



Lab 1

- How to use the Vivado tool to program with VHDL
- Design and simulation of simple circuits

Prof. Juan Antonio Clemente

Welcome screen



Vivado 2019.1

File Flow Tools Window Help Q Quick Access

VIVADO
HLx Editions

XILINX

Quick Start

- Create Project >
- Open Project >
- Open Example Project >

Tasks

- Manage IP >
- Open Hardware Manager >
- Xilinx Tcl Store >

Learning Center

- Documentation and Tutorials >
- Quick Take Videos >
- Release Notes Guide >

Recent Projects

- project_2
C:/hlocal/project_2
- test_VIVADO
C:/hlocal/test_VIVADO

Recent projects will appear here

Tcl Console

Escribe aquí para buscar

11:53
06/09/2019

How to create a project



- Click on File->Project->New and then, Next. The following screen will appear

The 'New Project' dialog box is shown. It has a title bar with a yellow arrow icon and a close button. The main area is titled 'Project Name' and contains the instruction: 'Enter a name for your project and specify a directory where the project data files will be stored.' There are two input fields: 'Project name:' with the text 'project_3' and 'Project location:' with the text 'C:/hlocal'. Below these fields is a checkbox labeled 'Create project subdirectory' which is checked. Underneath the checkbox, it says 'Project will be created at: C:/hlocal/project_3'. At the bottom of the dialog, there are four buttons: a help button (question mark in a circle), '< Back', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'.

Name your project as you want
ALWAYS in C:/hlocal. DON'T USE
SPACES or other 'special
characters'!! (e.g. ñ, á, etc.)

How to create a project



New Project

Project Type
Specify the type of project to create.

- ☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☒ Do not specify sources at this time
- ☐ **Post-synthesis Project**
You will be able to add sources, view device resources, run design analysis and implementation.
☐ Do not specify sources at this time
- ☐ **I/O Planning Project**
Do not specify design sources. You will be able to view pin configurations.
- ☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.
- ☐ **Example Project**
Create a new Vivado project from a predefined template.

Leave the by-default option "RTL project"

Select the target FPGA



New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts **Boards**

[Reset All Filters](#) [Update Board Repositories](#)

Vendor: Name: Board Rev:

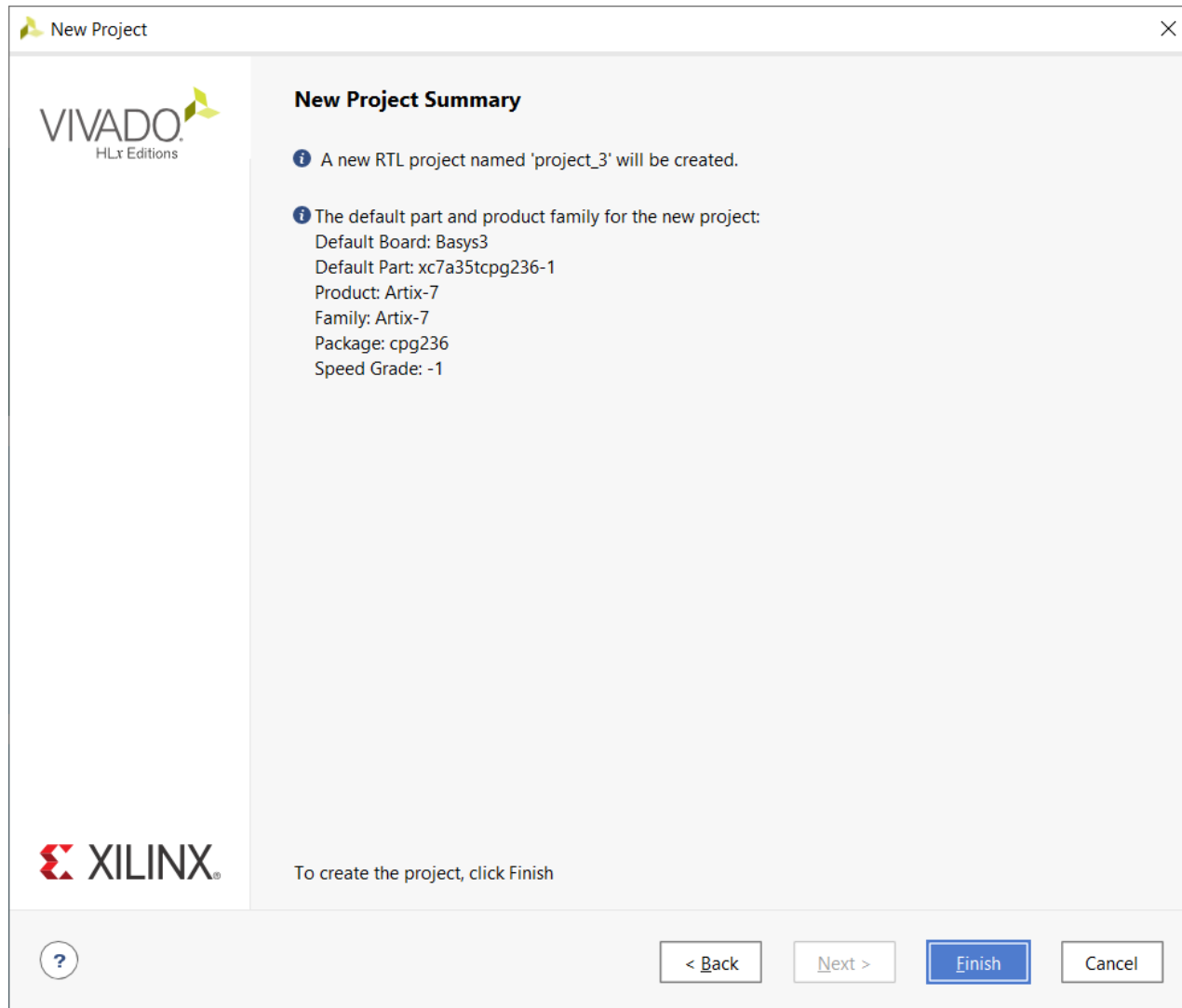
Search:

Display Name	Preview	Vendor	File Version	Part	I/O F
Basys3		digilentinc.com	1.1	xc7a35tcpg236-1	236

Click on "Boards", then select "digilentinc.com" as Vendor and "Basys3" as Name. Click here (to select the device) and then, click on "Next"

[?>](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Project summary



Voilà the main GUI of VIVADO



project_3 - [C:/hlocal/project_3/project_3.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Ready Default Layout

Flow Navigator PROJECT MANAGER - project_3

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

Sources

- Design Sources
- Constraints
- Simulation Sources
 - sim_1
- Utility Sources

Hierarchy Libraries Compile Order

Properties

Project Summary

Overview

Settings Edit

Project name:
Project location:
Product family:
Project part:
Top module name: Not defined
Target language: Verilog
Simulator language: Mixed

Board Part

Display name: Basys3
Board part name: diligentinc.com:basys3:part0:1.1
Board revision: C.0

