Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s AES Processors

Alireza Hodjat, *Student Member*, *IEEE*, and Ingrid Verbauwhede, *Senior Member*, *IEEE*

Abstract—This paper explores the area-throughput trade-off for an ASIC implementation of the Advanced Encryption Standard (AES). Different pipelined implementations of the AES algorithm as well as the design decisions and the area optimizations that lead to a low area and high throughput AES encryption processor are presented. With loop unrolling and outer-round pipelining techniques, throughputs of 30 Gbits/s to 70 Gbits/s are achievable in a 0.18-μm CMOS technology. Moreover, by pipelining the composite field implementation of the byte substitution phase of the AES algorithm (inner-round pipelining), the area consumption is reduced up to 35 percent. By designing an offline key scheduling unit for the AES processor the area cost is further reduced by 28 percent, which results in a total reduction of 48 percent while the same throughput is maintained. Therefore, the over 30 Gbits/s, fully pipelined AES processor operating in the counter mode of operation can be used for the encryption of data on optical links.

Index Terms—Advanced Encryption Standard (AES), cryptography, crypto-processor, security, hardware architectures, ASIC, VLSI.

1 Introduction

Optical networks require secure data transmission at rates over 30 Gbits/s. In one application [1], the optical switches require cryptographically secure random numbers to generate the encrypted stream of data. For this purpose, the Advanced Encryption Standard algorithm [2] designed for over 30 Gbits/s throughput is required to generate the sequences of random numbers. An encryption algorithm is never used stand-alone for security reasons. Therefore, it is combined with so-called modes of operation. One mode of operation, called the counter mode [3], is suitable for high throughput applications. It is used to produce high throughput cryptographically strong pseudorandom numbers. It has the advantage that the encryption algorithm can be pipelined because there is no feedback in this mode of operation. Fig. 1 shows the details of the AES algorithm in the counter mode of operation. In the case of a keystream generation with the counter mode, the initial 128-bit seed is loaded into the seed register (initial value of the counter register). Every clock cycle, the counter value is incremented and loaded into the AES unit to be encrypted. The encrypted value is the generated pseudorandom number. Starting from a secure nonrepeating initial seed, a sequence of a maximum of 2^{128} strings of 128-bit random numbers is generated.

This paper presents the area-throughput trade-offs of a fully pipelined, ultra high speed AES encryption processor. Different pipelined architectures that can achieve the required throughput for the above application and the area optimization opportunities for such designs are explored. Loop unrolling and inner and outer round pipelining of the AES algorithm are techniques that can help us to achieve the throughput of 30 to 70 Gbits/s using a 0.18-μm CMOS technology.

The rest of this paper is organized as follows: Section 2 will investigate the related work. In Section 3, the ultra high speed Advanced Encryption Standard algorithm is presented. The different steps of the AES algorithm and the design considerations that will lead to a high throughput design are described. Section 4 shows how the area efficient byte substitution phase of the AES algorithm can be implemented without any loss in speed and throughput. Section 5 presents performance trade-offs for the high throughput AES processor with area efficient byte substitution and on-the-fly key-scheduling. In Section 6, the architecture for the offline key scheduling is presented and the area-throughput trade-offs for the AES implementation with offline key scheduling unit is explored. Finally, Section 7 provides the conclusion of this paper.

2 Related Work

The Advanced Encryption Standard was accepted as a FIPS standard in November 2001 [2]. Since then, there have been many different hardware implementations for ASIC and FPGA. References [4], [5], [6], [7] are some of the early implementations of the Rijndael algorithm before it was accepted as the Advanced Encryption Standard [8]. Reference [9] is the first ASIC implementation of the Rijndael on silicon. Other ASIC implementations are [10], [11], [12]. These references mainly focus on area efficient implementation of the AES algorithm using Sbox (byte substitution phase) optimizations. They all use the suggestion of Rijmen [13], one of the inventors of the Rijndael algorithm. He suggested a way of optimization of the Sboxes based on transforming the original field of $\mathrm{GF}(2^8)$ to a composite field of $\mathrm{GF}((2^4)^2)$ or $\mathrm{GF}(((2^2)^2)^2)$ by isomorphic mapping. These all focus on area efficient implementations of one round of the AES algorithm without pipelining and,

- *A. Hodjat is with Broadcom Corporation, 16215 Alton Parkway, PO Box 57013, Irvine, CA 92619-7013. E-mail: alirezah@broadcom.com.*
- *I. Verbauwhede is with the Faculty of Engineering, Department of Electrical Engineering-ESAT/COSIC, Computer Security and Industrial Cryptography, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium. E-mail: Ingrid.Verbauwhede@esat.kuleuven.ac.be.*

Manuscript received 13 Dec. 2004; revised 1 June 2005; accepted 27 Sept. 2005; published online 22 Feb. 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0412-1204.

