TECHNICAL UNIVERSITY OF CLUJ-NAPOCA, ROMANIA



PROJECT A9:

*ADVERTISEMENT WITH MULTIPLE ANIMATIONS*

DIGITAL SYSTEMS DESIGN PROJECT, 2021

*Author:* Alexandra Ilovan

*Group:* 30412

*Year:* I

*Faculty:* Faculty of Automation and Computer Science – TUCN

*Specialization:* Computers (in English)

*Discipline:* Digital Systems Design (DSD)

*Coordinating teacher:* Noema-Laura Maier

TABLE OF CONTENTS

1. Project Specification
2. Theoretical Considerations
3. Schematics
4. Implementation Details
5. Experimental Results
6. Ulterior Developments
7. Bibliography
8. PROJECT SPECIFICATION

**Project A9:** Design an **advertisement with multiple animations**. 7 segment displays will be used in the making of the project. The displayed text will be formed out of symbols of an available alphabet. The advertisement will have multiple functioning regimes (minimum 4), which will be selected by the user from the commutators of the FPGA board. The usage of the quartz oscillator incorporated in the FPGA board is required (the respective clock signal should be divided, obviously). Functioning regime examples: “flowing” of the text from right to left, flickering, letter by letter display etc.

Since not all letters can be represented on a 7 segment display, a maximal alphabet will be created and the messages will be composed of the symbols of said alphabet. The message will be contained in a memory, so that changing it can be done with more ease. The project will be realized by **1 student**.

The implemented animations:

A0: Static

A1: Shifting of the letters from RIGHT to LEFT

A2: Shifting of the letters from LEFT to RIGHT

A3: Flickering of the entire text

A4: Display of the text letter by letter, all the letters being shown on one single anode

A5: ”Flowing” of the text from left to right

A6: Intermittent blinking effect of 2 letter-pairs: the first and last 2 letters

A7: Intermittent blinking effect of 2 letter-pairs: the middle and exterior 2 letters

1. THEORETICAL CONSIDERATIONS

*DEVELOPMENT ENVIRONMENT*

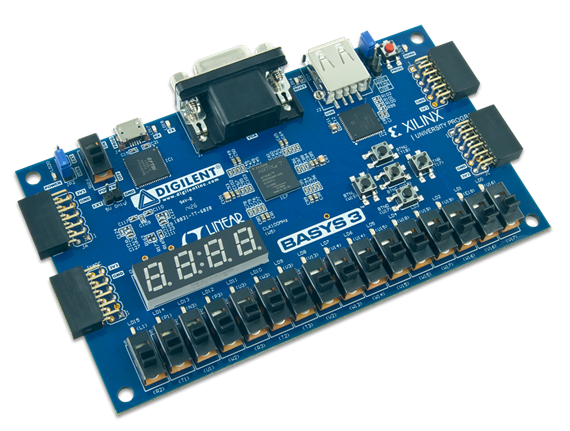
The implementation of the project was done using **Active-HDL**, a software owned by **ALDEC, Inc.** Active-HDL is a FPGA development environment built around common kernel HDL simulator. It supports text-based and graphical design entry and debugging tools, allows mixed-language simulation (VHDL/Verilog/EDIF/SystemC/SystemVerilog) and provides unified interface to various synthesis and implementation tools. It is only available on MS Windows platform. 1 (See Bibliography)

*HARDWARE LANGUAGE*

The project was written entirely using the **VHSIC Hardware Description Language (VHDL)**. VHSIC stands for “Very High Speed Integrated Circuit”. VHDL is generally used to write models that describe the behavior and structure of digital systems at multiple levels of abstraction (from the top system level down to logic gates), for design entry, documentation and verification purposes. Among its key advantages is the fact that it allows the description of a concurrent system, which can be modelled and simulated before translating the design into real hardware (gates and wires) in the synthesis step. 2 (See Bibliography)

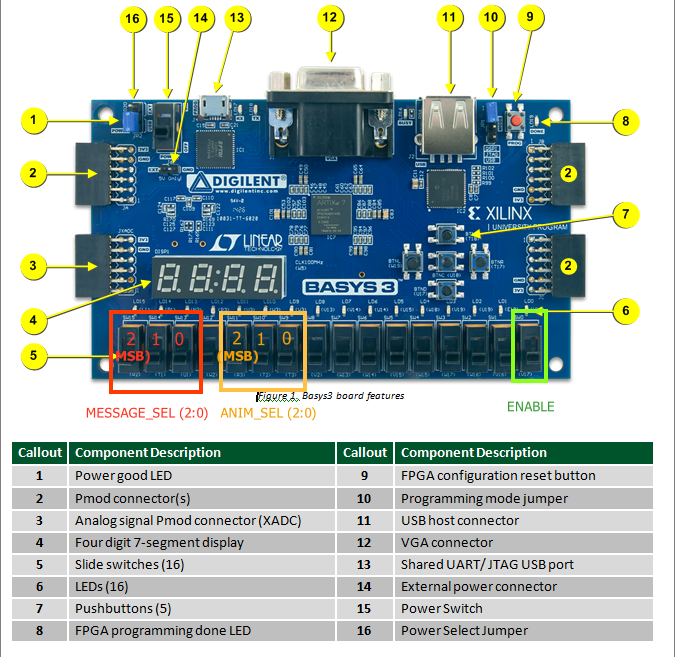
*BASYS3 FPGA BOARD: WALKTHROUGH*

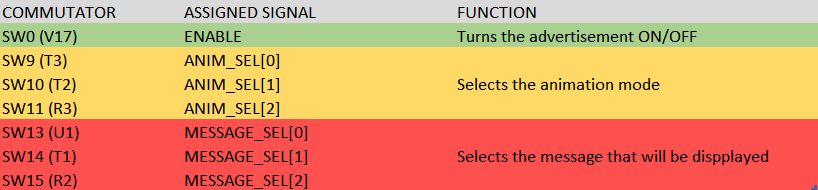
The design was implemented on a **Basys3 FPGA board**. The Basys3 board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7TM Field Programmable Gate Array (FPGA) from Xilinx. Its Xilinx part numbers is: XC7A35T-1CPG236C. 3 (See Bibliography)



Basys3 offers a wide collection of ports and peripherals, including:

* 16 user switches
* 16 user LEDs
* 5 user pushbuttons
* 4-digit 7-segment display
* 3 Pmod ports
* Pmod for XADC signals
* 12-bit VGA output
* USB-UART Bridge
* Serial Flash
* Digilent USB-JTAG port for FPGA programming and communication
* USB HID Host for mice, keyboards and memory sticks

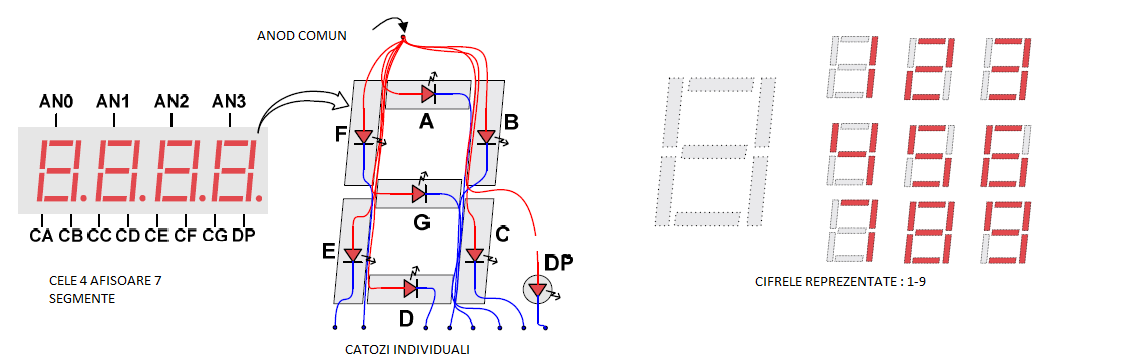
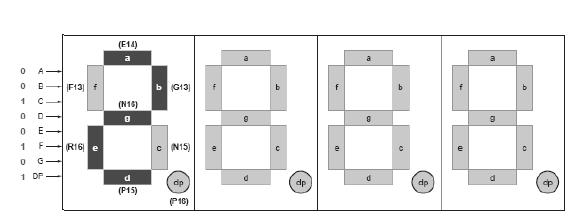
USED PINS, THE SIGNALS ASSIGNED TO THEM AND THEIR SEMNIFICATION



**About the incorporated 7-segment displays**

The Basys3 board contains a four-digit common anode 7-segment LED display. Each of the four digits is composed of seven segments arranged in a “figure 8” pattern, with an LED embedded in each segment.

