

**ATM**

Authors: Făt Vlad-Alexandru

Spătariu Raul-Andrei

Supervisor: ing. Mihai Timar

Date: 31.05.2023

Table of Contents

[**1 – Specifications** 3](#_gjdgxs)

[1.1– Black Box 3](#_30j0zll)

[1.2 – The functioning mode 5](#_1fob9te)

[2 – Design 6](#_3znysh7)

[2.1 – Execution](#_2et92p0) and Control Unit [6](#_2et92p0)

[2.2 –](#_tyjcwt) Seven Segment Display [7](#_tyjcwt)

[2.3 – Button](#_3dy6vkm) Controller [8](#_3dy6vkm)

[2.4 – Internal](#_1t3h5sf) Memory [8](#_1t3h5sf)

[2.5 – Dispenser](#_4d34og8) Money [10](#_4d34og8)

[2.6 – State](#_2s8eyo1) Diagram of the Control Unit [11](#_2s8eyo1)

[3. User Manual 13](#_lnxbz9)

[4 – Future Developments 14](#_35nkun2)

[5 – References 14](#_1ksv4uv)

## **1 – Specifications**

Design an ATM for withdrawals in EURO. It needs to function properly, each account with a pin and a minimum of four operations are needed. In our case we have cash withdrawal, cash deposit, balance query, change of pin and exit atm.

### 1.1– Black Box

A picture containing text, screenshot, diagram, font

Description automatically generated

The ATM implemented by us is similar to a normal function ATM from real-life. It has 4 available accounts that can be modified inside the code by the owner, the same thing applies for the PINs that correspond to each account. The ATM has 5 operations: balance query, withdrawal, deposit, change pin and exit atm.

The functionality is straight forward, we have 5 buttons that control the system and one switch that resets everything to the initial state, idle. The 4 buttons up, down, left and right control the four 7SD through witch the user either manages the input or sees the output, depending on the case.

To be able to access the operations the user needs to introduce the card number from 1 to 4, and the corresponding pin, any wrong input will lead to the ATM not going forward. For a better understanding this is the flow of instructions that the user needs to do in order to use the ATM at his full capacity:

1.Start the ATM

2.Introduce the card number

3.Introduce the right PIN

4.Select an operation from 1 to 4

5.According to the operation selected the user either needs to input or observe the output

Any wrong input will make the ATM light up an error led for one second and then go back to the last state, if the current operation is successful a success led will turn on, also when the ATM is waiting for the user input an waiting led will be turned on.

The ATM structure will become more clear when we go through the state diagram and show the architecture behind every component and the architecture of the whole project that is the EU and CU connected between them and different side components that each do certain specified instructions

The signals above represent the major inputs and outputs of the circuit. The 5 buttons are the ones that the user uses to navigate through the functionality of the ATM, the output “MESSAGE” and the three LEDs are the Graphical User Interface.

A more detailed explanation of the project’s functionality and how it maps to the Basys3 can be found in the following section and section 3.

### 1.2 – The functioning mode

The ATM module is a finite automata which functions according to the technical specifications asked for.

The module is built in such a way that the CU changes states at every clock cycle if possible, provides the correct outputs so the EU will know how to act accordingly to them, and based on the input got from the EU, the CU changes states or remains in the current state.

Below you will find the detailed schematic, further down we will break it into components to get a better understanding of the functionality of the ATM.

A picture containing diagram, plan, sketch, technical drawing

Description automatically generated

The main architecture

## 2 – Design

### 2.1 – Execution Unit and the Control Unit

The system's black box must be further broken down in order to find implementable components. We will do a **top-down** breakdown of the problem until we get to known circuits, and then we will implement **bottom-up**.

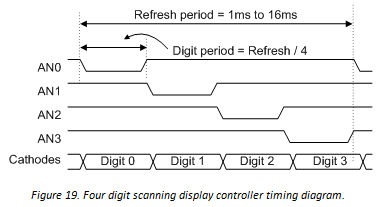
The first breakdown of any system is one in which we will differentiate between the **control logic** in the system and the **system resources**. The control logic is represented by the Control Unit (CU) and the resources are represented by the Execution Unit (EU).

