ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Design and Implementation of AES Algorithm Using FPGA

Ankita Nampalliwar[1]
Department of Electronics & telecommunication Engineering
G. H. Raisoni Institute of Engineering and Technology for Women
Nagpur - India

Sheeja Suresh[2]
Department of Electronics & telecommunication Engineering
G. H. Raisoni Institute of Engineering and Technology for Women
Nagpur - India

Abstract: ***AES (Advanced Encryption Standard) is a specification published by the American National Institute of Standards and Technology in 2001, as FIPS 197. AES describes a symmetric-key algorithm which uses the same key for both encrypting and decrypting the data. The block size is of 128 bits and the key size can be of 128, 192, or 256 bits. AES operates on a 4×4 matrix of bytes, known as the state. Some rounds of transformation converts the plaintext into the final cipher-text. Six plus the key size divided by 32 are the number of rounds required i.e. one round reads the state into four 4-byte variables, transforms the variables, xor's them by a 16-byte round key when targeting a variable-length plaintext. The plaintext has to be partitioned into separate cipher blocks and after that only it can be encrypted under some mode of operation, generally by using randomization which is based on an additional initialization vector.***

The cipher feedback (CFB) mode, output feedback (OFB) mode are specified in FIPS 81. The counter (CTR) mode is specified by NIST .The advantage of these modes is only using encryption algorithm for both encryption and decryption which may reduces the AES hardware price by 50% (no need of decryption hardware).

Keywords: AES, Encryption, Decryption, Pipelining, Verilog HDL.

I. INTRODUCTION

The number of individuals and organizations using wide computer networks for personal and professional activities has recently increased a lot. A cryptographic algorithm is an essential part in network security. A well-known crypto-graphic algorithm is the Data Encryption Standard (DES) which has been widely adopted in security products. However, serious considerations arise for long-term security because of the relatively short key word length of only 56 bits and from the highly successful cryptanalysis attacks. In November 2001, the National Institute of Standards and Technology (NIST) of the United States chose the AES algorithm as the suitable Advanced Encryption Standard (AES) to replace the DES algorithm. Since then, many hardware implementations have been proposed in literature some of them use field programmable gate arrays (FPGA) and some use application-specific integrated circuits (ASIC). The advantages of a software implementation include ease of use, ease of upgrade and portability. There are some limitations of software implementation as it offers only limited physical security, especially with respect to key storage. Conversely, cryptographic algorithms (and their associated keys) implemented in hardware are more physically secure since they cannot easily be read or modified by an outside attacker. The downside of traditional (ASIC) hardware implementations is the lack of flexibility with respect to algorithm and parameter switching. Reconfigurable hardware devices such as FPGAs are a promising alternative for the implementation of block ciphers. FPGAs are hardware devices whose function are not fixed and can be programmed in-system. In this paper, we present an implementation of the AES block cipher with Virtex II Pro FPGA using 0.13 mm and 90 nm process technologies. We have exploited the temporal parallelism available in the AES algorithm.

The chip designed contains the ten units. Each unit can execute one round of the algorithm. Ten rounds of the algorithm are executed in parallel in a chip using external pipelined design. Furthermore, using internal pipelining and key exchange pipelining will help to achieve output in minimum time period.

II. THE AES ALGORITHM AND PREVIOUS WORK

A. AES ALGORITHM.

The AES algorithm is a symmetric block cipher that processes data blocks of 128 bits using a cipher key of 128, 192, or 256 bits length. Here each data block consists of a 4!4 array of bytes known as the state, on which the basic operations of the AES algorithm are performed. Fig. 1 shows the AES encryption and decryption procedures.

