Management and analysis of physics datasets, Mod. A

First Laboratory

Antonio Bergnoli bergnoli@pd.infn.it

2/11/2021

Laboratory Introduction

Easy badges

nov. 2: 912057

nov. 3: 566633

Goals

- Install our FPGA software tool (Xilinx Vivado), in a virtual machine or from a native installation (Linux and Windows only)
- Become familiar with the Xilinx Vivado IDE.
- Implement the VHDL **Hello World**. Then synthesize, simulate and download the project in the Artix-7 FPGA Development Board.

Remote access to server

1. Establish a ssh tunnel

```
ssh -L 50002:147.162.190.205:3121 <your_username>@gate.cloudveneto.it
```

1. log into `lxilinx1.fisica.pd.infn.it`
2. give the command: `source /tools/Xilinx/Vivado/2018.3/setting.sh`
3. launch vivado hw server: `hw_server -s TCP:147.162.190.205:3121`

Remote access to server (ii)

1. Open your local Vivado instance
2. Select *Open Hardware manager*
3. Select *Open target Open New Target*
4. Connect to: *Remote Server*
5. Host name: `lxilinx1.fisica.pd.infn.it`
6. Port: *50002*

Software installation

Vivado preferred version: 2018.3

Web Links (registration required)

1. Windows version
2. Linux version

Virtual machine

1. Download and install the software for your operating system: https://www.virtualbox.org/wiki/Downloads
2. double click on the *.ova* file

Run the native installer (Linux)

1. Linux:for debian based distros : `apt install libtinfo5 libncurses5`
2. Linux: enter the Xilinx software Package; run `./xsetup` as root
3. Linux: `cd /tools/Xilinx/Vivado/2018.3/tools/data/xicom/cable_drivers/lin64/install_script`
4. Linux: run `./install_drivers`

Run the native installer (Windows)

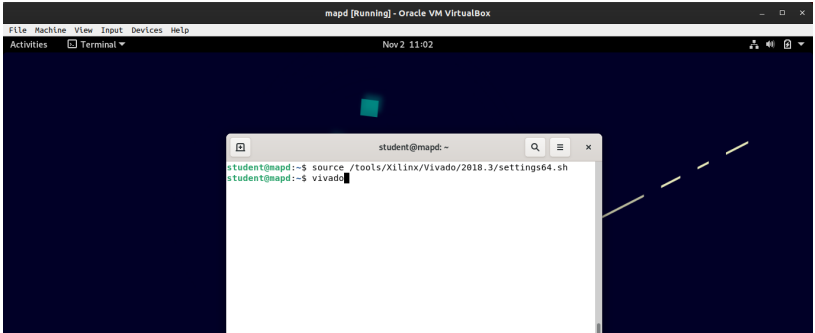
1. Enter the Xilinx software package, run `xsetup.exe`
2. remember to check *install cable drivers*

Virtual Machine

1. username: **student**
2. password: **student01**
3. root password: **student01**

