1 Design and implementation using didactic boards

1.1 Objectives

This work presents the resources necessary for laboratory activity. The basic notions used in modelling the behavior of digital circuits are studied. A step-by-step guide presents the development methodology with Xilinx ISE suite and the implementation on FPGA (Field Programmable Gate Array) boards. The activity will be carried out into an already existing project that integrates the TTL (Transistor-Transistor Logic) circuits. The project is described and analyzed in detail. Several circuits are implemented using basic logic gates and their functionality is tested according to their function (truth) table.

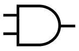
1.2 Resources necessary

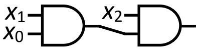
The following resources are required for the implementation and testing of the circuits:

- Nexys A7 development board, The reference manual [1].
- Xilinx ISE WebPACK 14.7 suite [2]; The lab works will be carried out in an already existing project, which contains the TTL circuits.

1.3 Fundamentals on modelling digital circuits behavior

The mathematical background behind modelling the behavior of digital circuits uses concepts from *boolean algebra* (conceived by mathematician George Boole [3]). It contains a set of logical operations, which are commonly used to describe their functionality. A digital circuit has several inputs and outputs, called *terminals* (or pins). Any output can be modeled as a function of the inputs, which means the inputs are the variables of the function. In boolean algebra the result of the function (output) and its variables (inputs) can have two values, either 1 (true) or 0 (false). These functions are also known as *boolean functions*. The values are associated with electrical properties, such as voltage. For transistor-based circuits of the TTL family [4], value 0 is associated with low voltage and 1 is associated with high voltage. The representation is called *positive logic*. When the polarity is reversed (0 means high voltage and 1 means low voltage) the representation is in *negative logic*.

Any boolean function can be described by its *truth table* (also called *function table*). The left side of the table contains all possible combinations of input values. The function value (response) for each combination is placed on the right side. An example of truth table for a 2-variable function is presented in Table 1. 1, left side. According to the table, the associated circuit will output 1 when x_1=1 and x_2=1, hence its name, the AND function. Considering its simplicity, the circuit is also called a *basic logic gate*. In boolean algebra AND uses the · (dot) operator (not to be confused with multiplication). The corresponding graphic symbol used within a circuit is , with inputs on the left and output on the right. There are AND gates with higher number of inputs, which implement

$x_{n-1}\cdot\ldots\cdot x_0$. They respect the same behavior: the output is 1 only when all inputs are 1. Since AND is associative, the extension to a higher number of inputs can be implemented using AND gates with less inputs. For instance, based on the property that $x_2\cdot x_1\cdot x_0 = x_2\cdot(x_1\cdot x_0)$, a 3-input AND gate can be implemented with 2-input AND gates: 

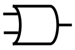
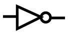
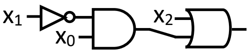
.

Table 1. 1 Truth-tables for AND (left), OR (middle) and NOT (right) gates

x_1	x_0	AND
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_0	OR
0	0	0
0	1	1
1	0	1
1	1	1

x	NOT
0	1
1	0

There are other types of basic logic gates, such as OR and NOT (also called INV). The operator for OR is represented by a + (attention! it doesn't represent summation). The expression x_1+x_0 will be 1, only if $x_1=1$ or $x_0=1$ (see Table 1. 1, middle). The graphic symbol for OR is . NOT is a 1-input gate and inverts its value. The representation in boolean algebra is NOT = $\overline{\mathrm{x}}$. Its symbol is . AND, OR and NOT gates implement the basic operations in boolean algebra. By repeated interconnections, an unlimited number of circuits can be implemented, representing boolean functions with any number of input variables. For instance, the circuit implementing $\mathrm{x}_2 + (\overline{\mathrm{x}_1}\cdot \mathrm{x}_0)$ is 

.

The priority of the operators inside a boolean expression is: 1) brackets; 2) NOT; 3) AND; 4) OR. The operations inside the circuit follow the rules from the truth tables, in left to right order, from inputs to outputs. Considering $x_2=0$, $x_1=0$ and $x_0=1$, and replacing their values inside the expression, the result is $\mathrm{x}_2 + (\overline{\mathrm{x}_1}\cdot \mathrm{x}_0) = 0 + (\overline{0}\cdot 1) = 0 + (1\cdot 1) = 0 + 1 = \mathbf{1}$. Similarly, the expression can be computed for any combination of input values, and the truth table can be generated. Several pairs of input values are presented as follows, with the corresponding computation of the expected values:

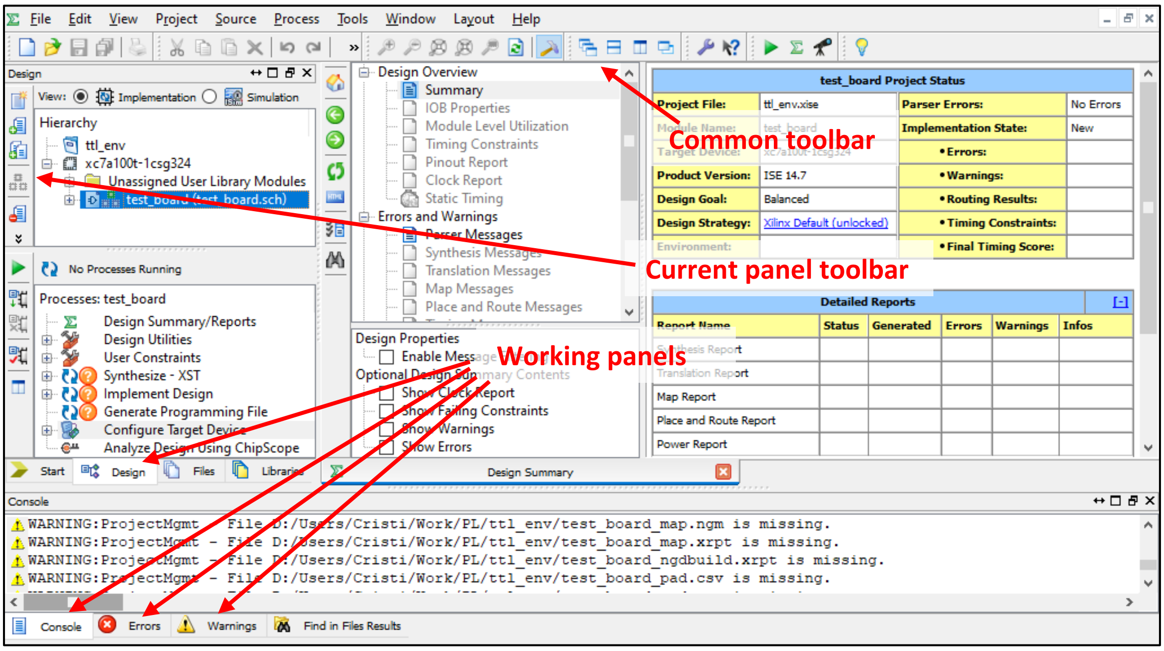
- $x_2=1, x_1=0, x_0=0$ → $\mathrm{x}_2 + (\overline{\mathrm{x}_1}\cdot \mathrm{x}_0) = 1 + (\overline{0}\cdot 0) = 1 + (1\cdot 0) = 1 + 0 = \mathbf{1}$;
- $x_2=0, x_1=1, x_0=1$ → $\mathrm{x}_2 + (\overline{\mathrm{x}_1}\cdot \mathrm{x}_0) = 0 + (\overline{1}\cdot 1) = 0 + (0\cdot 1) = 0 + 0 = \mathbf{0}$;
- $x_2=0, x_1=0, x_0=0$ → $\mathrm{x}_2 + (\overline{\mathrm{x}_1}\cdot \mathrm{x}_0) = 0 + (\overline{0}\cdot 0) = 0 + (1\cdot 0) = 0 + 0 = \mathbf{0}$.

1.4 Xilinx ISE Project Navigator – a step-by-step guide

The Project Navigator application is part of Xilinx ISE suite and offers support for designing digital circuits using a schematic editor.

1.4.1 Loading the *ttl_env* project

After launching Project Navigator, the **ttl_env** project must be loaded from File > Open Project... in the menu and selecting ttl_env.xise project file. Next, the application will display several working panels, as in Figure 1. 1. Some panels on the lateral side are dedicated to project management and those at the bottom offer information related to logic diagram interpretation and processing [**Note**: If the graphical user interface differs it can be reset to default from Layout > Load Default Layout.]:



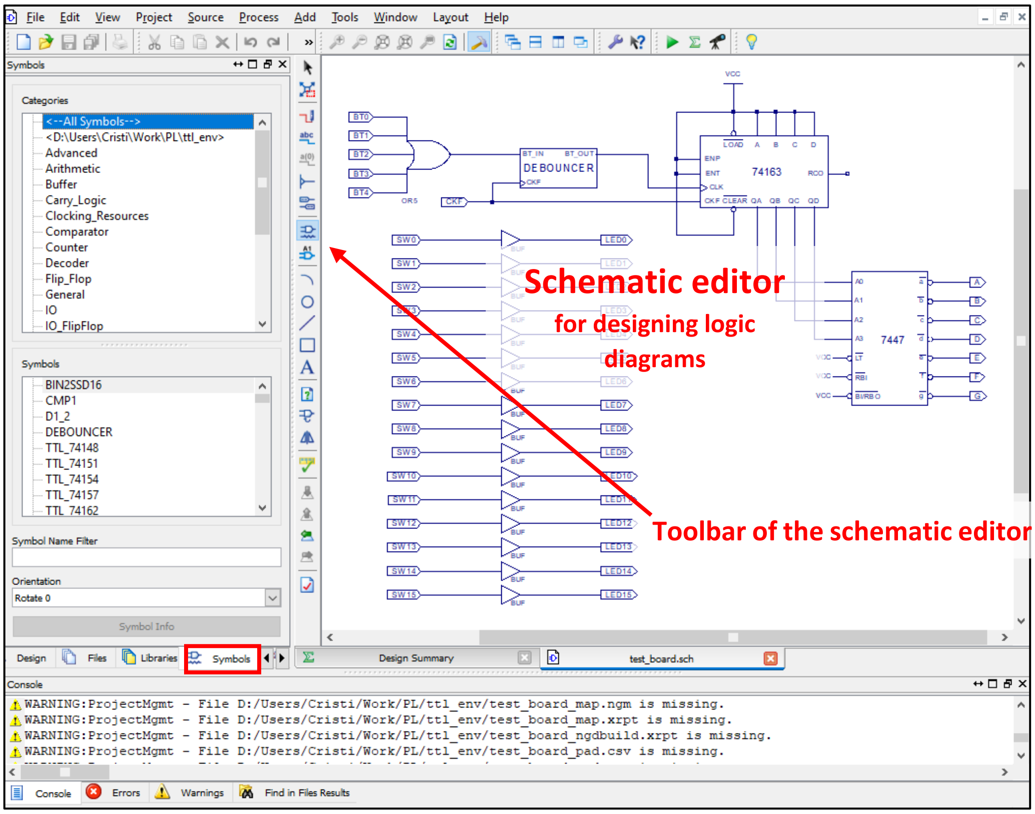
test_board Project Status			
Project File:	ttl_env.xise	Parser Errors:	No Errors
Module Name:	test_board	Implementation State:	New
Target Device:	xc7a100t-1csg324	• Errors:	
Product Version:	ISE 14.7	• Warnings:	
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:		• Final Timing Score:	

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Power Report					

```
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_map.ngm is missing.
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_map.xrpt is missing.
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_ngdbuild.xrpt is missing.
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_pad.csv is missing.
```

Figure 1. 1 Project Navigator interface

- **Design** panel (Figure 1. 1) – highlights the project structure organized as a set of interconnected circuits. Initially, the project contains the test circuit called test_board. With double-click on the name of the circuit the schematic editor will appear and the Symbols panel will open.
- **Symbols** panel (Figure 1. 2) – contains the list of components that can be integrated into the circuit diagram, represented by their symbols. It can be activated by clicking on Add Symbol in the toolbar of the schematic editor. In this panel the symbols are organized into several categories. Selecting a category in the upper layer, you will notice the list of corresponding circuits bellow. When selecting <--All Symbols--> at the top, the list of all available components will appear. A component can be searched into the current category by introducing its name in the Symbol Name Filter box. The most relevant categories are *Arithmetic*, *Buffer*, *General*, *Logic*, *Mux* and *Decoder*. The category next to <--All Symbols--> represents the local library attached to the project and contains the circuits from the TTL family. **Note**: The name of these circuits starts with **TTL_** followed by the code of the circuit.
- **Console**, **Errors**, **Warnings** (Figure 1. 1) – these panels display relevant information when the diagram is processed. It is useful for debugging features. The **Console** displays all messages generated in the process, while **Errors** and **Warnings** are related to specific messages. A typical warning appears when some inputs are not connected. In such a case the inputs will be automatically connected to 0 (Ground – GND). For connectionless outputs, Project Navigator warns about pruning the circuitry. Unlike errors, the warnings do not negatively affect the implementation process.



Schematic editor for designing logic diagrams

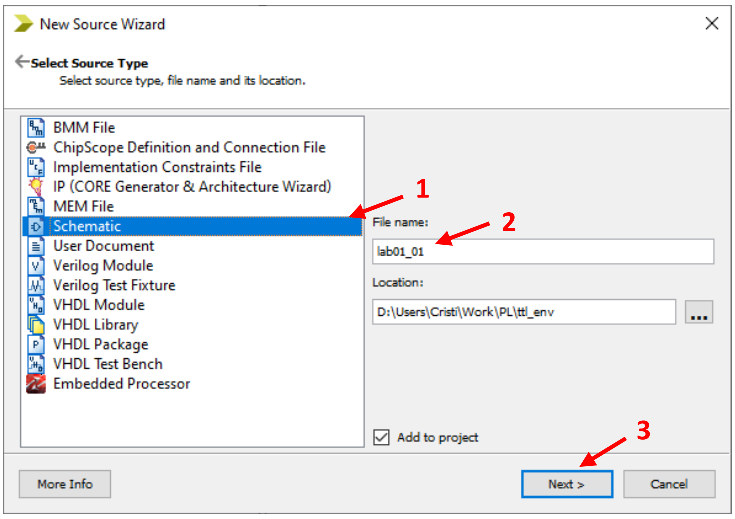
Toolbar of the schematic editor

```
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_map.ngm is missing.
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_map.xrpt is missing.
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_ngdbuild.xrpt is missing.
WARNING:ProjectMgmt - File D:/Users/Cristi/Work/PL/ttl_env/test_board_pad.csv is missing.
```

