**FIFO MEMORY**

Student: Halmai Erik

Group: 30412

Semigroup: I

Contents

[1 Contents 2](#_Toc9419119)

[2 Project Specification 3](#_Toc9419120)

[3 Block Scheme 3](#_Toc9419121)

[4 Components 5](#_Toc9419122)

[4.1 Control 5](#_Toc9419123)

[4.2 Status 6](#_Toc9419124)

[4.3 Memory 7](#_Toc9419125)

[4.4 BCD 7 Segment Decoder 8](#_Toc9419126)

[4.5 Frequency divider 10](#_Toc9419127)

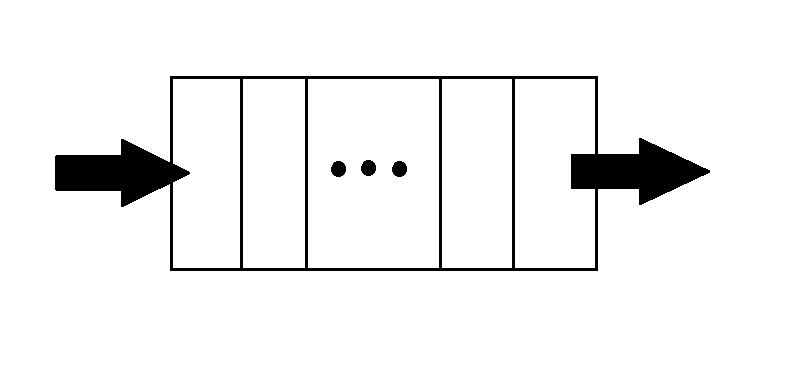
[5 Constraints file 10](#_Toc9419128)

[6 Instructions for use 11](#_Toc9419129)

[7 Upgrade Possibilities 16](#_Toc9419130)

# Project Specification

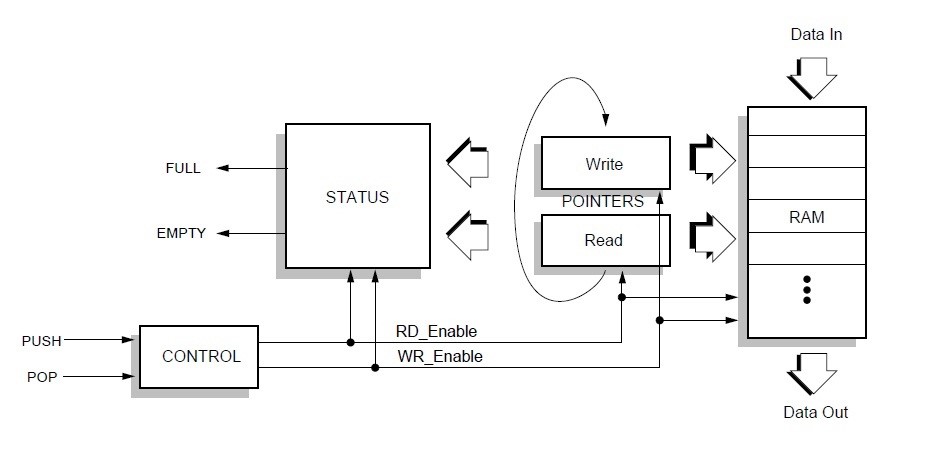
The project’s requirement is to implement a FIFO Memory using the documentation from the book *”Proiectarea sistemelor numerice folosind tehnologia FPGA”* by Sergiu Nedevschi, Zoltan Baruch and Octavian Creț.

 The FIFO Memory uses *First In First Out* or queue logic. This means that the first element read will be the first one displayed. This is done with two operations: push and pop. The push operation does the reading of the information into a RAM memory, while with pop we can display the information on the FPGA display.

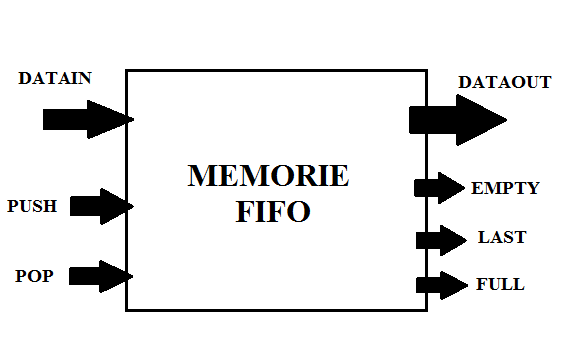
**Figure 1**

# Block Scheme

The FIFO Memory can be implemented using five logical blocks as you can see in figure 2. The “Control” block generates the validation signals of writing or reading. The “Status” generates the status of the memory (full or empty). The two pointers generate the addresses for writing and reading into the RAM memory.

The black box of the project can be seen in Figure 3.

**Figure 2**



**Figure 3**

The internal scheme of the FIFO Memory can be seen in Figure 4.

