A High-Performance Area-Efficient AES Cipher on a Many-Core Platform

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

***Abstract*—This paper presents the design and software implementation of a high-performance area-efficient Advanced Encryption Standard (AES) cipher on a many-core platform. A preliminary cipher design is partitioned and mapped to an array of 70 small processors, and offers a throughput of 16.625 clock cycles per byte. The usage of instruction and data memory, and the workload of each processor are characterized for further optimization. Through workload balancing and processor fusion, the throughput of the cipher is increased by 43% to 9.5 clock cycles per byte, while the number of processors utilized is reduced to 59, which is only 10.03 mm^2 in a 65 nm fine-grained many-core system. In comparison with published AES implementations on general purpose processors, our design has 3.6–10.7 times higher throughput per area. Moreover, the presented design shows 1.5 times higher throughput than the TI DSP C6201 and 3.4 times higher throughput per area than the GeForce 8800 GTX.**

***Index Terms*—Advanced Encryption Standard (AES), parallel mapping, many-core processor, Asynchronous Array of Simple Processors (AsAP).**

I. Introduction

In 2001, the National Institute of Standards and Technology (NIST) selected the Rijndael algorithm as the Advanced Encryption Standard (AES) [1] as a replacement for the Data Encryption Standard (DES) [2]. Since then, AES has been widely used in a variety of applications, such as communication systems, high performance servers, RFID tags and smart cards.

To adapt different applications' requirements, the flexibility of software solutions are attractive. Some AES implementations based on different software platforms have been reported in the literature. Matsui et al. proposed a bitsliced AES implementation on an Intel Core2, which achieves a throughput of 9.2 clock cycles per byte for a data block longer than 2048 bytes, equaling 1.85 Gbps when the core is running at its maximum frequency of 2.13 GHz [3]. The bitslice technique was first proposed by Biham for fast DES implementation on a software platform with a word size longer than 16 bits [4]. Bernstein et al. investigated the opportunities of reducing instruction count by combining different instructions together for various architectures, which alleviates the dependence between throughput and the length of data block [5]. Both bitslice and specific sets of instructions from SSSE3 (Supplemental Streaming SIMD Extensions 3 [6]) are utilized to enhance the performance of the Intel Core i7 920 as high as 6.92 clock cycles per byte [7]. Besides CPUs, there is also a trend to use GPUs (Graphic Processing Units) and DSP processors to implement AES encryption. Wollinger et al. compared different encryption algorithms on a TMS320C6X processor and achieved a AES encipher with a throughput of 14.25 cycles per byte [8]. 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 [9].

This paper presents a high-performance and area-efficient AES cipher on a fine-grained many-core system. By exploiting the advantages of fine-granularity data and task parallelism, the proposed implementation achieves good performance and higher throughput per area compared to other software platforms. The remainder of this paper is organized as follows. Section II introduces the AES algorithm and the features of the targeted many-core platform. Section III describes the proposed AES implementations in terms of partitioning, mapping and optimization. Section IV compares our work with other software implementations. Finally, Section V concludes the paper.

