Implementation of B.I.S.T Technique in an Aes for a Cryptocore

Sai Tarun Teja Surapaneni, Bindu Priya Makala and
K V Ratna Prabha

January 22, 2021

IMPLEMENTATION OF B.I.S.T TECHNIQUE IN AN AES FOR A CRYPTOCORE

ABSTRACT:

The main motive of this project is to design a crypto device with low complexity and high security by using **"ADVANCED ES"** Algorithm along with BIST technique. The selective application of technological and related procedural safeguards is an important responsibility of every Federal organization in providing adequate security to its electronic data systems and coming to BIST concept there are two main functions that must be performed on-chip in order to implement built-in self-test (BIST): test pattern generation and output response analysis. The most common BIST schemes are based on pseudorandom test pattern generation using linear feedback shift registers (LFSR'S) and output response compaction using signature analyzers. To accomplish high security for a system we are using the crypto devices technique in our project.

INTRODUCTION:

Most of the user now a day's using wireless communication for fast sending and receiving the mails in less time and in less cost. When this way of communication is going on, the unauthorized people who have the intension to know about our conversion will hack the information within that frequency. After hacking the information the hacker can know about what we are discussing. This leads to leakage of information. Nowadays, secure circuits are commonly used for applications such as e-banking, pay tv, cell phone... Because they hold personal data and must process secure operations, security requirements such as source/sink authentication, data integrity, confidentiality, or tamper resistance are maintained by means of several dedicated components. Confidentiality is ensured through cryptographic mechanisms generally implemented on co-processors.

PROPOSED MODEL:

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. Rijndael was designed to handle additional block sizes and key lengths; however they are not adopted in this standard. Throughout the remainder of this standard, the algorithm specified here in will be referred to as "the AES algorithm." The algorithm may be used with the three different key lengths indicated above, and therefore these different "flavours" may be referred to as "AES-128", "AES-192", and "AES-256".

This specification includes the following sections:

1. Definitions of terms, acronyms, and algorithm parameters, symbols, and functions.
2. Notation and conventions used in the algorithm specification, including the ordering and numbering of bits, bytes, and words.
3. Mathematical properties that is useful in understanding the algorithm.

4. Algorithm specification, covering the key expansion, encryption, and decryption routines.
5. Implementation issues, such as key length support, keying restrictions, and additional block/key/round sizes.

The standard concludes with several appendices that include step-by-step examples for Key. At the start of the Cipher, the input is copied to the State array using the conventions. After an initial Round Key addition, the State array is transformed by implementing a round function 10, 12, or 14 times (depending on the key length), with the final round differing slightly from the first Nr -1 rounds. The final State is then copied to the output.

The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine.

The Cipher is described in the pseudo code. The individual transformations - Sub Bytes (), Shift Rows (), Mix Columns (), and AddRoundKey () – process the State and are described in the following subsections…

It is very important to know that the cipher input bytes are mapped onto the state bytes in the order a0,0, a1,0, a2,0, a3,0, a0,1, a1,1, a2,1, a3,1 ... and the bytes of the cipher key are mapped onto the array in the order k0,0, k1,0, k2,0, k3,0, k0,1, k1,1, k2,1, k3,1 ... At the end of the cipher operation, the cipher output is extracted from the state by taking the state bytes in the same order. AES uses a variable number of rounds, which are fixed: A key of size 128 has 10 rounds. A key of size 192 has 12 rounds. A key of size 256 has 14 rounds.

During each round, the following operations are applied on the state:

1. Sub Bytes: every byte in the state is replaced by another one, using the Rijndael S-Box

2. Shift Row: every row in the 4x4 array is shifted a certain amount to the left

3. Mix Column: a linear transformation on the columns of the state.

4. AddRoundKey: each byte of the state is combined with a round key, which is a different key for each round and derived from the Rijndael key schedule.

