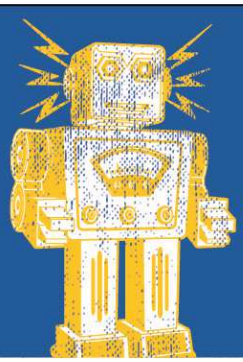
Elektronik

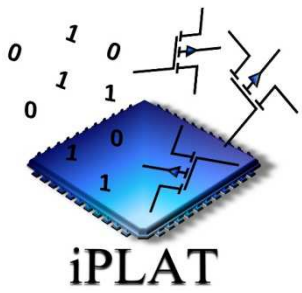


Digital System Design

Distance Learning Letter
Vivado HL WebPACK
Installation/Introduction

Gefördert von
MA23
Wirtschaft, Arbeit Statistik
StaDt+Wien

iPLAT – Competence team for Innovative Platforms for Mixed-Hardware/Software Systems
MA23 Project 18-06



Version: 1.2
Author: R. Höller, A. Puhm, F. Schrön, P. Rössler

Copyright Notice

Introduction

This Distance Learning Letter documents the installation process as well as how to use the FPGA implementation tool Vivado 2016.1 HL WebPACK (the documentation refers to release “2016.1” but will probably work also for future versions). This tool will be used in this course to implement a VHDL design on a Xilinx FPGA. The document concludes with an example project for the usage of the Vivado tool.

Installation

The Vivado tool from Xilinx is already installed on all PCs of the Department of Embedded Systems. If you want to work with the PCs, you can skip the installation part and start with the example in the section below. Otherwise, please follow the instructions in this section.
The installation file (Windows and Linux) is located on the CIS website of the lecture. If you need the password for the extraction of the file, please ask your lecturer. Alternatively, the current version of the software can also be downloaded from the Xilinx website (http://www.xilinx.com/). This requires a free Xilinx account and a free license. **Attention!** If you download a newer version of the Vivado Design Suite, it could be possible that this version is not compatible with the versions installed on the PCs of the Department of Embedded Systems. Additionally, the development boards used in this course are possibly not supported by a newer tool version.
In order to install the Vivado WebPACK, please follow the instructions in the given sequence[1]:

1. First, extract the ZIP-file (ask your lecturer for the password). Then, extract the TAR-file to a path of your choice. Execute the file `xsetup.exe`, which is located in the extracted folder.
2. Click “Next >” on the prompted screen, after checking if the used operating system is supported. If the used operating system is not supported, problems may occur during program usage.

[1] The following instructions are based on MS Windows. The installation on Linux is similar to the installation on Windows.

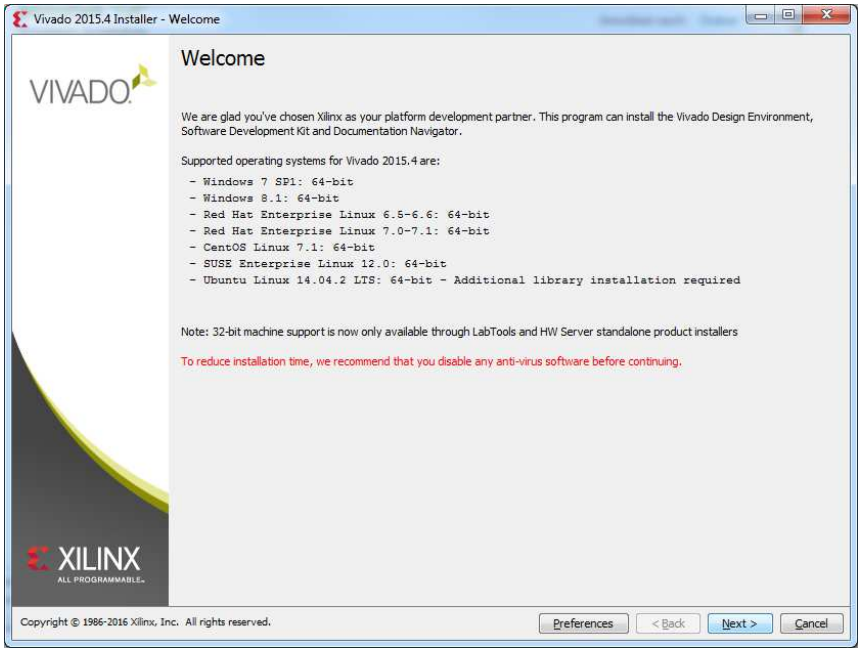


Figure 1: Vivado Installer Welcome screen

3. Accept the license agreements and click “Next >”.

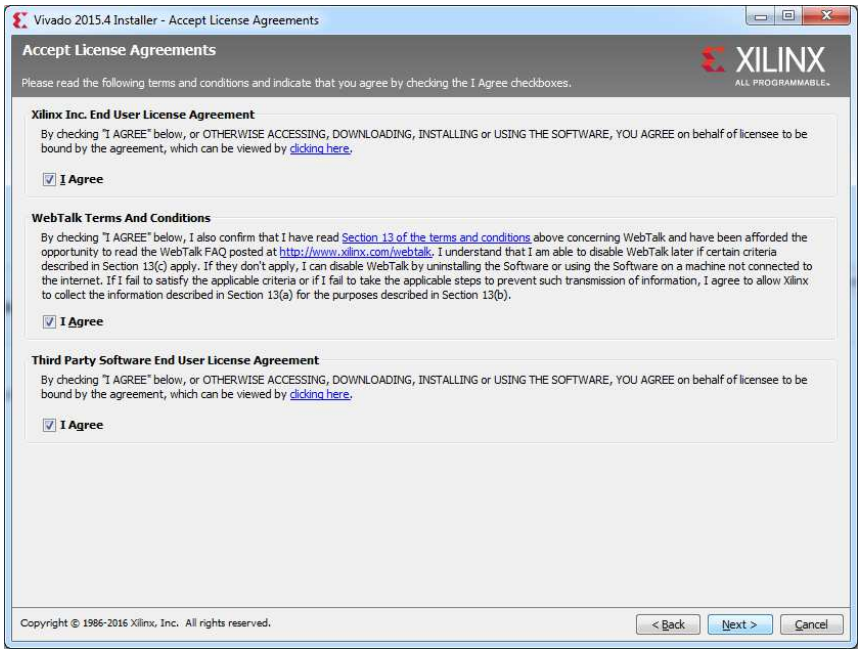


Figure 2: License Agreements

4. This page shows several different Vivado Editions which could be installed. Please select the Vivado HL WebPACK. This edition is a no-cost and device-limited version. Afterwards, click “Next >”.

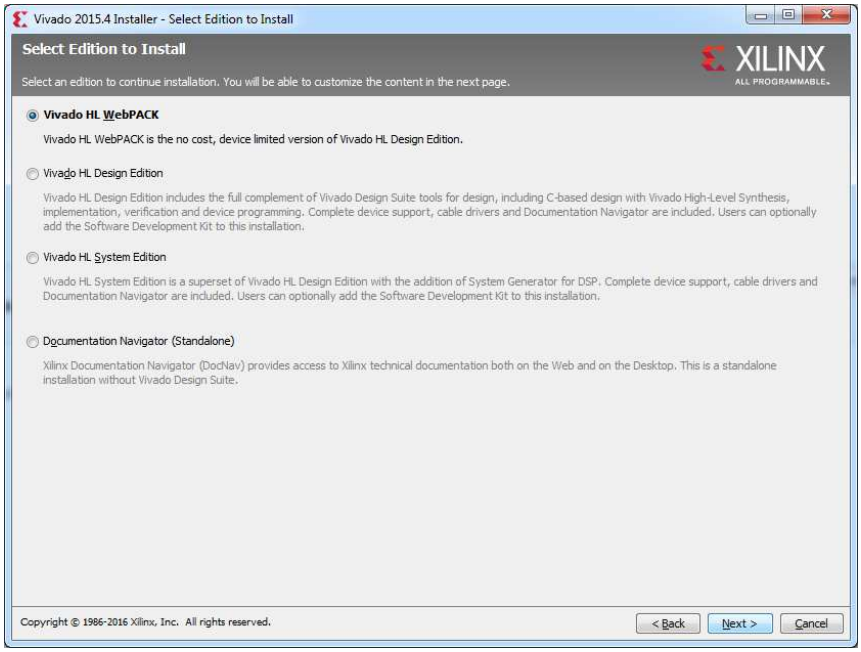


Figure 3: Edition selection

5. Now, choose the components to install. The components to select depend on the lecture:

   * **For the lecture “System on Chip Design” of the study program “Master Embedded Systems” ONLY**, select the Zynq-7000 device family (because this device is contained on the FPGA board that is used in this lecture) as well as the “Software Development Kit (SDK)”, see Figure 4.
   * **For all other lectures** select the Artix-7 device family because an Artix-7 chip is contained on the Basys3 development board that is used in the lecture, see Figure 5

   Click “Next >” to proceed.