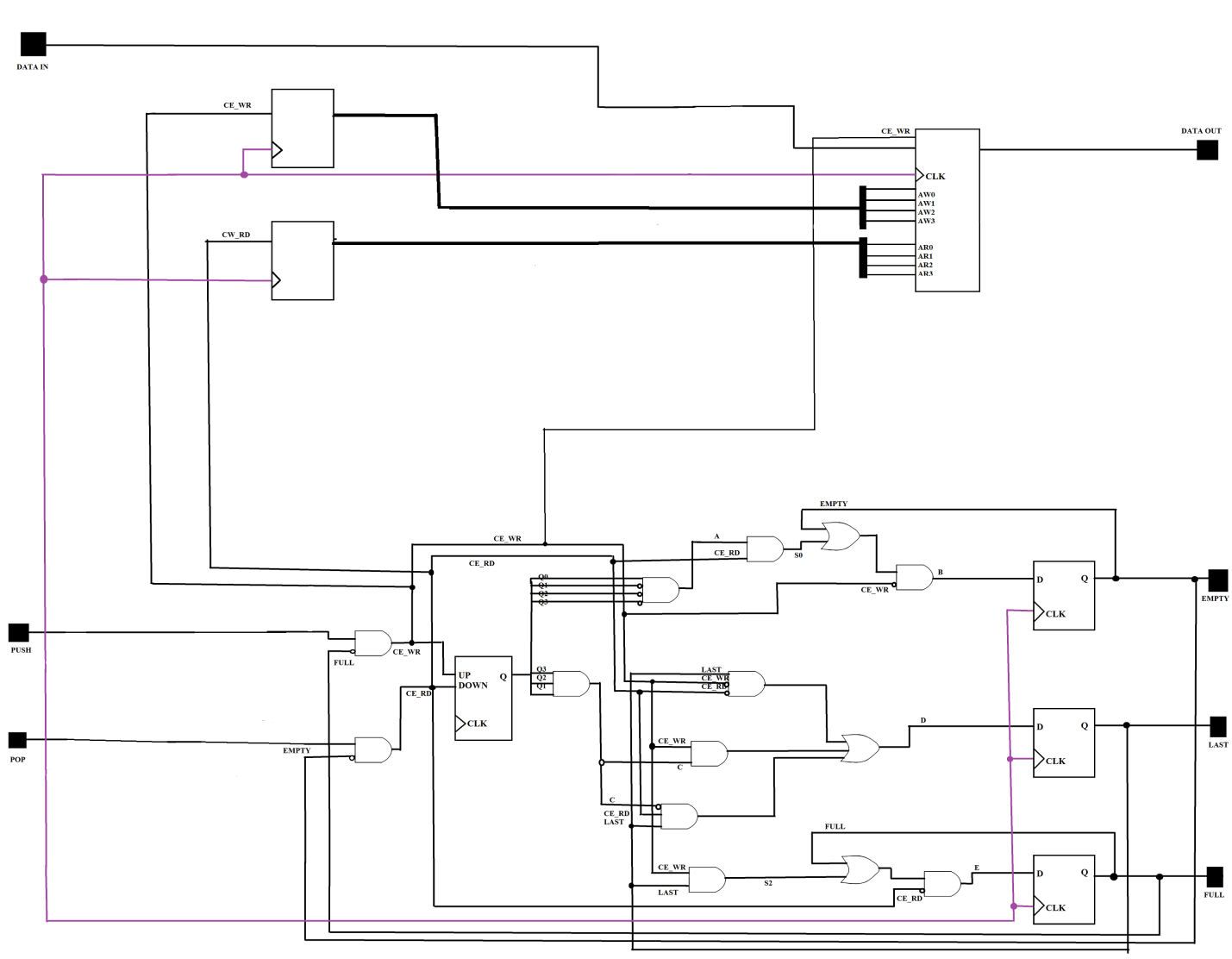


Figure 4

# Components

## Control

The control component creates two main signals: RD\_ENABLE for the the push operation, which adds a new element to the memory and WR\_ENABLE for the pop operation, which dispalys an element from the memory. This is done by some simple and gates using the inputs EMPTY, FULL and also POP and PUSH, so that you can’t pop and push at the same time.

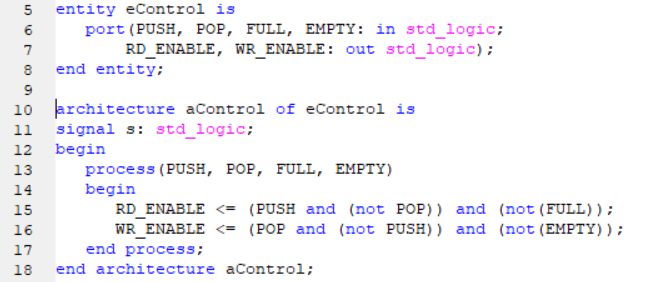


Figure 5

## Status

The status component’s main task is to show the user the state of the FIFO Memory. It can show three states, EMPTY, FULL and LAST all connected to three separate LEDs. This is done with the help of a Synchronous Bidirectional Binary Counter, which is implemented as a component of the STATUS block. The counter counts the number of elements in a 16x8 RAM Memory and so, with some simple and gates, it can calculate the three states:

* When the counter is in the 0000 state, EMPTY is activated
* When the counter is in the 1110 state, LAST is activated
* When the counter is in the 1111 state, FULL is activated

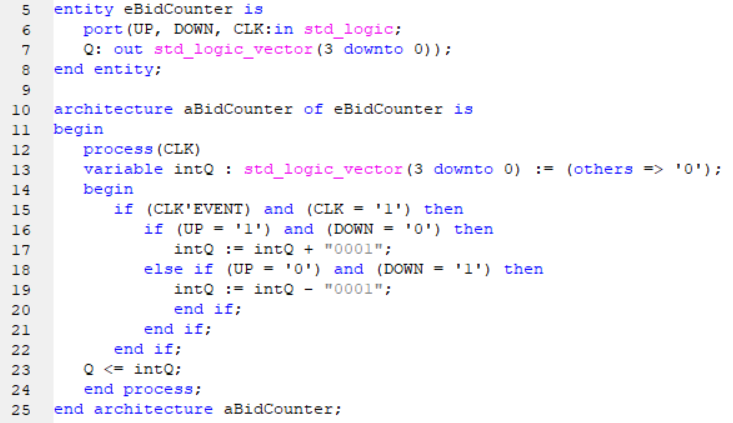
The VHDL code for the Bidirectional Counter can be seen in Figure 6.

Figure 6

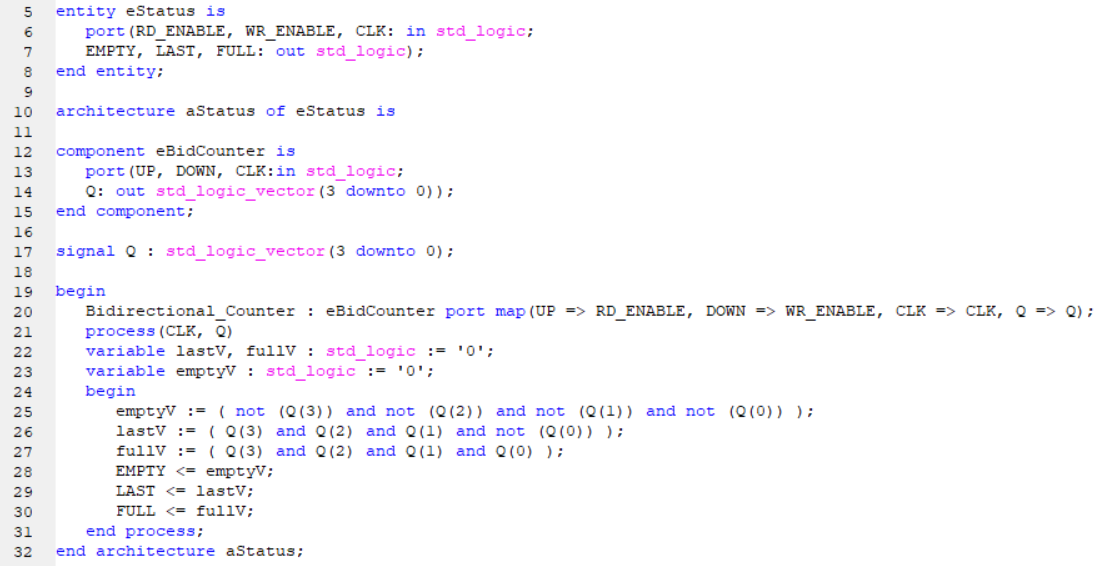
The VHDL code for the Status component can be seen in Figure 7.

