

# Lab 01: First look at Vivado

---

## Lab Goals

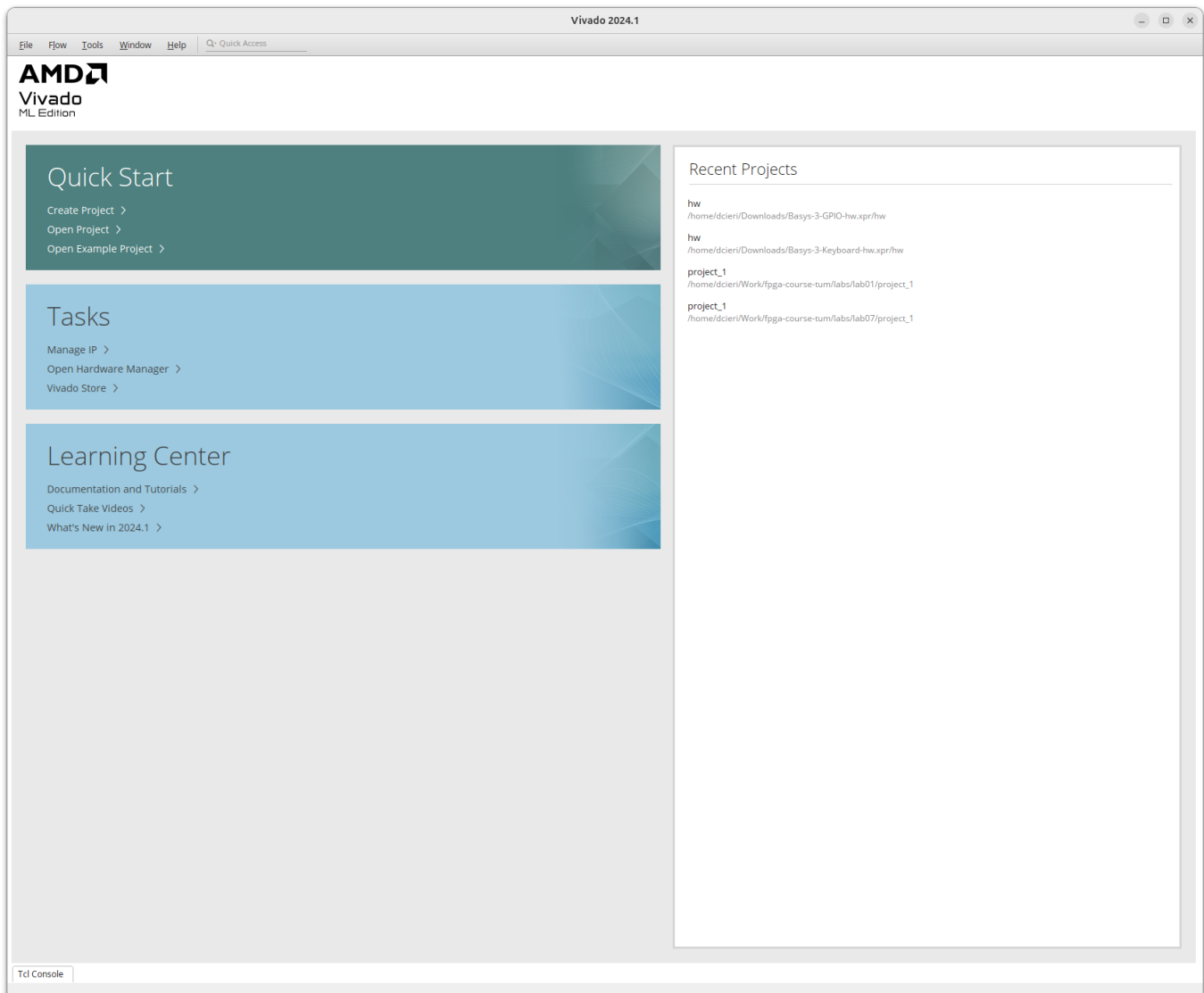
- Take a simple HDL design through synthesis, implementation and onto the development board, using AMD Vivado

## Starting Vivado and opening the project

Open a terminal window and launch vivado

```
vivado &
```

You will be then prompted with the launch screen.



Click on **Open Project**, and navigate to your home area, and open the `labs` directory.

Go inside the `lab01` folder, and open the Vivado project file `lab01.xpr`.

We are now in the Vivado design Environment

The left sidebar shows the **Flow Navigator**, which can be used to launch most of the flow steps. Some of the options are grayed out, since they require some steps in the flow to be run beforehand, e.g. you cannot open the synthesized design, before having run the synthesis.

The panel on the right shows the Project Summary. On the top, you can see the main settings of the project. Check for example, that the *Product Family* is `Artix-7` and that the *Project Part* is `xc7a35tcpg236-1`.

Above the pad, you will see the current context of the project, which is `PROJECT MANAGER - lab01`. The context affects the commands you can run, as we'll see later.

The left pad in the context shows the `Sources` window. Sources are subdivided in four categories.

- *Design Sources*: these are the files actually used by Vivado to synthesise the design
- *Constraints*: these are the files that maps your source file to the actual pins of the FPGA.
- *Simulation Sources*: files used to simulate the design
- *Utility Sources*: any other file in the project, not belonging to the above category. E.g. a python script.

Expand the *Design Sources* folder, if not yet done.

You should see a hierarchical view of modules in our project, using the following syntax.

```
<Module Name>(<Architecture Name>) (<Filename>) (<No. of submodules>)
```

