Lab 16: Using IP Cores with Vivado

In this lab, we learn how to use IP cores in Vivado, and how to package your design into an IP.

We'll use Xilinx IPs and our user IP to design an adder module to be implemented in the Basys3 board.

The design has the following interface

Port	Direction	Width	Type
A	IN	8	std_logic_vector
B	IN	8	std_logic_vector
RST	IN	1	std_logic
CLK	IN	1	std_logic
SUM	OUT	8	std_logic_vector

The port `A` is constrained to the switches 0-7 on the board, while `B` to the switches 8-15. The `RST` is wired to the central push button `BTC`. Finally, the `SUM` is connected to the LEDs 0-7.

The design should calculate the sum of `A` and `B`, according to the following truth table

RST	SUM
0	A + B
1	0

The inputs `A`, `B` and `RST` shall be debounced before that we can use them.

Exercise 1. Package the debouncer module in a user IP

Go to `~/labs/lab16` and open Vivado.

Create a new Vivado project with the following settings.

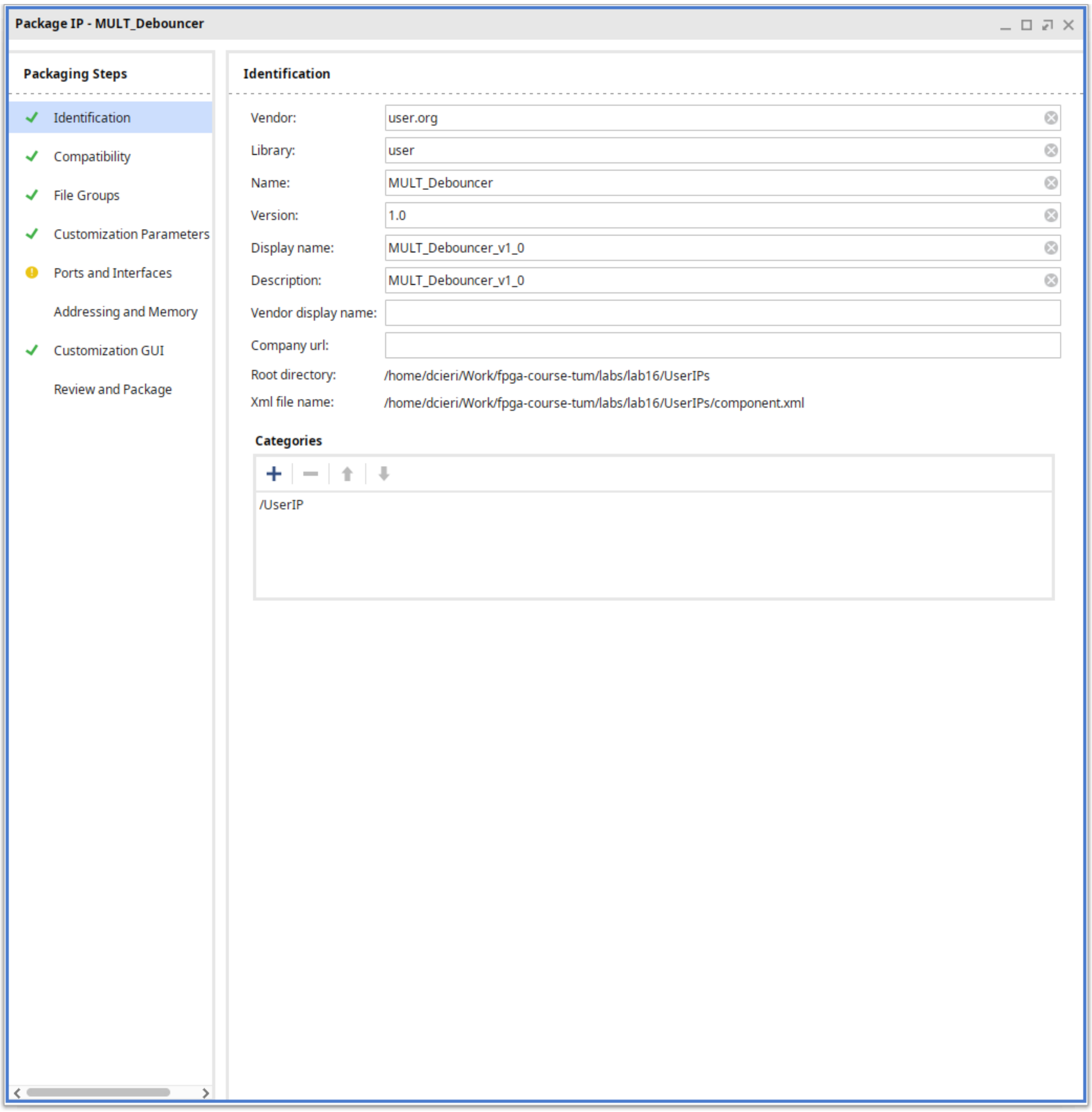
Setting	Value
Project Name	debouncer_ip
Project Type	RTL Project
Add Sources	Add `src/deb.vhd` and `src/multiple_debouncer.vhd` to the project. Select VHDL as target language
Default Part	Select Boards -> Basys3

Open the `MULT_Debouncer.vhd` file to get familiar with the design. The module instantiate a variable number of debouncers equals to the parameter `N_BUTTONS` set in the VHDL generic. The debouncing time `DEBOUNCE_COUNT_MAX` can also be configured.

Once, you finished analysing the code, click on the toolbar `Tools->Create And Package New IP`, and continue using the following settings.

Setting	Value
Create Peripheral, Package IP or Package a Block Design	Packaging Option: Package your current project
Package your Current project	IP location: `~/labs/lab16/UserIPs`

Once you click on Finish, a new Vivado window should open. Here, you will be prompted with the IP packager, where you can set information for your IP.



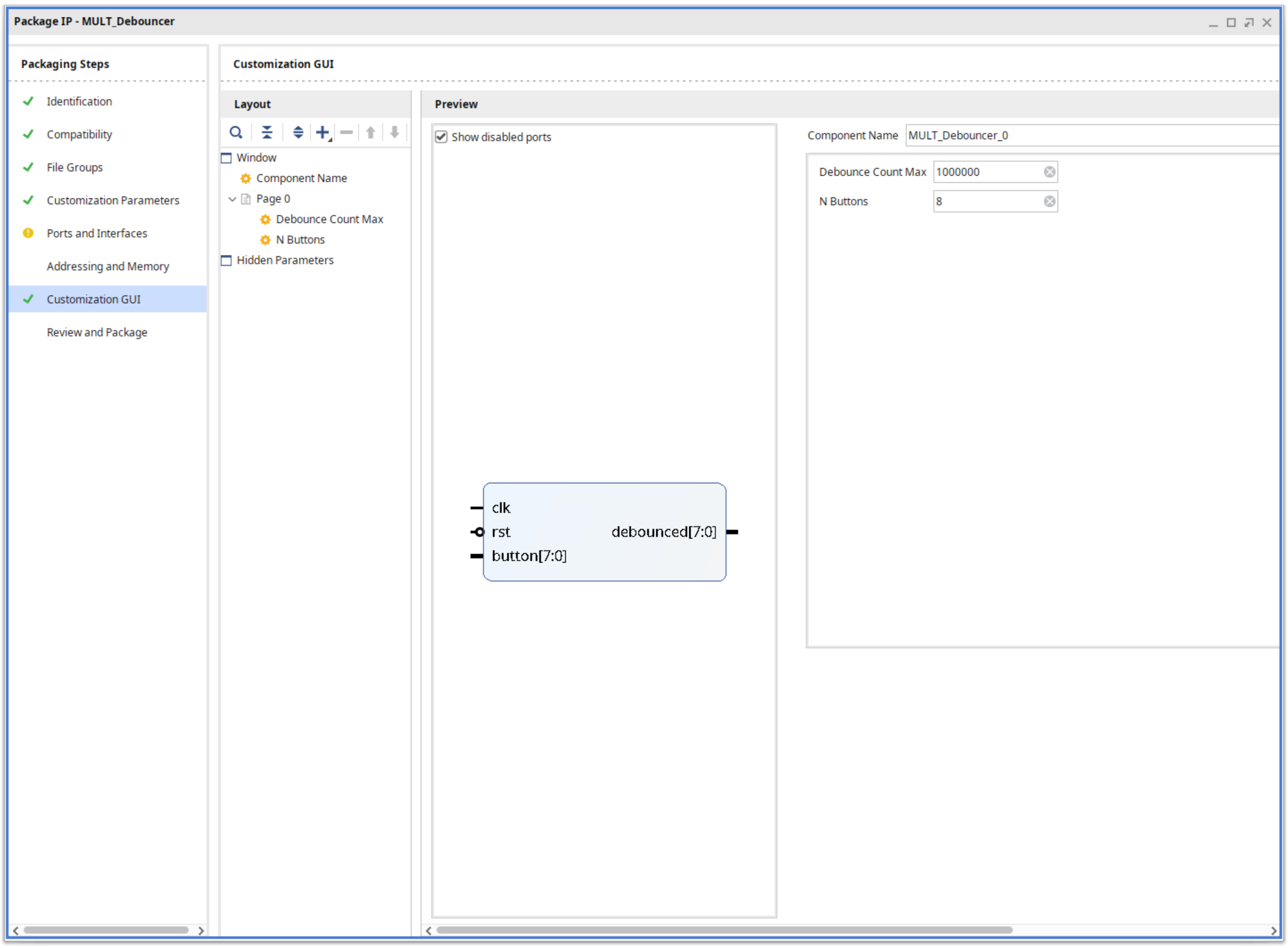
Vendor:	user.org
Library:	user
Name:	MULT_Debouncer
Version:	1.0
Display name:	MULT_Debouncer_v1_0
Description:	MULT_Debouncer_v1_0
Vendor display name:	
Company url:	
Root directory:	/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs
Xml file name:	/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs/component.xml