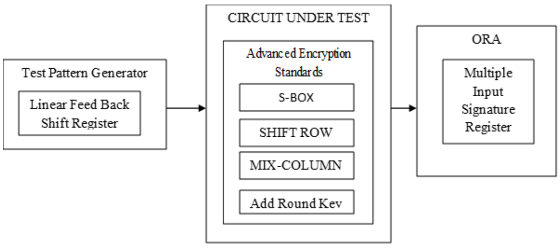

Test Pattern Generator
Linear Feed Back Shift Register
CIRCUIT UNDER TEST
Advanced Encryption Standards
S-BOX
SHIFT ROW
MIX-COLUMN
Add Round Key
ORA
Multiple Input Signature Register

Fig: Self-Test Technique for Crypto Devices

overall architecture of LBIST with AES. Starting block linear feedback shift register is used for Test pattern Generator. In computing, a linear-feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value.It is used to generate all type of test patterns for Circuit under Test. Here, in this concept 128 bit LFSR is used as test pattern generator. $2^{128} - 1$ patterns are generated by using above LFSR. Since, XOR gate is used to construct LFSR, all zeros combination is can't be generated.

Output Response Analyzer is last and vital device. Final output checking is done by this component. If any error occurred in whole process or not is checked by this ORA. ORA takes input from AES practical circuit and theoretical circuit, it compares both inputs using XOR gates, yields final output. 128 xor gates are used to compare produced outputs.

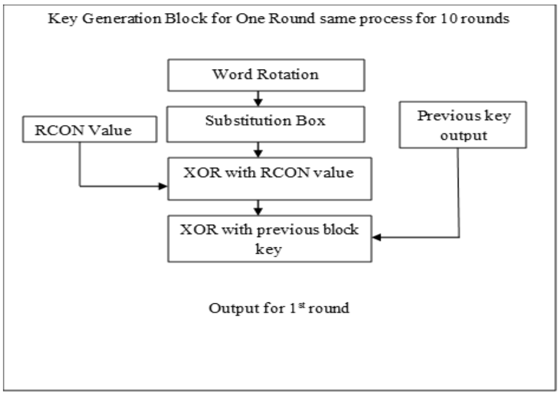


Fig : Key Generation Diagram

Symmetric-key algorithms use a single shared key; keeping data secret requires keeping this key secret. Public-key algorithms use a public key and a private key. The public key is made available to anyone (often by means of a digital certificate). A sender encrypts data with the public key; only the holder of the private key can decrypt this data. Since public-key algorithms tend to be much slower than symmetric-key algorithms, modern systems such as TLS and SSH use a combination of the two: one party receives the other's public key, and encrypts a small piece of data (either a symmetric key or some data used to generate it). The remainder of the conversation uses a (typically faster) symmetric-key algorithm for encryption. Computer cryptography uses integers for keys. In some cases keys are randomly generated using a random number generator (RNG) or pseudorandom number generator (PRNG). A PRNG is a computer algorithm that produces data that appears random under analysis.

PRNGs that use system entropy to seed data generally produce better results, since this makes the initial conditions of the PRNG much more difficult for an attacker to guess. In other situations, the key is derived deterministically using a passphrase and a key derivation function. The simplest method to read encrypted data is a brute force attack—simply attempting every number, up to the maximum length of the key. Therefore, it is important to use a sufficiently long key length; longer keys take exponentially longer to attack, rendering a brute force attack impractical. Currently, key lengths of 128 bits (for symmetric key algorithms) and 1024 bits (for public-key algorithms) are common.

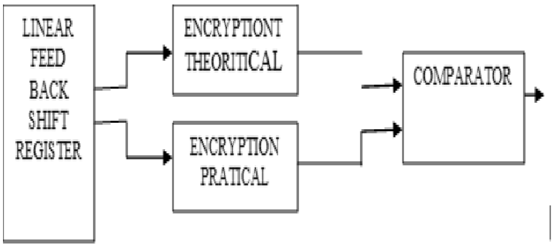


Fig : Self- Test for Encryption Side Diagram

A built in self test or built in test is a mechanism that permits a machine to test itself. We design BISTs to meet the requirements such as:

The main purpose of BIST is to reduce complexity of test/probe setup, by reducing the number of I/O signals that must be drives/Examined under tetser control, reduce the size. Both lead to reduce in hourly charges for automated test equipment(ATE) service. Similar to encryption, decryption block diagram looks similar only change is instead of encryption we use decryption.