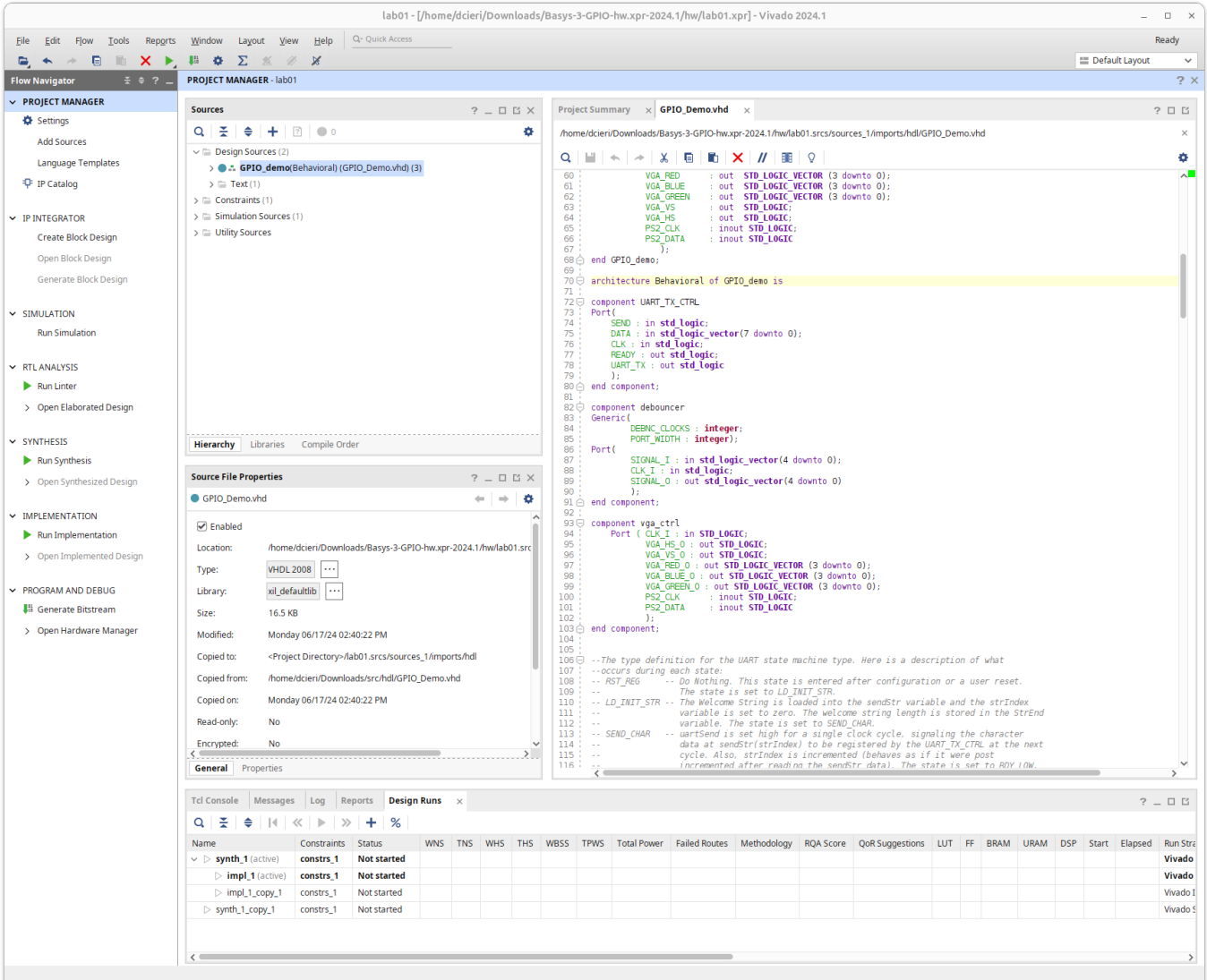
The top module of the design is highlighted in bold. In our case we have,

### **LED\_Counter (Behavioral) (LED\_Counter.vhd) (3)**

If you single click on this source file, you will see its properties pop up in the directly below panel (*Source File Properties*).

Some of this properties are just file information (E.g. the location), but other are important for the project, like the file type. For `LED_Counter.vhd`, we set a file type to `VHDL`, which is fine in this case. Please, note that Vivado refers to the 93 version of VHDL when saying `VHDL`. To use `VHDL 2008`, please change the file type.

Now double-click on `LED_Counter`. This will open the file in an editor window on the right.



The Vivado editor provides syntax highlighting and some other simple editor features.

## The Example Design (An LED Counter)

The design we are using for the first two labs is a simple 4-bit binary counter, that can be setup to to count up, down and be paused and reset.

The bits are then represented on the Basys3 board LD0-LD3

The functions are controlled by four buttons, that go high when pressed.

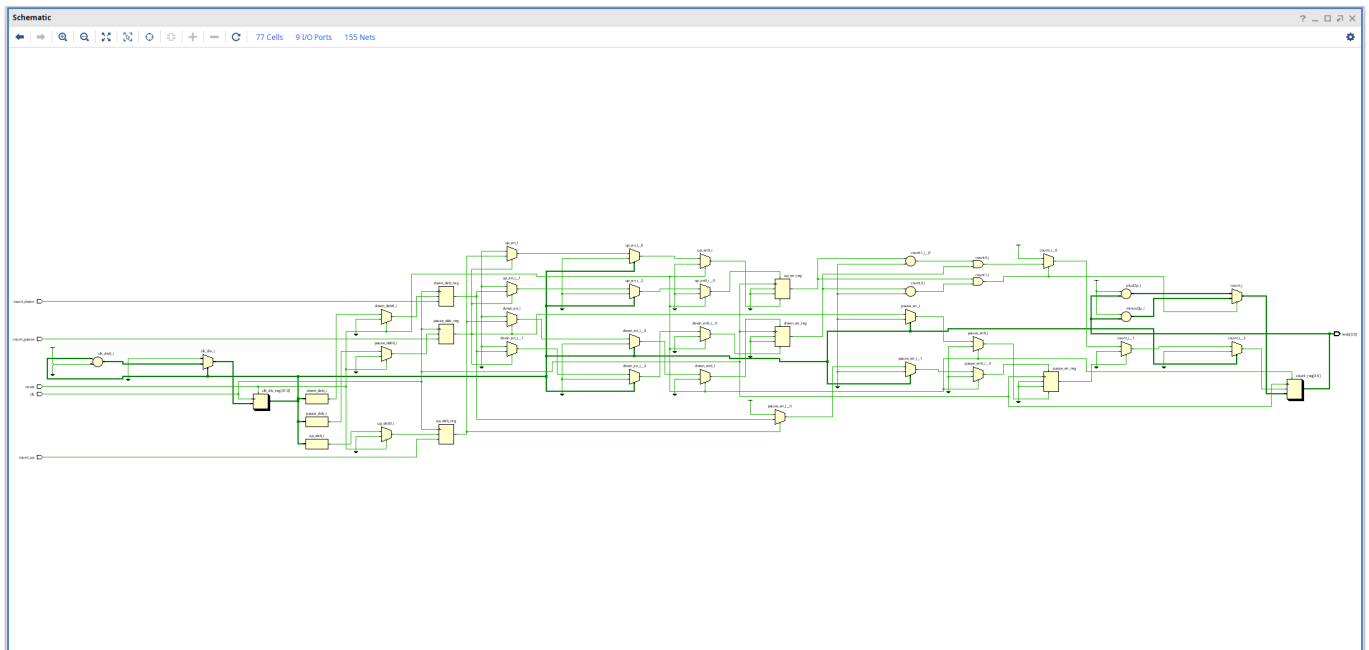
- BTC: Pause
- BTU: Up Counter
- BTD: Down Counter
- BTR: Reset Counter

There is a counting mechanism implemented that increases the LED counters every second, since using the system clock of the board (100 MHz) will be to fast for us to see any LED blinking.