Figure 7

## Memory

The Memory block contains 3 components: a RAM Memory, and two Synchronous Binary Counters.

The two counters have the task to calculate values of the address pointers AW and AR. When a value is entered and PUSH is pressed, AR counts up by one and when POP is pressed and a value is displayed, AW counts up by one.

The RAM has a capacity of 16x8 bits and it is dual ported, meaning that it has two address inputs, AW and AR.

The VHDL code for the Synchronous Binary Counter can be seen in Figure 8.

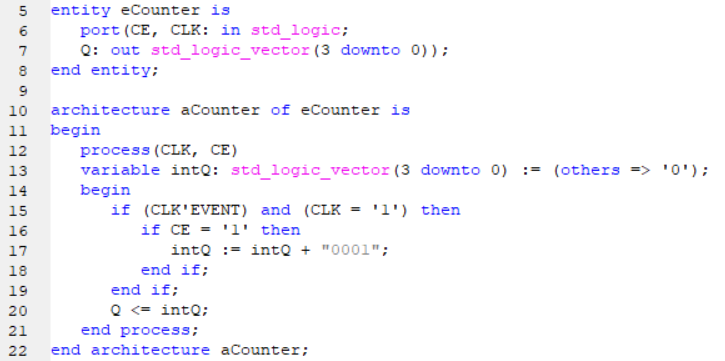


Figure 8

The VHDL code for the RAM Pointers can be seen in Figure 9.

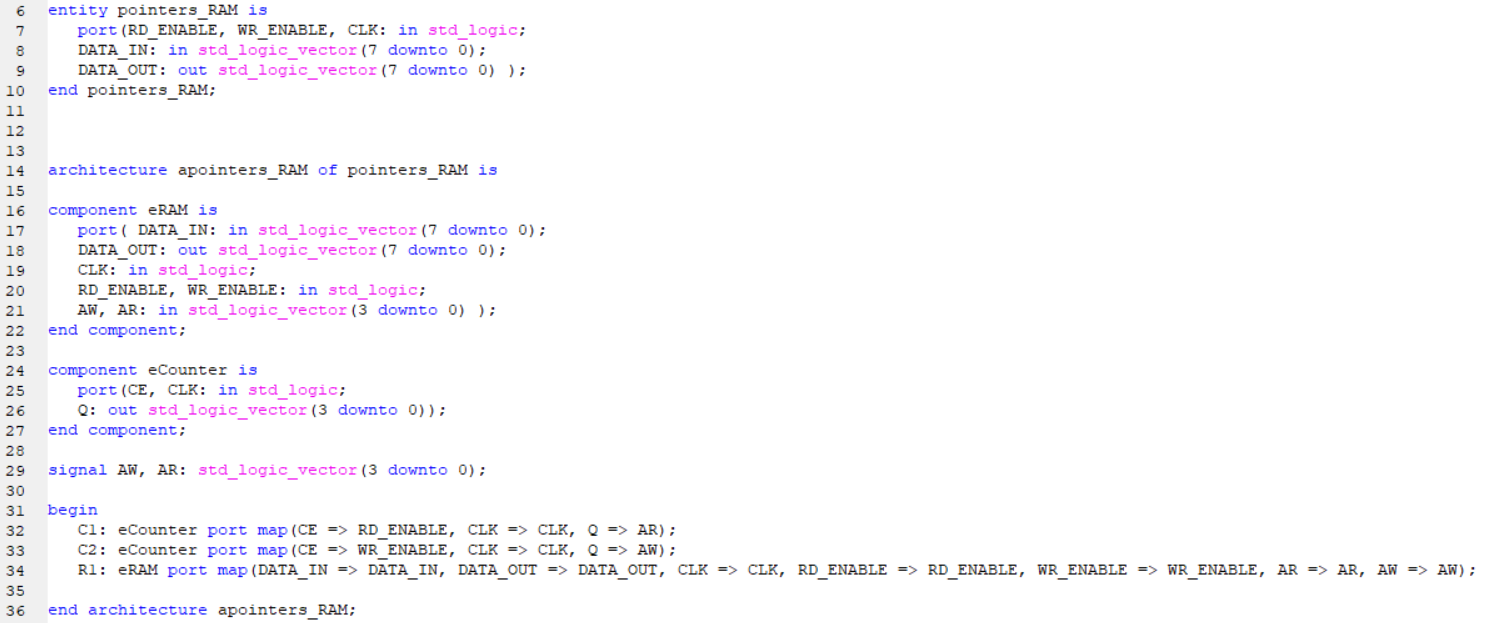


Figure 9

## BCD 7 Segment Decoder

The BCD block is used to display the values of the pushed and popped data.

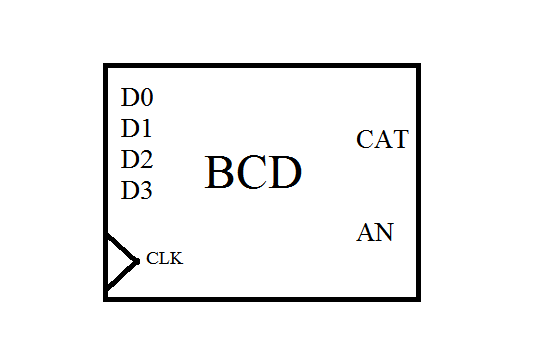
This is done using the 4 separate inputs of the FPGA display called **anod** and the 7 inputs of the BCD 7 Segment Decoder called **catod**, both being active low. These are the two outputs of this component, which are then used as the inputs of the FPGA display. The block uses four inputs: D0, D1, D2, D3, of 4 bits each, which represent the four digits that can be displayed on the FPGA’s display screen. The black box of this block can be seen in Figure 10.

Figure 10

Inside the block there is also a frequency divider for the clock signal, which reduces the internal clock of the Nexys 3 FPGA to 60 Mhz. This way, the user won’t be able to see the fact that the 4 digits are displayed sequentially. The waveform showing this sequential display cam be seen in Figure 11.