A picture containing text, diagram, plan, rectangle

Description automatically generated

### 2.2 – SEVEN SEGMENT DISPLAY

This is an important part of our project because it represents the interface through which the user can navigate through. For the implementation we followed the nexys4 reference manual to change through all the anodes in 1 millisecond.



Below is the logical diagram of the component :

A screenshot of a computer

Description automatically generated with medium confidence

### 2.3 – Button Controller

With this component the user can control what inputs he gives, the functionality is pretty simple with left or right we move through the seven segment display and with up or down we increment/decrement the value on the selected 7SD.

A picture containing text, diagram, line, parallel

Description automatically generated

### 2.4 – Internal Memory

To be able to keep track of the constantly modifying values of the cash from ATM, cash available in the accounts or the cash in each account we designed several RAM modules that we modify accordingly to what the user inputs.

\*RAM for the banknotes available in the ATM

A screenshot of a computer program

Description automatically generated with low confidence

\*RAM for the PIN’S of each account

A picture containing text, screenshot, font, number

Description automatically generated

\*RAM for the money available in each account

A screenshot of a computer

Description automatically generated with low confidence

### 2.5 – Dispenser Money

For the money dispenser, we have an output to tell how many banknotes of 100, 50, 10 or 5 to be given, if the withdrawn amount is less or equal than 1000€ and if it is a multiple of 5 we make use of the 100mHz clock to make the greedy algorithm for the distribution of banknotes. We have a counter which counts 1000 clock cycles, and when it reaches 1000, the output would be that the dispenser is done, and we can then go to a next state, either to withdraw the money or to go intro error, depending on the generated ‘ok’ signal, which indicates if the conditions were met.

A picture containing text, diagram, plan, technical drawing

Description automatically generated

### 2.6 – State diagram of the Control Unit

The state diagram is **not a flow chart**, but represents the control part, the decision part of any algorithm, and it can then be implemented directly in VHDL if done correctly.

**States** are represented by . A state represents a moment of time (a period). The **decisions** made in each state are represented by 

The **outputs** generated in each state are represented by . Inside the rectangle are the outputs that are true at that time.

A picture containing diagram, text, technical drawing, plan

Description automatically generated

A picture containing text, diagram, line, parallel

Description automatically generated

### A picture containing text, diagram, number, plan Description automatically generated

The internal schematic of the stack

## 3. User Manual

The ATM can have up to 4 accounts and the content of each account can be changed by the user in the RAM code. Below you can find the content of RAM that contains the money each account has, and if the user wants to add or change the amount he has to modify the contents.

A close-up of a computer code

Description automatically generated with low confidence

A picture containing text, screenshot, font

Description automatically generated

After this everything is set up, the circuit can be loaded onto the FPGA. The following instructions relate to the Basys3, since this is the board on which our final version was presented:

* In order to manually follow the program, the CLK signal should be mapped to one the internal CLK inside the constraint file.
* The 5 buttons are the inputs.
* The last switch from the left is the RESET that resets the ATM putting it in the first state
* The first three LEDs from the right are the waiting led(L1), success led(P1) and error led(N3)
* Both the INPUT and OUTPUT commands activate the SSD. The value processed by these instructions will be displayed on the SSD

## 4 – Future Developments

For the purposes of our project, the ATM was configured to perform actions only from the users perspective but not from the association it is implemented by, therefore a future development will be as the owner to be able to update banknotes in the ATM, in another words to fill it up when it empties.

Another thing worth mentioning will be that the current implementation of the inputs in not as flexible for the user as a keyboard, so that is the next step. After all, with easier data introduction will make the ATM faster.

## 5 – References

* The official Xilinx support and Q&A official forum https://support.xilinx.com