Optimized Full Parallelism AES Encryption / Decryption

Vidyashri.M.Chandkavathe[#1], Mrs. Rashmi S Bhaskar[*2],
[#] *M.Tech (VLSI and ES): Dept. ECE ,BNMIT Bengaluru, India*
** Assistant Professor: Dept. of ECE, BNMIT Bengaluru, India*

Abstract-

Cryptography is the art and technology of accomplishing security by using encoding messages to cause them to readable. In today's world Advanced Encryption Standard (AES) is widely used symmetric key encryption standard. Different architectural models are implemented in this paper. These implemented models adopt task level and data level parallelism so as to increase the performance. The proposed architecture utilizes parallelism to reduce the latency between the processors thus achieving higher speed and throughput. Compared to other different architectural model full parallelism achieves higher throughput and better performance.

Keywords — *symmetric key, data level parallelism, task level parallelism, Advanced Encryption Standard (AES)..*

I. INTRODUCTION

In nowadays virtual world there is excessive boom within the networking technology that leads a common way of life for interchanging of the data very significantly. So it is far more susceptible of duplicating of data and re-disbursed with the aid of hackers. Consequently we need to protect the facts while transmitting it. For shielding each stored and in transit data world, encryption has emerged as a disintegrable a part of all communication networks and records processing structures. With the increase of complex digital conversation security has become an increasingly more essential feature. Protecting the information like credit card banking transactions and social safety numbers through encryption is turning into increasingly critical to daily existence. Encryption is a totally common technique for promoting the facts security. The evolution of encryption is moving in the direction of a future of countless opportunities.

II. ADVANCED ENCRYPTION STANDARD (AES)

AES is a symmetric encryption which takes input of 128-bit and performs several rounds of adjustments to generate output cipher text. Every 128-bit information block is processed in a 4-by-4 array of bytes, known as the state. The key which is called as round key and its size may be 128, 192 or 256 bits. The quantity of rounds repeated inside the AES, Nr, is described by way of the length of the round key, that's 10, 12 or 14 for key lengths of 128, 192 or 256 bits, respectively. Fig. 1 indicates AES Encryption steps. The AES encryption carried out in opposite direction leads to decryption. There are four basic operations applied in encryption as follows.

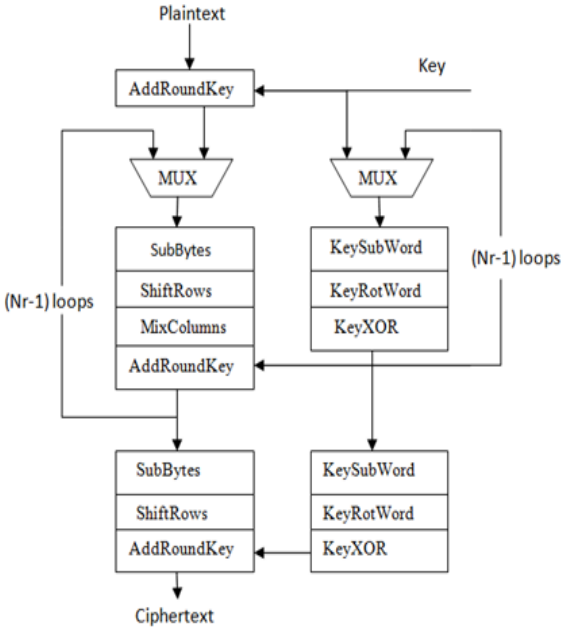

Fig.1 Block Diagram of AES encryption

A. SubBytes

The SubBytes operation carries out a byte substitution in nonlinear way. It replaces every byte in the input state by using some other byte consistent with the substitution field (known as the S-Box). The S-box is computed primarily based on a multiplicative inverse carried out in the finite field GF and a bitwise affine transformation. It substitutes all bytes of the country array the use of a LUT that is a 16x16 matrix of bytes, often referred to as S-field.

B. ShiftRow

The primary row in array is unaffected. The bytes in the second will be shifted one byte to the left. Similarly third, and forth rows are shifted with two bytes and three bytes to the left, respectively. This operation is nonlinear.

C. MixColumn

During the MixColumn process, every column of the state array is considered as a

polynomial over GF. Modulo is multiplied with a fixed polynomial a(x), which is given by

$$a(x)=\{03\}x^3+\{01\}x^2+\{01\}x+\{02\} \quad (1)$$

the result is the corresponding column of the output state.

