IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 3, June-July, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Parallel AES Encryption Engines Many-Core Processor Arrays

S.Viji[1], J.Kirubakaran[2], Dr.G.K.D Parasanna Venkatesan[3]

[1]ECE, STUDENT OF PGP COLLEGE OF ENGINEERING AND TECHNOLOGY,

[2]AP, PGP COLLEGE OF ENGINEERING AND TECHNOLOGY,

[3]VICE-PRINCIPAL, PROFESSOR, PGP COLLEGE OF ENGINEERING AND TECHNOLOGY.

ABSTRACT- By exploring different granularities of data-level and task-level parallelism,we map 16 implementations of an Advanced Encryption Standard (AES) cipher with both online and offline keyexpansion on a fine-grained many-core system.The smallest design Utilizes only six cores for offline key expansion and eight cores for online key expansion,while the largest requires 107 and 137cores, respectively. In comparison with published AES cipher implementations on general purpose processors,our design has 3.5-15.6 times higher throughput per unit of chip area and 8.2-18.1 times higher energy efficiency.Moreover, the design shows 2.0 times higher throughput than the TIDSPC6201, and 3.3 times higher throughput per unit of chip area and 2.9 times higher energy efficiency than the GeForce8800GTX.

I.INTRODUCTION

With the development of information technology, protecting sensitive information via encryption is becoming more and more important to daily life.In2001,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 an architecture is roughly proportional to the square root of it s increase incomplexity.The rule implies that if we double the logic area in a processor, the performance of the core speed sup around 40 percent. Many core architecture has the potential to provid en earlinear 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 paralle-lized. Therefore, if the target application has enough inherent parallelism, an architecture with thousands of small cores would offer a better performance than one with a few large cores with in the same die area

II. ADVANCED ENCRYPTION STANDARD

AES is asymmetric encryption algorithm, and it takes a 128-bit data block as input and performs several rounds of transformations to generate output ciphert ext.Each128-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 is10,12or14for keylengths of128,192or256 bits, respectively. Fig.1 shows the AES encryption steps with the key expansion process. For encryption, there are four basic transformations applied as follows.The most straight forward implementation of an AES cipher is to apply each step in the algorithm as a task in the data flow diagram as shown in Fig.3a. Then, each task in the data flow diagram can be mapped on one processor on the targeted many-coreplat form. We call this imple-mentation one-task one processor. For simplicity, all of the execution delay (shown in Table1), input rates, and output rate sin the following data flow diagrams are omitted. Since the keyexpansion is processing in parallel with the main algorithm, the through put of the OTOP implementation is determined by the (Nr-1 = 9) loop sin the algorithm.The OTOP implementation requires 10cores on AsA Pas shown in Fig.3b. The through put of the OTOP impleMentation is 3,582 clock cycles per data block, equaling 223.875clockcycles per byte.

1. SubBytes: The SubBytes peration is an on linear byte sub stitution. Each byte from the inpu tstate is replaced by another byte according to the sub stitu-tionbox(calledtheS-box). The S-boxis computed based on a multiplicative inverse in the finite field GF and a bit wise affine transformation.

2. ShiftRows: In the ShiftRows transformation,thefirstrowofthe state array remain sun changed. The byte sin the second,third,and forth rows are cyclically shifted by one, two, and three bytes to the left, respectively.

3. MixColumns: During the MixColumns process,each column of the state array is considered as a Polynomial over GF.After multiplying modulo there sult is the corresponding column of the output state.

4. Add Round Key: A round key is added to the state array using a bit wise exclusive-or (XOR) operation. Round key sare calculated in the key expansion process. If Round key sare calculated on the fly for each data block, it is called AES with on line key expansion.On the other hand, form ost applications, theen cryption keys do not change as frequently as data. Asaresult,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.

1. KeySubWord: The KeySubWord operation takes a four-byte input word and produce an output word by substituting each byte in the input to another byte according to the S-box.

2. KeyRotWord: Thefunction KeyRot Word takes a word [a3; a2; a1; a0,] performs a cyclic permutation, and returns the word [a2; a1; a0; a3] as output.

3. KeyXOR: Every word w[i] is equal to the XOR of the previous word,w[i-1]and the word Nk positions earlier,w[i-Nk] bits, Nkequals 4, 6 or 8 for the key lengths of 128,192or 256 bits, respectively.