Tcl Console

Name

synth_1
impl_1

TPWS Total Power Failed Routes LUT FF BRAMs URAM DSP Start Elapsed Run Strategy

Vivado Synthesis Defaults (Vivado Synthesis 2019)
Vivado Implementation Defaults (Vivado Implementation)

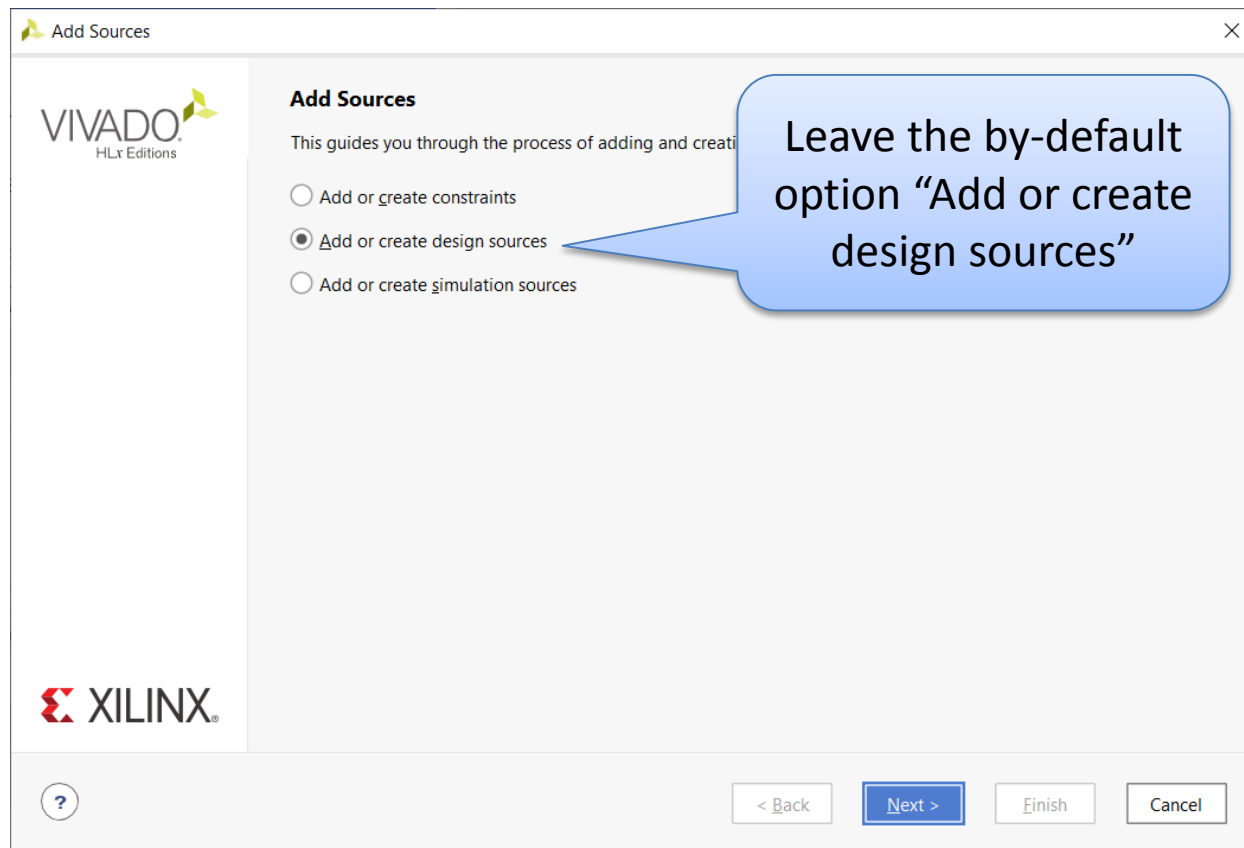
Project sources are here

Actions to be made with the project are here

Add or create new source files



- Click on “Add source” (top left), then...



Add or create new source files

Add Sources

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

	Index	Name	Library	Location
●	1	adder.vhd	xil_defaultlib	C:/Users/Juanan/Dropbox/Docencia/2019_2020/TOC-I/2.Laboratory/Lab1/vhdl/1-Intro_files
●	2	divider.vhd	xil_defaultlib	C:/Users/Juanan/Dropbox/Docencia/2019_2020/TOC-I/2.Laboratory/Lab1/vhdl/1-Intro_files

☐ Scan and add RTL include files into project

☒ Copy sources into project

☒ Add sources from selected directories

Buttons: Add Files, Add Directories, Create File

Navigation: < Back, Next >, Finish, Cancel

Help: ?

For this lab, we will use "Add Files". Then, select "adder" and "divider" (available in the Campus Virtual "1-Intro_files.rar")

By selecting "Add Directories", we add all the files of the selected directory

By selecting "Create File", we create a new file from scratch

VERY IMPORTANT: Keep the option "Copy sources into project" ACTIVATED

Add existing source files to our project

- The added files will be now visible in the Sources part:

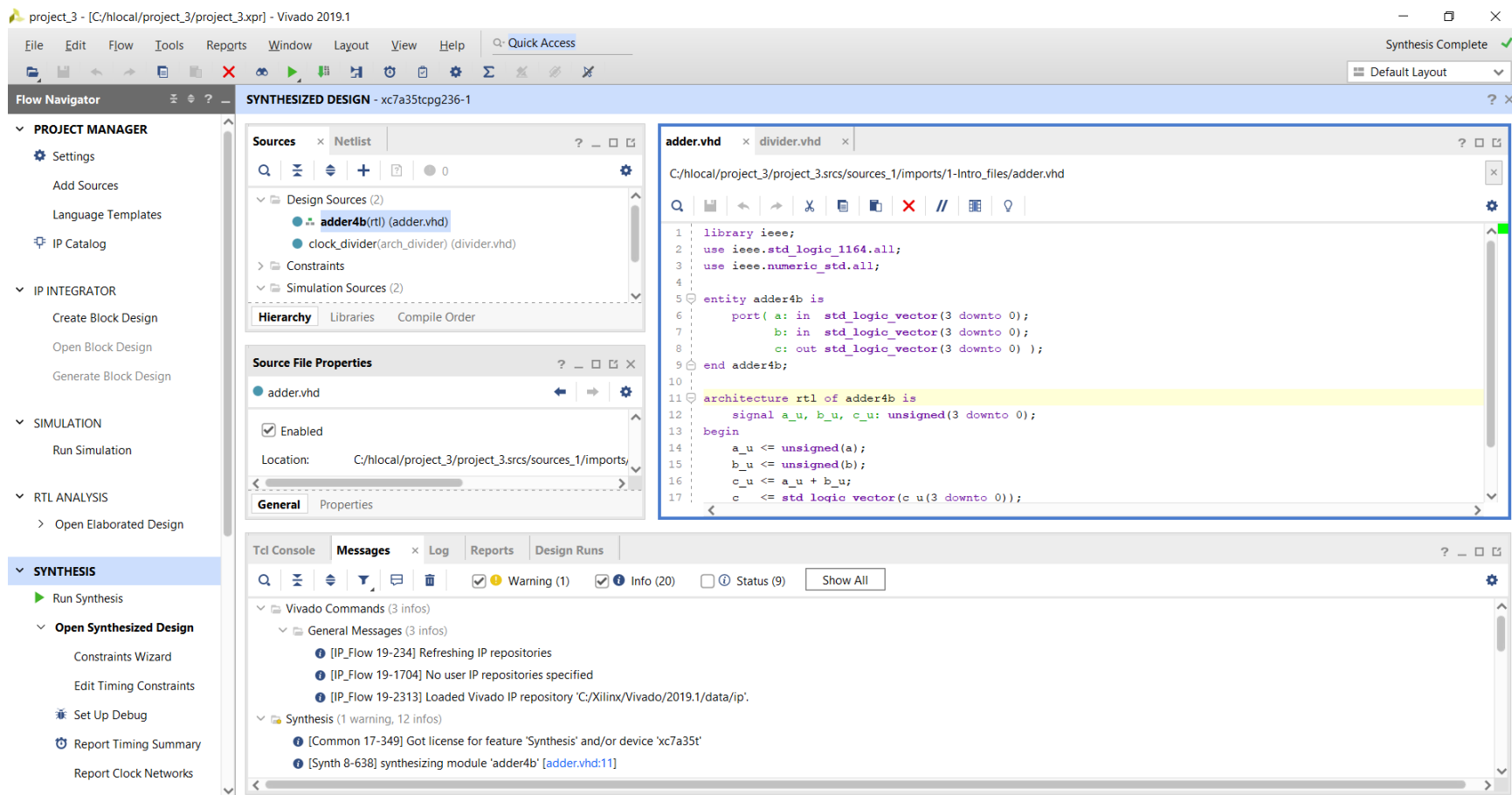
The screenshot shows the Vivado 2019.1 interface. The **Flow Navigator** on the left has the **PROJECT MANAGER** tab selected. The **Sources** panel in the center shows a hierarchy of files under **Design Sources (2)**: **adder4b(rtl) (adder.vhd)** and **clock_divider(arch_divider) (divider.vhd)**. The **adder4b(rtl) (adder.vhd)** file is highlighted with a blue box. A blue callout bubble points to this file with the text: "This symbol indicates that the adder.vhd file is the 'top file' in the hierarchy. This is the file on which the synthesis, implementation etc... will be made (discussed later)". Below the Sources panel, the **Source File Properties** dialog is open for **adder.vhd**. It shows the **Enabled** checkbox checked and the **Location** as **C:/hlocal/project_3/project_3.srcs/comp**. A blue callout bubble points to the **Set as Top** option in the **Source Node Properties...** context menu, with the text: "At any time, any other file can be the 'top file' by right-clicking on it, then selecting 'Set as Top'". The context menu also shows other options like **Open File**, **Replace File...**, **Copy File Into Project**, **Copy All Files Into Project**, **Remove File from Project...**, **Enable File**, **Disable File**, **Move to Simulation Sources**, **Move to Design Sources**, **Hierarchy Update**, **Refresh Hierarchy**, **IP Hierarchy**, **Set as Top**, **Set Global Include**, **Clear Global Include**, **Set as Out-of-Context for Synthesis...**, **Set Library...**, **Set File Type...**, **Set Used In...**, **Edit Constraints Sets...**, and **Edit Simulation Sets...**.



Add existing source files to our project



- By double-clicking on these files, one can see and edit the source code



Lab 1.a



This is a 4-bit adder (we will discuss this in detail in class)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity adder4b is
    port( a: in  std_logic_vector(3 downto 0);
          b: in  std_logic_vector(3 downto 0);
          c: out std_logic_vector(3 downto 0) );
end adder4b;

