TWO DIFFERENT ARCHITECTURES FOR THE HARDWARE IMPLEMENTATION OF THE RIJNDAEL ALGORITHM

Erica Mang
University of Oradea, Romania
Department of Computers Science
3-5 Armatei Romane St., 3700 Oradea
E-mail: emang@uoradea.ro

Abstract. Two architectures and VLSI implementations of the AES Proposal Rijndael, are presented in this paper. These alternative architectures are operated both for encryption and decryption process. They reduce the required hardware resources and achieve high-speed performance. Their design philosophy is completely different. The first uses feedback logic and reaches a throughput value equal to 259 Mbit/sec. It performs efficiently in applications with low covered area resources. The second architectures is optimized for high-speed performance using pipelined technique. Its throughput can reach 3.65 Gbit/sec.

Key words: Rijndael, VLSI, feedback logic, pipelining, throughput

1. Introduction

In our days, the need for secure transport protocols seems to be one of the most important issues in the communication standards. Of course, many encryption algorithms support the defense of private communications. However, the implementations of this algorithm is a complicated and difficult process and sometimes results in intolerant performance and allocated resources in hardware terms. The explanation for this fact is because these encryption algorithms were designed some years ago and for general cryptography reasons. In recent years, new flexible algorithms specially designed for the new protocols and applications have been introduced to face the increasing demand for cryptography.

In October of 2000, the National Institute of Standards and Technology (NIST) announced the cipher Rijndael as the Advanced Encryption Standard (AES) in order to replace the aging Data Encryption Standard (DES) (Schneider 1996).

In the Third Advanced Encryption Standard (AES) Candidate Conference (AES 2000), papers from different research groups were presented (Dandalis et al 2000; Elbirt et al 2000; Gaj et al 2000; Weeks et al 2000; Gaj et al 2001). The main purpose of these works was the evaluation of the AES finalist algorithms in terms of hardware implementation performance. In order to achieve this, all the authors used general purposes architectures and not specialized designs for each algorithm implementation. This is a fair methodology for comparison of different algorithms. On the other hand, this way is not well-suited to the implementation of each algorithm separately. In addition, in two of these works (Dandalis et al 2000; Elbirt et al 2000) only the encryption mode of operation was implemented and not the decryption. References (Elbirt et al 2000; Gaj et al 2000; Gaj et al 2001) do not support the on-chip-generation of the necessary for the algorithm encryption-decryption keys. In other words, the proposed designs do not support the completed operation of the algorithms and perform inefficiently in terms of both the encryption and decryption mode of data transformation.

Especially for the Rijndael algorithm, other works (Kuo et al 2001; Fischer et al 2001; Mroczkowski et al 2001) have been published. The proposed work in (Kuo et al 2001) is an uncompleted implementation of the algorithm's total operation. It supports only the encryption process. In (Mroczkowski et al 2001), two different designs are introduced, one for encryption and one for decryption. They have been implemented in two separate FPGA devices. This is not right way for the implementation of a block cipher. It is not efficient for the implementation of communications protocols, especially in integrated circuits with low allocation resource specifications. The proposed implementation in (Mroczkowski et al 2001) needs two different FPGA devices in order to ensure the complete operation of the algorithm.

In this paper, two architectures and VLSI implementations of the AES proposal are presented. These alternative designs operate both for encryption and decryption process in the same device. They are proposed in order to reduce the required hardware resources and to achieve high-speed performance. In the first design, the appropriate key expansion unit is

integrated with the encryption/decryption core. Performance analysis and comparison results with other works are also reported.

2. The Rijndael Encryption/Decryption Algorithm

A new block encryption algorithm called Rijndael has been developed and published by Daemen and Rijmen (Daemen et al 2001). This algorithm is an iterated block cipher with variable block length and a variable key length. The block and the key length can be independently specified to 128, 192, or 256 bits. The number of algorithm rounds depends on the block and key length.

