Implementation of Fast Pipelined AES Algorithm on Xilinx FPGA

Chityala Prathyusha[1], P. Sharmila Rani[2]

[1]M. Tech Scholar, VLSI-SD, E.C.E Department, Teegala Krishna Reddy Engineering College, Hyderabad, India
[2]Head, E.C.E Department, Teegala Krishna Reddy Engineering College, Hyderabad, India

Abstract: ***The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data also called Rijndael. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. Hardware-based cryptography is used for authentication of users and of software updates and installations. Software implementations can generally not be used for this, as the cryptographic keys are stored in the PC memory during execution, and are vulnerable to malicious codes. Hardware-based encryption products can also vary in the level of protection they provide against brute force rewind attacks, Offline parallel attacks, or other cryptanalysis attacks. The algorithm was implemented in FPGA due to its flexibility and reconfiguration capability. A reconfigurable device is very convenient for a cryptography algorithm since it allows cheap and quick alterations. The implementation of pipelined cryptography hardware was used to improve performance in order to achieve higher throughput and greater parallelism. The AES hardware was implemented in three modules contains of the encryption, the decryption and the key expansion module.***

Keywords: Cryptography, AES, DES, FPGA, efficient encryption/decryption implementation, pipeline.

1. Introduction

In 1997, the National Institute of Standards and Technology – NIST released a contest to choose a new symmetric cryptograph algorithm that would be called Advanced Encryption Standard – AES to be used to protect confidential data in the USA. The algorithm should meet few requirements such as copyright free, faster than the 3DES, cryptograph of 128 bit blocks using 128, 192 and 256 bit keys, possibility of hardware and software implementation, among others. In 2000, after analysis by cryptography experts, it was chosen the winner: Rijndael. The algorithm was created by the Belgians Vincent Rijmen e Joan Daemen. Hardware-based cryptography is used for authentication of users and of software updates and installations. Software implementations can generally not be used for this, as the cryptographic keys are stored in the PC memory during execution, and are vulnerable to malicious codes. Hardware-based cryptography, when implemented in a secure manner, is demonstrably being superior to software-based encryption. Hardware-based encryption products can also vary in the level of protection they provide against brute force rewind attacks, offline parallel attacks, or other cryptanalysis attacks. In this work we present an efficient cryptography hardware implementation and its improvement using pipelines. The algorithm was implemented in FPGA due to its flexibility and reconfiguration capability. A reconfigurable device is very convenient for a cryptography algorithm since it allows cheap and quick alterations. Therefore, a new architecture was developed using pipelines. The implementation of pipelined cryptography hardware was used to improve performance in order to achieve higher throughput and greater parallelism.

2. AES Rijndael

In order to better understand the AES structure it is necessary to know the definition of state in the algorithm. State is the matrix of bytes that is processed between many stages, or rounds, and therefore, it will be modified in each stage. In the Rijndael algorithm, the matrix size depends on the block size being used, composed of 4 lines and Nb columns. Here, Nb is the number of bits in the block, divided by 32, since 4 bytes represent 32 bits. Since the AES algorithm uses 128 bit blocks, the state will be composed by 4 lines and 4 columns. The key is grouped by the same fashion as the data block, whereas Nk is the number of columns. Nr is the number of rounds that will be run during the algorithm. The number of runs in the

AES will depend on size of the key, where Nr will be 10, 12 and 14, for Nk equals to 4, 6 and 8, respectively. On the encryption algorithm, there will be 4 phases: AddRoundKey, SubBytes, ShiftRows and MixColumns. Nevertheless, on the last stage, the MixColumns operation is suppressed. The decryption algorithm will use the respective inverse operations: InvAddRoundKey, InvSubBytes, InvMixColumns and InvShiftRows. As it was in the encryption phase, the InvMixColumns is suppressed on the last stage of decryption algorithm. The algorithm will be explained based on its specification. The values shown in the example are presented in hexadecimal format.

A. SubBytes

Each state byte is replaced by another in the S-box (replacement Box), as indicated in Fig. 1. The replacement follows a matrix, where the first hexadecimal value corresponds to the line positioning, and the second hexadecimal value corresponds to the column positioning. The inverse operation (decryption) is called InvSubBytes, and uses an inverse S-Box. As an example, the S-box outputs 24 for the input value A6 (Figure 2 - line A, column 6). On the same way, the inverse SBox outputs A6 for the input value 24 (Figure 3 - line2, column4).

