Minimum Area Cost for a 30 to 70 Gbits/s AES Processor

Alireza Hodjat and Ingrid Verbauwhede
Electrical Engineering Department
University of California, Los Angeles
{ahodjat, ingrid} @ ee.ucla.edu

Abstract

This paper presents the design decisions and area optimizations to obtain a high throughput, over 30 Gbits/s AES processor. 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 using inner round pipelining of the composite field implementation of the substitution phase and designing an offline key scheduling unit for the AES processor the area cost is reduced by 48 % while the same throughput is maintained. Therefore, the over 30 Gbits/s, fully pipelined AES processor operating in the counter mode can be used for the encryption of data on optical links.

1. Introduction

The data rate on a typical optical link is several tens of Gbits/s. In an optical network the transmitted data must be secure. Data encryption at rates over 30 Gbits/s is a requirement for such networks. 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, is suitable for high throughput applications. It is used to produce high throughput cryptographically strong pseudo-random number generations. It has the advantage that the algorithm can be pipelined because there is no feedback in this mode of operation.

In this paper we will discuss different pipelining options for the Advanced Encryption Standard (AES) [1], which is a symmetric key algorithm widely used for security of different applications. The encryption algorithm of the Advanced Encryption Standard in the counter mode of operation [2] is used to perform data encryption and generate random numbers at the rate of over 30 Gbits/s.

This paper presents the techniques and the architectures that can achieve the required throughput for the above application. Loop unrolling and inner and outer round pipelining of the AES algorithm are techniques that can help us to achieve the throughput of 30 Gbits/s to 70 Gbits/s using a 0.18 μm CMOS technology. However, the area cost will increase very much. Therefore, a pipelined implementation of the composite Galois Field design of the byte substitution phase and the offline key scheduling scheme are used to reduce the area. The maximum area reduction of 48 % was achieved using these techniques.

The rest of this paper is organized as follows. Section 2 will investigate the related work. In section 3 the ultra high speed AES algorithm is presented. The different steps of the algorithm and the design considerations that will lead to a high throughput design will be 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 the architectures and the throughput-area trade-off 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 further area optimization of the AES processor using the offline key scheduling scheme 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 [1]. Since then, there have been many different hardware implementations for ASIC and FPGA. References [3], [6], [9] and [10] present their architecture and their results for ASIC implementation. On the other hand, references [7] and [8] present an efficient implementation of the AES algorithm for FPGA. None of these references present a throughput of over 30 Gbits/s. Our previous work [12] presented the possibilities of achieving a throughput of over 30 Gbits/s encryption using the AES algorithm. This paper is a continuation of [12] which presents an AES processor that runs between 30 to 70 Gbits/s with minimum area cost. The new designs which are addressed in this paper can achieve the same throughput with a maximum of 48% reduction in the area cost.

3. Ultra high speed AES

In the Advanced Encryption Standard [1], the input data is 128 bits. The input key can be 128, 192, or 256 bits long and the algorithm repeats for N_r number of rounds. Figure 1 shows the different steps of the AES

Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design (ISVLSI'04)
0-7695-2097-9/04 $20.00 © 2004 **IEEE**

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Purdue University. Downloaded on April 4, 2009 at 14:12 from IEEE Xplore. Restrictions apply.

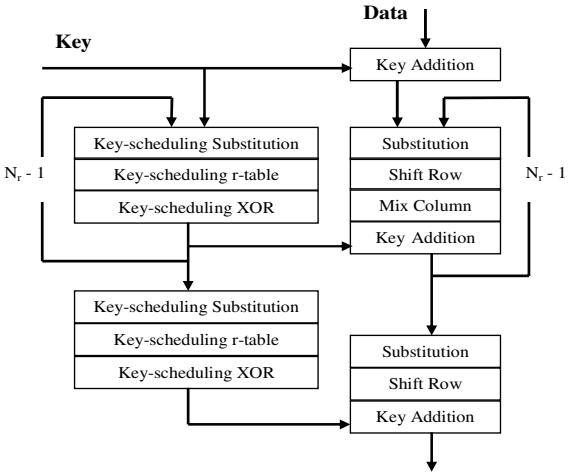


Figure 1: Advanced Encryption Standard

algorithm for each round. These are substitution, shift row, mix column, and key addition. All four 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. Besides 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.