/UserIP

Have a look at the various `Packaging Steps`. This is a short description of them.

- Compatibility: define the list of FPGA family to which our IP will be compatible. Keep it as it is.
- File Group: The file which are used to package your IP. Keep it as it is.
- Customization Parameters: Here you can see the parameters, that user can use to customise the IP. They should coincide with the generics in the module.
- Ports and Interfaces: The interface signals of your IPs. You should see a warning for the `clk` signal that we can ignore.
- Addressing and Memory: This defines eventual memory mapping in the IP. It's not relevant for our design.
- Customization GUI: Here you can have a look at the GUI page, that will appear when instantiating the IP in a Vivado project.
- Review and Package: This is the final window, where you can review the changes and package the IP.

In the *Customization GUI* page, you might notice that the `rst` is shown as active low, which is the default for Xilinx.



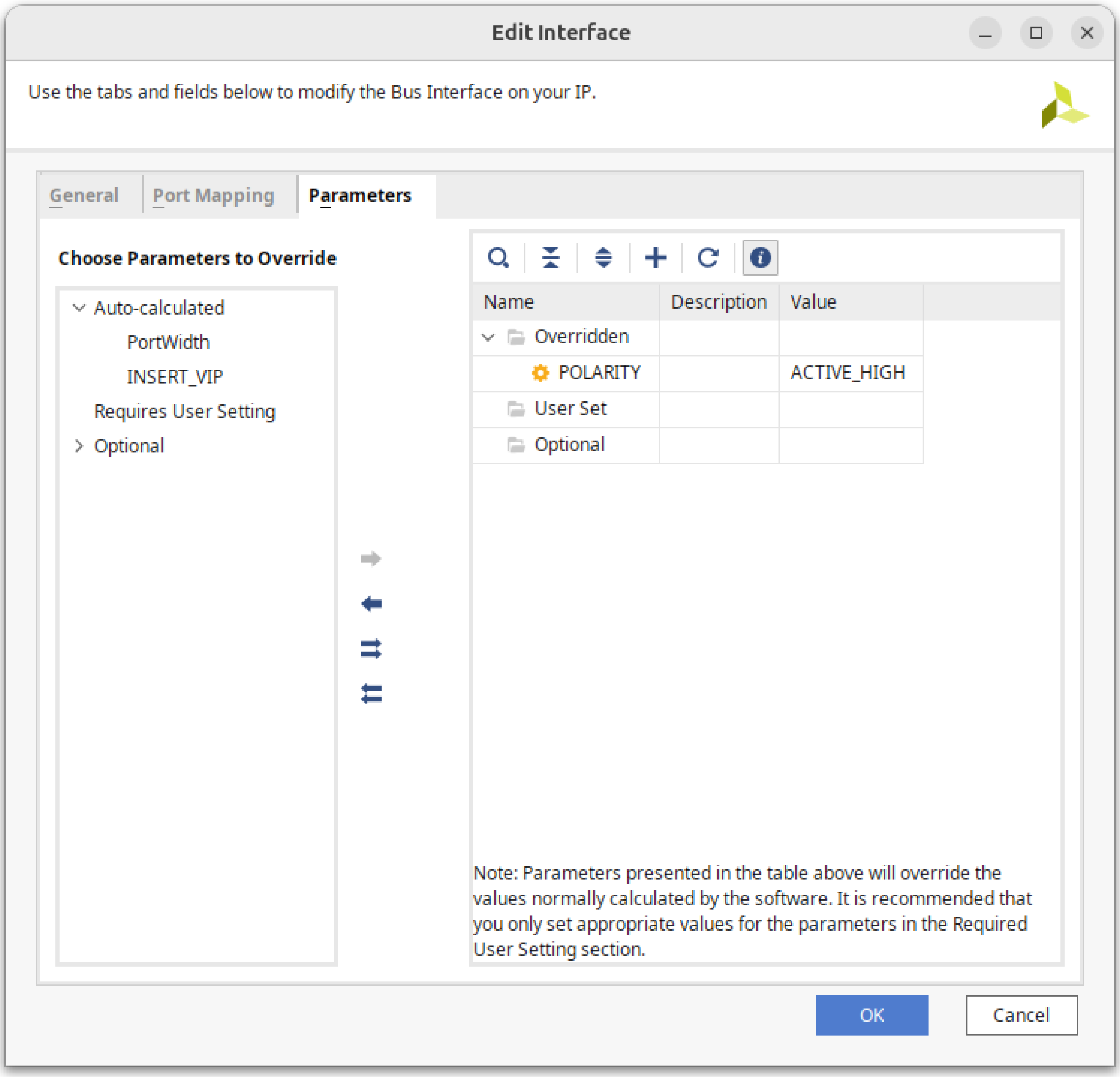
Component Name: MULT_Debouncer_0

Debounce Count Max: 1000000

N Buttons: 8

To fix this, go back to the Ports and Interfaces page, and double click on rst, under *Clock and Reset Signals*

Go to the *Parameters* tab, expand *Auto-calculated* in the left box, select `POLARITY` and click on the right arrow. In the right box, you should now see it under *Overridden*. Select its *Value* cell, and type `ACTIVE_HIGH`.


Edit Interface

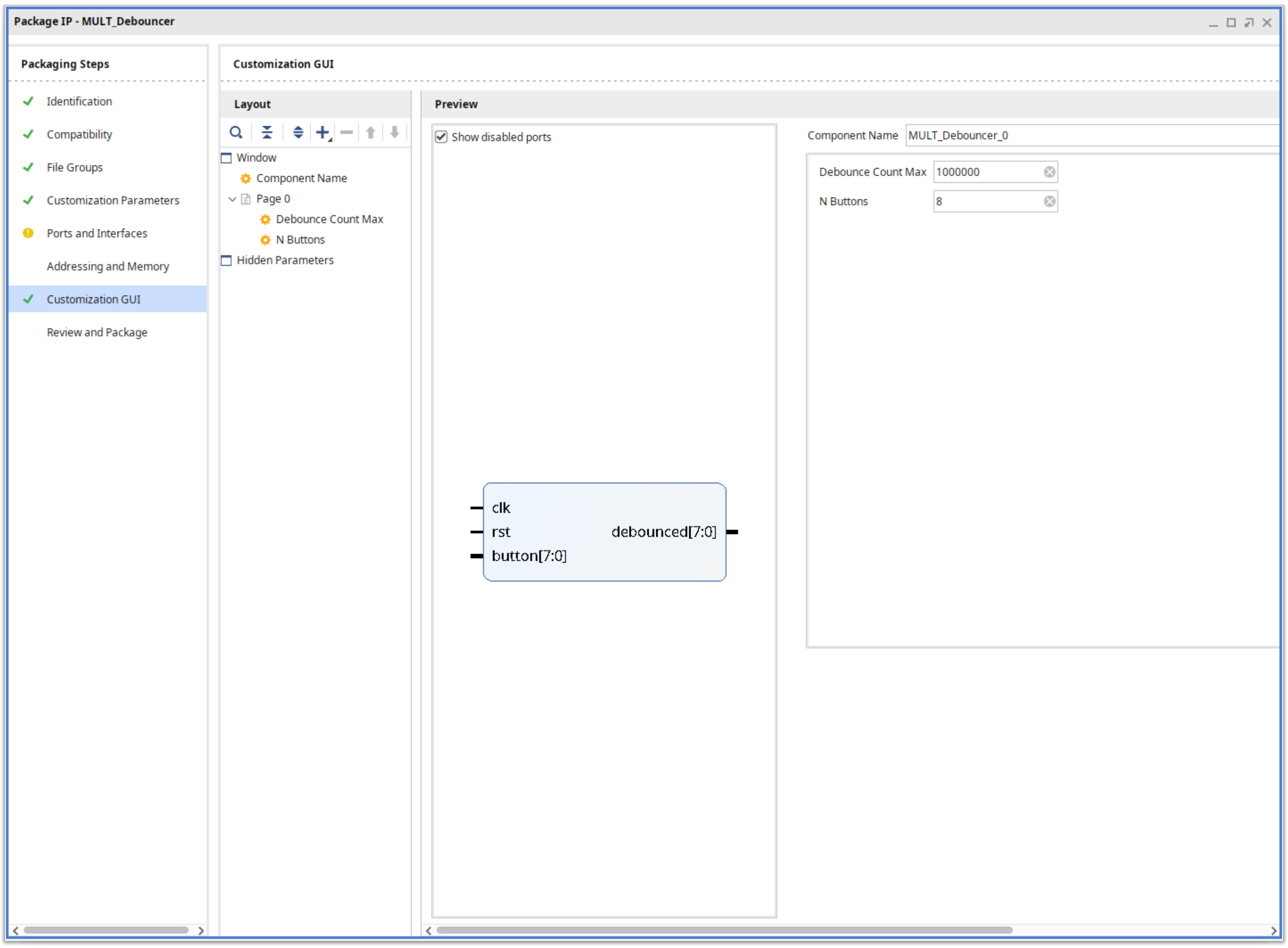
Use the tabs and fields below to modify the Bus Interface on your IP.