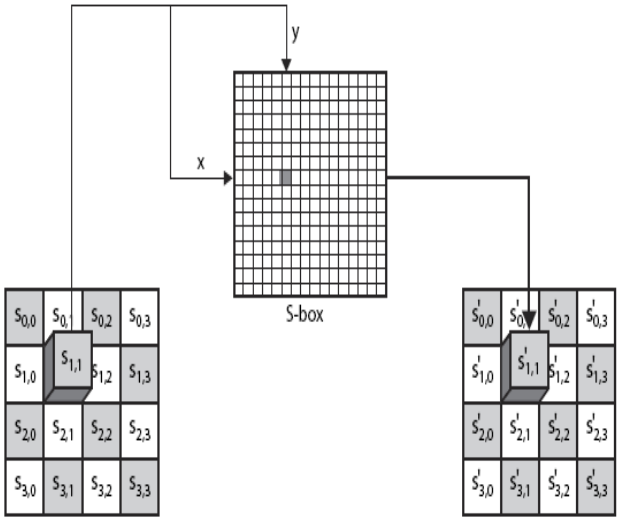

Figure 1: SubBytes operation process

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 2: S-Box

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figure 3: InvS-Box

B. ShiftRows:

It consists of a left shift on the state lines, replacing therefore their byte position, as indicated in Fig. 4. Line 0 suffers 0 shifting. Line 1 is shifted by one position and line 2 undergoes do 2 shifting positions. Line 3 is shifted by3 positions.

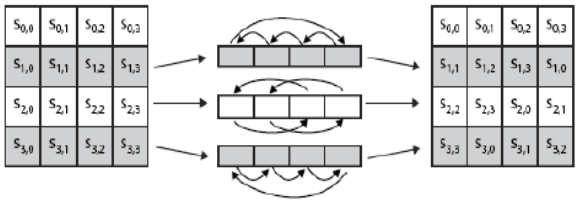

Figure 4: ShiftRows operation process.

The decryption algorithm performs the inverse operation InvShiftRows that consists of similar shifting as the ShiftRows, but shifted to the right.

C. MixColumns

In this operation, the state bytes are treated as polynomials of Galois Field algebra GF (28). The operation can be represented as a matrix multiplication, as indicated in Fig. 5, where S is the initial state and S´ is the final state, after the operation.

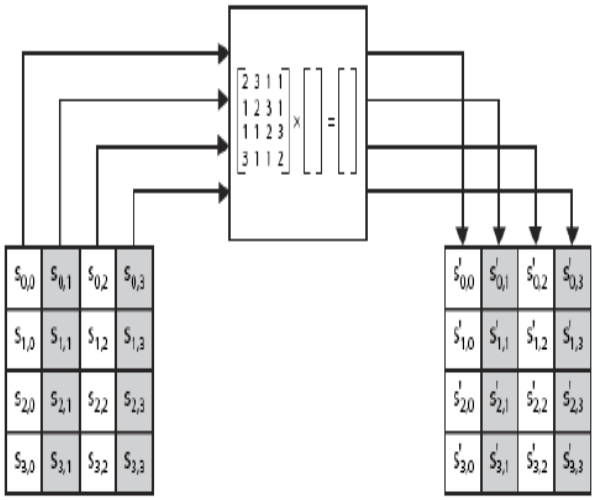

Figure 5: MixColumns operation process

The inverse operation, the InvMixColumns, consists of the multiplication using the inverse matrix. In the last round, on both the encryption and decryption algorithms, the MixColumns operation is suppressed. The C matrix (used in the encryption) and C´ matrix (used in the decryption) are

$$C = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \qquad C' = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$

D. AddRoundKey

It is an XOR operation between the state and the round key that it is generated from the main key through the Key Generation. The matrix of keys is represented by w columns or kx,y cells. AddRoundKey is used both in the encryption and decryption algorithms. The XOR is conducted on byte basis, as indicated in Fig 6, where the new byte $S'_{x,y}$ is given by $s_{x,y} \oplus k_{x,y}$.

