Energy-Efficient AES Ciphers on a Fine-Grained Many-Core System

Bin Liu and Bevan M. Baas
Department of Electrical and Computer Engineering
University of California, Davis

***Abstract*—By exploring different granularities of data-level and task-level parallelism, we propose 16 implementations of an Advanced Encryption Standard (AES) cipher with both online and offline key expansion on a fine-grained many-core system. The smallest design utilizes only 6 cores for offline key expansion and 8 cores for online key expansion, while the largest requires 107 cores and 137 cores, respectively. With frequency and voltage scaling, the power of different implementations could be reduced as much as 32%. In comparison with published AES cipher implementations on other software platforms, our design has 3.3–15.6 times higher throughput per chip area and 3.4–21.7 times higher energy efficiency.**

I. Introduction

The Rijndael algorithm was selected by the National Institute of Standards and Technology (NIST) as the Advanced Encryption Standard (AES) [1] to replace the Data Encryption Standard (DES) in 2001. Since then some AES implementations based on different software platforms have been reported in the literature. Matsui et al. proposed a bitslice AES implementation on Intel Core 2, which achieves a 9.2 clock cycles per byte throughput for a data chunk longer than 2048 bytes, equaling 1.85 Gbps when the core is running at its maximum frequency of 2.13 GHz [2]. Bernstein et al. investigated the opportunities of reducing instruction count and cycles by combining different instructions together for various architectures [3]. Both bitslice and specific sets of instructions from SSSE3 (Supplemental Streaming SIMD Extensions 3) are utilized to enhance the performance of Intel Core i7 920 as high as 6.92 clock cycles per byte [4]. There is also a trend to use GPUs (Graphic Processing Units) and DSP processors to implement the AES algorithm. Wollinger et al. compared different encryption algorithms on a TMS320C6X processor and achieved a 14.25 clock cycles per byte [5]. Manavski presented an AES implementation with a peak throughput of 8.28 Gbps on a GeForce 8800 GTX chip when the input data block is longer than 8 MB [6].

In this paper, we present various software implementations of the AES algorithm with different data and task parallelism granularity, and shows that AES implementations on a fine-grained many-core system can achieve high performance, throughput per unit of chip area and energy efficiency compared to other software platforms. Both the online and offline key expansion process for each implementation model are discussed. The reminder of this paper is organized as follows. Section II introduces the AES algorithm and the features of the targeted fine-grained many-core system. In Section III, various implementations are mapped on the targeted platform. Section IV presents the power optimization methodology. Section V compares our work with other software designs. Finally, Section VI concludes the paper.

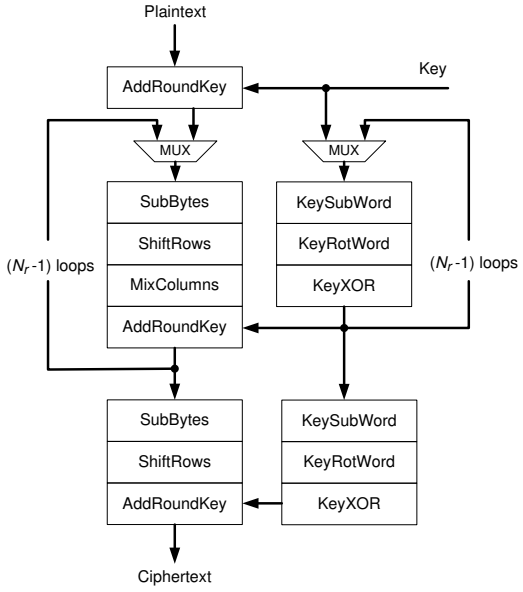

Fig. 1. Block diagram of AES encryption

II. AES and Targeted Many-core Architecture

A. 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 ciphertext. 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, N_r, 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 shows the AES encryption steps with the key expansion process. For encryption, there are four basic transformations applied as follows:

1) *SubBytes:* The *SubBytes* operation is a non-linear byte substitution. Each byte from the input state is replaced by another byte according to the substitution box (called

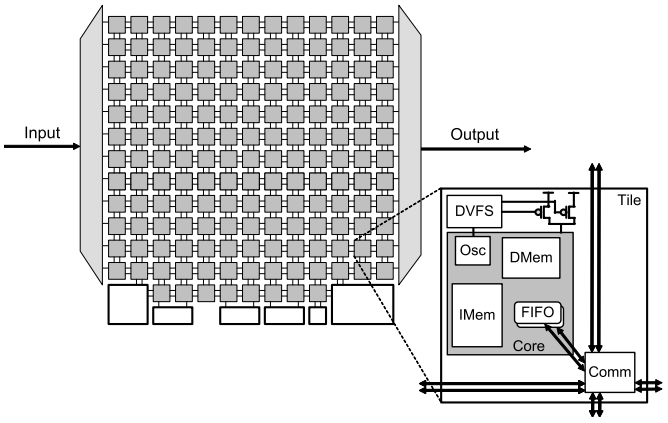

Fig. 2. Block diagram of the targeted many-core platform [7].

the S-box). The S-box is computed based on a multiplicative inverse in the finite field GF(2^8) and a bitwise affine transformation.

2) *ShiftRows:* In the *ShiftRows* transformation, the first row of the *state* array remains unchanged. The bytes in 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(2^8). After multiplying modulo x^4+1 with a fixed polynomial $a(x)$, 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.
4) *AddRoundKey:* A *round key* is added to the *state* array using a bitwise exclusive-or (XOR) operation. *Round key*s are calculated in the key expansion process. If *Round key*s 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 key*s 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.

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:* The function *KeyRotWord* takes a word $[a_3, a_2, a_1, a_0]$, performs a cyclic permutation, and returns the word $[a_2, a_1, a_0, a_3]$ 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, $w[i-Nk]$. Nk equals 4, 6 or 8 for the key lengths of 128, 192 or 256 bits, respectively.

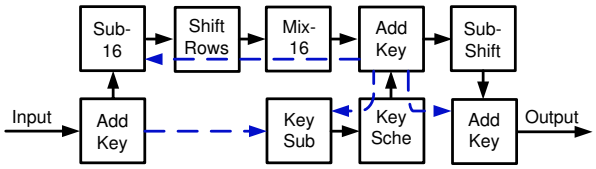

Fig. 3. One-task One-processor (OTOP) 9 cores AsAP mapping.

B. Fine-grained Many-core Processor Array

The targeted Asynchronous Array of Simple Processors (AsAP) architecture is an example of a fine-grained many-core computation platform, supporting globally-asynchronous locally-synchronous (GALS) on-chip network and dynamic voltage and frequency scaling (DVFS) [7].

Fig. 2 shows the block diagram of AsAP. The computational platform is composed of 164 small identical processors, three hardware accelerators and three 16 KB shared memories. All processors and shared memories are clocked by local fully independent oscillators and are connected by a reconfigurable 2D-mesh network that supports both nearby and long-distance communication [8]. Each tile on the platform can be statically configured to take input data from two links, while sending its output to other processors via dynamic configuration.

Each simple processor has a 6-stage pipeline, which issues one instruction per clock cycle. Moreover, no application-specific instructions are implemented. Each processor has a 128×32-bit instruction memory and a 128×16-bit data memory. Each processor occupies 0.17 mm^2 and has a maximum clock frequency of 1.2 GHz. The 167-processor chip was fabricated in 65 nm CMOS technology [7]. And each processor can run at one of the two power supply voltages to achieve the highest energy efficiency.

III. Proposed AES Implementations

In this section, we present the eight AES implementations with online key expansion in detail, since the offline implementations can be derived by removing the cores used for key expansion from the online designs. For simplicity, we focus on the situation with a 128-bit key and $N_r = 10$ in this paper.