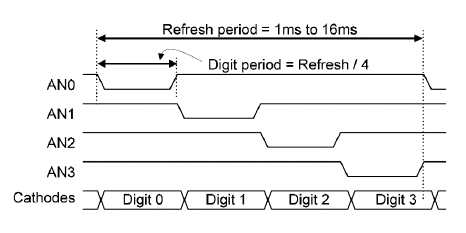


Figure 11

## Frequency divider

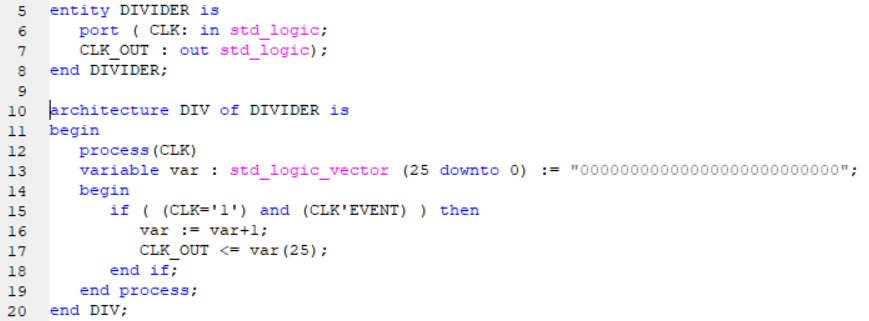
The Divider component is used to slow down the clock signal from 100 Mhz to 40 Mhz. With the clock signal at 40 Mhz errors such as an input being pushed two consecutive times can be prevented from happening. The Divider is done using a std\_logic\_vector type variable of 26 bits. Every time the clock signal is activated (goes from 0 to 1), this vector gets added 1 bit and when the 26th bit is one, the new clock signal, CLK\_OUT, is activated. The VDHL code for this Frequency Divider can be seen in Figure 12

Figure 12

# Constraints file

The project uses the Nexys 3 FPGA’s integrated 100 Mhz clock signal. The constraints file of the project, showing the buttons and LED lights of each input/ output, can be seen in Figure 13.