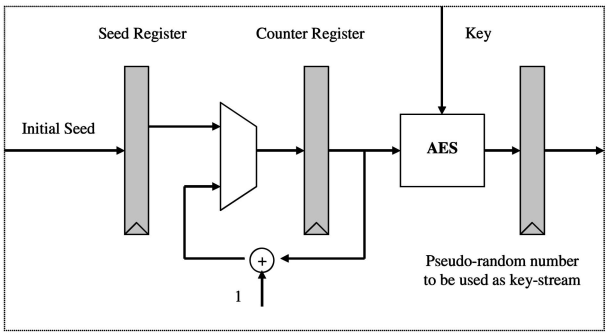

Fig. 1. AES in the counter mode of operation.

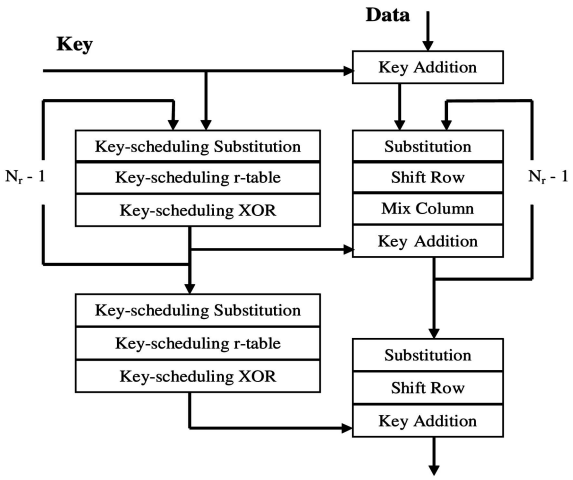

Fig. 2. Advanced Encryption Standard.

therefore, they can only provide a throughput rate of between 2 to 3 Gbits/s. Only reference [14] can achieve the throughput of 10 Gbps by implementing binary decision diagram (BDD) circuit architecture and TBoxes, which are the combination of Sboxes and the mix-column phase of the AES algorithm. On the other hand, there are several implementations for FPGA that can achieve a throughput rate of 1 to 20 Gbits/s because they unroll the encryption rounds and use pipelining. However, they don't use the composite field implementation of the Sboxes and mostly use look-up table implementations of the byte substitute phase and map them on the Block RAMs of the FPGA. Moreover, reference [15] gives a review of possible implementation strategies for hardware implementations of the AES algorithm. However, it does not publish implementation results. Our work is different from this reference because we present different implementation results (throughput and area cost).

None of the previously published AES implementations present a throughput rate of over 30 Gbits/s. In this paper, the possibilities of achieving a throughput of over 30 Gbits/s encryption using the AES algorithm with minimum area cost is explored. Our original contribution consists of combining the pipelining techniques with a composite field implementation. The architectures that are addressed in this paper can achieve the above throughput rate with a reduction up to 48 percent in the area cost compared to a straightforward pipelined design of the AES algorithm. Our approach is to unroll the round loop and use pipelining inside and between each round. Then, by designing an area efficient pipelined implementation of Sboxes using composite field implementation, the area is reduced significantly.

3 Advanced Encryption Standard

Fig. 2 shows the different steps of the AES algorithm [2]. The AES algorithm is performed in N_r number of rounds. The architecture of one round contains two different datapaths, the encryption datapath and the key scheduling datapath. In the AES algorithm, the data block is 128 bits long and the key size can be 128, 192, or 256 bits. The size of the key defines the number of rounds that the algorithm is repeated. The value N_r is equal to 10, 12, or 14 for the key length of 128, 192, or 256 bits, respectively.

There are different steps in each round of the encryption datapath. These are substitution, shift row, mix column, and key addition. The byte substitution step is a nonlinear operation that substitutes each byte of the round data independently according to a substitution table. Shift row step is a circular shifting of bytes in each row of the round data. The number of shifted bytes is different for each row. In the Mix column step, the bytes of each column are mixed together. This is done by multiplying the round data with a fixed polynomial modulo $x^4 + 1$. In the Add key step, the round data is XORed with the round key. All four of the above steps are required for every round except that the last round does not include the mix column phase. Similar steps are followed in the key scheduling flow. Except for the byte substitution phase, most of the operations in the AES algorithm are implemented using a chain of XORs. Byte substitution is the most critical part of this algorithm in terms of performance. The most efficient implementation of this phase will be discussed in the next section. The details of the 128 bit state representation and the different steps can be found in [2].