The different transformations of the algorithm architecture operate on the intermediate result, called State. The State can be pictured as a rectangular array of bytes. This array has four rows. The number of columns is called Nb and it is equal to block length divided by 32. The Key is also considered as a rectangular array with the same number of rows as State. The number of columns is equal to the key length divided by 32. This number is denoted as Nk. The number of rounds, Nr, depends on the values Nb and Nk. For block and key length equal to 128 bits, both values of Nb and Nk are equal to four and the number of rounds Nr is defined as 10. These specifications are served by the proposed implementations, which will be analyzed in detail in the next paragraphs.

A basic round transformation relies on combining operations from four fundamental algebraic functions that operate on arrays of bytes. These transformations are:

- *SubBytes:* Operates in each byte of the State independently. This mathematical substitution is constructed of the compositions of two transformations: multiplicative inverse in GF(2^8) and an affine mapping over GF(2) inverse in GF(2^8), too, and the inverse of the affine mapping transformation over GF(2).
- *ShiftRow:* Cyclically shifts the rows of the State over different offsets. The operation is almost the same in the decryption process except for the fact that the shifting offsets have different values.
- *MixColumn:* In this transformation, the columns of the State are considered as polynomials over GF(2^8) and are multiplied with a fixed polynomial
  $c(x)=´03´x^3+´01´x^2+´01´x+´02´$
  for encryption and with the polynomial
  $d(x)=´0B´x^3++´0D´x^2+´09´x+´0E´$
  for the decryption process.
  Both polynomial multiplications are modulo (x^4+1).
- *KeyAddition:* In this operation, the round key is applied to the State by simple bit by bit XOR. KeyAddition is the same for the decryption process.

Before the first round, a key addition layer is applied to the cipher data. This transformation is stated as the algorithm initial round key addition. The final round of the cipher is equal to the basic round with the MixColumn step removed. A key expansion unit is defined in order to generate the appropriate key, for every round, from the initial key value. When all rounds of transformation are completed, a cipher data block with the same length as the plain data has been generated.

The decryption process has the same structure as the encryption architecture. The only main difference is that for every function that is used in the basic round, the mathematical inverse of it is taken. The key expansion unit performs almost the same operation with the encryption process. The only difference is that the decryption of the round keys is obtained by applying the inverse MixColumn to the corresponding round keys. The initial value of the key for the decryption operation is changed. The appropriate basic decryption key must be loaded in the key buffer before the decryption beginning (Daemen et al 2001).

3. Hardware Architectures. VLSI Implementations

Two alternative architectures are proposed for the Rijndael algorithm in order to reduce the required hardware resources and to achieve high-speed performance. Both architectures serve the encryption and decryption process in the same hardware device.