Name	Description	Value
Overridden		
POLARITY		ACTIVE_HIGH
User Set		
Optional		

Note: Parameters presented in the table above will override the values normally calculated by the software. It is recommended that you only set appropriate values for the parameters in the Required User Setting section.

Click OK, and refresh the IP page with the circular arrow in the toolbar ⟳. Now expand, the `rst` signal row, left click on the `rst` port, and select *Auto Infer single bit Interface -> Reset*.

If you go back to the *Customization GUI* page, the reset port should look now correctly as active-high.


Package IP - MULT_Debouncer

Packaging Steps
- ✓ Identification
- ✓ Compatibility
- ✓ File Groups
- ✓ Customization Parameters
- ● Ports and Interfaces
- Addressing and Memory
- ✓ Customization GUI
- Review and Package

Customization GUI

Layout
- Window
  - Component Name
  - Page 0
    - Debounce Count Max
    - N Buttons
- Hidden Parameters

Preview

Show disabled ports

Component Name: MULT_Debouncer_0

Debounce Count Max: 1000000

N Buttons: 8

Go to *Review and Package* and click on *Package IP*. You can close the project once finished. Also close the other first project you created.

Exercise 2. Design the Adder

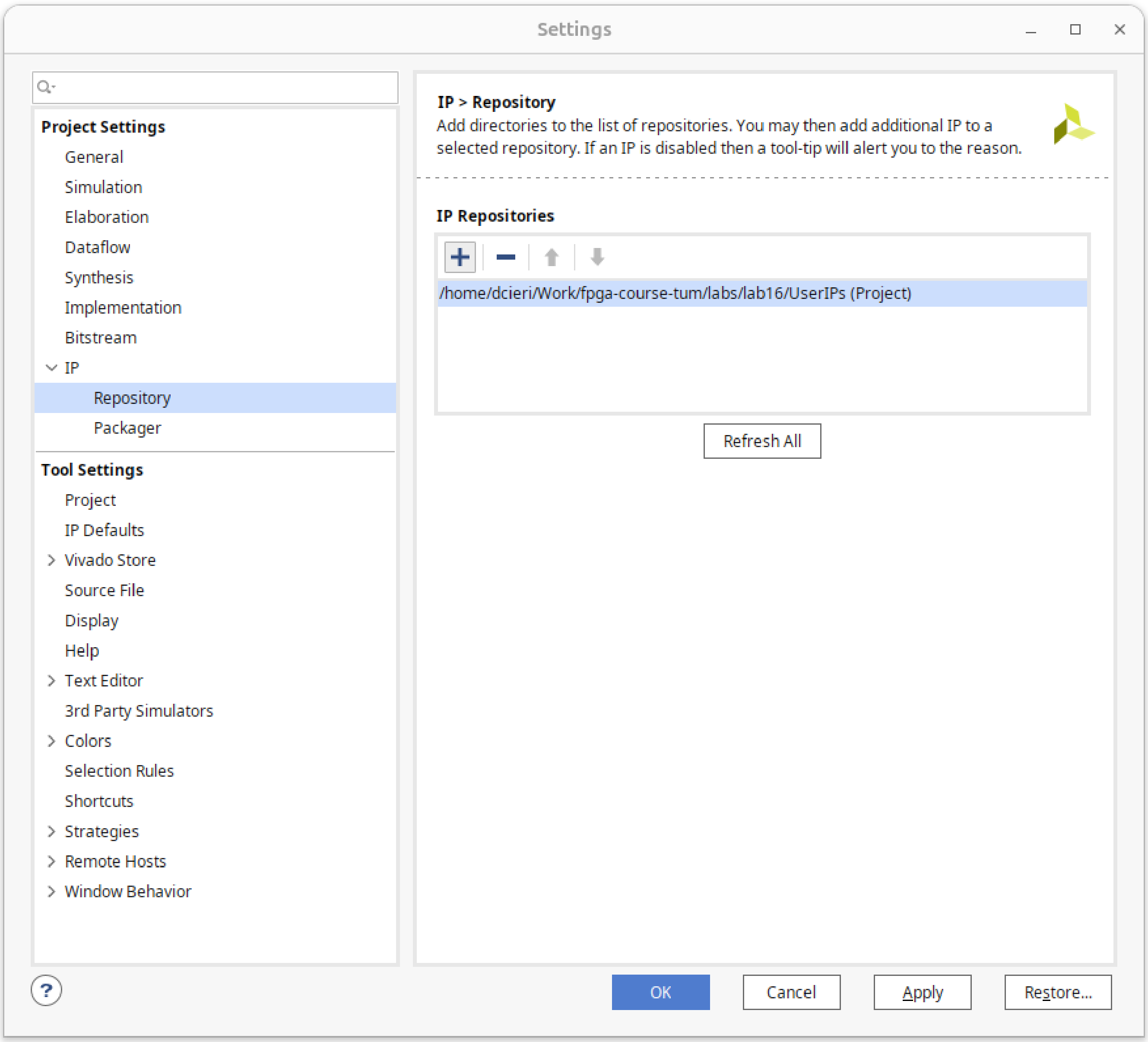
Open another Vivado window and create a new project with the following settings.

Setting	Value
Project Name	adder
Project Type	RTL Project
Add Sources	Add Files: `src/top_adder.vhd`. Select VHDL as target language.
Add Constraints	Add Files: `src/Basys3.xdc`
Default Part	Select Boards -> Basys3

In the *Define Module* window that pop-ups, you can already define the ports as described above, or click OK and modify the code manually.

Once you have defined the port interface, click on the *IP Catalog* on the left. Try to search for the IP, we just packaged. You'll notice that Vivado cannot found it.

The reason is that we need to tell Vivado the location of our User repository. Click on `Settings`, and select the `IP->Repository` page. Click on `+` and select the repository we just created `UserIPs`.



