ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2278-2566
Vol.01, Issue.03
July -2019
Pages: -307-313

EFFICIENT LATENCY AND POWER OPTIMIZED MODIFIED AES CRYPTOGRAPHY SYSTEM

1.MD PARWEZ ALAM, 2.G SARITHA

1. M.Tech, Dept. Of ECE, Nova College of Engineering and Technology, Ibrahimpatnam, A.P
2. Guide, Dept. Of ECE, Nova College of Engineering and Technology, Ibrahimpatnam, A.P

ABSTRACT: This project plays vital role in all type of communication applications. This project designs a novel low-transition linear feedback shift register (LFSR) that is based on some new observations about the output sequence of a conventional LFSR. Security of a hardware implementation can be compromised by a random fault or a deliberate attack. The traditional testing methods are good at detecting random faults, but they do not provide to secure all type of attacks. It requires a small set of deterministic tests to cover maximum percentage of single stuck-at faults. Thus, the test execution time is much shorter (at least two orders of magnitude). It has a higher resistance against stuck-at fault type of hardware Trojans. Further, this project can be extended to decrease power by using scan bit swapping LFSR. In this algorithm, all test patterns to circuit are generated using low power LFSR and, generated patterns are reordered, in such a way; power will be decreased while testing application. Latency reduction can be done by using scan chain reordering. Cell reordering plays vital role in transitions reduction to further improvement of timing constraint.

KEYWORDS: Linear feedback shit register, Advanced Encryption Standards, Scan chain reordering, Trojansns, stuck-at fault, Hardware optimization.

INTRODUCTION: THE fast development of Internet-of-Thing (IoT) devices enables the massive integration of technologies from sensing technology, communication technology, data processing, to cloud computing, and artificial intelligence. In this scenario, sensors in the perception layer collect data from the environment and do fast processing. Then, these data are transmitted through the network layers over the Internet to the cloud. In the cloud, data are further processed by different applications, for example, big data applications or data miningapplications to make decisions and/or to notify users, etc. However, IoT devices and data transmitted through multilayer networks may contain private data or secrete data; while the Internet environment exposes security issues such as personal privacy, cyber-attacks, and organized crimes. This recently raises the concerns about the security and privacy of the IoTs [1]–[3]. The solution to security and privacy problems is to include security features such as device identification, device/user authentication, and data encryption. These security functions are often based on the cryptographic algorithms, including public-key cryptography and symmetric cryptography, which occupy processing power and increase power and energy consumption. In contrast, IoT devices are supposed to be constrained low-cost devices with limited processing power, limited memory footprint, and even limited power/energy budget, for example, power-harvesting devices and batterybased devices. This leads to the importance of optimizing cryptographic algorithms in hardware for cost, throughput, and especially power and energy consumption. However, cost, throughput, and power/energy consumption are different features which are hard to achieve at the same time. In this paper, we chose to find a good tradeoff among them for advanced encryption standard (AES) [4], a widely-used block cipher for emerging IoT proposals, such as IEEE 802.15.4 [5], LoraWAN [6], Sigfox [7], and ZWave [8]. We also made comparison with an extreme lightweight data encryption algorithm PRESENT [9], a candidate for highly constrained devices. PRESENT is a hardware-oriented block cipher with reduced security level but it has small area footprint and very lowpower consumption. However, to the best of our knowledge, lightweight block ciphers, such as PRESENT, are not yet adopted to any IoT proposals. From its standardization in 2001 by the U.S. National Institute of Standards and Technology (NIST) to replace data encryption standard, AES has been studied by researchers in terms of security, performance, and hardware/software implementations. In terms of security, different IoT applications may require different security levels with

different power/energy budgets and different throughputs. At the algorithmic level, security level depends on the design of the algorithm and the length of the key. AES supports multiple security levels by providing three different key sizes. AES is proven to support long-term and very long-term security. Because of its popularity and proved security, AES is widely used in data encryption, security protocols, and secure applications. The optimization for AES in hardware is not only beneficial to IoT applications but also to other applications, which have the same constraints. In terms of implementation and performance, AES is designed to benefit from software optimization in modern computing systems. However, AES implementation in software not only introduces delay to data processing and transmission, but also increases the power and energy consumption. This is the main limitation of AES to constrained devices.