In order to achieve an ultra high speed implementation of the AES algorithm, there are a number of design decisions taken as follows:

1. For key lengths larger than 128 bits, the critical path of the AES algorithm sits in the key scheduling datapath. Therefore, in order to have a balanced critical path for both encryption and key scheduling datapaths, a key length of 128 bits long is used. This way the critical path is shorter and is in the encryption flow. Moreover, the number of rounds, N_r, will be fixed to 10 rounds.
2. For an ultra high speed design, the AES iteration loop has to be unrolled. If the datapath is shared for different rounds of the algorithm, then the throughput will significantly decrease. The highest possible throughput is achieved when one output sample is generated every clock cycle. This is possible only when the loop is unrolled and pipelining is applied.
3. Pipelining can be applied both for inside each round and around each round. Inner round pipelining will be presented in the next section. For outer round pipelining, the pipeline registers will be placed

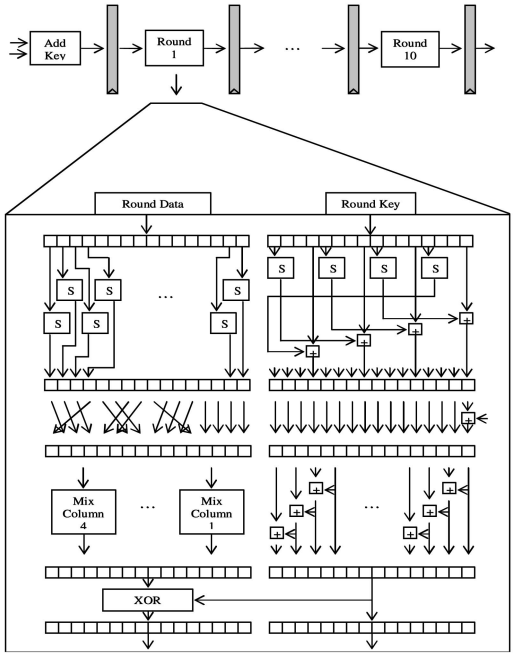

Fig. 3. Outer round pipelining for the AES algorithm.

between the datapath instances of each round. Fig. 3 shows the outer round pipelined implementation of the AES algorithm. There is one pipeline stage for each round and the key schedule is calculated on the fly.

4. The byte substitution phase is the slowest operation in the AES algorithm. In this phase, every byte of the data is substituted with another byte. The new byte is calculated using a nonlinear operation in $GF(2^8)$. There are two well-known implementations for the substitution phase of the AES algorithm. One approach is the direct implementation of the substitution boxes using lookup tables because all 256 cases of the substitution bytes can be precomputed and can be stored in a lookup table. The other approach is to use the $GF(2^4)$ operations to calculate the substitution value on the fly [11]. The former has minimum delay, but consumes a huge amount of area. The latter has efficient area consumption, but it has a long critical path [9]. The next section shows how we can achieve a minimum area and maximum throughput using a pipelined implementation of the second approach.

4 Area Efficient Byte Substitution

In the byte substitution phase, the input is considered as an element of $GF(2^8)$. First, the multiplicative inverse in $GF(2^8)$ is calculated. Then, an affine transformation over GF(2) is applied [2]. Since there are only 256 representations of one byte, all the byte substitution results can be calculated before hand. In this case, the implementation of an Sbox (substitution box) can be done by a look-up table, as shown in Fig. 4a. On the other hand, we can implement an Sbox using Galois Field operations. Calculating the multiplicative inverse of elements in $GF(2^8)$ is very expensive. The inventors of the AES algorithm suggest an algorithm that calculates the multiplicative inversion in $GF(2^8)$ using the $GF(2^4)$ operations [13]. Also, reference [11] presents one implementation of such an algorithm. This is the composite field implementation of the byte substitution phase. Fig. 4b shows how the complete Sbox can be designed using $GF(2^4)$ operations. We call it a nonpipelined Sbox using GF operators. Here, the input byte (element of $GF(2^8)$) is mapped to two elements of $GF(2^4)$. Then, the multiplicative inverse is calculated using $GF(2^4)$ operators. Then, the two $GF(2^4)$ elements are inverse mapped to one element in