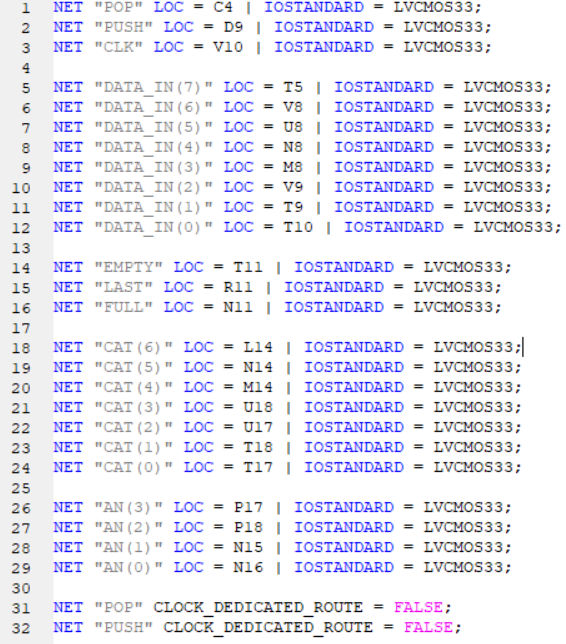
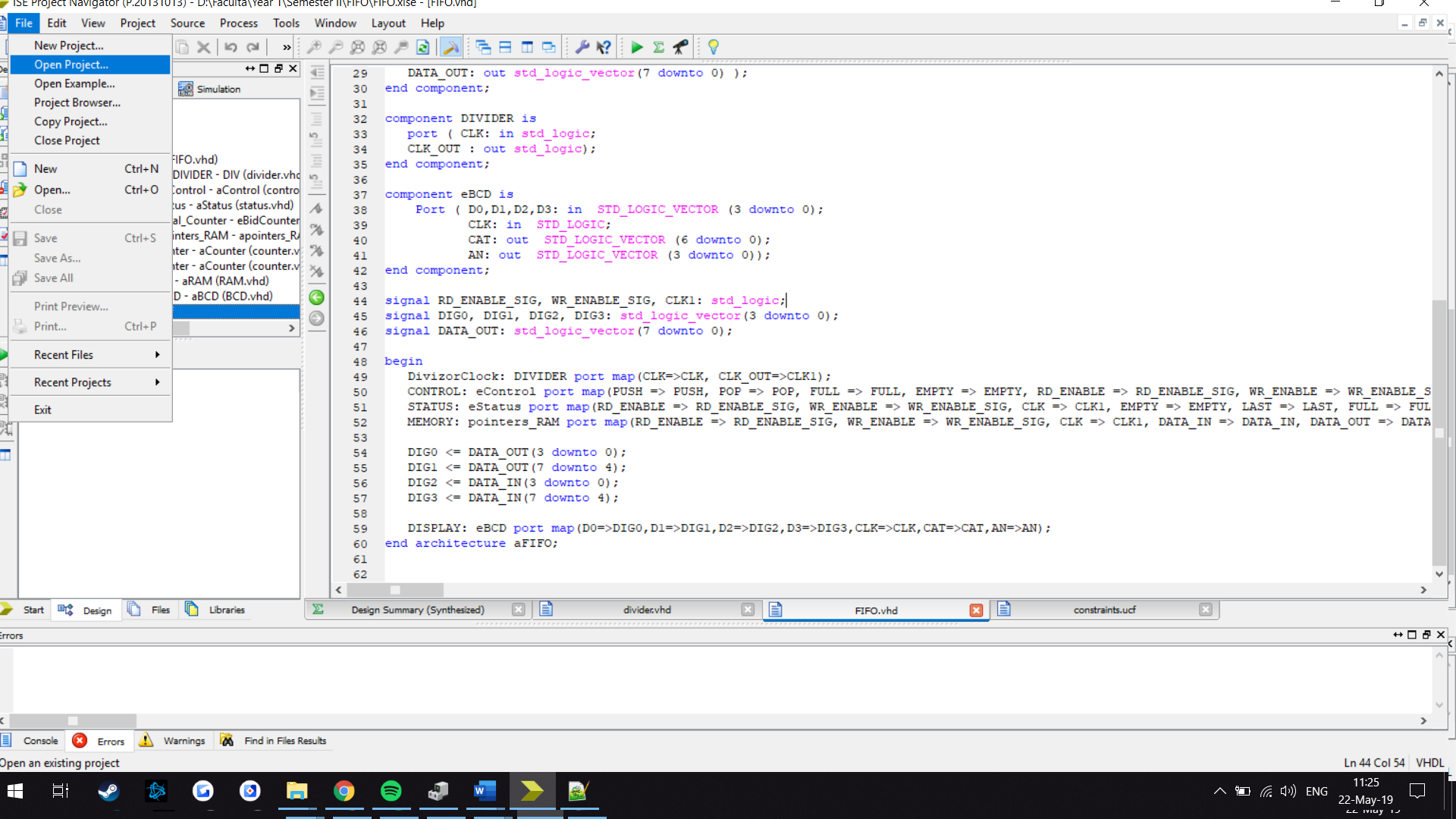
