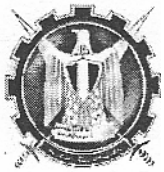
Proceedings of the 3rd ICEENG Conference, 14-16 May, 2002

Military Technical College
Kobry ElKobbah
Cairo, Egypt

ICEENG
2002

3rd International Conference
On Electrical Engineering
ICEENG 2002



A MICRO-ARCHITECTURE IMPLEMENTATION OF YAEA ENCRYPTION ALGORITHM UTILIZING VHDL AND FIELD PROGRAMMABLE GATE ARRAYS

Magdy Saeb[1], Ramy Zewail[2], Ahmed Seif [2]

ABSTRACT:

In this work, we present a hardware implementation of the encryption algorithm YAEA. The basic notion behind this encryption technique is a sequential search in a random binary file for an octet that represents the ASCII number of the plain-text character. Recording the location of this octet, one is able to assemble a file of near or far pointers to the locations of various octets. Rather inaccurately, we call this file the ciphered text. However, this file contains only pointers to the locations of the searched octets. Since there is negligible correlation between the pointer file and the plain-text characters, the method, we believe, is robust against any type of cipher attacks.
The hardware implementation utilizes Field Programmable Gate Array Technology (FPGA) and the hardware description language VHDL to conclude the realization. The basic micro-architecture of the encryption processor consists of a random binary number generator, an encryption-processing unit and a specially designed sequential memory unit. Furthermore, we demonstrate the results of various performance tests that we have carried out on the architecture. These tests were performed taking into considerations certain real-time applications.

KEYWORDS:

YAEA, encryption, data security, FPGA, VHDL, architecture.

[1] Professor, Computer Engineering Department, Arab Academy for Science, Technology & Maritime Transport, Alexandria, Egypt
[2] Student, Communication Engineering Department, Arab Academy for Science, Technology & Maritime Transport, Alexandria, Egypt

I. Introduction

In a previous work [1], Saeb and Baith have developed a conventional encryption algorithm that is called Yet Another Encryption Algorithm (YAEA). In this study, we present a micro-architecture hardware implementation of this algorithm. This implementation is achieved using the hardware description language VHDL and Field Programmable Gate Array Technology (FPGA). The approach is particularly appropriate for packet encryption for a proposed security meta-layer.

In the subsequent four sections, we provide a summary of the algorithm, and the methodology used in designing the hardware circuitry. The results are demonstrated by the timing diagrams and timing reports.

The details of the implementation process are shown in sections VI, and VII respectively. Moreover, a comparison with various encryption algorithm implementations, based on the delays encountered, is also provided in section VII. Finally, we present a summary and our conclusions.

II. The Algorithm

The algorithm is a sequential search technique using only two low-level instructions. These are CIL and XOR. The algorithm is summarized as follows:

1. In a plain text file, each character is sequentially replaced by its ASCII code.
2. Starting from a random location in a binary file, as given by a random binary number, one searches for an octet. This "sought-after-octet" has the same code sequence as the ASCII code of the plain-text character. The search is performed using circulate left instruction (CIL). If the correct octet is found, using XOR operation, its location is then recorded in an output file. This file is called, rather inaccurately, the ciphered text. However, the file contains only pointers to the location of the found octets.
3. The same procedure is then repeated for all other plain-text characters, and the addresses or pointers to their respective locations are recorded in the output file.

Depending on the degree of security level required, one can use either 16-bit pointers or 32-bit pointers. The 16-bit pointers are called near pointers. In this case, the random binary file size should not exceed 64 k bits. For higher level of security, 32-bit pointers, called far pointers, can be used with a maximum random binary file size of 4 G bits.

The decryption process is achieved by using the pointer file and the same random binary file that is available to both sender and receiver in advance. A summary of this method is shown in Fig. 1.

III. Methodology

The micro-architecture is designed taking into consideration real-time applications such as packet encryption. Recently, there has been a worldwide competition between various algorithms to select the most suitable algorithm for implementation using FPGA [3].

Our algorithm, consisting of only two low-level operations CIL and XOR and consequently appreciably short time delays, makes it a superior candidate for such applications. Moreover, the memory unit does not require address decoding, thus eliminating further delays. This is clearly demonstrated in section VII.