IP > Repository

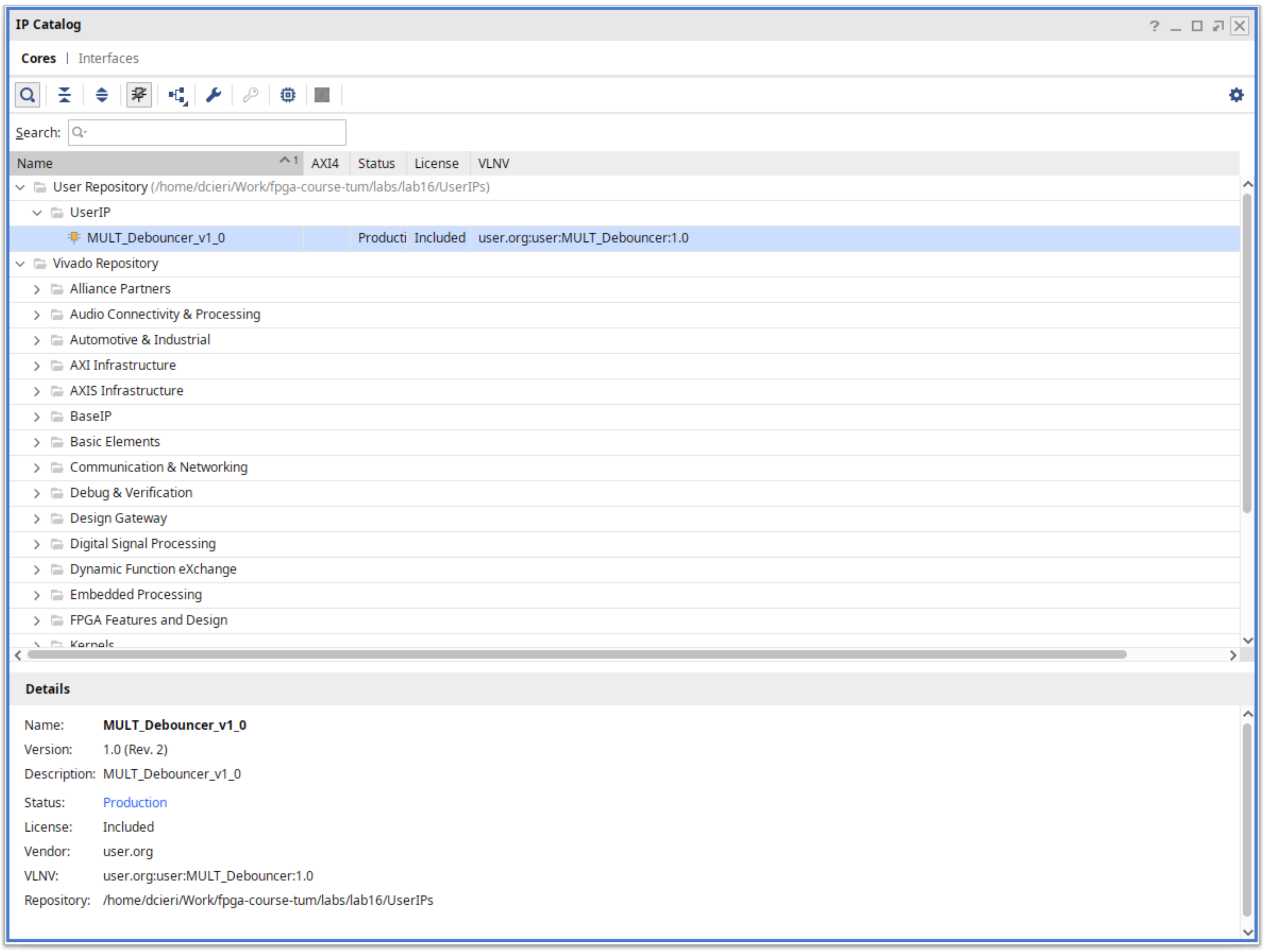
Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason.

IP Repositories

/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs (Project)

.

Click OK to close the window and save the settings. In the IP Catalog, you should now see a folder for the User Repository `UserIP`. If you expand it, you will see our `MULT_Debouncer_v1_0` IP.



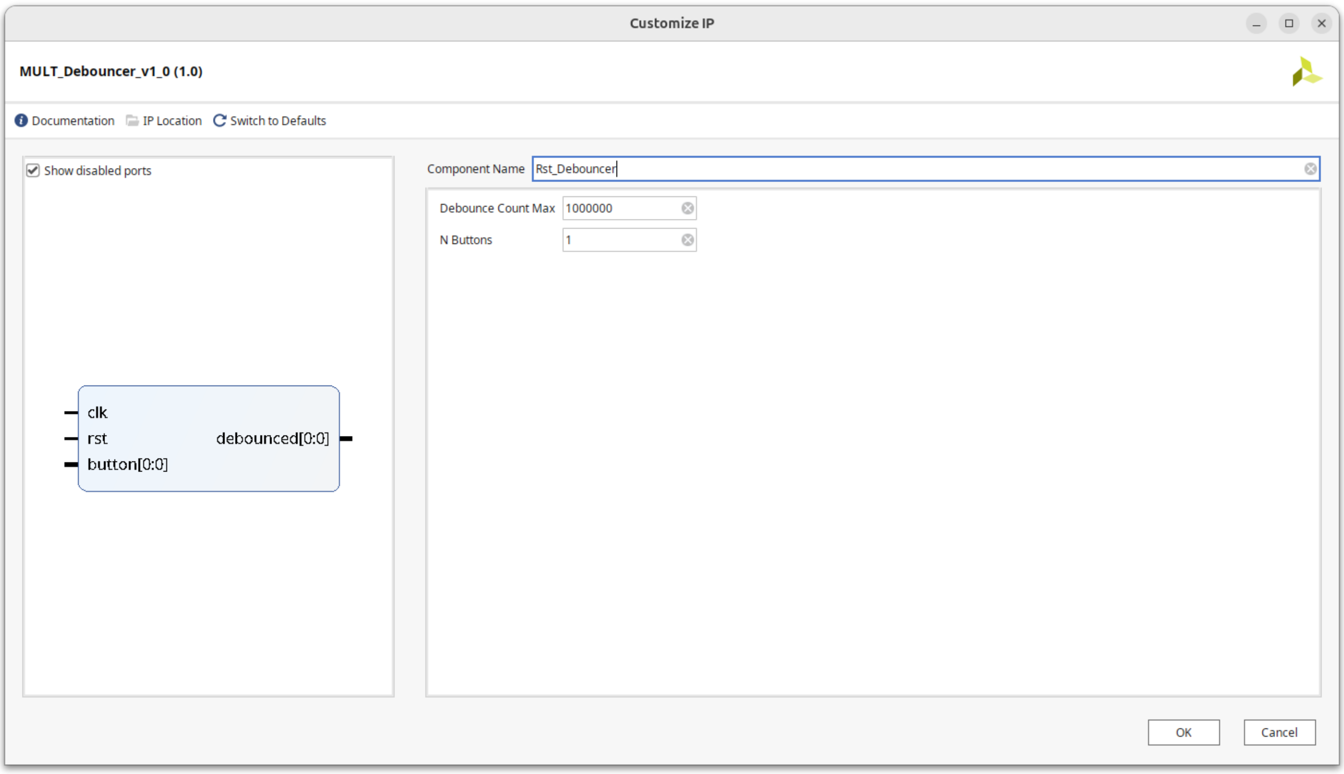
Name	AXI4	Status	License	VLNV
User Repository (/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs)				
UserIP				
MULT_Debouncer_v1_0		Producti	Included	user.org:user:MULT_Debouncer:1.0
Vivado Repository				
Alliance Partners				
Audio Connectivity & Processing				
Automotive & Industrial				
AXI Infrastructure				
AXIS Infrastructure				
BaseIP				
Basic Elements				
Communication & Networking				
Debug & Verification				
Design Gateway				
Digital Signal Processing				
Dynamic Function eXchange				
Embedded Processing				
FPGA Features and Design				
Kernels				

Name:	MULT_Debouncer_v1_0
Version:	1.0 (Rev. 2)
Description:	MULT_Debouncer_v1_0
Status:	Production
License:	Included
Vendor:	user.org
VLNV:	user.org:user:MULT_Debouncer:1.0
Repository:	/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs

.

Double click on our IP. We need to create two configuration for our IP. The first one has `N_BUTTONS=8` and will be used to debounce the `A` and `B` inputs. A second one has just one button (`N_BUTTONS=1`) and will debounce the `rst` signal.

Generate the two IPs, clicking on `Generate` both time. Give the IPs resonable names (e.g. `Switch_debouncer`, `Rst_debouncer`).



