Design and Implementation of an FPGA-Based 1.452-Gbps Non-pipelined AES Architecture

Ignacio Algredo-Badillo, Claudia Feregrino-Uribe, and René Cumplido

National Institute for Astrophysics, Optics and Electronics,
Luis Enrique Erro #1, CP 72840, Sta. Ma. Tonantzintla, Puebla, México
{algredobadillo, cferegrino, rcumplido}@inaoep.mx
http://ccc.inaoep.mx

Abstract. This work reports a non-pipelined AES (Advanced Encrypted Standard) FPGA (Field Programmable Gate Array) architecture, with low resource requirements. The architecture is designed to work on CBC (Cipher Block Chaining) mode and achieves a throughput of 1.45 Gbps. This implementation is a module of a configuration library for a Cryptographic Reconfigurable Platform (CRP).

1 Introduction

Cryptographic systems in diverse applications, like multimedia, WWW servers, the Transport Layer Security (TLS) protocol and secure mail protocols such as S/MIME, have provided a safe way for storing and transmitting information. These systems offer security based on complex architectures by adding cryptographic algorithms that may be hash functions, symmetric key algorithms and asymmetric key algorithms [1]. Each one can be used for multiple and different services, such as: authentication, integrity, data confidentiality, pseudorandom number generator, digital signatures and key establishment.

Secure communication protocols handle a symmetric key cryptosystem, a hash algorithm, and method for providing digital signatures and key exchange using public key cryptography [2] and have several operation modes. This work focuses on IPSec (Internet Protocol Security) due to its growing popularity [3], it operates at the network IP layer of the TCP/IP stack and utilizes CBC mode. It is an algorithm-independent protocol, securing traffic of whichever network topology. It has a set of open standards and protocols for creating and maintaining secure communications over IP networks. For example, IPSec leverages other important standards like IKE (Internet Key Exchange) protocol, authentication standards (Kerberos, RADIUS, and X.509 digital certificates), and encryption algorithms (AES and 3DES). Communication networks, like Gigabit Ethernet require processing speeds of 1 Gbps and it is expected that also future wireless personal area networks perform at these data rates [4]. These networks require flexible, high throughput systems which compute cryptographic algorithms that are more efficiently implemented in custom hardware that in software running on general-purpose processors (GPPs) [5]. Also, the hardware implementations offer more security than software ones because they cannot be as easily read or modified by an outside attacker [6]. Implementing cryptography in FPGA

M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3982, pp. 446 – 455, 2006.

devices provides a good alternative to custom and semi custom ASICs (Application Specific Integrated Circuits), which have an expensive and time-consuming fabrication, and more inflexibility or parameter switching [7], and GPPs and special-purpose processors, like DSPs (Digital Signal Processors) [8], that offer lower performance. The advantages of the FPGA reprogrammable devices are especially prominent in security applications, where new security protocols decouple the choice of cryptographic algorithms from the design of the protocol, and users select the cryptographic standard to start a secure session.

AES algorithm ensures the compliance with many of the security regulations, including IPSec and Suite B of recommended cryptographic algorithms, with keys sizes of 128 and 256 bits as announced by NSA (National Security Agency) at the 2005 RSA Conference [9].

2 AES Algorithm

The AES algorithm is a symmetric block cipher that can process data blocks of 128 bits, and it uses cipher keys of 128, 192, and 256 bits [10]. This work implements a hardware architecture of the AES algorithm for ciphering 128-bit data with 128-bit keys. Key length of 128 bits is selected, because it fits in the current IKE, and has the benefits of performance. One initial round is performed and ten round functions (see Fig. 1), where a round function has four transformations (non-linear byte substitution, constant Galois field multiplication, key addition, and an S-box substitution) to obtain the cipher text. An important operation is key expansion, which computes a key schedule or a 128-bits key in each round. The non-linear byte substitution and key expansion operations require S-box substitution, where one byte is substituted and determined by the intersection of the row and the column. These substitution values for the byte *xy* are defined in [10].