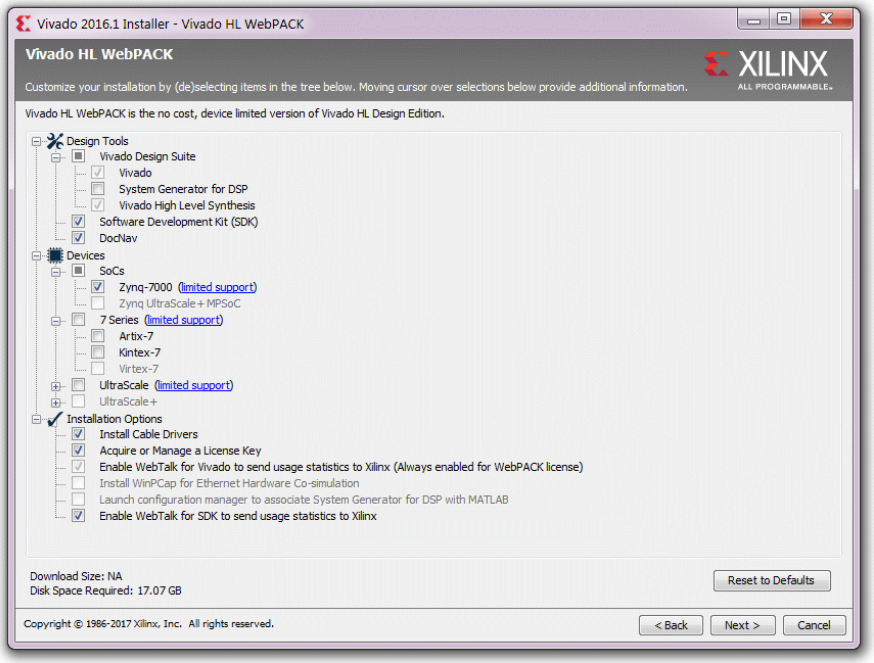


Figure 4: Component selection (**lecture "SoC Design", study program MES only!**)

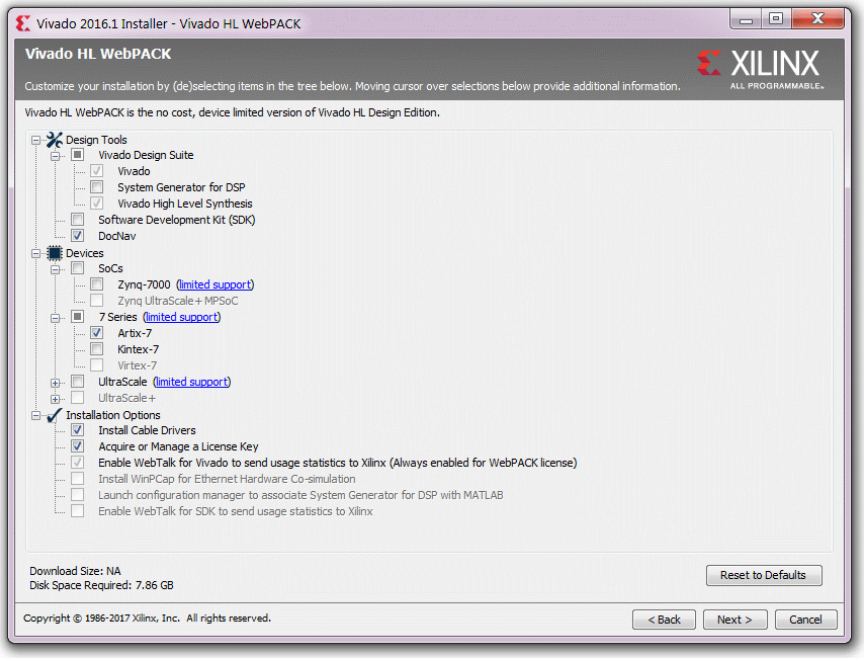


Figure 5: Component selection (**all other lectures**)

6. Choose the destination directory for the installation. **Do NOT use directory names that contain blanks or other special characters.** The easiest way is to maintain the default settings and press the "Next >" button. If the specified directory does not exist, you will be asked to create it. In this case, click "Yes".

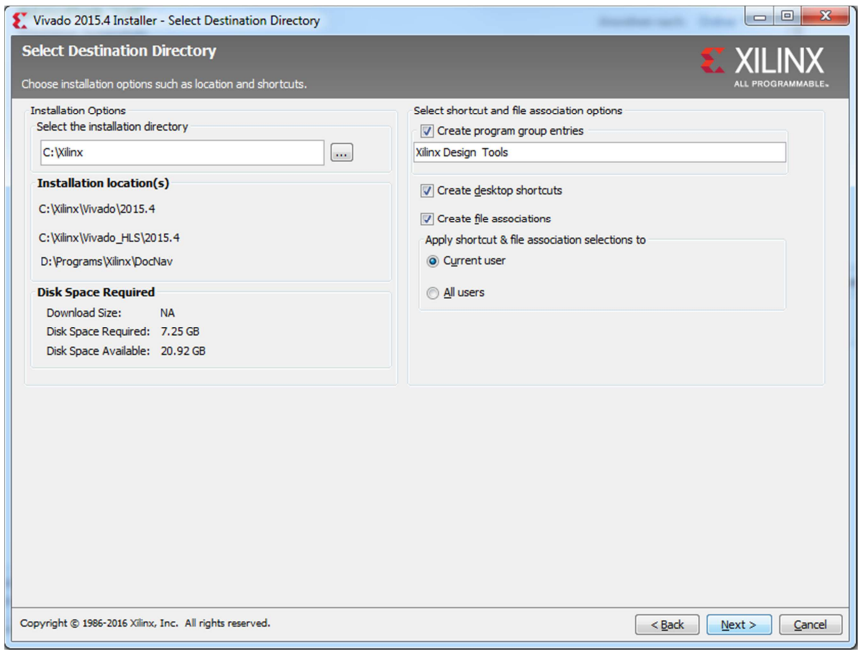


Figure 6: Destination selection

7. This screen gives you a short overview about the installation. Click “Install” to install the Vivado HL WebPACK.

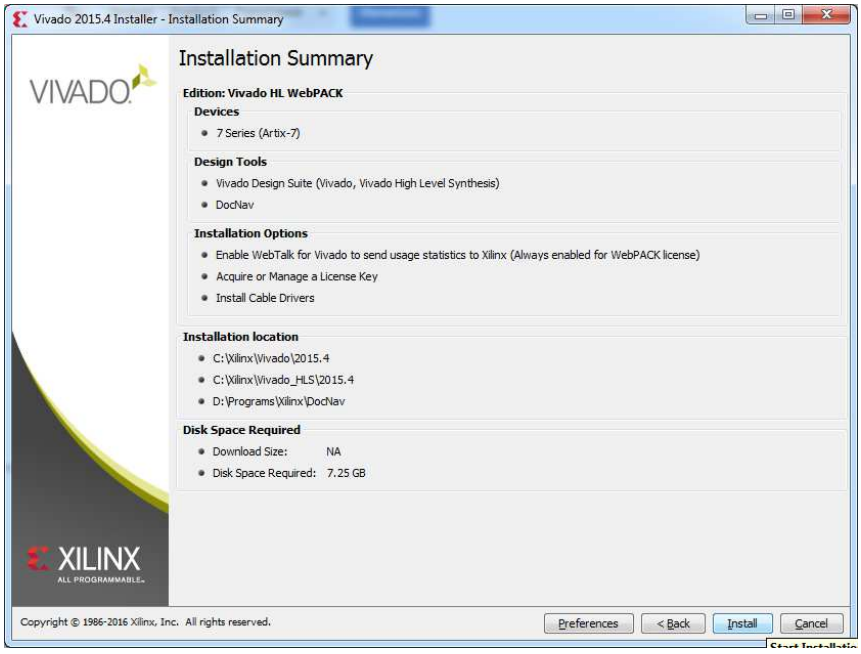


Figure 7: Installation summary

The installation will usually take a few minutes. After the installation, the license manager will open. Close it by pressing the “X” Button.

Useful Tips when using Vivado

Synthesis, implementation and bitstream generation are time-consuming design steps, especially when larger FPGA designs are implemented. In order to speed up Vivado the following hints can be useful:

- The default power settings of many laptop computers and notebooks are often configured to save battery power rather than to maximize processing power. Thus, have a look at the power settings of your PC and configure them to maximum performance.
- It was observed that Vivado performs very slow on some network connections. Therefore, it can be useful to disable your network connections (for example, simply disconnect the LAN cable of your PC if you have a wired network connection) while you work with Xilinx Vivado.

For the lecture “System on Chip Design” of the study program “Master Embedded Systems” you don’t need the remaining sections of this document. For all other lectures, proceed with a simple implementation example that is described in the following section.

The first implementation example

The following example describes the Vivado design flow by using a simple design. Download the file `running_led.zip` from the CIS website of this lecture und unzip it to

```
d:\work\
```

This results in a directory structure as shown in Figure 8.