MULT_Debouncer_v1_0 (1.0)

Component Name: Rst_Debouncer

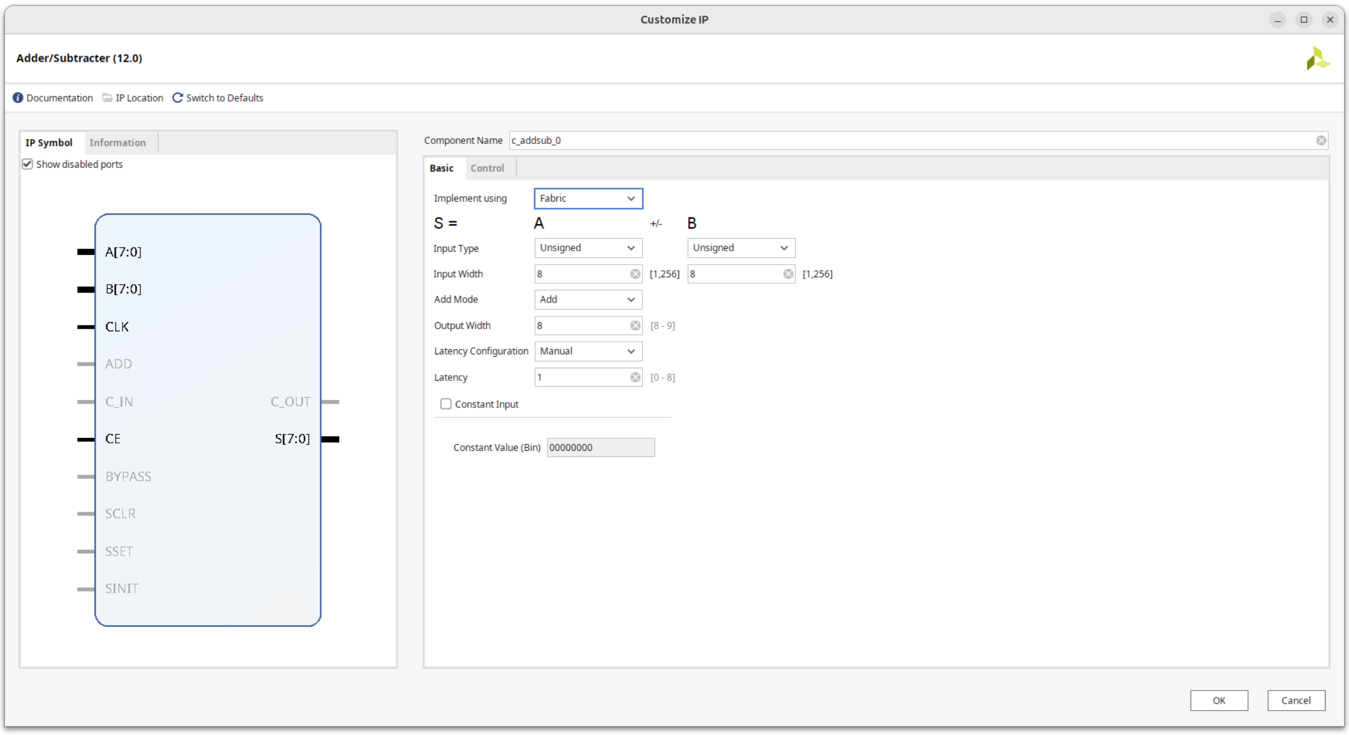
Debounce Count Max: 1000000

N Buttons: 1

Finally, we want to create an IP that performs the addition. Go back to the IP catalog and search for the *Adder/Subtracter* block. Double click and customise it in this way.

- Input type: Unsigned for both `A` and `B`
- Input width: 8 for both `A` and `B`
- Output width: 8
- On the Control Tab, deactivate the *Clock Enable*

Click OK, and generate the IP.



Adder/Subtracter (12.0)

Component Name: c_addsub_0

Implement using: Fabric

S = A +/- B

Input Type: Unsigned | Unsigned

Input Width: 8 [1,256] | 8 [1,256]

Add Mode: Add

Output Width: 8 [8 - 9]

Latency Configuration: Manual

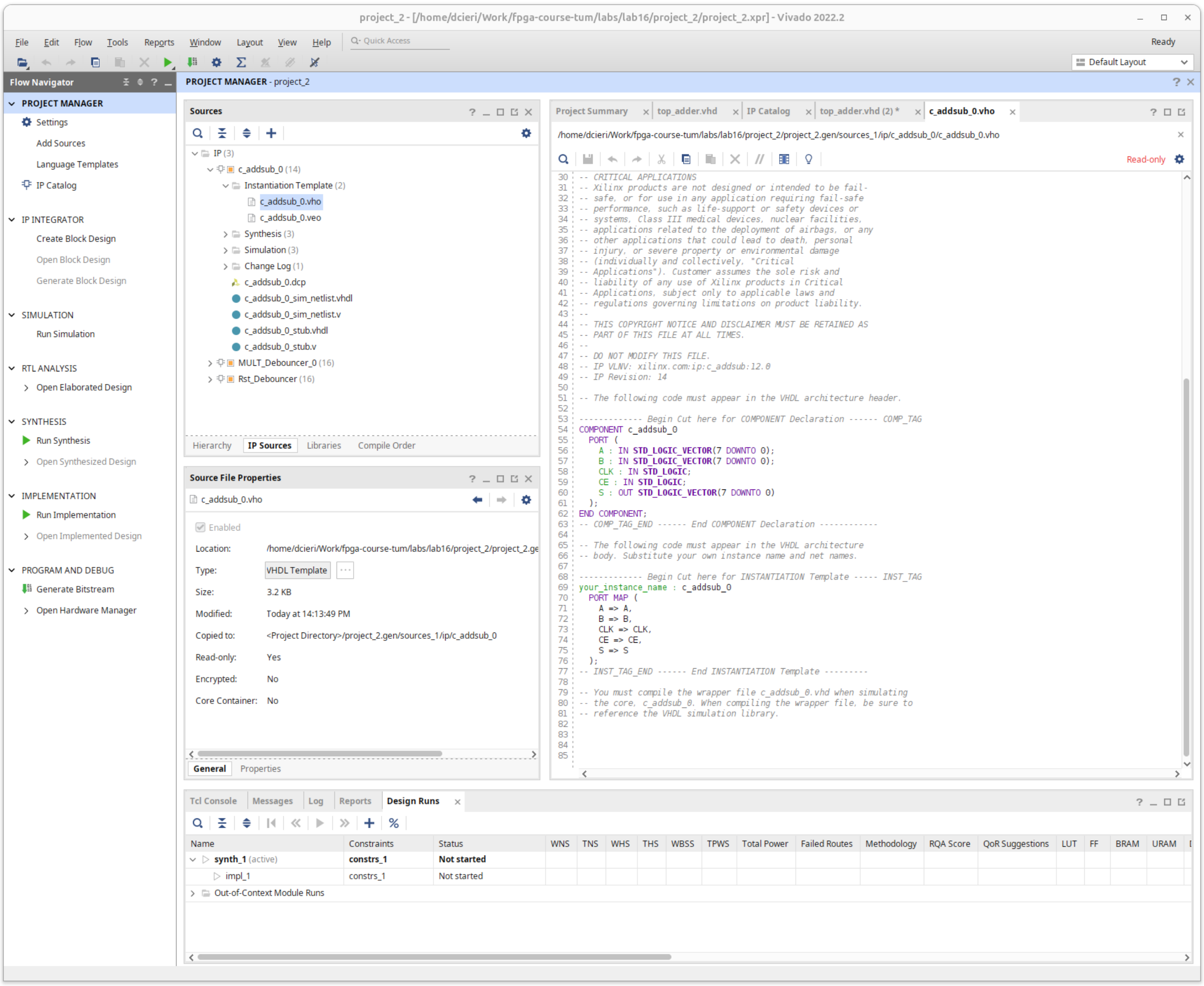
Latency: 1 [0 - 8]

Constant Input: ☐

Constant Value (Bin): 00000000

Now go to the *IP Sources* tab in the *Sources* window. You should see the IPs, we just created. If you expand them you will se the actual IP files, including the *Instantiation Template* for VHDL (`.vho`) and Verilog (`.veo`).