D. AddRoundKey

To the state array round key is added using a bit wise XOR operation. Key expansion process helps in calculating round keys. If round keys are calculated at the fly for each information block, it is far referred to as AES with online key expansion. Then again, for maximum packages, the encryption keys do not alternate as frequently as data. As a result, spherical keys can be calculated before the encryption technique, and saved regular for a time period in local memory or registers. This is referred to as AES with offline key expansion.

Further, there are 3 steps in each key expansion round.

1. KeySubWord: A four byte word is taken as input in this operation and substitutes each byte within the input to every other byte in line with the S-box to produce an output word.
2. KeyRotWord: The function KeyRotWord performs a cyclic permutation on a word [a3,a2,a1,a0], and returns the word [a2,a1, a0, a3] as output.
3. KeyXOR: The XOR operation is carried out between the previous word, w[i – 1], and the word Nk positions earlier, w[i – Nk] to get a word w[i]. Nk equals 4, 6 or 8 for the different key lengths of 128, 192 or 256 bits, respectively.

III. AES DECRYPTION PROCESS

Similar to encryption process decryption also have four different operations but in the inverse manner. The operation Add Roundkey is explained before in the AES encryption process. This operation remains same in the decryption.

A. Inverse Sub Bytes

Initially inverse S-box is generated. This S-box is generated by affine transformation which is followed by multiplicative inverse. Once this is generated then every byte from the input state is replaced with another byte. This replacement is carried out according to inverse S-box.

B. Inverse Shift Rows

As explained about shift rows operation before the rows are shifted left. But in case of decryption inverse of the shift rows operation is carried out. In this transformation only the first row is kept as it is without changing. The bytes in the second row are cyclically shifted right by one. Similarly bytes in the third and fourth rows are right by 2 and 3 respectively.

C. Inverse Mix Columns

In the state array each column is multiplied by one inverse column matrix which is standard one.

D. Add Round Key

Using a bitwise XOR operation a round key will be added to the state array. These round keys are calculated with the help of key expansion process. Here the key is not used as in the encryption process. In other words it is used in reverse manner i.e. first round uses key 10 and last round uses key 1.

IV. PROPOSED WORK

In this section we are going to implement two different architectures. Different architectures are One-Task One-Processor and Full Parallelism.

A. One-Task One-Processor

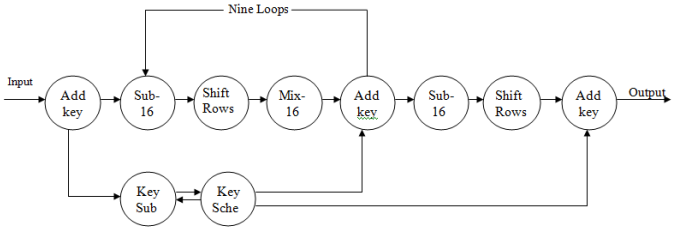

Figure 2: One-task One-Processor Data Flow Diagram

In case of One-Task One-Processor each task carried out is mapped to each processor. Each step is in the algorithm is considered as a task which is shown below in the dataflow diagram in fig 2. It is the simplest and straightforward AES implementation. Each task is mapped to one processor as shown in the figure above. Hence this type of implementation is called as one-task one-processor. The key expansion process is carried out in parallel with the main algorithm. The number of loops in the implementation determines the throughput. This implementation has nine loops which determine the throughput. It requires 10 processors. There are nine loops in this implementation and each loop has 4 different operations.

B. Full Parallelism

The throughput of the earlier implementation can be further increased by making use of Full-parallelism. This AES implementation combines two different implementations. Those are parallel sub bytes mix columns and loop unrolling. The dataflow diagram of full parallelism is shown below in the figure 3. It combines the operation of two implementation .Hence the throughput is expected to be higher than the OTOP model. It applies both task parallelism and data parallelism. Throughput of this implantation is determined by the MixColumns-4 operation.

