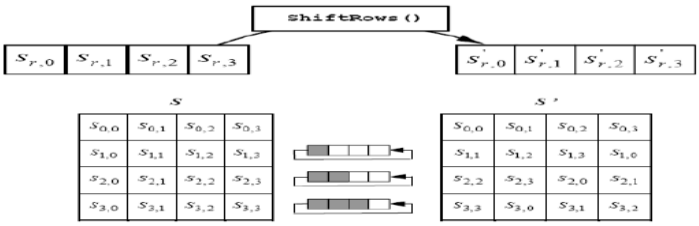

Figure 4.8 Inverse shift rows

Mix Column Transformation (Inv Mix Column):

Column per column, the state is individually transformed using the Mix Column function. Since GF (28) is a four-term polynomial, each column is multiplied by

$a(x)$ modulo $(x4 + 1)$ where $a(x) = \{03\}x3 + \{01\}x2 + \{01\}x + \{02\}$ given in the (figure 4.9)

This transformation can be expressed in matrix form as

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} \{02\} & \{03\} & \{01\} & \{01\} \\ \{01\} & \{02\} & \{03\} & \{01\} \\ \{01\} & \{01\} & \{02\} & \{03\} \\ \{03\} & \{01\} & \{01\} & \{02\} \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

For *invMixColumn()*, replace $a(x) = \{0E\}x^3 + \{09\}x^2 + \{0D\}x + \{0B\}$.

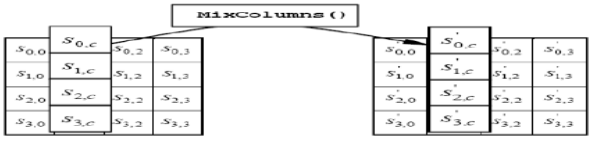

Figure 4.9 Inverse mix column

Transformation Matrix & Inverse Transformation Matrix:

A transformation matrix (M) transforms the elements in the binary field to the composite field GF $((2^3)^3)$. Then, the operations are done in composite fields to achieve the inverse which is then retransformed to binary field using an inverse transformation matrix (M^{-1}). Eventually, the two most and least significant bits are deleted to get to the uneven structure of the substitution box is explained in below (figure 4.10).

$$\beta \times q = \begin{pmatrix} 1&0&1&0&0&0&0&0 \\ 1&1&0&1&1&1&1&0 \\ 1&0&1&0&1&1&0&0 \\ 1&0&1&0&1&1&1&0 \\ 1&1&0&0&0&1&1&0 \\ 1&0&0&1&1&1&1&0 \\ 0&1&0&1&0&0&1&0 \\ 0&1&0&0&0&0&1&1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} \quad \delta^{-1} \times q = \begin{pmatrix} 1&1&1&0&0&0&1&0 \\ 0&1&0&0&0&1&0&0 \\ 0&1&1&0&0&0&1&0 \\ 0&1&1&1&0&1&1&0 \\ 0&0&1&1&1&1&1&0 \\ 1&0&0&1&1&1&1&0 \\ 0&0&1&1&0&0&0&0 \\ 0&1&1&1&0&1&0&1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix}$$

Figure 4.10 Transformation Matrix & Inverse Transformation Matrix

S BOX Truth Table:

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

figure 4.11 s-box truth table

Key Expansion Unit:

The key expansion uses the same approach as the encryption path but with additional S-box optimizations and data loading for various key sizes into the key registers. When not in use, the S-box inputs (figure 4.12) are covered by constant values to reduce dynamic power usage. To conserve space, the expanded key is instantly calculated and transmitted back into

the key registers. A key expansion module consists of two shift registers with four stages each, as well as a key transform module with four S-boxes and an XOR.

