Practicum 1

[Home] [Back]

Contents

- Introduction
- Practicum aims
- Reference documentation
- Navigate to the practicum directory
- Explore board schematics and master XDCs
- Power supplies
- Check jumper settings
- Connect the board to your personal computer
- Check cable drivers installation
- Launch the Vivado Hardware Manager
- Interact with the board from the Hardware Manager
- Read the device DNA
- Monitor the on-chip temperature through XADC
- Implement a simple RTL design targeting the Arty board
- Program the FPGA
- Program the external Quad SPI Flash memory
- Further readings

Introduction

[Contents]

In this first practicum in the electronics lab you are going to explore the Digilent Arty A7 development board and to learn how to program its Xilinx Artix-7 FPGA with a simple Verilog RTL design using Xilinx Vivado.

> **IMPORTANT !**
>
> All boards available in the lab mount a **Xilinx Artix-7 A35T** FPGA device. The original board by Digilent was referred to as *Arty* while the new revision of the same board is now referred to as *Arty A7*, still using an Artix-7 A35T device. There are a few small differences in board schematics between the original version and the second version in terms of power management, but apart from this all other schematic functionalities are the same.

Practicum aims

[Contents]

This introductory practicum should exercise the following concepts:

- locate Digilent Arty A7 development board reference documentation
- understand board schematics
- locate main circuit components on the board
- power the board from host computer using a simple USB cable
- review the usage of fundamental lab instrumentation for measurements (DMM, digital oscilloscope)
- check cable drivers installation and jumper settings
- use the Vivado *Hardware Manager* to interact with the FPGA
- program the FPGA and the external Quad SPI Flash memory

Reference documentation

[Contents]

All reference documents are open and freely available on Digilent website:

- *Arty Reference Manual*
  https://reference.digilentinc.com/reference/programmable-logic/arty/reference-manual
  https://reference.digilentinc.com/reference/programmable-logic/arty-a7/reference-manual
- *Arty Programming Guide*
  https://reference.digilentinc.com/learn/programmable-logic/tutorials/arty-programming-guide/start
- *Board Schematics*
  https://reference.digilentinc.com/_media/reference/programmable-logic/arty/arty_sch.pdf
  https://reference.digilentinc.com/_media/reference/programmable-logic/arty-a7/arty_a7_sch.pdf

PDF copies of all above documents are also part of this practicum and are available in the `doc/arty/` directory. For faster access to PDF documents from the command line it would be recommended to include in the search path of your operating system also the executable of your preferred PDF viewer application.

As an example, Linux users should already have the `evince` executable available in the search path:

```
% evince doc/arty/arty_board_reference_manual.pdf &
```

Very likely Windows users have Adobe Acrobat Reader installed on their machine and can update the `PATH` environment variable in the `login.bat` script in order to include the `acroread.exe` executable in the search path:

```
% acroread doc/arty/arty_board_reference_manual.pdf
```

The Evince PDF viewer is also available for Windows systems and can be installed from the official website. If you prefer a non-administrator installation a `.zip` of the software is also available at:

http://personalpages.to.infn.it/~pacher/teaching/FPGA/software/windows/Evince.zip

Navigate to the practicum directory

[Contents]

As a first step, open a **terminal** window and change to the practicum directory:

```
% cd Desktop/lae/fpga/practicum/1_arty
```

List the content of the directory:

```
% ls -l
% ls -l .solutions
```

Explore board schematics and master XDCs

[Contents]

Before connecting the board to your personal computer explore all board schematics. Try to recognize on the PCB all schematic components. Open the proper PDF file according to the board you are working with:

- `doc/arty/arty_board_schematics.pdf` for the original *Arty* board
- `doc/arty/arty_a7_board_schematics.pdf` for the new revision of the board referred to as *Arty A7*

With you preferred text editor application open also the main **Xilinx Design Constraints (XDC) file** provided by Digilent and available in the `.solutions/` directory:

```
% cp .solutions/arty_all.xdc .
```

```
% gedit arty_all.xdc &   (for Linux users)
% n++ arty_all.xdc       (for Windows users)
```

Get familiar with most important programmable I/O in the XDCs and locate physical resources on the board.

Please, be aware that the on-board **USB-to-JTAG circuitry** has been left **UNDOCUMENTED** by Digilent! You can easily recognize the main **FTDI chip** near the micro-USB connector indeed, but circuit details are not available.

> **QUESTION**
>
> A component on the PCB generates a 100 MHz clock signal. Which is the name of this component ? Where is placed on the PCB ?
>
> _____

Power supplies

[Contents]

Try to understand all possible **powering schemes** foreseen for the board. With a **digital multimeter (DMM)** perform some basic **continuity tests** to verify that different same-potential test-points on the board are shorted together, e.g. **GND** or **VCC**.

> **QUESTION**
>
> The **J8** connector is a 6-pins through-hole (TH) header that provides test points to probe **JTAG signals**. Where are placed **GND** and **VCC** on this connector ? Where are **TDI**, **TMS**, **TCK** and **TDO** signals ?
>
> _____

Check jumper settings

[Contents]

A few **jumpers** are available on the board and are used to **hard-program** some main functionalities of the board:

- external vs. USB power mode on J13 (legacy *Arty* board)
- **MODE** on JP1
- **CK_RESET** on JP2

Verify that all jumpers are properly inserted.

> **QUESTION**
>
> Which is the purpose of the **MODE** jumper ? What changes if this jumper is left unplaced ?
>
> _____

Connect the board to your personal computer

[Contents]

Despite the board can be powered from external power supply, for this course we will simply power the board using 5V from USB cable. Connect the board to the USB port of your personal computer using a **USB A to micro USB cable**. Verify that the **POWER** LED turns on.

Use the DMM to perform basic power checks. Repeat the measurement using the **oscilloscope**.

> **IMPORTANT !**
>
> Before using an "unknown" oscilloscope **always** verify that **BNC probes** are properly **compensated** by connecting the probes to the built-in oscilloscope square-wave generator. In case probes are over-compensated or under-compensated use a small screwdriver and operate on the probe trimmer.
>
> If you fill completely lost at this point ask to the teacher or ref. to:
>
> *https://www.electronics-notes.com/articles/test-methods/oscilloscope/scope-probe-compensation.php*

Check cable drivers installation

[Contents]

> **IMPORTANT !**
>
> If you are running Vivado from a **virtual machine** be sure that USB devices are properly forwarded to the virtual machine! As an example, if you use VirtualBox go through **Devices > USB** to make the Digilent board "visible" to the virtualized operating system.

Xilinx FPGAs are programmed using the **JTAG protocol**. In the past a dedicated (and expensive) **programming cable**, namely *Xilinx USB Platform Cable*, was required to program FPGA boards from a host computer. This dedicated cable (still in use for particular applications) **connects to a host computer USB port** (in the past to the "old style" serial port instead) and converts USB data into JTAG data.

For easier programming, the majority of new modern FPGA boards equipped with a Xilinx device provides an **on-board dedicated circuitry** that **converts USB to JTAG without the need of a dedicated cable**. That is, you can easily program your board by using a simple **USB Type A/Type B** or **USB Type A/micro USB** cable connected between the host computer and the board without the need of a dedicated programming cable.

On Digilent Arty/ Arty A7 boards this conversion is perfoemed by an integrated circuit by **FTDI (Future Technology Devices International)**. As already mentioned this circuitry has been left undocumented, but you can easily recognize the FTDI chip on the board close to the micro-USB connector.

In order to make the board visible to the host computer the operating system has to **properly recognize the on-board USB/JTAG hardware** requiring a specific **driver**. The **Xilinx USB/Digilent driver** is responsible for this.