Figure 1. 2 Symbols Panel and the Schematic Editor

1.4.2 Creating a new diagram

A blank diagram can be added to the project by pressing New Source button in the toolbar of the Desing panel. In the next step, the type of source will be set to Schematic, and the name of the diagram will be chosen (without special characters or spaces): e.g., **lab01_01** (Figure 1. 3). **Note**: Verify that Add to project is checked. Then press Next and Finish. The schematic editor will highlight the empty diagram, which can be added new components and connections. Zooming is possible using Zoom In and Zoom Out in the common toolbar at the top.



Select source type, file name and its location.

File name: lab01_01

Location: D:\Users\Cristi\Work\PL\ttl_env

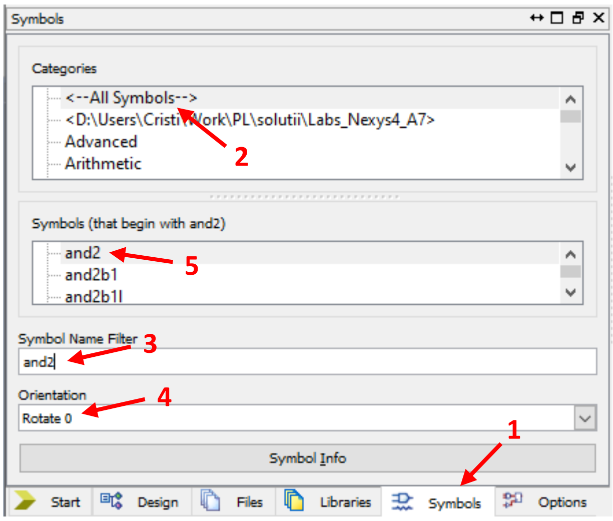
☑ Add to project

Figure 1. 3 Setting up a new diagram

When the project has several diagrams attached, only one can be set active. Setting **lab01_01** active is possible by right-clicking its name in the Desing panel and choosing Set as Top Module from the list of options. The active diagram is marked by the symbol in the project hierarchy. The active diagram can be changed any moment.

A diagram can be removed by right-clicking its name and choosing Remove from the list of commands.

Inside the diagram you will introduce an AND component with two inputs. In the Symbols panel select <--All symbols--> at the top. The search will include the entire list of components. Then type *and2* (2 indicates the number of inputs) in the Symbols Name Filter (Figure 1. 4). Click on **and2** in the filtered list and finally, click on the diagram to place the gate. You can add more gates by clicking several times. (**Note**: You can use the Orientation setting to change the symbol appearance before being placed on the diagram.)



Symbols

Categories

<--All Symbols-->
<D:\Users\Cristi\Work\PL\solutii\Labs_Nexys4_A7>
Advanced
Arithmetic

Symbols (that begin with and2)

and2
and2b1
and2b1l

Symbol Name Filter: and2

Orientation: Rotate 0

Figure 1. 4 Steps required for selecting **and2** in the list of symbols

The selection of a component within the diagram is possible by enabling the selection mode with a click on Select button. Once selected, an element can be drag-and-dropped and can be removed with right-click and Delete (or simply press the *Del* key).

Next, you will introduce wires on the inputs and outputs of AND2 gate. Click on Add Wire in the toolbar to enter the wiring mode. The wire can be placed by drawing it between the interconnected points. You should notice a specific mouse hover near connection points, meaning a click will generate a connection in place. A wire is finished when both ends are connected. An unconnected end is marked by a red rectangle.

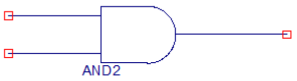


Figure 1. 5 Wires attached to AND2 having unconnected end points

At unconnected ends you will place input and output terminals. The input terminals are required to introduce signal values (0 or 1) in the circuit. The output terminals carry the results outside the circuit. Click on the Add I/O Marker button and place terminals near the end points of the wires to mark connections. (**Note**: Terminals can be attached directly to the pins of a symbol without wires.) The proper type of terminal (input or output) is automatically chosen by Project Navigator. The name of the terminal is automatically generated, nevertheless it is possible to change it by right-clicking the terminal and choosing Rename Port. The name should not contain special characters or spaces. Rename the input terminals to A and B and the output terminal to F1 (Figure 1. 6). **Note**: Wires connected to terminals carry their names, automatically. **Important**: Wires with a similar name are considered connected (a *logical connection*).