ADVANCED ENCRYPTION STANDARD (AES):-

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) and the Computer Security Act of 1987 (Public Law 100-235).

The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be 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; decrypting the cipher text converts the data back into its original form, 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.

This standard specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 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 -

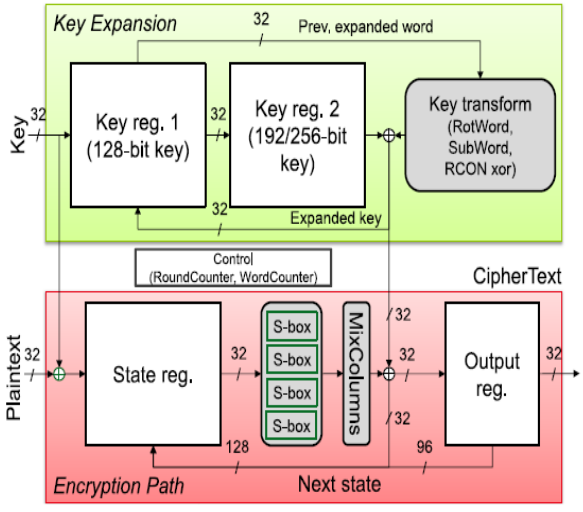
Sub Bytes (), Shift Rows (), Mix Columns (), and AddRoundKey () – process the State and are described in the following subsections.

All Nr rounds are identical with the exception of the final round, which does

Not include the Mix Columns () transformation.

A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

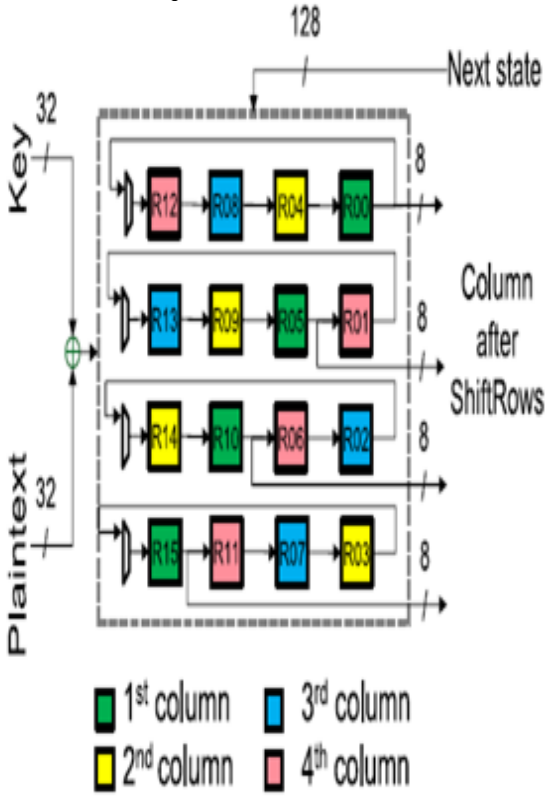
MODIFIED AES ARCHITECTURE:



Our proposed AES architecture.