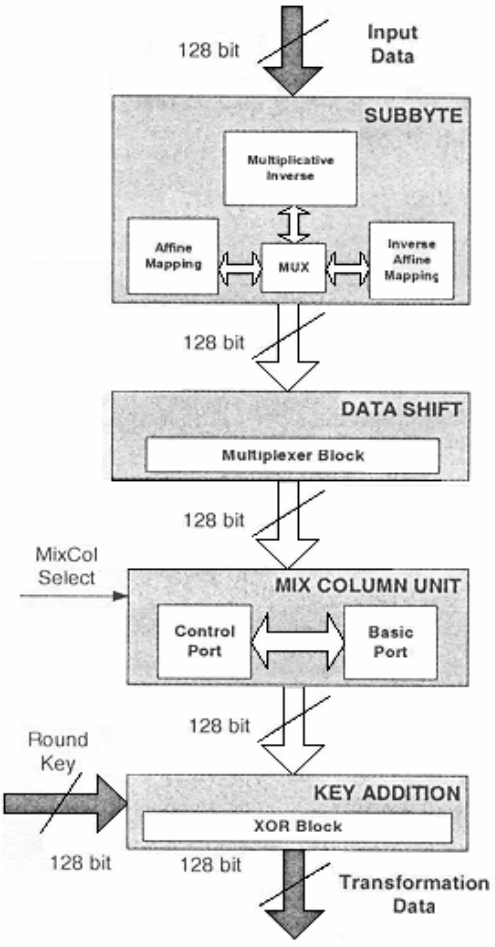

Figure 1. Basic block round architecture

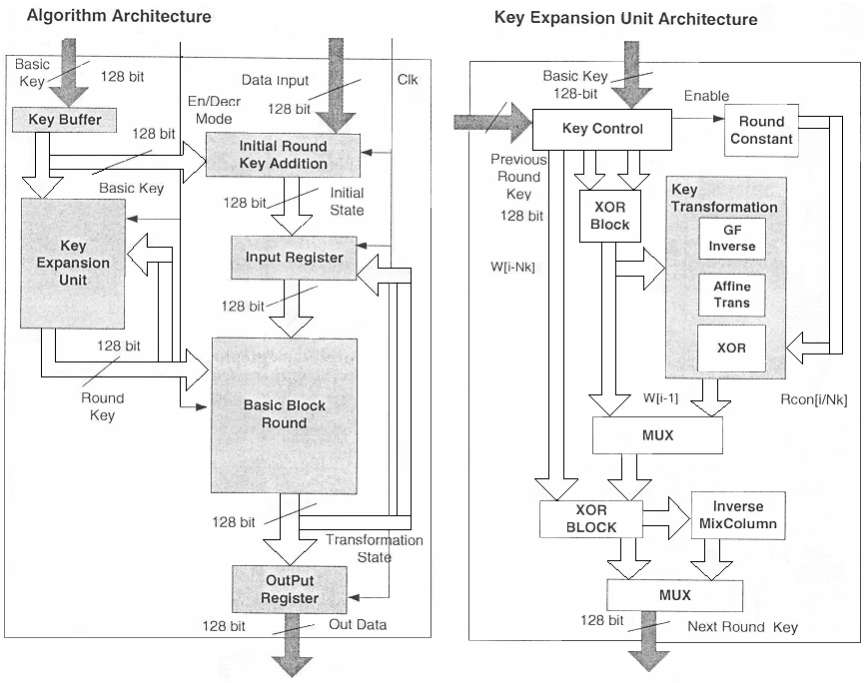

Figure 2. Key Expansion Unit architecture

3.1. Basic Block Round

The architecture of the basic block round is shown in Figure 1. As was already mentioned in the previous section, each basic round of the algorithm is composed of basic building blocks: SubByte, ShiftRow, MixColumn, and KeyAddition. The structure of SubBytes and MixColumn turned out to be challenging.

3.2. First Architecture with Implemented Key Expansion

The first proposed architecture is shown in detail in Figure 2. This architecture performs both the encryption and the decryption process, with input plaintext and key vector equal to 128 bit. The algorithm specifies 10 rounds for the State transformation and an extra initial round key addition.A key buffer of 128-bit width is used for the key storage.

In the initial round key addition transformation, the input state is XOR-ed with the input key. In the first step, the initial round key addition is executed and the key for the first round is calculated. In a clock cycle, one transformation round is executed and, at the same time, the appropriate key for the next round is calculated. The whole process reaches the end when 10 rounds of transformation are completed. The input Register is used to keep the transformed State after every round of operation. The State is forced to this register with the use of a feedback technique. The Basic Block Round architecture is shown in Figure 1 and has been described in detail in Section 3.1.

The Key Expansion Unit architecture is illustrated in Figure 2. The round keys are derived from the initial key. Two are the basic component of this unit, the Key Transformation and the Round Key selection. The total number of the round key bits is equal to the block length, multiplied by the number of rounds plus one. The proposed implementation with 128 bit block length and 10 rounds generates 10*128 bit round keys. The round keys are taken from the initial key in a complicated way, defined in detail in the algorithm specification (Daemen et al 2001).