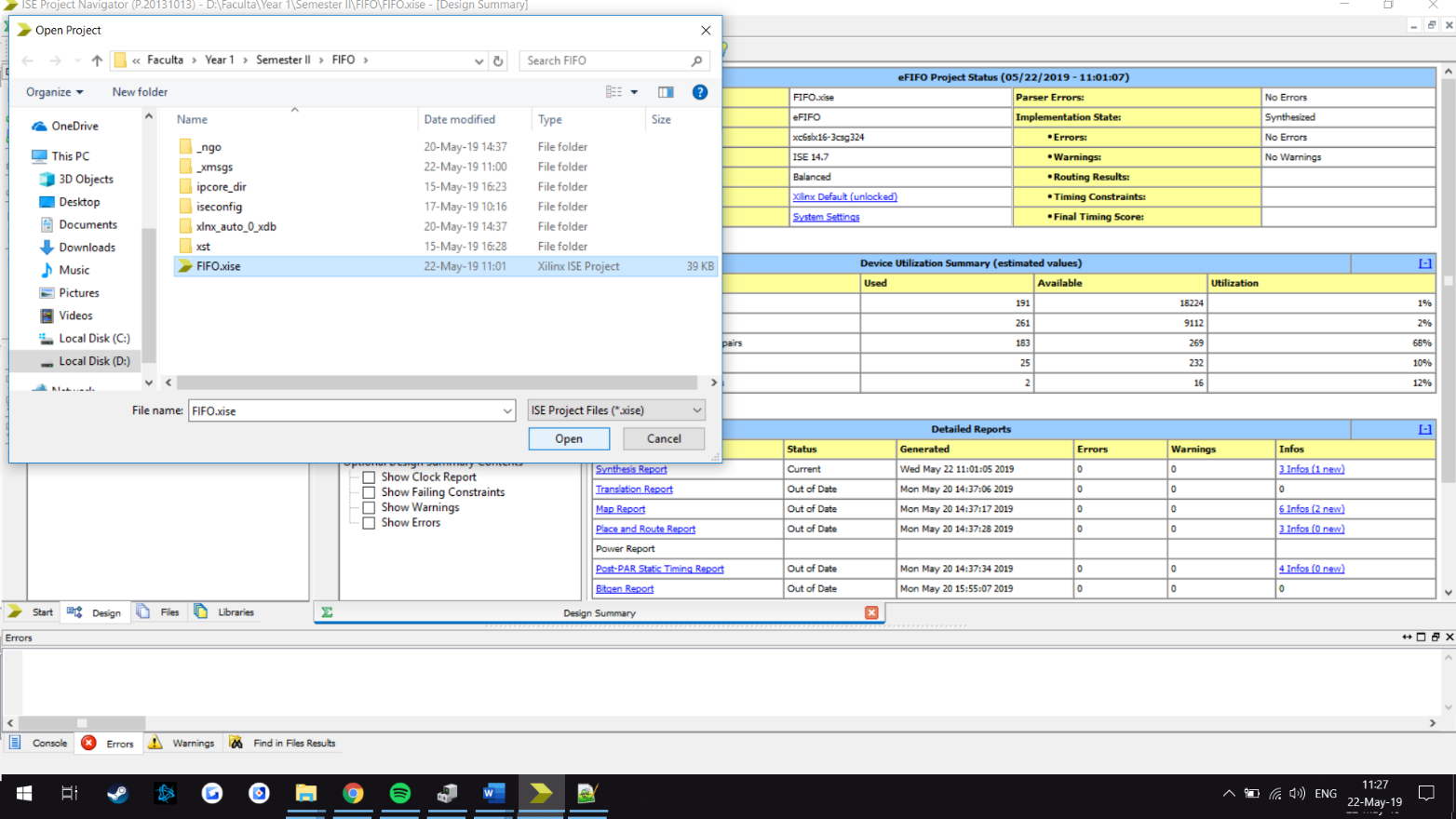
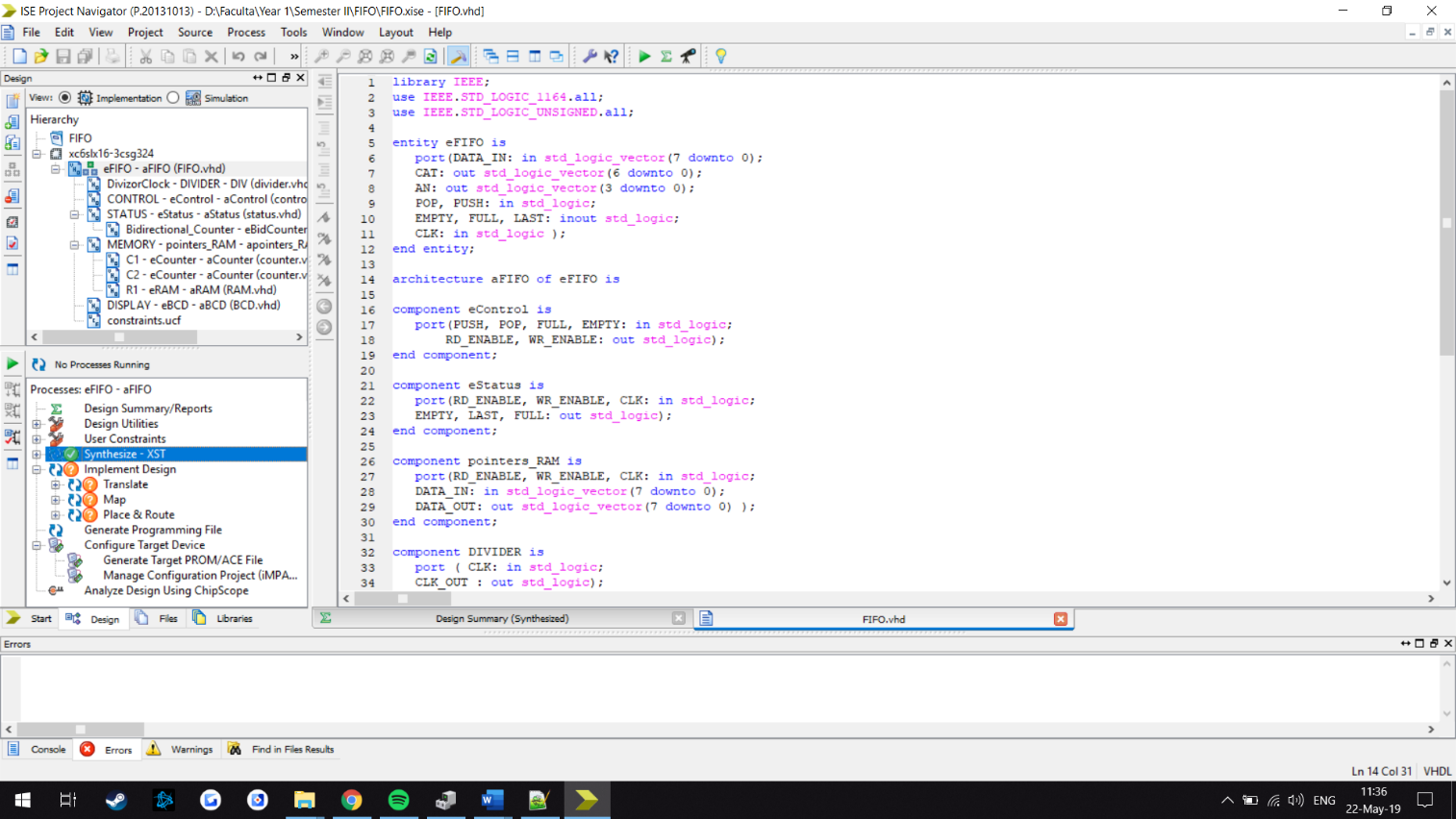