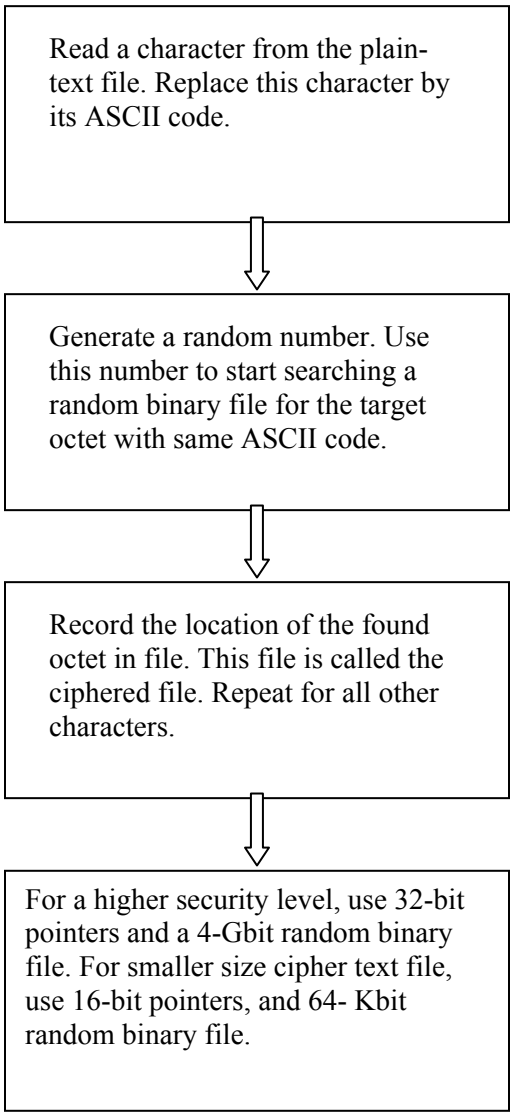


Fig. 1: Summary of the Encryption Algorithm YAEA

IV. The Micro-architecture

As shown in Fig. 2, the micro-architecture consists of three units, a random binary number generator, the processing unit, and a specially designed sequential memory unit.

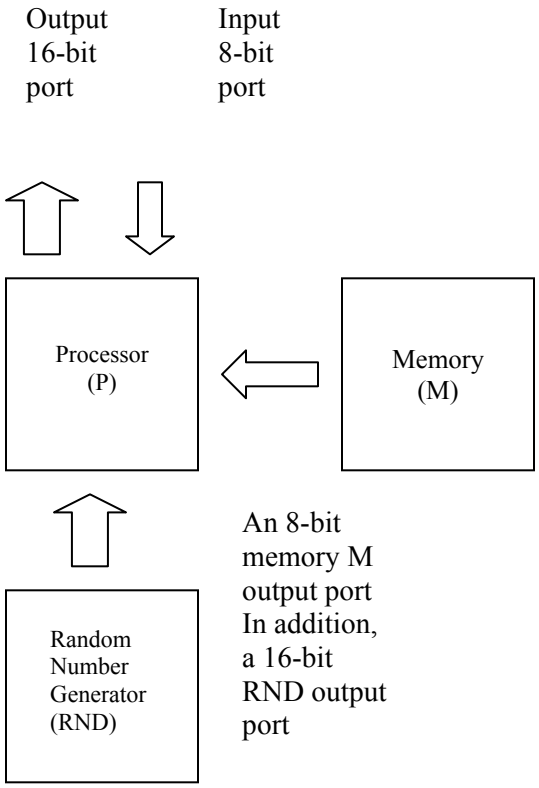


Fig. 2: The Micro-architecture of the Encryption Processor

The basic steps of the operation of this architecture can be explained as follows:

1. The ASCII code of a plain text character is used as input to the 8-bit input port of the processor. At the same time, a random binary number is acquired from the random number generator through the corresponding 16-bit second input port. The random number generator architecture is similar to a convolution encoder with all the RS flip-flop states are randomized. Details of this RND are shown in Appendix A.
2. The memory unit, organized as a series of sequential register file, is searched consecutively. Using CIL instruction and starting from a random location, one searches for an octet that represents the plain-text character. This is achieved

using the XOR instruction where a match is found if the XOR returns a 00 hexadecimal output. Evidently, the memory unit requires no address decoding and thus short memory cycle time. An internal processor counter keeps track of the bit location checked at all times.

3. The location of the found octet, as returned by an internal processor counter, is then sent to the output port of the processor. Depending on the memory configuration used, this output port is either 16-bit port for near pointers or 32-bit port for far pointers. We have implemented this micro-architecture using a 16-bit port and a memory unit consisting of 16 by 4096 subunits.
4. To decrypt the message, a reverse procedure is initiated with minor modification to the processor operation.

V. VHDL code

The architecture has been implemented employing VHDL hardware description language and Xilinx workstation for computer-aided design (CAD). The following few lines, show a sample VHDL for the processor implementation. The full project code is available from the authors upon request.

VI. Simulation Results

Fig. 3, shown below, illustrates the processor output for a given input bit sequence. Fig. 4, on the other hand, shows the random number generator-timing diagram. Lastly, Fig. 5 demonstrates the memory unit simulation results. All of the above Figs. 3, 4 and 5 were obtained using Xilinx CAD workstation. Detailed simulation results are shown in Appendix B.