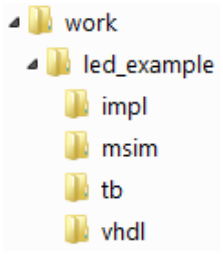


Figure 8: Directory structure of the running led example design

The VHDL code of the example design is shown below and can be found in the sub-directory `vhdl`.

```
--------------------------------------------------------------------------------
--                                                                            --
--                                                                            --
--                       X        XXXXXXXX  XXXXXXX                          --
--                       X        X         X      X                         --
--                       X        X         X      X                         --
--                       X        X         X      X                         --
--                       X        XXXXXXX   X      X                         --
--                       X        X         X      X                         --
--                       X        X         X      X                         --
--                       X        X         X      X                         --
--                       XXXXXXXX XXXXXXXX  XXXXXXX                          --
--                                                                            --
--                                                                            --
--       F A C H H O C H S C H U L E   -   T E C H N I K U M   W I E N        --
--                                                                            --
--                         Embedded Systems Division                          --
--                                                                            --
--------------------------------------------------------------------------------
--                                                                            --
--         Web:           http://www.technikum-wien.at/                       --
--                                                                            --
--         Contact:       hoeller@technikum-wien.at                           --
--                                                                            --
--------------------------------------------------------------------------------
--         Author:                 Roland Höller
--
--         Filename:               led.vhd
--
--         Date of Creation:       Sun Oct 20 12:14:48 2002
--
--         Version:                $Revision$
--
--         Date of Latest Version: $Date$
--
--         Description: Describe a simple running LED for the Basys3 board.
--------------------------------------------------------------------------------
--  CVS Change Log:
--
--  $Log$
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity led is
  port ( leds  : out std_logic_vector (15 downto 0);
         reset : in  std_logic;
         clk   : in  std_logic);
end led;

architecture rtl of led is

signal s_ckd : std_logic_vector(28 downto 0); -- used for prescaler
```

```
begin
  -------------------------------------------------------------------------------
  -- Divide the 100MHz input clock, generate slow enable signal.
  -------------------------------------------------------------------------------
  p_prescaler : process (clk, reset)
  begin
    if (reset = '1') then
      s_ckd <= (others => '0');
    elsif (clk = '1' and clk'event) then
      s_ckd <= unsigned(s_ckd) + conv_unsigned(1,1);
    end if;
  end process p_prescaler;

  -------------------------------------------------------------------------------
  -- Let LED run.
  -------------------------------------------------------------------------------
  p_leds : process (clk, reset)
  begin
    if (reset = '1') then
      leds   <= (others => '0');
    elsif (clk = '1' and clk'event) then
      if (s_ckd(28 downto 25) = "0000") then
        leds <= "0000000000000001";
      elsif (s_ckd(28 downto 25) = "0001") then
        leds <= "0000000000000010";
      elsif (s_ckd(28 downto 25) = "0010") then
        leds <= "0000000000000100";
      elsif (s_ckd(28 downto 25) = "0011") then
        leds <= "0000000000001000";
      elsif (s_ckd(28 downto 25) = "0100") then
        leds <= "0000000000010000";
      elsif (s_ckd(28 downto 25) = "0101") then
        leds <= "0000000000100000";
      elsif (s_ckd(28 downto 25) = "0110") then
        leds <= "0000000001000000";
      elsif (s_ckd(28 downto 25) = "0111") then
        leds <= "0000000010000000";
      elsif (s_ckd(28 downto 25) = "1000") then
        leds <= "0000000100000000";
      elsif (s_ckd(28 downto 25) = "1001") then
        leds <= "0000001000000000";
      elsif (s_ckd(28 downto 25) = "1010") then
        leds <= "0000010000000000";
      elsif (s_ckd(28 downto 25) = "1011") then
        leds <= "0000100000000000";
      elsif (s_ckd(28 downto 25) = "1100") then
        leds <= "0001000000000000";
      elsif (s_ckd(28 downto 25) = "1101") then
        leds <= "0010000000000000";
      elsif (s_ckd(28 downto 25) = "1110") then
        leds <= "0100000000000000";
      elsif (s_ckd(28 downto 25) = "1111") then
        leds <= "1000000000000000";
      end if;
    end if;
  end process p_leds;
end rtl;
```

Now start the Vivado design tool. The following window will appear.

Figure 9: Initial view of the Vivado design tool

The button “Create New Project” opens the New Project Wizard (This can also be done via the menu: File -> New Project…). The wizard will guide you through the creation of a new project. All the following configurations are valid for the Basys3 board. On the first screen, click “Next >”.

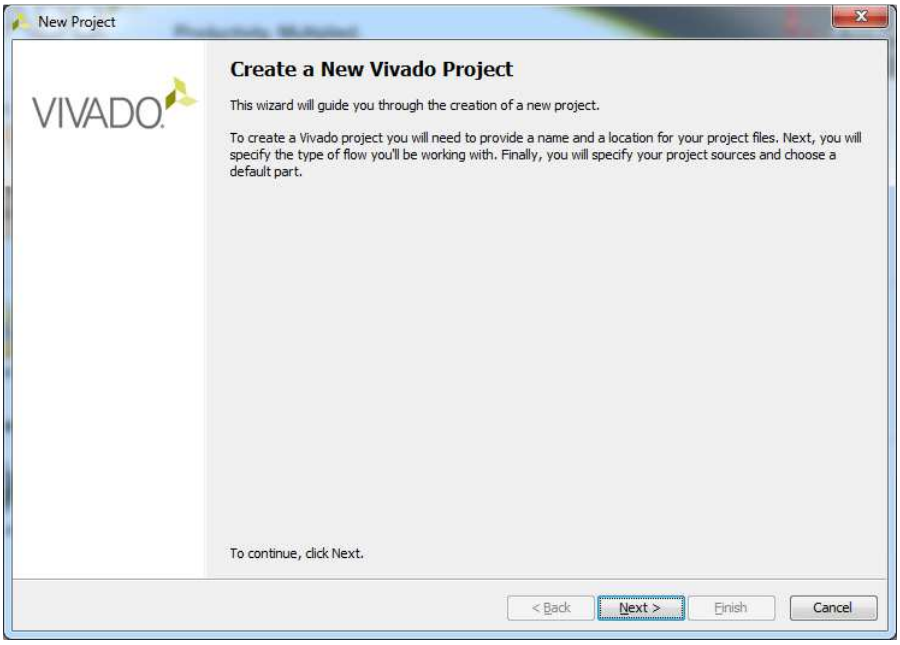


Figure 10: New Project wizard

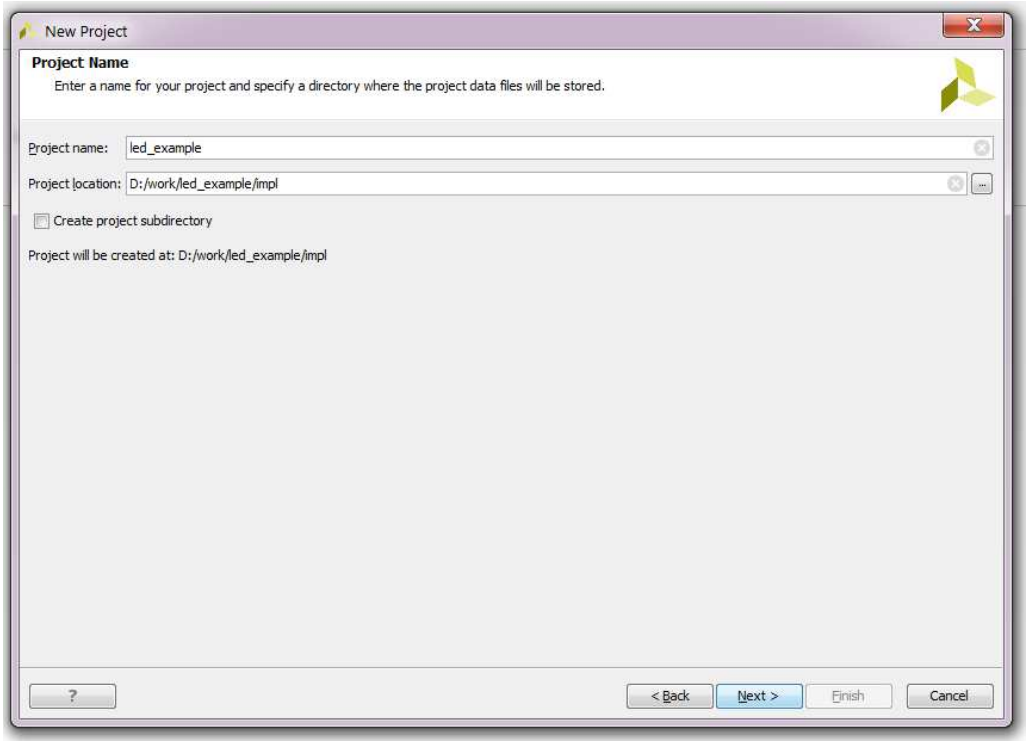


Figure 11: Project name and location

Now, the name of the project and the project location must be defined as shown in Figure 11. Omit special characters or blanks in both the project name and project location! Click “Next >” once you have select your name and project location.

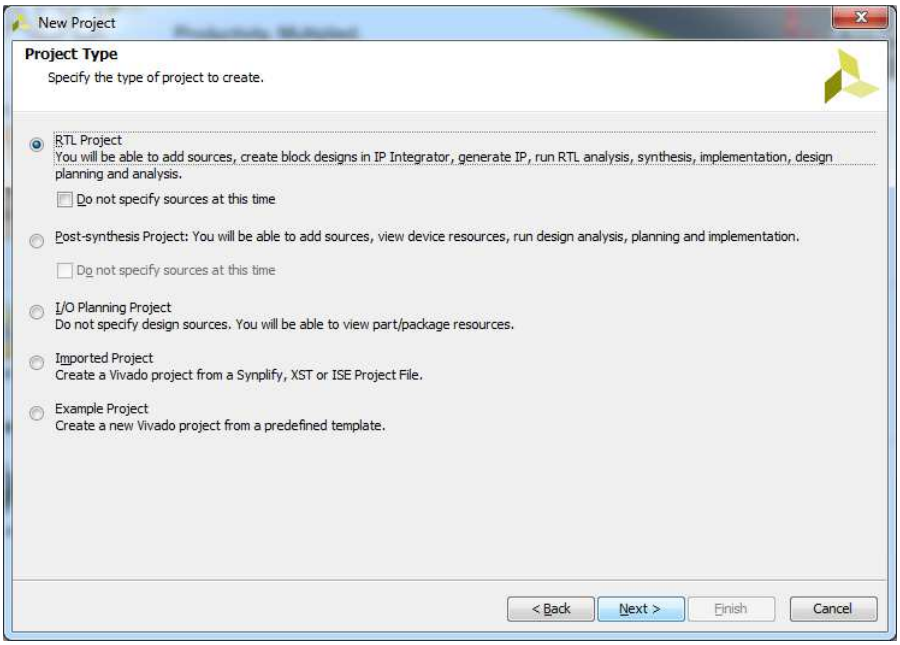


Figure 12: Project type

On this page, the project type must be specified. Please select an RTL Project and click “Next >” as shown in Figure 12.