It is especially important to get accustomed to the functionality of a 7-segment displays since the implementation of this system relies mainly on them.

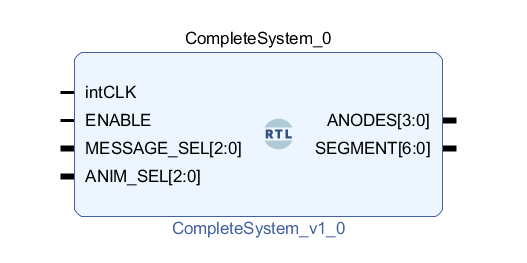


As it can be seen, a display has 7 leds, representing the segments and a symbol dp, representing the decimal point.

Each anode is ACTIVE-LOW.

1. SCHEMATICS

***BLACK BOX***



Inputs:

* **intCLK** – the internal clock signal of the board
* will be used for the temporization of the animations and their display functions
* it cannot be operated or modified by the user
* **ENABLE** – enable signal that turns the entire display of the advertisement **ON/OFF**

**ENABLE** = ‘0’ => the displays are turned **OFF**

**ENABLE** = ‘1’ => the displays are turned **ON**

* **MESSAGE\_SEL[2:0]** – signal on 3 bits used to select which message will be displayed (this offers the possibility of having a total of 8 different messages, however, only 5 messages are readily available in this case)

! If none of the hardcoded messages is chosen, the system will display the “\_ \_ \_ \_” string instead.

* **ANIM\_SEL[2:0]** – signal on 3 bits used to select a functioning regime/animation type of the current message that is being displayed

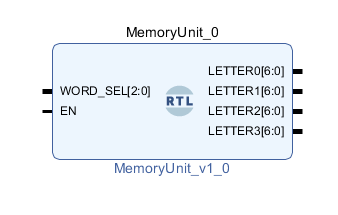
Outputs:

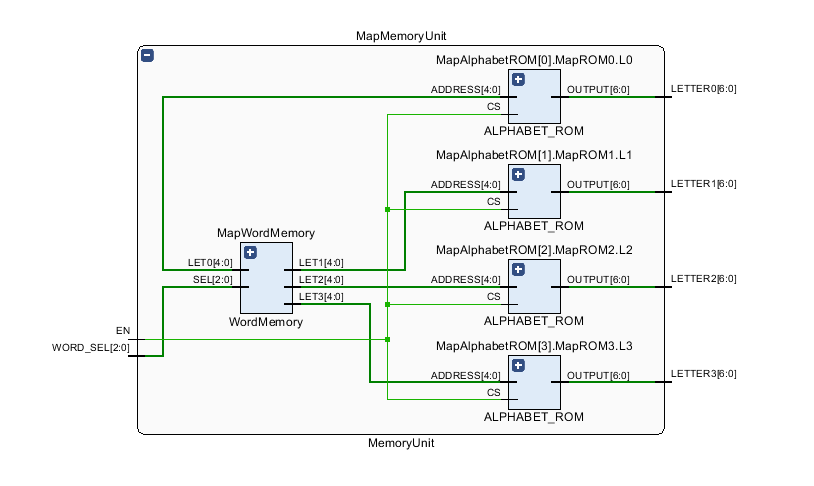
* **ANODES[3: 0]** – output signal on 4 bits
* connected to the anode ports of the FPGA board
* used for multiplexing each individual letter display
* **SEGMENT[6:0]** – output signal on 7 bits; connected to the common cathodes of the eight 7-segment displays

The Complete System Unit features two other big, main units:

1. The Memory Unit
2. The Shift Memory Unit
3. **The Memory Unit**

The **Memory Unit** stores the maximal alphabet and the words that will be displayed for the advertisement. It contains the following components:

* **WordMemory** – a memory which stores the words that need to be displayed
* **4 Read-Only Memories (ROMs)** with the identifier **ALPHABET\_ROM** - one for ****each letter of the selected word

****

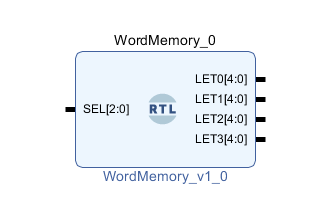
Inputs:

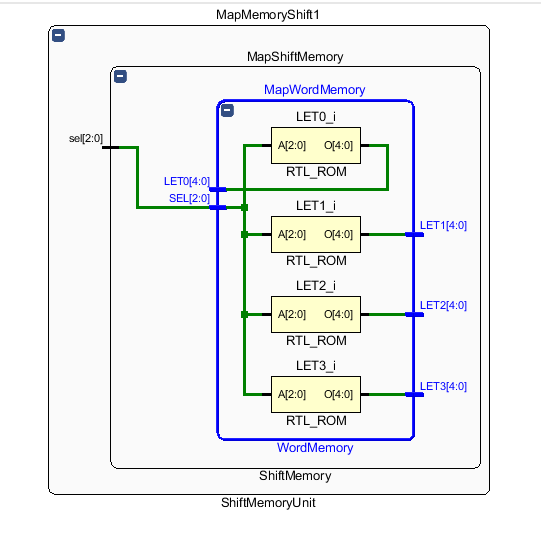
* **WORD\_SEL[2:0]** – will be connected to the **MESSAGE\_SEL[2:0]** signal
* **EN** – will be connected to the system’s **ENABLE** signal, and to the **Chip Select** of each ROM

Outputs:

* **LETTER0, LETTER1, LETTER2, LETTER3 ([6:0]) –** 7 bit signals, corresponding to the 7 segments each letter is represented on

*COMPONENTS – MORE IN DEPTH:*

* 1. **WordMemory –** selects the word that will be displayed and its outputs LET0, LET1, LET2, LET3 ([4:0]) represent the address of each letter in the **ALPHABET\_ROM**

****

* 1. **ALPHABET\_ROM**



Inputs:

* **ADDRESS[4:0]** – address of a certain letter in the memory; it is a signal on 5 bits since this system’s maximal alphabet consists of 29 characters, and the number 29 is represented on 5 bits in binary
* **CS** – ROM Chip Select; if it has the value ‘1’, the ROM chip is active
  + Connected to the EN (enable) signal

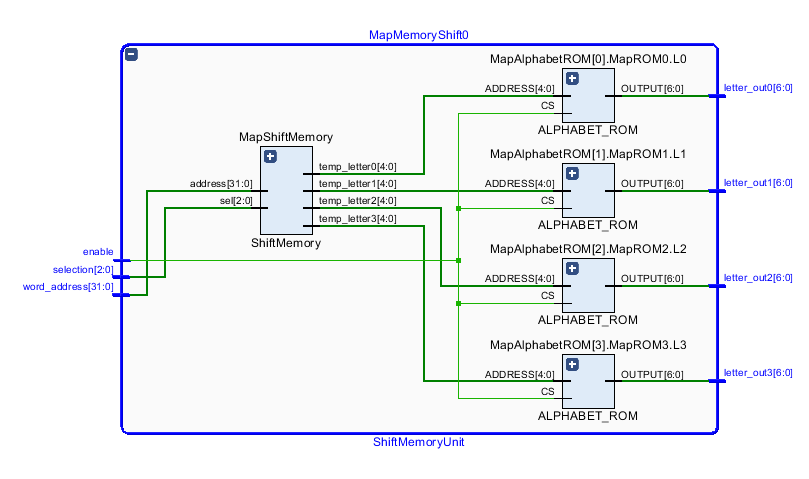
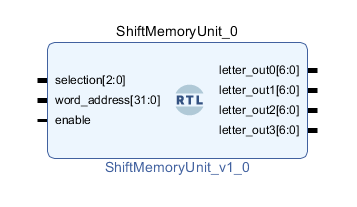
Outputs:

* **OUTPUT[6:0] –** 7-bit signal, holds the representation of the selected letter on a 7-segment display

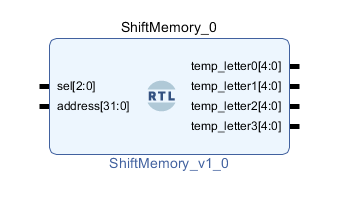
1. **The Shift Memory Unit**

The Shift Memory Unit stores all the phases a word goes through after one of its letters shifts to the left once. Its inputs and outputs are similar to the ones of the Memory Unit and they present the same functionalities, but there is one additional input which is the key of the unit: the **word\_address[31:0]** signal of INTEGER type. Like the name suggests, this signal represents the address of the start of a new “phase” of the word. In order to make it more clear, let’s take the following example:

If we shift a letter in the word “OREO” to the left we get “REOO”, and then “EOOR” and so on. Therefore, when word\_address = 0, it points to the word “OREO”, while when word\_address = 4, it points to “REOO”.

