Implementation of Enigma Machine Using Verilog on an FPGA

Deniz Engin
Istanbul Technical University
Istanbul, Turkey
Email: enginde@itu.edu.tr

Berna Ors
Istanbul Technical University
Istanbul, Turkey
Email: siddika.ors@itu.edu.tr

***Abstract*—The Enigma machine was used in the twentieth century for enciphering and deciphering secret messages. It has been implemented on an Field Programmable Gate Array (FPGA) in this paper. The Enigma machine is kind of poly alphabetic cipher. In other words, even if input is the same letter, output can be a different letter. Therefore, the Enigma machine can not be broken by using easy methods of cryptanalysis as frequency analysis.**

The Enigma machine in this paper has common properties with the Enigma machine which was used in the Second World War. The only difference is that it has been implemented on an FPGA rather than a mechanical implementation. Verilog hardware description language (HDL) has been used in this implementation.

***Keywords*—The Enigma Machine, Digital System Design, Cryptography**

I. Introduction

The Enigma machine was an electromechanical device and was used for ciphered communication [1]. Arthur Sherbius invented the Enigma machine and he took out his first patent in 1918. Different types of Enigma machines were produced between 1918 and 1938. The German military bought about thirty thousand Enigma machines and these machines had been sent distinctive areas during wars for secret communication. The Enigma machine is known by its fame in the Second World War. Because of breaking Enigma machine, course of events changed during the Second World War. Each type of Enigma machine has different parts. For instance, first Enigma machine which is basic while the Enigma machine which was used the Second World War is complicated. Day by day, the Enigma machines had been improved encryption features.

This project has been designed based on report titled "Enigma code breaking using a Field Programmable Gate Array" [2]. In this report, authors illustrated simple implementation of the Enigma machine on an FPGA and the brute-force breaking of the Enigma machine using Java.

In this work, some properties have been added such as plugboard and key settings, therefore hardware design has been altered. Plugboard settings can be defined as one of key settings which are important inputs for encryption. These settings were changed for each day during the Second World War. In previous project [2], key settings was not inputs, therefore key settings were identified once during the implementation. In this project, key settings can be altered for each encryption. Key settings are not only inputs but also they are used to configure the Enigma Machine before encryption.

This paper is organised as follows: Specifications of the Enigma machine and key settings have been described in Section II. Section III shows that how the Enigma machine works and implementation of the Enigma Machine on an FPGA. Simulation results are given in Section IV. Finally, conclusions are in Section V.

II. The Enigma Machine

The Enigma machine consists of some components which are a 26 letters keyboard, a 26 letters lampboard, an entry wheel, three rotating wheels (rotors), a reflector, and a plugboard (stecker). These components are illustrated in Fig. 1. There are 26 letters on the keyboard. When a letter is pressed on the keyboard, an electric signal is created on the keyboard and later another letter is sent to plugboard. The last part of the Enigma machine is lampboard which is connected to the plugboard and it demonstrates cipher letter. The entry wheel connects to the plugboard to the right wheel. It does not perform any changes. Three wheels come from the Enigma machine scrambler. Each wheel has 26 contacts which represent letters of the alphabet on each side. The reflector exchanges each letter in pair which is determined before encryption. There are some types of reflector such as B, C, and D. Due to reflector, a letter can never turn into itself, this feature makes easier cryptanalysis. In plugboard, number of cable which depends on key can be most 13. Some of the letters on the plugboard can be wired up to other letters. If one of them is not plugged to another letter, plugboard step skip. In other words, a letter can never change on the plugboard and then the letter is sent to next step which can be lampboard or wheel [1], [3], [4].

A. Key Settings

A key consists of some properties which are ordering of three wheels and initial wheel settings. Moreover, plugboard settings and type of reflector can be defined as key settings for encryption. According to these, encryption results change.