The algorithm demands a different operation mode of the key expansion unit, between encryption and decryption processes. The basic difference is that, in decryption, the round keys are obtained by applying the inverse MixColumn to the corresponding round keys.

The total execution time is one clock cycle for every round, plus one clock cycle for the initial round key addition. So, the system needs 11 clock cycles in order to completely transform a 128 bits data clock.

The Key Expansion Unit architecture is illustrated in Figure 2. The round keys are derived from the initial key. Two are the basic component of this unit, the Key Transformation and the Round Key selection. The total number of the round key bits is equal to the block length, multiplied by the number of rounds plus one. The proposed implementation with 128 bit block length and 10 rounds generates 10*128 bit round keys. The round keys are taken from the initial key in a complicated way, defined in detail in the algorithm specification (Daemen et al 2001).

The algorithm demands a different operation mode of the key expansion unit, between encryption and decryption processes. The basic difference is that, in decryption, the round keys are obtained by applying the inverse MixColumn to the corresponding round keys.

The total execution time is one clock cycle for every round, plus one clock cycle for the initial round key

addition. So, the system needs 11 clock cycles in order to completely transform a 128 bits data clock.

3.3 Second Architecture Using RAM for Key Storage

The second proposed architecture is shown in Figure 3. The main characteristics of this are:

1) the pipelining used technique and

2) the usage of a RAM for the key storage and loading.

It is not possible to apply pipelining in many cryptographic applications. However, the Rijndael cryptographic algorithm internal architecture provides the possibility of being implemented with pipelining technique. The pipelining architecture offers the benefit of high-speed performance. The implementation can be applied in applications with hard throughput needs. This goal is achieved by using a number of operating blocks with a final cost to the covered area.

The proposed architecture uses 10 basic round blocks, which are cascaded by using pipeline registers. In this architecture, 10 blocks of data can be transformed at the same time. The main disadvantage of the second proposed design is the increased required effective area. In order to face this problem, RAM was used for the key storage.

Many FPGAs provide embedded RAM, which many be used to replace the Key Expansion Unit and the internal buffer of these architecture for the initial key. In this way, the appropriate key for each round can be addressed from the RAM. External RAM blocks can also be used. The size of RAM megacells can be customizable to fit the application demands in terms of the key length.

In such architectures the switching time of the RAM is a factor that has to be considered in the total performance timing measurements.