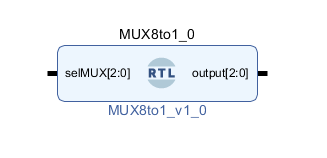


The Shift Memory Unit is using a ShiftMemory component (which, in turn, contains a WordMemory block) and 4 ALPHABET\_ROMs, one for each letter of the word.

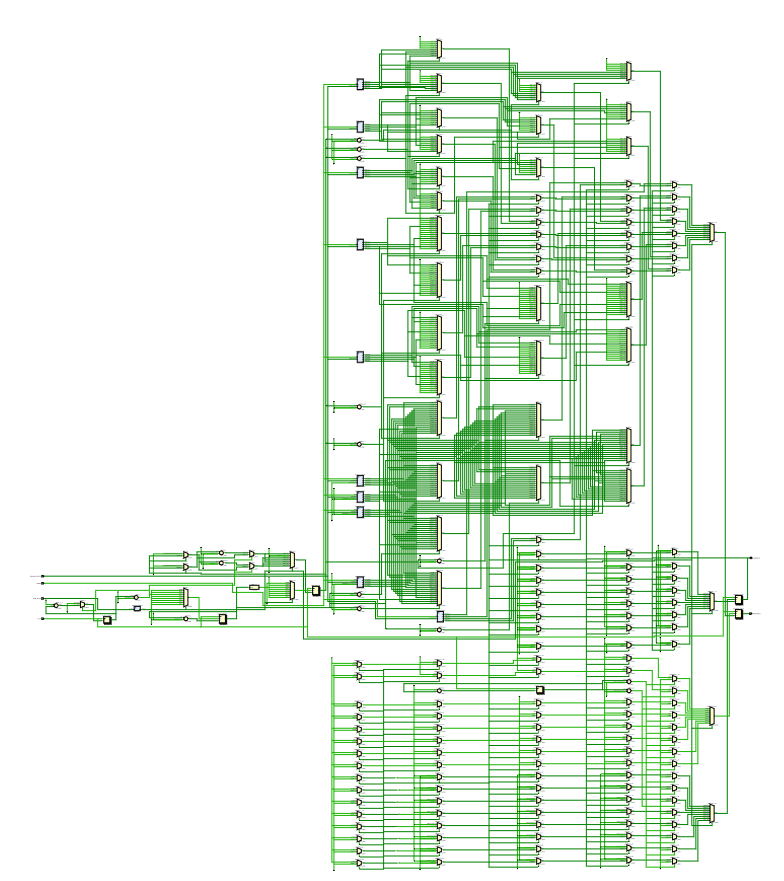


**Other components:**

**MUX8to1 –** connected to ANIM\_SEL, used for selecting the proper animation mode corresponding to the given signal

****

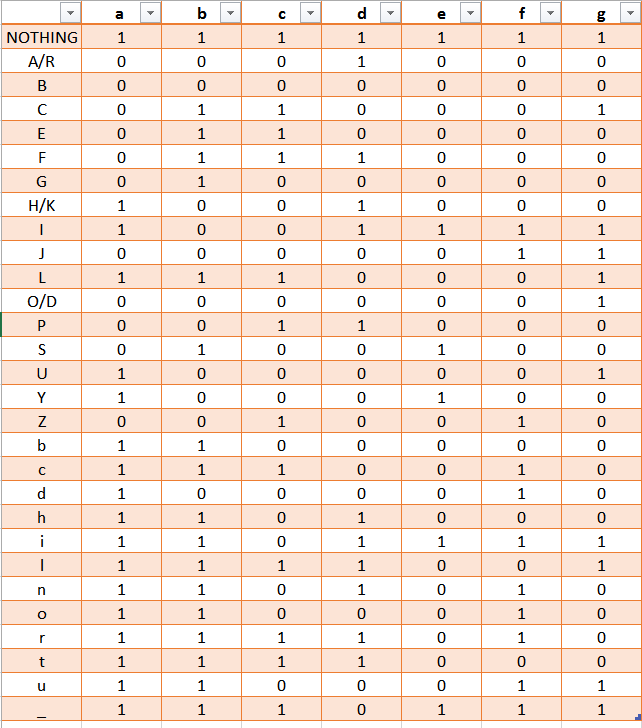
The detailed schematic for the Complete System Unit, generated by Vivado:

****

1. IMPLEMENTATION DETAILS

The first step in designing the system was to create a maximal alphabet that can be represented on 7-segment displays. An important aspect to take into consideration when doing so is the fact that 7-segment displays are ACTIVE-LOW.

**THE HARDCODED MAXIMAL ALPHABET:**

****

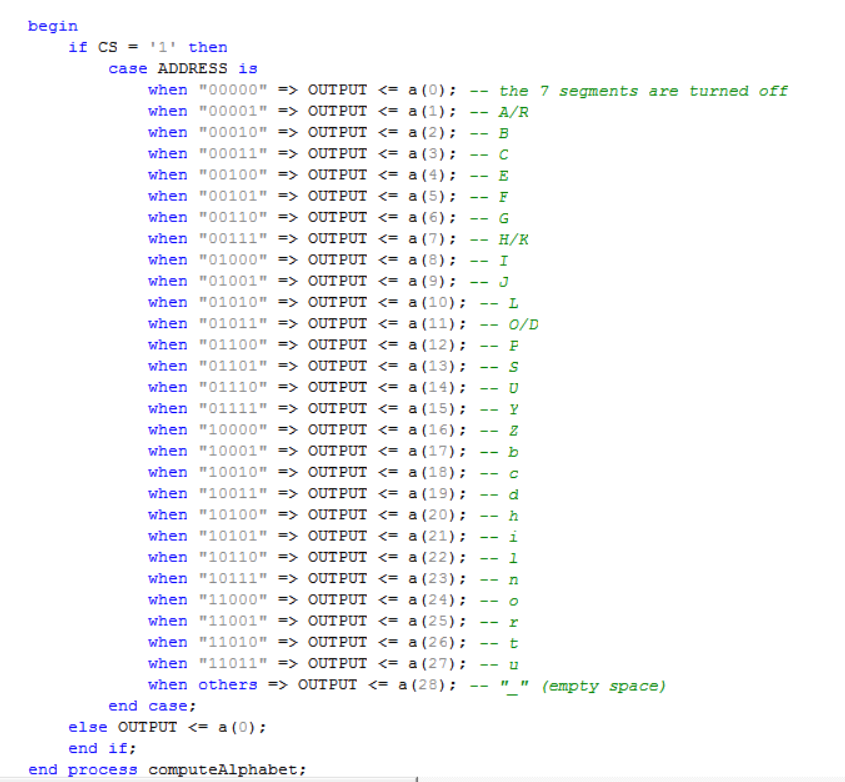
The code for the Read-Only Memory (ROM) needed to contain the letters of said alphabet:

The memory block contains an address signal on 5 bits, a Chip Select (or ROM Enable) and the Output (on 7 bits) which stores data in the same format as displayed above in the table.

