Journal of Engineering and Applied Sciences 3 (7): 598-601, 2008
ISSN: 1816-949X
© Medwell Journals, 2008

Compact Hardware Implementation of FPGA Based RC6 Block Cipher

[1]Faez F. Shareef, [2]Ashwaq Talib Hashim and [2]Waleed F. Shareef
[1]Department of Electrical and Electronic Engineering,
[2]Department of Control and Systems Engineering, University of Technology, Baghdad, Iraq

Abstract: This study presents the implementation of RC6 Block cipher that involve encryption and decryption on FPGA Virtex II device with highly compact architecture through reuse the same units for the identical operation in both algorithms. The proposed design also analyzes the cipher mathematical operation to fit it into FPGA available resources.

Key words: RC6, FPGA, encryption, resources sharing

INTRODUCTION

In 1997, the National Institute of Standards and Technology (NIST) initiated a process to specify a new symmetric-key encryption algorithm capable of protecting sensitive data. RSA Laboratories submitted RC6 (Erica, 2000) as a candidate for this Advanced Encryption Standard (AES). NIST announced 15 AES candidates at the 1st AES Candidate Conference (August 1998) and asked for public comments to select 5 finalist algorithms (August 1999) MARS, RC6, Rijndael, Serpent and Twofish. On 2000 NIST announced that Rijndael (Daemen and Rijmen, 1998) had been chosen to become the AES (Nechvatal *et al*., 2000). Though, the algorithm Rijndael was eventually selected, RC6 remains a good choice for security application.

Even though RC6 does not outperform the other finalists on FPGAs as it might in software, it still performs very well. And while, it others very good performance, it does so with exceptionally compact implementation. According to Gaj and Chodowiec (2000) Mars, Rijndael and Serpent all suffer in this regard. Note further that if both encryption and decryption are to be supported then Rijndael and Serpent once again suffer badly. The design of RC6 means that many of the encryption and decryption resources can be shared (Ronald *et al*., 2000).

RC6 BLOCK CIPHER

RC6 is a fully parameterized family of encryption algorithms. A version of RC6 is more accurately specified as RC6-w/r/b, where the word size is w bits, encryption consists of a nonnegative number of rounds r and b denotes the length of the encryption key in bytes.

Since, the AES submission is targeted at w = 32 and r = 20, this version of RC6 algorithm is implemented using a 32 bits word size, 20 rounds and 32 bytes (256 bits) encryption key lengths.

Basically, RC6 operates on units of 4 w-bit words using the following basic operations. The base-two logarithm of w will be denoted by lg w (Ronald *et al*., 1998).

a+b	:	Integer addition modulo 2^w.
a - b	:	Integer subtraction modulo 2^w.
$a \oplus b$	:	Bitwise exclusive-or of w-bit words.
$a \times b$	:	Integer multiplication modulo 2^w.
a<<<b	:	Rotate the w-bit word a to the left by the amount given by the least significant lg w bits of b.
a>>>b	:	Rotate the w-bit word a to the right by the amount given by the least significant lg w bits of b.
(A, B, C, D)= (B, C, D, A)	:	Parallel assignment of values on the right to registers on the left.

Key schedule algorithm: The user supplies a key of b bytes, where $0 \leq b \leq 255$. From this key, 2r + 4 words (w bits each) are derived and stored in the array S (0;...; 2r + 3). This array is used in both encryption and decryption.

Encryption and decryption algorithm: RC6 works with 4 w-bit registers A, B, C, D which contain the initial input plaintext as well as the output ciphertext at the end of encryption. The 1st byte of plaintext or ciphertext is placed in the least-significant byte of A, the last byte of plaintext or ciphertext is placed into the most-significant byte of D. Figure 1 shows the details encryption and decryption algorithms for the RC6-5/20/256.

Corresponding Author: Faez F. Shareef, Department of Electrical and Electronic Engineering, University of Technology, Baghdad, Iraq

```
Encryption algorithm
            $B = B + S[0]$
            $D = D + S[1]$
            for $i = 1$ to $20$ dc
                {
                    $t = (B \times (2B + 1)) <<< 5$
                    $u = (D \times (2D + 1)) <<< 5$
                    $A = ((A \oplus t) <<< u) + S[2i]$
                    $C = ((C \oplus u) <<< t) + S[2i + 1]$
                    $(A, B, C, D) = (B, C, D, A)$
                }
            $A = A + S[42]$
            $C = C + S[43]$
Decryption algorithm
            $C = C - S[43]$
            $A = A - S[42]$
            for $i = 1$ to $20$ dc
                {
                    $(A, B, C, D) = (D, A, B, C)$
                    $u = (D \times (2D + 1)) <<< 5$
                    $t = (B \times (2B + 1)) <<< 5$
                    $C = ((C-S)[2i + 1]) >>> t) \oplus u$
                    $A = ((A-S[2i]) >>> u) \oplus t$
                }
             $D = D - S[1]$
             $B = B - S[0]$
```

Fig. 1: Encryption and decryption algorithm of RC6-5/20/256

HARDWARE IMPLEMENTATION

As mentioned earlier, the RC6 encryption algorithm is similar (not identical) to decryption algorithm, this means that many of the encryption and decryption resources can be shared to reduce hardware size.