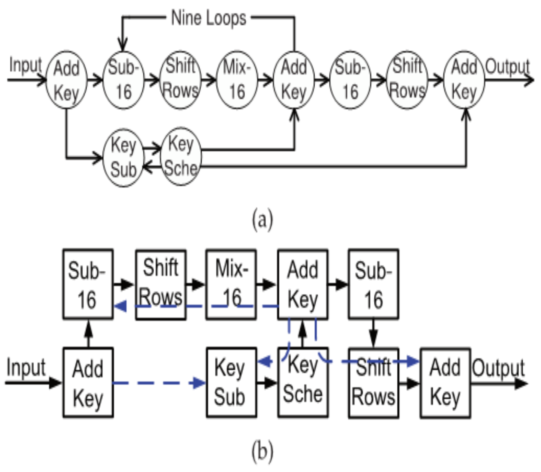

Fig. 3. One-task One-processor (a) dataflow diagram and (b) 10 cores AsAP mapping.

To achieve a moderate through put with fewer cores, we could unroll the main loop sin the AES algorithm by S times (S isdivisible by Nr-1),instead of Nr-1 times.For this example, the nine loop sin the AES algorithm could be split into three blocks, and each block loops three times. The data flow diagram and mapping of the Loop unrolled Three Time simple mentation are shown in Figs.5a and 5b, respectively. Compared to the OTOP model, the through put is improved to 1,098 cycles per data block, which equals 68.625 cycles per byte; while the mapping requires 24 cores,36 fewer than the Loop-unrolled Nine Times

implementation.

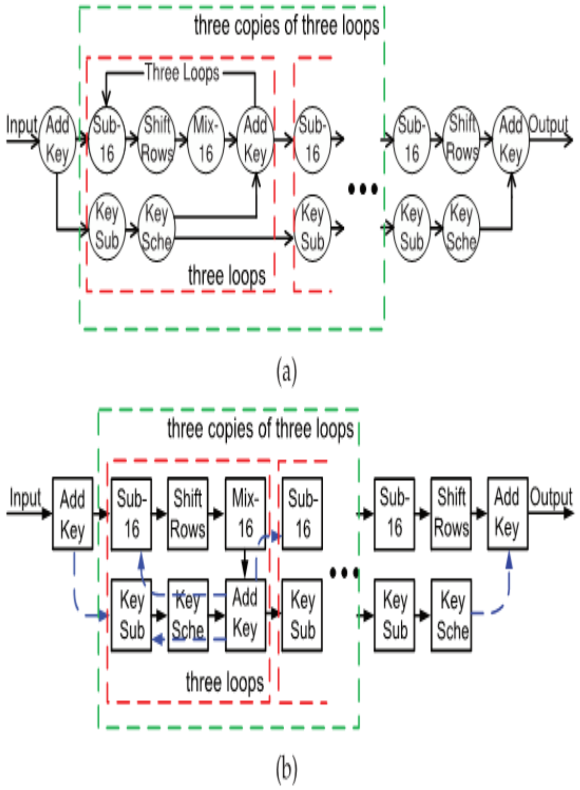

Fig. 5. Loop-unrolled Three Times (a) dataflow diagram and (b) 24 cores AsAP mapping.

IV PARALLEL-MIX COLUMNS

Besides loop unrolling, another way to increase the through put of the OTOP model is to reduce the main loop's latencyin 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 in dependent. Therefore, each Mix Columns-16 processor canbe replaced by four MixColumns-4s.Each MixColumns-4 actor computes only one column rather than a whole data block. As a result, the through 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.6a and 6b. Each core on our targeted computational plat form can only support two statically configured input ports.Three cores,each called Merge Core, are

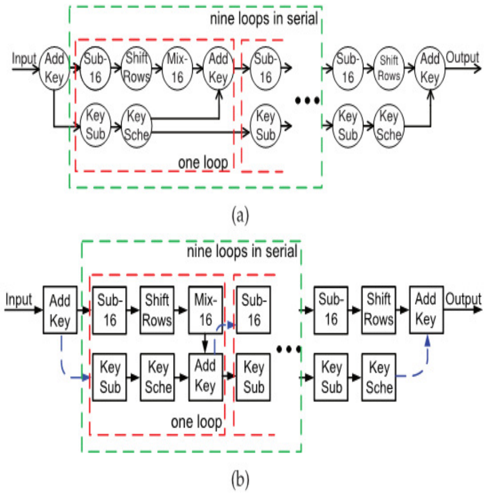

Fig. 4. Loop-unrolled Nine Times (a) dataflow diagram and (b) 60 cores AsAP mapping.