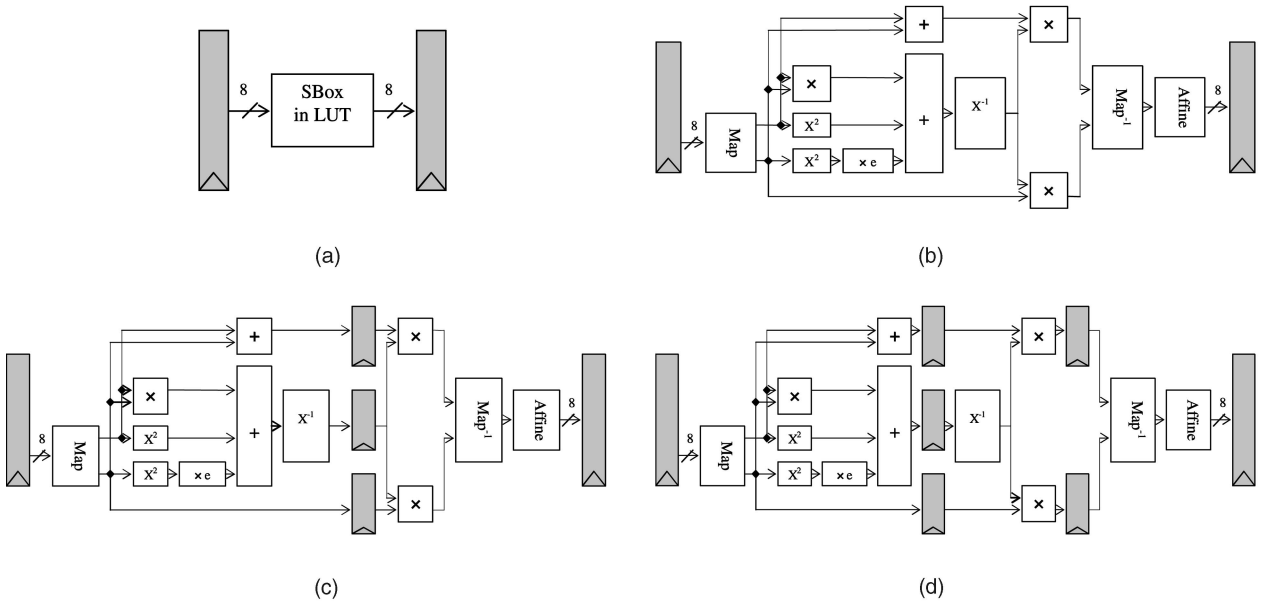

Fig. 4. Byte substitution architectures. (a) Sbox using LUT implementation. (b) Nonpipelined Sbox using GF operations. (c) Two-stage pipelined Sbox using GF operations. (d) Three-stage pipelined Sbox using GF operations.

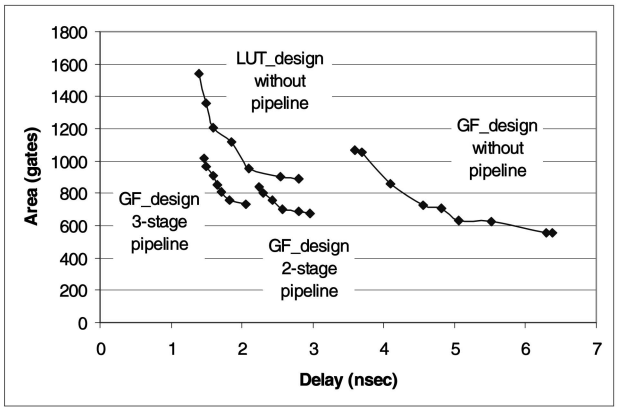

Fig. 5. The area-delay trade-off for the Sbox.

$GF(2^8)$. In the end, the affine transformation is performed. Notice that an inversion in $GF(2^4)$ can be efficiently implemented using look-up tables because there are only 16 possibilities for four bits. The details of the $GF(2^4)$ operators are from reference [11].

Although the composite field implementation of the Sbox is very area efficient, it suffers from a long critical path. This will reduce the overall throughput significantly. To overcome this drawback, further pipelining can be used. This is the inner round pipelining for the AES algorithm because the pipeline registers will appear inside of the byte substitution phase, which is inside of each round. Pipelining inside the Sbox will increase the number of registers used in the whole design and, therefore, the area can increase if the pipeline registers are not in the right place. Fig. 4c shows the two-stage pipelined implementation of an Sbox using GF operators. The critical path is broken in half and there are only three 4-bit registers. Fig. 4d shows the three-stage pipelined implementation of an Sbox using GF operators. The critical path is divided into three stages. Notice that the first pipeline stage is after the addition operation because it saves area. The addition operation could be part of the second pipe stage, but that would double the number of registers that are necessary for the first pipeline stage. Therefore, this way fewer registers are used and area is saved. Please note that the other operations, shift row, mix column, and key addition remain together in one pipeline stage.

Fig. 5 shows the area-delay trade-off of the different implementations of the byte substitution phase that were presented in Fig. 4. As was stated before, these results are for a 0.18-μm CMOS standard cell library with conservative wire load model. As seen in Fig. 5, the pipeline composite field implementation of the Sbox saves area, while it has the same critical path delay as the look-up table implementation. Depending on the required speed (throughput), either a two or three pipeline stages implementation should be used. The area cost of one Sbox using two-stage composite field implementation is 23 percent less than the LUT design with the same speed. For a three-stage composite field implementation, this cost is 32 percent less than the corresponding same-speed LUT design.