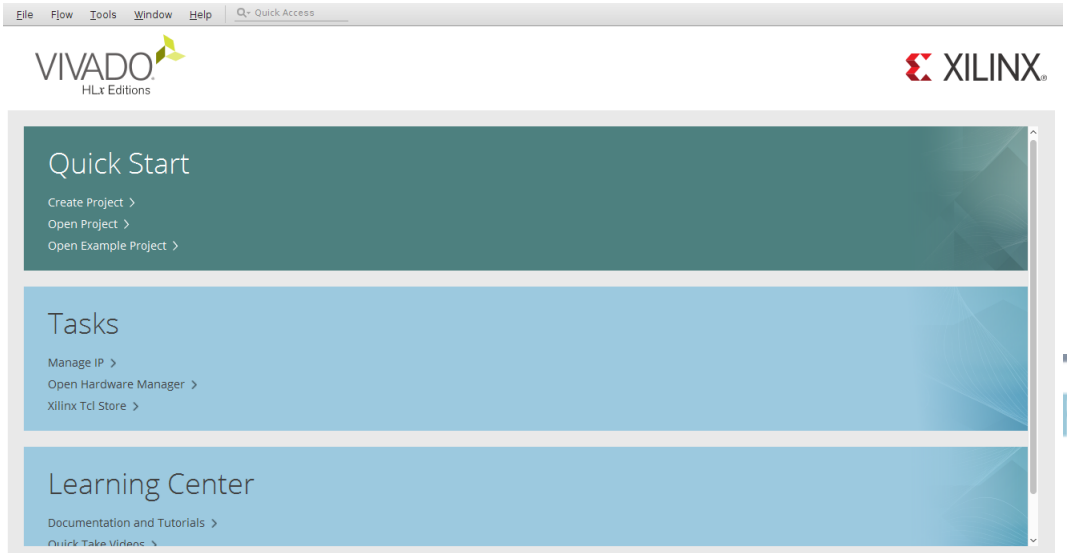
Run Vivado

1. open a terminal
   - press *windows key* or *command key* (OSX) or click *Activities*
   - type `term`
2. Type: `source /tools/Xilinx/Vivado/2018.3/settings.sh`
3. Type: `vivado`



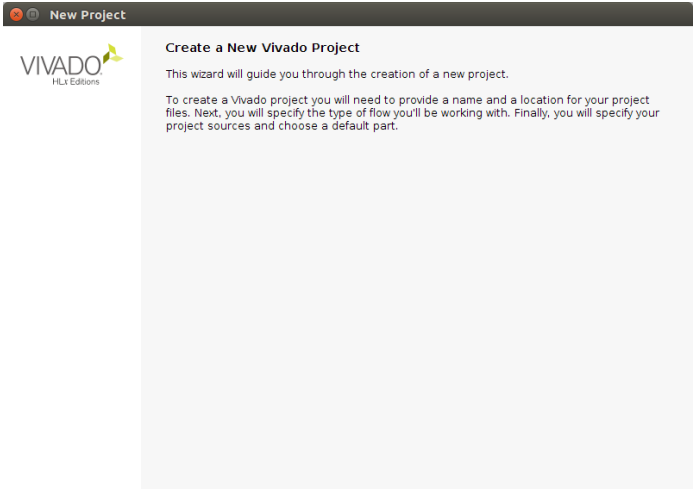
New Project

Presentation screen



Make a new project (1)

File → Project → New ...