By default the *Install Cable Drivers* option is already selected in the Xilinx Vivado installation wizard, thus at the end of the Vivado installation process cable drivers **should be automatically installed for you** on the system (this is the reason for which admin privileges are required to install the software).

Follow below instructions to check proper cable drivers installation.

LINUX

On Linux systems devices that are connected through USB can be listed using the `lsusb` command:

```
% lsusb
...
...
... Future Technology Devices International, Ltd FT2232C/D/H Dual UART/FIFO IC
```

Verify that a device from FTDI has been properly recognized by the system. In case `lsusb` is not installed, use:

```
% sudo apt-get install usbutils
```

```
% sudo yum install usbutils
```

according to the Linux distribution you are working with.

Beside `lsudb` you can use `dmesg` to query **kernel messages** (whatever happens on your PC can be traced using `dmesg`):

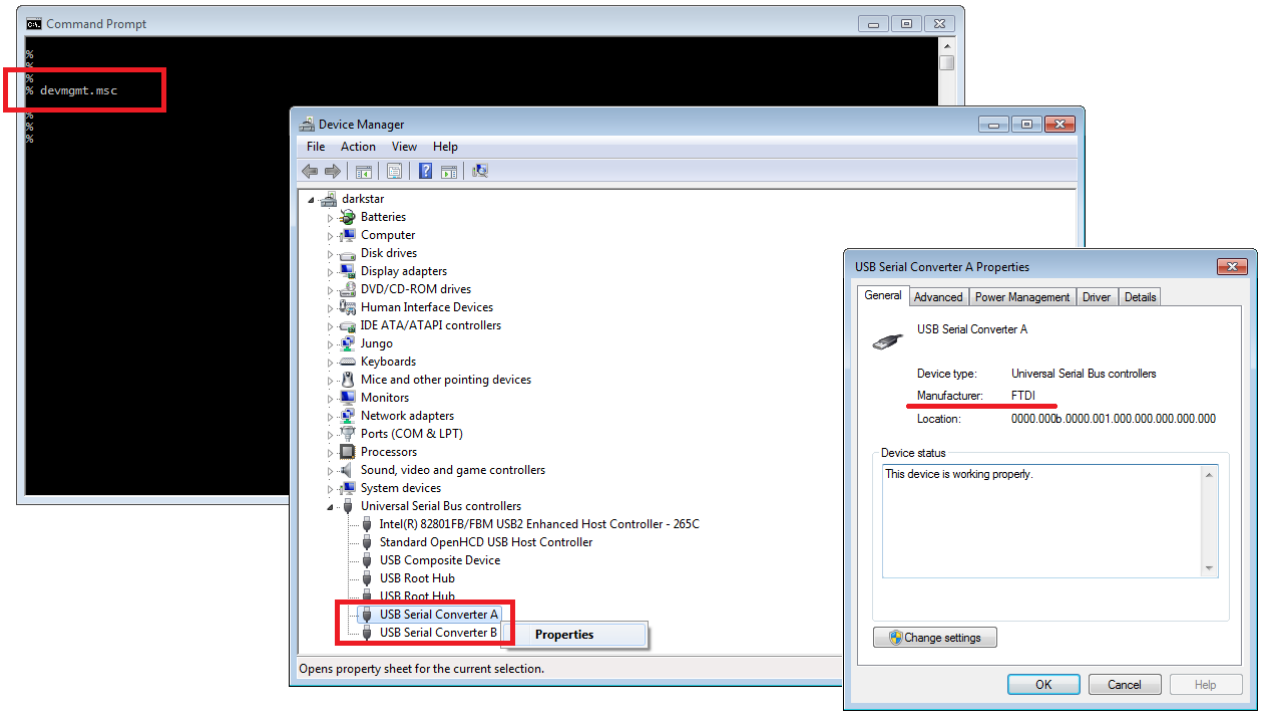
```
% dmesg
...
...
usb 1-1.3: FTDI USB Serial Device converter now attached to ttyUSB1
usbcore: registered new interface driver ftdi_sio
ftdi_sio: v1.5.0:USB FTDI Serial Converters Driver
```

WINDOWS

By default on Windows systems there are no command-line utilities to query hardware devices, therefore you have to open the *Device Manager* graphical interface. You can do this from the *Control Panel* or typing `devmgmt.msc` in the *Command Prompt* as follows:

```
% devmgmt.msc
```

If cable drivers are properly installed you should see in the list of USB devices two new devices named by Windows as *USB Serial Converter A/B*. Right-click on one of these devices to access the *Properties*. Verify in the *General* TAB of the device properties that the FTDI chip has been properly recognized. You can also inspect which **driver** is used by the operating system.



Launch the Vivado Hardware Manager

[Contents]

As part of the Xilinx Vivado design suite comes the so called **Hardware Manager**. The *Hardware Manager* is the tool used to **program** Xilinx FPGA devices after bitstream generation. Beside programming, the *Hardware Manager* can be also used to **debug** a design after firmware installation.

To launch the *Hardware Manager* you have to first start Vivado from the command line.

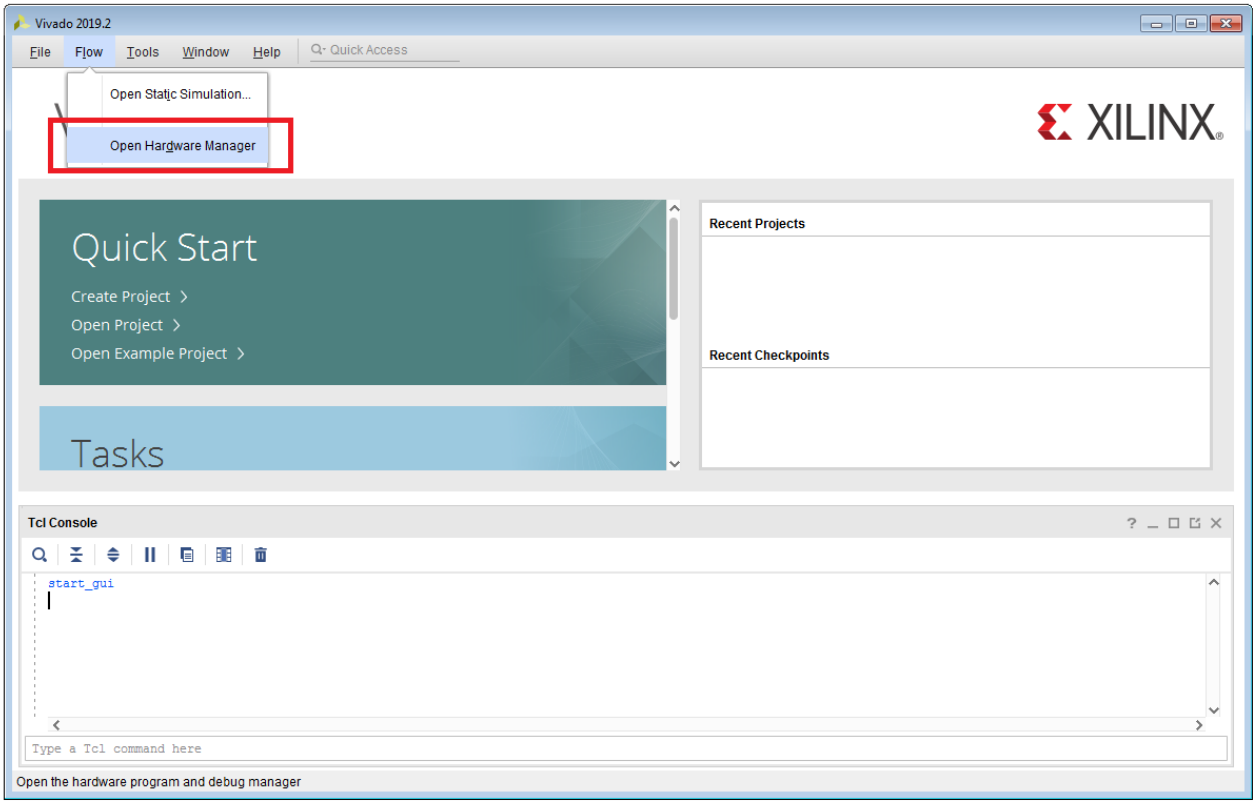
For Linux users:

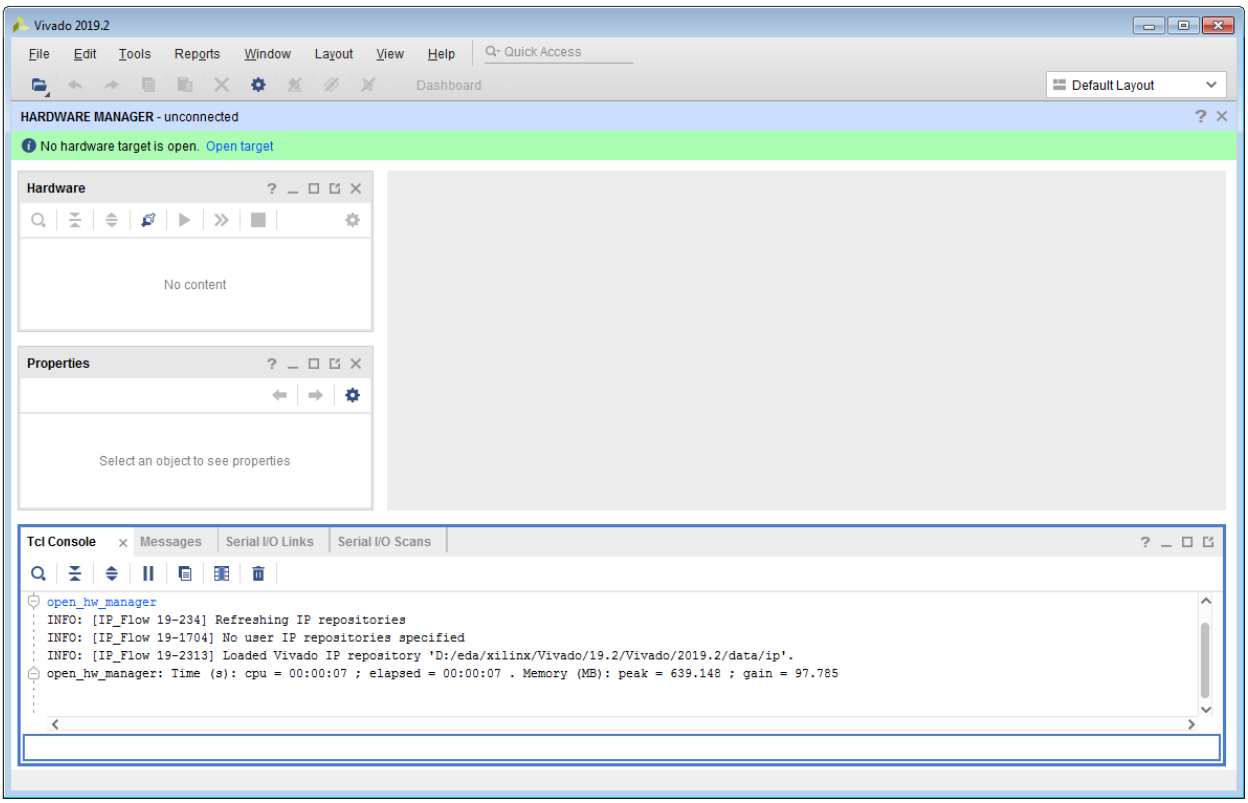
```
% vivado -mode gui &
```

For Windows users:

```
% echo "exec vivado -mode gui &" | tclsh -norc
```

You can then launch the *Hardware Manager* from the main Vivado GUI using **Flow > Open Harware Manager**. Alternatively you can type `open_hw_manager` in the Tcl console.