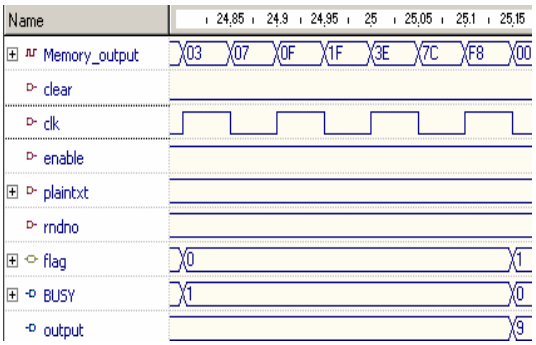


Fig.3: Processor timing diagram simulation results

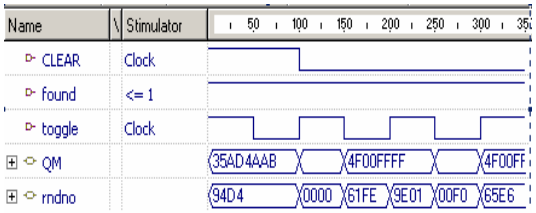


Fig. 4: The random number generator simulation results

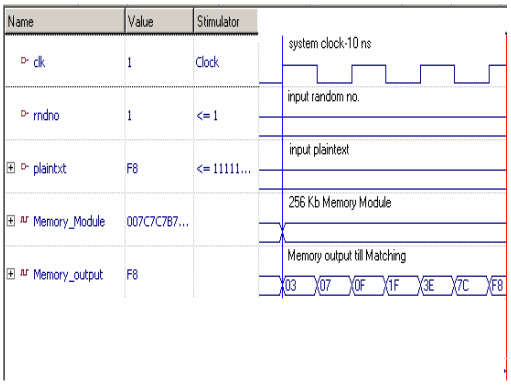


Fig. 5: The memory unit simulation results

VII. Implementation

Using XC4085 Xilinx family, and performing the necessary implementation steps, we obtained the following timing reports:
1. The Processor chip:
Maximum period: 7.530 ns at a maximum frequency of 132.802 MHz. A maximum combined path delay of 16.484 ns, and maximum net delay of 4.717 ns.
2. The Random Number Generator chip:
Maximum period: 48.291 ns at a maximum frequency of 20.708 MHz. A maximum combined delay of 53.711 ns, and a maximum net delay of 7.246 ns.
3. The Memory Unit chip:
Maximum period: 10.154 ns at a maximum frequency of 78.483 MHz. A maximum combined path delay of 11.372 ns, and maximum net delay of 3.377 ns.
The detailed output timing diagrams are shown in Appendix.
To demonstrate that this algorithm is suitable for hardware implementations in real-time applications we have surveyed the literature for comparable implementations.

The following table, recalled from [2,6], provides an assessment of various encryption algorithms based on timing reports.

It is clear that our implementation provides minimal delays in all of the shown categories. This has been partially achieved because only low-level instructions with simple sequential search are used in this algorithm. Moreover, the memory unit cycle time is greatly reduced by eliminating address-decoding delays.

Table 1: A comparison between various FPGA encryption algorithm implementations using XC4000XL, (References are given between square brackets)

Algorithm	***Speed in Mbps*** (Taken as reciprocal of maximum net delay)	***Area (CLBs)***
RC-6 [2,6]	61	1325
SERPENT [3,5]	139.3	3136
TWOFISH [2,6]	88	958
YAEA (XC4005XL)	149	135

VIII. Summary and Conclusion

Motivated by the need for a fast procedure suitable for packet-level encryption [3,4], we have provided a micro-architecture implementation of YAEA algorithm. The distinctive features of this implementation are as follows:

- An encryption processor is designed using only two low-level instructions CIL and XOR. These two instructions formed the basis of the algorithm and its hardware implementation.
- A random number generator, based on the idea of convolution encoders provided the required search random starting location. As seen from test results, this approach has produced fairly a high degree of randomization.

- A sequential memory unit, built using sets of register files, has eliminated the need for address decoding and its associated delays.

Comparing this architecture with other types of encryption algorithm implementations, we have demonstrated the superiority of our algorithm in time-critical applications. Packet-level encryption is one of these applications. As it is well known, this technique is applied when using public network links between two private local area networks. This is known as Virtual Private Networks (VPN). The implemented processor can be integrated in the system before the sender router and after the receiver router with slight modifications to the processor internal functions.

For higher degrees of data security, the process can be repeated on the resulting pointer file. That is using more than one round of encryption cycles. The number of rounds should be agreed upon by sender and receiver before transmission begins.