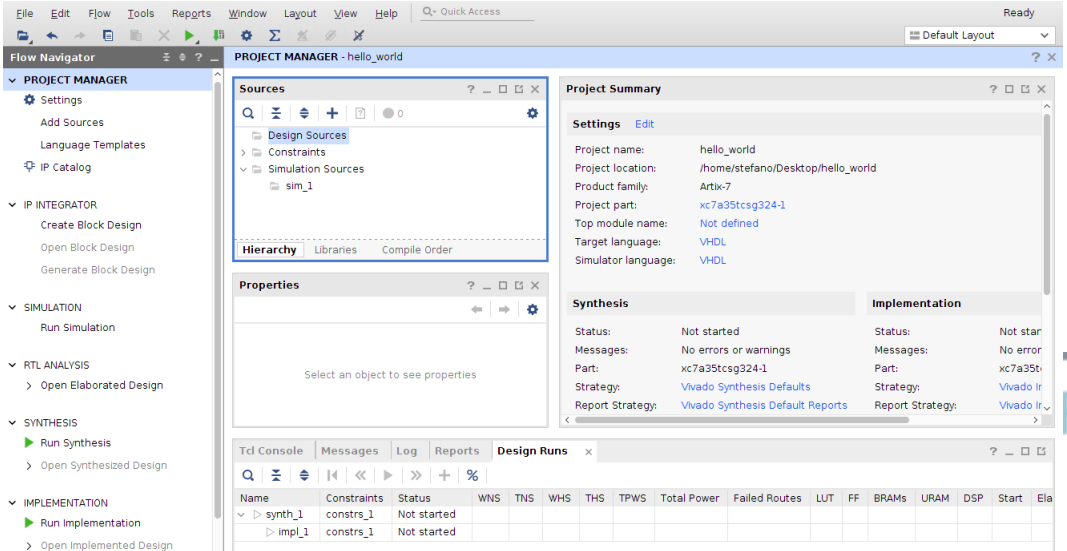


Make a new project (2)

1. Project name : **hello_world**;
2. Check "Create project subdirectory";
3. Next →;
4. Check "RTL Project";
5. Next →;
6. Target language : **VHDL**;
7. Simulator language : **VHDL**;
8. Next →;
9. Next →;
10. Search : **xc7a35tcsg324-1**;
11. Next →;
12. Finish →.

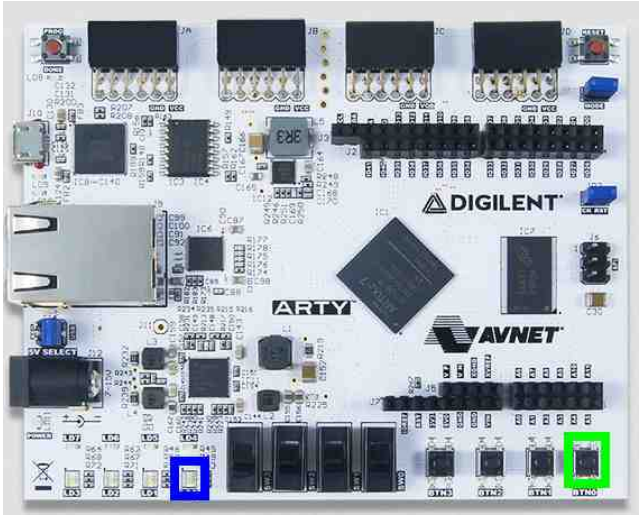
Evaluation Board - FPGA

Make a new project (3)