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

1. The critical path of the AES algorithm is in the key scheduling datapath for the key lengths larger than 128 bits. 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 moves in the encryption flow. Moreover, the number of rounds N_r will be fixed, equal to 10.
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 between the datapath instances of each round. Figure 2 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. Byte substitution is a critical phase of the AES algorithm. In this phase every byte of input data is

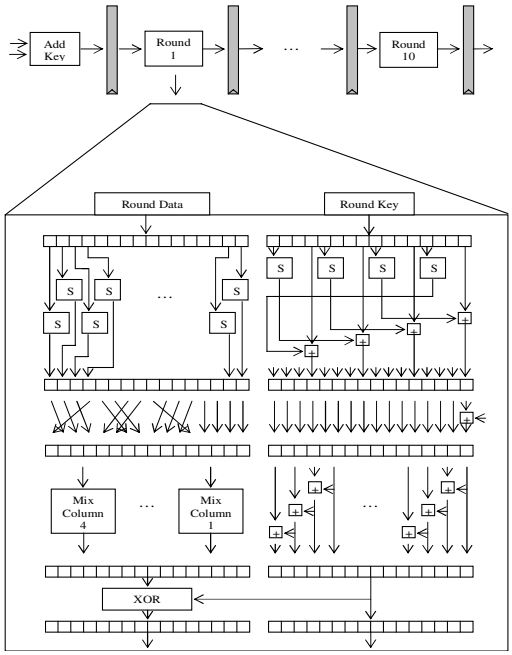


Figure 2: Outer round pipelining for AES algorithm

substituted with another byte. Either look-up tables can be used to implement the substitution phase or the substitution calculation has to be designed in logic. The former has minimum delay but consumes huge amount of area. The latter has efficient area consumption but it has a long critical path [10]. In the next section we show 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 elements of $GF(2^8)$. First the multiplicative inverse in $GF(2^8)$ is calculated. Then, an affine transformation over GF(2) is applied [1]. 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 a Sbox (substitution box) can be done by a look-up table, as shown in figure 3-a. On the other hand, we can implement a 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 [11]. Also reference [6] presents one implementation of such algorithm. This is the composite field implementation of the byte substitution phase. Figure 3-b shows how the complete Sbox can be designed using $GF(2^4)$ operations. We call it a non-pipelined Sbox using GF operators. Here, the input byte (element of $GF(2^8)$) is mapped to two elements of $GF(2^4)$. Then the

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Purdue University. Downloaded on April 4, 2009 at 14:12 from IEEE Xplore. Restrictions apply.

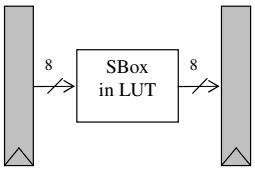
multiplicative inverse is calculated using $GF(2^4)$ operators. Then the two $GF(2^4)$ elements are inverse mapped to one element in $GF(2^8)$. In the end the affine transformation is performed. Notice that an inversion in $GF(2^4)$ can be efficiently implemented using look-up tables because there are only 16 possibilities for four bits. The details of the $GF(2^4)$ operators are from reference [6].

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 inner round pipelining for the AES algorithm because pipeline registers will appear inside of the byte substitution phase which is inside of each round. The only problem is that pipelining of 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. Figure 3-c shows the two-stage pipelined implementation of a Sbox using GF operators. The critical path is broken into half and there are only three 4-bit registers. Figure 3-d shows the three-stage pipelined implementation of a 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.

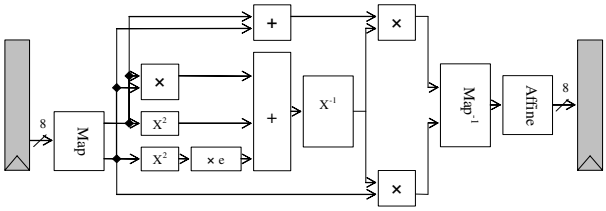
Figure 4 shows the area-delay trade-off of the different implementations of the byte substitution phase that were presented in figure 3. The results shown in this figure are for a 0.18 μm CMOS technology. The Synopsys synthesis tool is used. For synthesis, the UMC 0.18 μm CMOS standard cell library with conservative wire load model is used. This synthesis set-up is used for all the synthesis results in this paper. As seen in figure 4, 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 two or three pipelined implementation should be used. The area cost of one Sbox using two-stage composite field implementation is 23% less than the LUT design with the same speed. For a three-stage composite field implementation this cost is 32% less than the LUT design.