To reduce area and power consumption in the datapath, we minimized the number of flip-flops and control logics in the datapath by using shift registers with a special organization. Shift registers help simplify loading data and loading key steps. The 32-b of both plaintext and key are loaded at the same time into the state register and the key register by using shift operations. By minimizing the number of flip-flops, we also reduced the number of clock buffers and the power consumption of the clock tree because clock buffers in the clock tree consume a large amount of power. A further optimization is to select S-boxes with minimal power dissipation. Fig. 3 shows the organization of our proposed state register. The state register is organized so that after loading the input data and the input key, the encryption is done by shifting the data 32 b in each clock cycle. The state register consists of sixteen 8-b registers (forming a "state matrix") which are further divided into four 4-stage shift registers. AES standard specifies that ShiftRow is a permutation operation on the rows of the state matrix, while MixColum is an operation on the columns. However, in our design, based on ShiftRow specification, we completely eliminated ShiftRows by selecting the diagonal of the state matrix (from lower-left corner to upper-right corner). The output of the state register after each shift operation is one column of the state matrix after ShiftRow. This reduces the control logics for the state register, and completely removes the logic for ShiftRow steps. In our datapath, in contrast with 8-b architectures, MixColum is designed as pure combinational logics to reduce the number of flip-flops. Thanks to this structure, the state register's contents will be updated by next state data which are the contents of the output register concatenated with four last bytes of the round operation every four cycles (or after each round finishes) as described in Fig. 4. Consequently, we saved a 32-b register because we need to store only 3 × 4-B temporary data from the encryption path in the output register, while the last 32-b data are written back directly into the state register. The output register is a simple 4 × 3-stage shift register to save area and power.

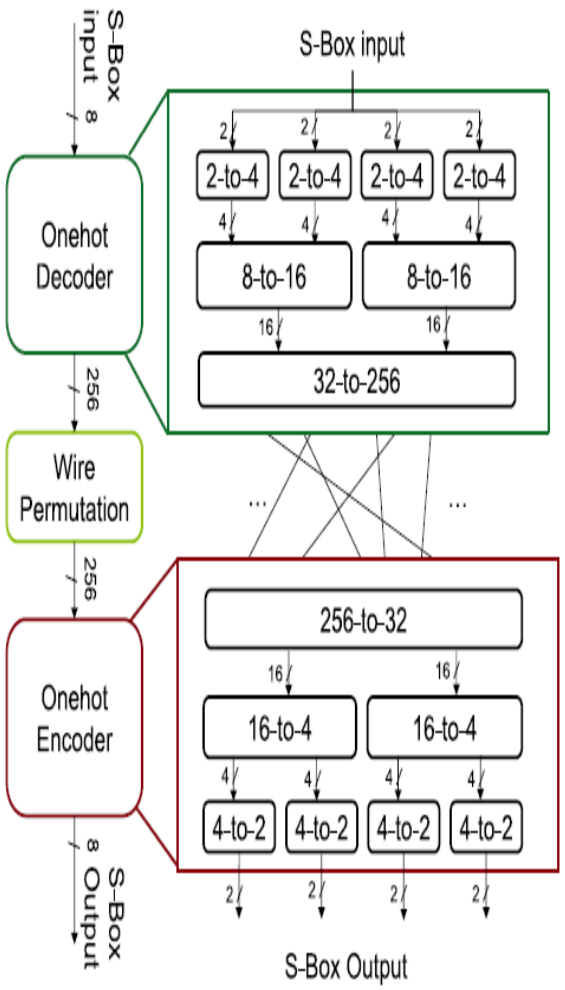
In between the state register and the output register, there are four S-boxes followed by the MixColums to enable processing 4 B in each clock cycle. The temporary results are stored in the output register. When the encryption finished, the results are written out from the output register. In the 128-b key configuration, AES encryption module needs ten rounds, which leads to 40 cycles to finish the encryption for a 128-b block of data. The total number of cycles to encrypt a block in our architecture is 44 cycles. For other key configurations, our architecture needs 52 and 60 cycles to encrypt a data block for 192- and 256-b key modes, respectively. Clock gating technique is applied on the state register and the output register separately to save the dynamic power consumption. For example, in data loading state, the clock to the output register is disabled to save power because there are no valid data to the output register. Furthermore, when in the inactive state, the output of these registers is not changed, which means that there is no activity in the encryption path. The power estimation results show that even in the highest throughput mode (44 cycles/encryption for 128-b key mode) the applied clock gating technique can save more than 13% of power.