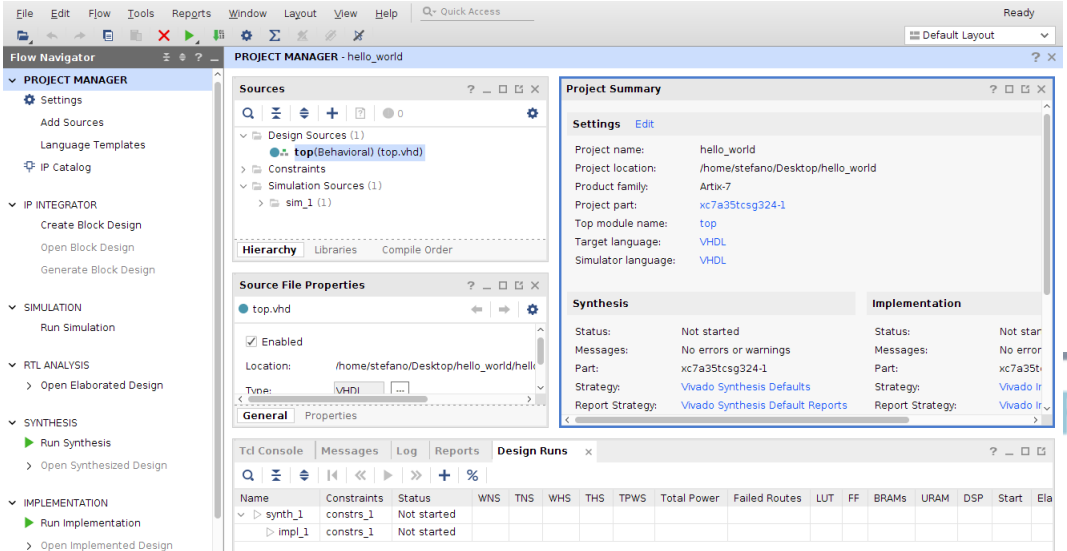


Hello World

Hello World

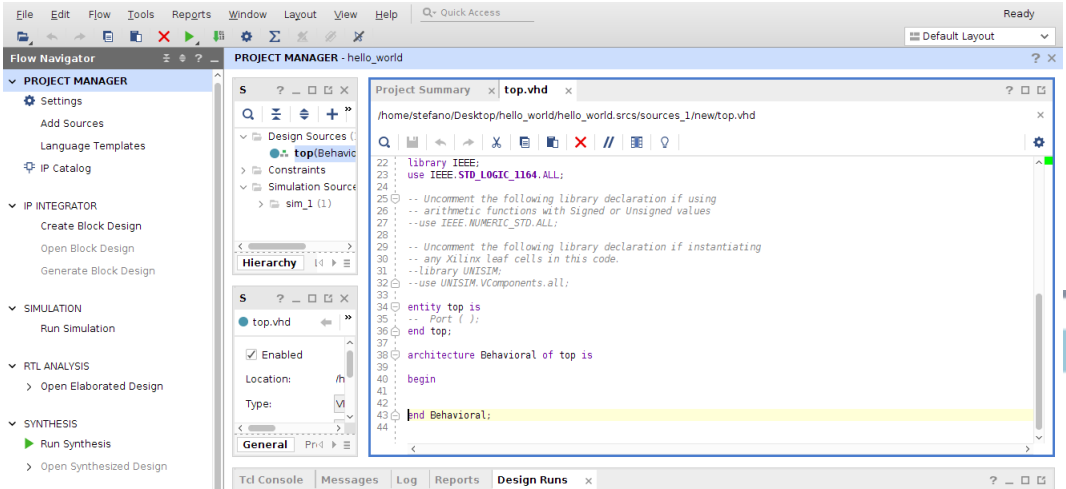


Make the source file (1)



Make the source file (2)

Double click on top.vhd.