architecture rtl of adder4b is
    signal a_u, b_u, c_u: unsigned(3 downto 0);
begin
    a_u <= unsigned(a);
    b_u <= unsigned(b);
    c_u <= a_u + b_u;
    c   <= std_logic_vector(c_u(3 downto 0));
end rtl;
```



Synthesize the design

project_3 - [C:/hlocal/project_3/project_3.xpr] - Vivado 2019.1

File Edit Flow Tools Repgrts Window Layout View Help Quick Access

Flow Navigator SYNTHESIZED DESIGN - xc7a35tcbpg236-1

Open Block Design
Generate Block Design

SIMULATION
Run Simulation

RTL ANALYSIS
Open Elaborated Design

SYNTHESIS
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic

Sources x Netlist
Design Sources (2)
adder4b(rtl) (adder.vhd)

Source File Properties
adder.vhd
Enabled
Location: C:/hlocal/project_3/project_3.srscs/sources_1/imports/
General Properties

adder.vhd
C:/hlocal/project_3/project_3.srscs/sources_1/imports/1-Intro_files/adder.vhd
1: library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity adder4b is
port (a: in std_logic_vector(3 downto 0);
b: in std_logic_vector(3 downto 0);
c: out std_logic_vector(3 downto 0));
end adder4b;
10:
11: architecture rtl of adder4b is
12: signal a_u, b_u, c_u: unsigned(3 downto 0);
13: begin
14: a_u <= unsigned(a);
15: b_u <= unsigned(b);
16: c_u <= a_u + b_u;
17: c <= std_logic_vector(c_u(3 downto 0));

Tcl Console Messages x Log Reports Design Runs
Warning (1) Info (20) Status (9) Show All
Vivado Commands (3 infos)
General Messages (3 infos)
[IP_Flow 19-234] Refreshing
[IP_Flow 19-1704] No user IP
[IP_Flow 19-234]
Synthesis (1 warning)
[Common 17-349]
[Synth 8-638] synt

In the Flow Navigator tab, click on Run Synthesis

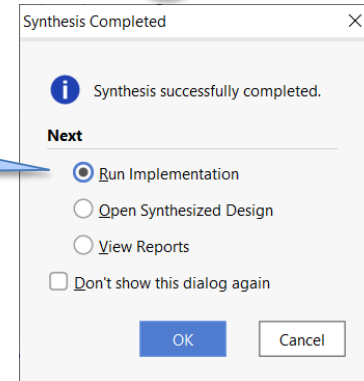
In the Log tab, we can see the progress of the task

10:0 Insert VHDL

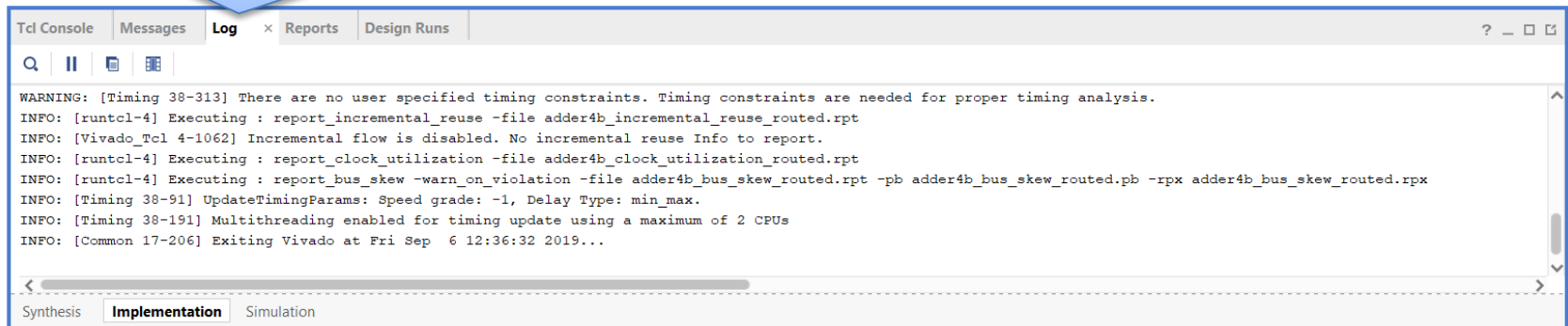


Synthesize the design

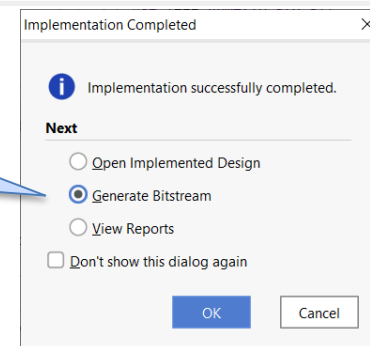
Once Synthesis is completed, the following window appears. We click on Run Implementation, then Next.



In the Log tab, you will see that there are separate messages for the Synthesis, Implementation and Simulation processes!



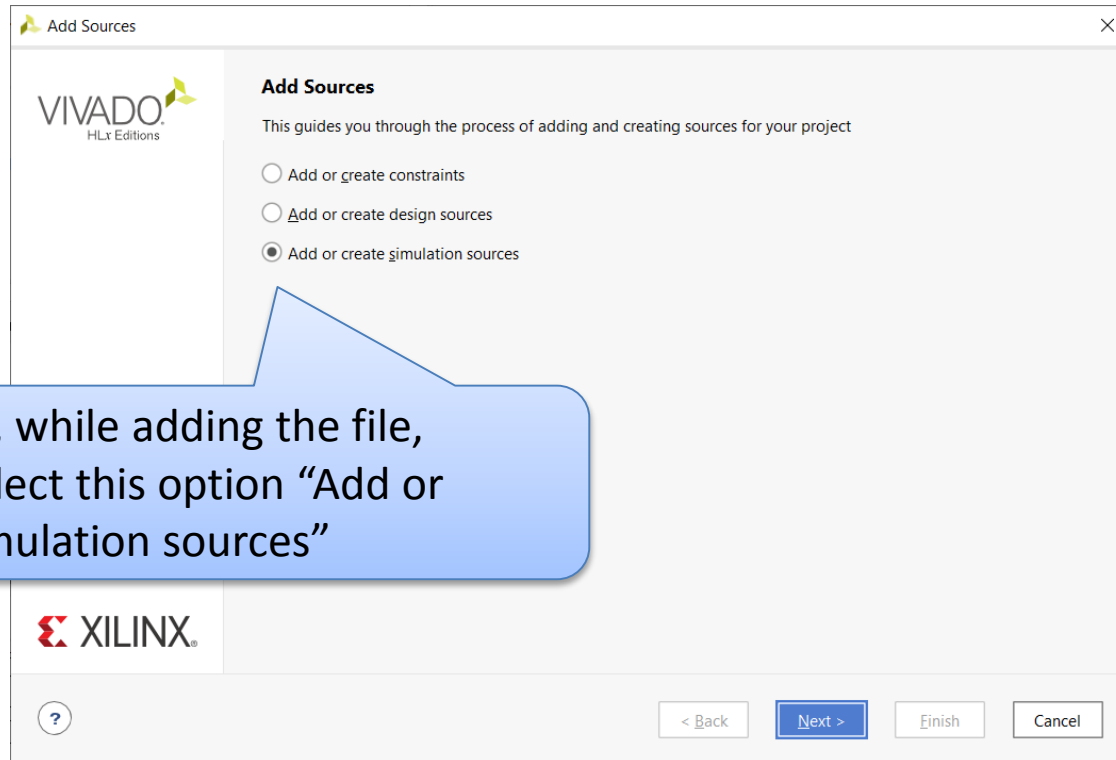
Once Implementation is completed, the following window appears. If we click on Generate Bitstream, IT WILL FAIL (a constraints file is needed). We just open implemented design



Check that the design is correct



- In order to check if a design is correct, you must simulate the circuit:
 - Add the testbench that is available in the Campus Virtual “simu_1a.vhd”.



This time, while adding the file, please select this option “Add or create simulation sources”

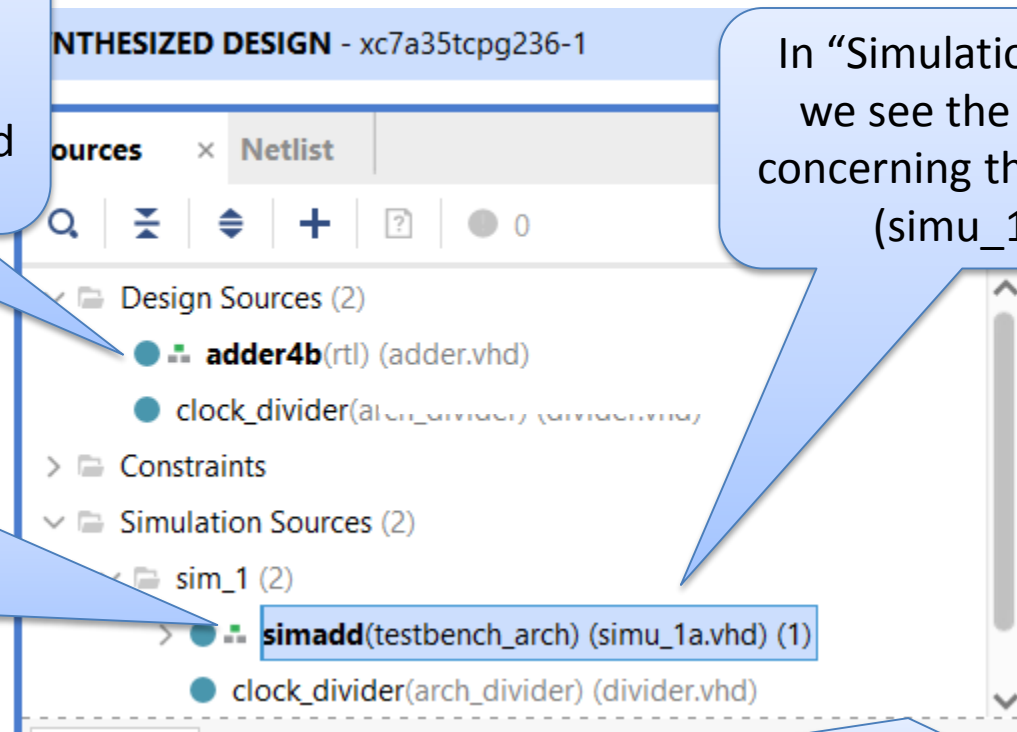


Simulation testbench

- Once added, we can see it in the main VIVADO GUI

In “Design Sources”, we see the source files concerning the design (adder.vhd and divider.vhd)

This symbol also appears in one of the simulation files. The file with the symbol will be the one that is simulated



In “Simulation Sources”, we see the source file concerning the simulation (simu_1a.vhd)

divider.vhd appears in “Design Sources” and “Simulation Sources” because it is still not part of the design hierarchy (it is not used from adder.vhd). **We will fix this later.**

Simulation testbench



- POSSIBLE PROBLEM: You have screwed it up and a simulation source has been included as a design source, or viceversa.
 - SOLUTION: At any time, you can right-click on the involved file and select “Move to Simulation Sources” or “Move to Design Sources” to fix this problem.

Simulation testbench (check the Campus Virtual)