Certainly, with smaller throughput the clock gating technique can even save much more power consumption.

SUBSTITUTION BOX:

The S-box has a big impact on area and power consumption of the AES design. In our architecture, we chose S-box implementation for the lowest power consumption. S-boxes may occupy up to 60% of the total cell area, while they consume about 10%–20% of the total power consumption. The smallest implementation of S-boxes until now is from Canright [18]. Canright S-box demonstrates optimized area (292 gates/S-box) but needs more power/energy consumption

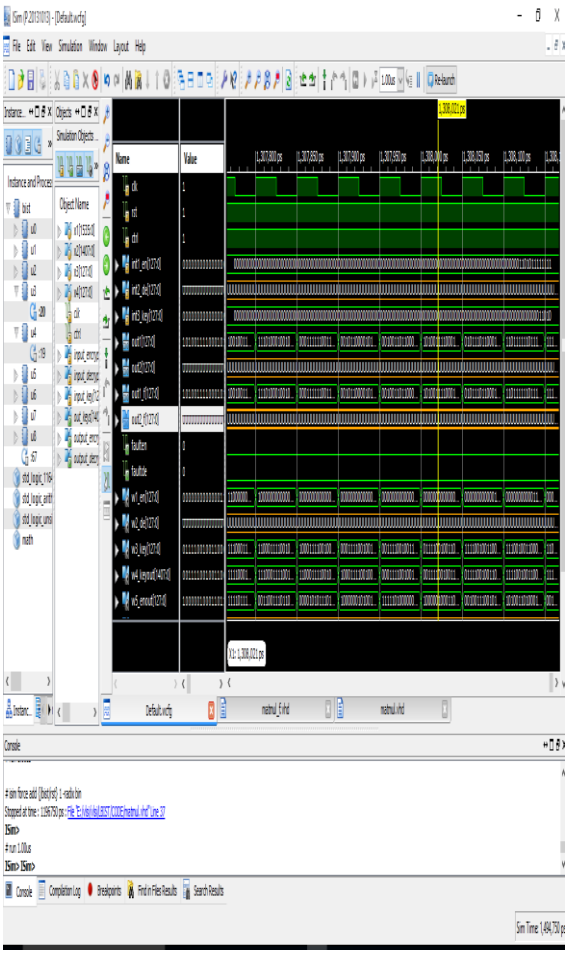


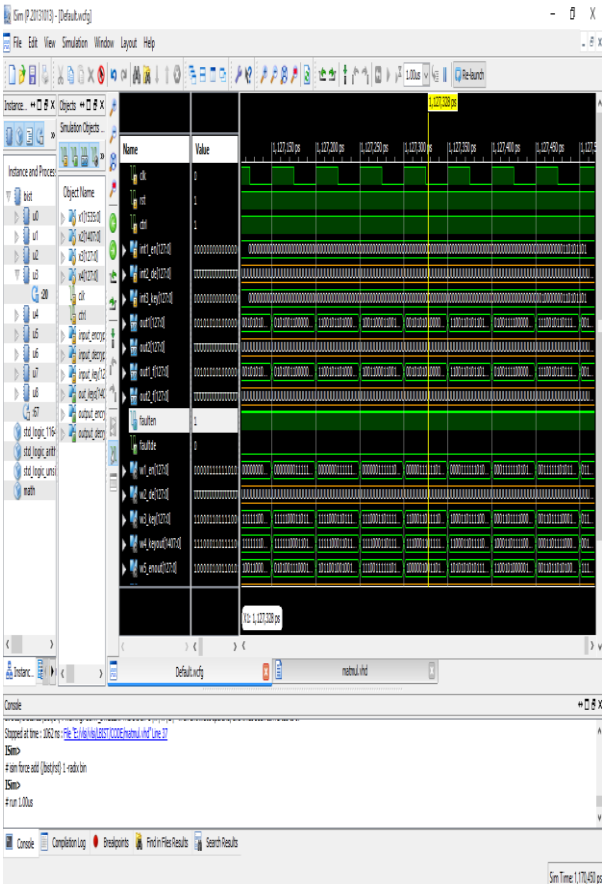
because it creates more activities especially in architectures with eight S-boxes. The most popular and straightforward S-box implementation is the LUT-based S-box. LUT-based S-box is bigger in terms of area (434 gates/S-box) but smaller in power/energy consumption than Canright S-box. The most efficient S-box in terms of power consumption is DSE S-box; however, it occupies a larger area. DSE S-box can be further optimized for power consumption using the structure proposed in [20] and described in Fig. 5. The idea is to use an onehot decoder to convert S-box inputs into onehot representation. The nonlinear operations are done by using wire permutation as in lightweight cryptography algorithms. After that, the S-box output in onehot encoding is converted back into theoriginal field. DSE S-Box can reduce the power consumption because it minimizes the activity inside the S-box circuit. After decoding state, only one signal changes its value to go to the encoding state. Most of the area lost is because of the size of encoder and decoder circuits. This optimization can leads to 10% power reduction to the whole design. Our synthesized DSE S-box has the size of 466 GEs/S-box that is 7% increase in size in comparison with LUT-based S-Box or 1.6 times the size of the smallest S-boxes. The S-boxes in our design consume only 10% of the total power consumption.

