ENIGMA MACHINE REIMPLEMENTATION ON VHDL

GROUP B-2

ALTHAF NAFI ANWAR	**2106634881**
ARMOND HARER	**2106634710**
IBRAHIM RIJAL	**2106633323**
RAYHAN AKBAR ARRIZKY	**2106632655**

PREFACE

In this report, we present our project on the reimplementation of the Enigma machine in VHDL. The Enigma was a German encryption device used during World War II to protect military communications. It employed a complex system of rotors and reflectors to encrypt and decrypt messages, making it difficult to break.

Our project aimed to recreate the functionality of the Enigma machine using VHDL, a hardware description language used to design and simulate digital systems. We focused on accurately implementing the Enigma's rotors and reflectors, as well as its plugboard for additional encryption.

Through this project, we were able to gain a deeper understanding of the Enigma machine and the challenges involved in breaking its code. We also learned about the capabilities of VHDL in modeling and simulating complex digital systems.

We would like to give a shoutout to our laboratory assistants and the lecturer for providing the knowledge and support that made this project possible. We are grateful for the knowledge and support provided by our laboratory assistants and the lecturer, and we would like to give them a big shoutout for their contributions to this project.

Depok, December 09, 2022

Group B-2

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

An Enigma Machine is one of the most sophisticated encryption machines in the 20th century. It was commonly used during World War II by the German Military to send an encrypted message to one another without the enemies being able to know what the message actually contains. This encryption was then cracked by Allied forces thanks to earlier decryption attempts by Poland under the leadership of mathematician Marian Rejewski in the early 1930s. However, there weren't any notable developments until Bletchley Park, a secret British intelligence facility, discovered a breakthrough thanks to the Bombe machine, a machine designed by Alan Turing and Gordon Welchman which can decrypt Enigma text faster than it would take an average human

Enigma is a combination of a mechanical and electrical subsystems, and is divided into several components. The keyboard, which serves as the input to the machine, a plugboard which scrambles user-determined sets of letters into each other, rotors which serve as the main encryption via permutation, a reflector at the end of the sets of rotors so the electrical signal can pass through the set of rotors once again, and the lampboard which serves as an output. To reinforce the encryption provided by the rotors, they are configured to rotate every certain amount of inputs (rightmost rotor rotates for every input, second closest rotor to the right rotates every time the first rotor does a full rotation at 26 inputs, and so on)

Our project aims to recreate the functionality of the Enigma machine using VHDL, a hardware description language used to design and simulate digital systems. VHDL allows us to accurately model the Enigma's rotors and reflectors, as well as its plugboard for additional encryption. By reimplementing the Enigma in VHDL, we hope to gain a better understanding of how it worked and how it was eventually broken.

In this report, we will provide a detailed overview of our project. We believe that this project not only serves as a valuable educational tool, but also offers insight into the design of historical encryption devices and the challenges involved in breaking them.

1.2 PROJECT DESCRIPTION

The goal of this project is to implement the Enigma machine in VHDL. This will involve designing a digital circuit that can replicate the functionality of the original Enigma machine, including the rotors, reflectors, and plugboard. The circuit will be able to accept an input message and encrypt it according to the rules of the Enigma machine, as well as decrypt a previously encrypted message. The project will include simulation and testing of the circuit to ensure correct functionality. This implementation of the Enigma machine in VHDL can be used for educational purposes, historical recreation, or even in modern cryptography applications.

1.3 OBJECTIVES

The objectives of this project are as follows:

1. To emulate the working mechanisms of Enigma in VHDL
2. To gain a better understanding on how Enigma Machine works.
3. To provide an educational tool for learning about digital system design and historical encryption devices.
4. To demonstrate the capabilities of VHDL on simulating complex digital systems.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	**Responsibilities**	**Person**
Plugboard	Emulating the plugboard section of the Enigma encryption machine via VHDL	Armond Harer

Rotor and Reflector mechanism	Emulating the rotor and reflector sections of the Enigma via VHDL	Althaf Nafi Anwar Ibrahim Rijal
Keyboard and Lampboard	Emulating the Keyboard and Lampboard Section of the Enigma machine via VHDL	Rayhan Akbar Arrizky
Main Program	Creating the VHDL program which will combine all the components together	Rayhan Akbar Arrizky

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

The tools that are going to be used in this project are as follows:

- ModelSim
- Intel Quartus
- IDE (such as VSCode, notepad++, and so on)
- Computers and laptops

2.2 IMPLEMENTATION

Our group's method of emulating Enigma via VHDL is to first create the program and its components using a text editor (for this application Visual Studio Code was mostly used), then running the programs via a compiler and analyzing the results (ModelSim and Intel Quartus were used for compiling and analyzing the output). Each group member worked on

certain components of the Enigma VHDL code based on the roles assigned (shown above), and uploaded the code to a Github repository for other members to analyze, rectify, and improve the code, while integrating said components into the main program.