## RTL Analysis

Vivado allows you to compile your code before actually starting the synthesis and implementation flow. On the left sidebar, click on *Open Elaborated Design* and click OK in the pop-up window.

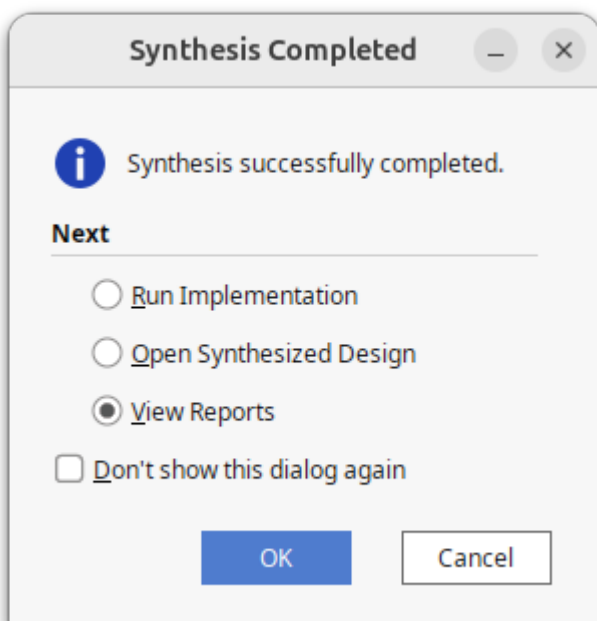
You can then click on *Schematic* on the sidebar. It should open a block diagram scheme of our design. This is useful a useful tool to check that our design is correctly interpreted by Vivado.



## Synthesis

Run now the synthesis, clicking on the **Run Synthesis** command on the right.

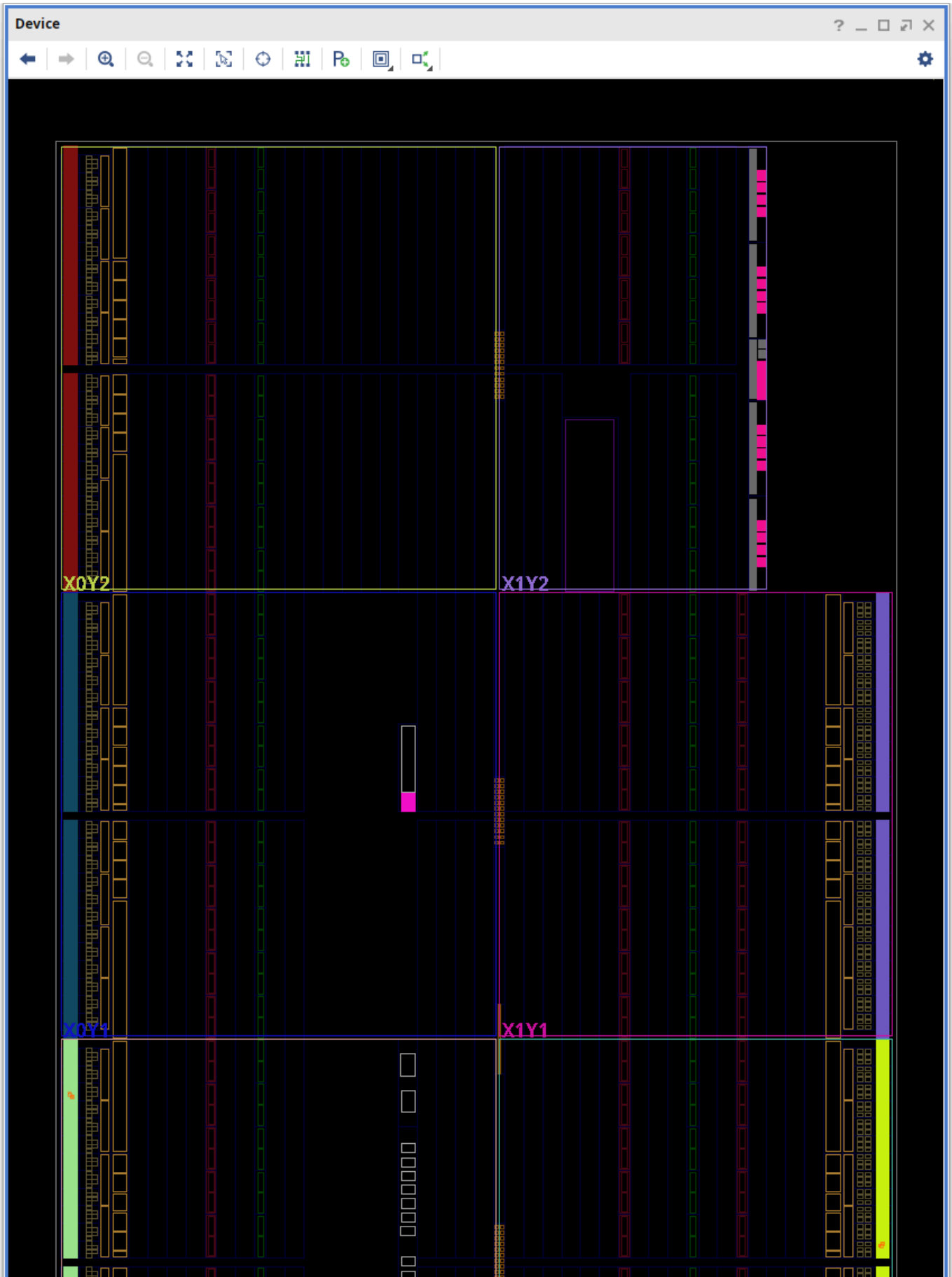
You can follow the synthesis flow, in the **Log** window on the bottom. Once the synthesis is completed, a pop-up window will appear.



Select **Open Synthesized Design** and click **OK**.

You are now in the **SYNTHESIZED DESIGN** Context as you can see from the top bar. Here you can see two new tabs appearing on the right pad.

- **Package**: This is the I/O planning where you can see the pins of the FPGA and its configurations
- **Device**: Here you can see the actual layout of the device.