Obtaining a high throughput Sbox implementation is done by means of two combined techniques. The first one is

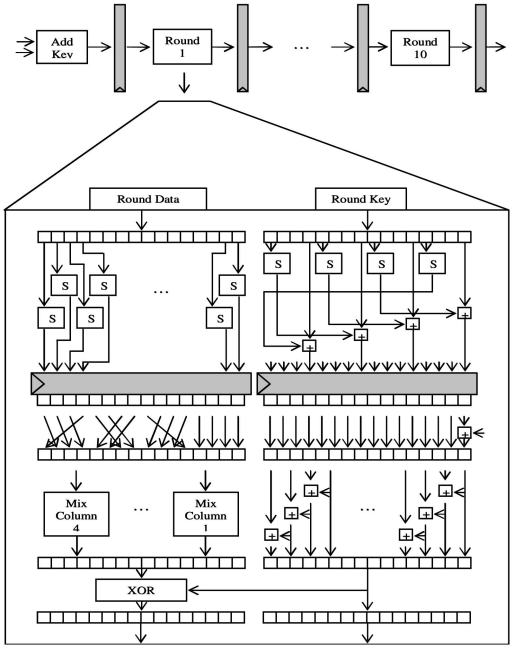

Fig. 6. AES with inner and outer round pipelining.

the selection of the field: $GF(2^8)$ or $GF(2^4)$. $GF(2^4)$ is traditionally chosen for its compact realization. This is, e.g., the case for reference [10] and it is also stated in [15]. However, it can also be combined with pipelining for high throughput. This has not been done before and is an original contribution of this paper. The $GF(2^8)$ is faster if no pipelining is used, but it cannot be pipelined. The second technique is within the logic level synthesis tool (Synopsys DC compiler in our case): The tool can be pushed for speed or for area. This gives a variation of design points, as shown by the individual curves on the figures. For instance, in Fig. 5, the delay of the slowest pipeline stage of the Sbox is plotted against the Area. Within one design option, corresponding to one curve, the area will go up if the synthesis tool is pushed for speed.

5 Performance Trade-Offs for the AES with On-the-Fly Key-Scheduling

This section presents the throughput-area trade-off of our high speed AES processor which includes an online (on-the-fly) key scheduling unit. Fig. 6 shows the inner and outer round pipelined architecture for the encryption and key scheduling datapath. In this figure, for the inner round pipelining, the inside of each round is divided into two pipeline stages. The first stage is the byte substitution phase and the second one includes the rest of the steps in each round, which are shift-row, mix column, and key addition. In the first experiment, the look-up table implementation of the byte substitute phase is used. We call this design the AES architecture with two pipeline stages per round and online key scheduling. It can be clocked with really high clock frequencies, but the area cost is very high. For area optimization, the conclusion of Section 4 can be used.

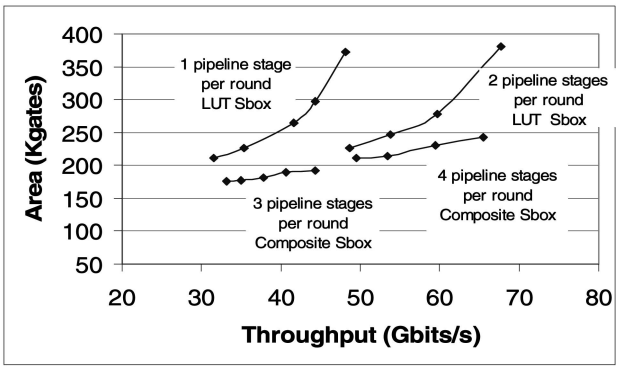

Fig. 7. The throughput-area trade-off of the AES processor with online key scheduling.

In Section 4, it was shown that the two or three stages pipelined implementation of an Sbox using GF operations can reduce the area significantly. Therefore, these two implementations of the byte substitution phase can be used instead of the look-up table implementation of Sboxes in Fig. 6. When the Sbox with two pipeline stages (Fig. 4c) is used, each round of the AES algorithm will have three pipeline stages. When the Sbox with three pipeline stages (Fig. 4d) is used, there are four pipeline stages inside of each round. These are the most area efficient AES implementations with online key scheduling, achieving a throughput between 30 to 70 Gbits/s. There is a significant area reduction in these two designs compared to the two stages pipelined architecture that uses the look-up table implementation for the Sbox.

Fig. 7 shows the throughput-area trade-off of the proposed architectures of the AES processor with online key scheduling. Four different pipelined architectures are compared together. The architecture with one pipeline stage per round with LUT Sbox is the implementation of Fig. 3. The design with two pipeline stages per round with LUT Sbox is the implementation of Fig. 6. The architecture with three pipeline stages per round with composite Sbox is the implementation of Fig. 6 when the Sbox of Fig. 4c is used. Similarly, the design with four pipeline stages per round with composite Sbox is based on Fig. 6 when the Sbox of Fig. 4d is used.