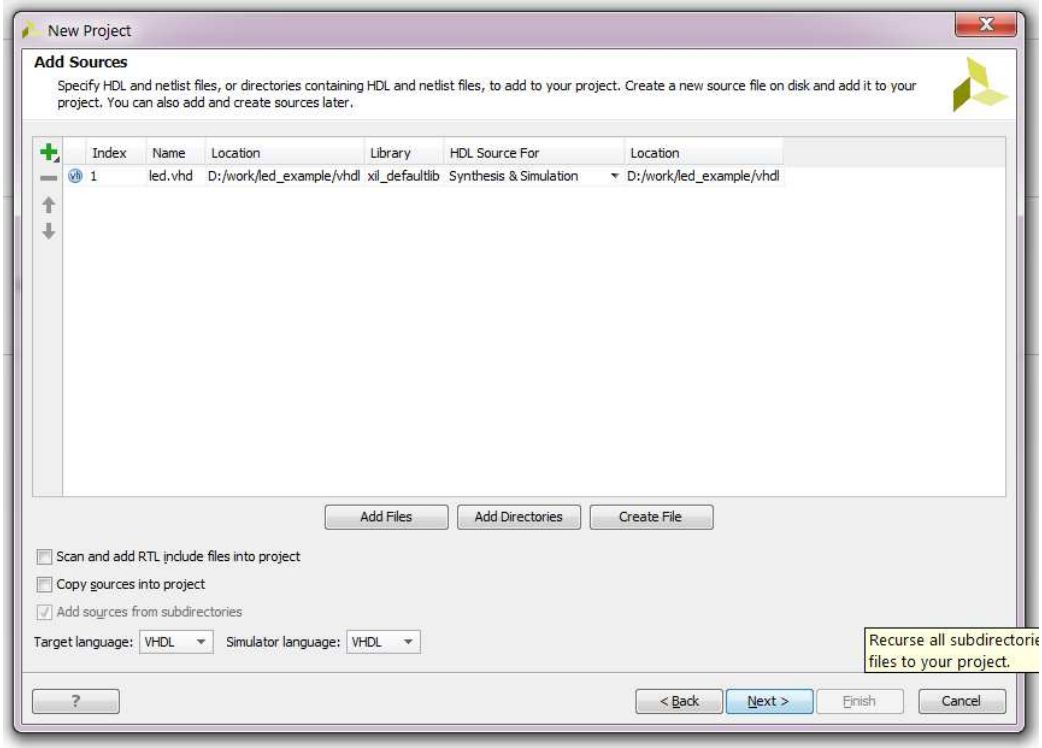


Figure 13: Adding sources

Figure 13 shows the dialog for adding sources. Click “Add Files” or the green plus on the left side and browse to the prepared VHDL file. **Make sure that VHDL is selected as target language! Also, make sure to uncheck “Copy sources into project”!** Click “Next >”. In the following dialogs “Add Existing IP” and “Add Constraints”, simply click “Next >”.

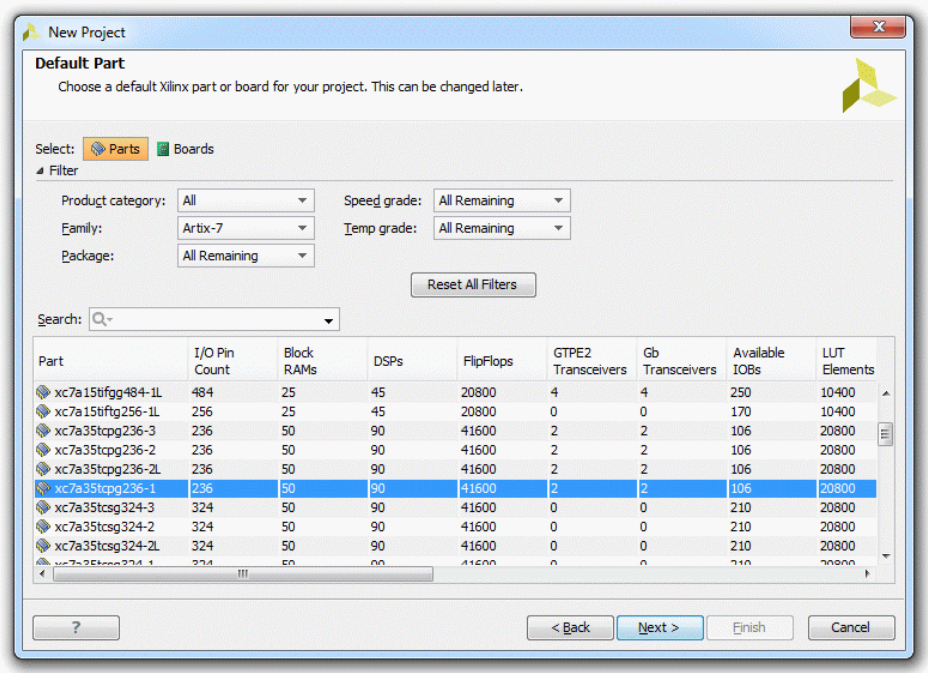


Figure 14: Selection of target chip

In this step, the target FPGA device must be selected. Choose xc7a35tcpg236-1 as shown in Figure 14 since this the FPGA device which is contained on the Basys3 board. Click “Next >”.

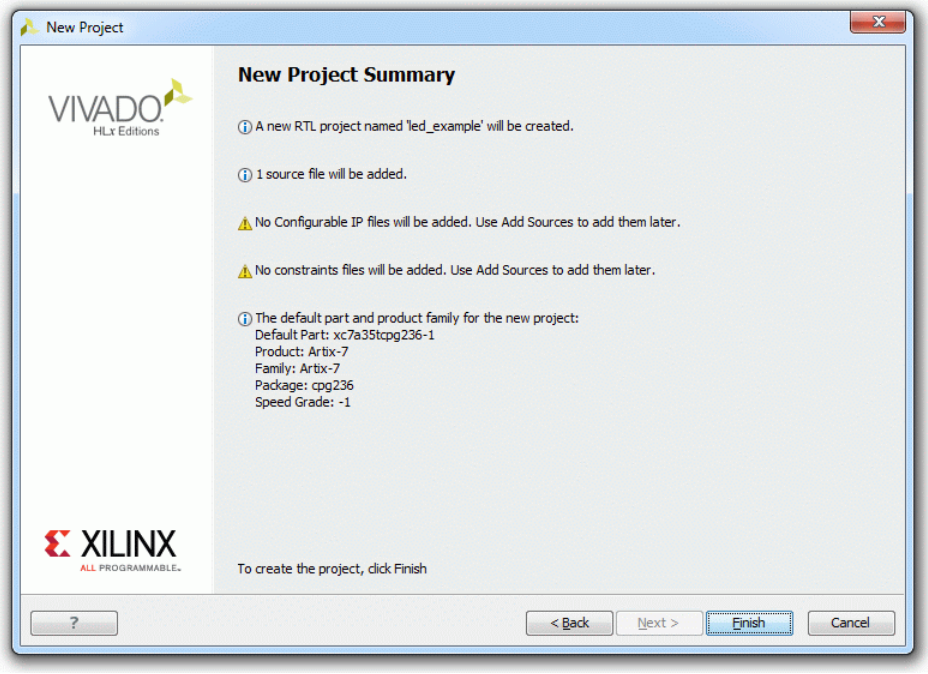


Figure 15: New Project Summary

In the final dialog, a project summary will be displayed as shown in Figure 15. Click “Finish” to create the project.

The Vivado IDE displays the initial view of the opened project, as shown in Figure 16.

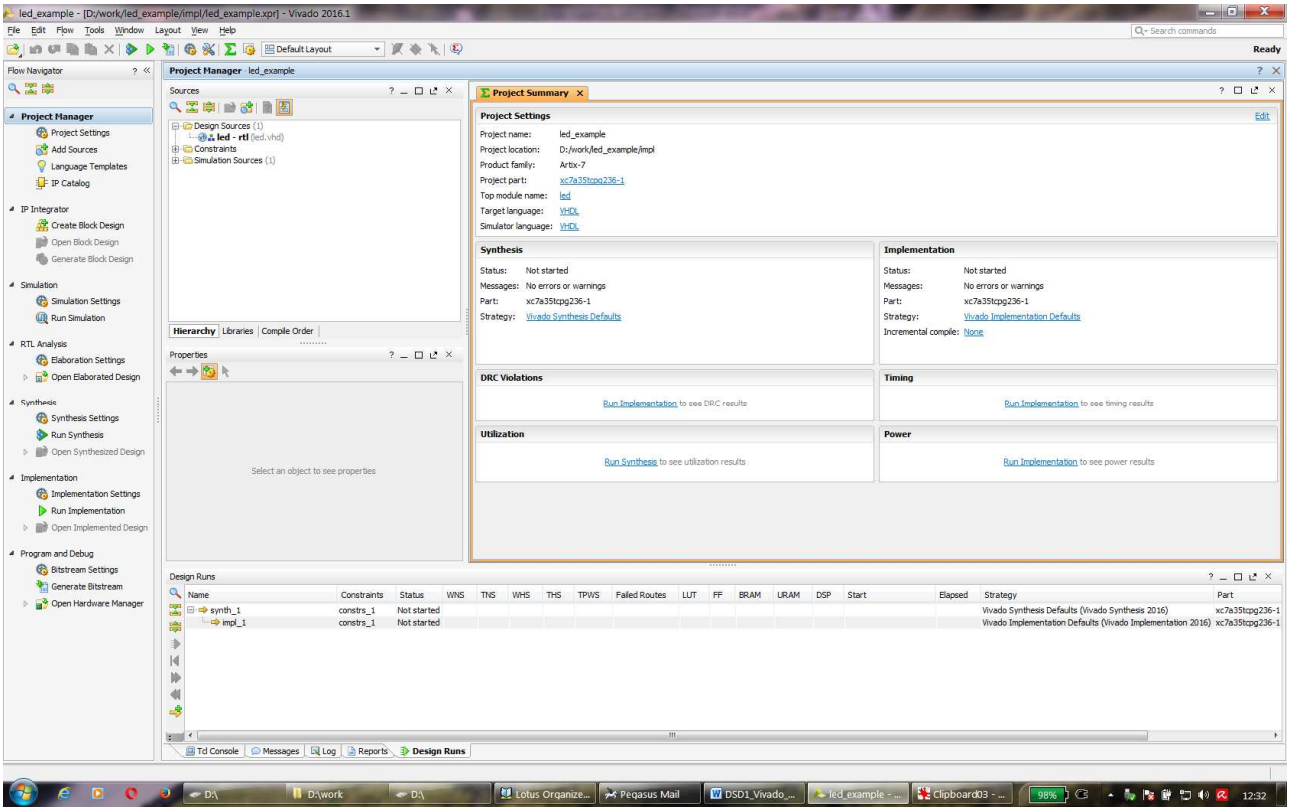


Figure 16: Initial Vivado IDE view

First, it is necessary to synthesize the design. In order to do this, click “Run Synthesis” in the Flow Navigator on the left side of the graphical interface. This will take a few minutes. If synthesis is finished, a window entitled “Synthesis completed” appears. In this window, click “Cancel”. A large part of the user interface, named “Project Summary”, shows the current status of the design flow. At this point, you will only see that the synthesis design step is already completed.