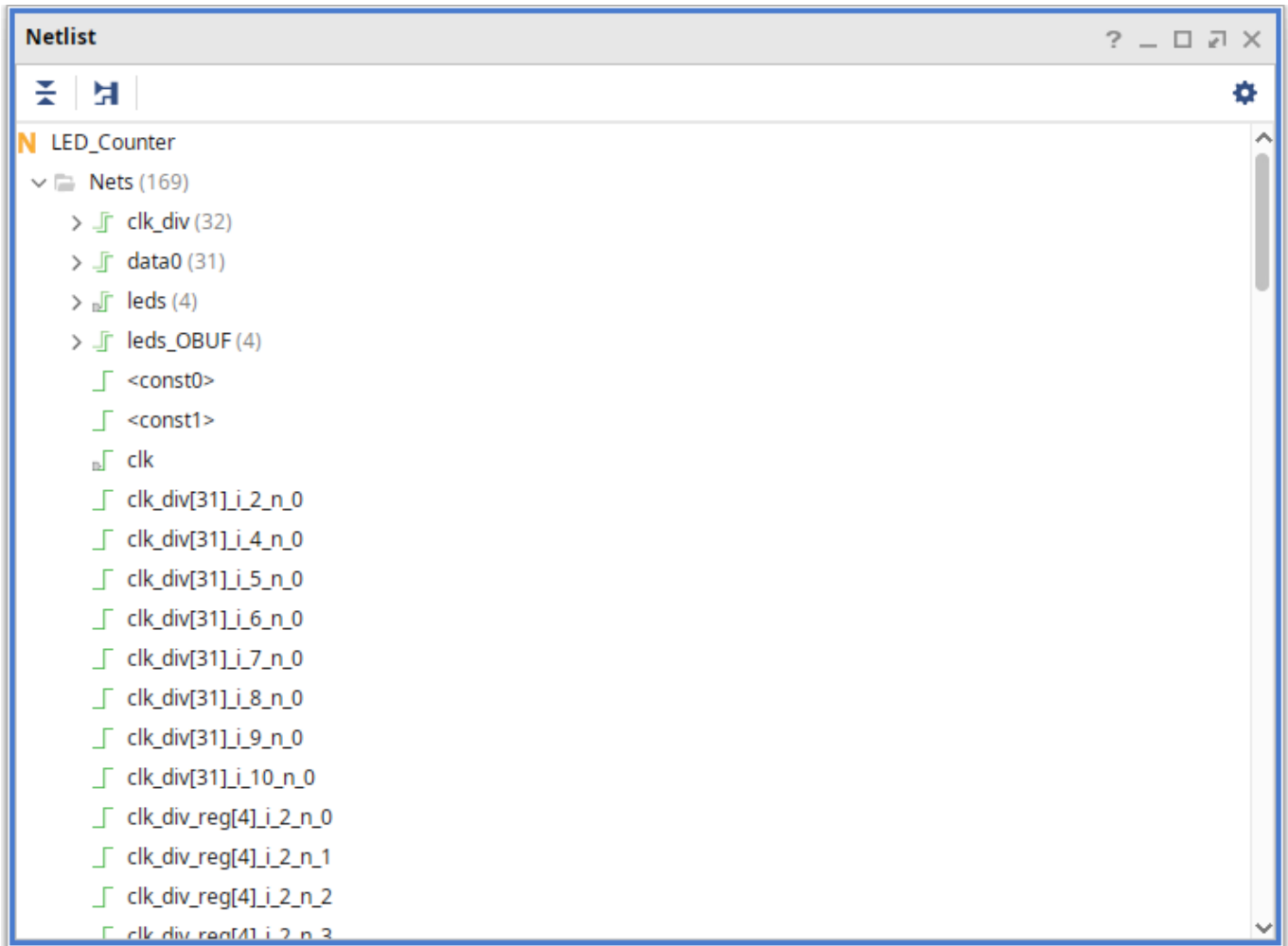
Let's have a look at the layout. The six blocks (X0Y0->X1Y2) are the six clock regions of the FPGA. The components on the FPGAs are painted in different colors.

- In blue, you have the logic slices.
- The green columns are DSP slices.
- Red are block RAMs.
- In orange are the clock resources.
- The colored blocks on the sides of the devices are the chip I/O.

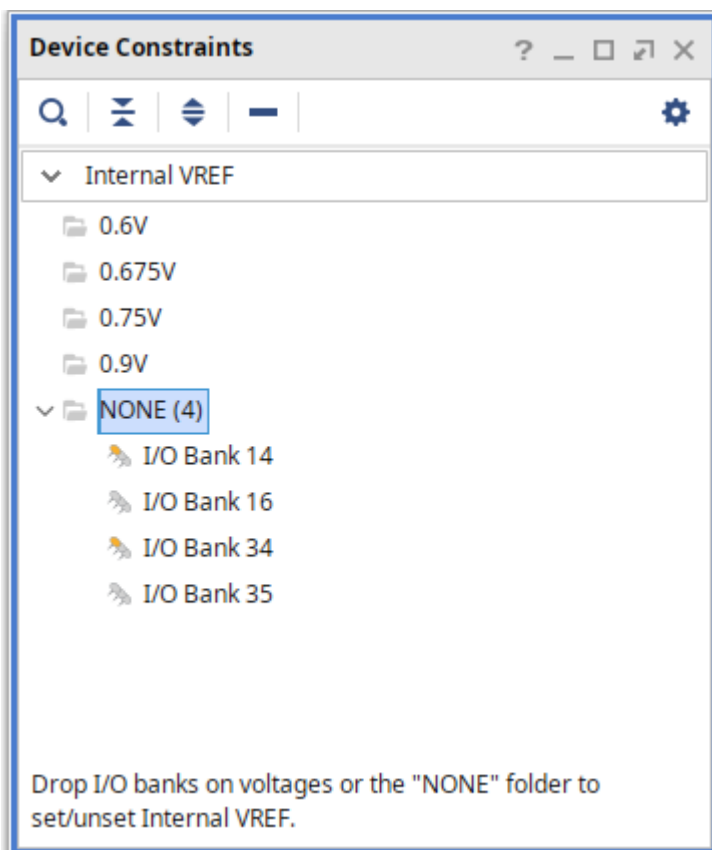
You can click on any area to see its properties (in the left box), and zoom in to select individual blocks.

There is now another pad in the window showing the [Netlist](#). This shows the nets and leaf cells in your design. You can expand each category and click on individual objects to see its property.