Double click on VHDL Instantiation template for the adder IP.



project_2 - [/home/dcieri/Work/fpga-course-tum/labs/lab16/project_2/project_2.xpr] - Vivado 2022.2

/home/dcieri/Work/fpga-course-tum/labs/lab16/project_2/project_2.gen/sources_1/ip/c_addsub_0/c_addsub_0.vho

```
-- CRITICAL APPLICATIONS
-- Xilinx products are not designed or intended to be fail-
-- safe, or for use in any application requiring fail-safe
-- performance, such as life-support or safety devices or
-- systems, Class III medical devices, nuclear facilities,
-- applications related to the deployment of airbags, or any
-- other applications that could lead to death, personal
-- injury, or severe property or environmental damage
-- (individually and collectively, "Critical
-- Applications"). Customer assumes the sole risk and
-- liability of any use of Xilinx products in Critical
-- Applications, subject only to applicable laws and
-- regulations governing limitations on product liability.
--
-- THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS
-- PART OF THIS FILE AT ALL TIMES.
--
-- DO NOT MODIFY THIS FILE.
-- IP VLNV: xilinx.com:ip:c_addsub:12.0
-- IP Revision: 14

-- The following code must appear in the VHDL architecture header.

------------- Begin Cut here for COMPONENT Declaration ------ COMP_TAG
COMPONENT c_addsub_0
  PORT (
    A : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    B : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    CLK : IN STD_LOGIC;
    CE : IN STD_LOGIC;
    S : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
  );
END COMPONENT;
-- COMP_TAG_END ------ End COMPONENT Declaration ------------

-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.

------------- Begin Cut here for INSTANTIATION Template ----- INST_TAG
your_instance_name : c_addsub_0
  PORT MAP (
    A => A,
    B => B,
    CLK => CLK,
    CE => CE,
    S => S
  );
-- INST_TAG_END ------ End INSTANTIATION Template ---------

-- You must compile the wrapper file c_addsub_0.vhd when simulating
-- the core, c_addsub_0. When compiling the wrapper file, be sure to
-- reference the VHDL simulation library.
```

Source File Properties: c_addsub_0.vho

Location:	/home/dcieri/Work/fpga-course-tum/labs/lab16/project_2/project_2.ge
Type:	VHDL Template
Size:	3.2 KB
Modified:	Today at 14:13:49 PM
Copied to:	<Project Directory>/project_2.gen/sources_1/ip/c_addsub_0
Read-only:	Yes
Encrypted:	No
Core Container:	No

Name	Constraints	Status
synth_1 (active)	constrs_1	Not started
impl_1	constrs_1	Not started
Out-of-Context Module Runs		

You can see here the template to instantiate the IP in your code.

Using the templates instantiate now the IPs, in the `top_adder.vhd` file, such that the output of the 8-bit debouncer modules is connected to the inputs of the `Adder` IP. The `S` output of the IP, shall be then connected to the output port `SUM`.

Once you finished, click on *Generate Bitstream* to launch the Vivado flow. If everything goes well, load the file into the Basys3 board, and test the functionalities by playing with the switches and the buttons.

Exercise 2. Design using Block Diagram

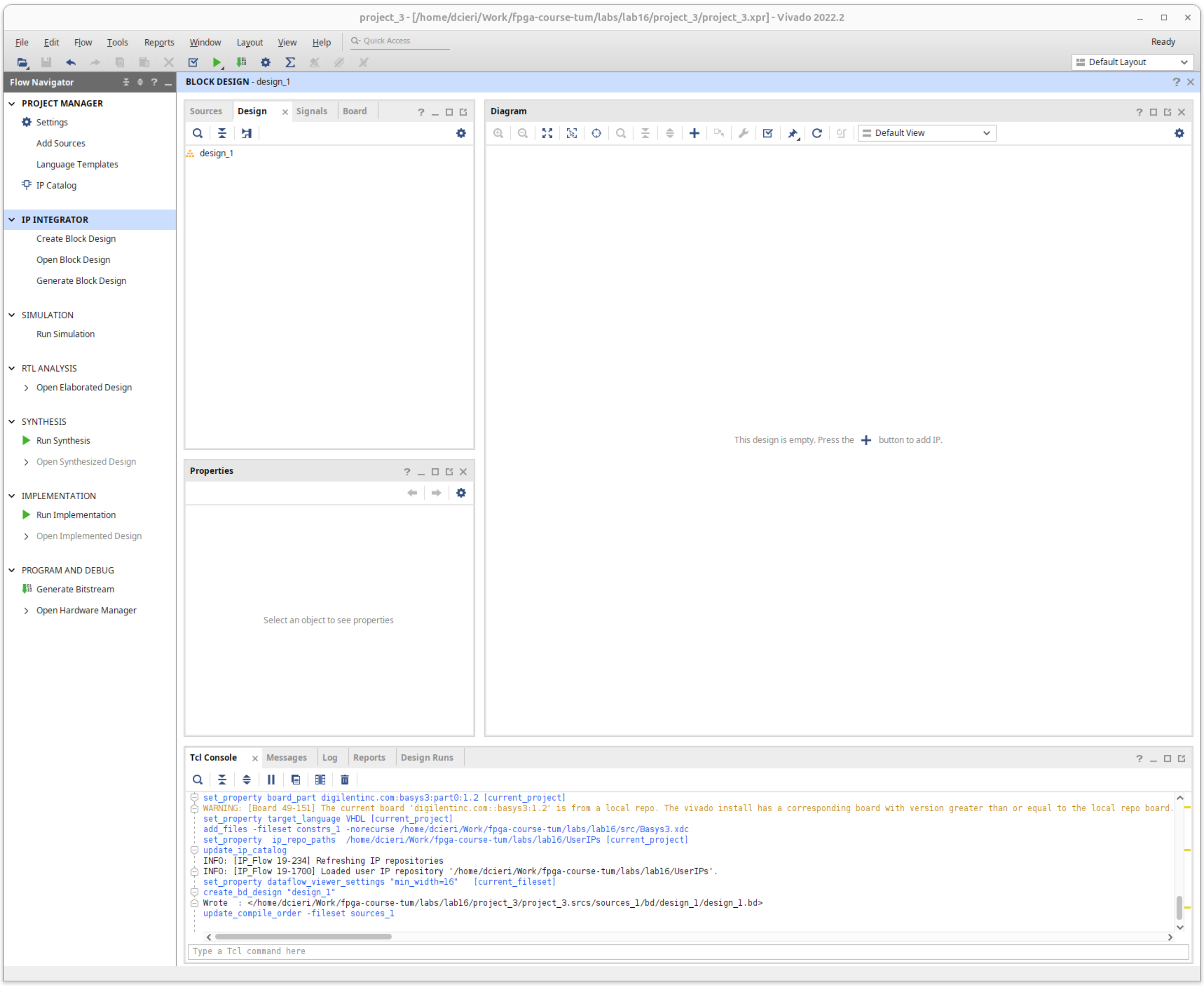
We'll try now to generate the same design, using the Vivado IP Integrator tool. Create a new Vivado project with the following settings.

Setting	Value
Project Name	`adder_bd`

Setting	Value
Project Type	RTL Project
Add Sources	Select VHDL as target language.
Add Constraints	Add Files: `src/Basys3.xdc`
Default Part	Select Boards -> Basys3

Click on Settings and add the user IP repository as before.

Click on *Create Block Desing* on the left sidebar. Give a reasonable name and click OK. You are now prompted with the graphic interface of Vivado to create block designs.