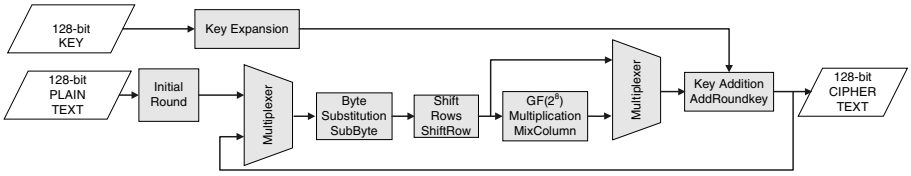

Fig. 1. General structure of AES Algorithm

3 Related Work

Recently, several algorithm-specific hardware architectures of AES algorithm have been reported in the literature, involving commercial and research works, with different design techniques, architectures and FPGA resources. Among these are iterative architectures on Virtex-II, Virtex-4, and a pipelined architecture on Virtex-II.

The commercial implementation in [11] has an iterative architecture, 128-bit data input, data output and key input buses. The datasheet presents FPGA implementations, where the "Fast version" has a throughput of 0.58 Gbps. The work in [12] presents a partitioning architecture without using the BRAMs (Blocks RAM). The

architecture is designed in two parts: 1) implementation of the Key Expansion, which calculates the round keys, and 2) implementation of the functional rounds to cipher 128-bit data. The implementation results of the second part show a throughput of 0.20 Gbps. In [13] AES implementation synthesis results are reported with three different key lengths, and the best throughput is 1.19 Gbps with 128-bit data buses. [14] describes an AES commercial product, which offers diverse operation modes and key lengths. The architecture uses 4 BRAMs in CBC mode to cipher data; the implementation needs 44 clock cycles at 93 MHz, performing at 0.27 Gbps.

The next two works are included for comparing their throughputs and FPGA resource utilization. The work in [15] reports four AES pipelined architectures, where two of them use BRAMs. The 7-stage AES architecture shows the highest throughput of 21.64 Gbps, at the expense of FPGA resources. [16] presents commercial implementations on the Xilinx Virtex-II FPGA, the main characteristics are a throughput of about 1.40 Gpbs, using 18 BRAMs and 1,125 slices.

The previous works demonstrate that implementing S-box on internal memory improves the throughput, decreases the used FPGA resources, and reduces the critical path time. Current architectures with greater throughputs use pipelined structures, which are mentioned only as a reference.

This work reports an AES architecture that aims to perform above 1 Gbps to meet the speed requirements of a Cryptographic Reconfigurable Platform that changes its configuration and functionality to suit the required cryptographic implementation.

4 AES Hardware Architecture and FPGA Implementation

In a communication line or in a transmission channel, the CBC operation mode does not permit pipelined architectures, because feedback operations are performed after ciphering a block [17] (see Fig. 2).

The architecture implemented is based on the AES standard algorithm specified in the Federal Information Processing Standards Publication 197 (FIPS-197) [10] of the National Institute of Standards and Technology. It performs in CBC mode to meet the IPSec requirements.

The aim of this work is to implement a fast and simple iterative AES architecture with low FPGA resource requirements. It was written and simulated in Active-HDL and implemented in Xilinx ISE 6 for the measurement of hardware parameters such as logic and clock frequency. The architectures were synthesized, mapped, placed and routed for an FPGA Xilinx XC2V1000-FG456. This device validates the architecture, and by no means is for a final product.

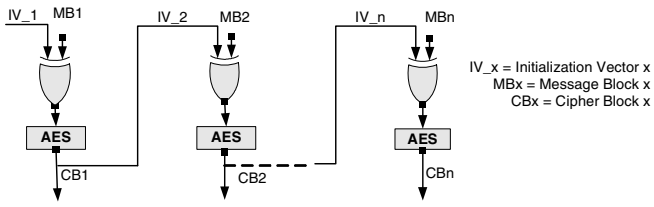

Fig. 2. AES algorithm in the CBC operation mode

The main modules of the architecture are: 1) *AES_CONTROL*, which outputs control signals and organizes the dataflow, 2) *AES_GENKEY*, which outputs the round keys, and 3) *AES_ROUND*, which ciphers the data (see Fig. 3).