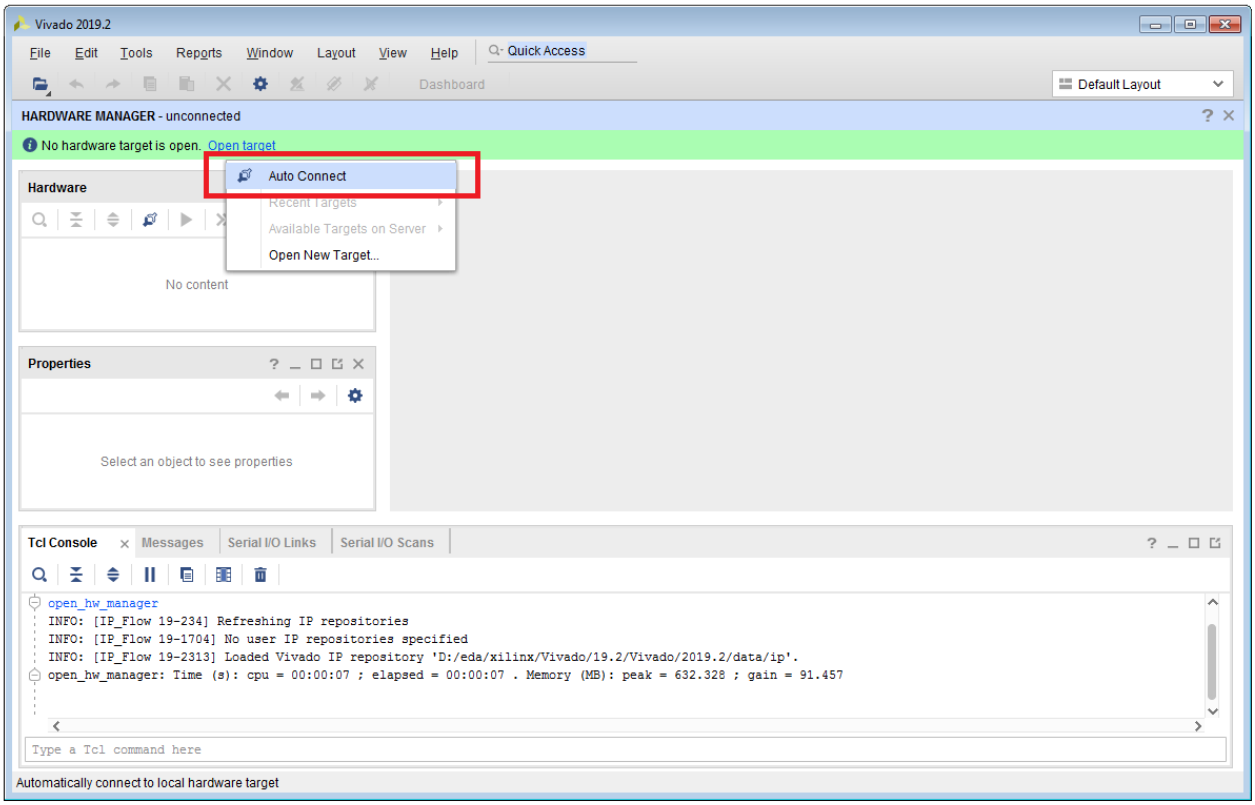


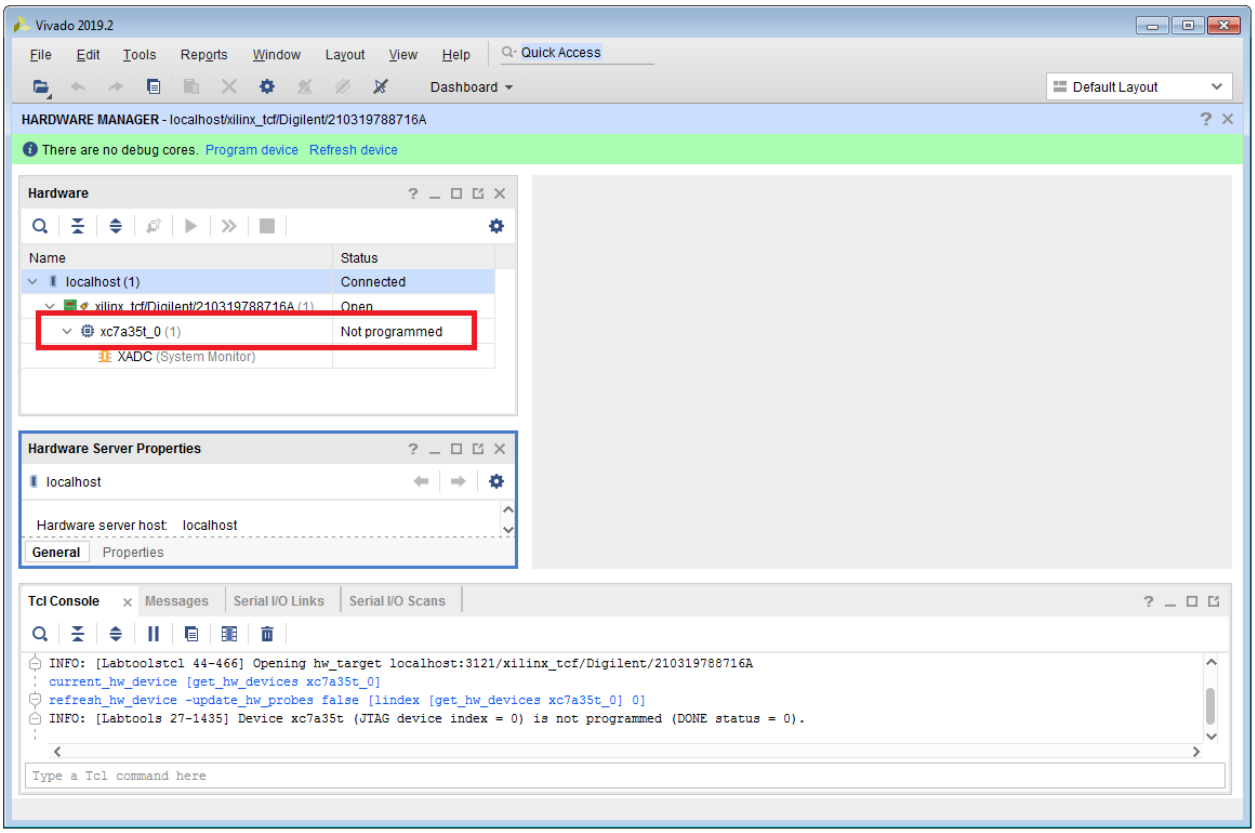


Interact with the board from the Hardware Manager

[Contents]

Try to establish a connection between the *Hardware Manager* and the Artix-7 FPGA device. To do this, simply left-click on **Open target > Auto Connect**.

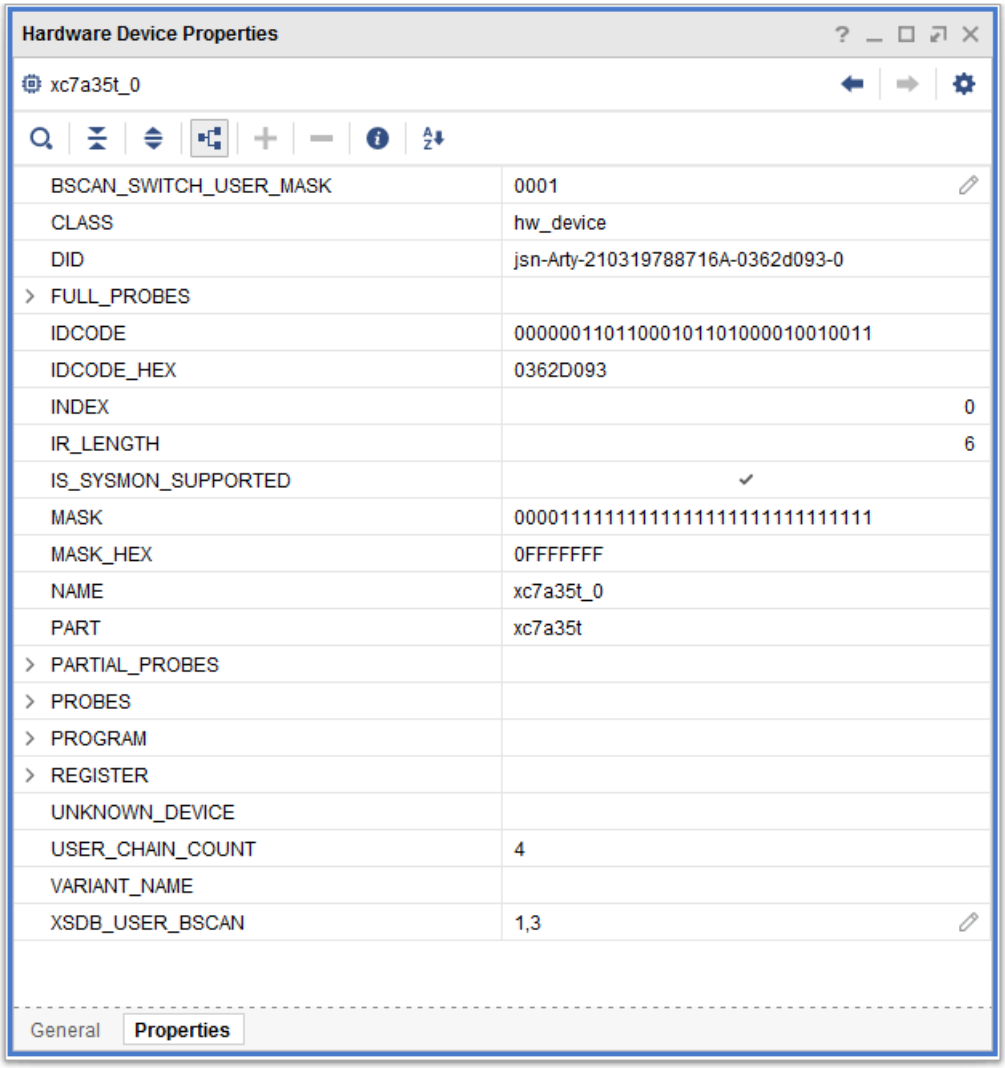




Observe the sequence of Tcl commands traced for you in the Tcl console:

```
connect_hw_server -allow_non_jtag
open_hw_target
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
```

If cable drivers are properly installed the *Hardware Manager* automatically recognizes the Artix-7 A35T on the board as `xc7a35t_0`.
Left-click on `xc7a35t_0` to select the device, then left-click on *Properties* to display the *Hardware Device Properties* form:



You can immediately detect if the FPGA is programmed by looking at the `DONE` status bit in the JTAG instruction register. Verify if the FPGA is already programmed using:

```
puts [get_property REGISTER.IR.BIT5_DONE [current_hw_device]]
```

Try to understand the meaning of the following Tcl commands:

```
puts [current_hw_server]
puts [current_hw_target]
puts [current_hw_device]
```

Read the device DNA

[Contents]

Each Xilinx FPGA has a unique **hard-coded device identifier**, also referred to as **device DNA**. This device DNA is a unique binary word "fused" by Xilinx after manufacturing. The identifier is nonvolatile, permanently programmed by Xilinx into the FPGA, and is unchangeable.