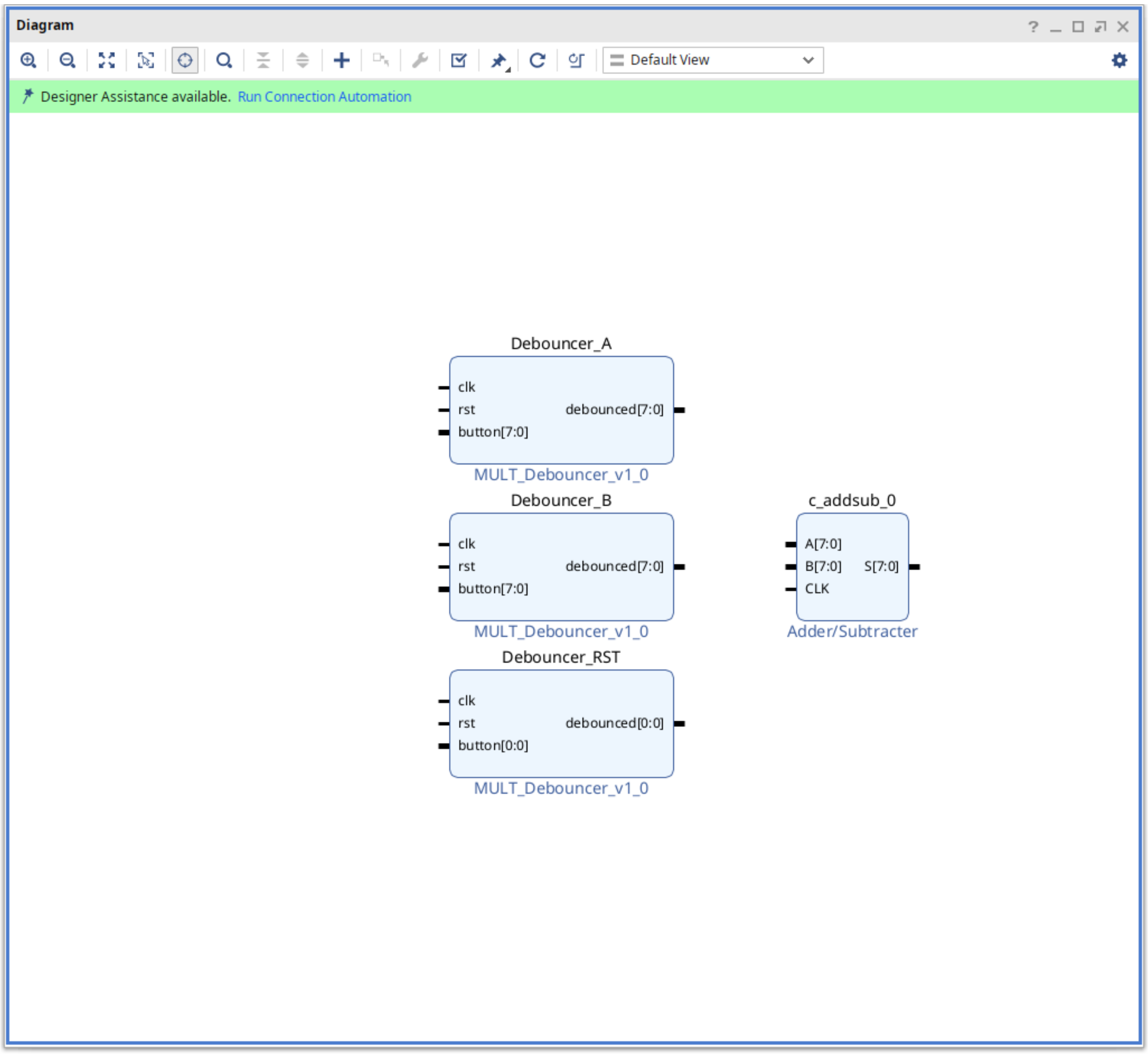


```
set_property board_part digilentinc.com:basys3:part0:1.2 [current_project]
WARNING: [Board 49-151] The current board 'digilentinc.com::basys3:1.2' is from a local repo. The vivado install has a corresponding board with version greater than or equal to the local repo board.
set_property target_language VHDL [current_project]
add_files -fileset constrs_1 -norecurse /home/dcieri/Work/fpga-course-tum/labs/lab16/src/Basys3.xdc
set_property  ip_repo_paths  /home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs [current_project]
update_ip_catalog
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1700] Loaded user IP repository '/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs'.
set_property dataflow_viewer_settings "min_width=16"   [current_fileset]
create_bd_design "design_1"
Wrote  : </home/dcieri/Work/fpga-course-tum/labs/lab16/project_3/project_3.srcs/sources_1/bd/design_1/design_1.bd>
update_compile_order -fileset sources_1
```

Click on the plus button to add the debouncer IP. You should see our block appearing in the diagram. Since we need three of them, select it, copy and paste it twice (CTRL-C CTRL-P) or right click Copy and right click in an empty place and Paste.

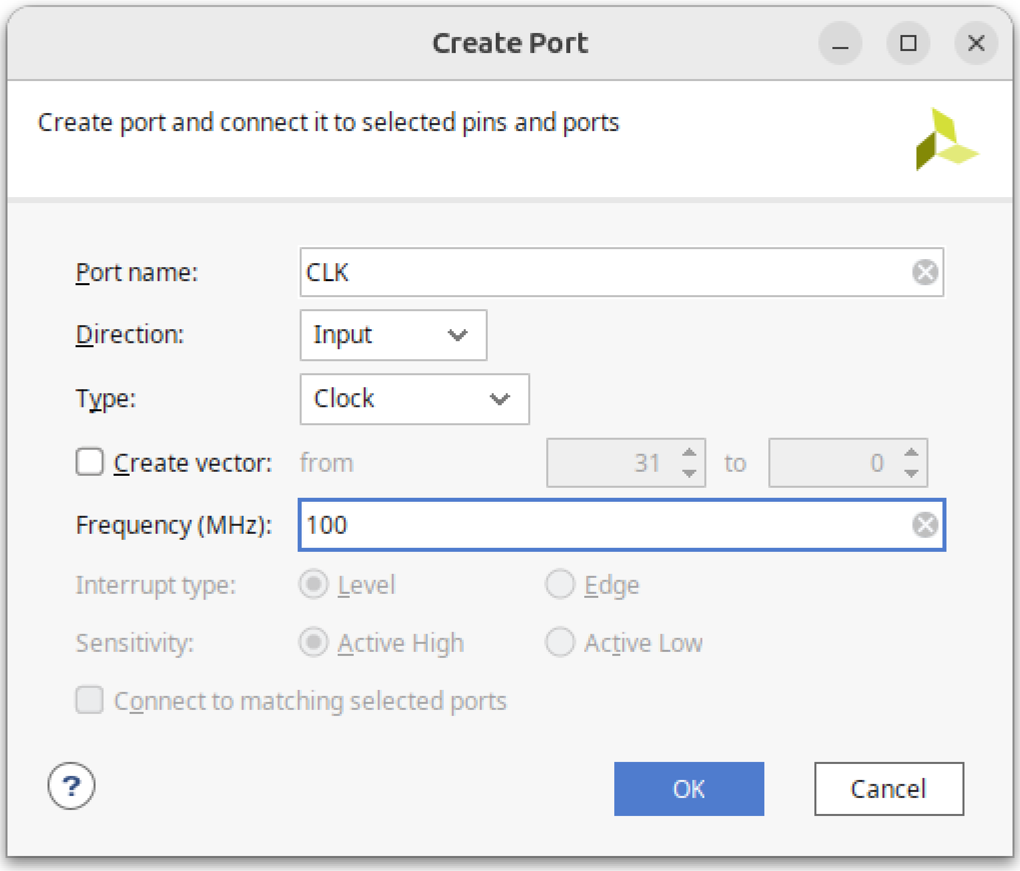
You can then change the block names by selecting them and modify the name in the left box *Block Properties*, *General* tab. Give them reasonable names as before. Double click on the module that you will use to the debounce the reset signal, and change the `N_BUTTONS` to 1.

Finally, click again on plus to add the `Adder/Subtracter` IP block. Double click on the block, and set it up as before. The diagram should now look like this.



As you notice, there are no connections or interface defined in the diagram at the moment.

Let's first add our interfaces. Right click on an empty space in the diagram, or type `CTRL-K`. You should see a pop-up window, that will create our ports. Create the port of our module as described at the beginning of the exercise, selecting the right type.