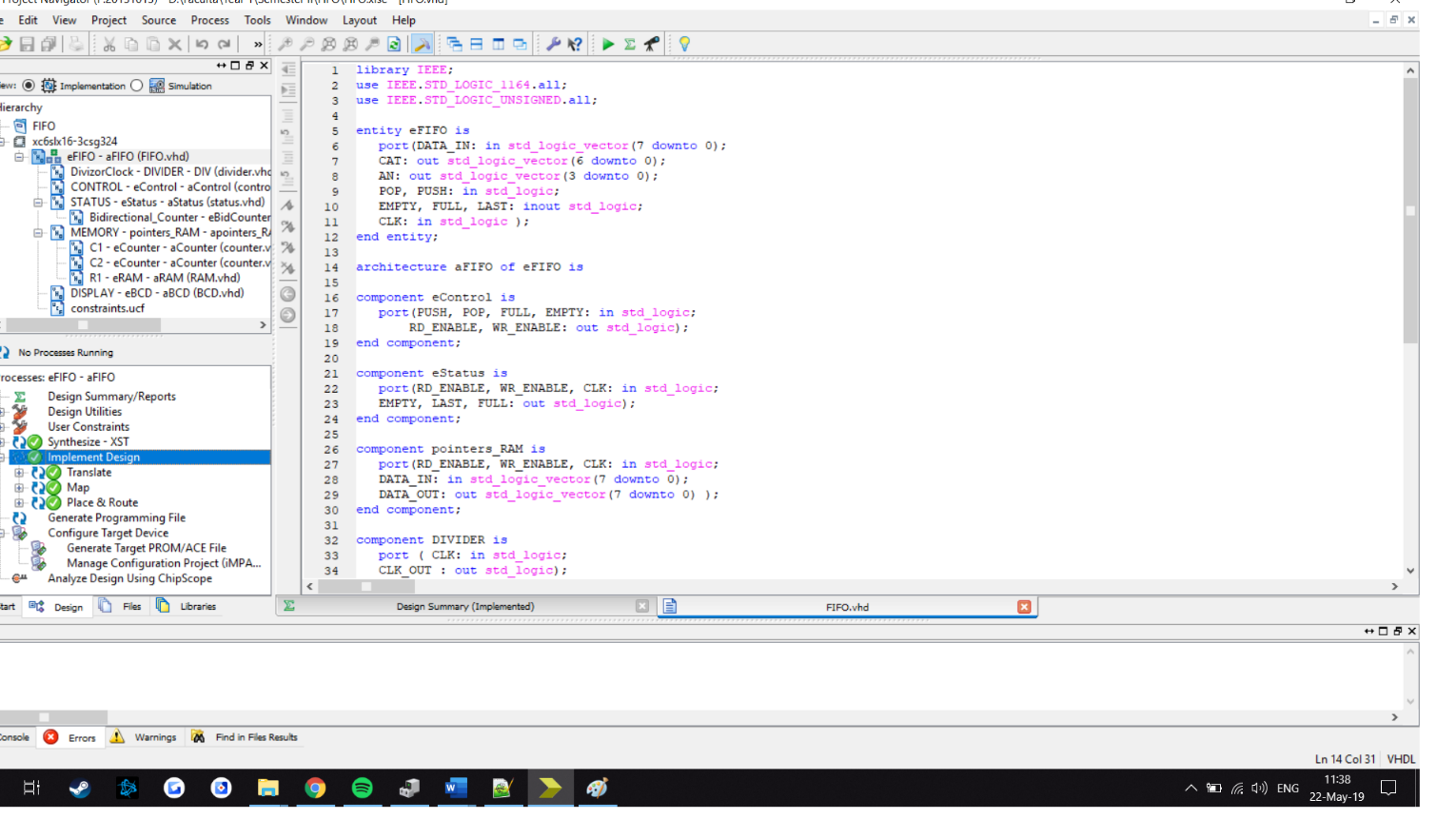
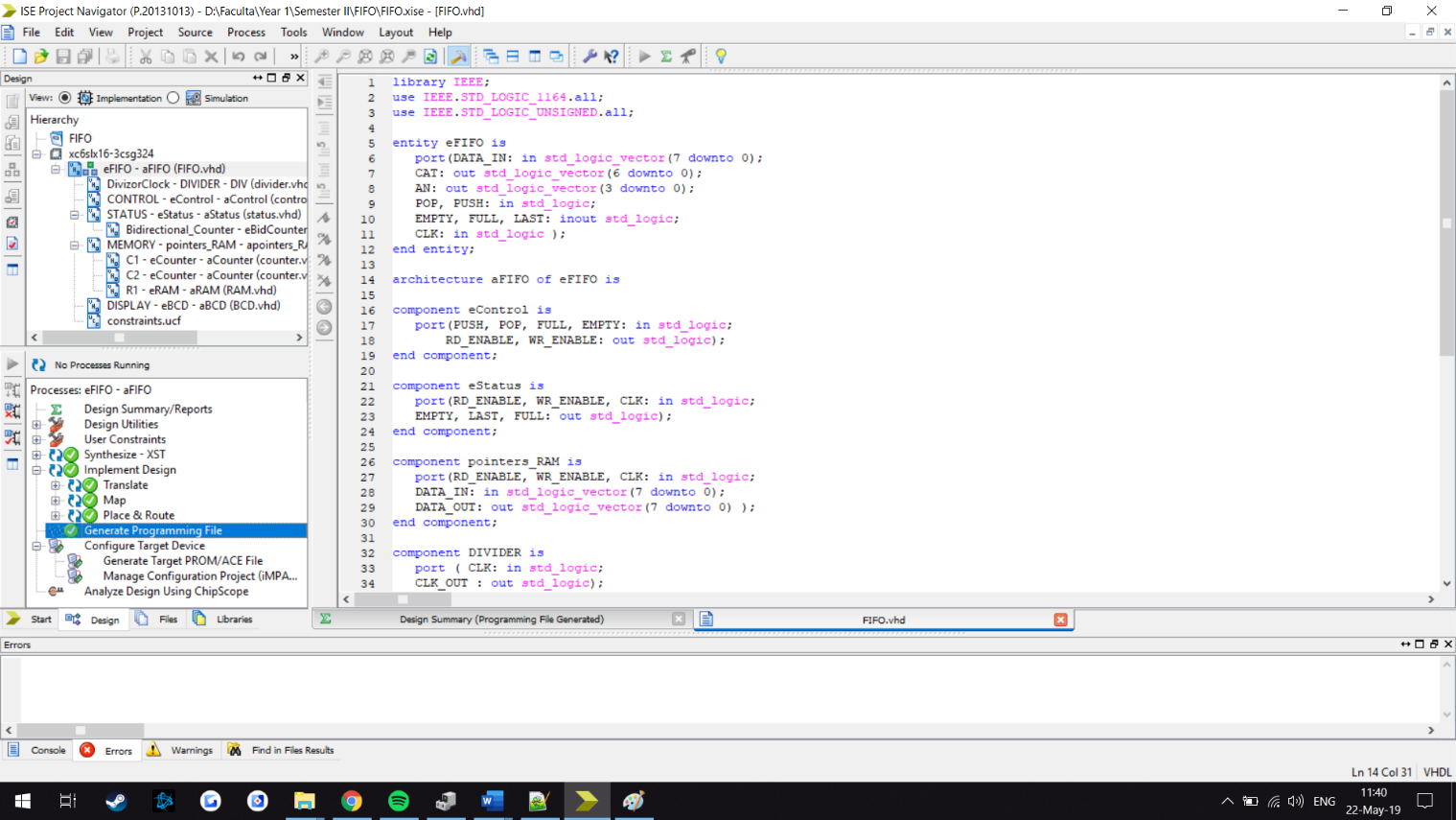