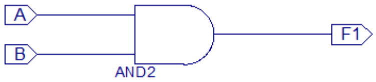


Figure 1. 6 The AND2 gate connected to input and output terminals

A new output function F2 will be added to the diagram. This function will extend the AND operation to 3 inputs using the already existing AND gate. Consequently, a new AND gate will be added to the diagram. The two inputs of this gate will be connected, one to the output of the first AND, and one to a third input terminal C. Its output will be connected to a second output terminal F2. The steps required are as follows:

1. Add a new AND2 gate to the diagram from the Symbols panel.
2. Connect the output of the previous AND2 to one of the inputs of the new gate. Doing so, will generate a new branch on the wire connected to F1, marked by a rectangle. **Note**: All branches represent the same wire and always carry the same value.
3. Connect wires to the free pins of AND2.
4. Link a C terminal on the unconnected input wire and a F2 terminal on the unconnected output wire. The result should be as in Figure 1. 7.
5. Save the diagram using Ctrl+S or File > Save from the menu.

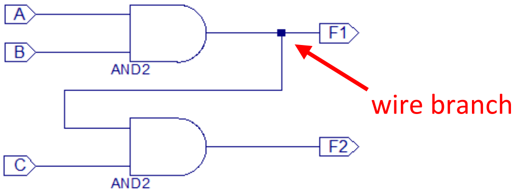


Figure 1. 7 Circuit with two output functions: F1=A · B and F2=(A · B) · C

Next, connect the input terminal C to a new output terminal named F3. **Note**: A direct connection from an input terminal to an output terminal is only allowed though a buffer called BUF. BUF can be found in the category *General*. After adding a BUF, the circuit should be as highlighted in Figure 1. 8.

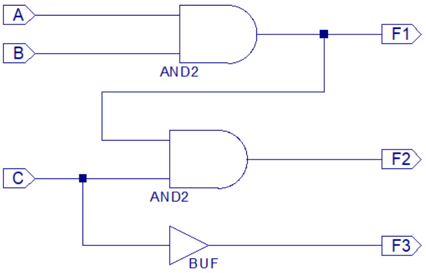


Figure 1. 8 Circuit with three output functions: F1=A · B, F2=(A · B) · C and F3=C

1.4.3 Defining design constraints

The circuits designed using the schematic editor are tested on the board shown in Figure 1. 9. It has **16 switches** and **5 push buttons** for generating logical values on the input terminals. The results registered on output terminals can be visualized on **16 leds**.

When pushed, a **button** generates 1, otherwise it always generates 0. A **switch** generates 1 when turned towards the leds and 0 on the other side. A **led** is off when receiving 0 and lights up when receiving 1. The switches and the leds are indexed from 0 to 15, in right to left order.

The input and output terminals in a diagram are associated (constrained) to buttons, switches and leds on the board. The correspondence is defined in a text file having extension .ucf (User Constraints File). Every diagram must be assigned a constraints file. The file is not case sensitive. To ease the task of elaborating the constraints file from scratch there is a _*template.ucf* file within the project folder, which entails all possible resources, in comments. A comment starts with character #. Your task will be to uncomment only those resources necessary for the current diagram.