Key schedule design: As this study discuss the hardware implementation of encryption and decryption algorithms only, the hardware implementation of the key schedule is not addressed. The key schedule algorithm was programmed in a non-synthesizable code of VHDL for simulation purpose.

Encryption and decryption design: According to Fig. 1, each algorithm is composed of 6 operations. The sequence of data flow through these 6 operations is explained in Fig. 2.

Pre and post-parallel assignment: In pre-parallel assignment stage, 4 internal registers, at the input, capture the input data and fed it to the next stage either directly (for encryption) or after permutated it (for decryption). On the output side the Post-Parallel Assignment carry out the same action upon the output data.

Multiplication process in f (x): The multiplication process involved in RC6 block cipher is the same for encryption

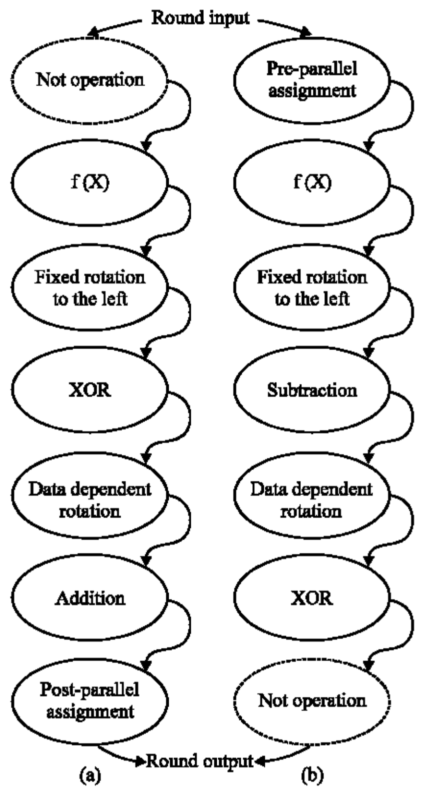

Fig. 2: Data flow diagram for (a) encryption algorithm (b) decryption algorithm

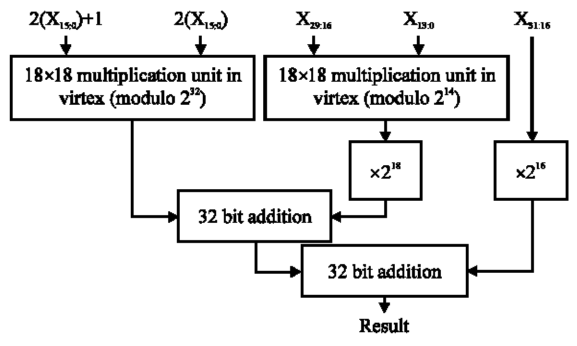

Fig. 3: Implementation of f (X)

and decryption. The key point of the implementation is the design of an arithmetic operator computing $f(X) = (X(2X+1)) \bmod 2^w$. Significant speed and area improvements are obtained by taking full advantage of small multiplier blocks available in Virtex II devices (Jean-Luc, 2002). Figure 3 shows the outline of the block diagram that performs the computation needed for f (X).

The idea is to split the 32×32 multiplication in f (X) function into 2 smaller multiplication operations (to fit into the 18×18 multiplier available in the architecture of Virtex II) and one addition operation, according to the algorithm described in Jean-Luc (2002).

The term $2\,(X_{15:0}) + 1$ is obtained by padding the term $X_{15:0}$ with 1 bit of value 1, while the operations ($\times 2^{16}$) and ($\times 2^{18}$) achieved by padding variables with 16 and 18 bits of value 0, respectively.

Rotation process: RC6 cipher involves 2 types of rotation process:

- Fixed rotation, where a 32-bit word is rotated to the left 5 times. This process is the same for encryption and decryption.
- Data dependent rotation, where a 32-bit word is rotated to the left by the amount given by the least significant 5-bits of another 32-bit word, for encryption. For the case of decryption the rotation is to the right.

A fixed rotation unit of 5-bits can easily be shared for encryption and decryption since, this type of rotation process is the same for both algorithms.

To design a unit able to perform the 2nd type of rotation process and can be shared for encryption and decryption algorithm, a Bidirectional Barrel shifter is needed. Figure 4 explain the method used develop ordinary Barrel shifter to become a Bidirectional.

The function of Bit reorder is to change the position of bit X(i) in vector $X_{31:0}$ according to the following relation:

$$X(i) = X(31-i) \text{ for } 0<i<31$$

The select input used to choose between encryption and decryption algorithm.