Make the source file (3)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity top is
  port (btn_in  : in  std_logic;
        led_out : out std_logic);
end top;
architecture Behavioral of top is
begin
led_out <= btn_in;
end Behavioral;
```

VHDL naming convention

Signals/components	Name
Clock	clk
Reset	rst
Input Port	$port\_in$
Output Port	$port\_out$
VHDL file name	$entityname.vhd$
Test bench file name	$tb\_entityname.vhd$
...	...

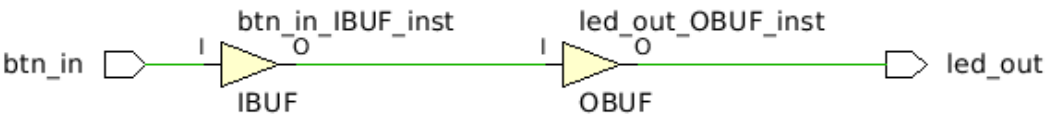
Synthesis (1)

Why we need it?

- To check possible errors in the code.
- To translate the VHDL code in a netlist. A netlist is a list of the logic gates, flip-flops, other components and a list of the connections between them.

Synthesis (2)

1. Run Synthesis →;
2. Cancel →;
3. Open Synthesized Design →;
4. Schematic.



Implementation (1)

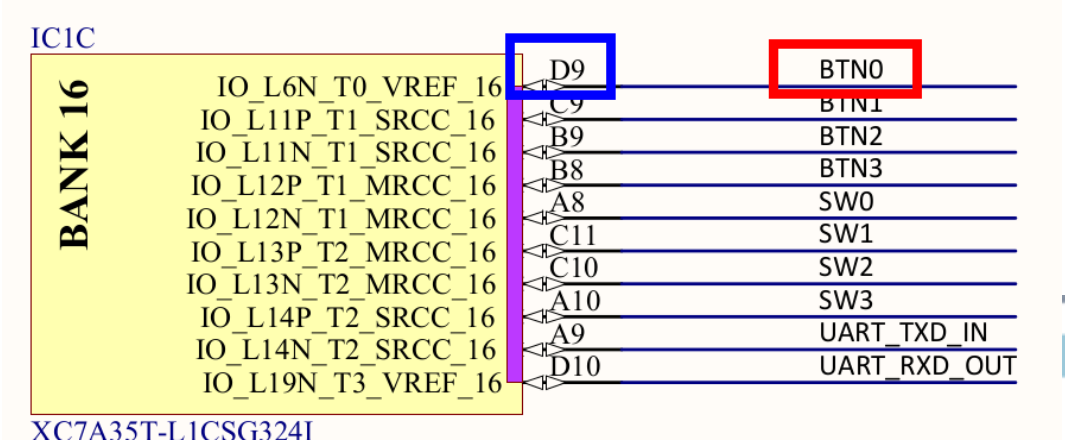
- In order to link the VHDL code to the FPGA pin, it is essential write a constraint file (.xdc);