We believe that the algorithm and its hardware implementation offer a prominent degree of LAN security against any type of cipher attacks. Moreover, the simplicity of the approach provides a relatively minimal delay for packet-level encryption technique.

References

[1] M. SAEB, A. BAITH, *"An Encryption Algorithm for Data Security,"* Recent Advances in Information Science & Technology, N.E. Mastorakis, (editor), World Scientific Publishing Company, pp. 350-354, 1998.
[2] BRUCE SCHNEINER, Applied Cryptography: Protocols, algorithms and source code in C, second edition, John Wiley & Sons Inc., 1996.
[3] VICTOR FISCHER, *"Realization of the Round 2 AES Candidates using Altera FPGA,"* www.micronic.sk, 2000.
[4] Ronald Rivest, *"The RC5 Encryption Algorithm,"* Dr. Dobb's Journal, #226, pp.146-148, January, 1995.
[5] Kris Gaj and Pawel Chodowiec, George Mason University, *"Comparison of the hardware performance of the AES candidates using reconfigurable hardware,"* http://ece.gmu.edu/crypto-text.htm, 2000.
[6] Alfred J. Menezes et al., Handbook of Applied Cryptography, CRC Press, 1997.

Appendix A:

THE RANDOM NUMBER GENERATOR (RND) ARCHITECTURE

The random number generator is based on the architecture of convolution encoders. It is constructed using a binary tree of RS flip-flops. The number of levels of the tree depends on the number of bits required to generate the random number. The initial states of all flip-flops are set randomly. These random states can be linked to some random events such as certain operating system flags.

A basic two-level module of the RND is shown below.

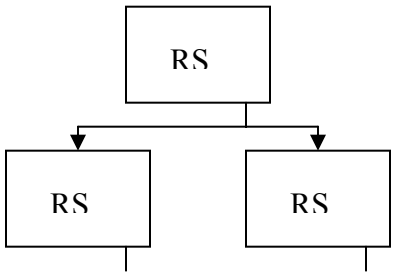


Fig. 6: The basic module of the RND

Appendix B:

THE TIMING DIAGRAMS AND CLBS AREA XILINX REPORTS

A part of the timing implementation results is given as follows:

DESIGN SPECIFICATION:
Target Device: xc4005xl
Target Package: pq208
Target Speed: -09
Mapper Version: xc4000xl -- C.16

The Encryption Module:

Timing summary:
-The Average Connection Delay for this design is: 1.510 ns
-The Maximum Pin Delay is: 4.717 ns

-The Average Connection Delay on the 10 Worst Nets is: 2.376 ns

Design statistics:
Minimum period: 7.530ns (Maximum frequency: 132.802MHz)
Maximum combinational path delay: 16.494ns

Device utilization summary:
Number of CLBs: 11 out of 196 5%
CLB Flip Flops: zero
CLB Latches: one
Four input LUTs: 13
Three input LUTs: two
Number of bonded IOBs: 34 out of 112 30%
IOB Flops: zero
IOB Latches: eight
Number of BUFGLSs: 2 out of 8 25%
Total equivalent gate count for design: 124
Additional JTAG gate count for IOBs: 1632

The Memory Module:
Timing summary:

-The Average Connection Delay for this design is: 1.894 ns
-The Maximum Pin Delay is: 3.397 ns
-The Average Connection Delay on the 10 Worst Nets is: 2.497 ns

Design statistics:
Minimum period: 10.154ns (Maximum frequency: 98.483MHz)
Maximum combinational path delay: 11.372ns
Maximum net delay: 3.397ns

Device utilization summary:
Number of CLBs: 6 out of 196 3%
CLB Flip Flops: eight
CLB Latches: one
Four input LUTs: 10
Three input LUTs: 1
Number of bonded IOBs: 12 out of 112 10%
IOB Flops: zero
IOB Latches: zero
Number of clock IOB pads: 1 out of 12 8%
Number of BUFGLSs: 2 out of 8 25%
Total equivalent gate count for design: 166
Additional JTAG gate count for IOBs: 576

The Random Number Module:
Timing summary:

-The Average Connection Delay for this design is: 2.739 ns
-The Maximum Pin Delay is: 7.246 ns
-The Average Connection Delay on the 10 Worst Nets is: 5.357 ns

Design statistics:
Minimum period: 48.291ns (Maximum frequency: 20.708MHz)
Maximum combinational path delay: 53.711ns
Maximum net delay: 7.246ns

Device utilization summary:

Number of CLBs: 95 out of 196 48%
CLB Flip Flops: zero
CLB Latches: 31
Four input LUTs: 170
Three input LUTs: 12
Number of bonded IOBs: 80 out of 112 71%
IOB Flops: zero
IOB Latches: 0
Number of BUFGLSs: 1 out of 8 12%
Total equivalent gate count for design: 1229
Additional JTAG gate count for IOBs: 3840