5. High speed AES with online key schedule

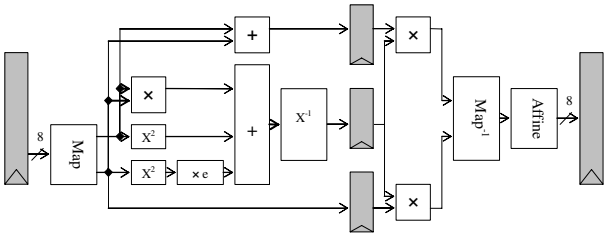
This section presents the architecture and throughput-area trade-off of our high speed AES processor which includes an online (on the fly) key scheduling unit. Figure 5 shows the inner and outer round pipelined architecture for the encryption and key scheduling datapath. The



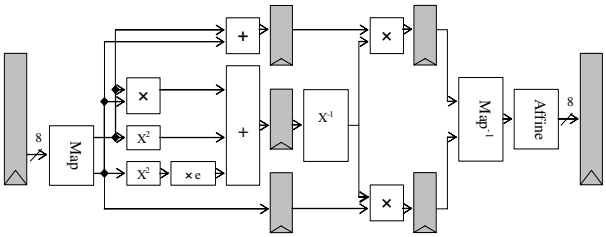
3-a) Sbox using LUT implementation



3-b) Non-pipelined Sbox using GF operations



3-c) Two-stage pipelined Sbox using GF operations



3-d) Three-stage pipelined Sbox using GF operations

Figure 3: Byte substitution architectures

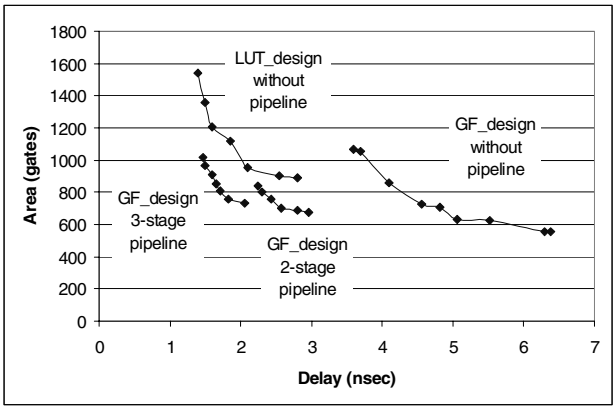


Figure 4: The area-delay trade-off for the Sbox

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Purdue University. Downloaded on April 4, 2009 at 14:12 from IEEE Xplore. Restrictions apply.

difference between this architecture and the design shown in figure 2 is that for the inner round pipelining the datapath inside 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. This design is synthesized using 0.18 μm CMOS technology. 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.

In section 4 it was shown that the two or three stages pipelined implementation of a 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 figure 5. When the Sbox with two pipeline stages (figure 3-c) is used, each round of the AES algorithm will have three pipeline stages., When the Sbox with three pipeline stages (figure 3-d) is used, there are four pipeline stages inside of each round. These are the most area efficient AES implementations with online key scheduling that can achieve 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.

Figure 6 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 figure 2. The design with two pipeline stages per round with LUT Sbox is the implementation of figure 5. The architecture with three pipeline stages per round with composite Sbox is the implementation of figure 5 when the Sbox of figure 3-c is used. The design with four pipeline stages per round with composite Sbox is the implementation of figure 5 when the Sbox of figure 3-d is used.

Figure 6 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 % less than the design with LUT Sbox implementation. Moreover, the architecture with four pipeline stages per round can cost up to 30 % 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 using the pipelined version of the composite filed implementation of Sbox can reduce the area while the same throughput is maintained, but the price that we have to pay is latency. Latency is defined by the number of cycles that it takes for each data sample 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. Since in our application the latency is not important and the main concern is throughput, therefore we can gain much by defining inner round pipeline stages.