1) One-task One-Processor (OTOP): The most straightforward implementation is to map each task in the AES algorithm on one processor, as shown in Fig. 3. Without impairing performance, the *SubBytes* and *ShiftRows* processors in the last round are fused into one processor (as *SubShift*) to save one processor.
2) Parallel-MixColumns: To increase the throughput of OTOP, each *MixColumns-16* processor can be split into four *MixColumns-4*s. The reason is that each *MixColumns-16* operates on a four-column data block, and the operation on each column of the data block is independent.
3) Parallel-SubBytes-MixColumns: Besides parallelizing the *MixColumns-16* processor, we parallelize one *SubBytes-16* into four *SubBytes-4*s to increase the throughput further. In order to save three processors, we

TABLE I

THROUGHPUT AND THE NUMBER OF CORES REQUIRED BY DIFFERENT IMPLEMENTATIONS. COMMUNICATION CORES ARE USED FOR ROUTING ONLY, INCLUDING *MergeCores* AND *DispatchCores*. ALL OF THE THROUGHPUT PER CORE NUMBERS ARE NORMALIZED TO THE PARALLEL-MIXCOLUMNS MODEL WITH ONLINE KEY EXPANSION.

Implementation	1/Throughput (cycles/byte)	Online Key Expansion			Offline Key Expansion		
		Total Cores	Comm. Cores	Normalized Throughput/Core	Total Cores	Comm. Cores	Normalized Throughput/Core
Small	167.375	8	0	1.53	6	0	2.04
One-task one-processor	223.875	9	0	1.01	7	0	1.30
Parallel-Mixcolumns	136.250	15	3	1	12	2	1.25
Parallel-SubBytes-Mixcolumns	84.375	18	3	1.35	15	2	1.61
Loop-unrolled Three Times	68.625	23	0	1.29	15	0	1.99
Loop-unrolled Nine Times	16.625	50	0	2.46	30	0	4.10
No-merge-parallelism	9.500	59	0	**3.65**	39	0	**5.52**
Full-parallelism	**4.375**	**137**	**30**	3.41	**107**	**20**	4.37

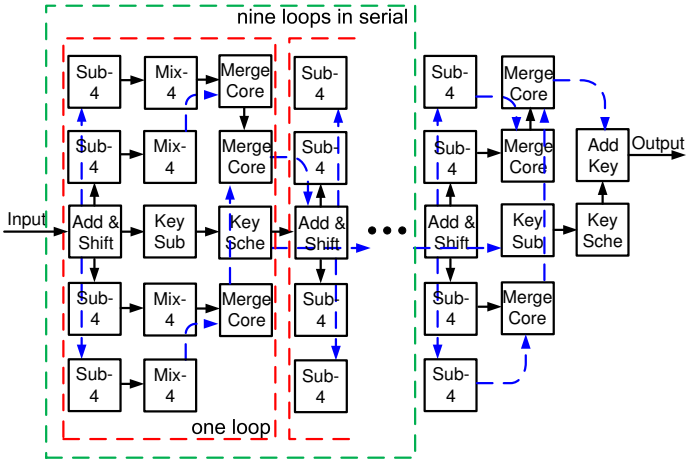

Fig. 4. Full-parallelism 137 cores AsAP mapping.

alternate the sequence of the *SubBytes* and the *ShiftRows* stages, which would not affect encryption results.

4) Loop-unrolled Three Times: Instead of parallelizing single processor from OTOP model, another way to increase the throughput is loop unrolling. In this implementation, the nine loops in the AES main algorithm are split into three blocks, and each block loops three times.
5) Loop-unrolled Nine Times: The loops in the AES main algorithm and the key expansion process are unrolled by nine and ten times, respectively. The *SubBytes* and *ShiftRows* processors in the same loop are fused into one processor, which saves 10 processors without impairing throughput.
6) Full-parallelism: The implementation combines the Parallel-SubBytes-MixColumns model and loop unrolling to achieve the highest throughput among all of the models introduced in this paper. It also requires 137 cores as shown in Fig. 4, which is the largest implementation of all. The *MergeCore*s are used for routing only.
7) Small: The Small model implements an AES cipher with the fewest cores.
8) No-merge-parallelism: The No-merge-parallelism model exploits as much parallelism as possible without introducing any cores dedicated to communication and highly optimized for area and throughput [9].