LFSR PESUDORANDOM TEST GENERATION: To develop a battery of statistical tests to detect non randomness in binary m sequences constructed using random number generators and pseudorandom no generators utilized in cryptographic applications, To produce documentation and software implementation of these tests, and To provide guidance in the use and application of these tests. Pseudorandom- generate patterns that appear to be random but are in fact deterministic (repeatable).Linear Feedback Shift Register (LFSR) Weighted pseudo-random test generation Adaptive pseudo-random test generation

Algorithmic Test Generation: List primary inputs controlling location where a fault should be detected.

Determine primary input conditions to activate a fault and to sensitize the primary outputs such that the fault can be observed.

Pseudo-Random Test Generation

- Large set of patterns is generated by simple HW or SW pseudo-random generator
- The set is used to stimulate a system with fault simulator
- Fault coverage is analyzed and algorithmic approach is used to cove r remain faults.

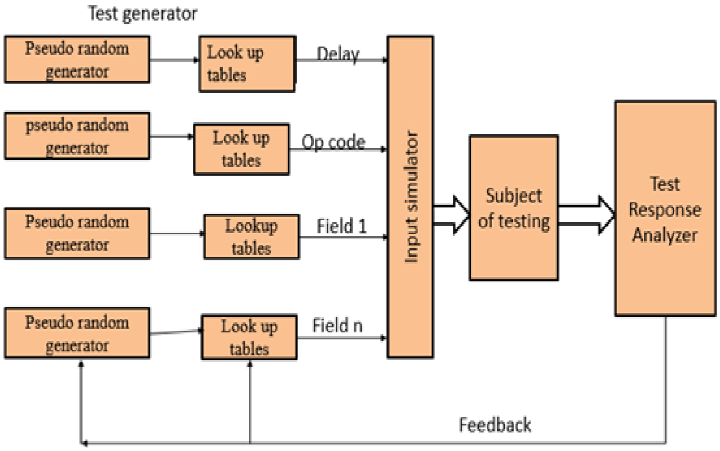


Fig : Complex Pseudo-Random test generator

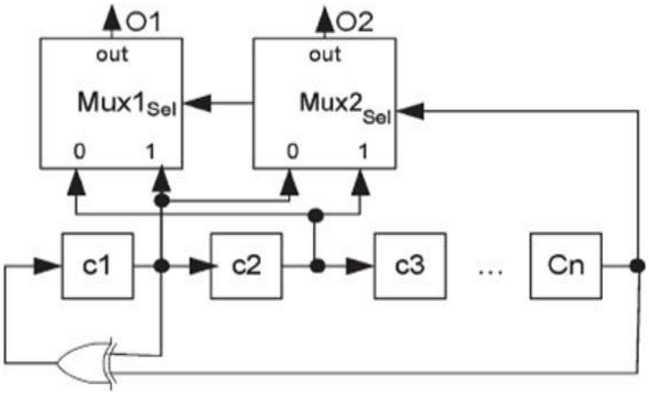
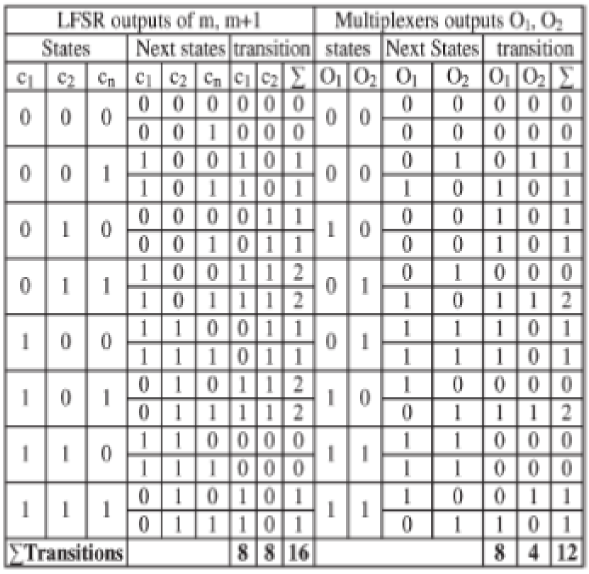


Fig: Bitswapping LFSR

LFSR with bit swapping technology. From this BIT SWAPPING technology we are going to reduce the peak power. By connecting multiplexers on the LFSR register as shown in above arrangement the number of transitions are decreased for that cell which are under bit swapping.

The below table shows the number of transitions in each register in LFSR without applying BIT SWAPPING technology, after applying bit swapping technology.