```
-- We add the libraries needed
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Entity declaration
ENTITY simadd IS
END simadd;

-- Architecture
ARCHITECTURE testbench_arch OF simadd IS
    -- Component declaration
    COMPONENT adder4b
    PORT (
        A : IN    std_logic_vector(3 downto 0);
        B : IN    std_logic_vector(3 downto 0);
        C : OUT   std_logic_vector(3 downto 0)
    );
    END COMPONENT;

-- Inputs
signal A : std_logic_vector(3 downto 0) := (others => '0');
signal B : std_logic_vector(3 downto 0) := (others => '0');

-- Outputs
signal C : std_logic_vector(3 downto 0);
```

...

Simulation testbench (check the Campus Virtual)



```
BEGIN
-- Instantiation of the unit under test
uut: adder4b PORT MAP (
    A => A,
    B => B,
    C => C );
-- Stimuli process
stim_proc: process
begin
    A <= "0000";      B <= "0000";
    wait for 100 ns;
    A <= "0101";      B <= "0100";
    wait for 100 ns;
    A <= "0000";      B <= "0111";
    wait for 100 ns;
    A <= "0011";      B <= "1000";
    wait for 100 ns;
    A <= "1011";      B <= "1111";
    wait for 100 ns;
    A <= "1001";      B <= "0110";
    wait;
end process;