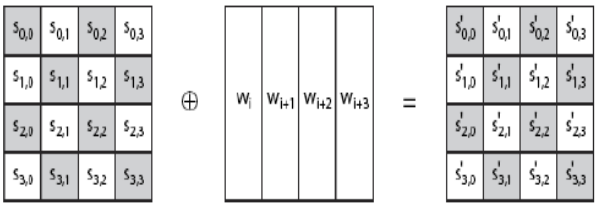

Figure 6: AddRoundKey operation process

E. Key Expansion

The Key size defines the number of rounds in the encryption/decryption algorithm, and it also defines its expansion process. Basically, the Key Expansion operation consists of three operations, as presented in Fig. 7. The first operation, RotWord, makes a one byte circular shifting on the word. The second operation, SubWord replaces each byte of the input word according to the S-Box. The third operation consists of XOR operations, as indicated in Figure 7.

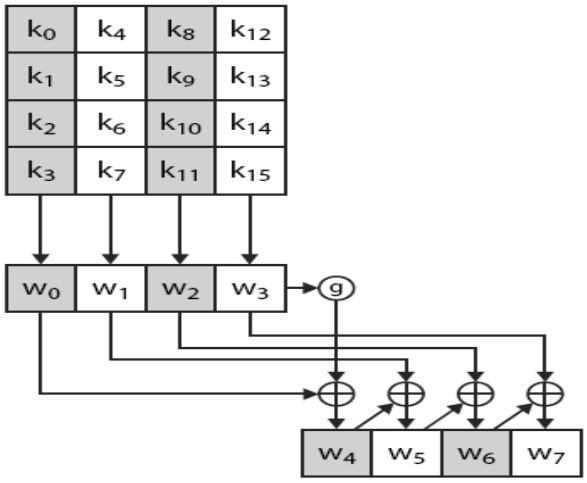

Figure 7: KeyExpansion operation process

3. Initial Implementation

The AES hardware was implemented in three modules: the encryption, the decryption and the key expansion module. The hardware is implemented as illustrated in Fig.8. It is composed of two 128 bit inputs that receive the key and the initial word to be encrypted (signals IN_INI_KEY and IN_INI_DATA).All the modules were independently tested and characterized, and therefore they can be used in any combination, according to the application. In order to conduct tests on all blocks, it was assembled a 128 bits encryption - decryption AES set in a Xilinx Spartan-3 FPGA.

After the tests on the Xilinx Spartan-3 FPGA, the hardware was also tested on a Xilinx Virtex-5 FPGA. The VHDL description implemented on both FPGAs is exactly the same, and no change was made in the VHDL description to fit any of the FPGAs. Important information is that the code is totally portable, it can be used in any FPGA since it was developed using the standard VHDL. Each module was developed independently from the others, and them they were mounted together.Figure8 shows the Encryption/Decryption block with it's I/Os. The round keys in which each key is called; and the word of each phase of encryption that is used after the calculations (feedback). The process consists of 10 rounds for 128 bits data .Each round consists of Substitute bytes, Shift rows, Mix columns, Add round key blocks. After 10 rounds the obtained is the encrypted result. The encrypted value is decrypted in another 10 rounds which are reverse to encryption. The decryption cycle consists of 10 rounds .Each round consists of Inverse Shift Rows, Inverse Sub bytes, Add round key, Inverse mix columns.