Firstly, three wheels can be positioned in six ways which are "012", "021", "102", "120", "201", "210". For example, order of wheels has been chosen as "210" in Figure 2. Secondly, wheel initial positions can be chosen from letters.

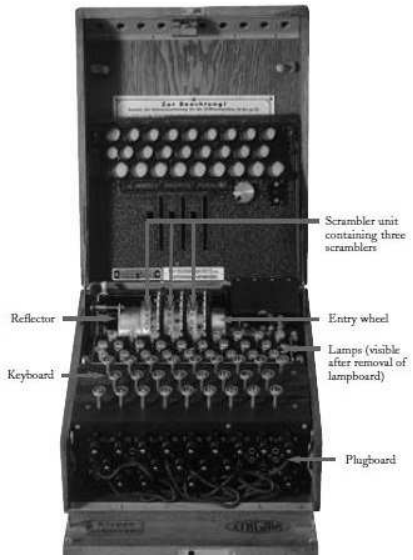

Figure 1: The Enigma Machine [1]

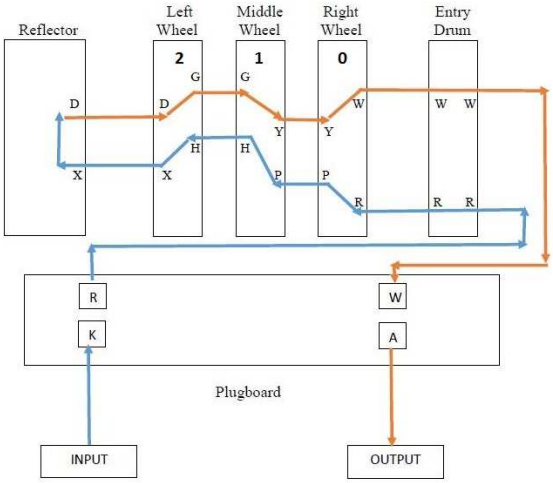

Figure 2: Block Diagram

Also, people who use the Enigma machine decide how many wire is used for encryption and which letters can be pair on the plugboard. Type of reflector is another important setting for the Enigma machine. In the Second World War, reflector B was used. As wheel, types of reflector is determined during manufacturing [1].

III. Implementation

As shown in Figure 2, the block diagram explains the algorithm in this project. First of all, the input letter comes at the plugboard. If the input letter is one of the letter pairs on the plugboard, alteration is performed. Otherwise, the input letter without alteration is sent to entry drum which transmits the letter to the right wheel. After essential alteration is performed on the right wheel, this wheel rotates one position. If the right wheel turns 26 times, the middle wheel turns once. Accordingly, if the middle wheel turns 26 times, the left wheel turns once. Therefore, the left wheel turns slowly. The letter is proceed throughout three wheels and performed alteration. When the letter comes at the reflector, it is swapped with the its pair and later it turn back the same path. Wheel positions are never changed in the return path. In this implementation, each letter has been linked to a number which defines as macro. For instance; A has been linked to zero, B has been linked to one, C has been linked to two and so on. In addition, new module has been written in order to display result as letter. This module assigns each number to ascii code. Thanks to this module, simulation results are meaningful that is input and output can be letter. Plugboard, wheel, and reflector are submodules in this project. Enigma is also the top module. All wheels have distinctive enciphering. For example, A has been linked to B on the left wheel, on the other hand, A has been linked to E on the right wheel. As mention above, encryption rules of the wheels are settled on during manufacturing and never change. Base wheel settings have been defined for each wheel in the top module, therefore same wheel module has been used for three wheels. Wheel orders and initial positions are determined by user. According to these, base wheel settings are applied and then the letter is enciphered in the system. Reflector has been chosen type B and keyed to the type B. Plugboard settings have been specified randomly. In this project, implementation of the