The initial round is computed by the *X01* gate, and the following ten rounds are executed by the *AES_ROUND* module. The round keys are added in *AES_ROUND* module and the intermediate cipher data are feedback to the same module until the final cipher data are obtained. The selection of the initial round data and the intermediate cipher data is made by the *M01* multiplexer. After several clock cycles, the final cipher data are addressed from multiplexer output.

AES_ROUND is the main module, it covers the four transformations defined in [10] (see Fig. 4). This module calculates the ten round functions, whereas the initial round operation and the key generation are externally operated.

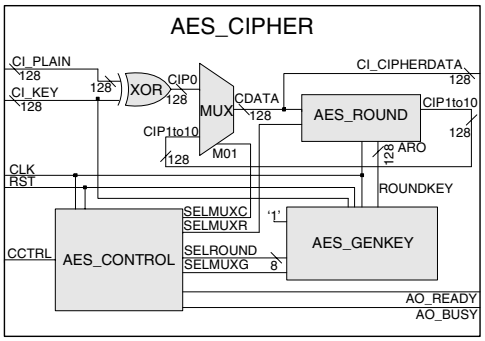

Fig. 3. General architecture of the AES implementation

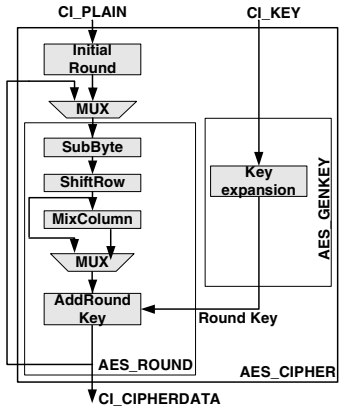

Fig. 4. The four transformations of the AES algorithm are integrated on the *AES_ROUND* module of the general architecture

The general architecture of the basic modular implementation is iterative, and the S-boxes are implemented using twenty distributed memories. *AES_Control* module is a 12-state FSM (Finite State Machine). The state diagram and FSM initial values are shown in Fig. 5. The *START* state modifies the value of the *SC_BUSY* signal. Only in this state, the signal will have value of logical zero, indicating the system is waiting for data. *SC_BUSY*='1' in other states indicates the system is busy ciphering. When *IC_STARTCIP*='1' and the actual state is *START*, FSM changes to the *LOAD* state, which registers the key and input data. In this state, the *OC_SELMUXC* signal controls the dataflow, both input data and intermediate cipher data ('1' for input data and '0' for cipher data), while *OC_SELMUXG* selects the input key or round keys ('1' for round keys and '0' for input key). Also, the initial round is computed, and the next ten states compute the ten rounds left. In each of the following clock cycles, from *ROUND1* to *ROUND10* states are active, and the *OC_ROUND* value changes for the *AES_GENKEY* module. The *ROUND10* state presents the final cipher data in the *CI_CIPHERDATA* bus. Only in this state, *OC_READY*='1' indicates a valid output. If

there are more data to cipher, the *IC_STARTCIP* signal should be in high level and the next state is *ROUND1*, else FSM is set to *START* state. In the first case, the next plain data are stored in the *ROUND10* state, and they are processed from *ROUND1* to *ROUND10* states.