In Intel's Quartus software we can use the "RTL Analysis" feature to obtain a simulated schematic and the state diagram of our device, while in Modelsim we can use the waveform feature to test whether or not the outputs are correct based on a certain combination of user-adjustable inputs. Both software are used in conjunction to ensure that our VHDL program is functional and that it's inputs and outputs are accurate.

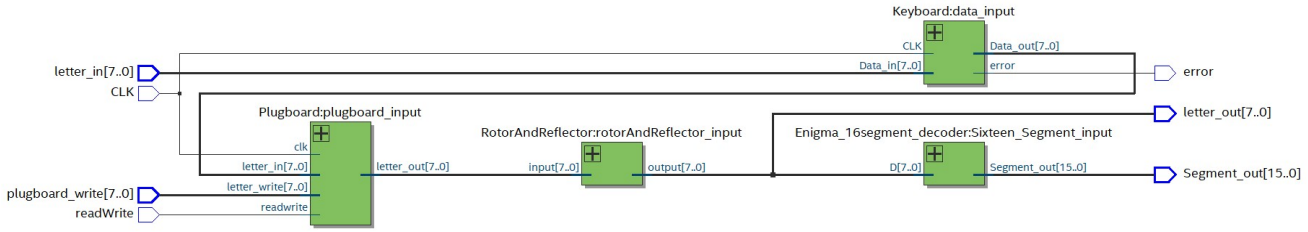


Fig 2.2.1. Schematic

Our design works by receiving a 8-bit binary ASCII input signal in the form of `std_logic_vector(7 downto 0)`. The program then passes it towards the entire mechanism and ends in a 16-Segment display encoder which can be displayed in 16-Segment display. Users can also change the enigma configuration by modifying the constants located in the configuration file to get different letter encryption.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

We validated our design by running two instances simultaneously, one to encrypt and another to decrypt. We give input to the former and passes the ouput directly to the latter. If both input and output matches in pair then our design is considered a success.

Much like how the VHDL code for our main project is divided into several components with one main component which unifies it together, each component has their own testbench, including the main component itself. With these testbenches we can examine and analyze how each of Enigma's components worked and compare it to our VHDL emulation. By analyzing the outputs of each component we can make sure that individual components are working as intended so as to avoid confusion later on in making the final top level component.

3.2 RESULT

After testing the several components that we had, by running simulations and synthesizing the code in Quartus and Modelsim we were able to get the waveforms and schematics of every main component in the project.

3.2.1 Rotor and Reflector

The schematic below shows the RTL model of the rotor and reflector component generated from Quartus Prime.