END testbench_arch;
```

Simulation



project_3 - [C:/hlocal/project_3/project_3.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Run Help Quick Access

Flow Navigator

- PROJECT MANAGER
 - Settings
 - Add Sources
 - Language Templates
 - IP Catalog
- IP INTEGRATOR
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- SIMULATION**
 - Run Simulation
- RTL ANALYSIS
 - Open Elaborated Design
- SYNTHESIS
 - Run Synthesis
 - Open Synthesized Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Set Up Debug
 - Report Timing Summary
 - Report Clock Networks

SIMULATION - Behavioral Simulation - Functional - sim_1 - simadd

Scope Sources

Name	Design Unit	Block T...
simadd	simadd(testbench_arch	VHDL E
uut	adder4b(rtl)	VHDL E

Objects Protocol Instance

Name	Value	Data T...
A[3:0]	9	Array
B[3:0]	6	Array
C[3:0]	f	Array

adder.vhd divider.vhd simu_1a.vhd Untitled 1

Name	Value	999,996 ps	999,998 ps	1,000,000 ps
A[3:0]	9	9	9	9
B[3:0]	6	6	6	6
C[3:0]	f	f	f	f

(MB): peak = 1811.117 ; gain = 4.730

Type a Tcl command here

write_bitstream ERROR

Default Layout

3- We can expand the size of this window by clicking on this symbol

1- In the SIMULATION part, click on "Run Simulation", then "Run Behavioral Simulation". We will discuss the other types of simulation in class

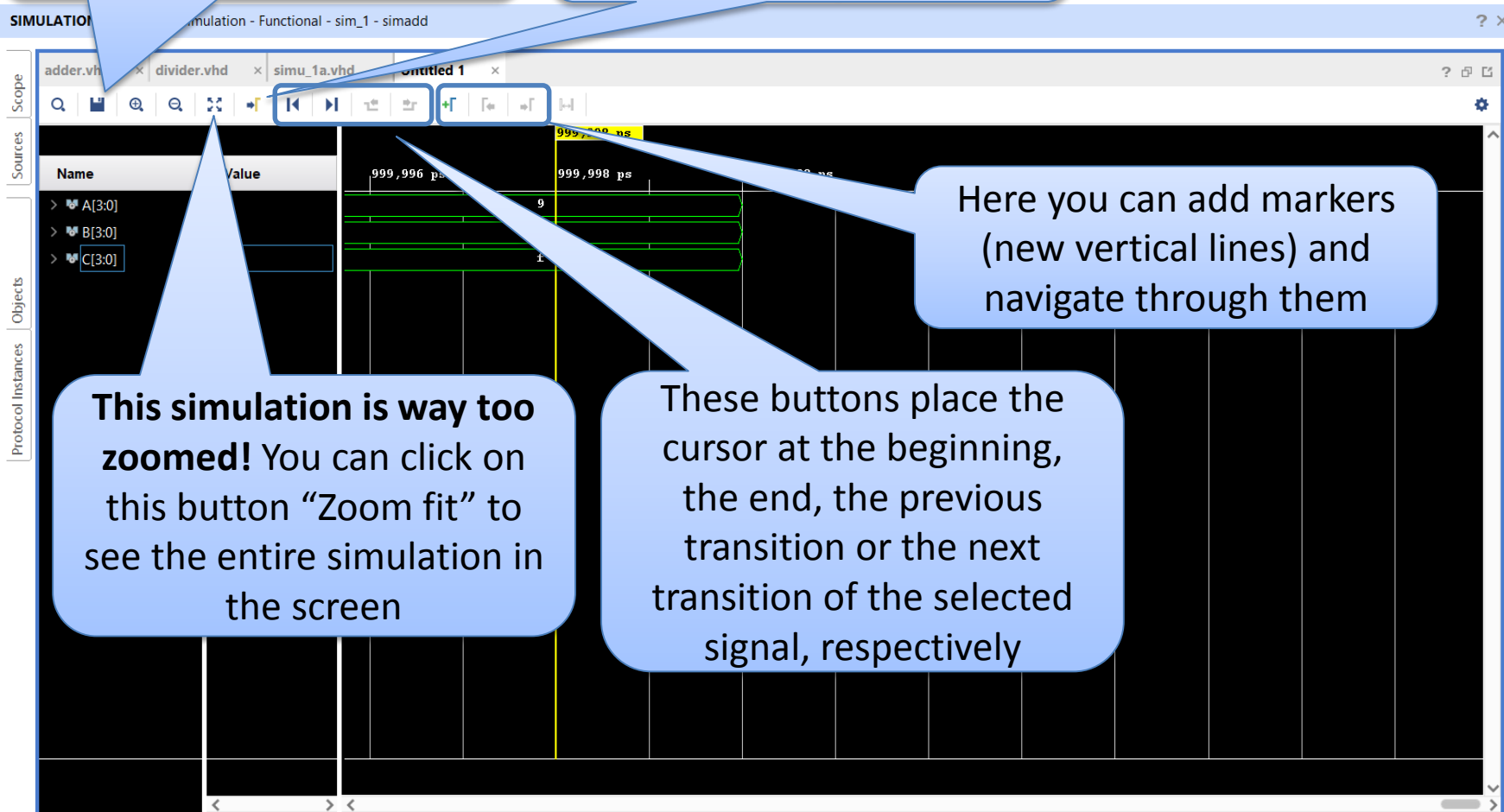
2- The result of the simulation will be displayed in this window



Simulation window

The waveform can be saved as a .wcfg file

This centers the view on the “cursor” (yellow vertical line)





Simulation

project_3 - [C:/hlocal/project_3/project_3.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Run Help Quick Access

Flow Navigator PROJECT MANAGER Settings Add Sources Language Templates IP Catalog IP INTEGRATOR Create Block Design Open Block Design Generate Block Design SIMULATION Run Synthesis Open Synthesized Design Constraints Wizard Edit Timing Constraints Set Up Debug Report Timing Summary Report Clock Networks

SIMULATION - Behavioral Simulation Functional - sim_1 - simadd

adder.vhd divider.vhd simu_1.vhd simadd_behav.wcfg

Name Value

Name	Value
A[3:0]	5 0
B[3:0]	4 7
C[3:0]	9 7

100 ns 200 ns 800 ns 900 ns

You can click here to restart the simulation

You can click here to run the entire testbench

You can click here to simulate for a specified period of time (in this case, 10 us)

Tcl Console Messages Log

Sim Time: 10500 ns

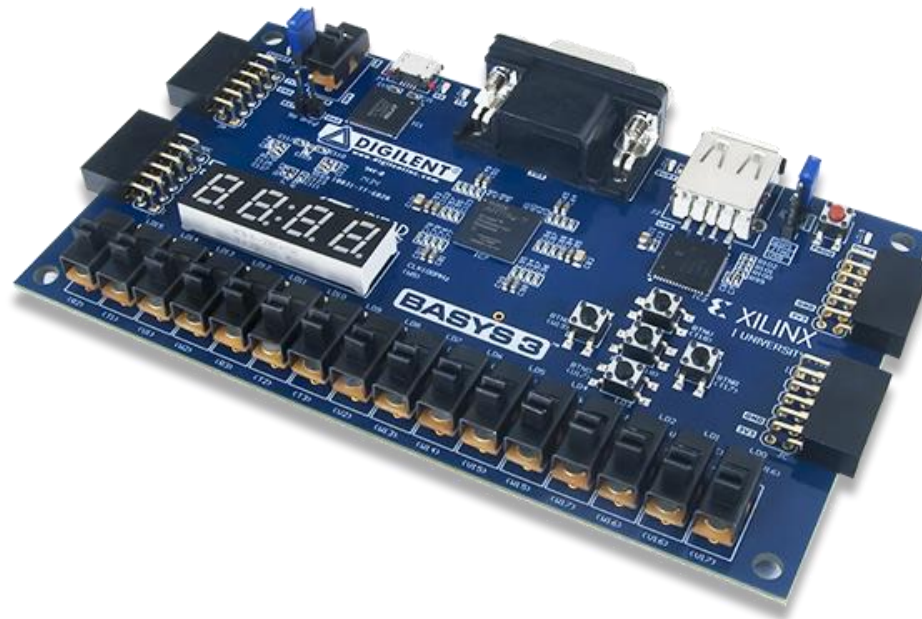
Implementation



- We can check that the circuit work the *real world*
 - The circuit will be implemented on an FPGA
 - The implementation tool needs to know where the inputs come from (push buttons or switches) and where to display the outputs (LEDs or 7-segment displays).
 - A constraints file (with extension .xdc) must be added with all this information.

Implementation

- This is the FPGA that we will use in all the labs:



<https://reference.digilentinc.com/reference/programmable-logic/basys-3/start>



Implementation

- .xdc file
 - You can find a complete .xdc file in the Campus Virtual of this particular board (downloaded from <https://reference.digilentinc.com/reference/programmable-logic/basys-3/start> → Master XDC files).
- This is a general file that allows associating your inputs and outputs with the physical IO devices of the board (LEDs, buttons, etc...).
- All the code in it is written on comments. You can uncomment the desired lines to use the IO devices that you need.
 - Each IO device of the board is associated with 2 lines starting with *set_property*.
- For instance, to use a switch for input pin A[0] (*adder.vhd*), it would be as follows:

```
Project Summary x pins.xdc * x
C:/hlocal/project_3/project_3.srscs/constrs_1/imports/1-Intro_files/pins.xdc

4  ## - rename the used ports (in each line, after get_ports) according
5
6  ## Clock signal
7  #set_property PACKAGE_PIN W5 [get_ports clk]
8  #set_property IOSTANDARD LVCMOS33 [get_ports clk]
9  #create_clock -add -name sys_olk_pin -period 10.00 -waveform [get_ports clk]
10
11 ## Switches
12 set_property PACKAGE_PIN V17 [get_ports {A[0]}]
13 set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
14 #set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
15 #set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
16 #set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
```

Uncomment these two lines and make sure that A[0] is written in the file (instead of sw[0])

Implementation



A and B (2 4-bit inputs) are associated to 8 switches like this

C (a 4-bit input) is associated to 4 LEDs like this

```
## LEDs
set_property PACKAGE_PIN U16 [get_ports {C[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C[0]}]
set_property PACKAGE_PIN E19 [get_ports {C[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C[1]}]
set_property PACKAGE_PIN U19 [get_ports {C[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C[2]}]
set_property PACKAGE_PIN V19 [get_ports {C[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C[3]}]
```

Project Summary x pins.xdc * x

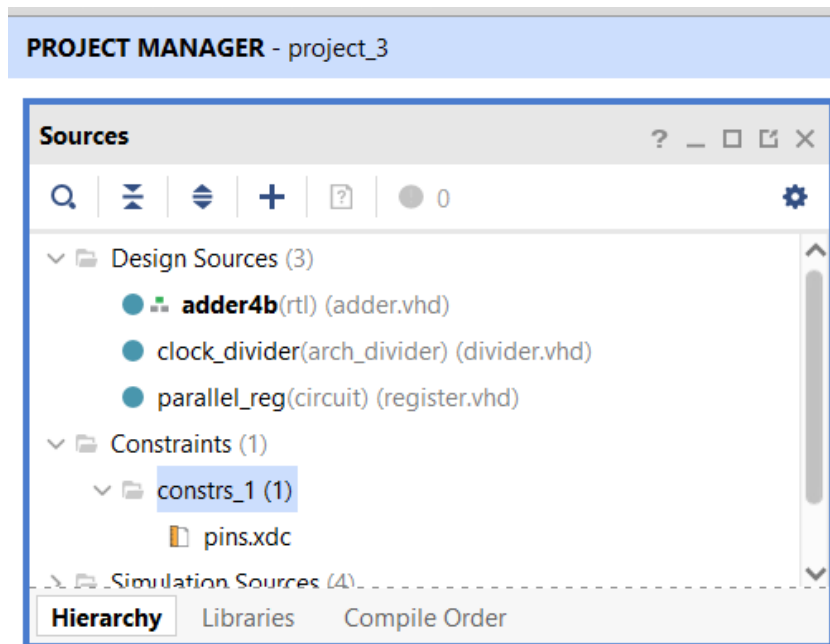
C:/hlocal/project_3/project_3.srscs/constrs_1/imports/1-Intro_files/pins.xdc