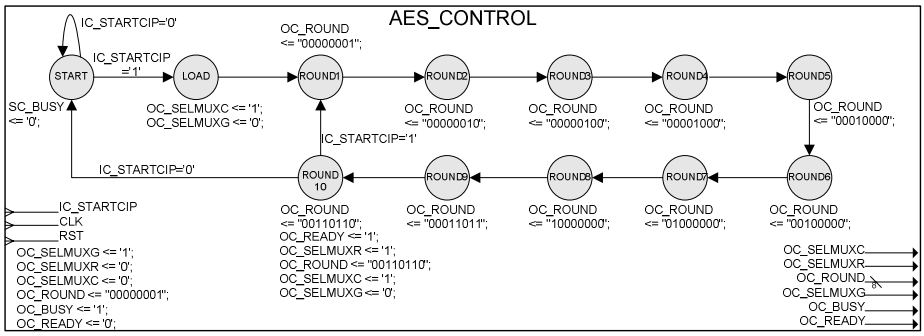

Fig. 5. State diagram of the *AES_CONTROL* module

If the system ciphers data, and it is maintained in the *ROUND0-ROUND10* loop, its output value will offer 128-bit cipher data every twelve clock cycles for 128-bit plain data and 128-bit key data. The throughput of the iterative architecture is given by [18]:

$$\text{Throughput} = \text{Plain\_data\_block\_sice} / ((\text{Clock\_period})(\text{Clock\_cycles})) \quad (1)$$

AES_GENKEY module is the key-expansion operation, which outputs a 128-bits key every round (see Fig. 6). S-boxes and XOR gates compute the round keys, the register stores the *CI_KEY* input or Round *Key* bus, and the multiplexer selects these keys. The S-boxes are implemented in four distributed memories. In the *LOAD* and *ROUND10* states the key input is stored, and from *ROUND1* to *ROUND10* states, round keys are stored. In the *ROUND10* state, the key input is stored because it is used by the *ROUND1-ROUND10* loop, when the system ciphers data successively.

The general structure of the *AES_ROUND* module is shown in the Fig. 7. This module computes the four transformations defined in [10]. The SubByte transformation is performed by S-boxes implemented in 16 distributed memories. The ShiftRow transformation is made by readdressing the *BYTESUB* bus to the *SHIFTROW* bus. This has the effect of cyclically shifting over different number of bytes. The MixColumn transformation operates $GF(2^8)$ multiplications over *SHIFTROW* bus, and it is performed by the *MIXCOL* module, which outputs the *MIXCOLUMN* bus. Finally, in the AddRoundKey transformation, the round key is added by a simple XOR operation. The multiplexer selects in the first nine rounds the *MIXCOLUMN* bus, whereas in the last round it selects the *SHIFTROW* bus. The multiplexer output is added to the *IKEY* bus (round key).

The *MIXCOL* module computes multiplications and additions over $GF(2^8)$. In [10] it is described a matrix multiplication with the fixed polynomial:

$$a(x) = \{3\}x^3 + \{1\}x^2 + \{1\}x + \{2\} \quad (2)$$

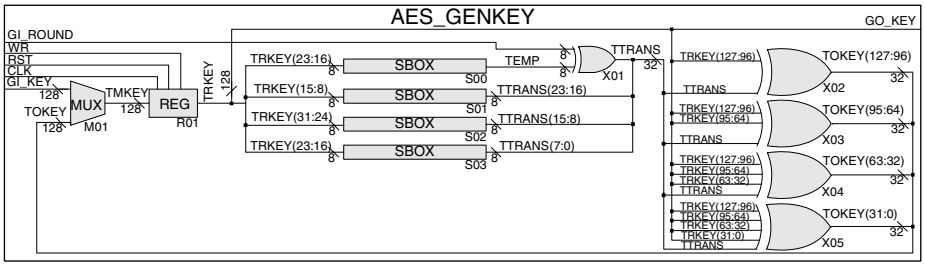

Fig. 6. Diagram of the *AES_GENKEY* module

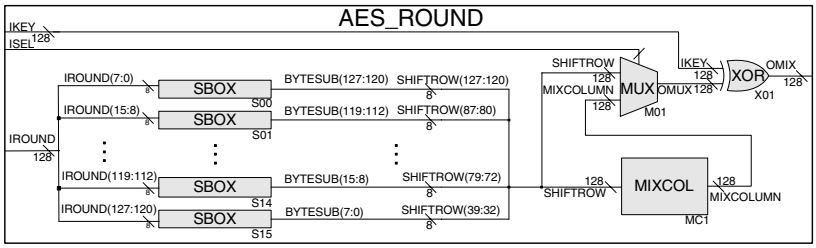

Fig. 7. Diagram of the *AES_ROUND* module

The equation computes multiplications {1} and {3}, and additions. The $GF(2^8)$ addition is the XOR operation and the $GF(2^8)$ multiplication is special since it is only necessary to multiply by some constants [19]. Constant multiplicands permit to implement XOR operations and multiplexers, and these substitute the multiplication described in [10]. For example, a section of *MIXCOL* module is shown in Fig. 8, where a MixColumn transformation is performed for the OMIX(127:120) byte, or

$$\begin{aligned} \text{OMIX}(127{:}120) <= {} & \{2\}*\text{IMIX}(127{:}120) \oplus \{3\}*\text{IMIX}(119{:}112) \oplus \\ & \{1\}*\text{IMIX}(111{:}104) \oplus \{1\}*\text{IMIX}(103.96). \end{aligned} \tag{3}$$

The {1}, {2} and {3} constant coefficients in (3) are multiplied in $GF(2^8)$. In multiplication by {1}, the result is equal to the non-one factor, so, *IMIX*(111:104) and *IMIX*(103:96) bytes are added by the *XOR49* gate.

The multiplication by {2} is a conditional 1-bit left shift implemented by a multiplexer. Its selector, *IMIX*(127), controls the overflow in $GF(2^8)$. If the value being multiplied is less than "10000000", the result is the value itself left-shifted by 1 bit, *IMIX*(126:120)&'0'. If the value is greater than or equal to "10000000", the result is the value left-shifted by 1 bit added with "00011011", *IMIX*(126:120)&'0' XOR "00011011". This prevents overflowing and keeps the product of multiplication in $GF(2^8)$.

The multiplication by {3} is reduced to additions and multiplications by {2}, where the last multiplications are conditional 1-bit left-shifts [19]. Multiplication by {3} can be decomposed as {3} = {2} + {1}. Thus:

$$\{3\}* \text{IMIX}(119:112) <= \{2+1\} * \text{IMIX}\{119:112\}$$
$$<= \{2\} * \text{IMIX}(119:112) + \{1\} * \text{IMIX}(119:112) \quad (4)$$

The multiplication by {3} is implemented by two multiplexers, three XOR gates, and multiplications by {1} and {2} implemented as mentioned in the above paragraph.

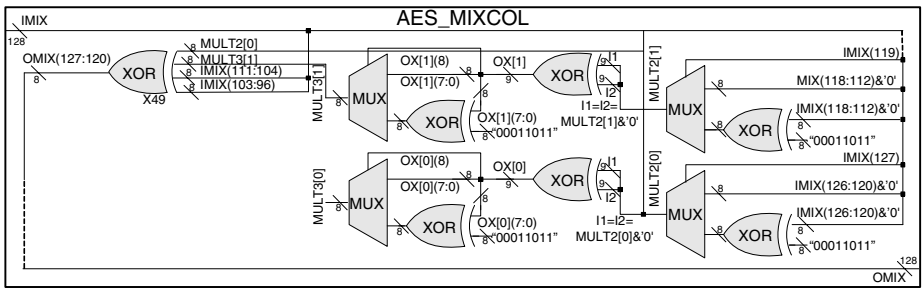

Fig. 8. Diagram of the operation in (3), which is part of the AES_ROUND module

As the IOBs requirements of the general AES architecture (see Fig. 3), exceeds the IOBs of XC2V1000-FG456 device, input data and key are stored in registers to be processed later on in parallel (see Fig. 9). So, the final architecture multiplexes the *CI_PLAIN* and *CI_KEY* 128-bit buses, requiring not additional clock cycles, since the data are stored in the last block processing time. In a given clock cycle, a bus is registered, and in the next clock cycle, the other bus. By successively ciphering data, the key and plain data are stored in run time, and each ten clock cycles, an *AO_CIP* output or cipher data are obtained.