PARALLEL-SUB BYTES-MIX COLUMNS

In the Parallel-MixColumns implementation, SubBytes-16 requires 132 cycles to encrypt one datablock, which contributes the largest execution delay in one loop.In order to increase the throughput further, we parallelize one SubBytes-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 SubBytes process is decreased to 40 cycles per block,only around one fourth as before.Therefore,the throughput of the Parallel-

SubBytes-MixCol-umns model is increased to 1,350 cycles per block, equaling 84.375 cycles per byte.The mapping graph of the Parallel-SubBytes-Mix-Columns implementation on AsAP shown in Fig.7b requires 22cores.Instead of parallelizing SubBytes-16 in tofour SubByte-4s,we can replace it with 16 SubBytes-1s. The effective execution delay of the SubBytes process is reduced to 10cycles. As a result, the latency of one-loop decreases to 120cycles.Therefore, the throughput of the cipher is increased to 67.5cycles per byte.However, it requires seven additional cores dedicated to communication (four Merge Cores and three Dispatch Cores),which impair the area and energy efficiency of the implementation.

. 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 is shown in Fig.10.To speedup the implementation, loop unrolling is applied in this model. Each MixColumns-16 is divided in to two MixColumns-8s, which helps reduce the effective delay of the MixColumns process.In order to eliminate additional communication processors and simplify the routing, we combine the SubBytes and the ShiftRows stages in one core.This implementation requires 59cores,and has a throughput of 152 cycles per block, equaling 9.5 cycles per byte.

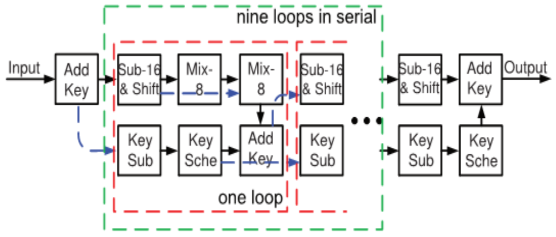

Fig. 10. Fifty Nine cores AsAP mapping of the No-merge-parallelism implementation.

As Fig.1 shows ,the AES cipher can be partitioned into a number of serial and parallel independent tasks corresponding to different step s in the algorithm. However,the through put of this partitioning is low due to the time-consuming loop operation in the algorithm. In order to enhance the through put, loop-un rolling is applied to break the dependency among loops and allow the cipher to operate on multiple data blocks simultaneously. To improve the through put as much as possible, we un roll 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-un rolling is shown in Fig.3. show a preliminary AES cipherimplementationbased on the dataflow diagram. Each task in the data flow diagram is mapped to one small processor. As shown inFig.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 encipher is:

$$\begin{aligned} N_{processors} &= (N_r - 1) \times N_{one-loop} + N_{last-round} \\ &= 9 \times 7 + 7 = 70 \end{aligned}$$

Fig.5 and Fig.6 summarize the instruction and data memory sages for each processor in the original design, respectively.Eachrocessorinthe70 core AES cipher uses an average of 28 words finstruction 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

DESIGN OPTIMIZATION-II

The execution time of both the SubShift and KeyScheduling pro-cessors are lessthan 152 cycles per data block, which means that the above processor fusion steps for processor countr eduction do not create any new bottlenecks for the design. Fig.8 shows the AES Cipher after through put and area optimization. Each loop operation in the AES algorithm requires six small processors. As are sult, the optimized cipher requires 59 core sin total. The processor activity of the final optimized cipher is shown in Fig.9. Similar as previous analysis, the processors before the first MixCol-8 processor,the AddKeyBF and Sub Shift BF, are stalled on their output.While the processors after the first MixCol-8 processor,The AddKey and Sub Shift, are stalled on their input. In summary, compared with the original design, the optimized cipher achieves a 43% higher through put(9.5against16.625cycles per byte) and requires 16% fewer processors (59insteadof70).100Time (cycles)

REFERENCES

[1]NIST,“AdvancedEncryptionStandard(AES),”http://csrgov/publications/fips/fips197/fips-197.pdf,Nov.2001.

[2]NIST,“DataEncryptionStandard(DES),”http://csrc.nispublications/fips/fips46-3/fips46-3.pdf,Oct.1999.

[3]I.Verbauwhede,P.Schaumont,andH.Kuo,“Design Performance Testing of a2.29gb/sRijndaelProcessor,J.Solid-StateCircuits, vol.38,no.3,pp.569-572,Mar.2003.

[4]D.MukhopadhyayandD.RoyChowdhury,“AnEfficient End Design of Rijndael Cryptosystem in 0:18m CMOS, 18thInt’lConf.VLSIDesign, pp.405-410,Jan.2005.

[5]J.L.HennessyandD.A.Patterson, ComputerArchitect Quantitative Approach, fourthed.MorganKaufmann,2007.

[6]S.MoriokaandA.Satoh,“A10-gbpsfull-AESCryptoDesign a Twisted BDDs-Box Architecture,”IEEETrans.VeryLarge IntegrationSystems,vol.12,no.7,pp.686-691,July2004.