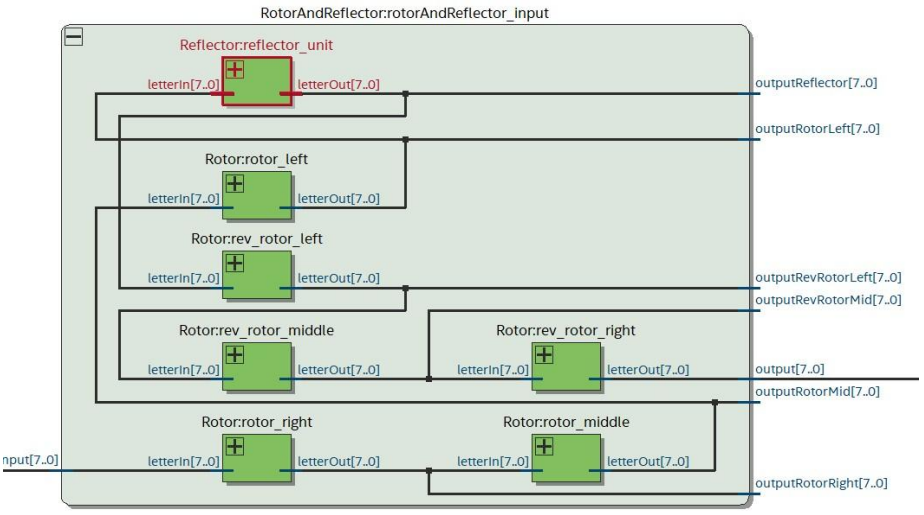


Fig 3.2.1 RTL schematic of the Rotor and Reflector component

As shown in the picture above there are two main types of components, a rotor and a reflector. In this segment of the enigma machine, the input signal first goes into the rightmost rotor and passes two other rotors before reaching the reflector, and then the sequence repeats after the signal passes through the reflector by going through the same three rotors although in the reverse direction. That sequence is captured directly in the code and can be seen in the schematic above.

The picture below shows the result of the simulation done in modelsim of the rotor and reflector testbench.

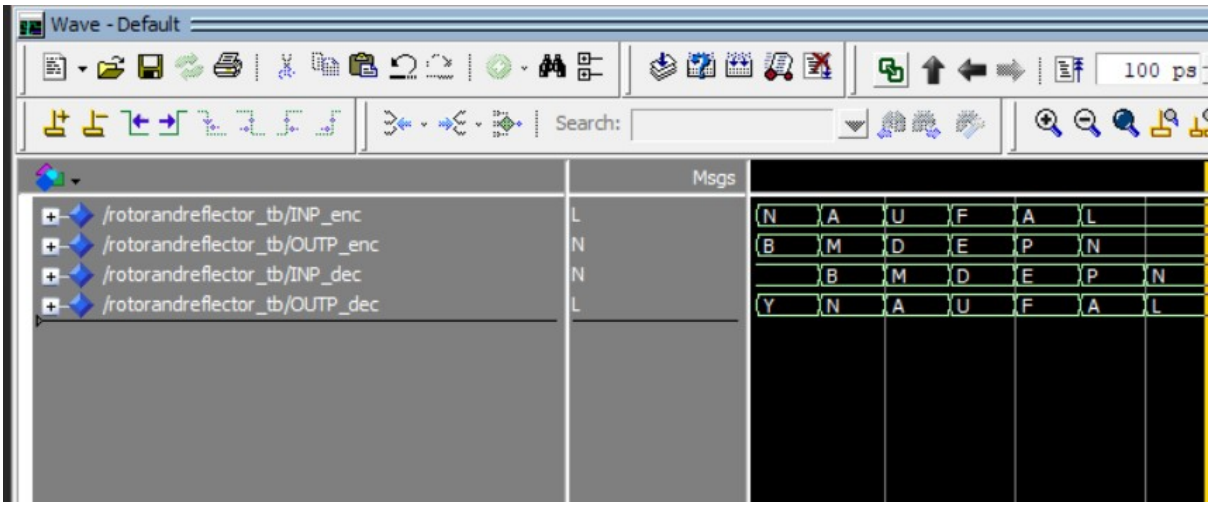


Fig 3.2.2 Waveform of the Rotor and Reflector component

As can be seen in the picture above, the rotor and reflector components are working as intended. Take for example the first case that maps the input, a letter A, to

a letter O. If you examine another case when it takes an input of letter O, the component maps it to a letter A. This behavior of mapping pairs of letters together is what enables the enigma machine to do encryption and decryption at the same time. Although for rotors this behavior is only consistent for cases with the same exact state.

3.2.2 Plugboard

The schematic below shows the RTL model of the plugboard component generated from Quartus Prime.

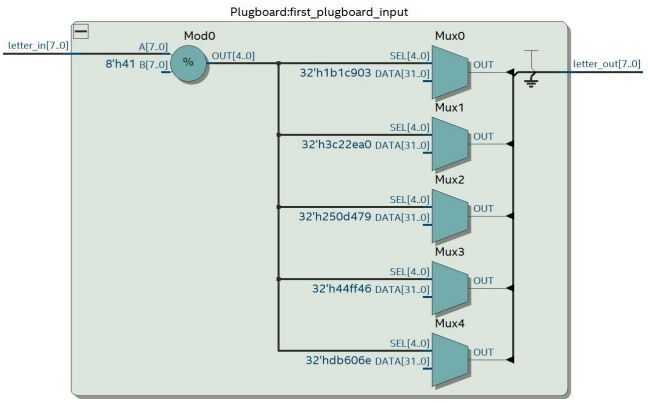


Fig 3.2.3 RTL schematic of the Plugboard

The picture below shows the simulation result of the plugboard in modelsim.

