Designing of S-Box Based On Null Convention Logic

D.Swathi[1], Mr.P.Gangadhar[2], D.V.Supriya[3]

[1] *PG Scholar(ES), Department of ECE, ABIT, Siddavatam, Andhra Pradesh, India*
[2]Head Of the Department, *Department of ECE*, ABIT,*Siddavatam, Andhra Pradesh, India*
[3]Asst Professor, *Department of ECE*, ABIT,*Siddavatam, Andhra Pradesh, India*

---***---

Abstract - ***A novel asynchronous combinational S-Box (substitution box) is design for AES (Advanced Encryption Standard) crypto-systems is proposed and validated. The S-Box is considered as the most critical component in AES crypto systems. The proposed design Substitution-Box is implemented based on a delay-insensitive (DI) logic paradigm known as Null convention logic (NCL), which supports useful properties for resisting SCAs including dual-rail encoding, clock-free operation, self-timed logic and monotonic transitions. These advantageous properties make it difficult for an attacker to decipher secret keys. The proposed NCL based Substitution Box have been done using Mentor Graphics EDA (Electronic Design Automation) tools.***

Key Words: *Advanced Encryption Standard(AES),NCL Delay-Insensitive (DI), S-Box, SCAs.*

1. INTRODUCTION

The National Institute of Standards and Technology, (NIST), solicited proposals for the Advanced Encryption Standard, (AES). The AES is a Federal Information Processing Standard, (FIPS), which is a cryptographic algorithm that is used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt, (encipher), and decrypt, (decipher), information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. Many algorithms were originally presented by researchers from twelve different nations. Fifteen (15), algorithms were selected from the first set of submittals. After a study and selection process five(5), were chosen as finalists. The five algorithms selected were MARS, RC6, RIJNDAEL, SERPENT and TWOFISH. The conclusion was that the five Competitors showed similar characteristics. On October 2nd 2000, NIST announced that the Rijndael Algorithm was the winner of the contest. The Rijndael Algorithm was chosen since it had the best overall scores in security, performance, efficiency, implementation ability and flexibility. The Rijndael algorithm was developed by Joan Daemen of Proton World International and Vincent Fijmen of Katholieke University at Leuven. The Rijndael algorithm is a symmetric block cipher that can process data blocks of 128 bits through the use of cipher keys with lengths of 128, 192, and 256 bits. The Rijndael algorithm was also designed to handle additional block sizes and key lengths. The AES cryptographic algorithm has four transformations for both encryption and decryption. They are ADD ROUNDKEY, SUBBYTES, SHIFTROWS and MIXCOLUMNS. The AES algorithm consists of a number of rounds that are dependent on the key size. SUBBYTES step is the first step of AES round. Each byte in the array is updated by an 8-bit S-Box, which is derived from the multiplicative inverse over *G.F (28)*. The S-Box is one of the most critical components in the implementation of AES hardware. It consumes the majority of power and is also the most vulnerable component to SCAs. Hence low power S-Box is constructed using null convention logic which is a self-timed logic .The S-Box consists of multiplicative inverse, affine transformation and inverse affine transformation, by combining the inverse function with an invertible affine transformation we can avoid attacks based on mathematics.

1.1 Null Convention Logic

NULL Convention Logic (NCL) is a symbolically complete logic which expresses process completely in terms of the logic itself and inherently and conveniently expresses asynchronous digital circuits. The traditional form of Boolean logic is not symbolically complete in the sense that it requires the participation of a fundamentally different form of expression, time in the form of the clock, which has to be very carefully coordinated with the logic part of the expression to completely and effectively express a process. We introduce NULL Convention Logic in relation to Boolean logic as a four value logic, and as a three value logic and finally as two value logic quite different from traditional Boolean logic. NULL Convention Logic (NCL) provides an asynchronous design methodology employing dual-rail signals, quad-rail signals. A Dual rail signal, D consists of two wires or rails D0 and D1 . These signals take any value from the data set {DATA0, DATA1, and NULL} as illustrated in Table-1.

Table -1: DUAL RAIL LOGIC

	DATA0	DATA1	NULL	ILLEGAL
D^0	1	0	0	1
D^1	0	1	0	1

The following table-2 shows example of AND gate Implemented by using Null Convention Logic.