Using the clock frequency and the number of clock cycles for one encryption, the throughput is calculated. For example, in the cases where one encrypted output is generated every clock cycle and the data is 128 bits long, the throughput is calculated by 128 * Clock Frequency. In each figure, each point refers to the area cost of the design and its maximum throughput. Therefore, by pushing the synthesis tool for more speed, we can achieve higher throughput, while the result costs more area. The main point of this paper is to explore the area-throughput trade-off for the AES algorithm so that the designer can pick the best choice depending on his application.

Fig. 7 shows that the inner and outer round pipelined architectures of the AES algorithm that use the pipelined architectures of the composite Galois Field implementation of the byte substitution phase can produce the throughput rate from 30 to 70 Gbits/s in much smaller area compared to the architectures that use the LUT-based implementation of Sboxes. The area cost for the architecture with three pipelined stages per round can be up to 35 percent less than the design with LUT Sbox implementation. Moreover, the architecture with four pipeline stages per round can cost up to 30 percent less area than the design with LUT Sbox implementation. Also, the area cost does not vary much when the design is synthesized for higher clock frequencies.

The inner round pipelining of the AES algorithm reduces the area while the same throughput is maintained, but the cost is an increase in latency. Latency is defined by the number of cycles that each data sample takes to go through the encryption datapath before the encrypted output is generated. When there is only outer round pipelining (one stage pipeline per round), the latency is 11 cycles. In the design with two pipeline stages per round, the latency is 21 cycles. For the fully inner and outer round pipelined designs with three or four pipeline stages per round, the latencies are 31 and 41 cycles, respectively. In our application, latency is not important and the main concern is throughput; therefore, we can gain much by defining inner round pipeline stages.

6 Performance Trade-Offs for the AES with Offline Key-Scheduling

In most encryption applications, the encryption key does not vary as frequently as data. More specifically, in our application, over 30 Gbits/s throughput is required for the optical link during each session. The key schedule is calculated for every session key and remains constant during the whole session. In this case, the online (on-the-fly) key scheduling datapath performs the same function for every input sample of data. Thus, there is further room for area optimization by calculating the key schedule of the AES algorithm offline. In this approach, for every session key, first the offline key scheduling unit calculates the required round keys for every round and stores them inside the round key registers. Then, the encryption datapath performs the AES algorithm on the input data samples and uses the stored round key values for the key addition phase. The most important reason that causes the area reduction is the following: Since, in our implementation, the AES round loop is unrolled, therefore there are an equal number of registers that store the value of round keys. Thus, by defining an offline key scheduling unit, there is no overhead in terms of the total number of required registers to store the round keys and no extra memory is required. On the other hand, in the offline key scheduling unit, there is no need to unroll the round loop of the key scheduling datapath and, therefore, only one round can be implemented. This way, the area is reduced significantly.

Fig. 8 shows the architecture of the offline key scheduling unit that is designed for our high speed AES processor. Fig. 8a shows the block diagram of the processor that includes the key scheduling controller, the offline key scheduling datapath, and the encryption pipeline. The *keysch_start* signal will activate the key scheduling unit. After 20 clock cycles the key schedule, which includes 11 128-bit round keys, is generated. Then, the *keysch_done*