**The computation of the alphabet:**

****

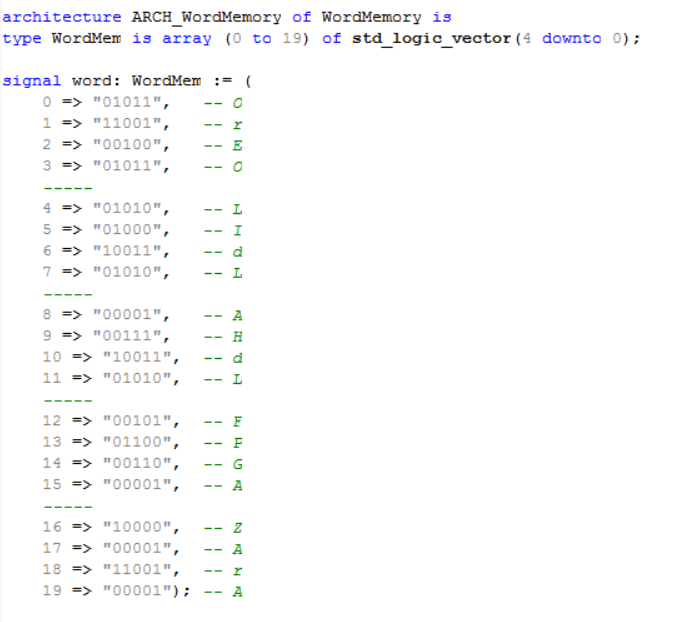
**Assigning the stored letters to their respective addresses:**

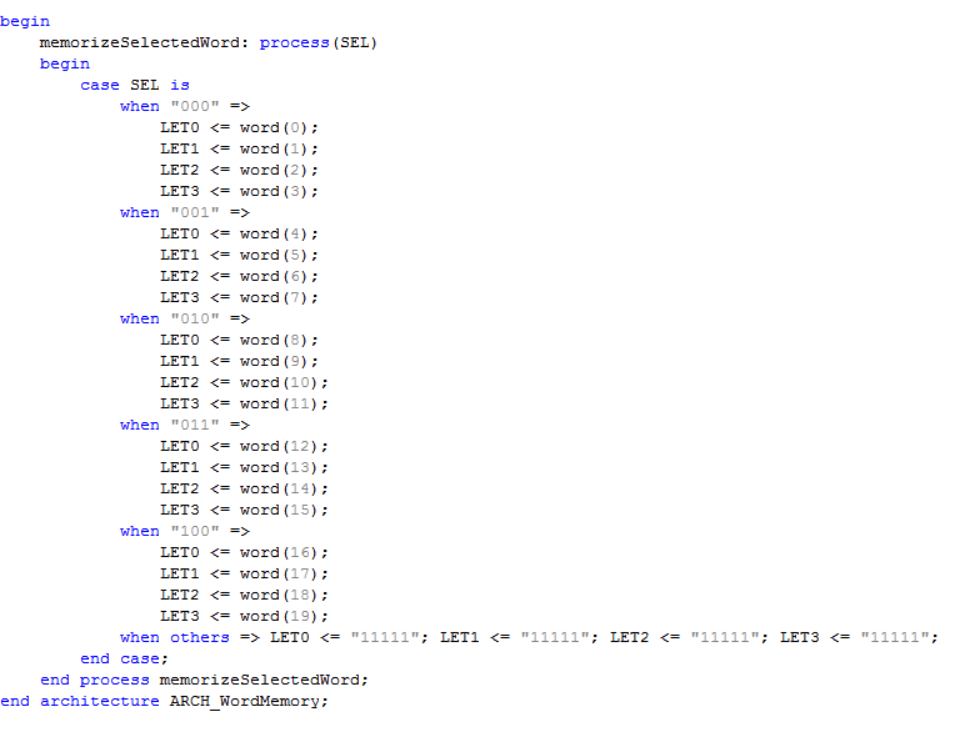
****

Now that the Alphabet Memory was implemented, a memory to store the words that have to be displayed is also needed. The linking between the Word Memory and the Alphabet Memory is done by means of the address signal. The address signal is encoded on 5 bits due to the fact that the alphabet contains a total of 29 characters.

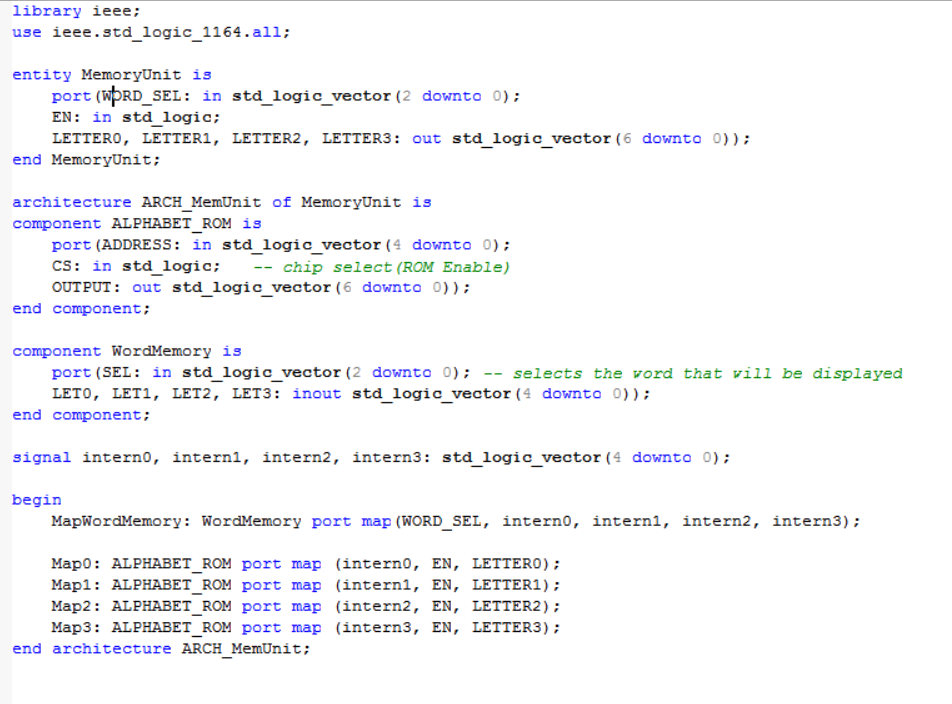
Within the Word Memory architecture, a new type, WordMem, which is an array of 20 elements of 5 bits each, was created in order to store in each element one letter of a word. Since 5 messages were stored in total, 20 arrays of 4 bits were enough to encode them. Then a process of selecting which message would be stored was created.

**Word Memory code:**

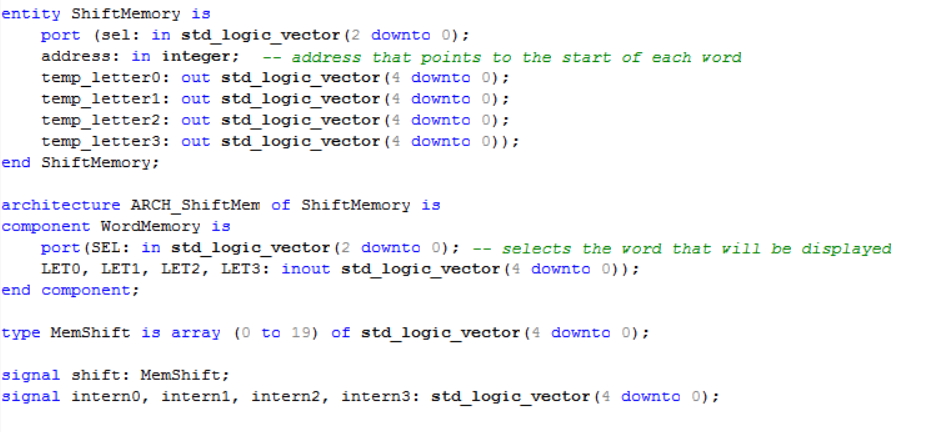
****

****

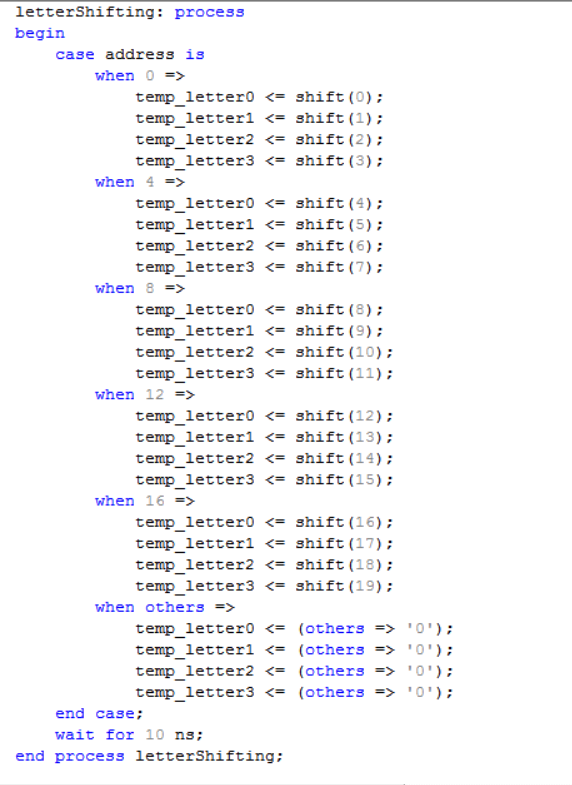
Ultimately, these 2 memory blocks were put together, thus completing the Memory Unit of the system.