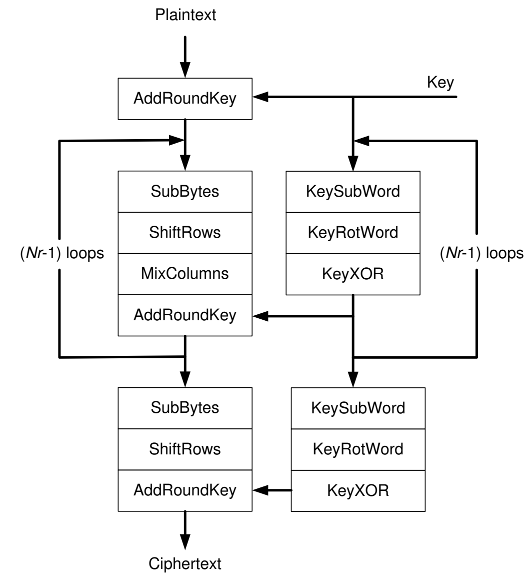

Fig. 1. Block diagram of AES encryption

II. AES and The Targeted Many-core Architecture

A. Advanced Encryption Standard

AES is a symmetric encryption algorithm, which takes a 128-bit data block as input and performs several rounds of transformations to generate output ciphertext. Each 128-bit data block in the AES is processed as a 4-by-4 array of bytes, called *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 the key lengths of 128, 192 or 256 bits, respectively. Fig. 1 shows the AES encryption steps with the key expansion process. In this paper, we focus our discussion on the situation with a 128-bit key and N_r = 10. In the case of encryption, there are four different 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: S-box). The S-box is computed based on a multiplicative inverse in the finite field GF(2^8) and the bitwise affine transformation:

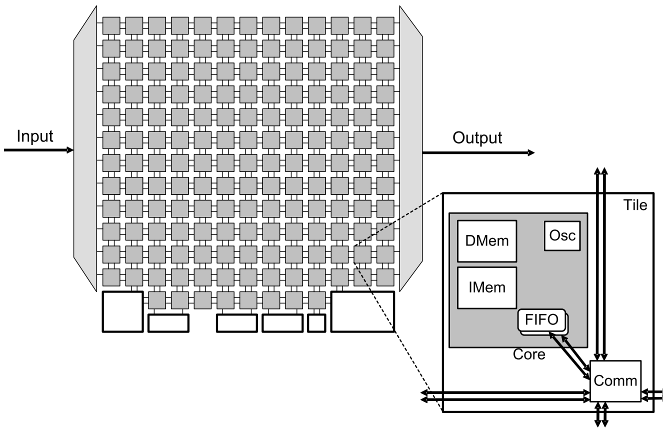

Fig. 2. Block diagram of the targeted many-core platform [10].

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (1)$$

where $\begin{bmatrix} b_7b_6b_5b_4b_3b_2b_1b_0 \end{bmatrix}$ and $\begin{bmatrix} b'_7b'_6b'_5b'_4b'_3b'_2b'_1b'_0 \end{bmatrix}$ represnt the bytes before and after substituition, respectively.

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 a polynomial over GF(2^8). After multiplying modulo x^4+1 with a fixed polynomial $a(x)$,

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

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 from the key expansion process.

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 accordin gto 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.

B. Fine-grained Many-core Processor Array

The targeted Asynchronous Array of Simple Processors (AsAP) architecture is an example of fine-grained many-core computation platforms, supporting globally-asynchronous locally-synchronous (GALS) clocking on-chip network and per-processor dynamic voltage and frequency scaling (DVFS) [11].

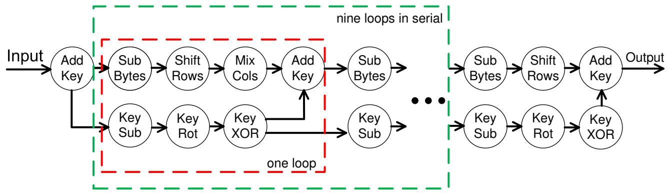

Fig. 3. Dataflow diagram of the loop-unrolled AES

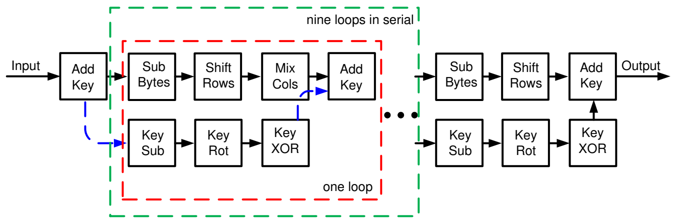

Fig. 4. Mapping diagram of the proposed 70-core AES cipher on AsAP.