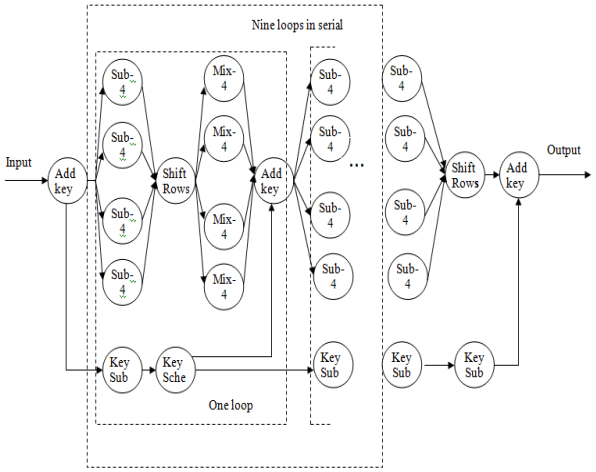

Figure 3: Full Parallelism Data Flow Diagram

V. RESULTS

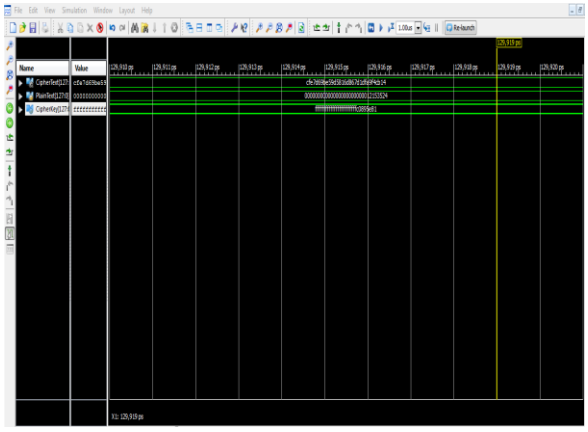

Fig 4. One-Task One-Processor

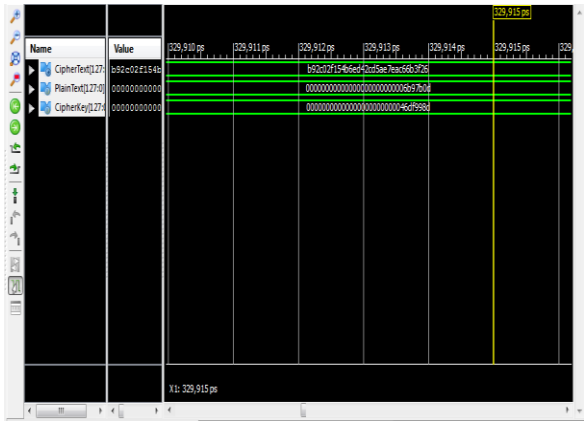

Fig 5. Full Parallelism

The above figures show simulation results of OTOP and full parallelism encryption.

Table 1. Encryption Delay for Different Models

Models	Encryption Delay(ns)
OTOP	100.114
Full Parallelism	28.698

VI. CONCLUSION

One-Task One-Processor makes use of one processor for each task. Full-Paralelism combines the Parallel-SubBytes-MixColumns model and loop unrolling to achieve high throughput. Data level parallelism and task level parallelism is used in full parallelism to reduce the latency between the processors and thus increasing the throughput. Encryption delay of full parallelism is less. Compared to OTOP model full parallelism has high throughput.

ACKNOWLEDGMENT

Authors wish to acknowledge Staff of Electronics & Communication department for their support. Authors also wish to acknowledge the BNMIT management for the encouragement in carrying out this work.

REFERENCES

[1] NIST, “Advanced Encryption Standard (AES),” http://csrc.nist.-gov/publications/fips/fips197/fips-197.pdf, Nov. 2001.

[2] A. Hodjat and I. Verbauwhede, “Area-Throughput Trade-Offs for FullyPipelined 30 to 70 Gbits/s AES Processors,”IEEE Trans. Computers, vol. 55, no. 4, pp. 366-372, Apr.

[3] S. Morioka and A. Satoh, “A 10-gbps full-AES Crypto Design with a Twisted BDD s-Box Architecture,” IEEE Trans. Very Large Scale Integration Systems, vol. 12, no. 7, pp. 686-691, July 2004.

[4] [D. Mukhopadhyay and D. RoyChowdhury, “An Efficient end to End Design of Rijndael Cryptosystem in 0:18_m CMOS,” Proc. 18th Int’l Conf. VLSI Design, pp. 405-410, Jan. 2005