****

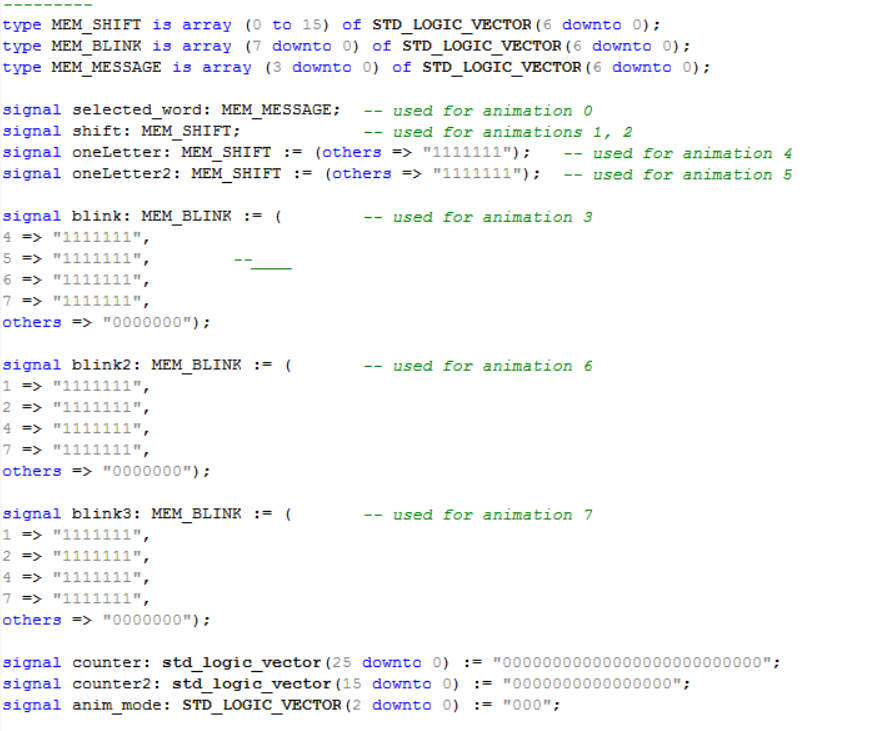
One of the key points of this system is the functionality of the animations. One of these animations consists of reproducing a shifting effect, hence why a Shift Memory along with a Shift Memory Unit were implemented. In order to create this, another data type was created – MemShift, denoting a sequence of bits that retains the positions of each letter of a word while it is being shifted.



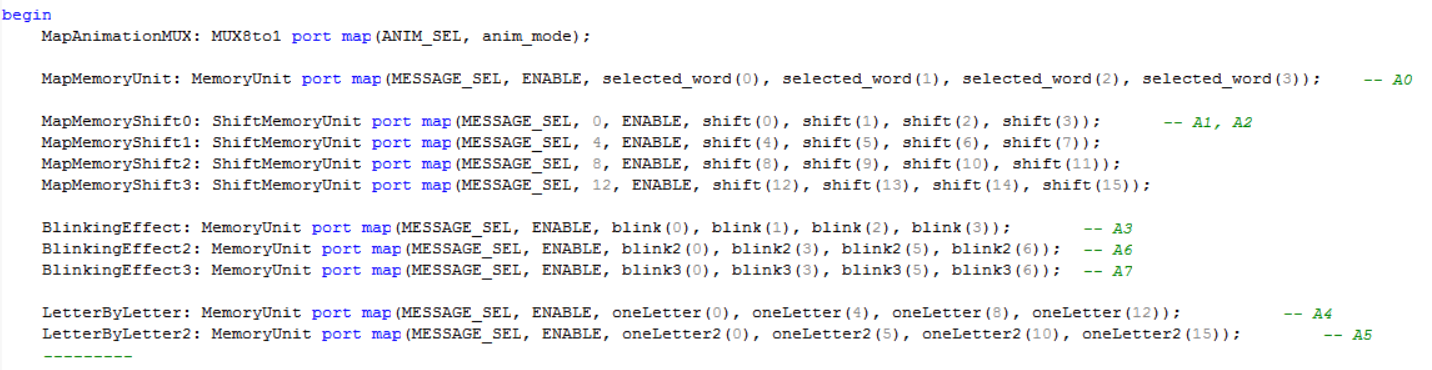




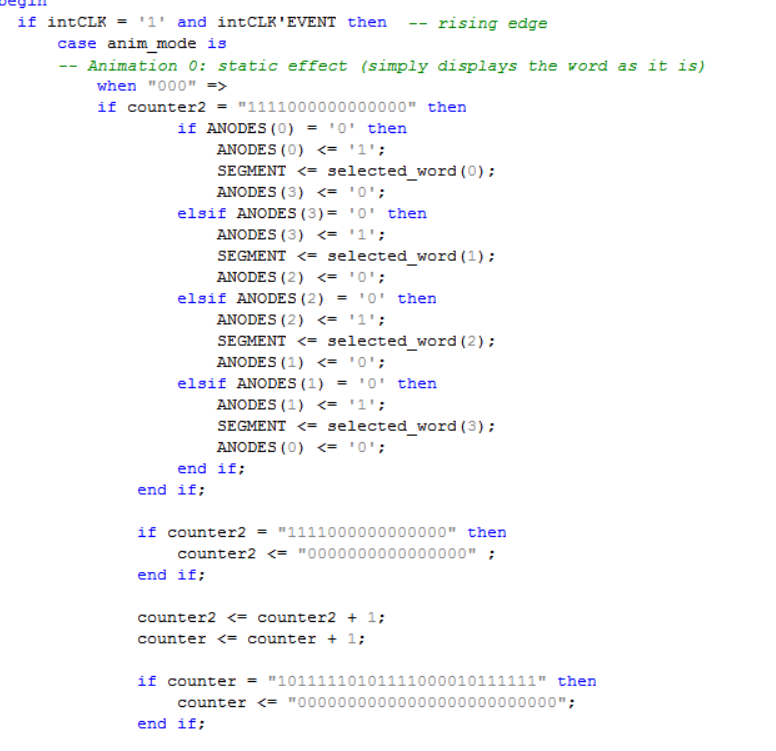
The animations were realized by using the Memory and Memory Shift Units, along with a few signals of the newly defined types MEM\_SHIFT, MEM\_MESSAGE, MEM\_BLINK.