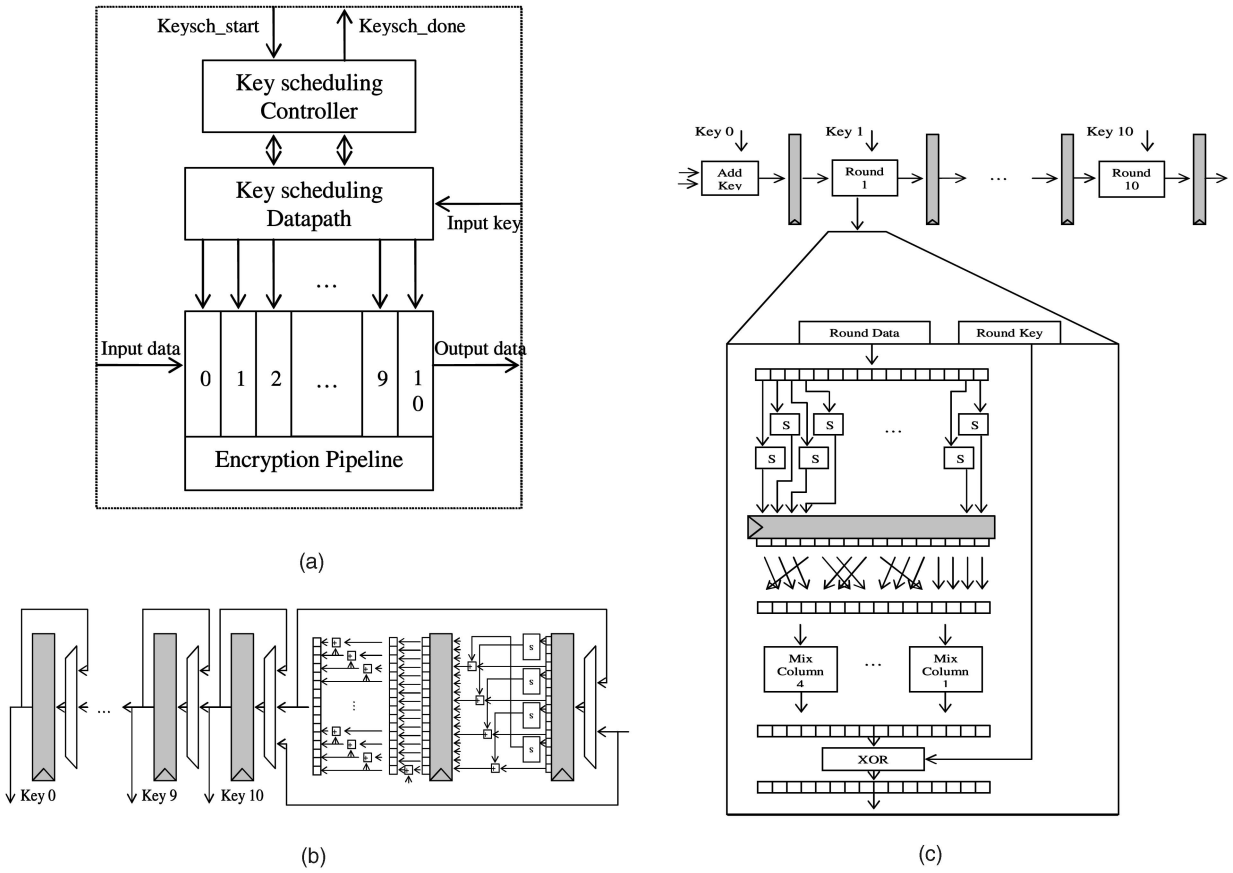

Fig. 8. (a) Block diagram of the processor with offline key scheduling unit. (b) The offline key scheduling datapath. (c) Encryption pipeline for the AES design with offline key scheduling.

signal is asserted, which indicates that the processor is ready to perform the encryption of the input data.

Fig. 8b shows the inside of the offline key scheduling datapath. One round of the key scheduling algorithm with two pipeline stages is designed in the feedback loop. Every two clock cycles, one round key is generated and is shifted to the round key registers. After a total of 20 cycles, all the round keys are calculated and stored in the round key registers. Notice that the Key_0 register will contain the input key for round 0.

Fig. 8c shows the new encryption pipeline, which does not include the key scheduling datapath. This is the unit that is used for the encryption pipeline block that is shown in Fig. 8a. Also notice that Fig. 8c is similar to the design in Fig. 6, with the difference that the online key scheduling datapath is removed. Following the same methodology that was mentioned in Section 5 for the choice of Sboxes, there will be three different AES implementations that will have two, three, or four pipeline stages inside each round of the algorithm. The two stage pipelined design uses the look-up table implementation of the Sboxes. The three stage pipelined design uses the pipelined Sbox implementation of Fig. 4c and the four stage pipelined design uses the pipelined Sbox implementation of Fig. 4d. All these architectures are synthesized using the 0.18-μm CMOS technology. The synthesis results show that the architectures with an offline key scheduling unit can further reduce the area up to 28 percent.

Fig. 9 shows the throughput-area trade-off of the proposed architecture of the high speed AES with offline key scheduling unit. When the offline key scheduling unit is used, the area cost for the architecture with three pipeline stages per round can be up to 37 percent less than the design with LUT Sbox implementation for the same speed. Moreover, the architecture with four pipeline stages per round can cost up to 33 percent less area than the same-speed design with LUT Sbox implementation. Therefore, by

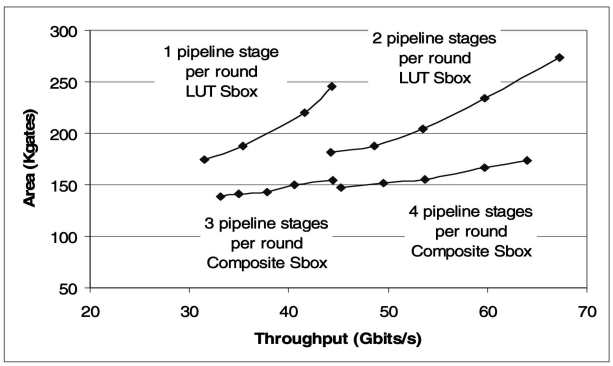

Fig. 9. The throughput-area trade-off of the AES processor with offline key scheduling.

using the pipelined implementation of the composite Galois Field implementation of the Sbox presented in Section 4 and the offline key scheduling unit that is presented in this section, the maximum area reduction of 48 percent is achieved for the high AES core that provides a throughput rate over 30 Gbits/s up to 70 Gbits/s.