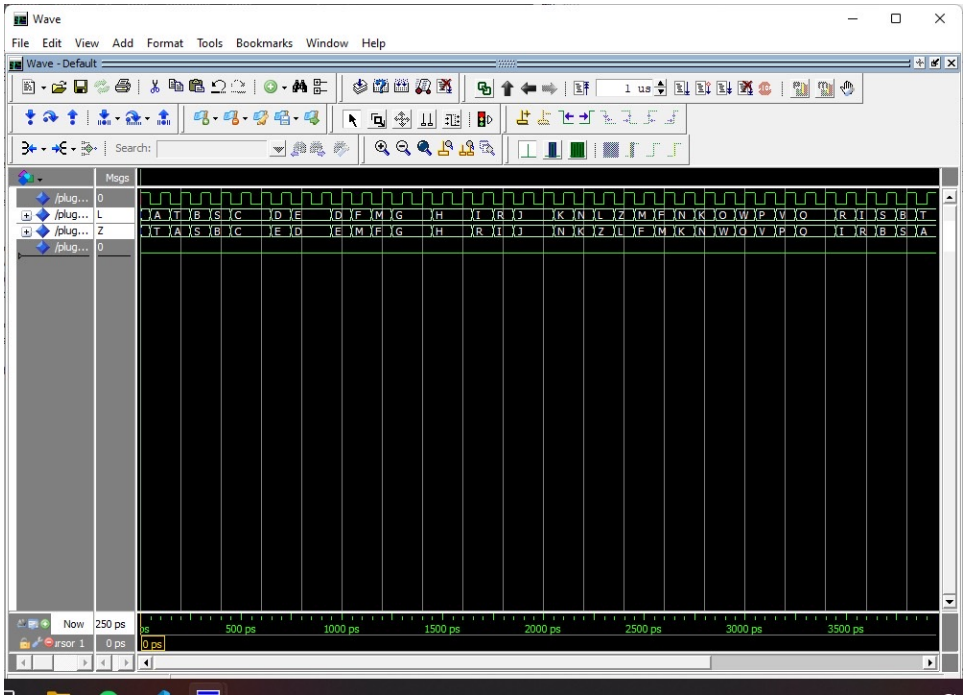


Fig 3.2.4 Plugboard waveform simulation result

As can be seen on the simulation result above, the plugboard component works by swapping letter pairs that's defined in the configuration file. The letter A is mapped to the letter T in this configuration, and it can be seen in the first and second case of the waveform, so it is confirmed that the component is working as intended. Although the plugboard maps a letter to another letter as a pair, it works quite differently from the Rotors. Because the configuration in the plugboard is static, similar to how a reflector is configured, the plugboard will give you the same answer for the same character given, because it's not dynamic like how the rotors can turn around and change its mapping based on the current position/orientation.

3.2.3 Keyboard

The schematic below shows the RTL model of the keyboard component generated from Quartus Prime.

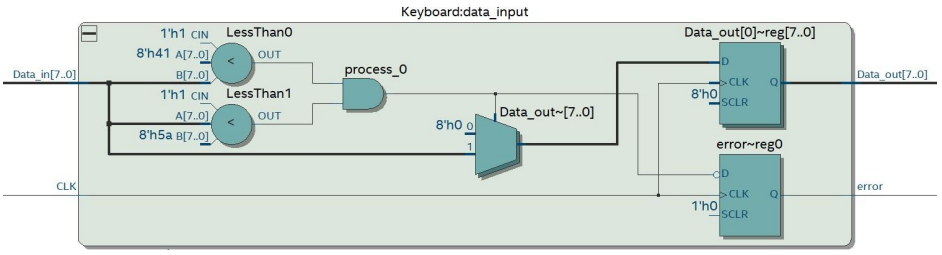


Fig 3.2.5 RTL schematic of the Keyboard

The keyboard component works as an input device from the user by receiving input and passing it in the form of a std_logic_vector (7 downto 0) signal to the next component which is the plugboard.