You can easily get the device DNA from the value of the `FUSE_DNA` register, either from the *Hardware Device Properties* form or using Tcl to query the property.

Type the following command in the Tcl console and observe the output:

```
get_property REGISTER.EFUSE.FUSE_DNA [current_hw_device]
```

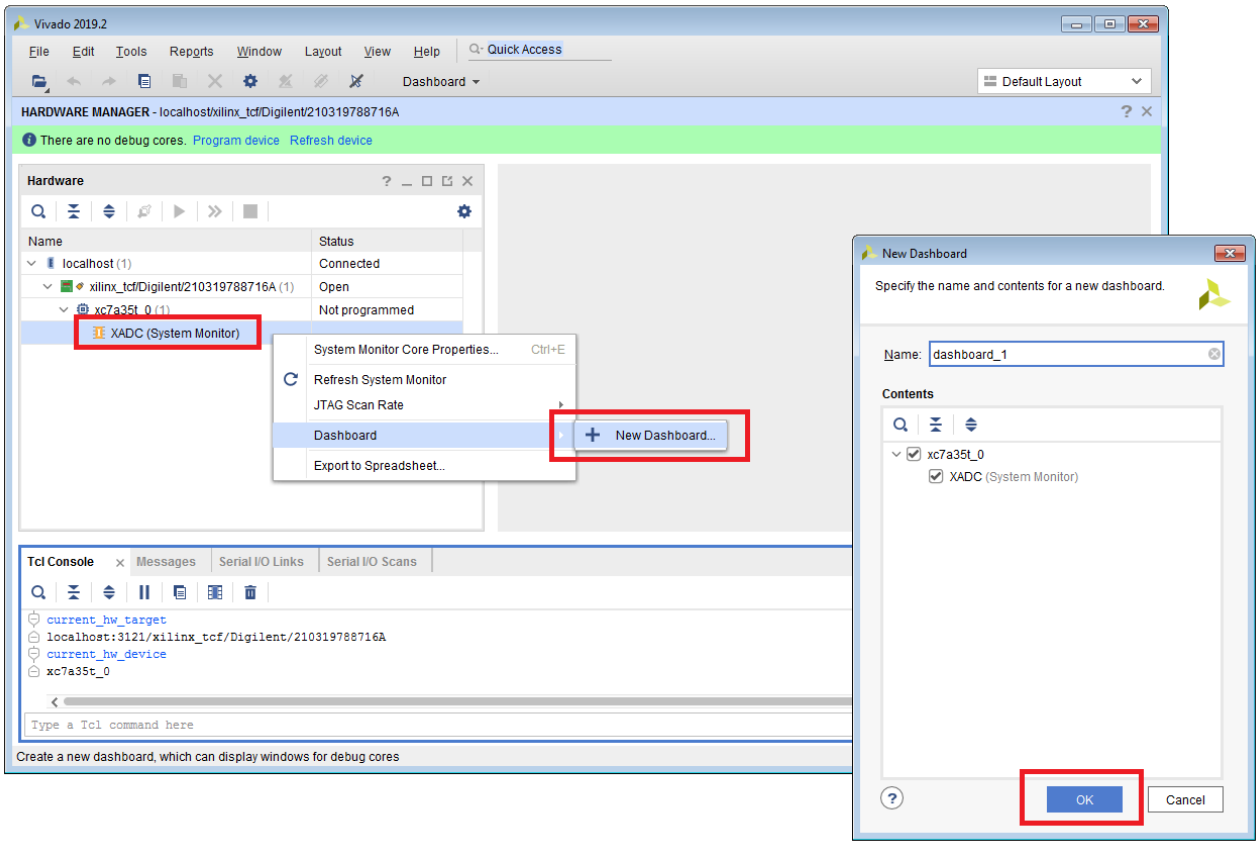
> **QUESTION**
>
> Which is the value of the device DNA of the FPGA connected to your computer ? How many bits are used for the device DNA ?
>
> _____

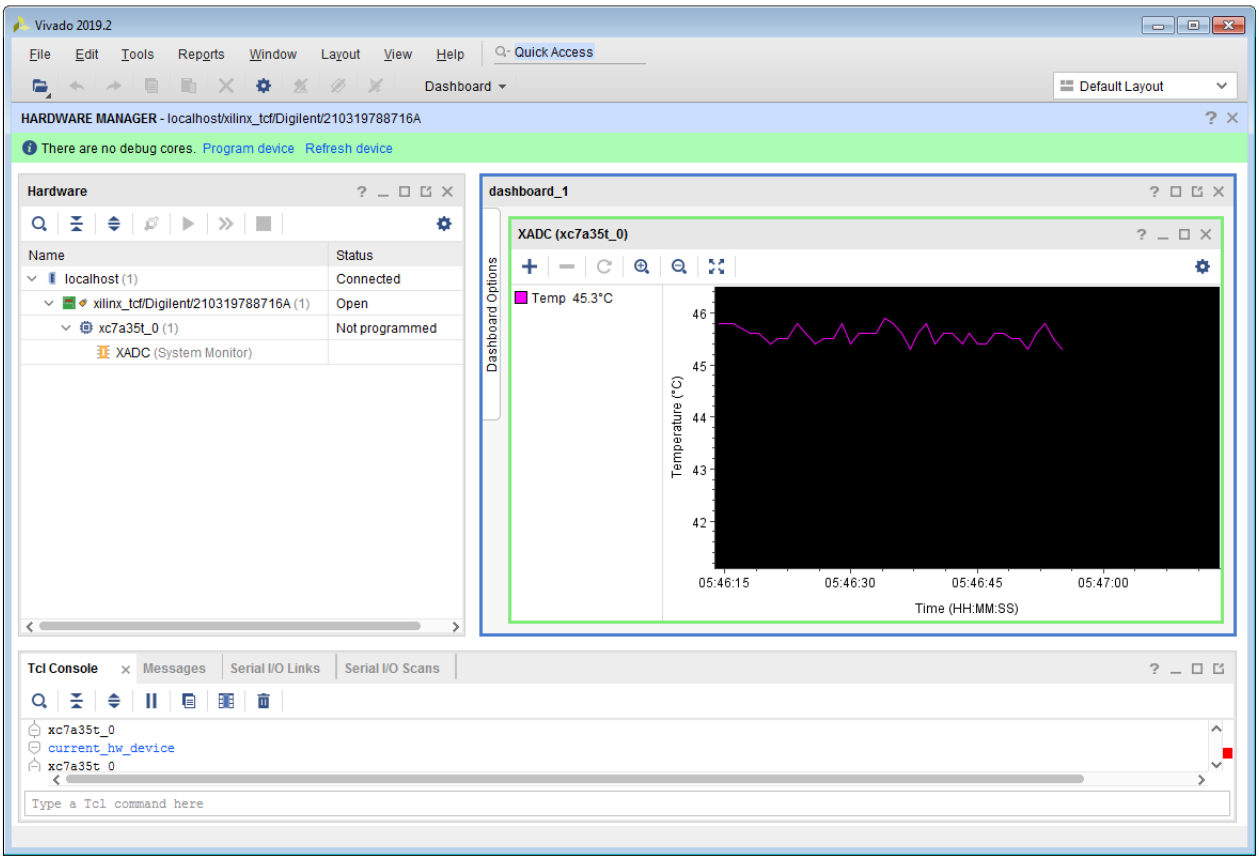
Monitor the on-chip temperature through XADC

[Contents]

All 7-Series and SoC Xilinx FPGAs have an embedded 1-MS/s 12-bit Analog to Digital Converter (ADC), referred to as **XADC**. The XADC can be compiled and used in the RTL code as an IP core, but is also available from the *Harware Manager* for monitoring purposes.