Initially, the twenty S-boxes were implemented in twenty distributed memories, and the architecture achieved a throughput of 0.92 Gbps, with a clock frequency of 86.94 MHz, 2,335 used slices, 4,327 LUTs, 263 IOBs and 10 clock cycles.

AES architecture is part of the CRP[20], which requires cryptographic implementations for transmission speeds of 1 Gbps, and due to (1), *Plain_data_block_size* has a constant value of 128 bits, whereas *Clock_cycles* in this design has a value of 10 and *Clock_period* of $(86.94 \text{ MHz})^{-1}$. Consequently, a reduction of *Clock_cycles* implies an architecture with unrolled rounds, which increases the use of FPGA resources and the critical path time or decrease the clock frequency. A reduction of critical path time, by the modification of *Clock_period* is the more practical option.

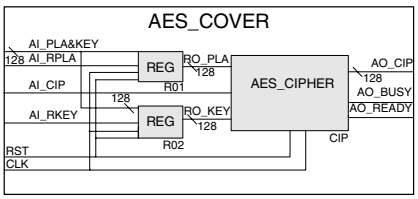

Fig. 9. Final general AES architecture

The implementation of twenty distributed memories for twenty S-boxes requires proportional FPGA resources to place and route them, which results in a critical path time proportional to the FPGA utilized logic. So, modular architecture is redesigned to use 10 dual-port embedded memories, instead of twenty, decreasing the used FPGA resources, and thus the critical path time.

5 Implementation Results and Comparisons

The implementation results of the AES architecture are shown in Table 1, and these are taken from the post-Place & Route reports. Changing the twenty internal memories by ten dual-port memories decreases the critical path time from 11.50 ns to 8.80 ns, reduces the FPGA resources, and eliminates some intermediate registers. The implementation results indicate that, in terms of required FPGA resources, the S-box substitution is the dominant element of the AES implementation.

Table 1. Implementation results of the non-pipelined iterative AES architecture

Period (ns)	Clock (MHz)	IOBs (out of 324)	Slices	4-Input LUT	Latency (Clk cycles)	Throughput (Gbps)
8.80	96.42	263	586	847	10	1.45

These results show less wired and logic FPGA resources, which are 586 slices and 847 LUTs. This leads to a compact architecture with a lower critical path time, a higher clock frequency (96.42 MHz) and a throughput of 1.45 Gbps.

Different device families will yield different performance results. Important measurements of hardware AES implementations to consider are FPGA utilized resources, clock frequency, latency and throughput, see Table 2.

Table 2 reports measurements on non-pipelined hardware architectures on Virtex and Virtex-II FPGAs, suitable for CBC mode implementation. This work reports an AES architecture with the excellent performance and low resource requirements.

Table 2. Result comparison of the AES implementations

Work / Device	FPGA Resources		Clock (MHz)	Latency (Clk cycles)	Throughput (Gbps)
	Logic	Memory			
[12] – XCV200E-6	425 CLBs	-	77.80	-	0.20
[14] – XCV250-5	791 slices	4 BRAM	93	44	-
[11] – XCV400e-8	1672 LUT	-	50.20	11	0.58
[13] – Virtex-II Pro	2703 LUT	44 BRAM	196	-	1.19
[16] – Virtex-II	1125 slices	18 BRAM	-	-	1.40
This work XC2V1000	586 slices	10 BRAM	96.42	10	1.45

The general approach used in this work aimed to obtain an iterative architecture with low hardware resources utilization. The modular design was optimized in a way that the algorithm functionality was not altered (e.g. eliminating basic modules like

registers or multiplexers). Distributed memories were replaced by dual-port memories to handle data in parallel and registers were added for data multiplexing and key storage in order to reduce the critical path, resulting in less hardware that in turn results in a more efficient place & route process and higher throughput.