Table -2: EXAMPLE FOR NCL AND GATE

A	B	AND GATE	NCL LOGIC AND GATE
0	0	0	01
0	1	0	01
1	0	0	01
1	1	1	10

2. SUBSTITUTION-BOX

Sub-Bytes or S-Box is a nonlinear transformation that uses 16 byte substitution tables (S-Boxes). An S-Box is the multiplicative inverse of a Galois field GF(2^4) followed by an affine transformation. Although two Galois Fields of the same order are isomorphic, the complexity of the field operations may heavily depend on the representations of the field elements. Composite field arithmetic can be employed to reduce the hardware complexity. Three multipliers in GF(2^4) are required as a part of finding the multiplicative inverse in GF(2^8). Fig. shows the GF(2^4) multiplier circuit. As can be seen from the figure the GF(2^4) multipliers consist of 3 GF(2^2) multipliers with 4 XOR Gates and with constant multiplier θ. This constant multiplier which has 2 bits input extracts the lower bit output as the higher bit input, while the higher output bit will be the result of XOR operation between the 2 input bits. Full derivation of this multiplier circuit can be found. The following Fig-1 shows S-Box Architecture.

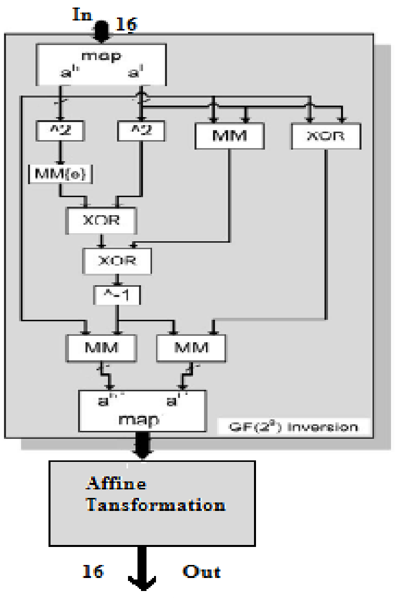


Fig -1: S-Box Architecture.

The legends for the blocks within the multiplicative inversion module from above are illustrated in the Figure.

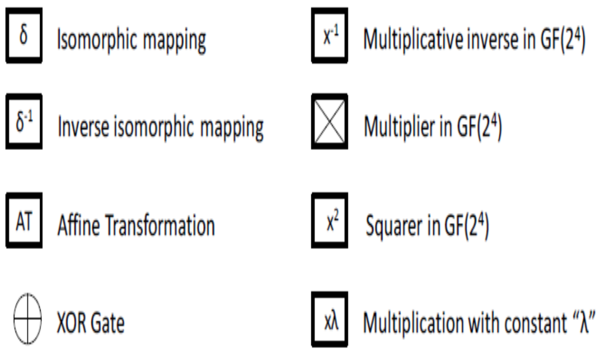


Fig -2: Legends for the building blocks within the multiplicative inversion module.

2.1 Multiplicative Inversion

Each of these operators can be transformed into individual blocks when constructing the circuit for computing the multiplicative inverse. From this simplified equation, the multiplicative inverse circuit GF(28) can be produced as shown in Figure 2.

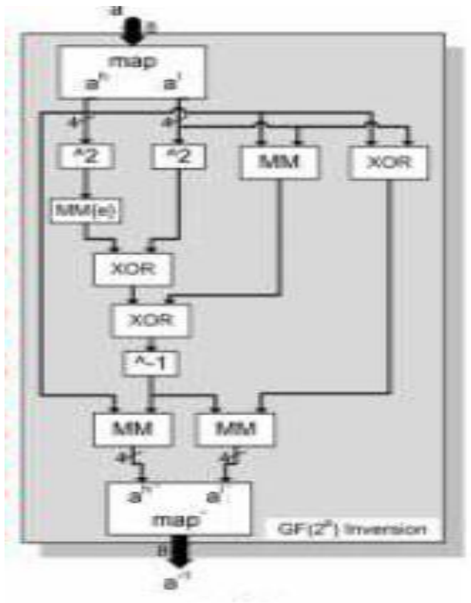


Fig -3: Multiplicative Inversion.