The throughput, number of cores used and normalized throughput per core of all implementations are listed in Table I. All implementation mappings are described in detail in [10].

IV. ENERGY OPTIMIZATION BASED ON DVFS

The power number of each implementation working under 1.3 V and 1.2 GHz is listed in Column 3 of Table II. Since AsAP cores can be set to run at different frequencies and two supplied voltages, it provides an opportunity to optimize the power of each implementation further.

A. Frequency Scaling

Based on the execution activity, each processor has an optimal running frequency. The optimized frequencies make processors execute in 100% activity as much as possible without impairing performance.

For example, the critical processors of the full-parallelism's model are *MixColumn-4*s, which execute 70 cycles and decide the throughput of the implementation. As a result, the *MixColumn-4*s must run at a frequency as high as possible to guarantee the best throughput performance of the system. For other processors in the same model, the optimal frequency is obtained as:

$$f_{Opt.i} = \frac{N_{Exe.i}}{70} \cdot f_{MixColumn-4} \quad (2)$$

where $N_{Exe.i}$ is the number of execution cycles of the i^{th} processor for processing one data block, respectively.

By running at these optimal frequencies, the power wasted by processors waiting for available input or output buffer is eliminated. The optimized power based on frequency scaling of each implementation is listed in Column 4 of Table II. Compared with the case of all processors running at 1.2 GHz, frequency scaling could reduce the power up to 15.42%.

B. Dual Supply Voltage Scaling

Since only the critical processors in each implementation are required to be run at the highest frequency, the supply voltages could be decreased for other processors depending on their own frequencies. The supply voltage reduction can lower the

TABLE II
THROUGHPUT AND POWER NUMBER OF EACH AES IMPLEMENTATION WITH ONLINE KEY EXPANSION WHEN ALL THE CORES WORK UNDER 1.3 V AND 1.2 GHZ.

Implementation	Max. Throughput (Mbps)	Power Before Opt. (mW)	Freq. Scaling		Freq. and Volt. Scaling	
			Power (mW)	Saving Perc.	Power (mW)	Saving Perc.
Small	57.83	179.5	157.7	12.14%	141.3	21.28%
One-task one-processor	43.24	169.0	146.2	13.49%	130.0	23.08%
Parallel-Mixcolumns	71.05	333.7	316.2	5.24%	294.0	11.90%
Parallel-SubBytes-Mixcolumns	114.73	412.9	391.8	5.11%	365.3	11.53%
Loop-unrolled Three Times	141.06	413.2	362.9	12.17%	315.0	23.77%
Loop-unrolled Nine Times	582.26	1353.3	1203.7	11.05%	929.4	31.32%
No-merge-parallelism	1018.95	2357.9	2075.8	11.96%	1732.4	26.53%
Full-parallelism	2212.57	6064.7	5129.8	15.42%	4534.7	25.23%

system's power consumption further as a result of the quadratic relationship between supply voltage and power. Our targeted computational platform can support two different supply voltages, Vdd_{High} and Vdd_{Low}. For each implementation, we can choose either one of these voltages for each processor. If at Vdd_{Low} the processor can reach its optimal running frequency, its supply voltage should be set as Vdd_{Low}; otherwise, Vdd_{High} is applied.

The total power after frequency and voltage optimization of each implementation is listed in Column 6 of Table II. And the power saving could be achieved up to 31.32%.