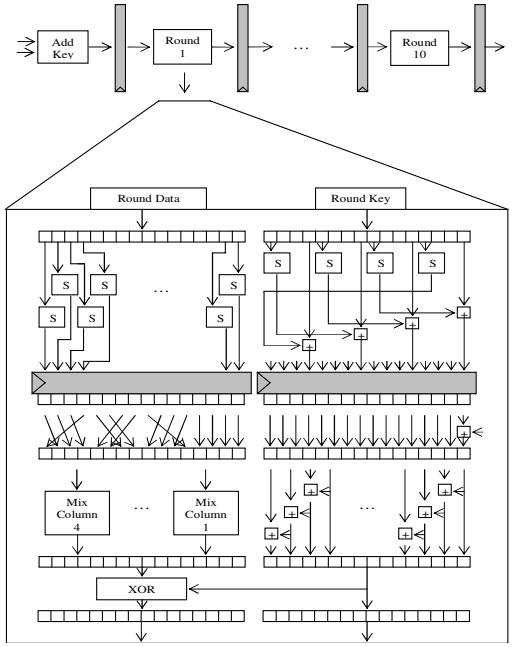


Figure 5: AES with inner and outer round pipelining

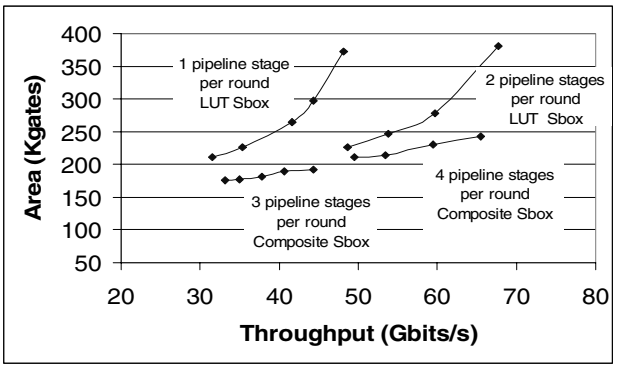


Figure 6: The throughput-area trade-off of the AES processor with online key scheduling

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Purdue University. Downloaded on April 4, 2009 at 14:12 from IEEE Xplore. Restrictions apply.

6. High speed AES with offline key schedule

In most encryption applications, the encryption key does not vary as frequent as data. Specially 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 because the session key is fixed. 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. This way the area consumption is reduced and the same throughput rate is maintained for the encryption of input data in each session. The most important reason that causes this area reduction is the following. Since in our implementation the AES round loop is unrolled, therefore there are a number of registers that keep the value of round keys. Thus, by defining an offline key scheduling unit, there is no overhead in terms of the number of required registers 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.

Figure 7 shows the architecture of the offline key scheduling unit that is designed for our high speed AES processor. Figure 7-a 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 of 128-bit round keys, is generated. Then the *keysch_done* signal is asserted which indicates that the processor is ready to perform the encryption of the input data.

Figure 7-b 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 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. Also because there are only four Sboxes used in the offline key scheduling unit, the fastest Sbox which is the look-up table implementation, is used.

Figure 8 shows the new encryption pipeline which does not include the key scheduling datapath. This is the exact unit that is used for the encryption pipeline block that is shown in figure 7-a. Also notice that figure 8 is similar to the design in figure 5 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 figure 3-c and the four stage pipelined design uses the pipelined Sbox implementation of figure 3-d. 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 reduce the area up to 28 %.

Figure 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 % less than the design with LUT Sbox implementation. Moreover, the architecture with four pipeline stages per round can cost up to 33 % less area than the design with LUT Sbox implementation. Therefore, by using the pipelined implementation of the composite Galois Field

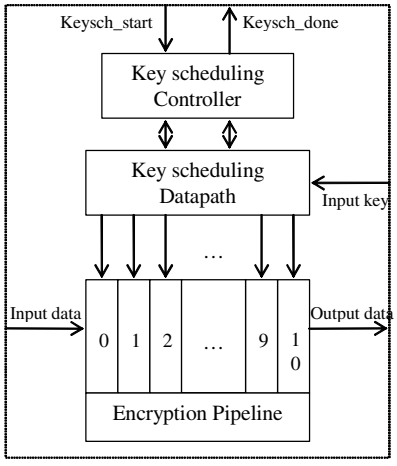


Figure 7-a: Block diagram of the processor with offline key scheduling unit

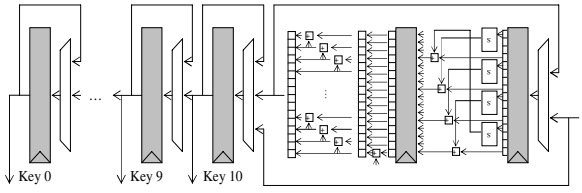


Figure 7-b: The offline key scheduling datapath

Authorized licensed use limited to: Purdue University. Downloaded on April 4, 2009 at 14:12 from IEEE Xplore. Restrictions apply.

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 % is achieved for the high AES core that provides a throughput rate over 30 Gbits/s.