2.2 GF(2^4) Multiplier

Sub Bytes is a nonlinear transformation that uses 16 byte substitution tables (S-Boxes). An S-Box is the multiplicative inverse of a Galois field GF(2^4) followed by an affine transformation. Although two Galois Fields of the same order are isomorphic, the complexity of the field operations may heavily depend on the representations of the field elements. Composite field arithmetic can be employed to reduce the hardware complexity.Three multipliers in GF(2^4) are required as a part of finding the multiplicative inverse in GF(2^8). Fig. shows the GF(2^4) multiplier circuit. As can be seen from the figure the GF(2^4) multipliers consist of 3 GF(2^2) multipliers with 4 XOR Gates and with constant multiplier θ. This constant multiplier which has 2 bits input extracts the lower bit output as the higher bit input, while the higher output bit will be the result of XOR operation between the 2 input bits. Full derivation of this multiplier circuit can be found.

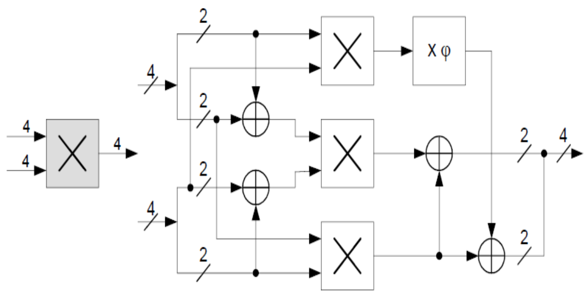


Fig -4: Hardware implementation of multiplication in GF(2^4).

2.3 GF (2^2) Multiplier

While each finite field is itself not infinite, there are infinitely many different finite fields; their number of elements (which is also called cardinality) is necessarily of the form p ^{n} where p is a prime number and n is a positive integer.

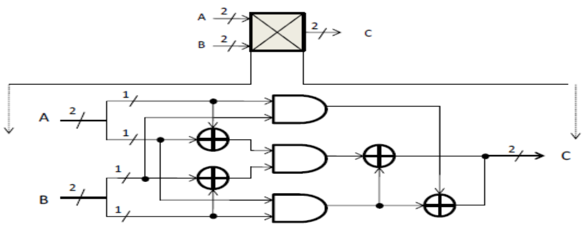


Fig -5: Hardware implementation of multiplication in GF(2^2).

2.4 Constant Multiplier (Xϕ)

Constant Multiplier represented as ^{Φ}.The following figure shows Hardware implementation of multiplication with constant ^{Φ}.

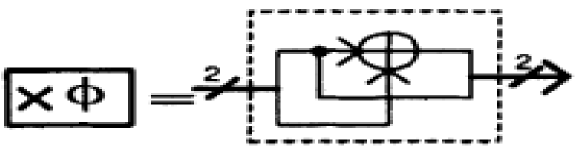


Fig -6: Hardware implementation of multiplication with constant ^{Φ}.

2.5 Isomorphic Mapping

The multiplicative inverse computation will be done by decomposing the more complex GF(28) to lower order fields of GF(21), GF(22) and GF((22)2). In order to accomplish the above, the following irreducible polynomials are used.

$$
\begin{array}{ll}
GF(2^2) \rightarrow GF(2) & : x^2 + x + 1 \\
GF((2^2)^2) \rightarrow GF(2^2) & : x^2 + x + \varphi \\
GF(((2^2)^2)^2) \rightarrow GF((2^2)^2) & : x^2 + x + \lambda
\end{array}
$$

where $\varphi = \{10\}_2$ and $\lambda = \{1100\}_2$.

Computation of the multiplicative inverse in composite fields cannot be directly applied to an element which is based on GF(28). That element has to be mapped to its composite field representation via an isomorphic function, δ.

$$\delta \times q = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} \qquad \delta \times q = \begin{pmatrix} q_7 \oplus q_5 \\ q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_6 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_6 \oplus q_4 \oplus q_1 \\ q_6 \oplus q_1 \oplus q_0 \end{pmatrix}$$

Fig -7: Isomorphic Mapping.

2.6 GF(2^4) Squarer

It consists of bitwise xor operation. A bitwise operation operates on one or more bit atterns or binary numerals at the level of their individual bits. It is a fast, primitive action directly supported by the processor, and is used to manipulate values for comparisons and calculations. On simple low-cost processors, typically, bitwise operations are substantially faster than division, several times faster than multiplication, and sometimes significantly faster than addition. While modern high-performance processors usually perform addition and multiplication as fast as bitwise operations, the latter may still be optimal for overall power/performance due to lower resource use.