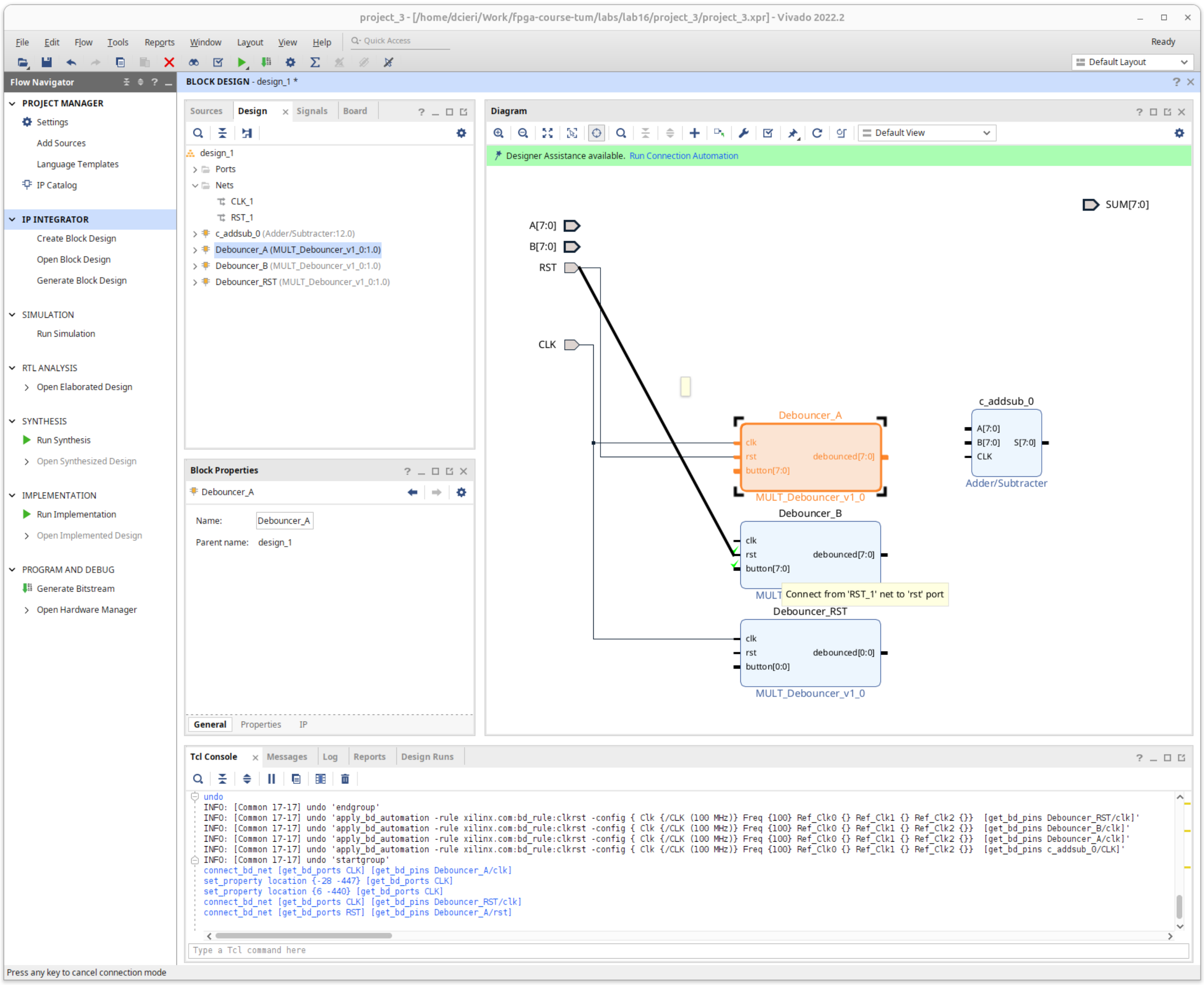

Create Port

Create port and connect it to selected pins and ports

Port name:	CLK
Direction:	Input
Type:	Clock
Create vector:	from 31 to 0
Frequency (MHz):	100
Interrupt type:	[✓ Level] Edge
Sensitivity:	[✓ Active High] Active Low

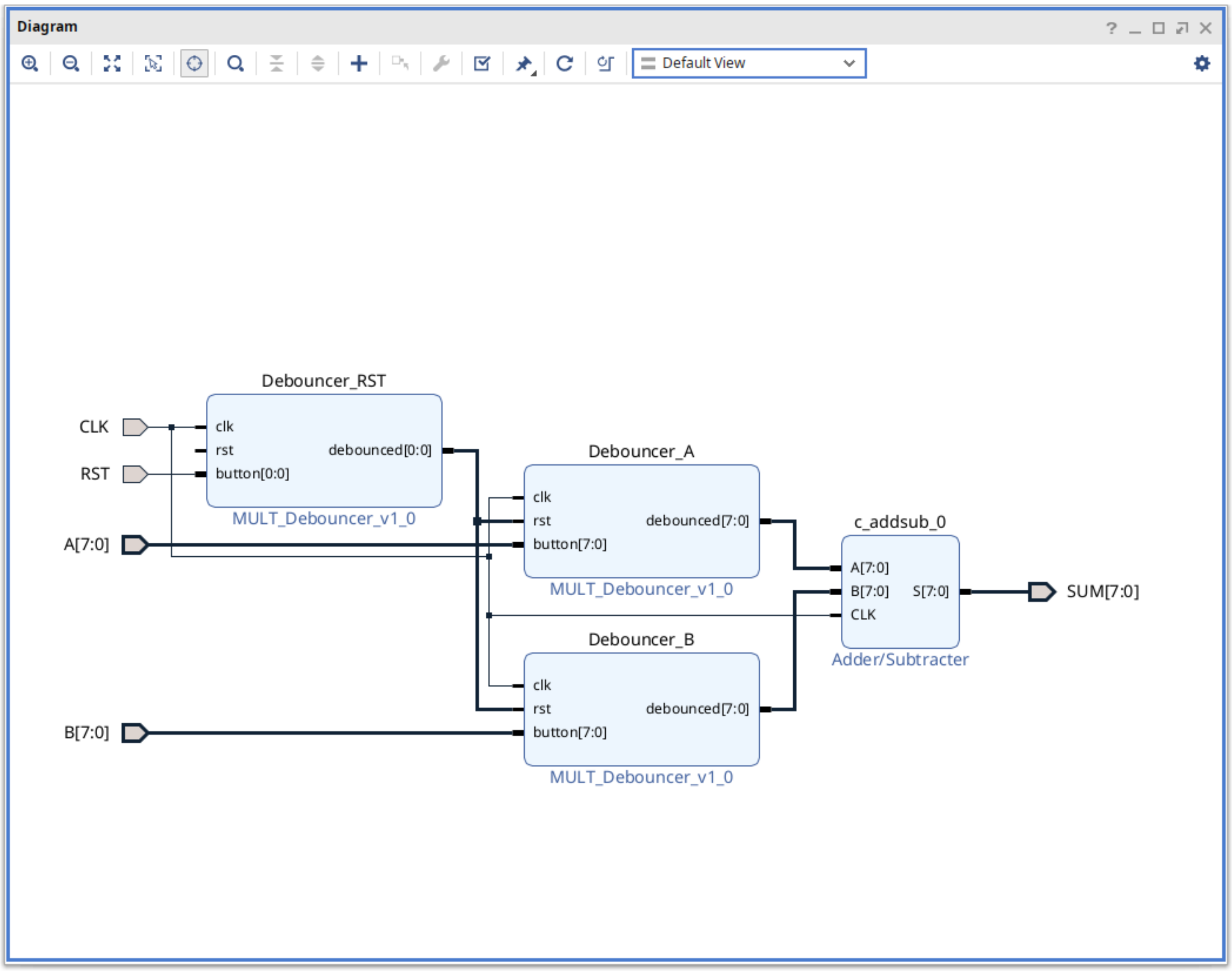
Connect to matching selected ports

Once you created all the ports, start connecting them to the modules in the design, by click on them when you see the pencil Icon, and select the corresponding ports.


project_3 - [/home/dcieri/Work/fpga-course-tum/labs/lab16/project_3/project_3.xpr] - Vivado 2022.2
BLOCK DESIGN - design_1 *
Designer Assistance available. Run Connection Automation
Debouncer_A
Debouncer_B
Debouncer_RST
c_addsub_0
Adder/Subtracter
MULT_Debouncer_v1_0
Connect from 'RST_1' net to 'rst' port
Press any key to cancel connection mode

```
undo
INFO: [Common 17-17] undo 'endgroup'
INFO: [Common 17-17] undo 'apply_bd_automation -rule xilinx.com:bd_rule:clkrst -config { Clk {/CLK (100 MHz)} Freq {100} Ref_Clk0 {} Ref_Clk1 {} Ref_Clk2 {}}  [get_bd_pins Debouncer_RST/clk]'
INFO: [Common 17-17] undo 'apply_bd_automation -rule xilinx.com:bd_rule:clkrst -config { Clk {/CLK (100 MHz)} Freq {100} Ref_Clk0 {} Ref_Clk1 {} Ref_Clk2 {}}  [get_bd_pins Debouncer_B/clk]'
INFO: [Common 17-17] undo 'apply_bd_automation -rule xilinx.com:bd_rule:clkrst -config { Clk {/CLK (100 MHz)} Freq {100} Ref_Clk0 {} Ref_Clk1 {} Ref_Clk2 {}}  [get_bd_pins Debouncer_A/clk]'
INFO: [Common 17-17] undo 'apply_bd_automation -rule xilinx.com:bd_rule:clkrst -config { Clk {/CLK (100 MHz)} Freq {100} Ref_Clk0 {} Ref_Clk1 {} Ref_Clk2 {}}  [get_bd_pins c_addsub_0/CLK]'
INFO: [Common 17-17] undo 'startgroup'
connect_bd_net [get_bd_ports CLK] [get_bd_pins Debouncer_A/clk]
set_property location {-28 -447} [get_bd_ports CLK]
set_property location {6 -440} [get_bd_ports CLK]
connect_bd_net [get_bd_ports CLK] [get_bd_pins Debouncer_RST/clk]
connect_bd_net [get_bd_ports RST] [get_bd_pins Debouncer_A/rst]
```

Once you are finished, you should have a block diagram similar to the following one.



Click on the Validate Design icon . You should get a warning that the RST input of the Reset Debouncer block is not connected and will be grounded. That's fine for us. Ignore and click OK. Save the Design (Ctrl-S or the save icon) and close the Block Design context window.

Back to the *Project Manager* context, expand the *Design Sources*, you should see our block design there. Right click on it, and select Generate HDL Wrapper. Keep the default and click OK.

Vivado should have now created the wrapper top module for our design. Open it and check that its port correspond to the one in the constraint file.

If everything looks fine, generate the bitstream once again, load the firmware onto the board, and play with the switches to validate our design.