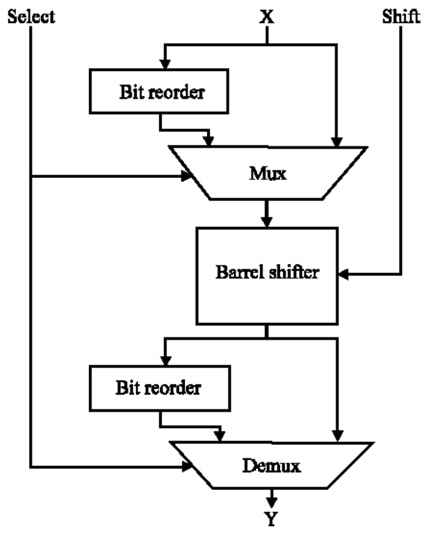

Fig. 4: Bidirectional barrel shifter

Addition, subtraction and Ex-OR: As shown in Fig. 3, the XOR and Addition operations in encryption algorithm are correspondent to Subtraction and XOR in decryption algorithm, respectively. Swapping between the 2 types of operation is also dependent on the select input.

SIMULATION RESULTS

This study presents the proposed design specifications (area and speed). Since, RC6 shows better result in ECB mode, most works focus on the pipelined architecture of the cipher and this study is not an exception. Two versions of the design are introduced; the first is iterative, while, the other is pipelined.

The iterative version consists of one encryption/ decryption module that represents 1 round of RC6 cipher. The data is fed into this module for 20 times in order to produce the final ciphertext (in addition to Pre- and Post-Whitening).

The 2nd version of the design represents fully pipelined architecture. It consists of 20 encryption/ decryption modules connected to each other as a long chain. Comparing to the iterative version, this architecture boosts the throughput as well as the size.

In order to give a better conceive to the benefit achieved from the hardware implementation that exploit resources sharing between encryption and decryption algorithms, it was compared with another implementation that perform encryption algorithm only. The latter implementation was also presented into 2 versions: Iterated and pipelined.

The comparison presents specifications (area and speed) for the combined encryption/decryption module versus the specifications of encryption only module. Table 1 hold the comparison results for the iterated and pipelined versions of the design, while Table 2 presents results of some other researchers.

Table 1: Specifications of the compact design compared to the specifications of encryption only design for iterative and pipelined versions

Design	Version	Slices	Delay (ns)	Throughput (Gb s^{-1})
Combined	Pipelined	9,485	11.752	10.89
Encryption/decryption	Iteration	1,097	59.728	0.10
	Pipelined	6,297	10.682	12.00
Encryption only	Iteration	494	19.449	0.32

Table 2: Results of some other researchers

Reference	Technology	Throughput (Gb s^{-1})
NSA team (Weeks *et al.*, 2000)	0.5 µm CMOS	2.20
Gaj and Chodowiec (2001)	XCV1000-6 (4 devices)	13.20
	Itanium (733 MHz)	0.33
Haenni (2000)	G4 (450 MHz)	0.47
Jean-Luc (2002)	Xc2V3000-6	15.20

The results presented in this study, were produced by simulate the proposed architecture in ModelSim 4.5 environment. The architecture itself was designed in the hardware description language VHDL. The target technology was a Xilinx Virtex II FPGA (XC2V10000).

CONCLUSION

The resource sharing concept shows better results when exploited with the pipelined version of the design. The proposed module of pipelined encryption/decryption offers almost the same throughput, while using less than twice of slices count compared to the pipelined encryption module only. As contrary, the iterated encryption/decryption module present <1/3rd of throughput of the iterated encryption module only and use more than twice of its slices count.

REFERENCES

Daemen, J. and V. Rijmen, 1998. AES Proposal: Rijndael.

Erica, M., 2000. CRIPTOR1.0. VLSI Implementation of the RC6 Block Cipher. http://csrc.nist.gov/archive/aes/round2/comments/20000511-emang-1.pdf.

Gaj, K. and P. Chodowiec, 2001. Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays. In: Proc. RSA Security Conf. Cryptography's Track, Springer-Verlag, pp: 84-99. http://ece.gmu.deu/crypto/publications.htm.

Gaj, K. and P. Chodowiec, 2000. Comparison of the hardware performance of the AES candidates using reconfigurable hardware. In: Proc. 3rd AES Conf., New York, pp: 40-54.

Haenni, J.O., 2000. Architecture EPIC et jeux d'instructions multimédias pour applications cryptographiques. Ph.D Thesis, Swiss Federal Institute of Technology Lausanne.

Jean-Luc, B., 2002. High throughput implementations of the RC6 block cipher using Virtex-E and Virtex-II Devices. Génie Logiciel et calcul symbolique Projet Arénaire. Rapport de Recherche, 4495. http://www.ens-lyon.fr/LIP.

Nechvatal, J., E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti and E. Roback, 2000. Report on the Development of the Advanced Encryption Standard (AES).

Ronald, L.R., M.J.B. Robshaw and Y.L. Yin, 2000. The Case for RC6 as the AES, Comdidate Conference.

Ronald, L.R., M.J.B. Robshaw, R. Sidney and Y.L. Yin, 1998. The RC Block Cipher. AIST AES propposal, http://theory.lcs.mit. edu/rivest/rc6.pdf.

Weeks, B., M. Bean, T. Rozylowicz and C. Ficke, 2000. Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms. Technical Report, National Security Agency. http://csrc.nist.gov/encryption/aes/round2/r2anlsys.htm.