```
7  #set_property PACKAGE_PIN W5 [get_ports clk]
8      #set_property IOSTANDARD LVCMOS33 [get_ports clk]
9      #create_clock -add -name sys_clk_pin -period 10.00 -wav
10
11 ## Switches
12 set_property PACKAGE_PIN V17 [get_ports {A[0]}]
13     set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
14 set_property PACKAGE_PIN V16 [get_ports {A[1]}]
15     set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
16 set_property PACKAGE_PIN W16 [get_ports {A[2]}]
17     set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
18 set_property PACKAGE_PIN W17 [get_ports {A[3]}]
19     set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
20 set_property PACKAGE_PIN W15 [get_ports {B[4]}]
21     set_property IOSTANDARD LVCMOS33 [get_ports {B[4]}]
22 set_property PACKAGE_PIN V15 [get_ports {B[5]}]
23     set_property IOSTANDARD LVCMOS33 [get_ports {B[5]}]
24 set_property PACKAGE_PIN W14 [get_ports {B[6]}]
25     set_property IOSTANDARD LVCMOS33 [get_ports {B[6]}]
26 set_property PACKAGE_PIN W13 [get_ports {B[7]}]
27     set_property IOSTANDARD LVCMOS33 [get_ports {B[7]}]
28 #set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
29     #set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
30 #set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
```



Implementation

- The *pins.xdc* file is added to the project by clicking in “Add Sources” → “Add or Create Constraints”
- The file will be visible in the sources panel of the VIVADO GUI.

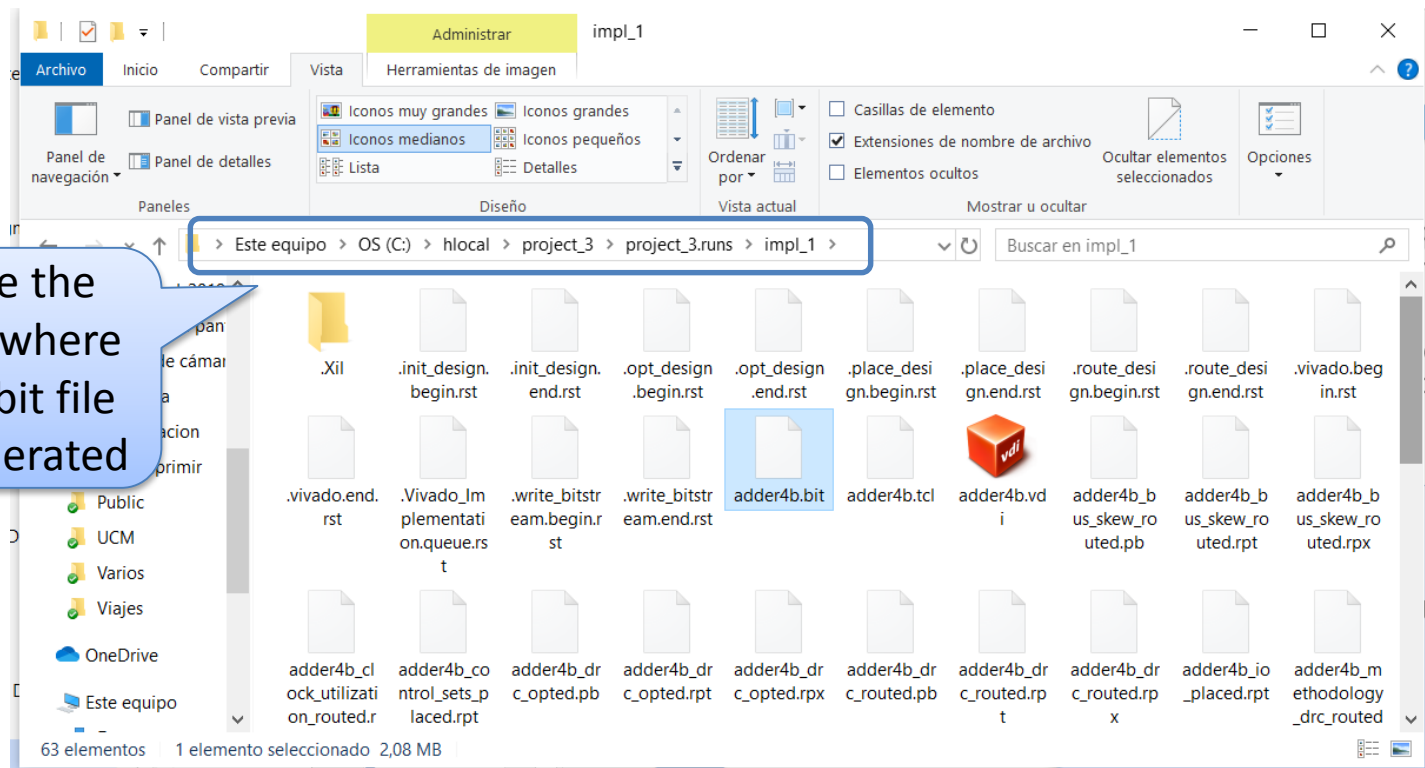




Implementation

- The last steps are clicking on the “Run Implementation” option (bottom left of the main GUI) and finally, “Generate bitstream”. When it is done, click on “Open Hardware manager”
 - This last step will generate an implementation file (*top_entity_name.bit*), which can be used to program the FPGA.

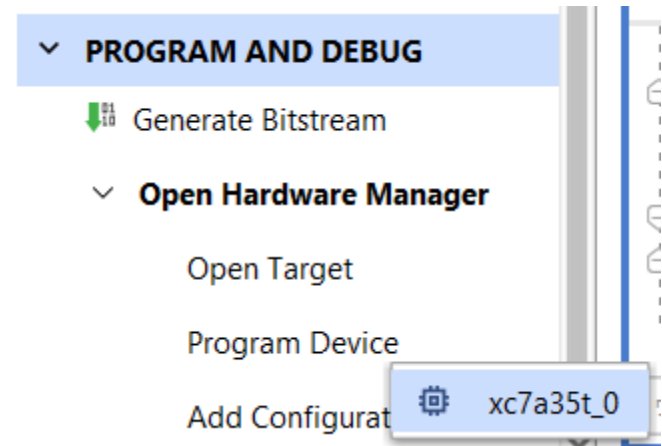
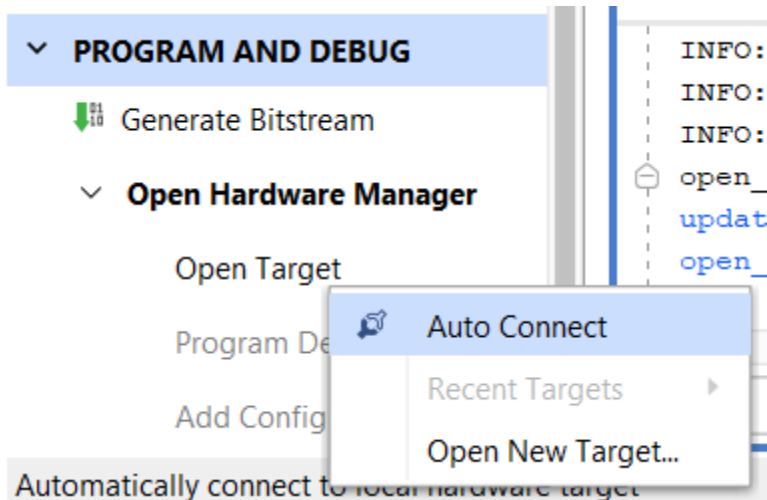
Note the path where the .bit file is generated



Download .bit on the FPGA



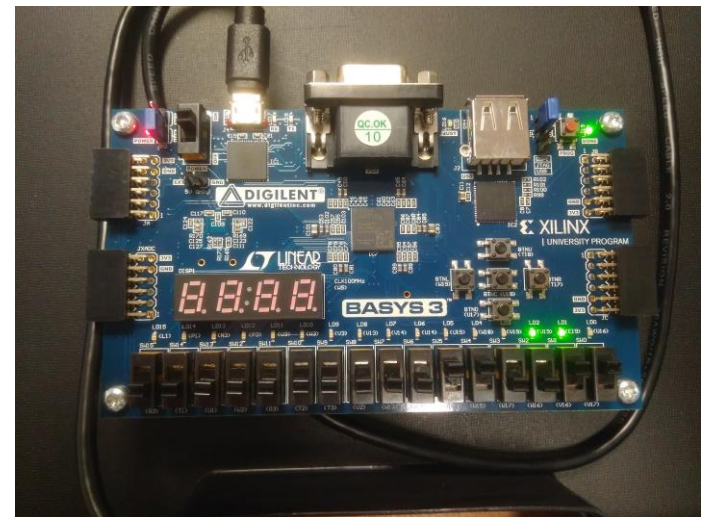
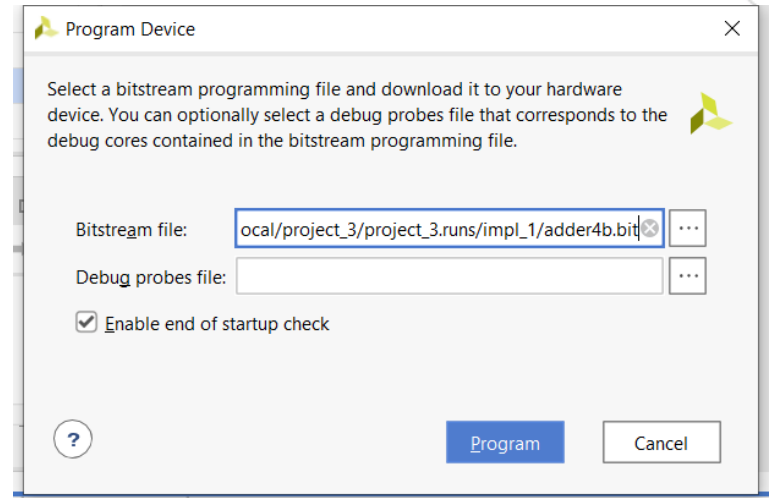
1. Borrow the FPGA to the lab assistants.
2. Connect the USB cable to one of the USB ports of the PC and make sure that the POWER switch is in position ON.
 - The device will be programmed with a by-default .bit file (which makes the 7-segment displays to show numbers from 0 to 9).
 - DON'T STEAL THE USB CABLE (other students are going to need it).
3. If you didn't open the "Open Hardware Manager", click on Target → Auto Connect
4. Anyway, click on Program Device → xc7a35t_0



Download .bit on the FPGA



- Finally, a .bit file must be provided (the .bit file of the open project is selected by default). Click on “Program”.
- The bitstream will be downloaded to the FPGA and you can use it as expected (try to move SW7-SW0 to see how the result of the addition is made visible on LD3-LD0).
- To go back to edit a source file, click on “PROJECT MANAGER” (top left of the main VIVADO window).



Lab 1.b



- Simulate and implement a register with parallel input / parallel output.
- We are going to program two versions of it:
 - WITHOUT frequency divider (simulation).
 - WITH frequency divider (implementation).

Lab 1.b



-- register with parallel input / parallel output