The next step is to create a so called “constraint file”. From the Flow Navigator, select “Add Sources”. Figure 17 shows the window which will open.

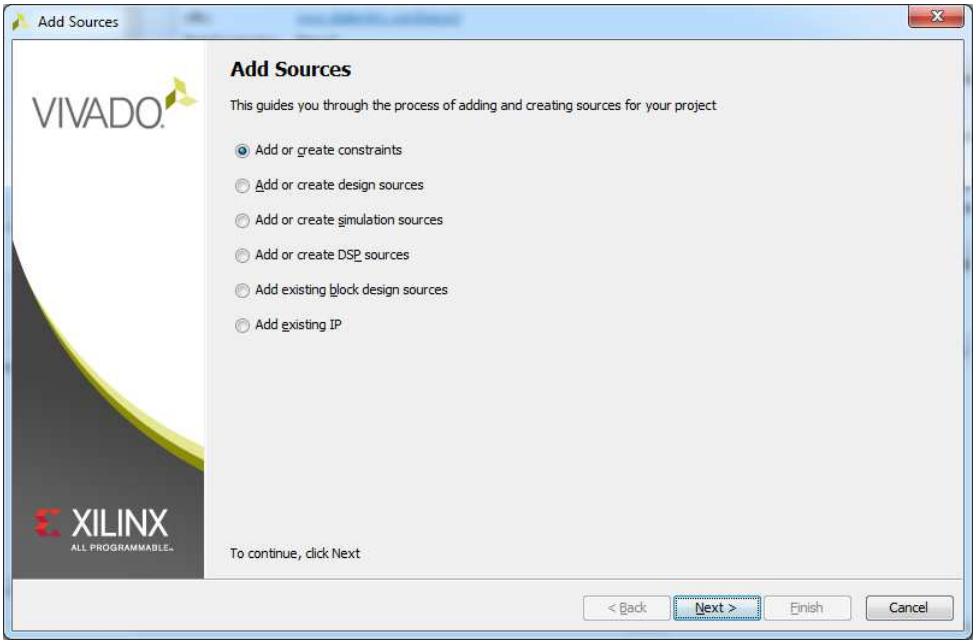


Figure 17: Adding sources in the flow navigator

Select “Add or create constraints” and click “Next >”.

Click “Create File” or click the green plus on the left side to create a constraint file. The window shown in Figure 18 will open.

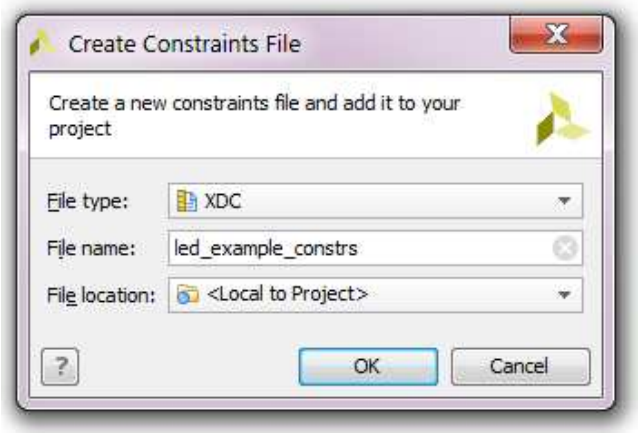


Figure 18: Create constraints file

Name the file as shown in the figure above and click “OK”. In the following dialog, click “Finish”.

In order to instruct Vivado that all constraints added to the design will be saved in the target XDC (Xilinx Design Constraints) file, set the constraint file as the target XDC file: In the Sources window, right-click `led_constrs.xdc` and choose “Set as Target Constraint File” from the context menu, as shown in Figure 19.

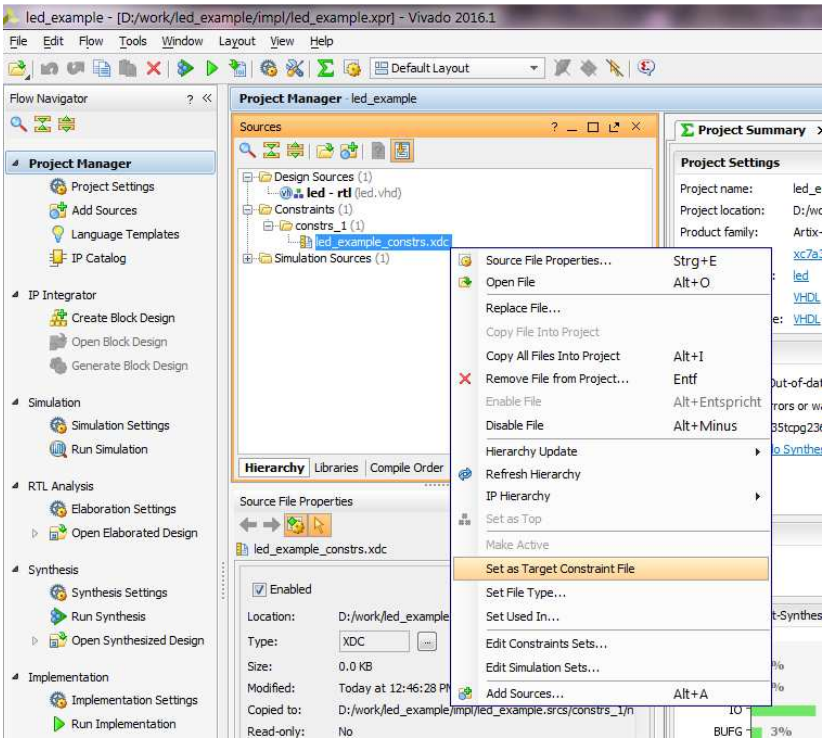


Figure 19: Setting the target constraint file

The next step is to open the synthesized design and use the Vivado Timing Constraints Wizard. The Timing Constraints Wizard analyzes the gate level netlist and finds missing constraints. In this design, the Timing Constrains Wizard is used to define a constraint for the source clock (named `clk` here). In the Flow Navigator, click “Constraints Wizard” as shown in Figure 20. If a dialog appears called “Synthesis is Out-of-date”, click “Open Design”.

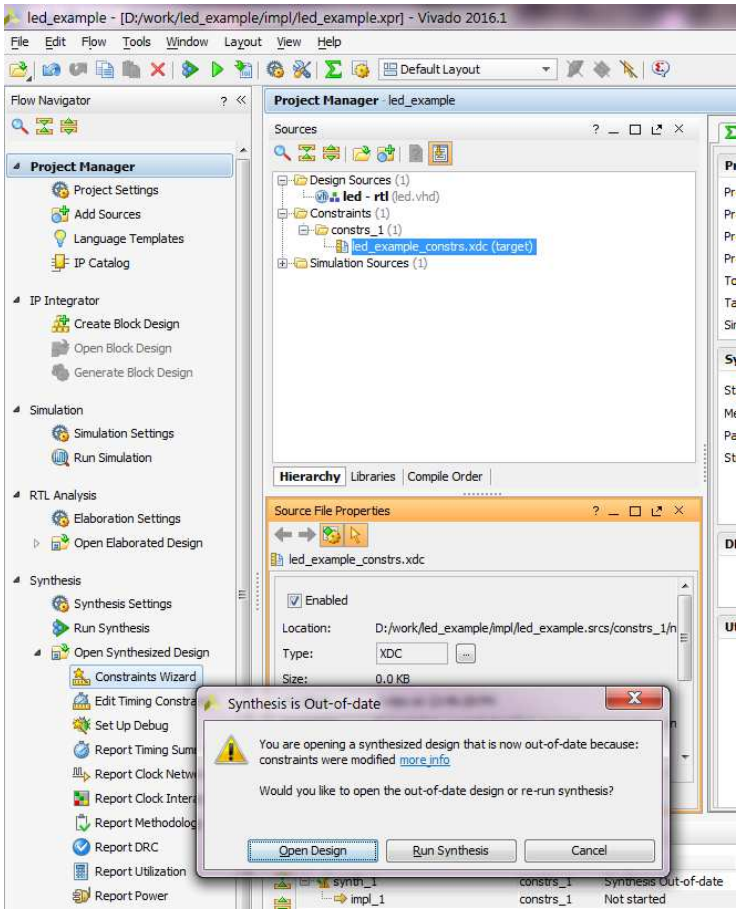


Figure 20: Constraints Wizard

In the appearing window, click “Next >”. The following window shows the primary clocks that have been found by Vivado in the design. Change the clock frequency to 100 MHz as shown in Figure 21, since this is the clock frequency that is generated by the clock oscillator contained on the Basys3 FPGA board.

Click “Skip to Finish >>” and finally click “Finish”.