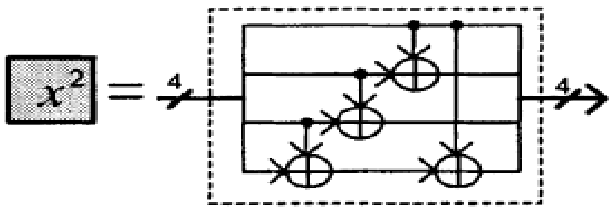


Fig -8: Hardware diagram for Squarer in GF(2^4).

2.7 Multiplication With Constant λ in GF (2^4):

Multiplication of two elements of GF(24) can be done by simply multiplying the two polynomials. However, the product would be a polynomial with a degree possibly higher than 3.

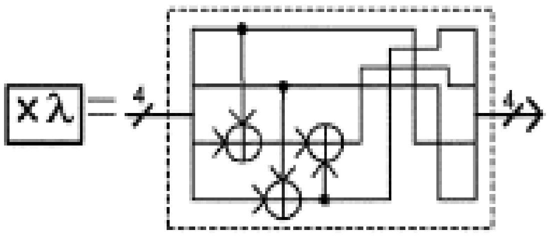


Fig-9: Hardware implementation of multiplication with constant λ.

2.8 Addition In GF(2^4)

Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation.

2.9 Square –Multiply Approach

The inversion in GF(2^4) can be implemented by different approaches. Taking x € GF(2^4),x^{-1} = x^{14} = ((x^2)^2)^2 . (x^2)^2.x^2.Hence, the inversion can be implemented. By repeat squaring and multiplying, This approach is illustrated in Fig.10.

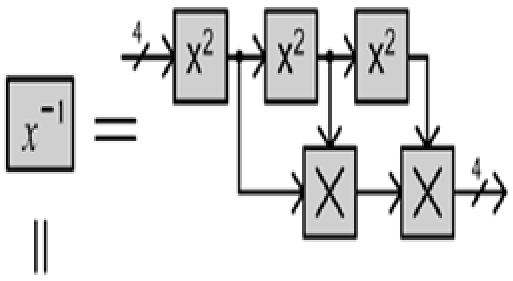
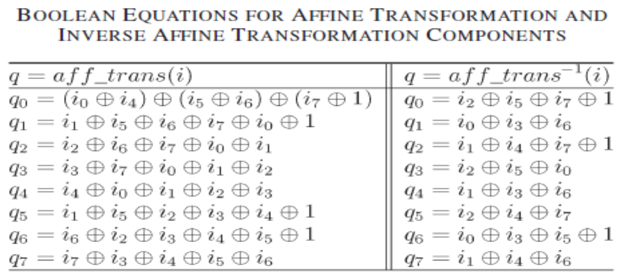


Fig-10: Hardware implementation of Square multiplication in GF(2^4).

2.10 Affine Transform

The Affine Transformation can be represented in matrix form and it is shown below.



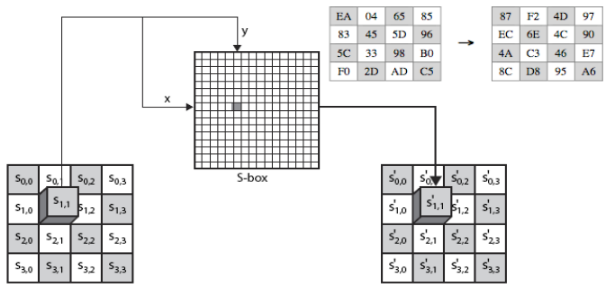
BOOLEAN EQUATIONS FOR AFFINE TRANSFORMATION AND INVERSE AFFINE TRANSFORMATION COMPONENTS

$q = aff\_trans(i)$	$q = aff\_trans^{-1}(i)$
$q_0 = (i_0 \oplus i_4) \oplus (i_5 \oplus i_6) \oplus (i_7 \oplus 1)$	$q_0 = i_2 \oplus i_5 \oplus i_7 \oplus 1$
$q_1 = i_1 \oplus i_5 \oplus i_6 \oplus i_7 \oplus i_0 \oplus 1$	$q_1 = i_0 \oplus i_3 \oplus i_6$
$q_2 = i_2 \oplus i_6 \oplus i_7 \oplus i_0 \oplus i_1$	$q_2 = i_1 \oplus i_4 \oplus i_7 \oplus 1$
$q_3 = i_3 \oplus i_7 \oplus i_0 \oplus i_1 \oplus i_2$	$q_3 = i_2 \oplus i_5 \oplus i_0$
$q_4 = i_4 \oplus i_0 \oplus i_1 \oplus i_2 \oplus i_3$	$q_4 = i_1 \oplus i_3 \oplus i_6$
$q_5 = i_1 \oplus i_5 \oplus i_2 \oplus i_3 \oplus i_4 \oplus 1$	$q_5 = i_2 \oplus i_4 \oplus i_7$
$q_6 = i_6 \oplus i_2 \oplus i_3 \oplus i_4 \oplus i_5 \oplus 1$	$q_6 = i_0 \oplus i_3 \oplus i_5 \oplus 1$
$q_7 = i_7 \oplus i_3 \oplus i_4 \oplus i_5 \oplus i_6$	$q_7 = i_1 \oplus i_4 \oplus i_6$