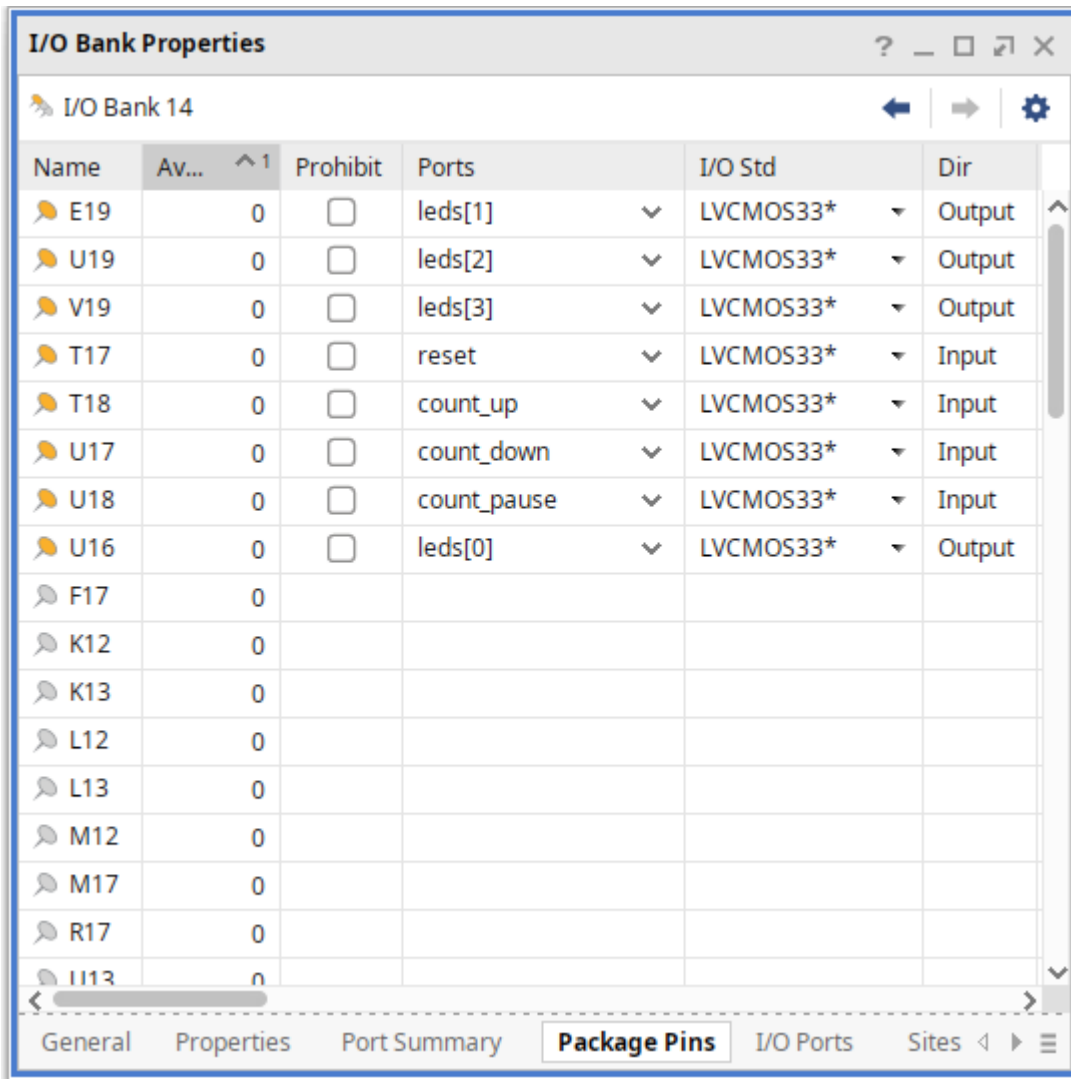
Clicking on the object, will also highlight it in the device view, if placed. Since implementation hasn't been run yet, no object has been placed.



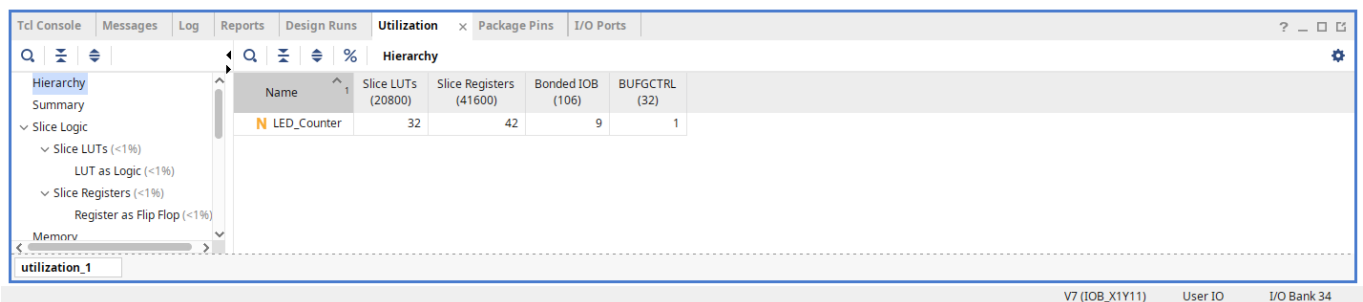
On the left panel, you can see the device constraints. Here, you can see which I/O bank on the device has constraints to the actual device pin.



If you click for example on **I/O Bank 14**, you should see the **I/O Bank Properties** window opening in the directly below pad. Here you should see the names of the port of your design, connected their relative pins. You can click on the **Float** button, on the top right of the window, to have a better look.



Click now on **Report Utilization** in the sidebar. This will create a estimate report of the resources that our design will use on the FPGA. You can also click on the percentage icon **%**, to have get a relative usage with respect to the available resources on the device.



Once you finished exploring, close the **Synthesized design** context, clicking on the **X** symbol on the blue bar. You should get a warning message. Just click OK to confirm.