The encryption procedure is as follows:-

- After an initial round key addition, a round function consisting of four different transformations—byte-sub, shift-row, mix-column, and add-round-key—is applied to the data block in the encryption procedure.
- The round function is performed 10, 12, or 14 times, depending on the key length.
- The mix-column operation is not applied to the last round.
- The byte-sub operation is a nonlinear byte substitution that operates independently on each byte of the state using a substitution table (S-Box).
- The shift-row operation is a circular shifting on the rows of the state with different numbers of bytes (offsets).
- The mix-column operation mixes the bytes in each column by the multiplication of the state with a fixed polynomial modulo x4C1.
- Add-round-key operation is an XOR that adds a round key to the state in each iteration, where the round keys are generated during the key expansion phase.
- The byte-sub transformation (S-Box operation), which consists of a multiplicative inverse over GF(28) and an affine transform, is the most critical part of the AES algorithm in terms of computational complexity.
- However, the S-Box operation is required for both encryption and key expansion.
- Conventionally, the coefficients of the S-Box and inverse S-Box are stored in the lookup tables, or a hard-wired multiplicative inverter over GF (28) can be used, together with an affine transformation circuit.
- The decryption procedure of the AES is basically the inverse of each transformation.
- However, the standard decryption procedure is not identical to the encryption procedure.
- That is, the sequence of transformations for decryption is different from that for encryption though the form of the key schedules for encryption and decryption the same.
- There is, however, an equivalent version of the decryption procedure that has the same structure as the encryption procedure.
- The equivalent version has the same sequence of transformations as the encryption procedure (with transformations replaced by their inverses).

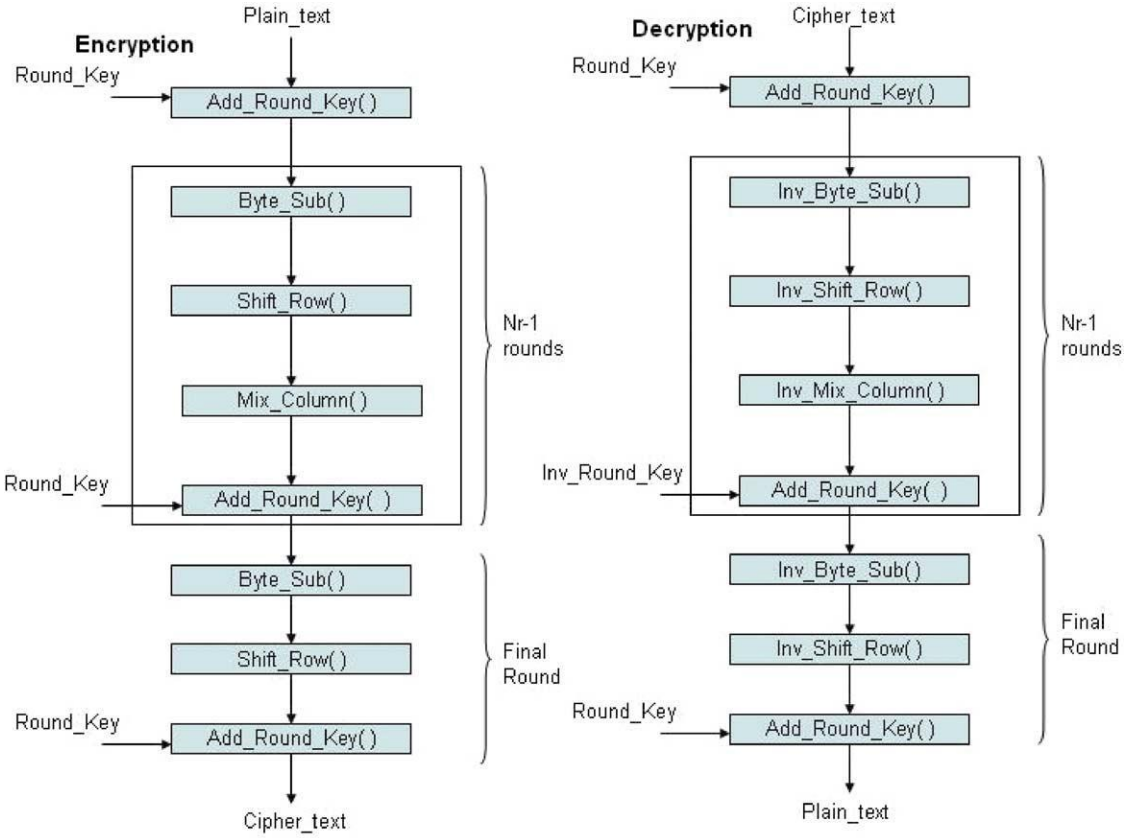

Fig 1. The AES algorithm (equivalent version) (Nr: 10, 12, or 14 depending on key length).