LFSR outputs of m, m+1									Multiplexers outputs O_1, O_2							
States			Next states			transition			states		Next States		transition			
c_1	c_2	c_n	c_1	c_2	c_n	c_1	c_2	$\sum$	O_1	O_2	O_1	O_2	O_1	O_2	$\sum$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			0	0	1	0	0	0			0	0	0	0	0
0	0	1	1	0	0	1	0	1	0	0	0	1	0	1	1
			1	0	1	1	0	1			1	0	1	0	1
0	1	0	0	0	0	0	1	1	1	0	0	0	1	0	1
			0	0	1	0	1	1			0	0	1	0	1
0	1	1	1	0	0	1	1	2	0	1	0	1	0	0	0
			1	0	1	1	1	2			1	0	1	1	2
1	0	0	1	1	0	0	1	1	0	1	1	1	1	0	1
			1	1	1	0	1	1			1	1	1	0	1
1	0	1	0	1	0	1	1	2	1	0	1	0	0	0	0
			0	1	1	1	1	2			0	1	1	1	2
1	1	0	1	1	0	0	0	0	1	1	1	1	0	0	0
			1	1	1	0	0	0			1	1	0	0	0
1	1	1	0	1	0	1	0	1	1	1	1	0	0	1	1
			0	1	1	1	0	1			0	1	1	0	1
$\sum$Transitions						8	8	16					8	4	12

Table: The Number of Transitions in Each Register in LFSR

From the above fig. if ctrl=1 then it perform encryption operation. So now it can perform the encryption operation i.e., Add round key, Shift Row, Substitution Byte and mixed mul operation. Already discussed inabove sections about these operations. After that we get the output as cyper text.

From the above fig. if ctrl=0 then it perform decryption operation. So now it can perform the decryption operation ie.,Inv Add round key, Inv Shift Row, Inv Substitution Byte and Inv mixed mul operations . Already discussed in above sections about these operations. After that we getthe output as plain text.

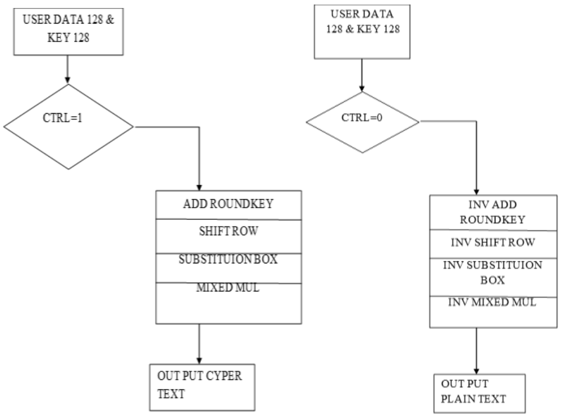
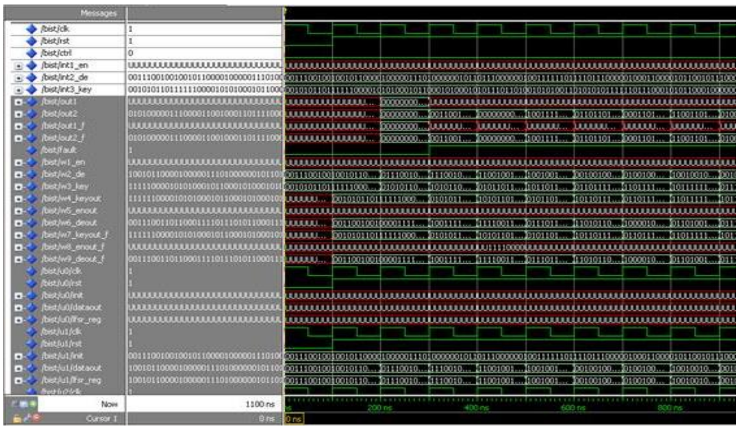