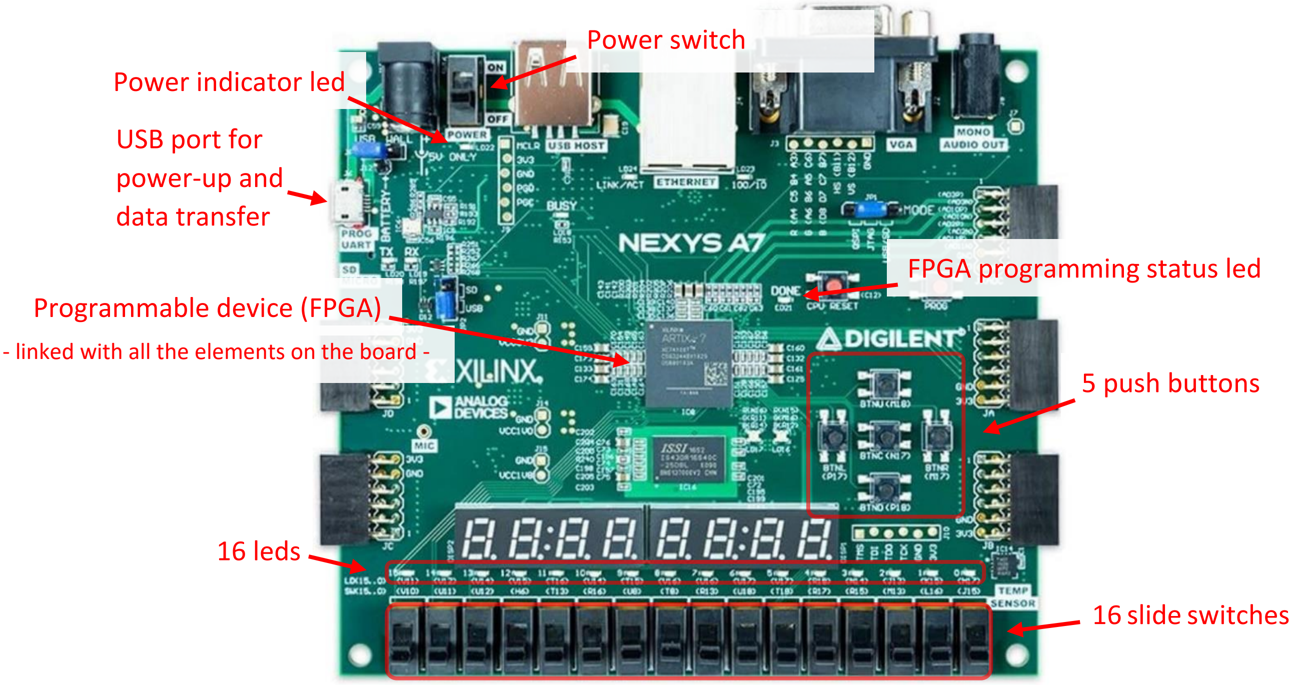
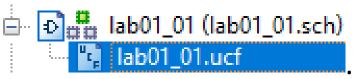

Power switch
Power indicator led
USB port for power-up and data transfer
FPGA programming status led
Programmable device (FPGA)
- linked with all the elements on the board -
5 push buttons
16 leds
16 slide switches
NEXYS A7
DIGILENT
XILINX
ANALOG DEVICES

Figure 1. 9 Testing board

Before assigning a constraints file, a copy of *_template.ucf* should be renamed as the diagram and with the .ucf extension. Hence, for diagram lab01_01 a copy of *_template.ucf* should be renamed to *lab01_01.ucf*. The assignment can be carried out if lab01_01 is the active diagram of the project (the top module). This is indicated by the symbol in the Design panel. Check this out and then press the Add Source . Select *lab01_01.ucf* from the disk and finish the process. The constraints file will appear under the diagram name in the project hierarchy 

lab01_01 (lab01_01.sch)
lab01_01.ucf

. Double-click the constraints file in the hierarchy to open for edit.

Inside the constraints file you will notice all available resources carrying predefined names in double quotes, immediately after keyword NET. The lines corresponding to leds, and switches, necessary for the current diagram, will be uncommented by removing the preceding # character. Also, their names will be adjusted to match the names of the terminals within the diagram. Considering that input terminals A, B, C will be associated to switches 0, 1 and 2 respectively, you will have to uncomment lines 13-15 and modify them as follows (changes in red squares):

```
## Switches

NET "A" LOC=J15 | IOSTANDARD=LVCMOS33 | CLOCK_DEDICATED_ROUTE=FALSE; # sw0

NET "B" LOC=L16 | IOSTANDARD=LVCMOS33 | CLOCK_DEDICATED_ROUTE=FALSE; # sw1

NET "C" LOC=M13 | IOSTANDARD=LVCMOS33 |CLOCK_DEDICATED_ROUTE=FALSE; # sw2
```

Note: The values of LOC attribute (LOC is short for location) are identical with the codes mentioned on the board near the switches. The situation is similar for push buttons and leds.

Considering F1, F2 and F3 will be connected to leds 0, 1 and 2, lines 40-42 will be uncommented and changed accordingly:

```
## LEDs

NET "F1" LOC=H17 | IOSTANDARD=LVCMOS33; # led0

NET "F2" LOC=K15 | IOSTANDARD=LVCMOS33; # led1

NET "F3" LOC=J13 | IOSTANDARD=LVCMOS33; # led2
```

Save the constraints file with *Ctrl+S* or File > Save from the menu.

1.4.4 Generating the programming file

The programming file can be generated from **Generate Programming File** at the bottom of Design panel (Figure 1. 10), by a right-click and choosing one of the commands available: Run, ReRun or Rerun All. Various messages will appear at the console. In case of errors, depending on their gravity, the entire process might stop. The issues reported must be solved accordingly, and the entire process relaunched with Rerun All.