```
library IEEE;
use IEEE.std_logic_1164.all;

entity parallel_reg is
    port( rst, clk, load: in  std_logic;
          I:          in  std_logic_vector(3 downto 0);
          O:          out std_logic_vector(3 downto 0) );
end parallel_reg;
```

File register.vhd

Lab 1.b



architecture circuit of parallel_reg is
begin

```
    process(rst, clk)
    begin
        if clk'event and clk = '1' then
            if rst = '1' then
                0 <= "0000";
            elsif load = '1' then
                0 <= I;
            end if;
        end if;
    end process;
end circuit;
```

File register.vhd



Implementation Lab 1.b

- Clock pin:

Uncomment these three lines (by deleting the #'s)

Make sure that your clock signal is named exactly as it appears here (in this case, clk)

```
## Clock signal  
set_property PACKAGE_PIN W5 [get_ports clk]  
set_property IOSTANDARD LVCMOS33 [get_ports clk]  
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
```

This line indicates that clk is a clock signal with a period of 10.00 ns (the oscillator in pin W5 generates such a clock signal). The CAD tool will place and route it accordingly to optimize the Static Timing Analysis (STA, this concept is discussed in Lesson 2)

- Use the clock divider to feed a 1HZ clock to the register in order to be aware (as humans) of the transitions.
- In addition, make sure that you assign 6 switches (1 for reset, 1 for load, and 4 for input I) and 4 LEDs (for output O).

Grading



- Follow the instructions of the slides to simulate and implement the designs of Labs 1.a and 1.b.
- You have to show me these two parts working on the FPGA (don't forget the clock divider).
- There will be an EXTRA part (0.15 pts).