FIG: ASM CHARTS FOR ENCRYPTION AND DECRYPTION

RESULTS:



```
Messages
/bist/clk            1
/bist/rst            1
/bist/ctrl           0
/bist/int1_en        UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/int2_de        0011100100100101100001000001110100
/bist/int3_key       0010101101111110000101010001011000
/bist/out1           UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/out2           0101000001110000110010001101111000
/bist/out1_f         UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/out2_f         0101000001110000110010001101111000
/bist/fault          1
/bist/w1_en          UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/w2_de          1001011000010000011101000000101100
/bist/w3_key         1111100001010100010110001010001011
/bist/w4_keyout      1111110000101010001011000101000100
/bist/w5_enout       UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/w6_deout       0011100110110001111011101011000111
/bist/w7_keyout_f    1111110000101010001011000101000100
/bist/w8_enout_f     UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/w9_deout_f     0011100110110001111011101011000111
/bist/u0/clk         1
/bist/u0/rst         1
/bist/u0/init        UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/u0/dataout     UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/u0/lfsr_reg    UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
/bist/u1/clk         1
/bist/u1/rst         1
/bist/u1/init        0011100100100101100001000001110100
/bist/u1/dataout     1001011000010000011101000000101100
/bist/u1/lfsr_reg    1001011000010000011101000000101100
/bist/u2/clk         1
Now                  1100 ns
Cursor 1             0 ns
```

Fig : Final simulation report of our project

The above fig. is the final simulation report of the project. Which contains various signals that are used in this project.

CONCLUSION:-

In this project a solution is presented that consists in using an AES-based cryptographic core commonly embedded in secure system. Three addition modes are added to the current mission of the AES crypto core. One for pseudo- random test pattern generation & one for signature analysis. Efficiency of these three modes has been demonstrated. Extra cost in terms of area is very low compared to other techniques. Because only one AES core will be originally embedded in the system. This reduces the reduction of test cost will lead to the reduction of overall production cost & 100% security of data.

REFERENCES

[1] Sudhir Rao Rupanagudi, Varsha G. Bhat, Abhiram Srisai, M. Harshavardhan, S. Namitha, S. Durgaprasad, Y. Harshitha, K. R. Kavya, Feba Chellappan, B. A. Harshitha, V. Vathsala, M. H. Surekha, G. N. Vachana, Vasanthi Satyananda, "Optimized area and speed architectures for the mix column operation of the advanced encryption standard", Robotics Automation and Sciences (ICORAS) 2017 International Conference on, pp. 1-5, 2017.

[2] Takahiro Suzuki, Sang-Yuep Kim, Jun-ichi Kani, Ken-Ichi Suzuki and Akihiro Otaka ―Real- time Demonstration of PHY Processing on CPU for Programmable Optical Access Systems‖,
Cisco Vis. Netw. Index, San Jose, CA, USA, Cisco White Paper, Feb. 3, 2016

[3] Mayada E. Mohamed, Sharief F. Babiker―An Efficient Implementation of a Fully Combinational Pipelined S-Box on FPGA‖ 2017 International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT), Coimbatore, India, 2017, pp. 1-9.

[4]A. Hafsa, N. Alimi, A. Sghaier, M. Zeghid and M. Machhout, "A hardware-software co-designed AES-ECC cryptosystem," 2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET), Hammamet, 2017, pp. 50-54.

[5] D. Minoli, K. Sohraby and B. Occhiogrosso, "IoT Security (IoTSec) Mechanisms for e- Health and Ambient Assisted Living Applications," 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), Philadelphia, PA, 2017, pp. 13-18..

[6]A. Kumar and A. Agarwal, "Research issues related to cryptography algorithms and key generation for smart grid: A survey," 2016 7th India International Conference on Power Electronics (IICPE), Patiala, India, 2016, pp. 1-5.

[7]W. Nowakowski, P. Bojarczak and Z. ukasik, "Performance analysis of data security algorithms used in the railway traffic control systems," 2017 International Conference on Information and Digital Technologies (IDT), Zilina, 2017, pp. 281-287.

[8]B. Bhat, A. W. Ali and A. Gupta, "DES and AES performance evaluation," International Conference on Computing, Communication & Automation, Noida, 2015, pp. 887-890.

[9]B. Indrani and M. K. Veni, "An efficient algorithm for key generation in advance encryption standard using sudoku solving method," 2017 International Conference on Inventive Systems and Control (ICISC), Coimbatore, 2017, pp. 1-8.

[10]N. D. Vaidya, Y. A. Suryawanshi and M. Chavan, "Design for enhancing the performance of Advance Encryption Standard algorithm VHDL," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016, pp. 1-5.