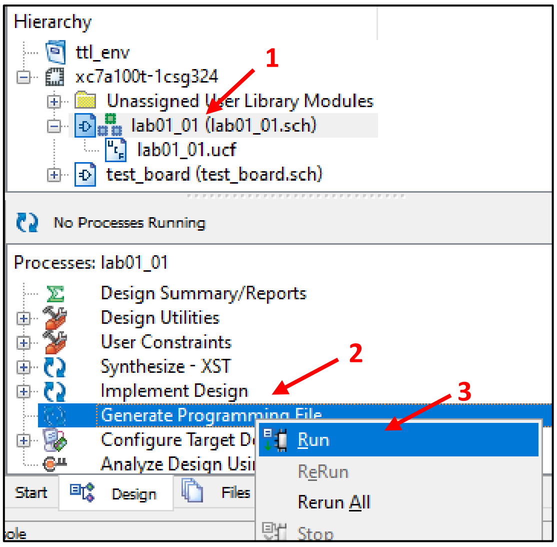


Figure 1. 10 Generating the programming file

If the programming file is successfully generated, the file lab01_01.bit will appear in the project folder along with a green icon: ✓ Generate Programming File.

1.4.5 Programming the FPGA and testing on board

Important: Before any programming session, the board (Figure 1. 9) must be connected to the working station using an USB cable and must be powered-up from the power switch. The power indicator led will light up.

You will launch the ISE iMPACT tool by right-clicking on **Configure Target Device** and selecting Run from the menu (Figure 1. 11).

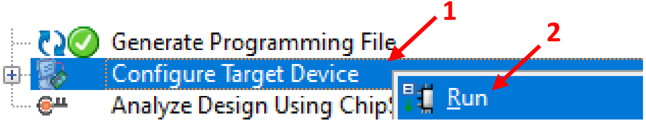


Figure 1. 11 Launching the ISE iMPACT tool

A new working session ca be created inside ISE iMPACT by double-clicking on Boundary Scan (Figure 1. 12).

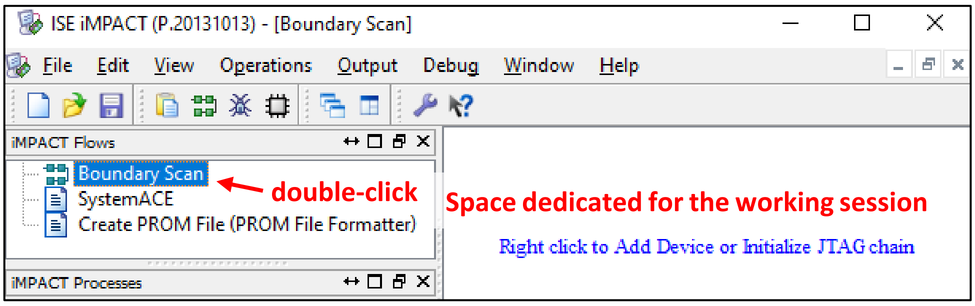
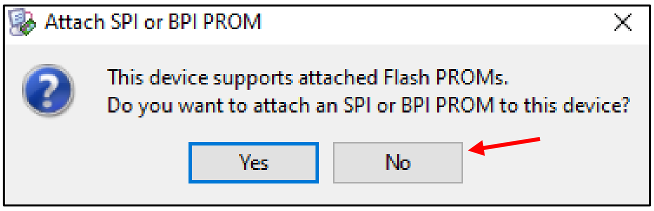

ISE iMPACT (P.20131013) - [Boundary Scan]
File
Edit
View
Operations
Output
Debug
Window
Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Formatter)
double-click
iMPACT Processes
Space dedicated for the working session
Right click to Add Device or Initialize JTAG chain

Figure 1. 12 Creating a working session in ISE iMPACT

Steps required to initialize the connection with the board:

1. Right-click in the working session and choose Initialize Chain.

2. Confirm the association with the programming file and select lab01_01.bit from the project folder.
3. **Important**: Always refuse the attachment of SPI or BPI PROM when asked.



Attach SPI or BPI PROM

This device supports attached Flash PROMs.
Do you want to attach an SPI or BPI PROM to this device?

4. Press OK in the next window showing programming settings.

A successful connection will highlight the FPGA symbol in the working space, its model, and the name of the current configuration file, as in Figure 1. 13.