7 Conclusion

This paper presents the architecture, synthesis results and the area-throughput trade-offs of different pipelined architectures of the AES algorithm. Area efficient architectures for fully pipelined high speed AES processors that can provide an encryption throughput of 30 to 70 Gbits/s for a 0.18-μm CMOS ASIC technology are presented. Loop unrolling and inner and outer round pipelining are used to reduce the critical path and increase the maximum throughput. By using a pipelined design of the composite field implementation of the byte substitution phase of the AES algorithm, the area is reduced up to 35 percent. Also, by designing an offline key scheduling unit for the high speed AES processor, an area reduction of an extra 28 percent is achieved. Therefore, the total area cost of the final architecture is reduced up to 48 percent without any loss in throughput. The area efficient AES architecture with throughput rate of over 30 Gbits/s is used in the counter mode of operation for the encryption of data streams in optical networks.

Acknowledgments

This material is based upon work supported by the Space and Naval Warfare Systems Center-San Diego under contract No. N66001-02-1-8938. This funding is gratefully acknowledged.

References

[1] H. Chan, A. Hodjat, J. Shi, R. Wesel, and I. Verbauwhede, "Streaming Encryption for a Secure Wavelength and Time Domain Hopped Optical Network," *Proc. IEEE Intl Conf. Information Technology (ITCC 2004),* Apr. 2004.

[2] US Nat'l Inst. of Standards and Technology, Advanced Encryption Standard, http://csrc.nist.gov/publication/drafts/dfips-AES.pdf, 2001.

[3] M. Dworkin, "Recommendation for Block Cipher Modes of Operations," SP 800-38A 2001, Dec. 2001.

[4] K. Gaj and P. Chodowiec, "Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard Using Field Programmable Gate Arrays," *Proc. Cryptographers Track RSA Conf. (CT-RSA 2001),* pp. 84-99, 2001.

[5] T. Ichikawa et al., "Hardware Evaluation of the AES Finalists," *Proc. Third AES Candidate Conf.,* Apr. 2000.

[6] K. Gaj and P. Chodowiec, "Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware," *Proc. Third Advanced Encryption Standard Candidate Conf. (AES3),* pp. 40-54, Apr. 2000.

[7] V. Fischer, "Realization of the Round 2 Candidates Using Altera FPGA," *Comments Third Advanced Encryption Standard Candidates Conf. (AES3),* Apr. 2000.

[8] Nat'l Inst. of Standard and Technology Web site, http://www.nist.gov/aes/, 2006.

[9] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and Performance Testing of a 2.29 Gb/s Rijndael Processor," *IEEE J. Solid-State Circuits (JSSC),* Mar. 2003.

[10] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," *Proc. ASIACRYPT 2001,* pp. 239-254, 2001.

[11] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC Implementation of the AES Sboxes," *Proc. RSA Conf. 2002,* Feb. 2002.

[12] T.-F. Lin, C.-P. Su, C.-T. Huang, and C.-W. Wu, "A High-Throughput Low-Cost AES Cipher Chip," *Proc. IEEE Asia-Pacific Conf. ASIC,* pp. 85-88, 2002.

[13] V. Rijmen, "Efficient Implemenation of the Rijndael S-Box," http://www.iaik.tu-graz.ac.at/research/krypto/AES/old/~rijmen/rijndael/sbox.pdf, 2006.

[14] S. Morioka and A. Satoh, "A 10-Gbps Full-AES Design with a Twisted BDD S-Box Architecture," *IEEE Trans. VLSI,* vol. 12, no. 7, July 2004.

[15] X. Zhang and K.K. Parhi, "Hardware Implementation of Advanced Encryption Standard Algorithm," *IEEE CAS Magazine,* vol. 2, no. 4, Dec. 2002.

Alireza Hodjat received the BS degree in electrical engineering from the University of Tehran, Iran, in 1999 and the MS degree in electrical engineering from the University of California, Los Angeles (UCLA), in 2002. He received the PhD degree in the field of embedded computing systems from the Electrical Engineering at UCLA in December 2005. He is with Broadcom Corporation, Irvine, California. His research interests include hardware/software codesign and Application Specific Instruction Set coprocessor architectures and VLSI implementations for secure embedded systems. He is a student member of the IEEE.

Ingrid Verbauwhede received the electrical engineering degree in 1984 and the PhD degree in applied sciences from the K.U.Leuven, in Leuven, Belgium, in 1991. She was a lecturer and visiting research engineer at the University of California, Berkeley, from 1992 to 1994. From 1994 to 1998, she was a principal engineer first with TCSI and then with Atmel in Berkeley, California. She joined the University of California, Los Angeles, in 1998 as an associate professor and the K.U.Leuven in 2003. Her interests include circuits, processor architectures, and design methodologies for real-time, embedded systems in application domains such as security, cryptography, digital signal processing, and wireless applications. She is a senior member of the IEEE and a member of the IEEE Computer Society.