- therefore we need two files: the schematic of the evaluation board and the xdc file of the FPGA mounted on the board;
- **schematic**
- **constraints file**

In this first example :

- The input (btn_in) has to be connected to the btn0 button.
- The output (led_out) has to be connected to the led0. (In particular to the blue led. $\rightarrow$ LED0_B)

Implementation (2)

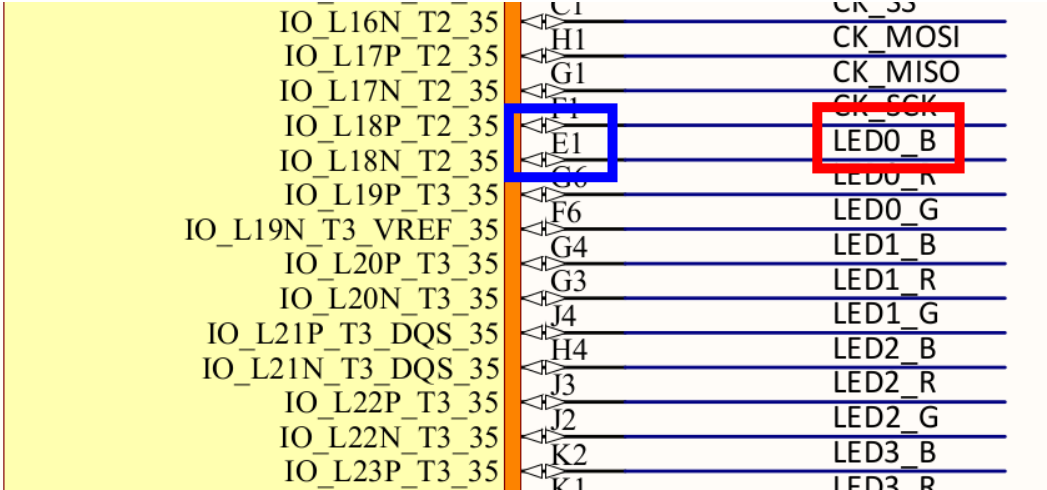
Find in the *schematic* btn0.



It is connected to the FPGA pin **D9**.

Implementation (3)

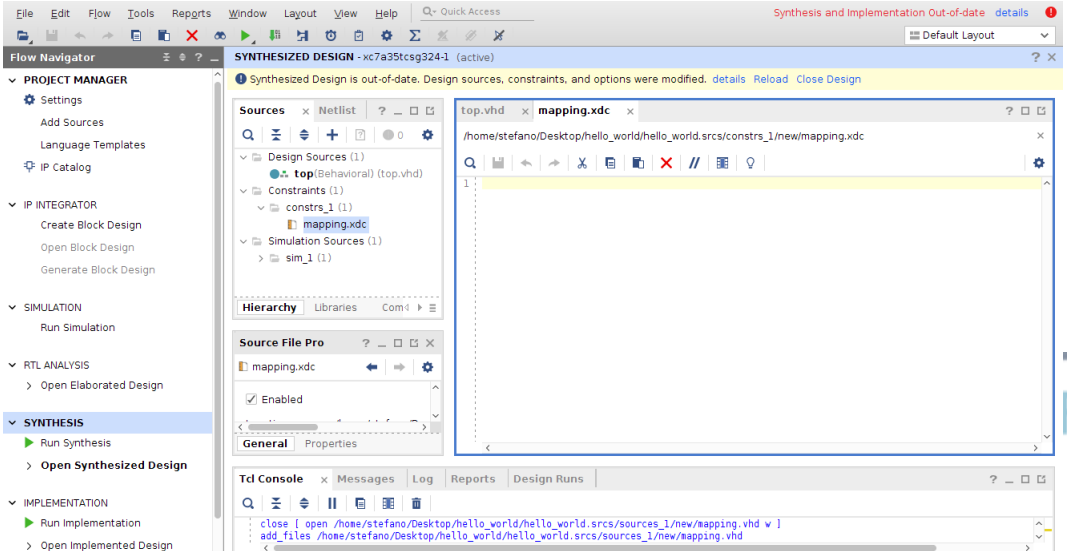
Find in the *schematic* led0_b.



Implementation (4)

1. Add sources;
2. Check ”Add or create constraints”;
3. Next $\rightarrow$;
4. Check ”Create File”;
5. File name: ”mapping”;
6. Finish $\rightarrow$;

Implementation (5) Open the file mapping.xdc.



Implementation (6)

1. Open the file previously downloaded and find **D9** and **E1**;
2. Copy the equivalent lines in the file mapping.xdc;
3. Substitute led_out for led0_b;
4. Substitute btn_in for btn[0];
5. Uncomment the lines (Delete #).