Right-click on XADC and select **Dashboard > New Dashboard**. In the *New Dashboard* window you can specify a name for the new plotter, or leave the default value. Left click on **OK** to create the new dashboard.





By default the XADC monitors the on-chip temperature, but you can also add other measurements, just right-click somewhere below `Temp` and select **Add Sensor(s)**.

The data exchange between the FPGA and the computer uses the same JTAG protocol used for device programming. Observe at the oscilloscope JTAG signals **TCK**, **TMS**, **TDI** and **TDO** using through-hole test points on **J8**.

> **QUESTION**
>
> Which is the default frequency of the JTAG clock **TCK** ?
>
> _____

Compare your measurement at the oscilloscope with the output of the following Tcl commnd:

```
get_property PARAM.FREQUENCY [current_hw_target]
```

Reduce the JTAG clock to 5 MHz with the following Tcl command:

```
set_property PARAM.FREQUENCY 5000000 [current_hw_target]
```

Verify the result at the oscilloscope.

Close the *Hardware Manager* and exit from Vivado once happy:

```
close_hw_manager
exit
```

Implement a simple RTL design targeting the Arty board

[Contents]

Map on the board the simple NOT-gate (inverter) already discussed in the first lab. To save time, copy from the `.solutions/` directory the `Inverter.v` source file:

```
% cp .solutions/Inverter.v
```

Create also a new `Inverter.xdc` file with your **text-editor** application:

```
% gedit Inverter.xdc &   (for Linux users)

% n++ Inverter.xdc       (for Windows users)
```

Try to **write yourself design constraints** required to implement the design on real hardware. As an example, map the inverter input to some slide-switch or a push-button, while the inverter output on a simple LED. Use the main `arty_all.xdc` as a reference for the syntax.

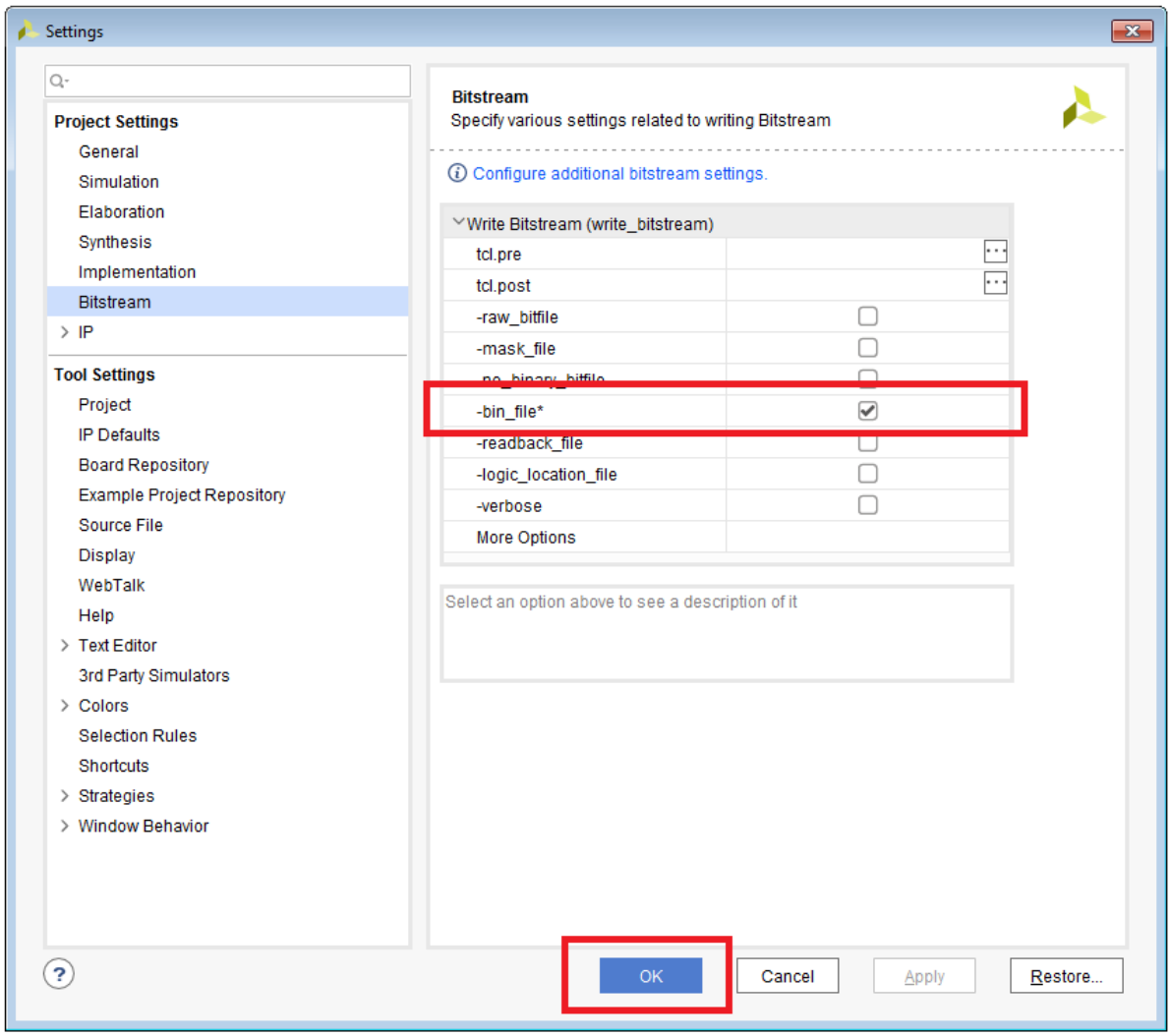
Once ready, open Vivado in graphical mode and try to run the FPGA implementation flow in *Project Mode* up to bitstream generation.

If you run out of time you can also run the *Project Mode* flow using a Tcl script. Copy from the `.solutions/` directory the sample `project.tcl` script:

```
% cp .solutions/project.tcl
```

You can then `source` the script from the Vivado Tcl console:

```
source project.tcl
```

