Implementation of Parallel AES Encryption Engines for Multi-Core Processor Arrays

Y. VAMSI KRISHNA¹, V. SÂNDHYA²
¹Assistant Professor, Dept of ECE, MLRITM, Hyderabad, India.
²PG Scholar, Dept of ECE, MLRITM, Hyderabad, India.com.

Abstract: The Advanced Encryption Standard (AES) is an encryption standard chosen by the National Institute of Standards and Technology (NIST) in 2001, which has its origin in the Rijndael block cipher. In this work, we derive three novel composite field arithmetic(CFA) Advanced Encryption Standard (AES) S-boxes of the field GF. The best construction is selected after a sequence of algorithmic and architectural optimization processes. Furthermore, for each composite field constructions, there exists eight possible isomorphic mappings. Therefore, after the exploitation of a new common sub expression elimination algorithm, the isomorphic mapping that results in the minimal implementation area cost is chosen. High throughput hardware implementations of our proposed CFA AES Sboxes are reported towards the end of this paper. Through the exploitation of both algebraic normal form and seven stages fine-grained pipelining, our best case achieves a high throughput in field-programmable gate array. At the same time while encryption process immunity of encryption is taken into the account. Thus the data corruption due to Single Event Upset can be avoided and the performance was increased.

Keywords: Advanced Encryption Standard (AES), ASAP, Fine-Grained, Many-Core, Parallel Processor, Software, Synchronous Dataflow.

I. INTRODUCTION

With the development of information technology, protecting sensitive information via encryption is becoming more and more important to daily life. In 2001, The National Institute of Standards and Technology (NIST) selected the Rijndael algorithm as the Advanced Encryption Standard (AES), which replaced the Data Encryption Standard (DES). Since then, AES has been widely used in a variety of applications, such as secure communication systems, high-performance database servers, digital video/audio recorders, RFID tags, and smartcards. According to Pollack’s Rule, the performance increase of architecture is roughly proportional to the square root of its increase in complexity. The rule implies that if we double the logic area in a processor, the performance of the core speed up around 40 percent. Much core architecture has the potential to provide near-linear performance improvement with complexity. For instance, instead of building a complicated core twice as large as before, a processor containing two cores (each is identical to the other) could achieve a possible performance improvement if the application can be fully parallelized. Therefore, if the target application has enough inherent parallelism, architecture with thousands of small cores would offer a better performance than one with a few large cores within the same die area.

II. ADVANCED ENCRYPTION STANDARD

AES is a symmetric encryption algorithm, and it takes a 128-bit data block as input and performs several rounds of transformations to generate output cipher text. Each 128-bit data block is processed in a 4-by-4 array of bytes, called the state. The round key size can be 128, 192 or 256 bits. The number of rounds repeated in the AES, Nr, is defined by the length of the round key, which is 10, 12 or 14 for key lengths of 128, 192 or 256 bits, respectively.
Fig 1. Block diagram of AES encryption. For encryption, there are four basic transformations applied as follows:

**Sub Bytes**: The Sub Bytes operation is a nonlinear byte substitution. Each byte from the input state is replaced by another byte according to the substitution box (called the S-box). The S-box is computed based on a multiplicative inverse in the finite field GF(28) and a bitwise affine transformation.

**Shift rows**: In the Shift Rows transformation, the first row of the state array remains unchanged. The bytes in the second, third, and fourth rows are cyclically shifted by one, two, and three bytes to the left, respectively.

**Mix columns**: During the Mix Columns process, each column of the state array is considered as a polynomial over GF(28). After multiplying modulo x4+1 with a fixed polynomial a(x), given by a(x)={03}x3+{01}x2+{01}x+{02} the result is the corresponding column of the output state.

**Addroundkey**: A round key is added to the state array using a bitwise exclusive-or (XOR) operation. Round keys are calculated in the key expansion process. If Round keys are calculated on the fly for each data block, it is called AES with online key expansion. On the other hand, for most applications, the encryption keys do not change as frequently as data. As a result, round keys can be calculated before the encryption process, and kept constant for a period of time in local memory or registers. This is called AES with offline key expansion. In this paper, both the online and offline key expansion AES algorithms are examined. Similarly, there are three steps in each key expansion round.

**Keysubword**: The KeySubWord operation takes a fourbyte input word and produces an output word by substituting each byte in the input to another byte According to the S-box.

**Keyrotword**: The function Key Rot Word takes a word [a3; a2; a1; a0], performs a cyclic permutation, and returns the word [a2; a1; a0; a3] as output.

**Key xor**: Every word w[j] is equal to the XOR of the previous word, w[j - 1], and the word Nk positions earlier, w[j - Nk]. Nk equals 4, 6 or 8 for the key lengths of 128, 192 or 256 bits, respectively. The decryption algorithm applies the inverse transformations in the same manner as the encipherment. As a result, we only consider the encryption algorithm in this work for simplicity, since the decipherment yields very similar results.