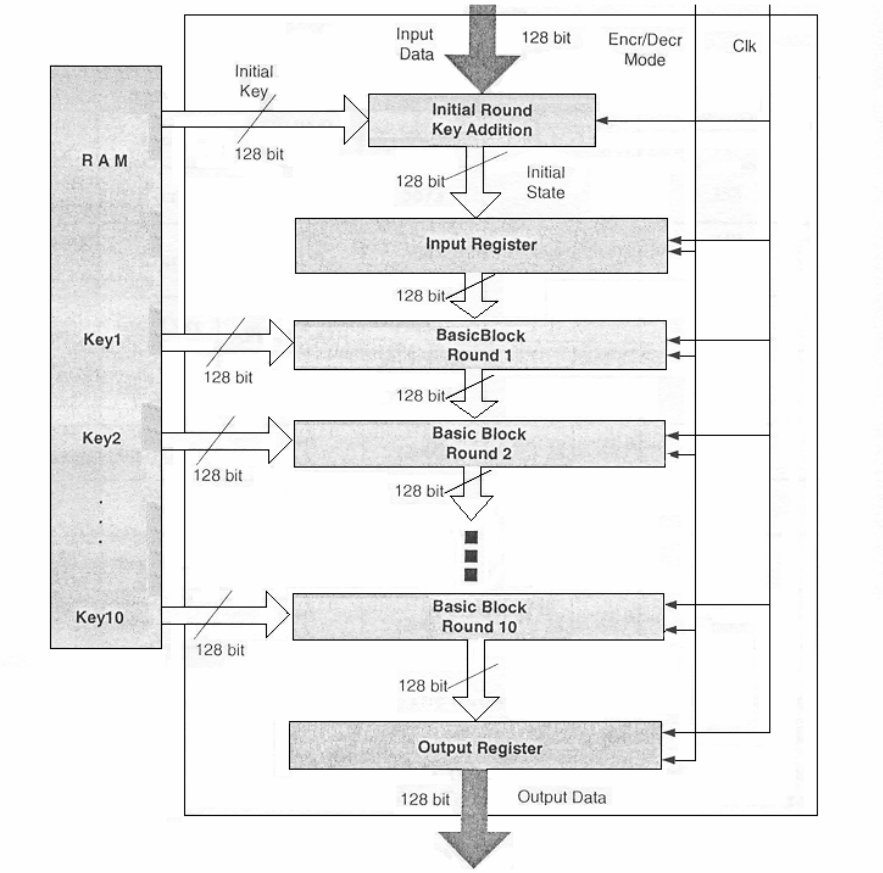

Figure 3. Architecture using RAM for key loading

4. Performance Analysis

Each one of the proposed architectures was implemented by using VHDL, with structural description logic. Both implementations were simulated for the correct encryption and decryption operation using the test vectors provided by the AES submission package (AES 2000). The VHDL codes of the two designs are synthesize, placed and routed using FPGA devices of Xilinx (Virtex) (Xilinx 2001). The two architectures were simulated again for the verification of the correct functionality in real time operating conditions.

The measurements of the performance analysis are shown in Table 1. Measurements from other designs are added in the same table.

The first architecture was optimized with covered area constraints. Xilinx Virtex XCV300BG432 was selected for this architecture implementation. The throughput reaches the value of 259 Mbit/sec for both encryption and decryption process. This architecture operates with an external clock with frequency of 22 Mhz. In the proposed architecture, the critical path is 45 ns.

Table. 1. The measurements of the performance

Arch.	Proc.	FPGA	CLB	Fre.	Thr.
First	En/De	XCV300BG560	2358	22	259
Second	En/De	XCV1000BG560	17314	28.5	3650
(Dandalis et al 2000)	Encr.	Xilinx	5673	-	353
(Elbirt et al 2000)	Encr.	XCV1000BG560	5302/10992	14.1/31.8	300/1940
(Gaj et al 2000)	En/De	Xilinx	2902	25.9	331
(Weeks et al 2000)	En/De	ASIC	$35x10^6um^2$	-	265
(Kuo et al 2001)	Encr.	ASIC	3.96 mm^2	100	910
(Fischer et al 2001)	En/De	Altera	845 LE	-	750
(Mroczkowski et al 2001)	Decr.	Altera	2885	41.5	248

The throughput is calculated with the following formula:

$$\text{Throughput}=\text{block\_size}*\text{frequency}/\text{total clock cycles} \quad (1)$$

The transformed block size is 128 bit and the frequency is 22 Mhz. The necessary clock cycles for one block encryption or decryption are 11.

For the second pipelining architecture, the device has 128k bits of embedded RAM, divide in 32 RAM blocks, that are separate from configured to provide a maximum of 384K bits of RAM independent of the supported embedded RAM. The Virtex block RAM also includes dedicated routing to provide an efficient interface with both Configurable Logic Blocks (CLBs) and other block RAMs.

The throughput in the pipelining architecture is give by:

$$\text{Throughput}= \text{block\_size}/\text{Tclkbasic} \quad (2)$$

where Tclkbasic is he delay of a single round, including register delay. Tclkbasic is 35 ns. The width of the transformed block size is 128 bits. The second architecture achieves throughput 3.65 Gbit/sec. The external clock frequency is 28.5 Mhz. All the compared architectures operate with data and key block width of 128 bits. Someone could claim that the proposed first architecture has a little bit slower performance at about 10, 15 percent compared with the other architectures. Nevertheless, this is a physical result of the algorithm philosophy and not a tradeoff. In this cryptographic algorithm, the key expansion unit is partially modified in the case of decryption process. Especially, as the Rijndael introducers clarify in their AES-proposal specifications (Daemen et al 2001), the InvMixColumn has to be applied to all round keys except the first and the last one, during the decryptions process. In our first architecture proposed, the critical path is specified of the key expansion unit. In order to have a hardware implementation that supports both encryption and decryption the critical path of the key expansion unit for the slower process defines the critical path of the total system.