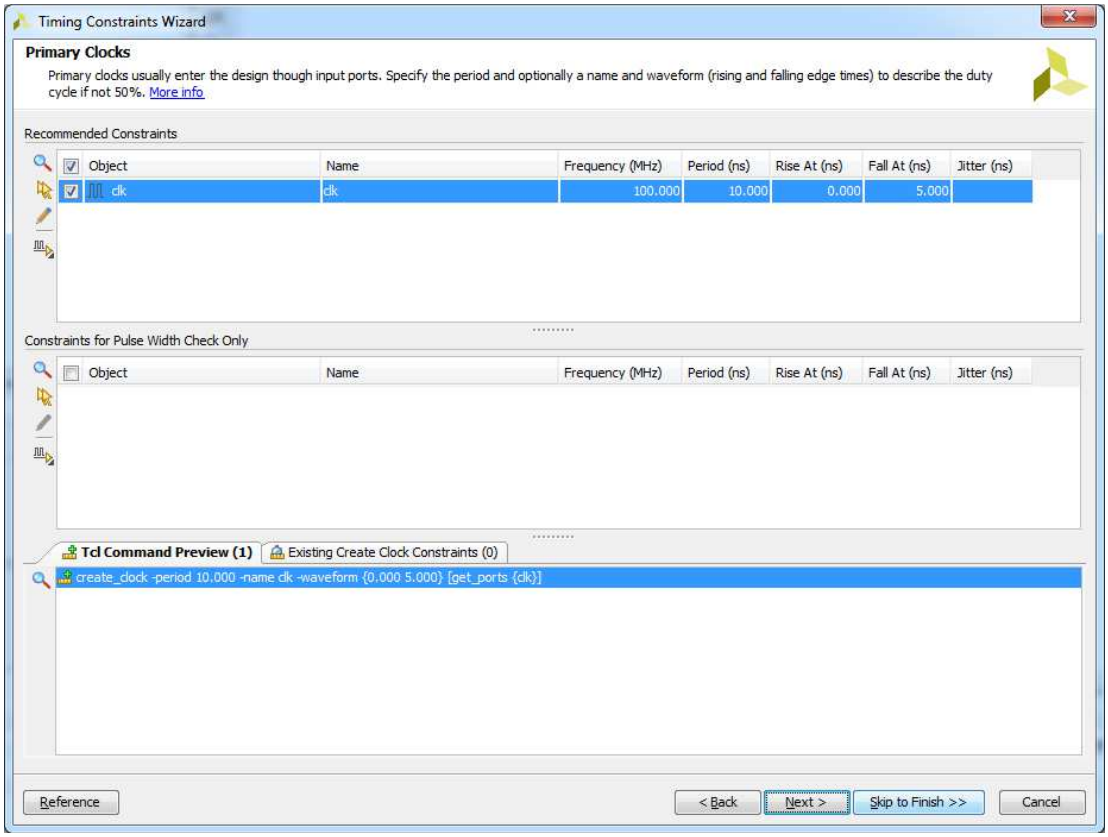


Figure 21: Setting the clock frequency

The next step is to assign the ports of the VHDL top-level entity design to IO pins of the FPGA. Select “Layout -> I/O Planning” from the menu bar. Afterwards, the bottom area of the user interface shows a window called “I/O Ports”. Unfold “leds” and “Scalar ports”, as shown in Figure 22.

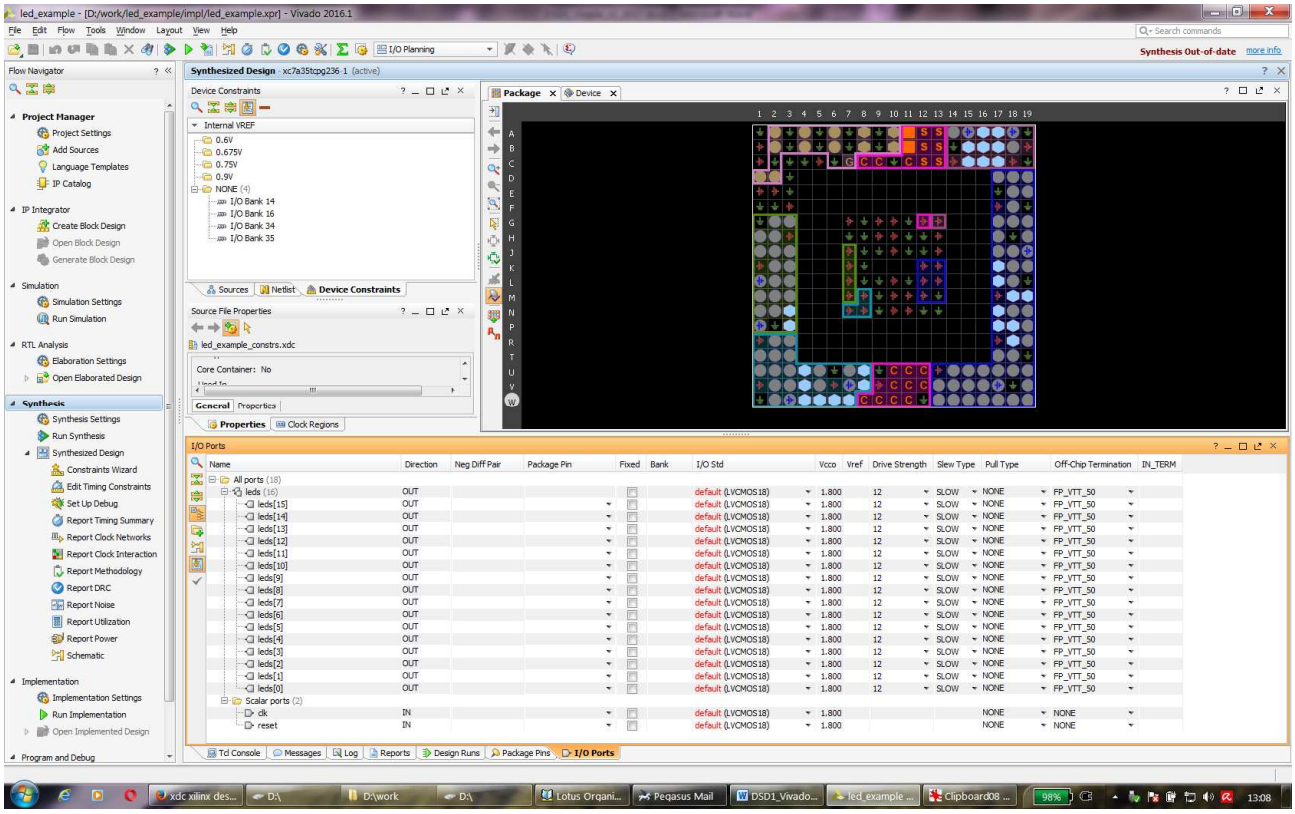
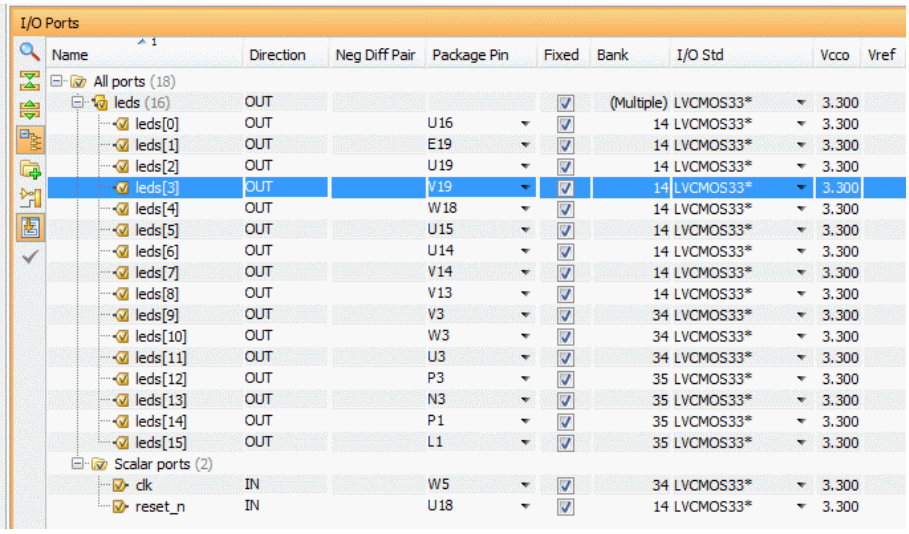


Figure 22: Assigning IO pins

For all signals select the required pin (column: “Package Pin”), and the required “I/O Std”, as shown in Figure 23. The pin names of the LEDs, the reset button and the clock oscillator can be found in the board schematics (see board documentation on the CIS website). They are also printed on the Basys3 board.



I/O Ports

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref
All ports (18)								
leds (16)	OUT			☑	(Multiple)	LVCMOS33*	3.300	
leds[0]	OUT		U16	☑	14	LVCMOS33*	3.300	
leds[1]	OUT		E19	☑	14	LVCMOS33*	3.300	
leds[2]	OUT		U19	☑	14	LVCMOS33*	3.300	
leds[3]	OUT		V19	☑	14	LVCMOS33*	3.300	
leds[4]	OUT		W18	☑	14	LVCMOS33*	3.300	
leds[5]	OUT		U15	☑	14	LVCMOS33*	3.300	
leds[6]	OUT		U14	☑	14	LVCMOS33*	3.300	
leds[7]	OUT		V14	☑	14	LVCMOS33*	3.300	
leds[8]	OUT		V13	☑	14	LVCMOS33*	3.300	
leds[9]	OUT		V3	☑	34	LVCMOS33*	3.300	
leds[10]	OUT		W3	☑	34	LVCMOS33*	3.300	
leds[11]	OUT		U3	☑	34	LVCMOS33*	3.300	
leds[12]	OUT		P3	☑	35	LVCMOS33*	3.300	
leds[13]	OUT		N3	☑	35	LVCMOS33*	3.300	
leds[14]	OUT		P1	☑	35	LVCMOS33*	3.300	
leds[15]	OUT		L1	☑	35	LVCMOS33*	3.300	
Scalar ports (2)								
clk	IN		W5	☑	34	LVCMOS33*	3.300	
reset_n	IN		U18	☑	14	LVCMOS33*	3.300	

Figure 23: Selection of package pins and I/O standards

Next, Vivado must be configured to the correct configuration voltage, according to the schematics of the Basys3 board. From the menu bar choose “Tools -> Edit Device Properties”. In the section “Configuration” select “3.3” as “Conf. Voltage” and “VCCO” as “Configuration Bank Voltage Selection”, see Figure 24. Click “OK”. In order to save the changes into the constraint file, use the shortcut “CTRL + S” or choose “File -> Save Constraints” in the menu bar. Close the synthesized design by clicking “File -> Close Synthesized Design” afterwards.