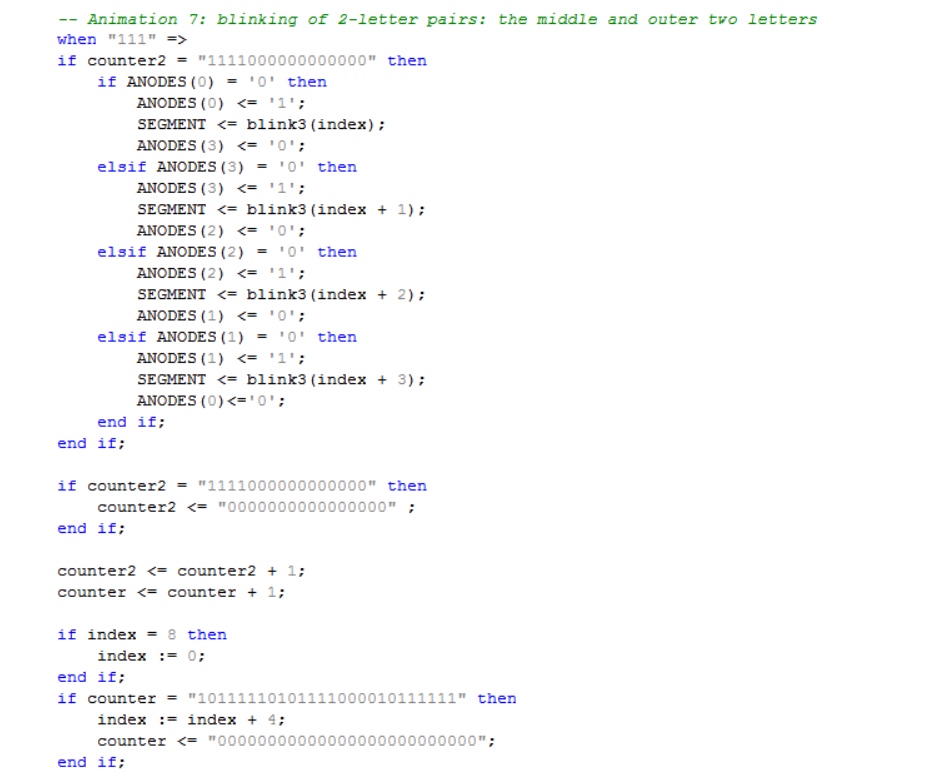
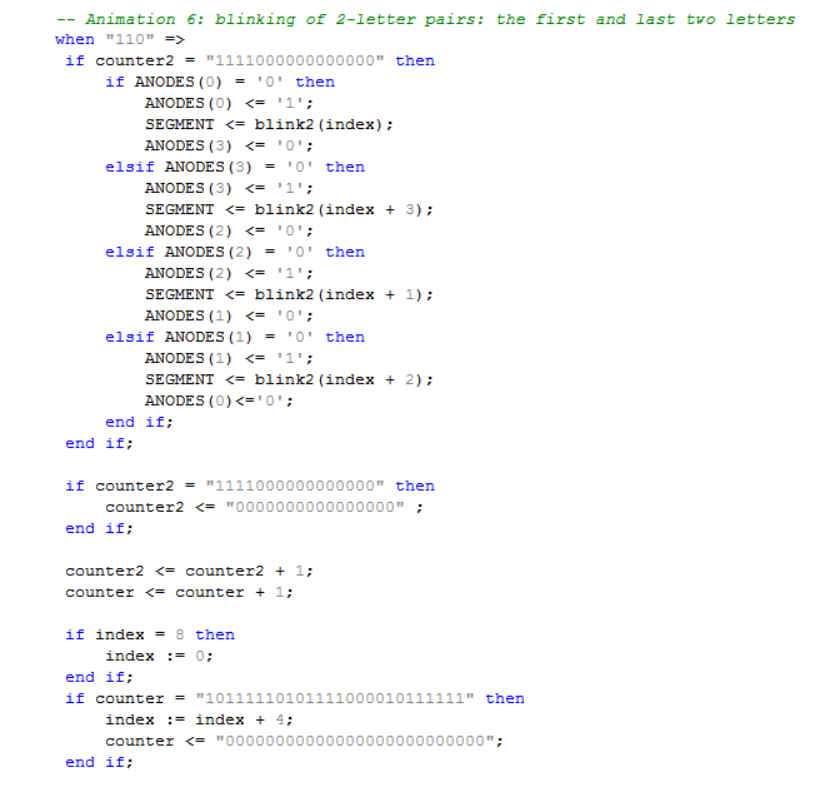
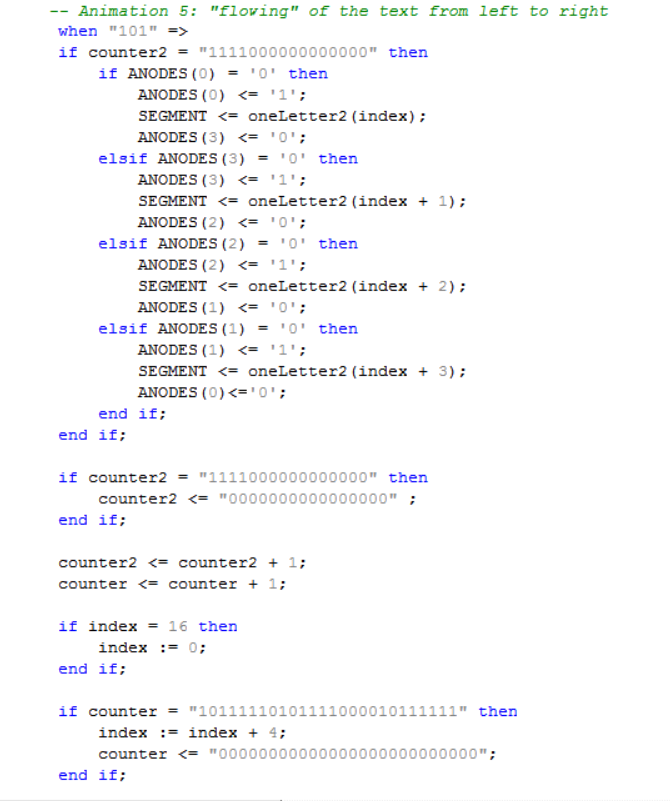
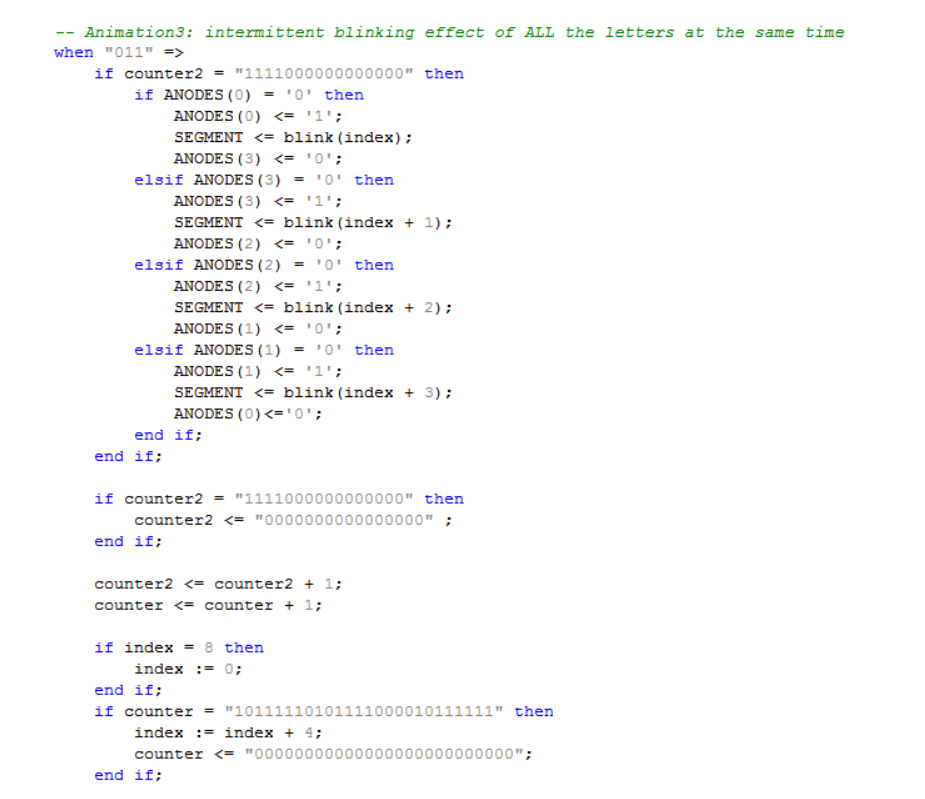
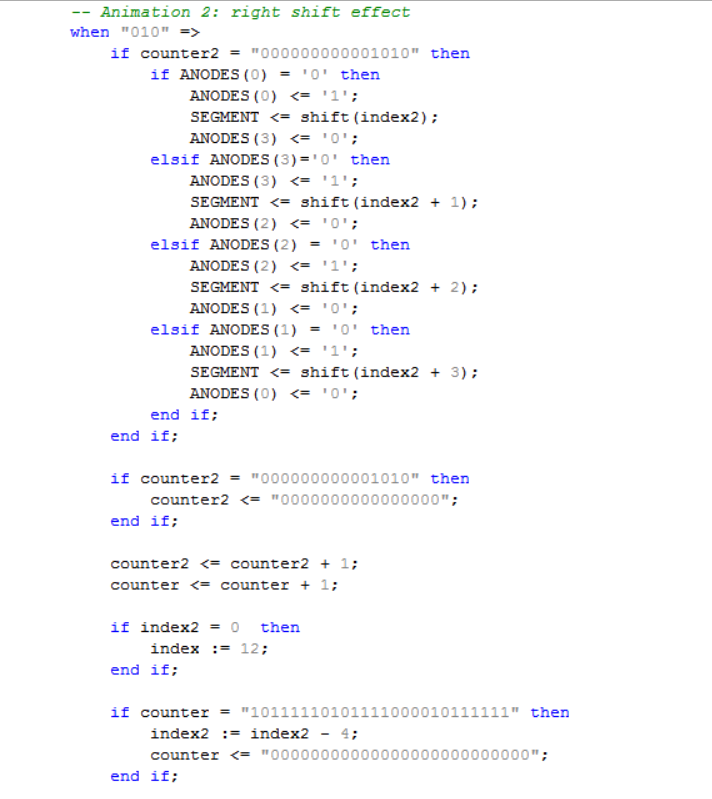
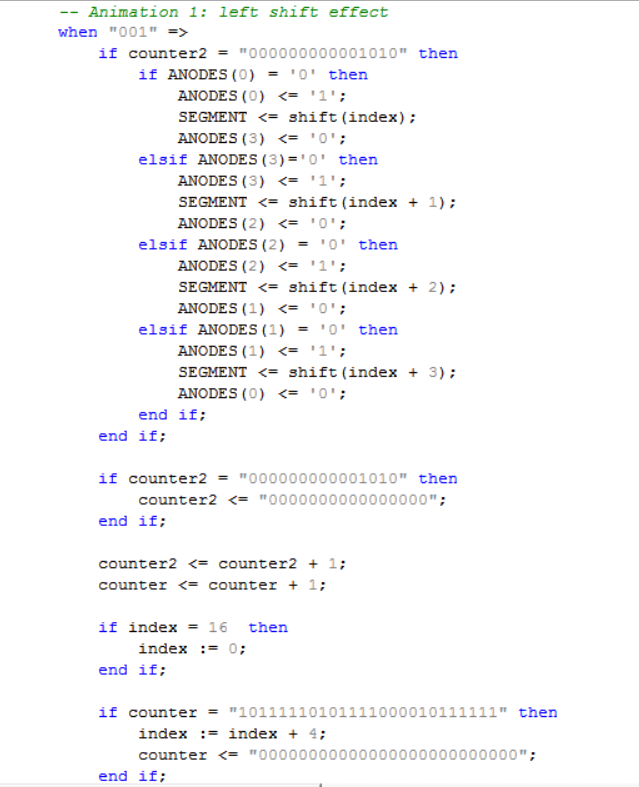