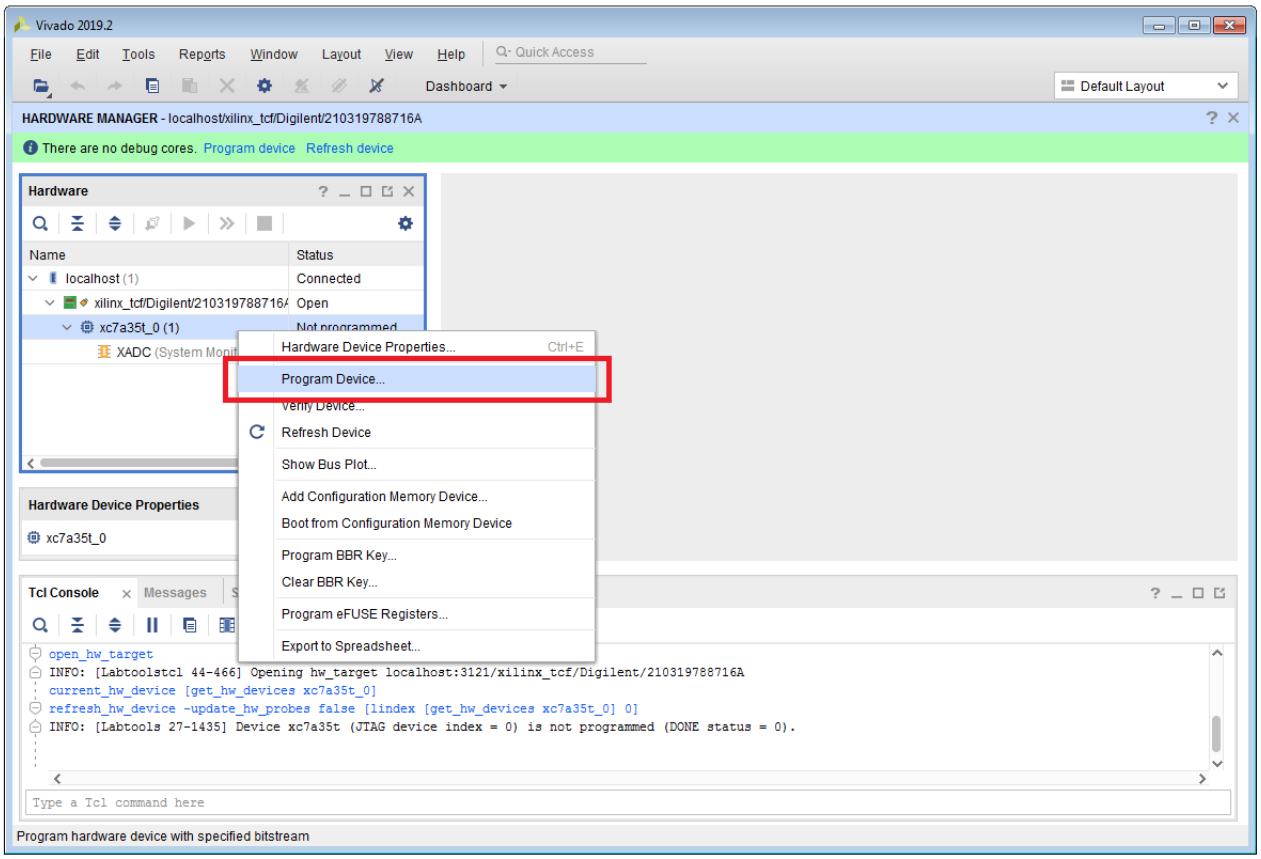
> **IMPORTANT !**
>
> In order to be able to program also the external Quad SPI Flash memory you must generate the **raw binary version** (`.bin`) of the bitstream file (`.bin`). To do this when working in Project Mode right-click on **Generate Bitstream** in the **Flow Navigator** and select **Bitstream settings**, then check the `-bin_file` in the table:
>
> 
>
>
> Alternatively use the following Tcl command:
>
> ```
> set_property STEPS.WRITE_BITSTREAM.ARGS.BIN_FILE true [get_runs impl_1]
> ```
>
> before running the bitstream generation flow.

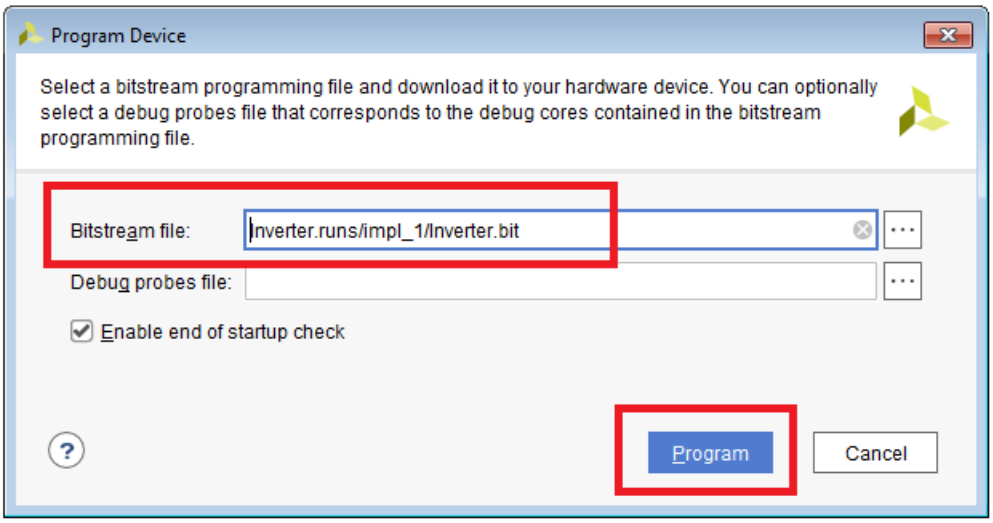
Program the FPGA

[Contents]

Open the *Hardware Manager* and establish a new connection through USB/JTAG ad discussed before.

To program the FPGA simply right-click on the `xc7a35t_0` device and select **Program Device**, then specify the **bitstream file** `Inverter.bit` generated by Vivado and placed in the `Inverter.runs/impl_1/` directory. Finally, left-click on **Program**.





Observe the **DONE** LED turning-off and then turning-on once the programming sequence has completed. Debug your firmware on the board.

You can also **observe the bitstream** at the oscilloscope through JTAG. To do this, probe the **TDI** signal at the oscilloscope on **J8** and then re-program the FPGA.

Review all Tcl programming commands traced in the Tcl console.

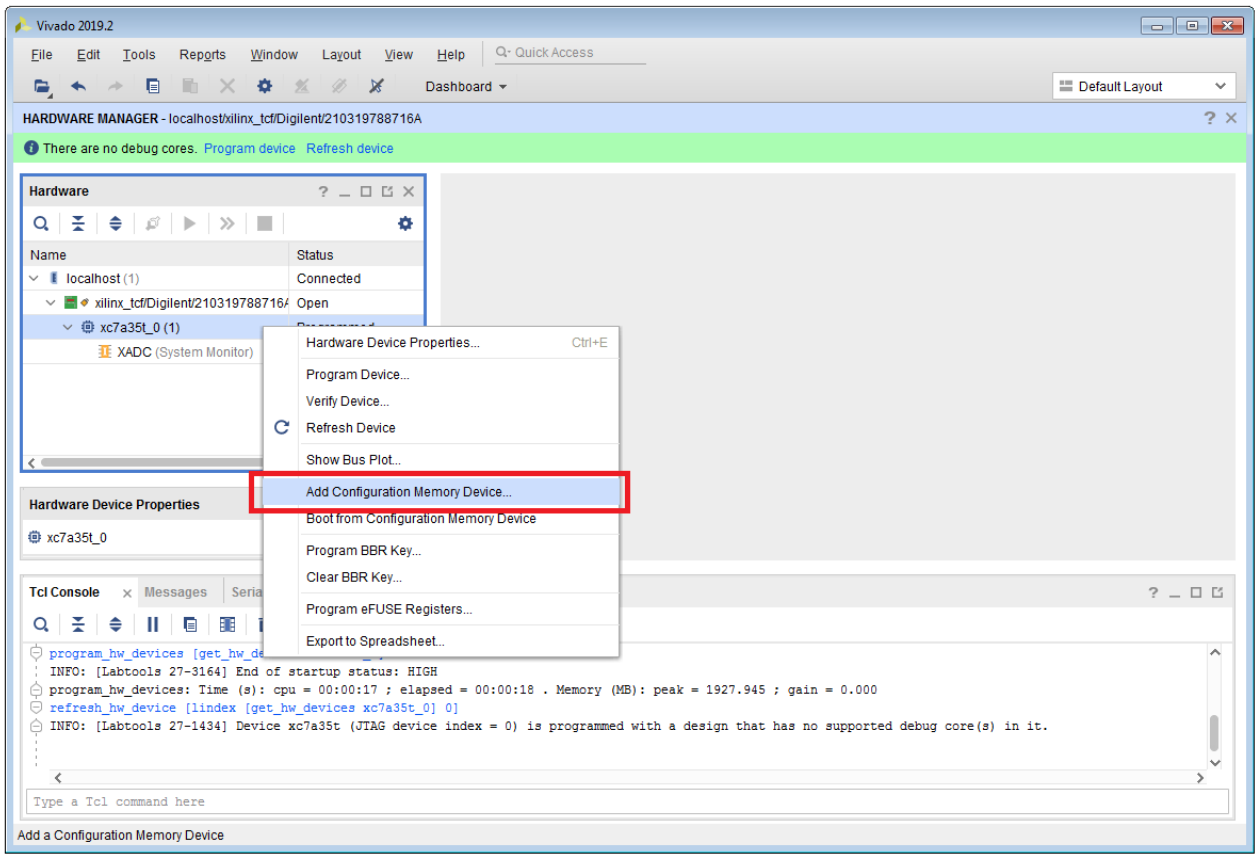
Program the external Quad SPI Flash memory

[Contents]

The firmware loaded into the FPGA is stored into a volatile RAM inside the chip. By default the FPGA configuration is therefore **non-persistent** across power cycles and you have to **re-program the FPGA** whenever you **disconnect the power** from the board.