MODIFIED AES S-BOX GENERATION:

Our modified AES S-Box generation process follows the construction procedure of the original AES. The whole process differs only in the selection of the irreducible polynomial and specially designated byte. 1) Multiplicative Inverse Table: In the Rijndael AES, all the arithmetic operations are performed over the Galois Field (28). In our work, the Galois Field (24) is considered. The number of irreducible polynomials of degree 4 over GF(2) are x4 + x + 1, x4 + x3 + x2 + x + 1 and x4 + x3 + 1. All the generated values of the multiplicative inverse table and substitution box depend on the selection of irreducible polynomial. For our experiment purpose, we choose x4+x+1as our irreducible polynomial but we can select any of the irreducible polynomials which are mentioned above.

RESULT:





CONCLUSION:-

Crypto may be seen as a continuous struggle between cryptographers & cryptanalysts. Attacks on cryptography have an equally long history. The security of cryptographic modules for providing a practical degree of protection against white-box (total access) attacks should be examined in a totally un-trusted execution environment.

So many developers design so many devices to protect the data very powerful when it is done right, but it is not a panacea. But by using this crypto devices technique we are providing secure scan architecture can easily be integrated into the scan-based DFT design flow as the synthesis register can be specified to the corresponding bit of the secret key. The secure control circuit & multiplexers between the MKR & secret key can be inserted.

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. S. Reddy, "Easily testable realizations for logic functions," IEEE Transactions on Computers, vol. 21, no. 11, pp. 1183–1188, 1972.
2. S. Golomb, Shift Register Sequences. Aegean Park Press, 1982.
3. R. K. Brayton, C. McMullen, G. Hatchel, and A. Sangiovanni-Vincentelli, Logic Minimization Algorithms For VLSI Synthesis. Kluwer Academic Publishers, 1984.
4. E. McCluskey, "Built-in self-test techniques," IEEE Design and Test of Computers, v Vol. 2, pp. 21–28, 1985.
5. D. H. Green, "Families of Reed-Muller canonical forms," International Journal of Electronics, vol. 70, pp. 259–280, 1991.
6. M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design. Jon Willey and Sons, New Jersey, 1994
7. H.-J. Wunderlich, "BIST for systems-on-a-chip," Integration, the VLSI Journal, vol. 26, no. 1-2, pp. 55 – 78, 1998.
8. M.G. Kuhn, R.J. Anderson. Soft tempest: hidden data transmission using electromagnetic emanations. Information Hiding 1998,LNCS 1525,pp.124-142,1998.