## Implementation

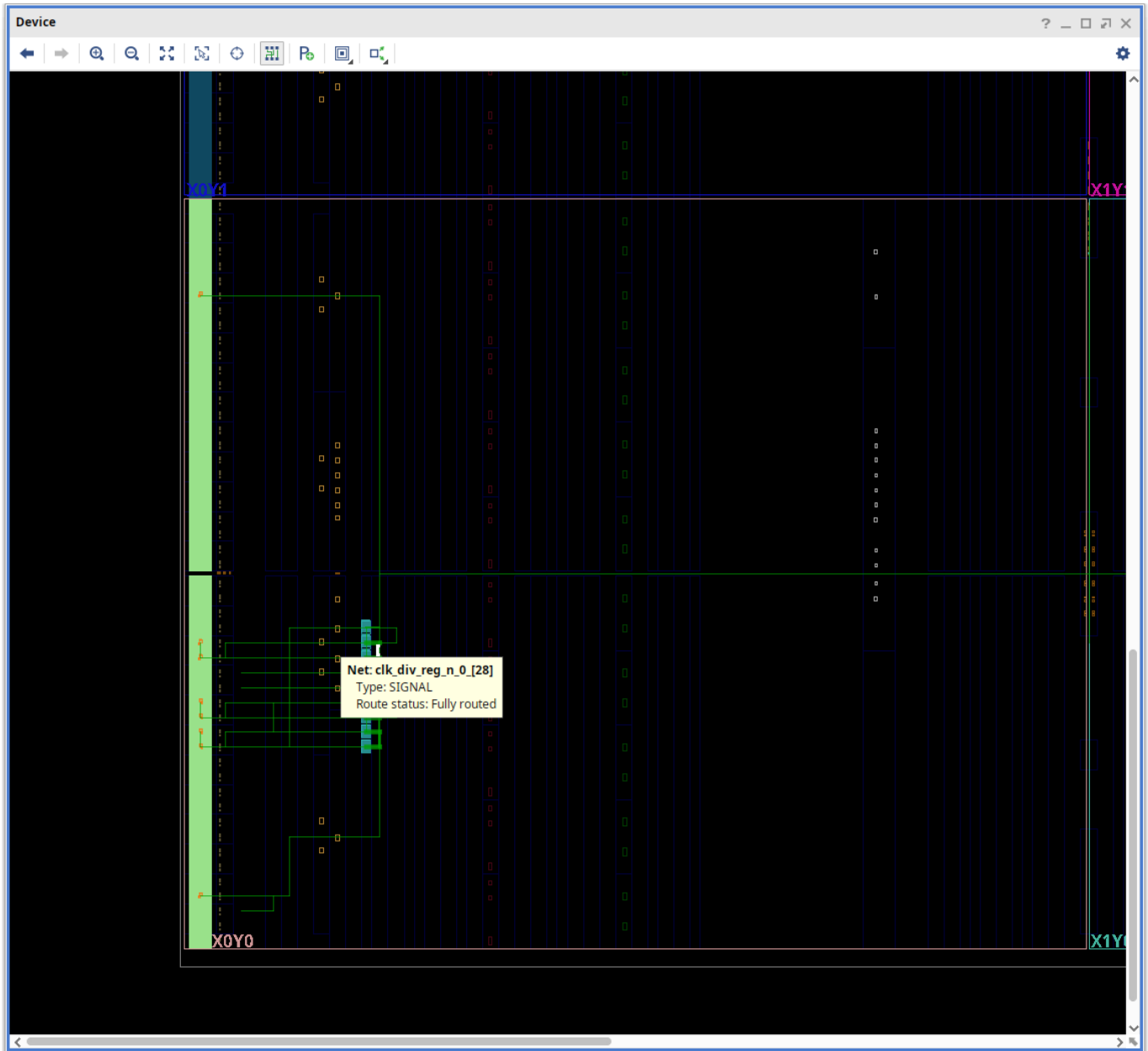


Run now the implementation by clicking on **Run Implementation** on the sidebar. Similarly to the synthesis, you can follow the progress in the log on the bottom panel.


Once it completes, you'll receive another pop-up message.

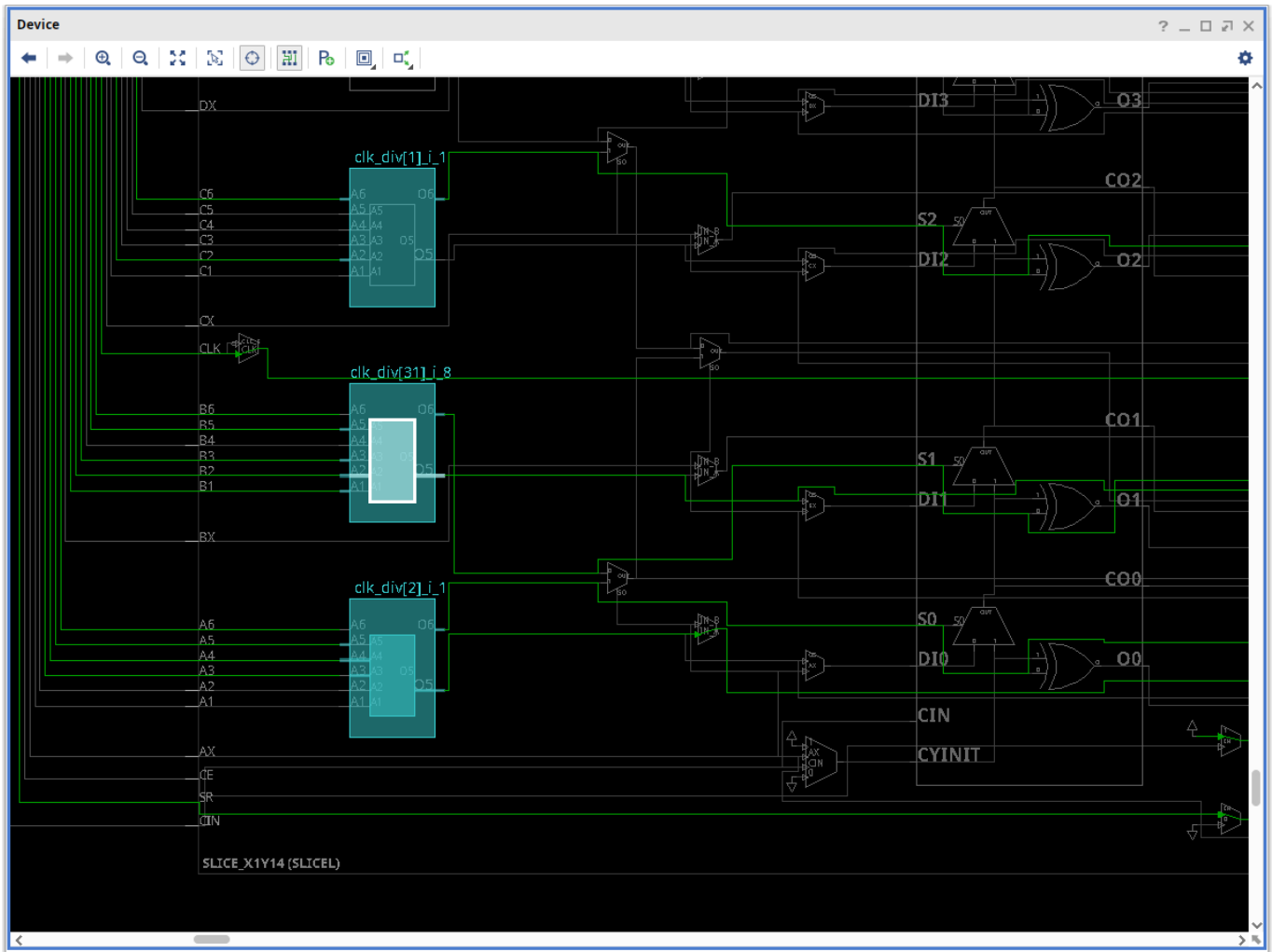
Open now the *Implemented Design*, and go to the **Device** tab.