The two proposed architectures support encryption and decryption in the same dedicated hardware device. So, in a comparison attempt, in hardware performance with other architectures that support only encryption (Dandalis et al 2000; Elbirt et al 2000), these special algorithm characteristics must be considered. Some other designs (Elbirt et al 2000; Gaj et al 2000) do not support the appropriate key scheduling unit in the implemented device. In the first architecture proposed, the appropriate key expansion unit has been integrated in the same FPGA device. This extra feature of these architecture adds, of course more allocated hardware recourses and decreases the algorithm core performance.

5. Conclusions

Two different philosophies of VLSI architectures for the design and implementation of the Rijndael encryption algorithm have been presented.

The first uses feedback logic and reaches throughput value equal to 259 Mbit/sec. This architecture supports key expansion unit in the same device and performs efficiently in applications with low covered area resources.

The second is optimized for high-speed performance using pipelining technique with high data throughput of 3.65 Gbit/sec.

The resulting VLSI circuits achieve data rates significantly high, supporting both operation processes (encryption/decryption) of Rijdael algorithm. They can be applied to online encryption/decryption needs of high speed networking protocols like Asynchro-nous Transfer Mode (ATM) or Fiber Distributed Data Interface (FDDI).

References

”Third Advanced Encryption Standard (AES) Candidate Conf.”, 2000. http://crscr.nist.gov/encryption/aes//round 2/conf3/aes3conf.htm.

J. Daemen and V Rijndael, *"AES Proposal: Rijndael",* 2001.

A. Dandalis, V.K. Prasanna, and J.D.P. Rolim, "*A Comparative Study of Performances of AES Final Candidates Using FPGAs*," Proc. Third Advanced Encryption Standard (AES) Candidate Conf., Apr. 2000.

A.J. Elbirt, W. Yip, Bchetwynd, and C. Paar, "*An FPGA Based Performance evaluation of the AES Block Cipher Candidate Algorythm Finalists,*" Proc. Third Advanced Encryption Standard (AES) Candidate Conf., Apr. 2000.

V. Fischer and M Drutarovsky, *"Two Methods of Rijndael Implementation in Reconfigurable Hardware*", Proc. CHESS 2001, May 2001.

K.Gaj and P. Chodowiec, "*Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware,"* Proc. Third Advanced Encryption Standard (AES) Candidate Conf., Apr. 2000.

K. Gaj. And P. Chodowiec, *"Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard Using Field Programmable Gate Array" ,* Proc. RSA Security Conf., Apr. 2001.

H. Kuo and I. Verbauwhede, "*Architectural Optimization for a 1.82 Gbits/sec VLSI Implementation of the AES Rijndael Algorithm" ,* Proc. Chess 2001, May 2001.

P.Mroczkowski, *"Implementation of the Block Chiper Rijndael Using Altera FPGA",* 2001.

B. Schneider, *Applied Cryptography – Pprotocols Algorithm sand Source Code in C,* New York: John Wiley & Sons, 1996.

B.Weeks, M.bean, T. Rozylowicz, and C.Ficke, *"Hardware Performance Simulations of round 2 Advanced Encryption Standard Algorithms",* Proc. Third Advanced Encryption Standard (AES) Candidate Conf., Apr 2000.

Xilinx Inc., San JOSE, Calif., *"Virtex, 2.5 V Field Programable Gate Array",* 2001, www.xilinx.com.