Fig. 2 shows the block diagram of AsAP. The computation platform is composed of 164 small identical processors, three hardware accelerators and three 16 KB shared memories. All processors and shared memories are connected by a reconfigurable 2D-mesh network that supports both nearby and long-distance communication [12]. Each programmable processor has a simple 6-stage pipeline, which issues one instruction per cycle. Moreover, no specialized instructions are added. Each processor has a 128×32-bit instruction memory, a 128×16-bit data memory and two 64×16-bit FIFOs. The area of every processor occupies 0.17 mm^2 in 65 nm CMOS technology. Each processor can operate at a maximum clock frequency of 1.2 GHz at 1.3 V.

III. Proposed AES Implementations

A. Preliminary Design – Partitioning and Dataflow Mapping

As Fig. 1 shows, the 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 N_r-1 times, which equals nine in our design. The dataflow diagram after loop-unrolling is shown in Fig. 3.

Fig. 4 shows a preliminary AES cipher implementation based on the dataflow diagram. Each task in the dataflow 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 encipher is:

$$\begin{aligned} N_{processors} &= (N_r-1)\times N_{one-loop} + N_{last-round} \\ &= 9\times7+7=70 \end{aligned} \quad (3)$$

Fig. 5 and Fig. 6 summarize the instruction and data memory usages for each processor in the original design, respectively. Each processor in the 70-core AES cipher uses an average of 28 words of 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.

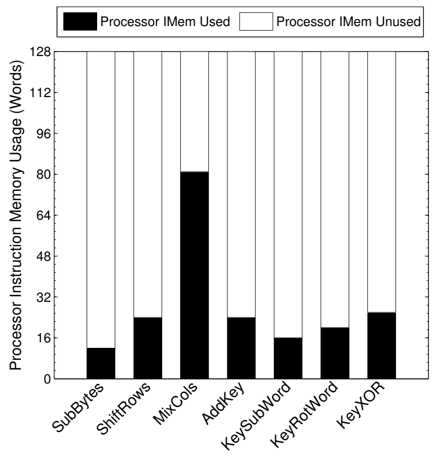

Fig. 5. Instruction memory usage of the proposed 70-core implementation.

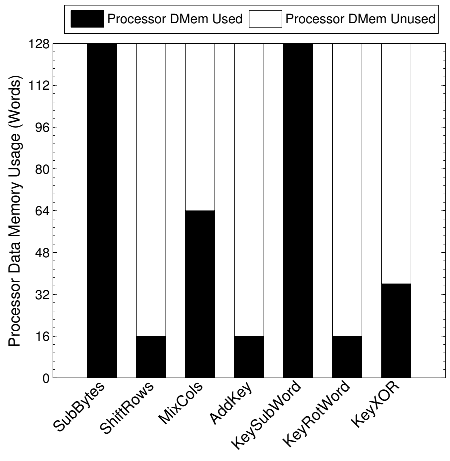

Fig. 6. Data memory usage of the proposed 70-core implementation.

B. Preliminary Design – Throughput Evaluation

To evaluate the throughput of the design, the execution time of each processor is examined separately while it runs alone without connected with other processors. This evaluation shows the pure execution time of each processor without any stall on input or output caused by other processors. Table I presents the execution time of each processor when processing one 128-bit data block with a 128-bit round key. As shown, the *MixColumns* processors are the bottlenecks of the system, which is at least twice slower than other processors. By running the cipher on the targeted many-core platform, we can obtain that the throughput of our design is 266 clock cycles per data block, which equals 16.625 clock cycles per byte.