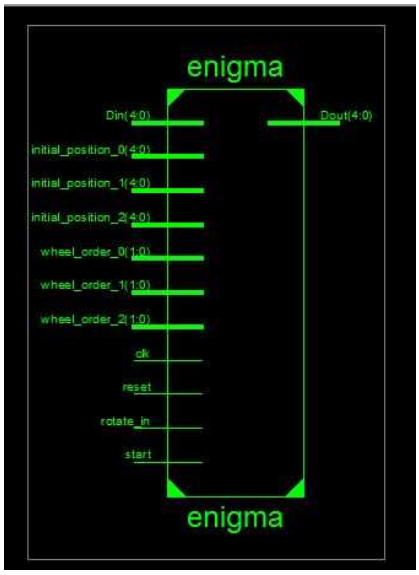

Figure 3: RTL Schematic 1

Enigma machine has been realised. Register Transfer Level (RTL) Schematic demonstrates inputs and output of the overall system in Figure 3. Key settings which are wheel order, wheel positions and input letter have been defined as input. Besides, output has been defined as encrypted letter. As illustrated in Figure 4, components of the Enigma machine can be shown.

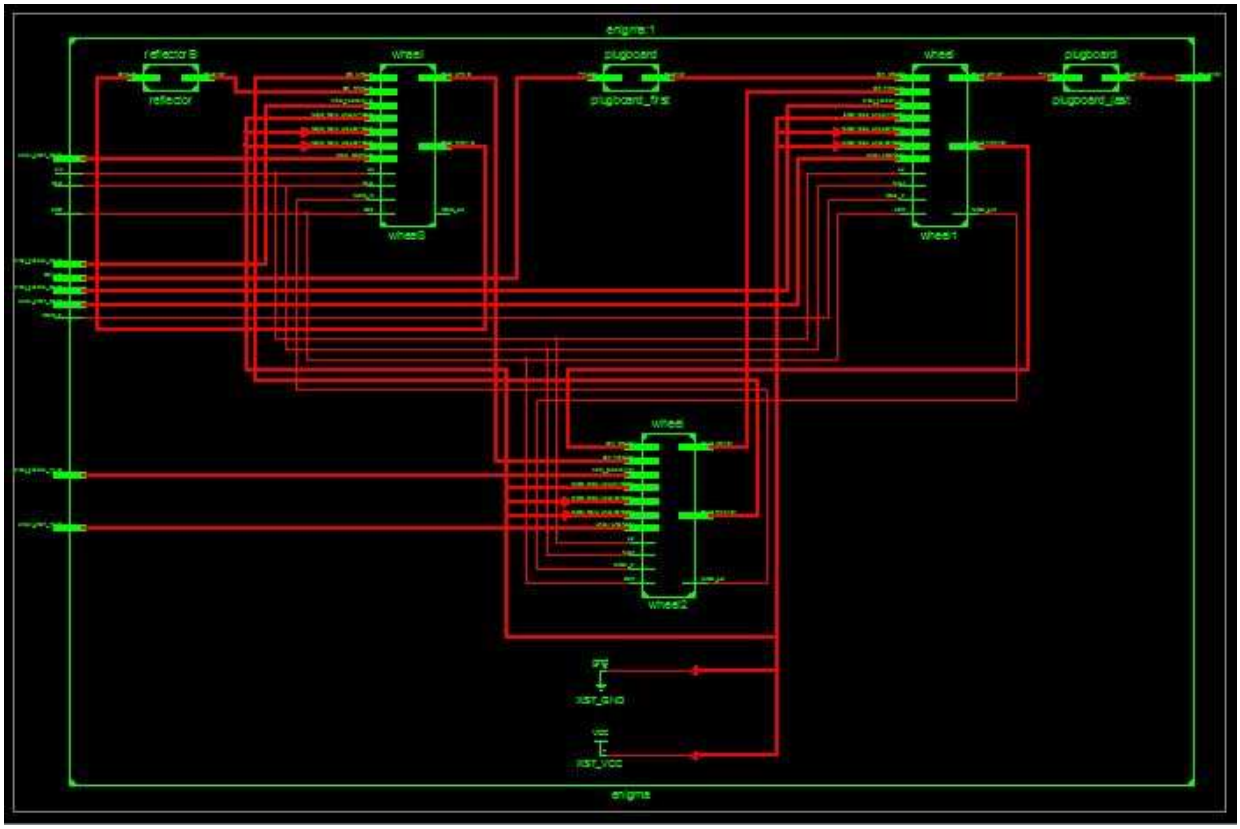

Figure 4: RTL Schematic 2

IV. Simulation Results

There are four main modules which are plugboard, reflector, enigma, wheel in this project. Different modules of the Enigma machine have been tested separately and then overall system has been tested.

In this project, Xilink ISE Design Suite 14.7, Verilog HDL and ISIM simulator have been used and Spartan 3E has been chosen for implementation. Device utilization summary of the design has been generated and it is demonstrated in Figure 5. Synthesis results illustrate that estimated footprint and blocks before running on the board.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	354	960	36%
Number of Slice Flip Flops	140	1920	7%
Number of 4 input LUTs	651	1920	33%
Number of bonded IOBs	427	66	646%
Number of GCLKs	1	24	4%

Figure 5: Device Utilization Summary

A. Plugboard Test

Plugboard settings have been chosen that B has been linked to T, G has been linked to Z, M has been linked to U, Q has been linked to Y. Pin has been defined as number of the input letter, Pout has been defined as number of the output letter. Letter displays the output letter. If the input letter is G which refers to 6, the expected output letter is Z which refers to 25. Simulation results demonstrate that the output letter is Z in Figure 6.

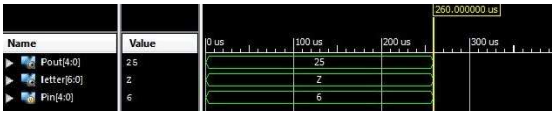

Figure 6: Plugboard simulation result

B. Reflector Test