V. RELATED WORK AND COMPARISON

In order to make a fair comparison, all of the referenced data are scaled to 65 nm CMOS technology with a supply voltage of 1.3 V. The area data are scaled to 65 nm with a $1/(s^2)$ reduction, where s equals the ratio between the minimum feature size of the old technology and 65 nm. The delay and power data are scaled by SPICE simulation results of a fanout-of-4 (FO4) inverter under different technologies and supply voltages with predictive technology model (PTM) [11].

In this section, we use the metrics of throughout per chip area (Mbps/mm^2) and energy per workload bit (nJ/bit) to compare the area efficiency and energy efficiency of various designs. Compared to the highly optimized AES ciphers on CPUs with bitslice [2], [4], the proposed AES cipher on AsAP has 3.5–15.6 times higher throughput per unit of chip area and consumes 9.8–21.7 times less energy to encrypt a fixed amount of data. Compared with TI DSP C6201, an 8-way VLIW architecture for high performance DSP applications, our design has 2 times higher throughput. Compared to GeForce 8800 GTX, our design shows a 3.3 times higher throughput per unit of chip area and 3.4 times higher energy efficiency.

VI. CONCLUSION

We have presented 16 different AES cipher implementations with both online and offline key expansion on a fine-grained many-core system and optimized their power with frequency and voltage scaling. The smallest design requires only 6 processors, equaling 1.02 mm^2 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. The design on the fine-grained many-core system achieves energy efficiencies approximately 3.4–21.7 times higher than other software platforms, and performance per area on the order of 3.3–15.6 times higher.

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge support from NSF Grant 0430090, 0903549 and CAREER Award 0546907, SRC GRC Grant 1598 and 1971, CSR Grant 1659, UC Micro, ST Microelectronics, Intel, and Intellasys.

REFERENCES

[1] NIST, "Advanced encryption standard (AES)," http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, Nov. 2001.

[2] Mitsuru Matsui and Junko Nakajima, "On the power of bitslice implementation on intel core2 processor," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, vol. 4727 of *Lecture Notes in Computer Science*, pp. 121–134. 2007.

[3] Daniel Bernstein and Peter Schwabe, "New AES Software speed records," in *Progress in Cryptology - INDOCRYPT 2008*, vol. 5365 of *Lecture Notes in Computer Science*, pp. 322–336. 2008.

[4] Emilia Kasper and Peter Schwabe, "Faster and timing-attack resistant AES-GCM," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, vol. 5747 of *Lecture Notes in Computer Science*, pp. 1–17. 2009.

[5] T. Wollinger, M. Wang, J. Cuajardo, and C. Paar, "How well are high-end DSPs suited for the AES algorithm?," in *The Third AES Candidate Conference*, Apr. 2000, pp. 94–105.

[6] S.A. Manavski, "CUDA compatible GPU as an efficient hardware accelerator for AES cryptography," in *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*, Nov. 2007, pp. 65–68.

[7] D. N. Truong, W. H. Cheng, T. Mohsenin, Z. Yu, A. T. Jacobson, G. Landge, M. J. Meeuwsen, A. T. Tran, Z. Xiao, E. W. Work, J. W. Webb, P. Mejia, and B. M. Baas, "A 167-processor computational platform in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.

[8] A. T. Tran, D. N. Truong, and B. M. Baas, "A reconfigurable source-synchronous on-chip network for GALS many-core platforms," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 6, pp. 897–910, Jun. 2010.

[9] Bin Liu and Bevan M. Baas, "A high-performance area-efficient AES cipher on a many-core platform," in *IEEE Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Nov. 2011.

[10] Bin Liu and B. M. Baas, "Parallel AES encryption engines for many-core processor arrays," *to appear on IEEE Transactions on Computers*.

[11] Aaron Stillmaker, "Exploration of technology scaling of CMOS circuits from 180 nm to 22 nm using PTM models in HSPICE," *Technical Report UC Davis*, June 2011.