### III. PARALLEL-MIX COLUMNS

Besides loop unrolling, another way to increase the throughput put of the OTOP model is to reduce the main loop’s latency in the AES algorithm. In a single loop, the execution delay of Mix Columns-16 results in 60 percent of the total latency. Each Mix Columns-16 operates on a four-column data block, and the operation on each column is independent. Therefore, each Mix Columns-16 processor can be replaced by four Mix Columns-4s. Each MixColumns-4 actor computes only one column rather than a whole data block. As a result, the throughput put of the Parallel-Mix Columns implementation is increased to 2,180 cycles per block, equaling 136.25 cycles per byte. The data flow diagram and mapping of the Parallel-Mix Columns model are shown in Figs.2a and 2b. Each core on our targeted computational platform can only support two statically configured input ports. Three cores, each called Merge Core, are

![Fig2. data flow diagram and mapping of the Parallel-Mix Columns model.](image-url)

**A. parallel-Sub Bytes-Mix Columns**

In the Parallel-Mix Columns implementation, SubBytes-16 requires 132 cycles to encrypt one data block, which contributes the largest execution delay in one loop. In order to increase the throughput further, we parallelize one Sub Bytes-16 in to four SubBytes-4s, which is shown in Fig.7a. In this implementation, each SubBytes-4 processes 4 bytes rather than 16 bytes in one data block. The effective execution delay of the Sub Bytes process is decreased to 40 cycles per block, only around one fourth as before. Therefore, the throughput of the Parallel Sub Bytes- Mix Columns model is increased to 1,350 cycles per block, equaling 84.375 cycles per byte. The mapping graph of the Parallel-Sub Bytes- Mix-Columns implementation on AsAP shown in Fig.7b requires 22 cores. Instead of parallelizing SubBytes-16 in to four SubByte-4s, we can replace it with 16 SubBytes-1s. The effective execution delay of the Sub Bytes process is reduced to 10 cycles. As a result, the latency of one-loop decreases to 120 cycles. Therefore, the throughput of the cipher is increased to 67.5 cycles per byte. However, it requires seven additional cores dedicated to communication.
Implementation Of Parallel Aes Encryption Engines For Multi-Core Processor Arrays

(four Merge Cores and three Dispatch Cores), which impair the area and energy efficiency of the implementation.

IV. AES MODES OF OPERATION

The AES encryption algorithm accepts one data block and the key and produces the encrypted data block. The input and output data blocks are of identical size. The decryption algorithm accepts one encrypted data block and the key to produce the encrypted data block. Several modes of operation have been defined to apply the AES block cipher to encryption of more than one 128 bit block of data. The most commonly used modes with AES are: electronic code book (ECB) mode, cipher block chaining (CBC) mode, output feedback (OFB) mode, cipher feedback (CFB) mode, and counter (CTR) mode. ECB and CTR are known as no feedback modes whereas CBC, CFB, and OFB are known as feedback modes. In addition, ECB and CBC are referred to as block cipher modes as they require the entire data block before the start of the encryption, and OFB, CFB, and CTR are referred to as stream cipher modes as they operate in a stream-like fashion.

A. Output Feedback Mode

In the OFB mode the output of the encryption is fed back into the input to generate a key generate the cipher data, as illustrated in fig 4 SEU propagation during encryption in OFB mode. In Fig. 4, if an SEU occurs during encryption in the OFB mode then all the subsequent blocks will be corrupted starting from the point where the fault has occurred. This is because the key stream required for encryption and decryption is independent of the plain and cipher data and hence the feedback propagates the faults from one block to another until the end of the encryption process.

Fig. 3. Data flow diagram after loop-un rolling.

B. No-Merge-Parallelism

In contrast to the Small model, the No-merge parallelism model exploits as much parallelism as possible without introducing any cores dedicated to communication, including Merge Cores and Dispatch Cores. The mapping graph of the No merge-parallelism implementation on AsAP. To speedup the implementation, loop unrolling is applied in this model. Each MixColumns-16 is divided into two MixColumns-8s, which helps reduce the effective delay of the Mix Columns process. In order to eliminate additional communication processors and simplify the routing, we combine the Sub Bytes and the Shift Rows stages in one core. This implementation requires 59 cores, and has a throughput of 152 cycles per block, equaling 9.5 cycles per byte. AES cipher can be partitioned into a number of serial and parallel independent tasks corresponding to different steps in the algorithm. However, the throughput of this partitioning is low due to the time-consuming loop operation in the algorithm. In order to enhance the throughput, loop-unrolling is applied to break the dependency among loops and allow the cipher to operate on multiple data blocks simultaneously. To improve the throughput as much as possible, we unroll the loops in both the AES algorithm and the key expansion process by Nr 1 times, which equals nine in our design. The data flow diagram after loop-unrolling is shown in Fig. 3. shows a preliminary AES cipher implementation based on the dataflow diagram. Each task in the data flow diagram is mapped to one small processor. As shown in Fig. 4, seven small processors are required for one loop, four for the AES algorithm and three for the key expansion process, respectively. Therefore, the total number of processors used in this cipher is:

\[
N_{\text{processors}} = (N_r-1) \times N_{\text{one-loop}} + N_{\text{last-round}}
\]

\[
= 9 \times 7 + 7 = 70
\]

Fig. 5 and Fig. 6 summarize the instruction and data memory sages for each processor in the original design, respectively. Each processor in the 70 core AES cipher uses an average of 28 words in instruction memory, which is 22% of all available instruction memory; and an average of 55 words of data memory, which is 43% of all available data memory.
46875. The fault propagation for a single bit error, which was
introduced during the encryption of the 20,000th block at the
Sub Bytes transformation of the 4th byte in the third round.
The propagation of a single bit error that was introduced
during the encryption of the 40,000th block at the Mix
Columns transformation of the 7th byte in the 6th round. In
contrast, if a bit is corrupted during transmission, only a
single bit in the plain data is affected and the error does not
propagate to other parts of the message again for the same
reason that the key stream does not depend on the plain or
cipher data. So the transmission fault is not propagated.
This property is very useful to applications such as satellites
where the transmission channels are very noisy. Hence the OFB
mode has an advantage over the CBC and CFB modes in that
any bit errors that might occur inside cipher data are not
propagated to affect the decryption of subsequent blocks.

V. IMPLEMENTATION AND RESULTS
The implementation of the proposed system using VHDL
Hardware Description Languages and the simulation Results
are as follows (fig 5 and 6).

VI. CONCLUSION
We have presented 16 different AES cipher implementations
with both online and offline key expansion on a fine-grained
many-core system. Each implementation exploits different
levels of data and task parallelism. The smallest design
requires only six processors, equaling 1:02 mm² in a 65 nm
fine-grained many-core system. The fastest design achieves a
throughput of 4.375 cycles per byte, which is 2.21 Gbps
when the processors are running at a frequency of 1.2 GHz.
We also optimize the area of each implementation by
examining the workload of each processor, which reduces the
number of cores used as much as 18 percent. The design on
the fine-grained many-core system achieves energy
efficiencies approximately 2.9-18.1 times higher than other
software platforms, and performance per area on the order of
3.3-15.6 times higher. Overall, the fine-grained many-core
system has been demonstrated to be a very promising
platform for software AES implementations.

VII. REFERENCE
Encryption Engines for Many-Core Processor Array.
encryption for space applications.
(AES),”.
Design and Performance Testing of a 2.29 gb/s Rijndael
Processor.
An Efficient end to End Design of Rijndael Cryptosystem in
0:18_m CMOS.
[6] J.L. Hennessy and D.A. Patterson, Morgan Kaufmann,
Crypto Design with a Twisted BDD s-Box Architecture.
Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s
AES Processor.
Agarwal, S.K. Hsu, H. Kaul, M.A. Anders, and R.K.
Krishnamurthy Apr. 2006,53 gbps Native GF(024P2)
Composite-Field AES-Encrypt/Decrypt Accelerator for
Content-Protection in 45 nm High-Performance
Microprocessors.
ghits/s Fully Pipelined AES Processor on FPGA.
[12] C.J. Chang, C.-W. Huang, K.-H. Chang, Y.-C. Chen,
and C.-C. Hsieh, Nov. 2008 High Throughput 32-Bit AES
Implementation in FPGA.
Perez and J. Gomez-Pulido, Nov. 2008. A New Methodology
to Implement the AES Algorithm Using Partial and Dynamic
Reconfiguration.
Implementation Of Parallel Aes Encryption Engines For Multi-Core Processor Arrays

Author's Profile:

Y. Vamsi Krishna received the B.Tech degree in Electronics and Communication Engineering and M.Tech degree in Digital Electronics and Communication Systems. Currently, he is working as Assistant Professor in the Department of Electronics and Communication Engineering, Marri Laxman Reddy Institute of Technology and Management, Hyderabad, India. He having 3 years of teaching experience.

V.Sandhya received the B.Tech degree in Electronics and Communication Engineering and pursuing M.Tech degree in Embedded systems from Marri Laxman Reddy Institute of Technology and Management, Hyderabad, India. Her area of interests includes Robotics, VLSI Design, ASIC and Signal Processing.