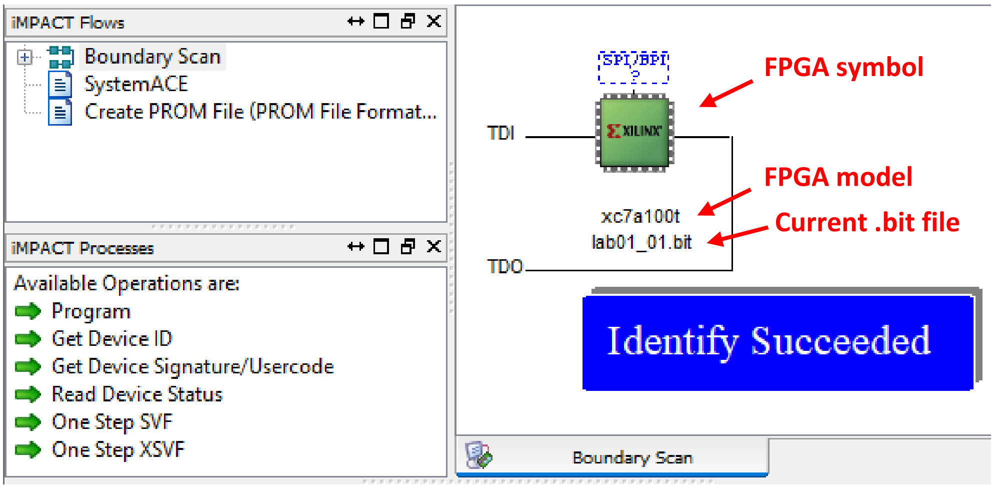


Figure 1. 13 ISE iMPACT has successfully connected to the board

The final step is the physical configuration of the FPGA board with the programming file. Right-click on the circuit symbol and select Program from the list (Figure 1. 14). Immediately after this step you will notice the programming status led lights up. From this moment the board behaves according to the circuitry on the diagram. You can use the switches to send 0 and 1 input values, and you will notice the results on the leds: F1=1, if A=B=1, F2=1, if A=B=C=1 and F3=C.

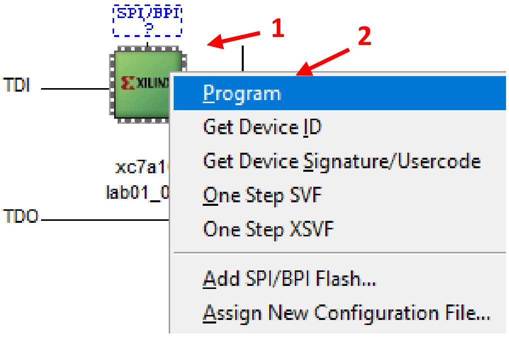


Figure 1. 14 Launch the programming session after attaching the .bit file

The current configuration is maintained for as long as the board is powered and will be lost when switching off. The ISE iMPACT session can be maintained even when new changes are made to the current diagram in Project Navigator. To save time, you should not close ISE iMPACT after each tested circuit. If you apply changes to the diagram and generate a new programming file with the same name, you may return to ISE iMPACT and Program the board without having to pass through all the steps required for connection.

If the name of the programming file changes, supposing you want to test another diagram, you should choose Assign New Configuration File… first (Figure 1. 14). You will have to select the new .bit file from the disk and then you can Program the board. Again, passing through all the steps required for connection is not necessary.

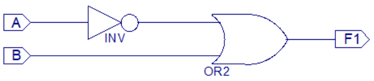
Note: When closing the ISE iMPACT (at the end of the class), you will be asked to save the current project. It is not recommended to save the current project in ISE iMPACT.

Important: Project *ttl_env* can be cleaned of files generated in time, by executing the script clean_win.bat in Windows or clean_lnx.sh in Linux. Both scripts can be found in the project folder. The scripts will keep the .sch diagrams and the .ucf files.

1.5 Assignments

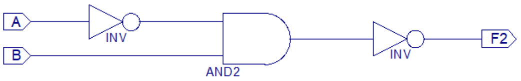
1. Add the following circuits to the *ttl_env* project (use a separate diagram for each circuit) and test them on the board using the truth tables. Name the diagrams and use switches and leds of your choice.

a) F1 = $\overline{\mathrm{A}} + \mathrm{B}$



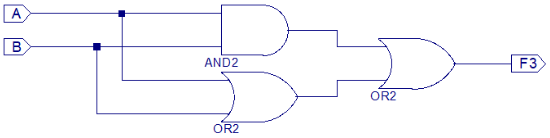
A	B	F1
0	0	1
0	1	1
1	0	0
1	1	1

b) F2 = $\overline{\overline{\mathrm{A}} \cdot \mathrm{B}}$



A	B	F2
0	0	1
0	1	0
1	0	1
1	1	1

c) F3 = $(\mathrm{A} \cdot \mathrm{B}) + (\mathrm{A} + \mathrm{B})$



A	B	F3
0	0	0
0	1	1
1	0	1
1	1	1

1.6 Bibliography

[1] Digilent, Nexys A7 Reference Manual, Available online: https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual

[2] AMD Xilinx, ISE WebPACK Design Software, Available online: https://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html

[3] Wikipedia, George Boole, Available online: https://en.wikipedia.org/wiki/George_Boole

[4] Wikipedia, Transistor–transistor logic, Available online: https://en.wikipedia.org/wiki/Transistor-transistor_logic