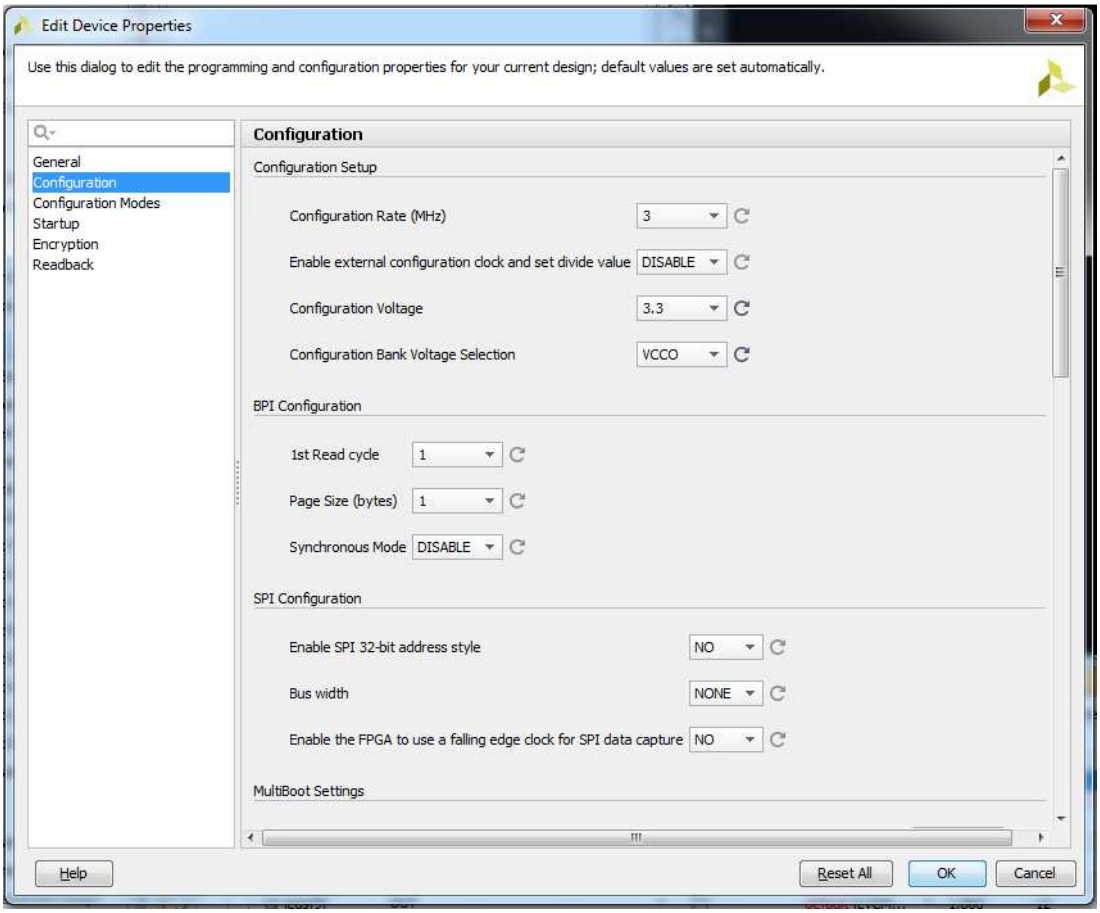


Figure 24: Selection of the configuration voltage

All configurations have now been set. Click “Generate Bitstream” in the Flow Navigator in order to generate the FPGA bitstream which will be downloaded to the FPGA board. Click “Yes” when asked whether the synthesis and implementation should launch. Generating the bitstream will take some minutes of time. If bitstream generation is finished, click “Cancel” in the pop-up window. Now, the project summary is updated and information about the design will be displayed (Figure 25).

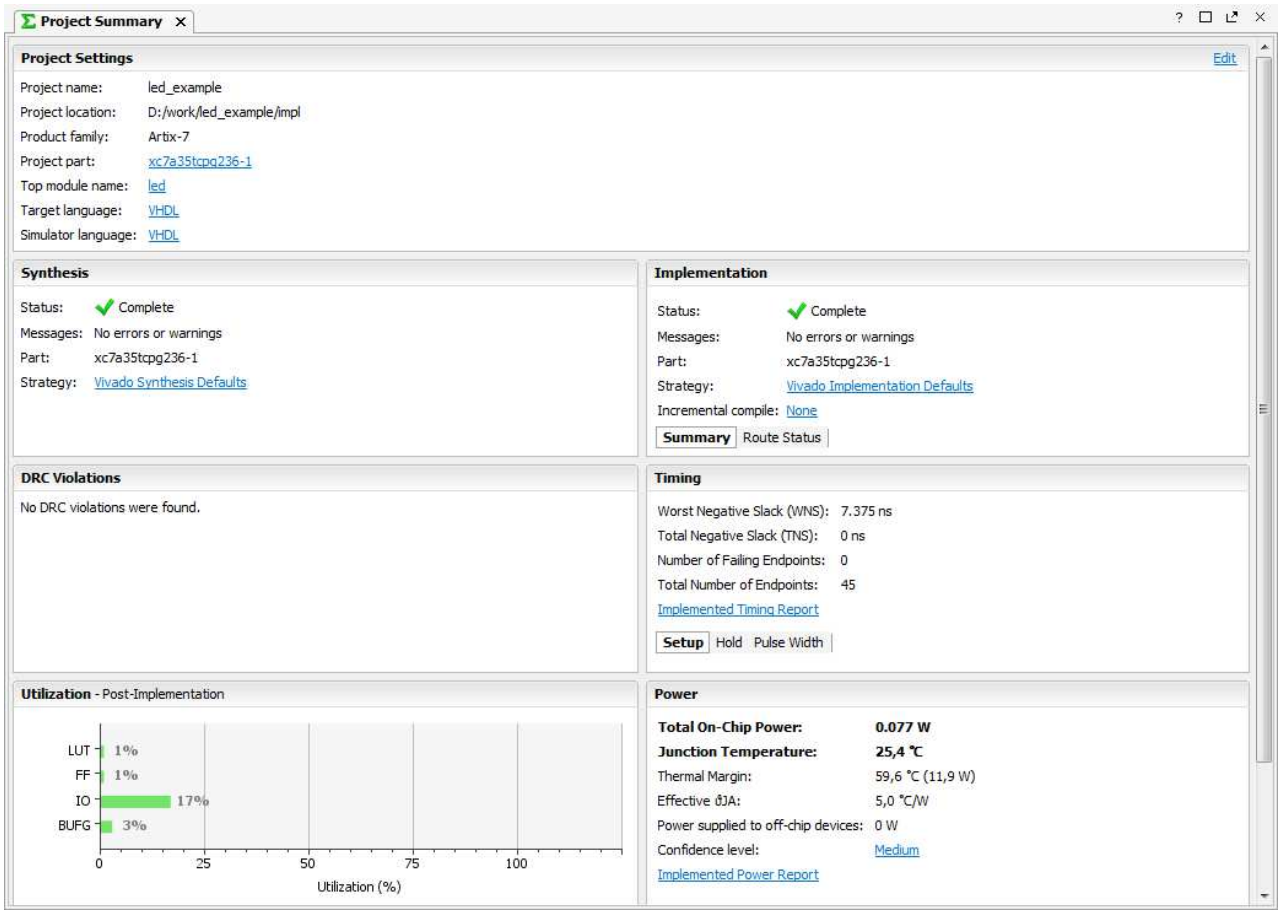


Figure 25: Project summary of the running LED example

As shown in Figure 25 the design compiles with “No errors or warnings”. The field “Utilization” shows the number of resources that are consumed by the current design. In this case, 1% of the combinatorial resources (LUT = Look Up Tables), 1% of the sequential resources (FF = Flip Flops) and 17% of the IOs that are available on the xc7a35tcpg236-1 FPGA are needed to implement the running LED design example. Another important value is the “Worst Negative Slack” shown in the field “Timing”. This value indicates the timing margin of the design. The larger the slack is, the less critical is it for Vivado to meet the desired clock frequency of the design (100 MHz for the running LED example, see Figure 21). A negative value would indicate that the desired clock frequency is too high and that the design will not run on the FPGA (for example, if you would enter a target frequency of 500 MHz instead of 100 MHz, you would see a “Timing Violation” warning after bitstream generation). More information on timing will be given in the lecture “Digital System Design 2”.

For downloading the bitstream to the FPGA board, make sure that the jumpers JP2 and J6 are set as displayed in Figure 26 (JP2 must be set to “USB”; no jumper must be inserted into J6). The setting of jumper JP1 doesn’t care.

Figure 26: Jumper setting for Basys3 power supply via USB[2]

If the jumpers are set properly, power on the Basys3 board via the switch located near the power jumper (see Figure 26).