The port maps:



**Animations:**

****

****

When it comes to the animations, the most relevant element, without which the user would not be able to see the advertisement is the clock of the board, which functions on 50 MHz, which is why we need a way of slowing down the execution time. A counter on 26 bits was the solution. By using such a thing that counts each second or slows down the execution by one second, the human eye can notice freely the changes that occur in the animation.

The clock frequency is being divided by using counter and counter2. Generally, after one second passes, we upload another word onto the board.

V. EXPERIMENTAL RESULTS & USAGE INSTRUCTIONS

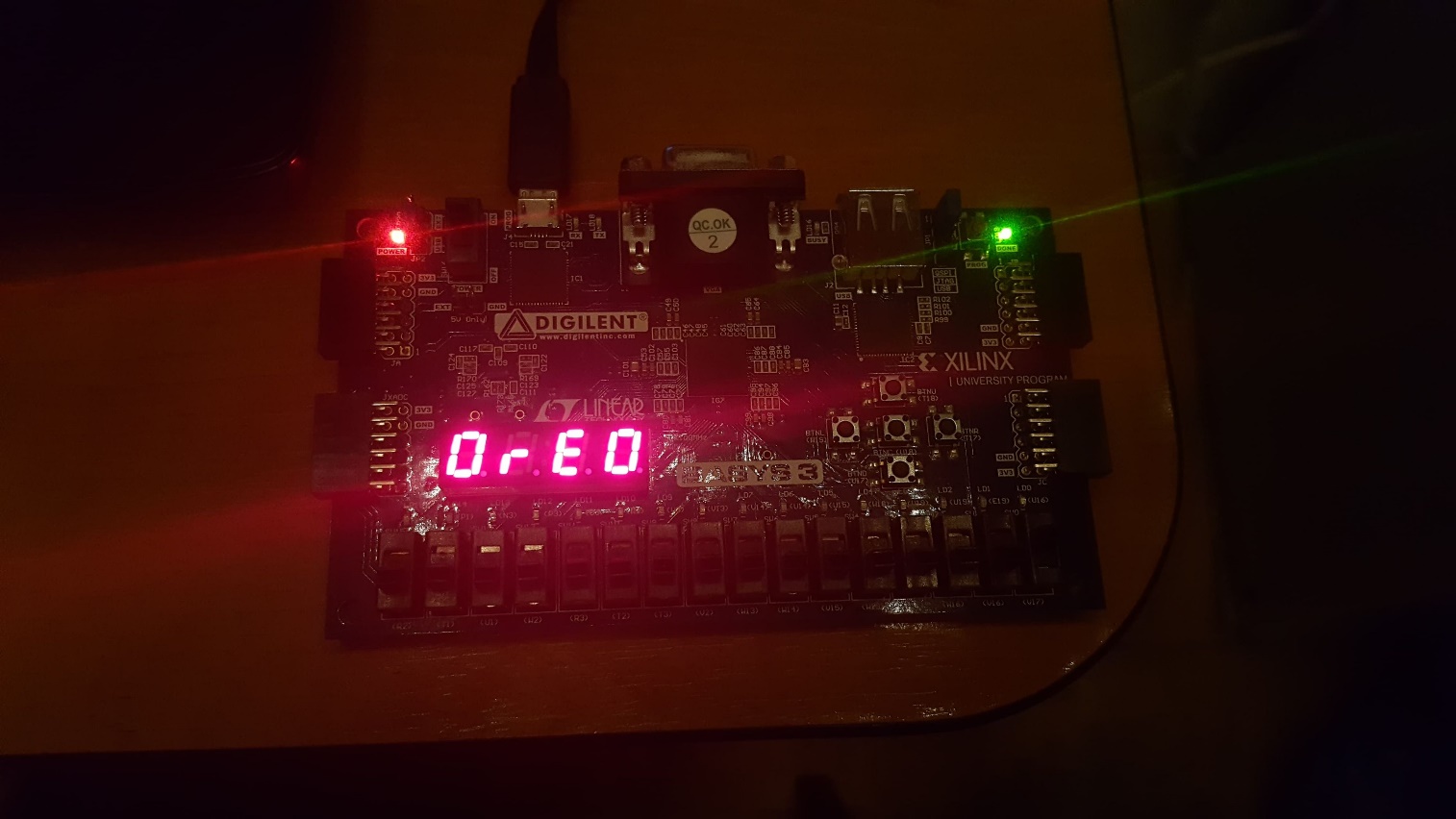
In order to implement the design on the Basys3 board, the usage of an additional software (**Vivado**) was needed.