A more detailed analysis of the processor execution activity for encrypting one 128-bit data block is shown in Fig. 7. The activity of each processor (the amount of time spent executing, rather than stalling) is indicated by the black bar in the figure. The white bars indicate the time stalled on output, while the gray bars indicates the time spent waiting for input to arrive. Fig. 7 shows that the *MixColumns* processors are running all the time and force other processors on the critical path to stall either on output while sending data or input while receiving data. Therefore, the *MixColumns* processors

TABLE I
EXECUTION TIME OF ALL PROCESSORS FOR PROCESSING ONE 128-BIT DATA BLOCK WITH A 128-BIT ROUND KEY ON ASAP

Processor Name	Execution Time (clock cycles)
SubBytes	132
ShiftRows	38
MixColumns	266
AddRoundKey	22
KeySubWord	56
KeyRotWord	26
KeyXOR	56

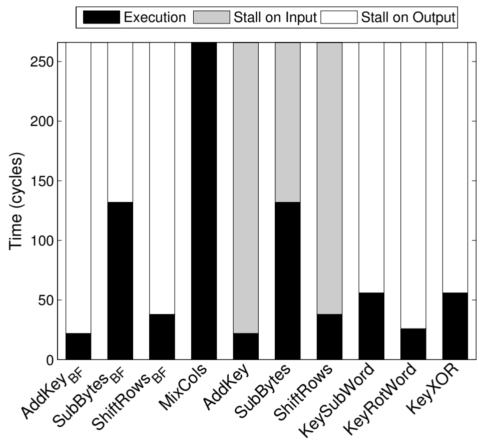

Fig. 7. Overall activity of processors while processing a 128-bit data block in the proposed 70-core cipher.

are the bottlenecks of our design. The $AddKey_{BF}$,$SubBytes_{BF}$ and $ShiftRows_{BF}$ represent the processors before the first *MixColumns* processor. They stall on output for more than 50% of the time because the downstream *MixColumns* processor is not fast enough to consume their outputs. On the other hand, the *AddKey, SubBytes* and *ShiftRows* processors after the first *MixColumns* processor stall on their input due to the slower data rate of the upstream *MixColumns* processor.

C. Design Optimization I – Increasing Throughput

Throughput is one of the most significant metrics in our design, since most of applications have critical requirements on the encryption data rate. Based on the previous analysis, the *MixColumns* processors limit the throughput of the cipher. On a many-core platform, the most convenient solution for increasing throughput is to parallelize the *MixColumns* operation into multiple processors. Since the *MixColumns* operations on each column of a data block are independent (each data block is a 4-by-4 array of bytes), each *MixColumns* processor can be replaced by two *MixCol-8* processors. Then, each *MixCol-8* computes only two columns rather than a whole data block. As a result, the delay of the system's bottleneck is reduced to approximately half of the preliminary design. Simulation results show that each *MixCol-8* processor requires 152 clock cycles for processing one data block, and is the new bottleneck of the optimized cipher. Therefore, the throughput of the cipher is increased by approximately 43% to 152 clock cycles per data block, equaling 9.5 clock cycles per byte. However, this optimization also brings a requirement of 10 more processors to implement the cipher.

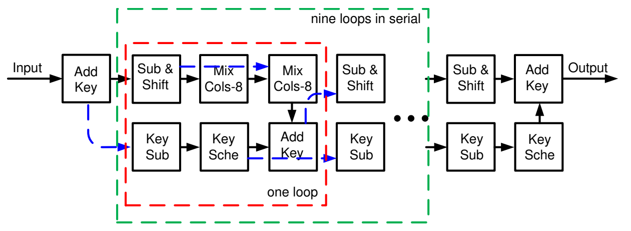

Fig. 8. Mapping diagram of the optimized 59-core AES cipher on AsAP.

D. Design Optimization II – Reducing Processors

Besides throughput, area is another significant metric in system design. Less area means less silicon, therefore less cost. From a many-core processor perspective, area is represented by the number of cores required to implement applications. Smaller area translates into fewer cores and leaves more opportunities for dealing with other applications on the same platform simultaneously. Therefore, besides enhancing the throughput of our design, we also try to reduce the number of cores used for the implementation without impairing the performance.