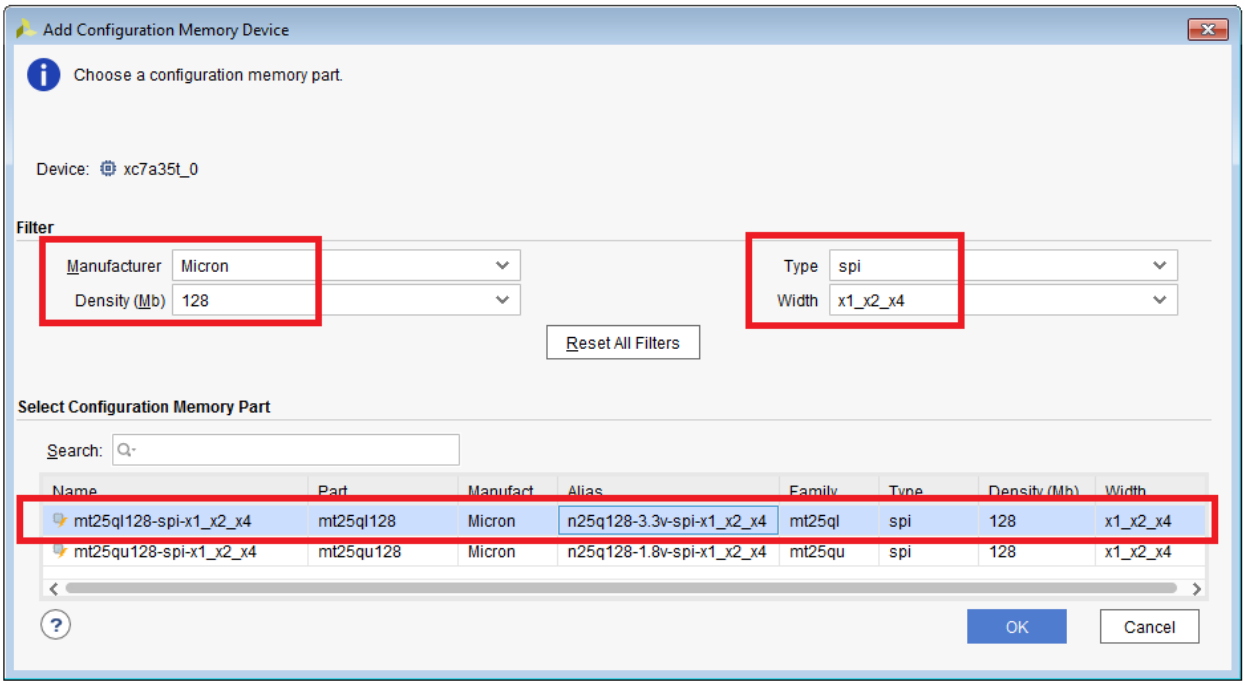
In order to get the FPGA automatically programmed at power up you have to write the FPGA configuration into some dedicated **external flash memory**. The Digilent Arty A7 board provided a **128 MB Quad SPI Flash memory** by Microsemi for this purpose.

To program the external memory we have to first add the memory to the JTAG chain from the *Hardware Manager*. To do this, right click on the `xc7a35t_0` device and select **Add Configuration Memory Device**.

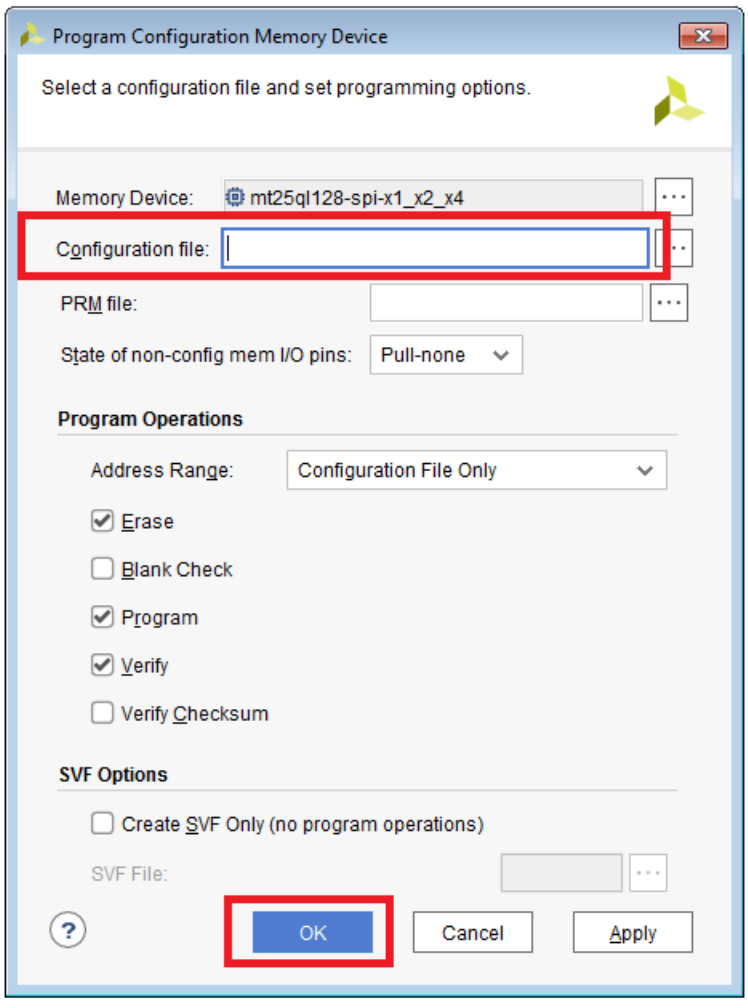


The device name to be selected is **mt25ql128-spi-x1_x2_x4** as follows:

- *Manufacturer*: **Micron**
- *Density (Mb)*: **128**
- *Type*: **spi**
- *Width*: **x1_x2_x4**



Left click on **OK** once selected. Finally, specify the **memory configuration file** to be flashed in the memory. In this case the file to be specified is the **raw binary** `Inverter.bin` created by Vivado in the `Inverter.runs/impl_1/` directory. Left-click on **OK** to start the memory programming.



Once the firmware has been written to the external memory the **DONE** status LED turns on again.

Review all Tcl programming commands traced in the Tcl console.

Close the USB/JTAG chain from the *Hardware Manager* using:

```
disconnect_hw_server
```

Finally, disconnect and then riconnect the USB cable from the computer to verify the **firmware persistence across power cycles**.

> **QUESTION**
>
> Disconnect the USB cable from the computer and then remove the **MODE** jumper on **JP1**. What happens when you reconnect the USB cable ?
>
> _____

Further readings

[Contents]

If you are interested in more in-depth details about the usage of the *Hardware Manager* and other FPGA programming features, please refer to Xilinx official documentation:

- *Vivado Design Suite User Guide: Programming and Debugging (UG908)*
- *7 Series FPGAs Configuration User Guide*