- To achieve this equivalence, a change of key schedule is needed. In addition, two separate changes are needed to bring decryption structure.
- The standard decryption round has the structure inv-shift-row, inv-byte-sub, add-round-key, and inv-mix-column.
- Thus, the first two stages of the decryption round need to be interchanged, and the second two stages of the decryption round need to be interchanged.
- The equivalent version of the decryption procedure is also shown in Fig. 1.

B. PREVIOUS WORK.

There exist many presentations of hardware implementations of AES algorithms in literature. Some of them will be briefly introduced here considering throughput. In 2001, Elbirt et al compared five candidate algorithms (including AES algorithm) for AES block cipher using FPGA implementations. Here, the throughputs of AES algorithm were in 187.8 Mbps w1.94 Gbps. In 2003, many implementations are shown in literature. Verbauwhede et al presented an ASIC implementation under the through-put of 2.29 Gbps. Su et al reduced hardware overhead of the S-Box by 64% and the throughput of their pipelined implementation using ASIC was 2.38 Gbps. McLoone and McCanny utilized look-up tables to implement the entire AES round function under the throughput of 12 Gbps using FPGAs. In 2004, Hodjat and Verbauwhede's FPGA implementation showed a high throughput of 21.54 Gbps using a fully pipelined approach with inner-round pipelining and outer-round pipelining.

III. THE AES IMPLEMENTATION USING A FAULT PIPELINED DESIGN

A. ENCRYPTION DATAPATH – PIPELINE DESIGN.

The goal of this implementation is to achieve the highest possible throughput. We have used the bottom-up design approach, implementing the elementary operations first before designing the final data path. A block based top-level implementation of the design for encryption is shown in Fig. 2. Round_1 through Round_10 represent the individual rounds in

the AES-128 encryption. The pipelining between each of the rounds will achieve a high performance encryption implementation. Although implementing an iterative pipelining based approach is one option, for clarity and simplicity, we have used a fully expanded implementation for all ten rounds. The data generated in each individual round is used as the input for the next round. Exploiting the loop, level parallelism which the AES offers, we found that, a simple ten stage pipelining of the top level with pipelined at the lower level blocks of the hierarchy is all that is needed to achieve high throughput. This is one of the easiest methods where high performance can be achieved in a very minimal amount of time thus reducing the overall design implementation cycle. In fig 2 , there is a pipeline stage between each round, i.e. the design is fully pipelined. However, our internal (inside each round) pipeline design is different .In each round, our design has three pipeline stages, one immediately after byte-sub operation, one just after shift-row operation, and the last just before data output. In each round, the design have four or seven pipeline stages, one after a byte-sub operation and three or six in a byte-sub operation. In addition, our design has one pipeline stage (before XOR operation between Round Key Block) in key generation blocks Internally, the key expansion block renders itself as a pipelined implementation between each of the key creations from Key1 through Key10. This is automatically realized by the parallel nature of the design. These additional pipelines make it possible for our implementation to obtain a higher throughput.

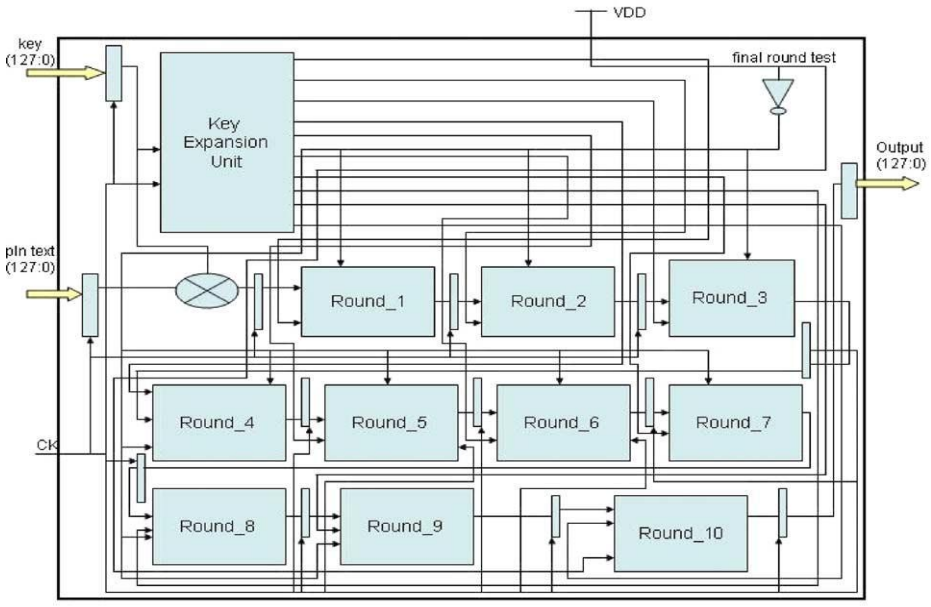

Fig 2. A pipelined AES and AES-128 encryption implementation.