SHORT WALKTHROUGH THROUGH THE PROCESS ONE HAS TO FOLLOW IN VIVADO:

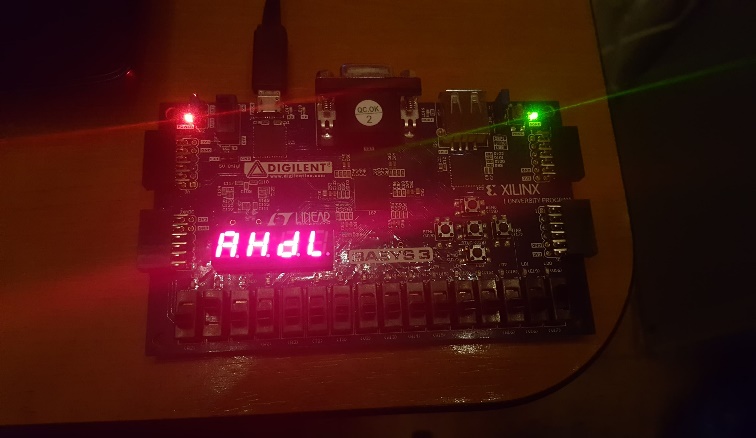
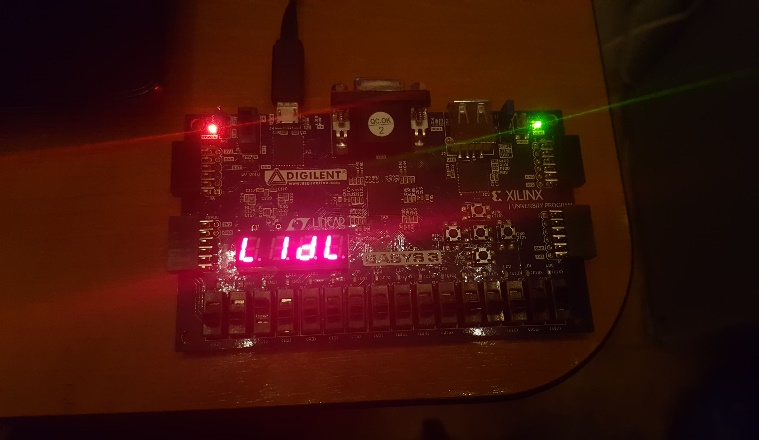
1. Plugging the FPGA board into the computer/laptop
2. Adding the VHDL sources to a newly created project
3. Creating a constraint file (with the extension .xdc), in which the signals of the Complete System Unit entity are configured and assigned to the desired pins
4. Synthesizing and implementing the design by pressing the following buttons in this exact order: “Run Synthesis”, “Run Implementation”, “Generate Bitstrem”, “Open Hardware Manager”, “Open Trace (AutoConnect)”, “Program Device”.

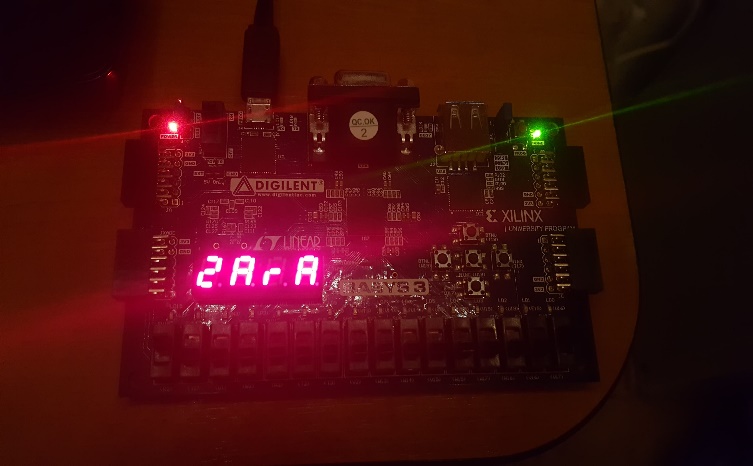
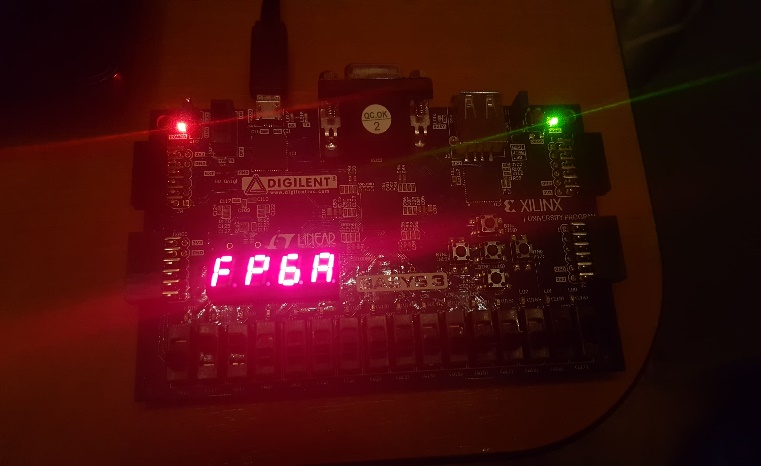
Once the “Program Device” button is pressed, the design is already running on the board.

After turning the ENABLE switch on, the text “OrEO” (OREO - the first message) is shown on the board:



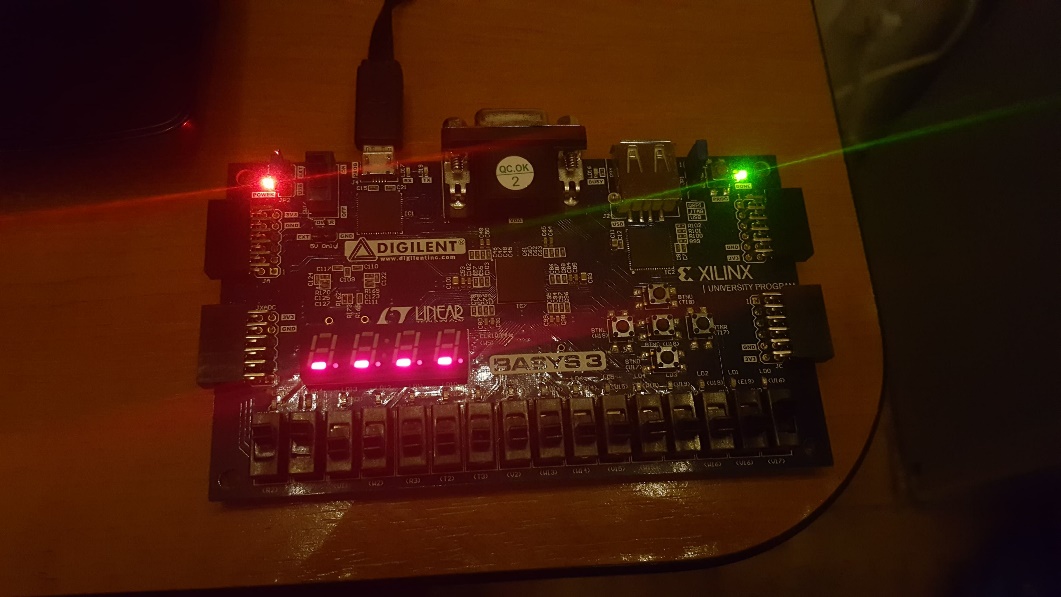
After playing around with the switches to which the MESSAGE\_SEL[2:0] signal is assigned, one can get the following outputs:





* “LIdL” for the second message
* “AHdL” for the third message
* “FPGA” for the fourth message
* “ZArA” for the fifth message

Since only 5 messages have been stored for the adverstisement, in case MESSAGE\_SEL[2:0] is assigned any of the values “101”, “110” or “”111”, the display will look like this:



As mentioned previously, there are 8 types of animations available, and they are obtained by switching ON and OFF the commutators corresponding to the ANIM\_SEL[2:0] signal.

**Animation 0: Static effect/Plain display of text** –pictures of it are presented above

**Animation 1: Shifting of the letters from RIGHT to LEFT**

****

**Animation 2: Shifting of the letters from LEFT to RIGHT**

**Animation 3: Flickering of the entire text**

****

**Animation 4: Display of the text letter by letter**, all the letters being shown on one single anode



**Animation 5: ”Flowing” of the text from left to right**

****

**Animation 6: Intermittent blinking effect of 2 letter-pairs: the first and last 2 letters**

****

**Animation7 : Intermittent blinking effect of 2 letter-pairs: the middle and outer 2 letters**



VI. ULTERIOR DEVELOPMENTS

Some possible modifications for the further development of the system include the following:

* Allowing the user to enter the advertised text manually, by introducing some peripherals such as a keyboard or a phone to serve as input devices
* Replacing the 7-segment displays with a wider LCD/a billboard
* Switching to a Random Access Memory (RAM) instead of a ROM so that the modification of the messages would be easier to perform
* Allowing the user to choose the speed with which the letters are being displayed
* Implementing the possibility for the user to customize the alphabet, expand it and/or add symbols that are not necessarily letters to it

VII. BIBLIOGRAPHY

1. <https://en.wikipedia.org/wiki/Aldec>
2. <https://en.wikipedia.org/wiki/VHDL#Advantages>
3. <https://reference.digilentinc.com/basys3/refmanual>