Fig -11: Affine Transformation.

2.11 SubBytes Transformation

The SubBytes transformation is a non-linear byte substitution, operating on each of the state bytes independently. The Sub-Bytes transformation is done using a once-pre calculated substitution table called S-box. The S-box is computed based on a multiplicative inverse in the finite field GF (2^8) and a bitwise affine transformation. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points, and also any opposite fixed points. The S-box table contains 256 numbers (from 0 to 255) and their corresponding resulting values. This approach has the significant advantage of performing the S-box computation in a single clock cycle, thus reducing the latency and avoids complexity of hardware implementation. The following Figure:12 shows Substitute Bytes Stage of the AES algorithm.


y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$
→

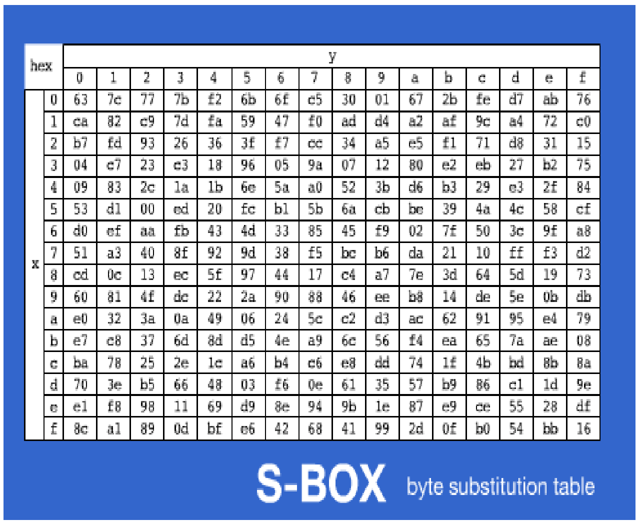
EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Figure.12: Substitute Bytes Stage of the AES algorithm.

2.12 S-Box Lookup Table

The S-Box transformations can be represented via a lookup table as follows. The input to the lookup tables is a byte B [7-0] where x and y denote it low and high nibbles x [3-0] = B [7-4], y [3-0] = B [3-0].The output byte is encoded in the table as a two digits number in hexadecimal notation. For example, S-Box lookup for the input 85 (x=8; y=5 in hexadecimal notation) yields 97 in hexadecimal notation.



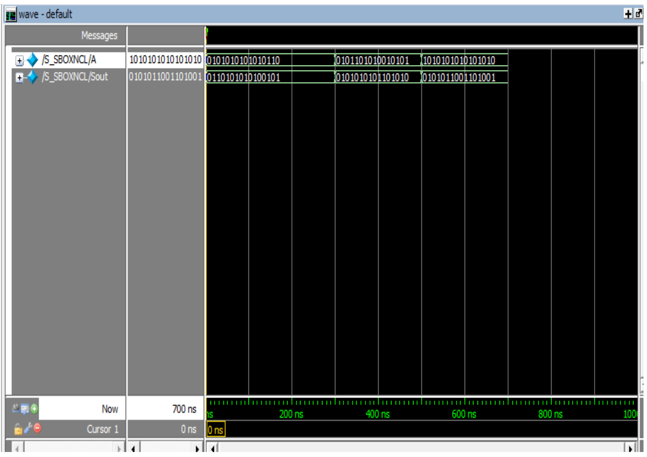
hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	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

S-BOX byte substitution table

Figure.13: S-BOX

2. SIMULATION RESULT

3.1 S-BOX



```
wave - default
Messages
/S_SBOXNCL/A     1010101010101010   0101010101010110   0101101010010101   1010101010101010
/S_SBOXNCL/Sout  0101011001101001   0110101010100101   0101010101101010   0101011001101001
Now      700 ns    ns   200 ns   400 ns   600 ns   800 ns   100
Cursor 1   0 ns    0 ns
```

Fig -14: Simulated report of NCL Substitution- Box.