B. ROUND KEY GENERATION.

For our implementation, we have chosen to use a hierarchical simultaneous key generation methodology. This is similar in approach to the fly key generation method. However, there is internal sub-pipelining for each of the sub-stages of the key creation. This would create the key for a single round. The XST FPGA synthesis environment will be able to automatically recognize constant logic and assign constant logic '0' or '1' to the appropriate 'Tie' cells from the library. The output is the key for the next round. We have implemented the round key generation as a simple state table substitution of the 32 bits of the input key and thereby implementing XOR operations for producing the expanded 128 bits round key. Using internal pipelining would greatly reduce the minimum clock period needed to assure the correct functionality of round key generation. Hence, implementing an extremely fast block in this preliminary stage of the design itself is very much possible. Maximum post synthesis clock frequency of 478.9 MHz has been for key generation, with this implementation. Placing the pipeline registers and loading the data using input/output registers (IR/OR) is the key to achieving high performance. IR is an input register used to load the input data. PR is the pipeline register used with intermediate data processing and OR is the output register. It can be inferred that it takes three clock cycles for the output to appear from key_in to key_out. What this would mean is that, for each of the ten rounds, to produce the key in each successive round would require at least three clock cycles. The Round Keys Block

simply uses the Round Key Block described previously to create the round keys for all the individual rounds. Hence, for 128 bits data/key encryption, it uses ten instances of the Round Key Block mentioned above to create all the ten round keys. Inside this block, actually each of the keys from Key1 to Key10 is created using the key expansion algorithm. In this implementation, each round key is created by instantiating the Round Key Block. Any decent parallel hardware architecture offers naturally high performance. Exploiting this concept, we have used internal pipelining within each of the round key creation stages. The use of balanced internal pipelining in between stages of a parallel architecture helps in reducing the flip-flop to flip-flop clock delay. As a result, it maximizes the performance of a design while guaranteeing minimum clock speed.

IV. CONCLUSION

In this paper we presented a hardware implementation increasing throughput for AES encryption algorithm. By using an efficient inter-round and intra-round pipeline design, we get the 20 clock cycles in 1 sec and hence we achieved a throughput much higher than any other implementations reported in the literature.

References

1. National Institute of Standards and Technology (US), Advanced Encryption Standard, http://csrc.nist.gov/publication/drafts/dfips- AES.pdf.
2. C.P. Su, T.F. Lin, C.T. Huang, C.W. Wu, A high-throughput low-cost AES processor, IEEE Commun. Mag. 42 (12) (2003) 86–91.
3. M. McLoone, J.V. McCanny, AES FPGA implementations utilizing look-up tables, J. VLSI Signal Process. Syst. 34 (3) (2003) 261–275.
4. F.X. Standaert, G. Rouvroy, J.J. Quisquater, J.D. Legat, Efficient implementation of AES encryption in reconfigurable hardware: improvements and design tradeoffs, in: CHES 2003, LNCS 2779, 334–350.
5. G.P. Saggese, A. Mazzeo, N. Mazzocca, A.G.M. Strollo, An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm, in: FPL 2003, LNCS 2778, pp. 292–302.
6. K. Jarvinen, M. Tommiska, J. Skytta, A fully pipelined memoryless 17.8 Gbps AES-128 encryptor, in: International Symposium on Field Programmable Gate Arrays, 2003, pp. 207–215.
7. A. Hodjat, I. Verbauwhede, A 21.54 Gbits/s fully pipelined AES processor on FPGA, in: IEEE Symposium on Field-Programmable Custom Computing Machines, 2004.
8. A.J. Elbirt, W. Yip, B. Chetwynd, C. Paar, An FPGA-based performance evaluation of the AES block cipher candidate algorithm