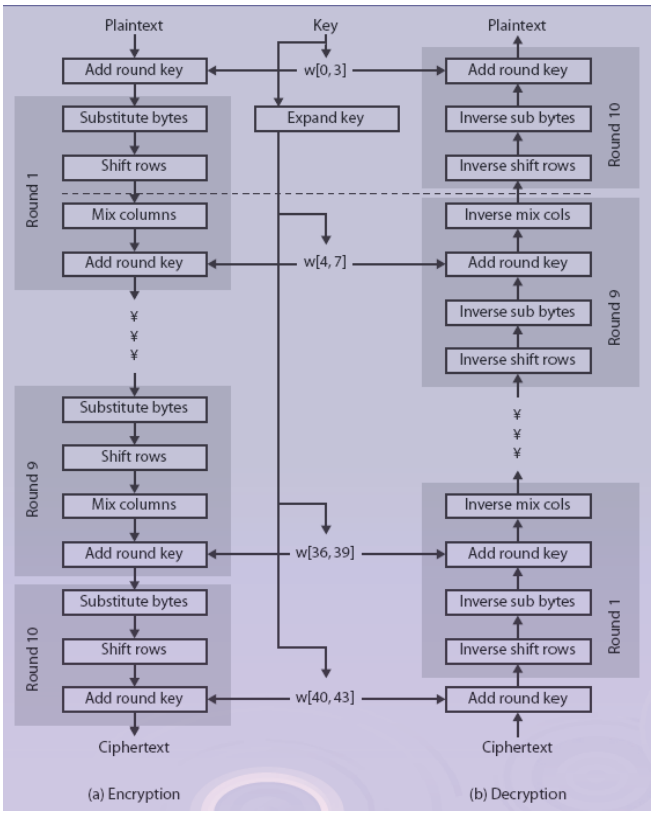

Figure 8: Block Diagram of AES Algorithm

4. Pipelined Implementation

Pipelining is one of the most efficient means of improving performance in high-end processor architectures. In order to achieve higher throughput and greater instruction-level Parallelism modern microprocessors contain deeply pipelined function units with arbitrary structural hazards. Historically, design techniques for hardware pipelines with structural hazards have been successfully developed and used in vector and pipelined supercomputers. The classical hardware pipeline design theory developed more than 3 decades ago was driven by this need. In our case, we used some levels of cryptography pipelining and greater frequencies were achieved. These levels of pipeline were implemented using Xilinx Virtex-5.Using our modular blocks (Key Expansion, Encrypt and Decrypt) we developed a pipelined cryptography hardware with one, two and five levels of cryptography, improving the efficiency of the process.

4.1 Pipelined Results Comparison

Levels of Cryptography	Input/Output Interval [ns]	Latency [ns]
1	13,5	13,5
2	7,5	15,5
5	6	28

The interval I/O represents the period of time that the data buses will be idle. This interval was decreased from 13,5ns (without pipelines) to 6 ns (with 5 levels of cryptography). It does show a great improvement in hardware efficiency by using the same FPGA board.

5. Chip Scope

Chip Scope is embedded, software based logic analyzer. By inserting an "integrated controller core" (icon) and an "integrated logic analyzer" (ila) into your design and connecting them properly, you can monitor any or all of the signals in your design. ChipScope provides you with a convenient software based interface for controlling the "integrated logic analyzer," including setting the triggering options and viewing the waveforms.

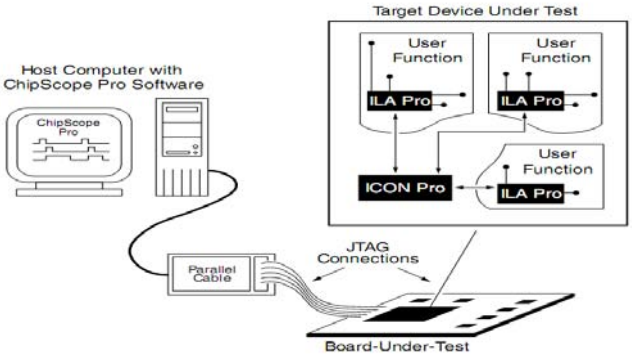

Figure 9: Chip Scope

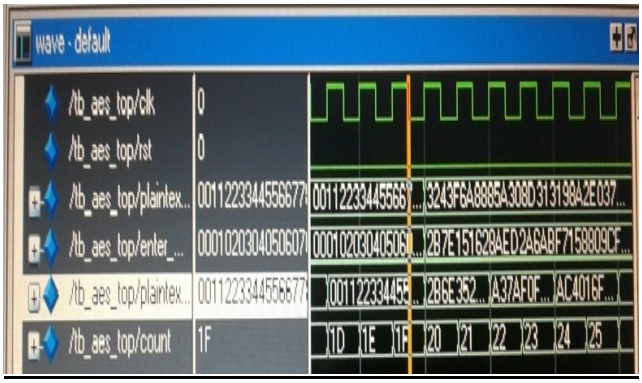

Figure 10: Simulation Waveform

5.1 Waveform Description

RST is used to reset module or clear previous data, CLK is used for the synchronization, when the raising edge of CLK is '1' then count is a counter, go on counting from '0' to '10'.The decrypted value obtained at the decryption cycle is same as the encrypted value at encryption cycle after 10 rounds for 128 bit key.