Table III shows the simulation result for both the synchronous S-Box and the NCL S-Box. On the NCL S-Box output column, the results are shown as 16 bits, which are the extended dual-rail signals. For example, for input 01, the NCL S-Box output is 01 10 10 10 10 10 01 01, and this dual rail encoded data word is equivalent to 01111100 in binary, which is equal to the output of the conventional synchronous S-Box.

Table -3: S-BOX AND NCL S-BOX OUTPUT

S.N0	INPUT	S-BOX OUTPUT	NCL S-BOX
1	01	01111100	0110101010100101
2	38	00111000	0101101010010101
3	ff	11111111	1010101010101010

3. CONCLUSION

In this research we show how we can impletemented NCL based AES S-Box Technique successfully by using 'verilog' language. Communications performed in today's world require proper secured and safety communication from un-authorized user. Hence we can see that secured and safety communication is established through the use of cryptography. The asynchronous AES S-Box design is based on self-time logic referred to as Null Convention Logic, which supports beneficial properties for resisting SCA's and DPA: clock free, dual-rail signal, and monotonic transitions. These beneficial properties make it difficult for an attacker to decipher secret keys embedded within the cryptographic circuits. The future enhancement of this project is that, we can implement the null convention logic concept in advanced encryption standard cryptographic process, so that a highly secured and safety communication can be established. The proposed NCL based Substitution Box is designed by using Mentor Graphics EDA (Electronic Design Automation) tools.

REFERENCES

[1] NIST, "Advanced Encryption Standard (AES), FIPSPUBS 197, National Institute of Standards and Technology", NIST, Nov 2001.
[2] P. Kocher, J. Jaffe and B. Jun, "Introduction to differential power analysis and related attacks", Technical Report, Cryptography Research Inc., San Francisco, California, 1998.
[3] S. Moore, R. Anderson, P.Cunningham, R. Mullins and G. Taylor, "Improving smart card security using self-time circuits", Proceeding of Eighth International Symposium on Asynchronous Circuits and System, pp. 211-218, IEEE Computer Society, 2002.
[4] Akashi Satoh, Sumio Morioka, Kohji Takano and Seiji Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization.", Springer-Verlag Berlin Heidelberg, 2001.
[5] Vincent Rijmen, "Efficient Implementation of the Rijndael S-Box.", Katholieke Universiteit Leuven, Dept. ESAT. Belgium.
[6] W. Johannes, O. Elisabeth and L. Mario, "An ASIC Implementation of the AES S-Boxes", Topics in cryptology, CT-RSA 2002, LNCS, Vol. 2271, pp. 29-52, Jan 2002
[7] K. Fant and S. Brandt, "Null convention logic TM: a complete and consistent logic for asynchronous digital circuit synthesis," International Conference on Application Specific Systems, Architectures and Processors, pp. 261–273, 1996.
[8] Jun Wu, Yiyu Shi, and Minsu Choi, "Measurement and Evaluation of Power Analysis Attacks on Asynchronous S-Box" in instrumentation and measurement, vol. 61, no. 10, october 2012.
[9] S. Mangard, T. Popp and B. Gammel, "Side-Channel Leakage of Masked CMOS Gates", Topics in Cryptology-CT-RSA2005 pp. 351-365, Springer, 2005
[10] J.Wu, Y.-B. Kim, andM. Choi, "Low-power side-channel attack-resistant asynchronous s-box design for AES cryptosystems", in Proc. 20th Symp. Great Lakes Symp. VLSI, 2010, pp. 459–464.
[11] K. M. Fant and S. A. Brandt, "Null convention logic: A complete and consistent logic for asynchronous digital circuit synthesis," in Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors, p. 261, 1996.
[12] K. M. Fant and S. A. Brandt, "Null convention logic system," US Patent (US5305463), 04 1994.
[13] S. C. Smith, "Speedup of null convention digital circuits using null cycle reduc-tion," Journal of System Architecture, vol. 52, no. 7, pp. 411–422, 2006.
[14] K. M. Fant, S. A. Brandt, "Null convention logic. A complete and consistent logic for asynchronous digital circuit synthesis," in Proc. Of the International Conference on Application Specific Systems, Architectures, Chicago, 1996,pp.261–273.