6 Conclusions

This work reports a hardware architecture of AES algorithm in CBC operation mode. In terms of area requirements, throughput and hardware efficiency, this architecture exhibits excellent abilities compared to the most recent AES architectures, implemented in Virtex and Virtex II devices. This work shows a simple design and an efficient architecture that requires minimal logical resources and is suitable for devices with limited silicon area.

Its performance results and low resource requirements make the architecture suitable as a module for the CRP platform, which handles several cryptographic algorithms and is applicable in secure communication systems, where devices or networks require cryptographic solutions with high flexibility and high throughput.

References

1. P. Kocher, R. Lee, G. McGraw, A. Raghunathan, S. Ravi, Security as a New Dimension in Embedded System Design, DAC 2004, June 2004.
2. F. Crowe, A. Daly, T. Kerins, W. Marnane, Single-Chip FPGA Implementation of a Cryptographic Co-Processor, Proceedings of IEEE International Conference on Field Programmable Technology (FPT'04), December 2004.
3. Ixia, IPSec Virtual Private Networks: Conformance and Performance Testing, Whitepaper, November 2003.
4. L. Quinn, P. Mehta, A. Sicher, Wireless Communications Technology Landscape, White Paper, Dell, February 2005.
5. G. Umamaheshwari, A. Shanmugan, "Efficient VLSI Implementation of the Block Cipher Rijndael Algorithm, Academic Open Internet Journal, Volume 12, 2004. Available at: http://www.acadjournal.com/.
6. G. Bertoni, J. Guajardo, C. Paar, Architectures for Advanced Cryptographic Systems, Idea Group Inc, 2004.
7. K. Gaj, P. Chodowiec, Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware, Proceedings of the 3rd Advanced Encryption Standard (AES) Candidate Conference, April 2000.
8. Y. Li, T. Callahan, E. Darnell, R. Harr, U. Kurkure, J. Stockwood, Hardware-Software Co-Design of Embedded Reconfigurable Architectures, ACM, 2000.
9. National Security Agency, Fact Sheet NSA Suite B Cryptography. Available at: http://www.nsa.gov/ia/industry/crypto_suite_b.cfm
10. Federal Information Processing Standards Publication 197, Announcing the Advanced Encryption Standard (AES), November 2001.
11. Barco-Silex, AES Encryption and Decryption BA411AES Factsheet, March 2005. Available at: http://www.barco.com/.

12. T. Liu T, C. Tanougast, P. Brunet, Y. Berviller, H. Rabah, S. Weber, An Optimized FPGA Implementation of an AES Algorithm for Embedded Applications, International Workshop on Applied Reconfigurable Computing 2005 (ARC2005), February 2005.
13. J. Lu, J. Lockwood, "IPSec Implementation on Xilinx Virtex-II Pro FPGA and Its Application", Reconfigurables Architectures Workshop (RAW), April 2005.
14. Algotronix Ltd, "AES Core Product Description", November 2004. Available at: http://www.algotronix.com/.
15. A. Hodjat, I. Verbauwhede, "A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA", IEEE Symposium on Field-Programmable Custom Computing Machines, April, 2004.
16. Helion Technology Limited, "OVERVIEW DATASHEET – Helion cores. Available at: www.heliontech.com/.
17. A. Menezes, P. V. Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
18. K. Gaj, P. Chodowiec, Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard Using Field Programmable Gate Array, Proceedings in RSA Security Conference – Cryptographer's Track, April 2001.
19. J. McCaffrey, You're your Data Secure with the New Advanced Encryption Standard, MSDN Magazine, Issue November 2003, Available at: http://msdn.microsoft.com.
20. Ignacio Algredo Badillo, René Cumplido Parra and Claudia Feregrino, "Design and Implementation of a High Performance Cryptographic Reconfigurable Platform", 2004 XIII International Congress on Computing, 13-15 Oct, Mexico (In Spanish).