The small usage of memory (shown in Fig. 5 and Fig. 6) and the large slack of most processors (shown in Fig. 7) offer us an opportunity to merge different processors together to reduce the area of our design without sacrificing throughput. Two processor fusion steps are applied to the previous design. First, we merge the adjacent *SubBytes* and *ShiftRows* processors to form a new processor called *SubShift*. The *SubShift* takes 148 cycles for processing one data block, and occupies 102 words of instruction memory and 128 words of data memory, respectively. Second, the neighboring *KeyRot* and *keyXOR* processors are fused into one *KeyScheduling* processor, which takes 60 cycles to expand a 128-bit key, requires 31 words of instruction memory and 36 words of data memory, respectively.

The execution time of both the *SubShift* and *KeyScheduling* processors are less than 152 cycles per data block, which means that the above processor fusion steps for processor count reduction do not create any new bottlenecks for the design. Fig. 8 shows the AES cipher after throughput and area optimization. Each loop operation in the AES algorithm requires six small processors. As a result, the optimized cipher requires 59 cores in 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 $AddKey_{BF}$ and $SubShift_{BF}$, are stalled on their output. While the processors after the first *MixCol-8* processor, the *AddKey* and *SubShift*, are stalled on their input. In summary, compared with the original design, the optimized cipher achieves a 43% higher throughput (9.5 against 16.625 cycles per byte) and requires 16% fewer processors (59 instead of 70).

IV. Related Work and Comparison

A comprehensive comparison of the state-of-the-art software implementations of AES are summarized in Table II. In order to make a fair comparison, all of the data are scaled to 65 nm CMOS technology. The area is scaled to 65 nm with a $1/s^2$ reduction, and the delay is scaled with a $1/s$ reduction according to full-scaling rules [13].

We use the metric throughput (cycles/byte) and throughput per area (Mbps/mm^2) to compare the performance and area efficiency of various implementations. As shown in Table II, compared to the highly optimized AES ciphers on CPUs with the bitslicing method [3], the proposed AES design on AsAP can achieve up to 1.7 times higher throughput and 3.6–9.2 times higher throughput per chip area. Besides bitslice, special SIMD instructions are applied to improve the throughput and efficiency of the AES implementation on CPUs further [7]. Even so, our design on AsAP still shows a comparable throughput, with 8.3–10.7 times higher throughput per area. The TI DSP C6201 is an 8-way VLIW architecture for high performance DSP applications. The referenced data shows that our design has 1.5 times higher throughput. The area of the TI DSP C6201 is not available, but we believe that our design has a much higher throughput per area.

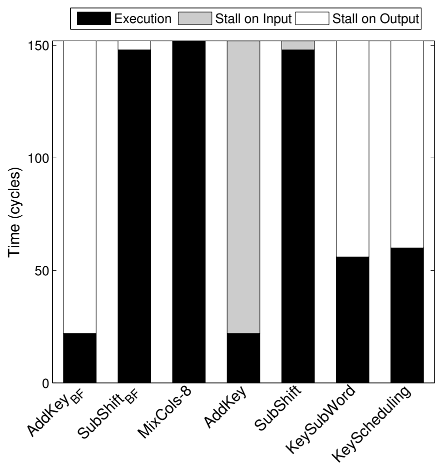

Fig. 9. Overall activity of processors while processing a 128-bit data block in the optimized 59-core cipher.

The AES implementation on GeForce 8800 GTX achieves the highest throughput in our referenced designs [9]. This advantage comes from the application of the T-Box method, which combined the *SubBytes, ShiftRows* and *Mixcolumns* phases into one single look-up table [14]. This method is highly optimized for SIMD architectures with large memory. However, the design on AsAP still shows 3.4 times better area efficiency. It implies that our design would achieve a 3 times higher throughput, if we map our design multiple times on a AsAP chip, which has the same die area as GeForce 8800 GTX.