3.2.4 Top Level

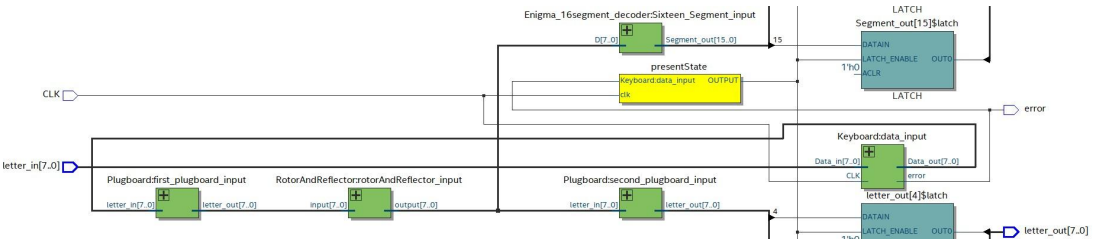


Fig 3.2.6 RTL schematic of the Top Level component

The picture above shows the Top Level code in the RTL model. The top level component is what binds all components together and provides the mapping for the inputs and outputs of every sub-components. In this case, an enigma machine has a certain data flow to follow. First, the input comes from the Keyboard component where it takes input from the user and forwards it to the plugboard which is one aspect of the encryption. After the signal goes through the plugboard and is already scrambled, it goes through another layer of encryption through the rotors where the signal will be scrambled three more times before being scrambled again by the reflector which will bounce the signal to the same three rotors from the backwards direction which will do three more steps of encryption. After going through the rotors/reflector section, the signal will travel through the plugboard once again before finally exiting the circuit as an output, usually shown with the help of lamps representing each letter of the alphabet or in this case a 16 segment display.

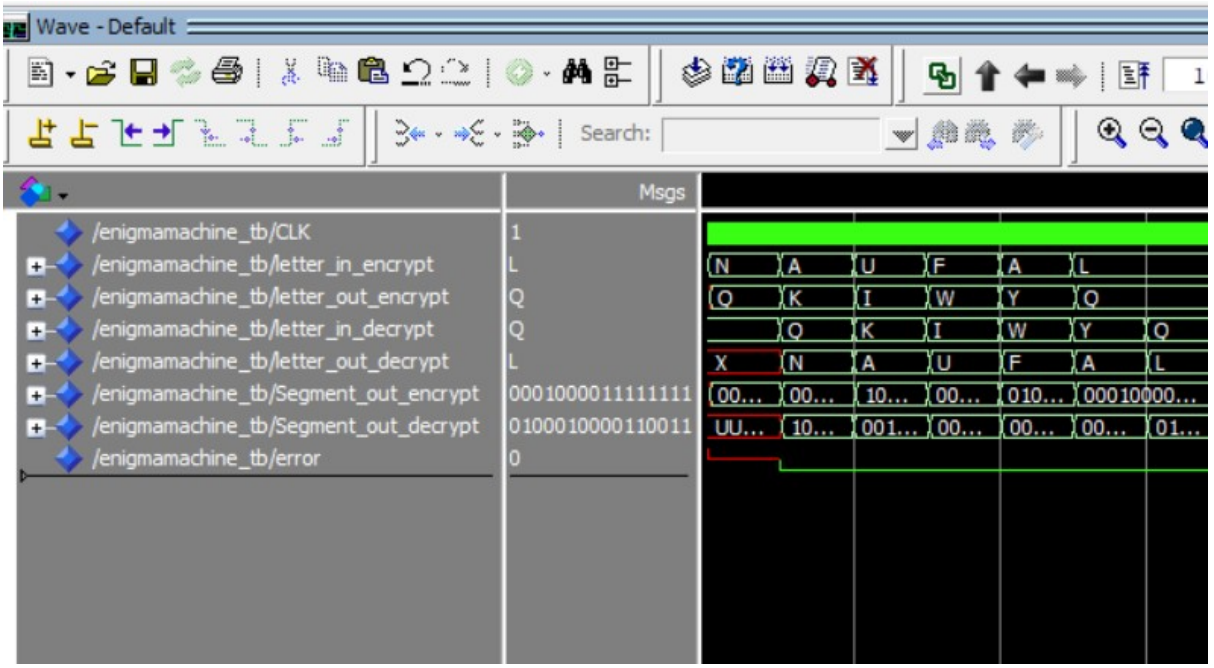


Fig 3.2.7 Waveform of the Top Level component

The picture above shows a waveform simulation of the top level component in Modelsim. It binds all the components together and maps the inputs and outputs of every sublevel component. In the testbench above, we tried encoding a string of letters and got the encoded output of “QKIWYQ”. After inputting the encrypted string in another instance of the enigma machine with the same state, it goes through the process of decryption and we get back the initial string of letters.