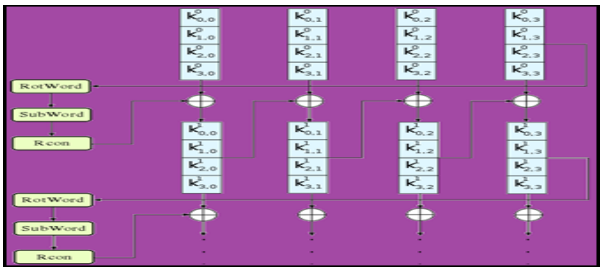

Figure 4.12 Key Expansion Unit

Flow Chart:

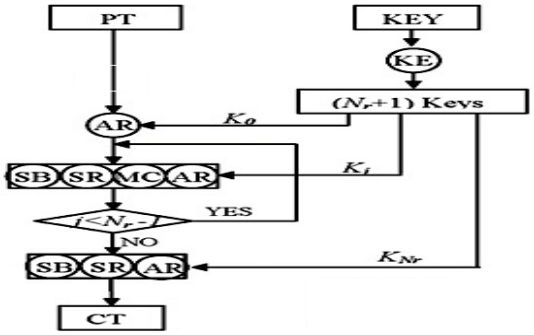

Figure 4.13 AES design flow chart

V.RESULTS

Sub Bytes:

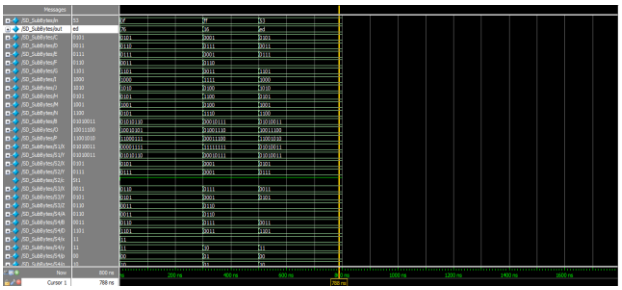

Figure 8.1 Sub bytes output

Shift Rows:

Figure 8.2 shift rows output

Mixed Column:

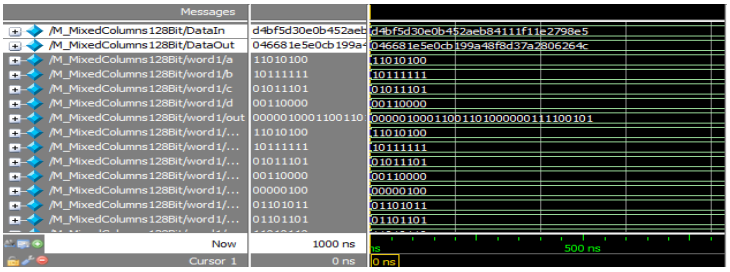

Figure 8.3 mixed column output

Add Round Key:

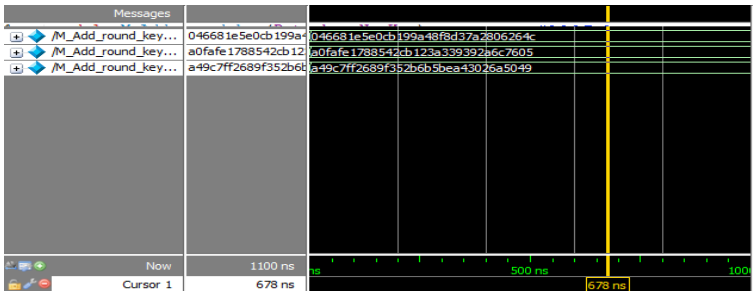

Figure 8.4 Add Round Key output

Encryption:

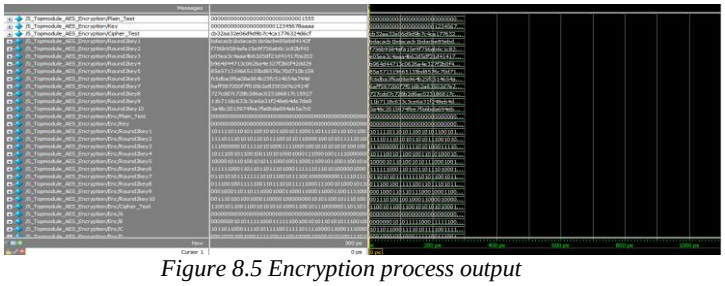

Figure 8.5 Encryption process output

Single Round Operation:

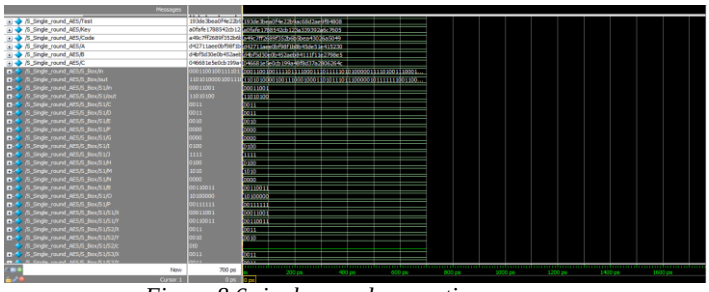

Figure 8.6 single round encryption process

AES Encryption:

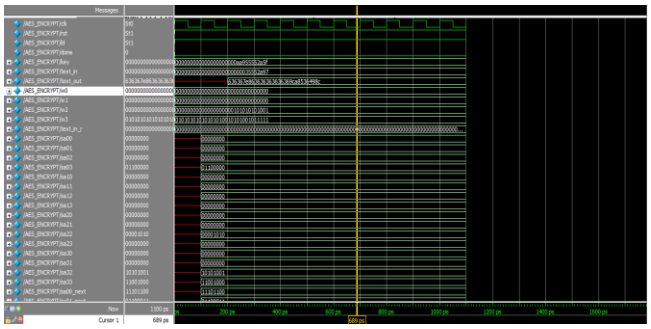

Figure 8.7 AES Encryption

Clock Gating

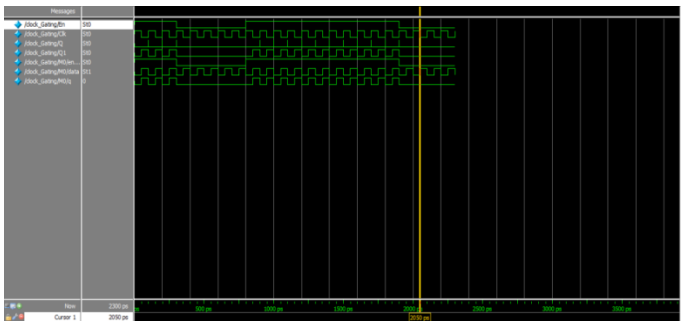

Figure 8.8 Clock gating

Nano Aes Clock Gating Encryption

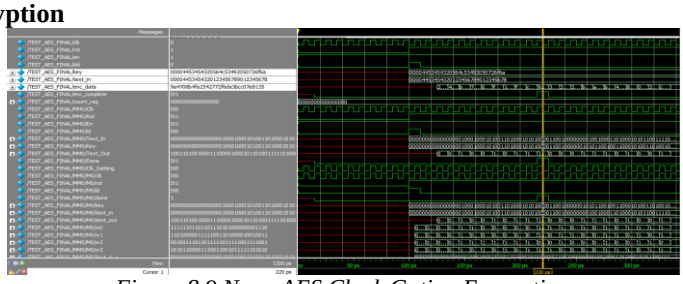

Figure 8.9 Nano AES Clock Gating Encryption

Nano Aes Clock Gating Encryption & Decryption

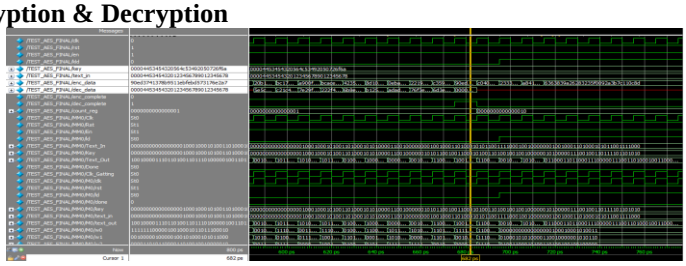

Figure 8.10 Nano AES clock gating encryption and decryption

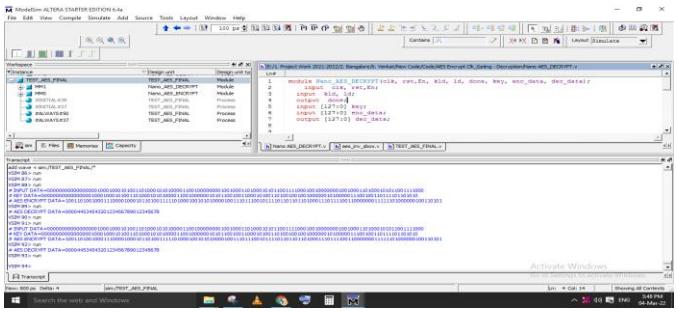

Figure 8.11 AES final output

VI.CONCLUSION

Nano AES is a popular symmetric cryptography algorithm that is highly safe and is utilised in several networks and applications. AES is a viable algorithm for small IoT devices as a result. In this paper, we developed a compact AES architecture for IoT devices with limited resources. The architecture had an 8-bit data path and two designated register banks for storing keys, intermediate results, and plain text. Shift-Rows were executed inside of the State- Register to lessen the amount of logic needed. Also, the design shared optimised Sub-Bytes with the encryption and key expansion phases. In addition, we created mix-Columns, a suitable block for low-area design, with 8-bit input and output.

References

1. *K. Fu and J. Blum, "Controlling for cybersecurity risks of medical device software," Commun. ACM, vol. 56, no. 10, pp. 35–37, Oct. 2013.*
2. *D. Halperin, T. Kohno, T. S. Heydt-Benjamin, K. Fu, and W. H. Maisel, "Security and privacy for implantable medical devices," IEEE Pervasive Comput., vol. 7, no. 1, pp. 30–39, Jan./Mar. 2008.*