In order to download the bitstream to the FPGA, open the “Hardware Manager” located in the Flow Navigator and click “Open Target” -> “Auto Connect” as shown in Figure 27.

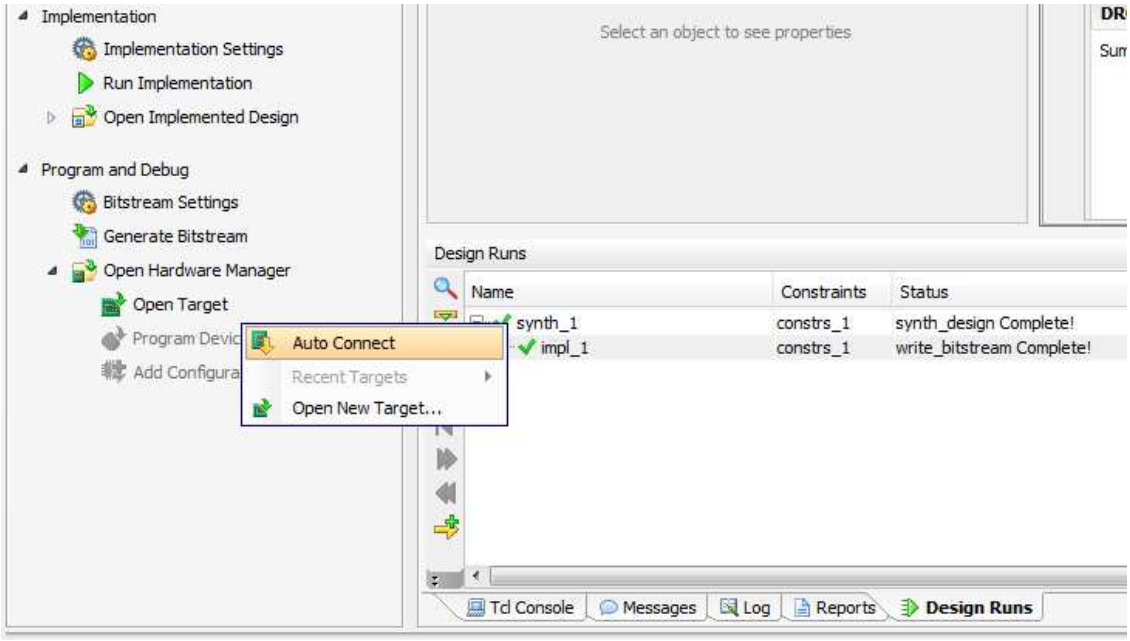


Figure 27: Hardware manager

Now, click on “Program Device” below “Open Target” and select the device as shown in Figure 28.

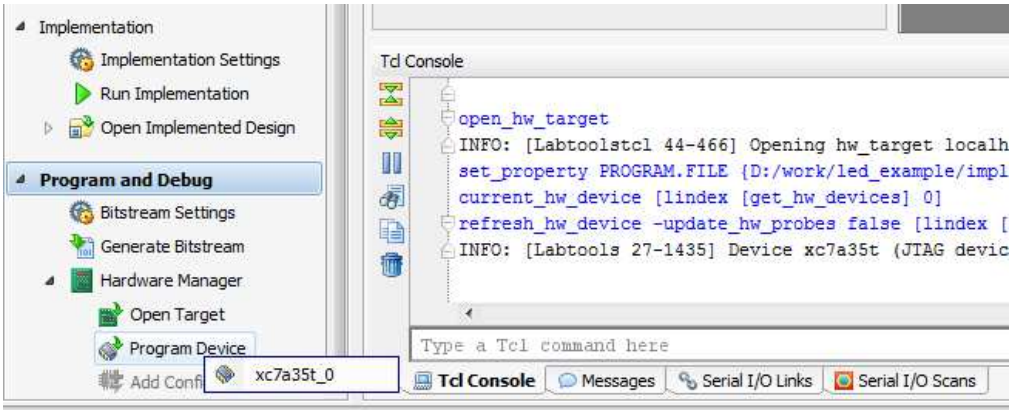


Figure 28: Selection of device

A pop-up window will appear. The path to the bitstream file is already selected. Click on “Program”. Finally, the running LED example shall perform on the Basys3 board. Check whether the reset button (button BTNC) also operates as expected.

Appendix: Entering Design Constraints without the GUI

Entering design constraints (clock frequency, IO pins, configuration voltage …) over the GUI (as shown in Figure 21 to Figure 24) is a very intuitive way and well suited for beginners. However, for designs with a many IOs this method quickly becomes tedious. Therefore, another method to enter design constraints is presented in the following. This method is especially of extreme help if you want to reuse design constraints from existing projects.

For the following example it is assumed that you have implemented the running LED example (as described previously) which resides in the directory

d:/work/led_example/impl

Furthermore, it is assumed that we want to create a modified version of this Vivado project called “led_example2”. In this project we want to make use of the same IO pins as in the project “led_example” as well as an additional IO pin which connects to the switch SW0 contained on the Basys3 board. Moreover, it is assumed that the signals “clk” and “reset” of the project “led_example” have been renamed to “clock” and “btnc” in the VHDL code of the project “led_example2”.

First, create a new Xilinx Vivado project, for example, in the directory

d:/work/led_example2/impl

by following the instructions described in Figure 9 to Figure 17 (define the project name, project location as well as the FPGA device and add all VHDL source files). Next, create a directory called

d:/work/led_example2/impl/led_example.constrs/

and copy the file

d:/work/led_example/impl/led_example.srcs/constrs_1/new/led_example_constrs.xdc

from the project “led_example” to the sub-directory of the project “led_example2” created previously. Finally, rename the file to

d:/work/led_example2/impl/led_example.constrs/led_example2_constrs.xdc

Open this file with any text editor. The XDC (Xilinx Design Constraints) file contains all constraints of a Vivado project, like the desired clock frequency, the IO pins, configuration voltage etc. Modify/add the following lines in the XDC file which are highlighted in red:

```
create_clock -period 10.000 -name clock -waveform {0.000 5.000} [get_ports clock]
set_property PACKAGE_PIN U16 [get_ports {leds[0]}]
set_property PACKAGE_PIN E19 [get_ports {leds[1]}]
set_property PACKAGE_PIN U19 [get_ports {leds[2]}]
set_property PACKAGE_PIN V19 [get_ports {leds[3]}]
set_property PACKAGE_PIN W18 [get_ports {leds[4]}]
set_property PACKAGE_PIN U15 [get_ports {leds[5]}]
set_property PACKAGE_PIN U14 [get_ports {leds[6]}]
set_property PACKAGE_PIN V14 [get_ports {leds[7]}]
set_property PACKAGE_PIN V13 [get_ports {leds[8]}]
set_property PACKAGE_PIN V3 [get_ports {leds[9]}]
set_property PACKAGE_PIN W3 [get_ports {leds[10]}]
set_property PACKAGE_PIN U3 [get_ports {leds[11]}]
set_property PACKAGE_PIN P3 [get_ports {leds[12]}]
set_property PACKAGE_PIN N3 [get_ports {leds[13]}]
set_property PACKAGE_PIN P1 [get_ports {leds[14]}]
set_property PACKAGE_PIN L1 [get_ports {leds[15]}]
set_property PACKAGE_PIN W5 [get_ports clock]
set_property PACKAGE_PIN U18 [get_ports btnc]
set_property PACKAGE_PIN V17 [get_ports sw0]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[12]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {leds[13]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[14]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds[15]}]
set_property IOSTANDARD LVCMOS33 [get_ports clock]
set_property IOSTANDARD LVCMOS33 [get_ports btnc]
set_property IOSTANDARD LVCMOS33 [get_ports sw0]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

Proceed with Figure 17, select “Add or create constraints” and click “Next”. At this point **do not click “Create File” (as shown in Figure 18)** but click “Add Files”, select the XDC file

d:/work/led_example2/impl/led_example.constrs/led_example**2**_constrs.xdc

and click the “Finish” button.

Afterwards you can immediately generate the bitstream as described above (follow instructions after Figure 24).

Abbreviations

VHDL	Very High Speed Integrated Circuit Hardware Description Language
FPGA	Field Programmable Gate Array
XDC	Xilinx Design Constraints
CAD	Computer Aided Design

Additional literature

Half of the knowledge needed for the design of digital systems is CAD tool knowledge. In this course, only very basic features of the Vivado implementation software can be presented. For a more detailed insight into the different steps of the implementation flow, it is recommended to use the documentation of the Vivado software (menu: “Help > Documentation and Tutorials”).

Questions

Take a look at the different steps that have been run automatically by the flow. Compare these steps with the distance learning letter “IC Design Flow”. Take a look at the different files generated in the flow (can be found in the project directory you selected when using the New Project Wizard). Look at the different reports that have been generated. Can you determine how many resources of the FPGA are used for the design? Try to open the synthesized design in the Flow Navigator by clicking on “Synthesis”. What can you see here?

Version

Version 0.1, 2006-08-21	Create document
Version 0.2, 2006-08-29	Update document
Version 0.3, 2010-08-20	Change document from Xilinx to Altera
Version 0.4, 2010-12-07	Correct some errors
Version 0.5, 2011-11-06	Document translated and updated
Version 0.6, 2012-05-03	Add appendix „Eingabe/Übernahme von Constraints ohne GUI“
Version 0.7, 2013-08-01	Update to Quartus II version, 13.0, SP1. Minor changes
Version 0.8, 2014-09-02	Minor changes
Version 0.9, 2015-08-31	Update template
Version 1.0, 2016-03-22	Change document from Altera to Xilinx
Version 1.1, 2016-03-31	Proofreading
Version 1.1, 2017-02-01	Comments concerning study program MES added. Clean directory structure added.

If you find errors or inconsistencies, please report them to the supervisors of this lecture via email. Thank you!