As mention in the implementation section, Reflector B has been chosen for this project. According the this reflector, if the input letter is A, the output letter must be Y. Similarly, if the input letter is Y, the output letter must be A. Din has been defined as number of the input letter, Dout has been defined as number of the output letter. Letter displays the output letter. If the input letter is C which refers to 2, the expected output letter is U which refers to 22. Simulation results demonstrate that the output letter is U in Figure 7.

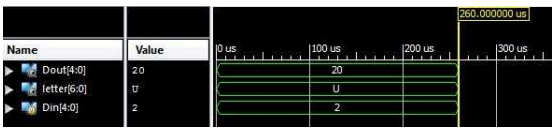

Figure 7: Reflector simulation result

C. Rotate Test

If the right wheel rotates 26 times, the middle wheel rotates once. In other words, when 26 letters are enciphered on the Enigma machine, the middle wheel rotates once.

Two variables which are "rotate_in" and "rotate_out" have been used to control rotating. "rotate_in" and "rotate_out" have been defined as one bit. If "rotate_in" is one, the wheel rotates once. On the other hand, if the wheel turns 26 times, "rotate_out" is one. Then, next wheel rotates once. "rotate_out" of the right wheel has been connected to "rotate_in" of the middle wheel. In a similar way, "rotate_out" of the middle wheel has been connected to "rotate_in" of the left wheel. Position which is the top of simulation results represents the right wheel position. "rotate_in" and "rotate_out" belong to the right wheel. Second position which is the middle of the simulation results represents the middle wheel position. In addition, "rotate_in" and "rotate_out" which are below in simulation results belong to the middle wheel.

As demonstrated in Figure 8, simulation results show that if the right wheel turns 26 times, "rotate_out" which belogs to the right wheel is one. Then, "rotate_in" which belongs to the middle wheel is one. Moreover, position alters.