3. *M. Rostami, W. Burleson, A. Jules, and F. Koushanfar, "Balancing security and utility in medical devices?" in Proc. 50th ACM/EDAC/IEEE Int. Conf. Design Autom., May/Jun. 2013, pp. 1–6.*
4. *M. Zhang, A. Raghunathan, and N. K. Jha, "Trustworthiness of medical devices and body area networks," Proc. IEEE, vol. 102, no. 8, pp. 1174–1188, Aug. 2014.*
5. *H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, "Smart-grid security issues," IEEE Security Privacy, vol. 8, no. 1, pp. 81–85, Jan./Feb. 2010.*
6. *M. Mozaffari-Kermani, M. Zhang, A. Raghunathan, and N. K. Jha, "Emerging frontiers in embedded security," in Proc. 26th Int. Conf. VLSI Design, Jan. 2013, pp. 203–208.*
7. *R. Roman, P. Najera, and J. Lopez, "Securing the Internet of things," Computer, vol. 44, no. 9, pp. 51–58, Sep. 2011.*
8. *T. H.-J. Kim, L. Bauer, J. Newsome, A. Perrig, and J. Walker, "Challenges in access right assignment for secure home networks," in Proc. USENIX Conf. Hot Topics Secur., 2010, pp. 1–6.*
9. *M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure independent fault detection schemes for the Advanced Encryption Standard," IEEE Trans. Comput., vol. 59, no. 5, pp. 608–622, May 2010.*
10. *M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A low-power high performance concurrent fault detection approach for the composite field S-box and inverse S-box," IEEE Trans. Comput., vol. 60, no. 9, pp. 1327–1340, Sep. 2011.*
11. *M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight high performance fault detection scheme for the Advanced Encryption Standard using composite fields," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 1, pp. 85–91, Jan. 2011.*