V. Conclusion

In this paper, a high-performance area-efficient AES cipher is implemented on a many-core platform. The preliminary cipher is partitioned and mapped to an array of 70 small processors compliant with the algorithm steps in AES, and has a throughput of 16.625 clock cycles per byte. By balancing workload between different processors, and merging processors with small usage of instruction memory and data memory, the throughput of the cipher is increased by 43% to 9.5 clock cycles per byte; while the number of processors utilized for the design is reduced to 59, which is 10.03 mm^2 in a 65 nm fine-grained many-core system. In comparison with other AES implementations on state-of-art general purpose processors, the proposed cipher on AsAP achieves 3.6–10.7 times higher throughput per area. Additionally, our design shows 1.5 times higher throughput than the TI DSP C6201 and 3.4 times higher throughput per area than the GeForce 8800 GTX.

TABLE II
COMPARISON OF AES CIPHER IMPLEMENTATIONS ON DIFFERENT SOFTWARE PLATFORMS. THE ORIGINAL DATA ARE PRESENTED WITH DIFFERENT CMOS TECHNOLOGIES. FOR COMPARISON, AREA AND DELAY NUMBER ARE SCALED TO 65 NM TECHNOLOGY. ASSUMING A $1/(s^2)$ REDUCTION IN AREA, A $1/s$ REDUCTION IN DELAY BASED ON FULL-SCALING RULES.

Platform	Method	Technology (nm)	Area (mm^2)	Max Freq. (MHz)	Throughput (cycles/byte)	Scaled Throughput (Mbps)	Scaled Area (mm^2)	Scaled Throughput/Area ($Mbps/mm^2$)
Pentium 4 561 [3]	Bitslice	90	112	3600	16	2492	58.42	42.66
Athlon 64 3500 [3]	Bitslice	90	193	2200	10.6	2299	101	22.76
Core2 Duo E6400 [3]	Bitslice	65	111	2130	9.19	1854	111	16.70
Core2 Quad [a] Q6600 (one core) [7]	Bitslice + SSSE3	65	286/2 = 143	2400	9.32	2060	143	14.41
Core2 Quad [a] Q9550 (one core) [7]	Bitslice + SSSE3	45	214/4 = 53.5	2830	7.59	2065	112	18.44
Core i7 [a] 920 (one core) [7]	Bitslice + SSSE3	45	263/4 = 65.75	2668	6.92	2135	133	16.05
TI C6201 [8]		180	NA	200	14.25	311	NA	NA
GeForce 8800 GTX [9]	T-box	90	484	575	NA	11500	252	45.63
This Work AsAP	key expan. cores are not included	65	6.63	1210	9.5	1019	**6.63**	**153.70**

[a] The throughput results from [7] are for only one core, so the area numbers are also scaled to one core.
[b] All referenced designs do not consider key expansion, thereby the cores used for key expansion of the AES implementation on AsAP are not included for a fair comparison.

VI. 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. The authors also acknowledge the support of the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

REFERENCES

[1] NIST, "Advanced encryption standard (AES)," http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, Nov. 2001.
[2] NIST, "Data encryption standard (DES)," http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, Oct. 1999.
[3] 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.
[4] Eli Biham, "A fast new DES implementation in software," in *Fast Software Encryption*, vol. 1267 of *Lecture Notes in Computer Science*, pp. 260–272. 1997.
[5] 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.
[6] "Supplemental streaming SIMD extensions 3," http://en.wikipedia.org/wiki/SSSE3.
[7] 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.
[8] 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.
[9] 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.
[10] D. Truong, W. Cheng, T. Mohsenin, Zhiyi Yu, T. Jacobson, G. Landge, M. Meeuwsen, C. Watnik, P. Mejia, Anh Tran, J. Webb, E. Work, Zhibin Xiao, and B. Baas, "A 167-processor 65 nm computational platform with per-processor dynamic supply voltage and dynamic clock frequency scaling," in *VLSI Circuits, 2008 IEEE Symposium on*, Jun. 2008.
[11] 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.
[12] 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.
[13] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic, *Digital Intergrated Circuits – A Design Perspective*, Prentice-Hall, New Jersey, NJ, second edition, 2003.
[14] J. Daemen and V. Rijmen, *The Design of Rijndael*, Springer-Verlag, New York, NY, 2002.