To show the routing click on the **Routing Icon** .



Now you can select a leaf cell in the **Netlist** window, and you should see where it has been placed on the device.

Select the **Auto-fit selection** icon , and select a leaf cell in the netlist. You should now see the inside of the slice where the leaf cell has been stored.



In this view, we use the following color scheme (which is user-configurable)

- Grey: not used
- Cyan: Placed leaf cell
- Green: Placed net.

You can finally run the **Report Utilization**, to see what resources are actually used by the implemented design. Do you see differences with respect to the Report obtained in the synthesized design?

Close now the implemented Design context window.

## Bistream Generation

Generate the bitstream, by clicking on **Generate Bistream** on the left sidebar. Again, once it completes, you will see a pop-up window.

Select **Open Hardware Manager** and click OK.

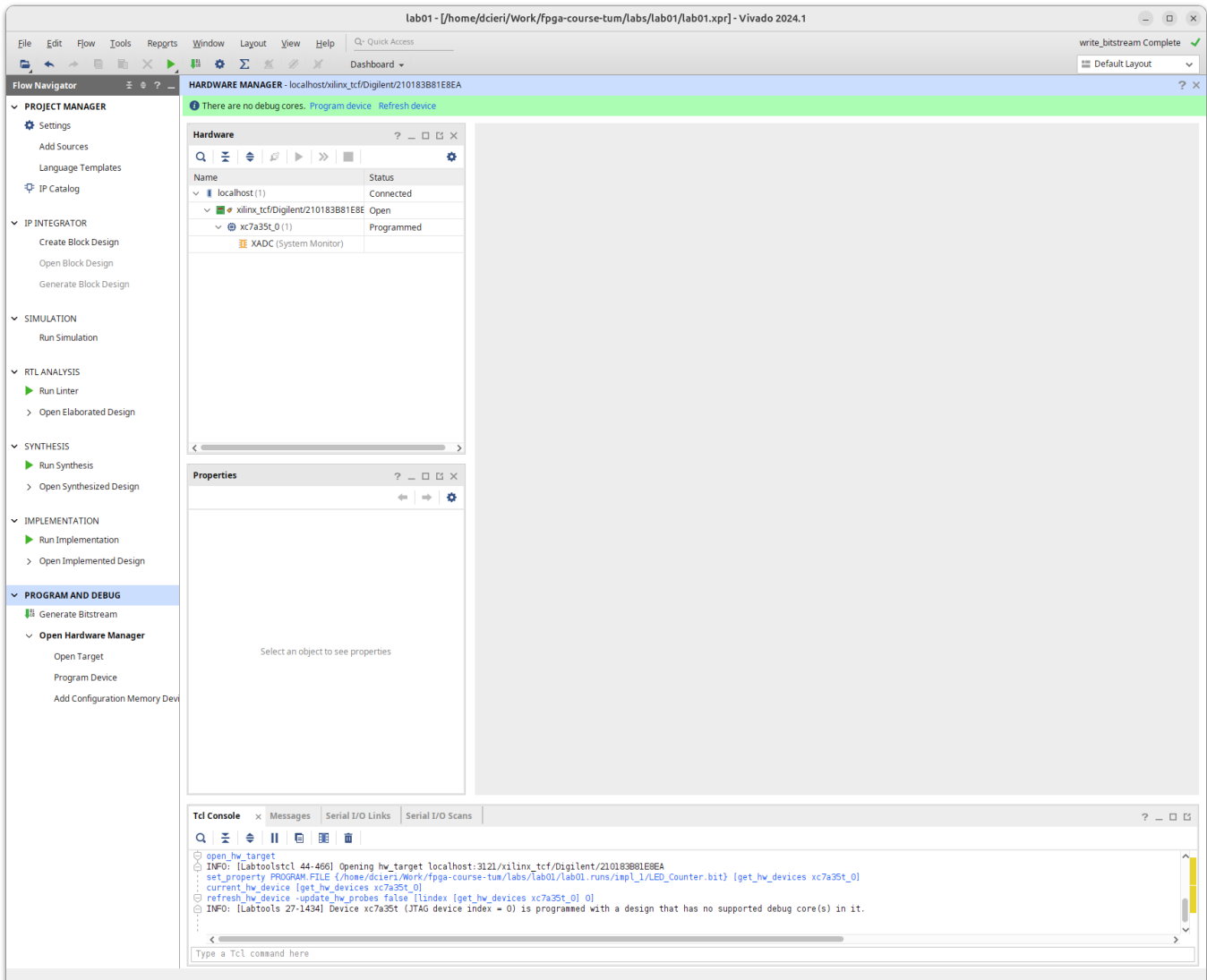
## Hardware Manager

The hardware manager allows us to program the FPGA on our Basys-3 board. First of all connect the board to your laptop, using the provided microUSB cable, if not yet done. Switch on the board, using the POWER switch on the top left corner of the board.

The board contains a default firmware which is loaded on the FPGA, that exercises most of the I/Os. We want now to load our design.