7. Conclusion

This paper presented area efficient architectures for fully pipelined high speed AES processors that can provide an encryption throughput of 30 to 70 Gbits/s. Loop unrolling and inner and outer round pipelining were used to reduce the critical path and increase the maximum throughput. By using the pipelined design of the composite field implementation of the byte substitution phase of the AES algorithm the area is reduced up to 35%. Also by designing an offline key scheduling unit for the high speed AES processor an area reduction of up to 28% was achieved. Therefore, the total area cost of the final architecture was reduced up to 48% without any loss in the throughput. The most 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.

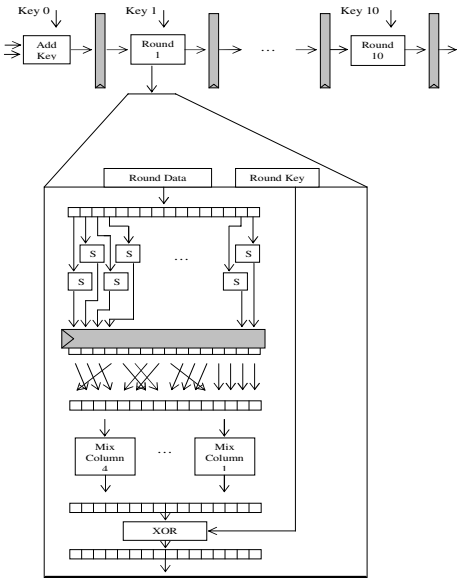


Figure 8: Encryption pipeline for the AES design with offline key scheduling

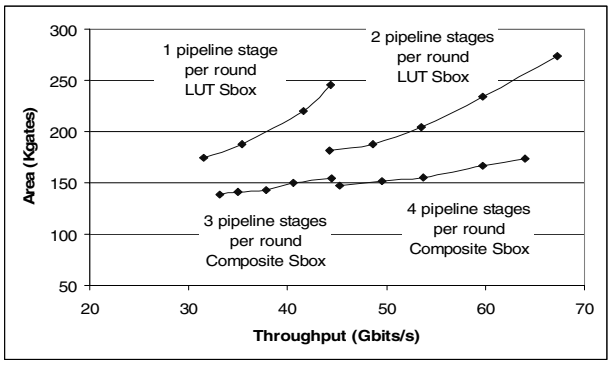


Figure 9: The throughput-area trade-off of the AES processor with offline key scheduling

8. Acknowledgment

The authors would like to acknowledge Professor Miodrag Potkonjak for his feedback in this project. 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.

9. References

[1] National Institute of Standards and Technology (U.S.), Advanced Encryption Standard. Available at: http://csrc.nist.gov/publication/drafts/dfips-AES.pdf

[2] M. Dworkin, SP 800-38A 2001, "Recommendation for Block Cipher Modes of Operations", December 01.

[3] A. Satoh et al, "A Compact Rijndael Hardware Architecture with S-Box Optimization", ASIACRYPT 2001, LNCS 2248, pp. 239-254, 2001.

[4] http://www.nist.gov/aes/

[5] H. Kuo, P. Schaumont, and I. Verbauwhede, "A 2,29 Gbits.sec, 56 mW non-pipelined Rijndael AES Encryption IC in a 1.8 V, 0.18 um CMOS technology," Proc. 2002 CICC, pp. 147-50, May 2002.

[6] J. Wolkerstorfer, E. Oswald, M. Lamberger, "An ASIC Implementation of the AES Sboxes", Proc. RSA Conference 2002, San Jose, CA, February 2002.

[7] M. McLoone, J. McCanny, "High Performance Single Chip FPGA Rijndael Algorithm Implementations", Workshop on Cryptographic Hardware and Embedded Systems, Paris, 2001.

[8] A. Elbirt, W. Yip, B. Chetwynd, C. Paar, "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists", IEEE Trans. of VLSI Systems, 9.4, pp.545-557, August 2001.

[9] T. Ichikawa et al, "Hardware Evaluation of the AES Finalists", in Proc. 3th AES Candidate Conference, New York, April 13-13, 2000.

[10] I. Verbauwhede, P. Schaumont, H. Kuo, "Design and Performance testing of a 2.29 Gb/s Rijndael Processor", IEEE Journal of Solid-State Circuits (JSSC), March 2003.

[11] V. Rijmen, "Efficient Implemenation of the Rijndael S-box", available at: http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf

[12] A. Hodjat, I. Verbauwhede, "Speed-Area Trade-off for 10 to 100 Gbits/s Throughput AES Processor', 37th Asilomar Conference on Signals, Systems, and Computers, November 2003.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Purdue University. Downloaded on April 4, 2009 at 14:12 from IEEE Xplore. Restrictions apply.