```
set_property  PACKAGE_PIN E1 [get_ports { led_out }];
set_property  IOSTANDARD LVCMOS33  [get_ports { led_out }];
#IO_L18N_T2_35 Sch=led0_b
set_property  PACKAGE_PIN D9 [get_ports { btn_in }];
set_property  IOSTANDARD LVCMOS33  [get_ports { btn_in  }];
#IO_L6N_T0_VREF_16 Sch=btn[0]
```

Implementation (7)

Why we need it?

- To "merge" the netlist and the constraint file, creating a unique design project file.
- To map the components listed in the netlist, in the resources provided by the FPGA.
- To place the resources in the chip and to route them together according to the constraints.

1. Run Implementation →;

Programming (1)

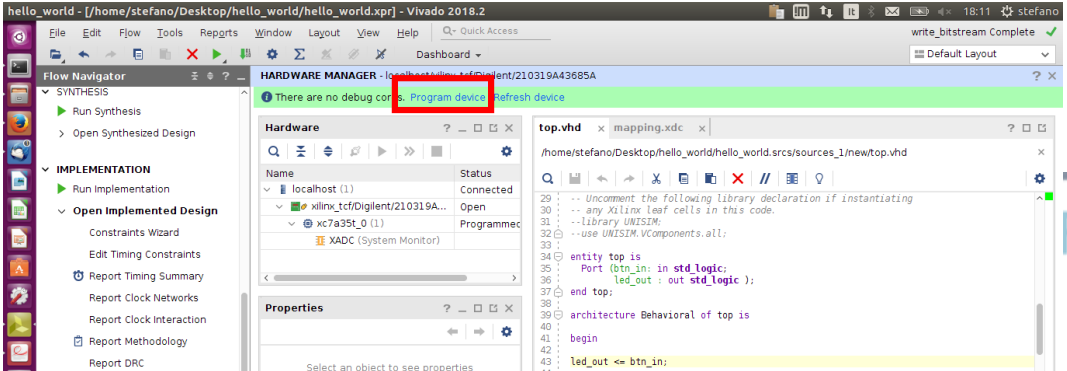
Generate Bitstream $\rightarrow$;

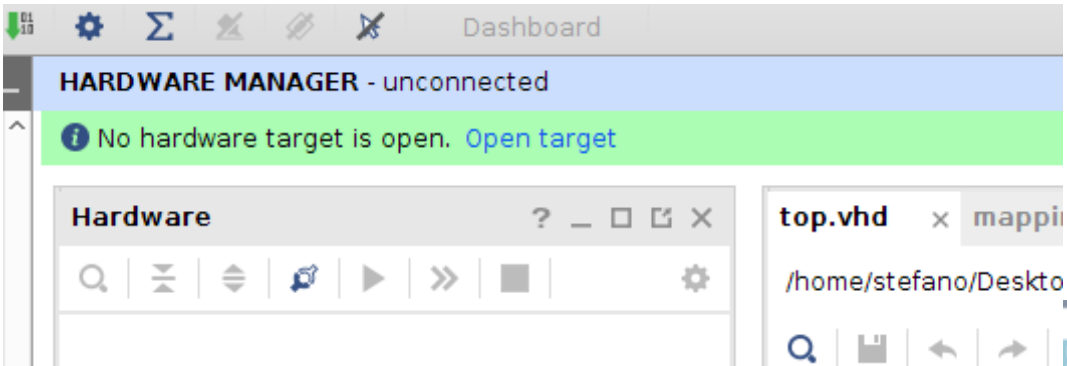
Why we need it?

- To generate the bitstream file (.bit);
- this file represent the final configuration to set the FPGA;
- the file is then downloaded into the FPGA.

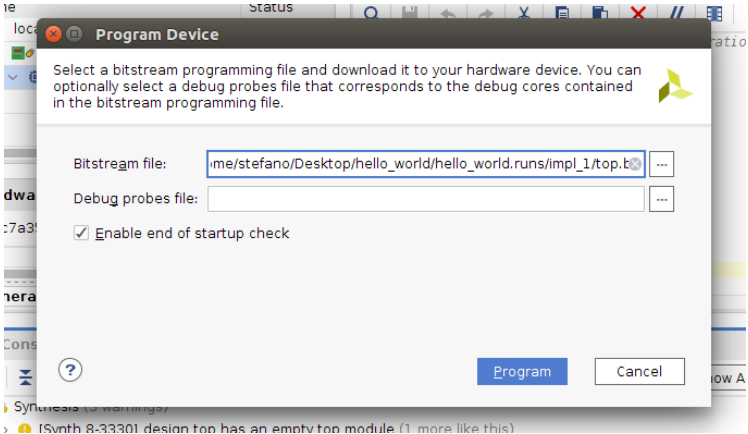
Programming (2)

1. Connect the evaluation board to PC by the usb cable.
2. Open Hardware Manager →.
3. Open Target →.
4. Auto Connect.




Dashboard
HARDWARE MANAGER - unconnected
No hardware target is open. Open target
Hardware
top.vhd
/home/stefano/Deskto

Program device →.



(If not checked, check ”Enable end of startup check”.)

Programming (5)

Try to press btn0.

Homework

Suggested exercises

- Redo 1,2,3, ..., N times the exercise ”Hello World”.
- Get the code more complicated (a little bit). For example instantiate more inputs. Hence repeat each step of ”Hello World”. An example is reported in the next slide.
- Other complications. More inputs and more outputs.

Hello World with two inputs

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity top is
  port (btn_in  : in  std_logic_vector(1 downto 0);
        led_out : out std_logic);
end top;
architecture Behavioral of top is
begin
  led_out <= btn_in(0) xor btn_in(1);
end Behavioral;
```