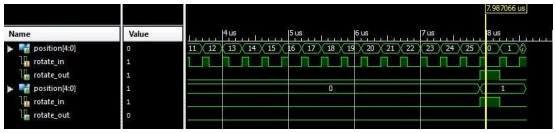

Figure 8: Rotate simulation results

D. Wheel Test

The right wheel has been tested in this section. One module has been written for all wheels. If the one wheel works properly, all wheels work properly that can be assumed. There are two encryption on the wheel. Correspondingly, there are two inputs and two outputs. "Din_left" has been defined as input of forward encryption and "Dout_right" has been defined as output of forward encryption. Similarly, "Din_right" has been defined as input of back encryption and "Dout_left" has been defined as output of back encryption. "rotate_in" and "rotate_out" have been explained in the rotate test subsection.

If "Din_left" is A which refers to 0 and the position is B which refers to 1, "Dout_right" must be K which refers to 10. If "Din_right" is G which refers to 6 and the position is B which refers to 1, "Dout_left" must be K which refers to 10. In this test, "Dout_left" and "Dout_right" are same. However, this is coincidence. As illustrated in Figure 9, simulation results are correct. In addition to this, when the "rotate_in" is one, position is changed. Only "rotate_in" which belongs to the right wheel has been defined as an input. For each input letter, it must be one.

E. Enigma Test

Enigma is the top module and Figure 10 demonstrates the overall system results. Din has been defined as the input letter. Dout has been defined as the output letter which is encrypted

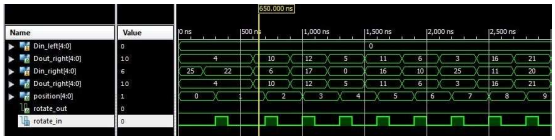

Figure 9: Wheel test simulation results

letter. Letter displays the output letter. Key settings which are initial positions, wheel orders and plugboard pairs have been chosen randomly. In this settings, Din has been chosen A and then it was encrypted. Dout is S which is refer to 18.

"initial_position_0" has been defined as initial position of the right wheel and chosen B which refers to 1.

"initial_position_1" has been defined as initial position of the middle wheel and chosen B which refers to 1.

"initial_position_2" has been defined as initial position of the left wheel and chosen B which refers to 1.

"wheel_order_0", "wheel_order_1", and "wheel_order_2" have been defined as order which can be 0, 1, or 2. According to this order, each wheel is located in the Enigma machine. As mention in the previous sections, "rotate_in" which belongs to the right wheel and it has been defined as an input.

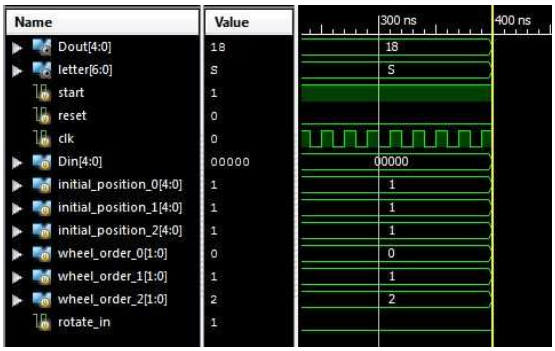

Figure 10: Enigma simulation results

V. Conclusion

As a result of implementation, a cryptosystem has been realised on an FPGA. In addition, simulation results illustrate that cryptosystem works properly. Implementation of the Enigma machine is previous project [2] which has been altered. Thus, the Enigma machine which was used to the Second World War has been implemented.

References

[1] S. Singh, *The code book : how to make it, break it, hack it, crack it*. Delacorte Press, 2002.

[2] C. van Reeuwijk, "Enigma code breaking using a field programmable gate array," Delft University of Technology Parallel and Distributed Systems, Report Series PDS-2002-001, 2002.

[3] A. M. Turing and B. J. Copeland, *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma*. Oxford University Press, 2004.

[4] R. F. Churchhouse, *Codes and Ciphers: Julius Caesar, the Enigma, and the Internet*. Cambridge University Press, 2001.