6. Conclusion

This paper presents an approach for the implementation of an AES algorithm on an FPGA using VHDL high-speed and high-density FPGAs. FPGAs features speed, accuracy, power, compactness, and cost .Configurable latency, resolution and pipelining. This article presented a fast and efficient AES cryptography hardware structure that can find many applications. The circuit implementation is very efficient and can be customized to a wide range of applications. The pipelining can be used in faster devices and buses. It represents an improvement over the non-pipeline version and can support many new applications.

References

[1] FIPS FIPS-197, Federal Information Processing Standards Publication FIPS-197, Advanced Encryption Standard (AES),
[2] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 1999.
[3] Daemen, J. and Rijmen, V., The design of Rijndael: AES — The Advanced Encryption Standard. Springer-Verlag, 2002
[4] Schneier, B., Applied Cryptography: Protocols, Algorithms and Source Code in C. John Wiley & Sons, Inc. 2nd Ed, 1996.
[5] Gomes, O. S. M.; Pimenta, T. C.; Moreno, R. L., "A Highly Efficient FPGA Implementation", 2nd Latin America Symposium on Circuits and Systems(LASCAS-2011), February 2011.
[6] Daemen, J. and Rijmen, V. A Specification for the AES Algorithm. NIST (National Institute of Standards and Technology)
[7] http://csrc.nist.gov/archive/aes/rijndael/wsdindex.html, 2010.
[8] Klima, R. E., SIGMON, N., AND STITZINGER, E. Applications of abstract algebra with Maple, CRC Press, Boca Raton, FL. 2000.
[9] C. Chien, D. Chien, C. Chien, I. Verbauwhede and F. Chang, "A hardware implementation in FPGA of the Rijndael algorithm", The 2002 45th Midwest Symp. Circuits and System (MWSCAS-2002), Vol. 1, 4 --.7 August 2002, pp. 507-509.
[10] I. Algredo-Badillo, C. Feregrino-Uribe and R. Cumlido-Parra, "Design and implementation of an FPGA-based 1.452 Gbps nonpipelined AES architecture', The 2006 Int. Con! Computational Science and Its Applications (ICCSA 2006), Lecture Notes in Computer Science, Vol. 3982 (Springer-Verlag, 2006), pp. 446--455
[11] J. Zambreno, D. Nguyen and A. Choudhary, "Exploring area/delay
[12] tradeoffs in an AES FPGA implementation", Proc. Int. Colif, Field Programmable Logic and Its Applications (FPL), Lecture Notes in Computer Science, Vol. 3203 (Springer-Verlag 2004), pp. 575-585.
[13] E. J. Swankoski, V. Narayanan, M. Kandemir and M. J. Irwin, "A parallel architecture for secure FPGA symmetric encryption", 18th Int. Parallel and Distributed Processing Symp. (IPDPS'04) - Workshop, Santa Fe, New Mexico, 26-.30 April 2004, p. 123.
[14] E. Lopez-Trejo, F. Rodriguez-Henriquez and A. Diaz-Perez, "An efficient FPGA implementation of CCM using AES", The 8th Int. Con! Information Security and Cryptology (ICJSC'05). Lecture Notes in Computer Science (Springer 2005), pp. 208-215.
[15] Arshad Aziz and Nassar Ikram, "Memory efficient implementation of AES S-boxes on FPGA", Journal of Circuits, Systems, and Computers, Vol. 16, No.4 (2007) 603--611

[16] Dur-e-Shahwar Kundi, Saleha Zaka, Qurat-Ul-Ain and Arshad Aziz, "A Compact AES Encryption Core on Xilinx FPGA", 2nd IEEE International Conference on Computer, Control & Communication (IEEE IC4-2009) Karachi, Pakistan Vol:1 pp:1-4, 2009.
[17] P. M. Kogge. “The Architecture of Pipelined Computers”, McGraw-Hill Book Company, New York, NY, 1981
[18] J.H. Pate1 and E.S. Davidson, “Improving the throughput of a pipeline by insertion of delays”, In Proc. of the 3rd Ann. Symp. On Computer Architecture, pages 159-164, Clearwater, FL, Jan. 19-21, 1976.
[19] N. Sklavos, X. Zhang, “Wireless Security & Cryptography: Specifications and Implementations”, CRC-Press, A Taylor and Francis Group. ISBN: 084938771X, 2007