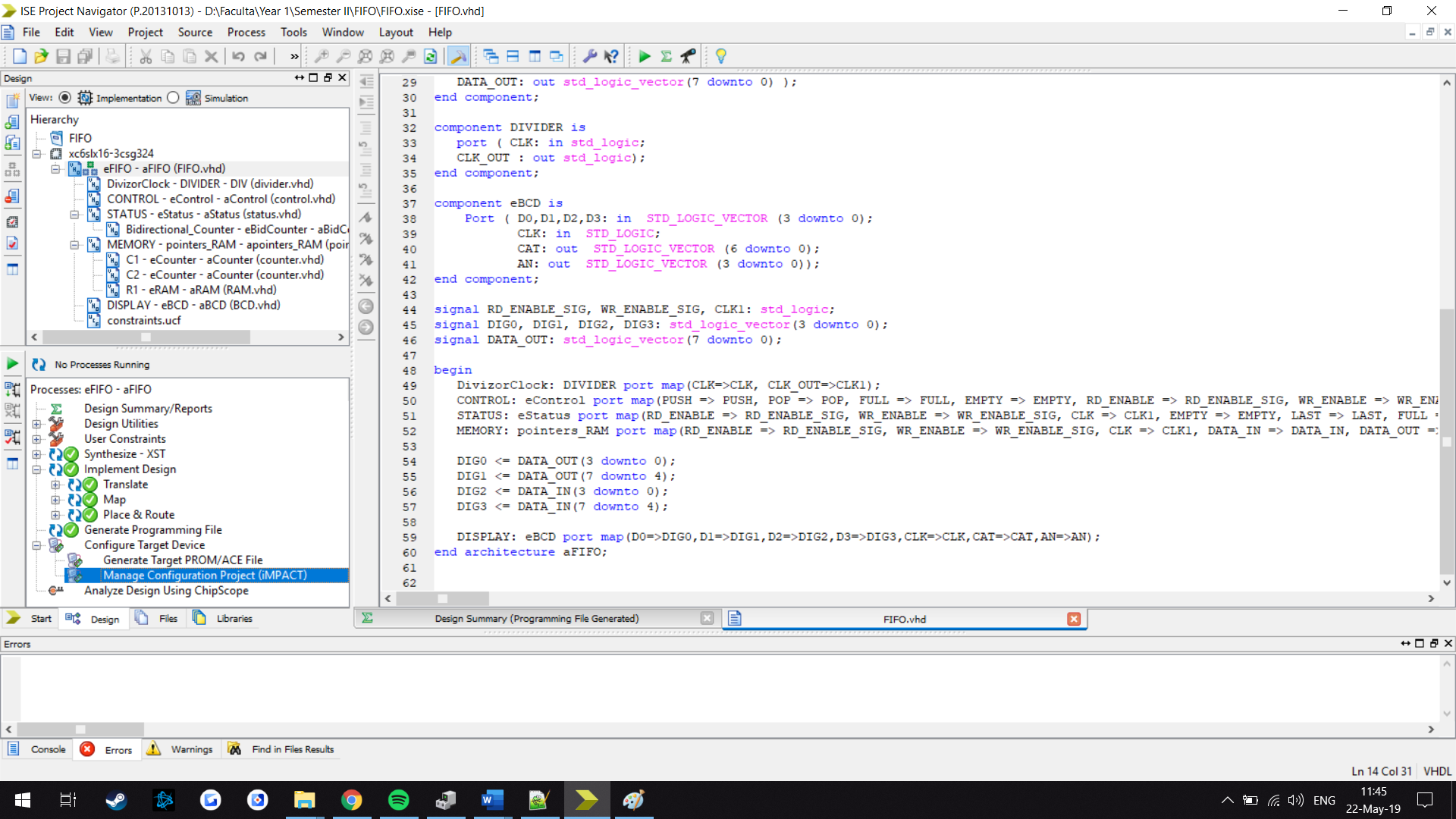
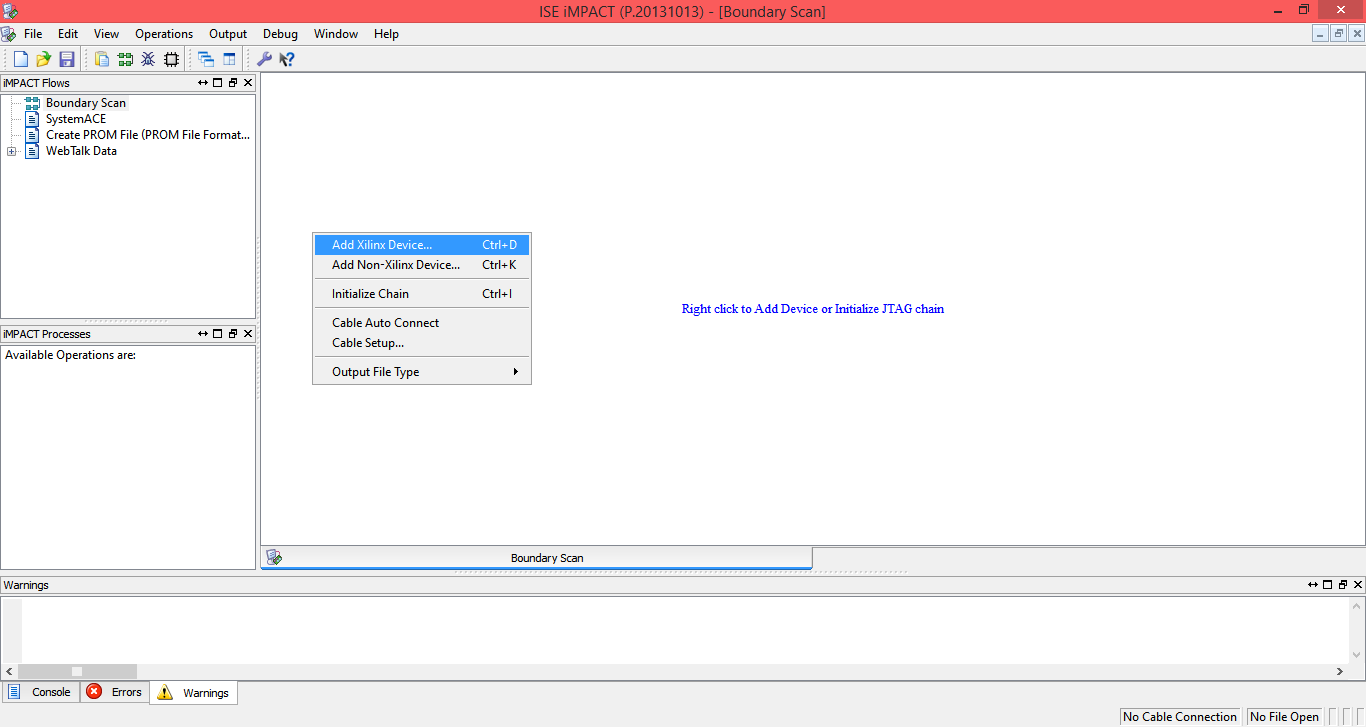
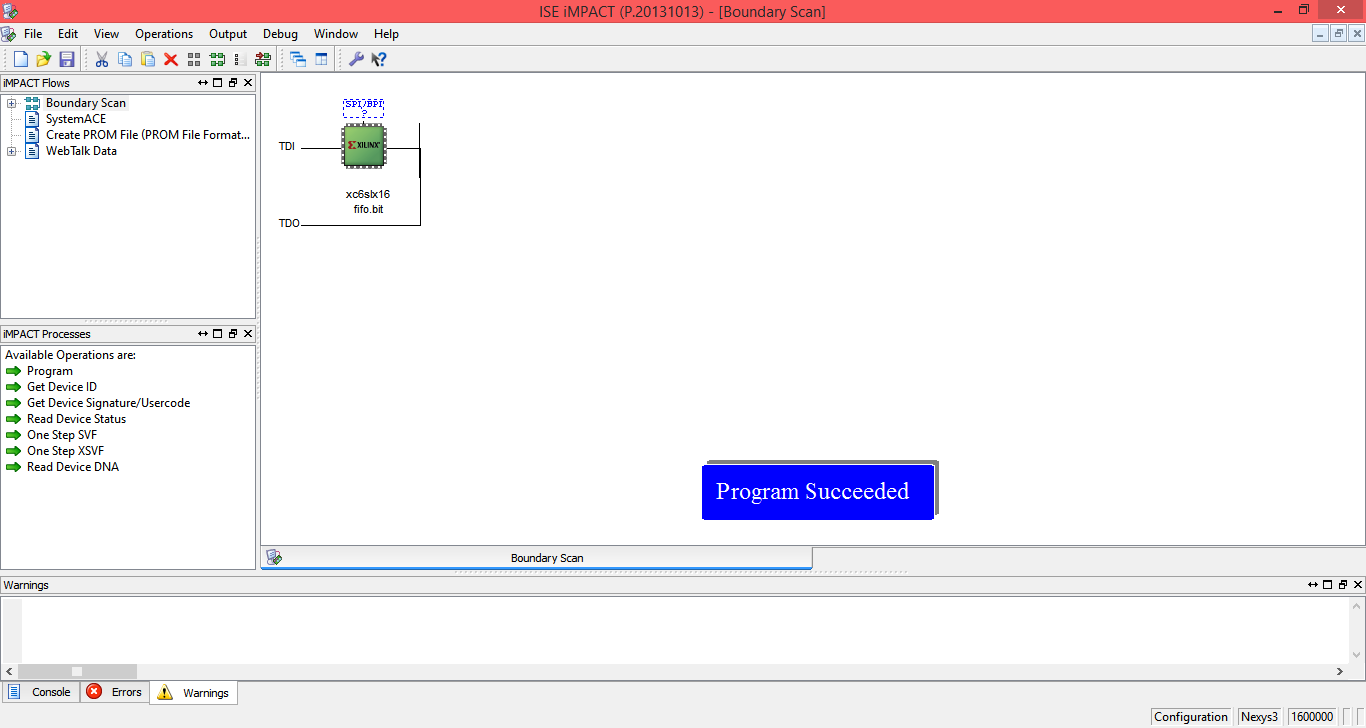
Figure 13

# Instructions for use

1. Install the program ISE Design Suit from the Xilinx site and connect a Nexys 3 FPGA to your device
2. Open the program and click on File/Open Project
3. Select FIFO.xise and click Open
4. In the Hierarchy tab press on eFIFO – aFIFO (FIFO.vhd) and in the Proccesses tab double click on Synthesize and let the project synthesize until a green check appears next to Synthesize – do the same with Implement Design and Generate Programming File after





1. Once all three are marked with a green check, double click on Manage Configuration Project (iMPACT) and ISE iMPACT will open
2. In the ISE iMPACT windows click on Boundary Scan and in the white window that opens right click and select Add Xilinx Device
3. Select FIFO.bit from the window and press Open. Right click on the new icon that appears and press Program. A little blue window should appear saying Program Succeeded

# Upgrade Possibilities

* A RESET button could be added to reset the whole FIFO Memory to its initial state.
* Bugs that might occur could be fixed
* The Memory’s capacity could be upgraded