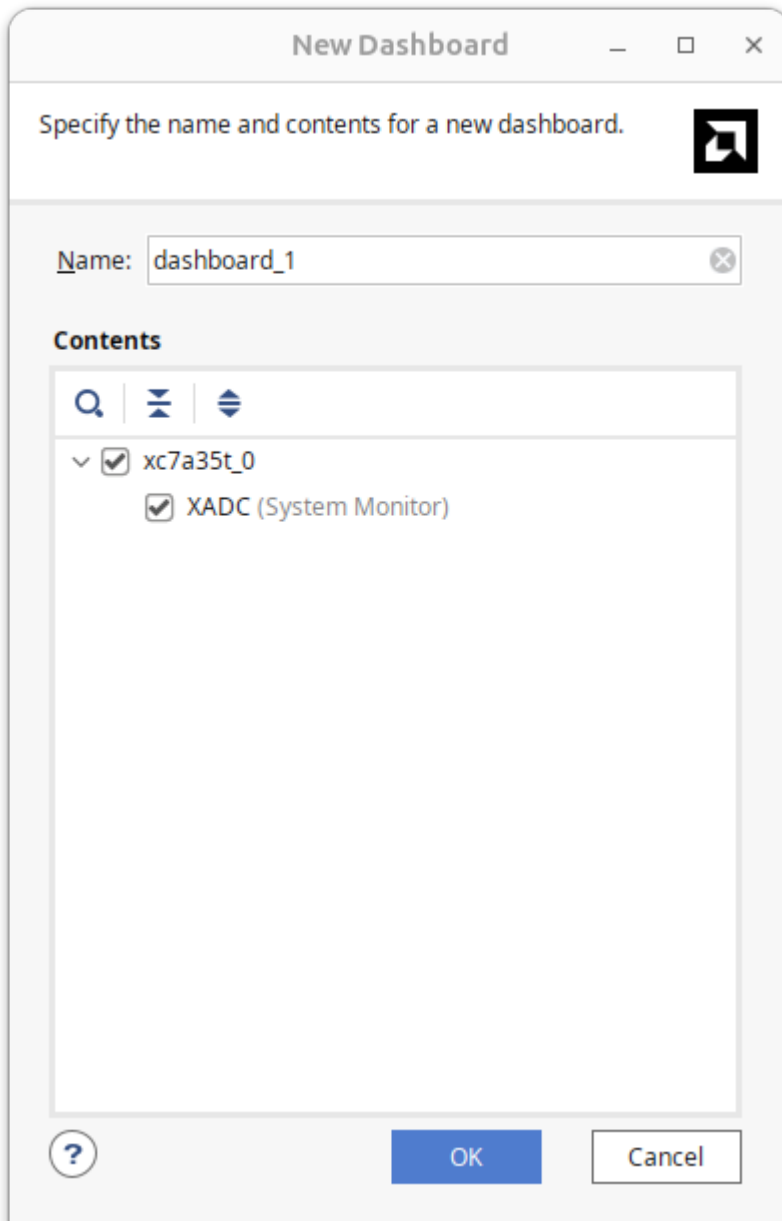
In the hardware manager, click on **Open Target** on the left sidebar. Since we have just one board connected, it is enough to click on auto-connect.

If everything went well you should see the board appearing in the **Hardware** panel.

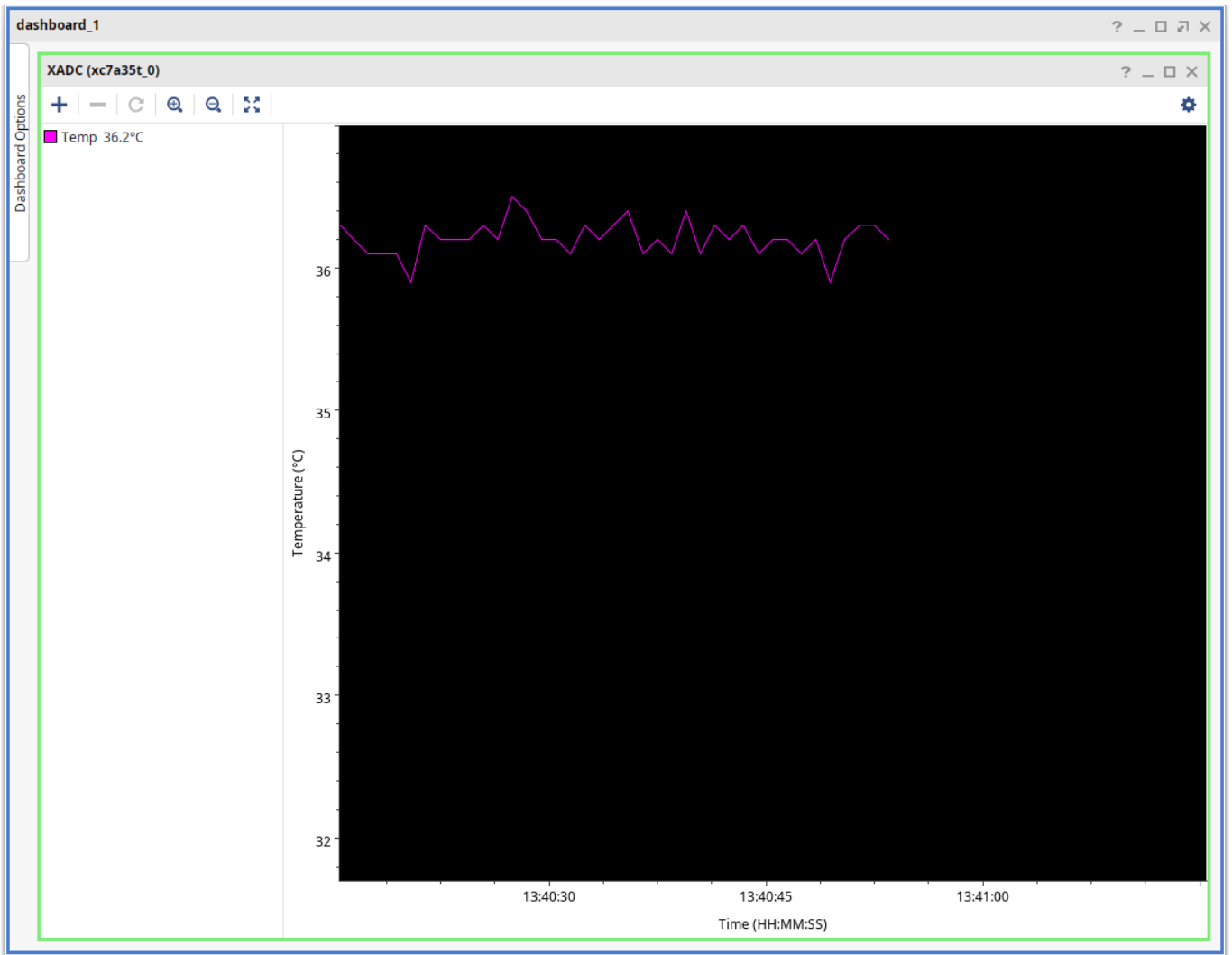


Here you can see already some information on the board, like its ID and the part number of the FPGA, which is the same as in our design.

The FPGA contains also an XADC system monitor, which monitors the temperature on the device. If you double-click on **XADC**, a window should pop-up asking to create a **New Dashboard**. Leave the default and click OK.



You should now see the temperature on the board as a function of time.



Let's program now our FPGA, clicking on **Program Device** on the sidebar. It should select automatically the bitstream we just generated. Keep the default values and click **Program**.

The 'Program Device' dialog box provides the following information and controls:

- Instructions:** Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.
- Bitstream file:** `ork/fpga-course-tum/labs/lab01/lab01.runs/impl_1/LED_Counter.bit`
- Debug probes file:** (Empty field)
- Enable end of startup check:**
- Buttons:** **Program** (highlighted in blue), **Cancel**

You should see now on the board Led 0 to 3 going up and down following the counter we implemented. You can push the buttons, to try the other feature of our design.