3.3 ANALYSIS

When the user enters an input it will pass through several components, the first one being the plugboard. The plugboard allows the user to connect several pairs of letters together, so that when one letter is entered, the signal would go through the plugboard component and be replaced with the pair letter that is configured in the configuration file, before passing through the rotors and the electrical circuit.

Afterwards, the signal would go through the rotor array to scramble the signal further. There are three rotors in this design with each rotor having their own scramble pattern. The reflector is used to reflect the signal that came from the rotor array with every letter reflected to a different letter, note that the reflector pattern has to be in pairs. The reflected signal will enter the rotor again from the opposite side and with opposite order, which means the

scramble pattern for each rotor needs to be reversed. What makes it very difficult to crack the enigma is everytime user gives an input the rotors change their rotation, making it have a different scramble pattern for each letter given.

Finally, the signal goes through the plugboard once more and can be read directly or passed to an encoder which can be displayed on a 16-segment display.

CHAPTER 4

CONCLUSION

- VHDL can be used to reimplement complex electrical circuit mechanisms.
- Most algorithm can be implemented in VHDL, except the one that rely on intensive memory usage.
- Encryption on an enigma machine is done by the plugboard and reflector
- Our implementation of the Enigma machine in VHDL was successful, and demonstrated the feasibility of using VHDL for digital system design projects.
- What makes enigma so hard to crack is its capability to change its scramble pattern for every input it receives.

REFERENCES

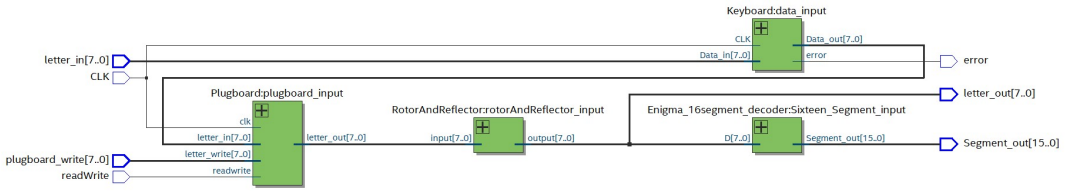
[1] RCode breaking during WWII. Enigma Machine by 101Computing.net. (n.d.). Retrieved December 9, 2022, from https://www.101computing.net/enigma/

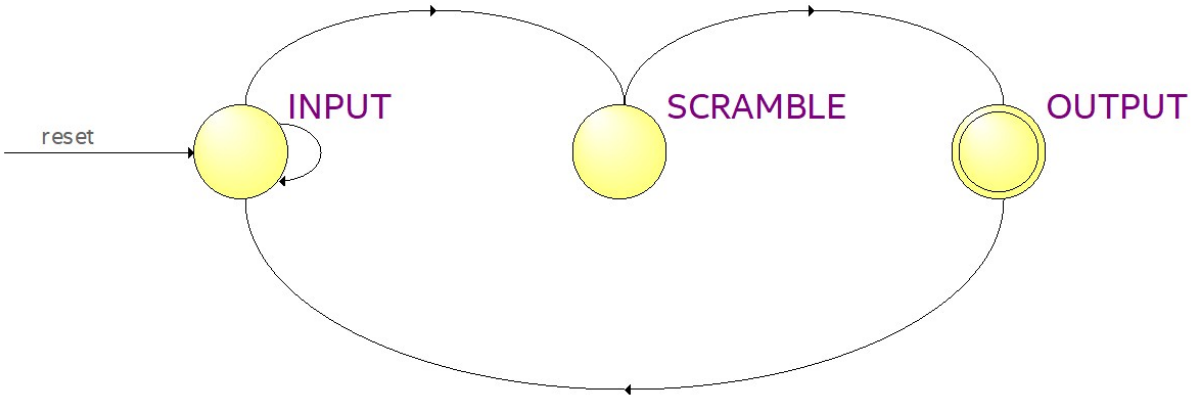
[2] G. Welchman, “From Polish Bomba to British Bombe: The birth of ultra,” in *Codebreaking and signals intelligence*, 1St Ed., S.l.: ROUTLEDGE, 1986, pp. 71–110.

[3] M. Rajewski, “How Polish Mathematicians Deciphered the Enigma,” IEEE Annals of the History of Computing, vol. 3, no. 3, pp. 213–234, July 1981.

APPENDICES

Appendix A: Project Schematic





Appendix B: Documentation