## 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	Туре
А	IN 8		std_logic_vector
В	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  ${\tt A}, {\tt B}$  and  ${\tt RST}$  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

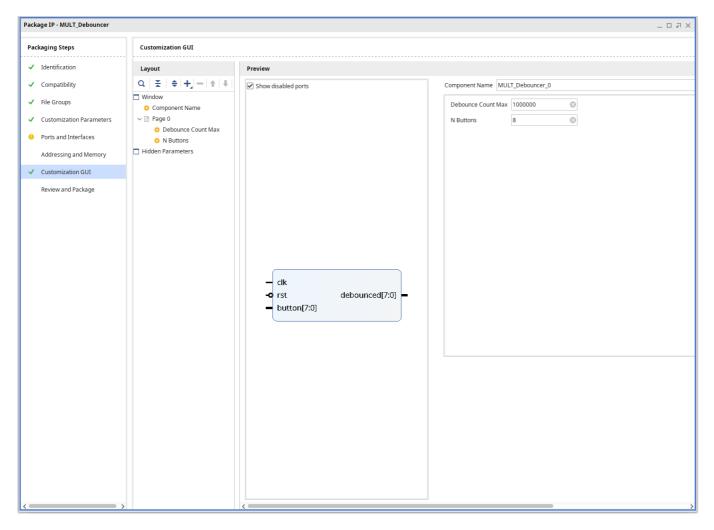
Once you click on Finish, a new Vivado window should open. Here, you will be prompted with the IP packager, where you can set information for your IP.

Pack	age IP - MULT_Debouncer			_ D @ X
Pac	kaging Steps	Identification		
~	Identification	Vendor:	user.org	$\otimes$
4	Compatibility	Library:	user	$\otimes$
~	File Groups	Name:	MULT_Debouncer	$\otimes$
		Version:	1.0	$\otimes$
~	Customization Parameters	Display name:	MULT_Debouncer_v1_0	$\otimes$
•	Ports and Interfaces	Description:	MULT_Debouncer_v1_0	$\otimes$
	Addressing and Memory	Vendor display name:		
~	Customization GUI	Company url:		
	Review and Package		/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs	
		Xml file name:	/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs/component.xml	
		Categories		
		+   -   1   1	l de la construcción de la constru	
		/UserIP		
/				

Have a look at the various Packaging Steps.

- 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: Package your IP and close the project.

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



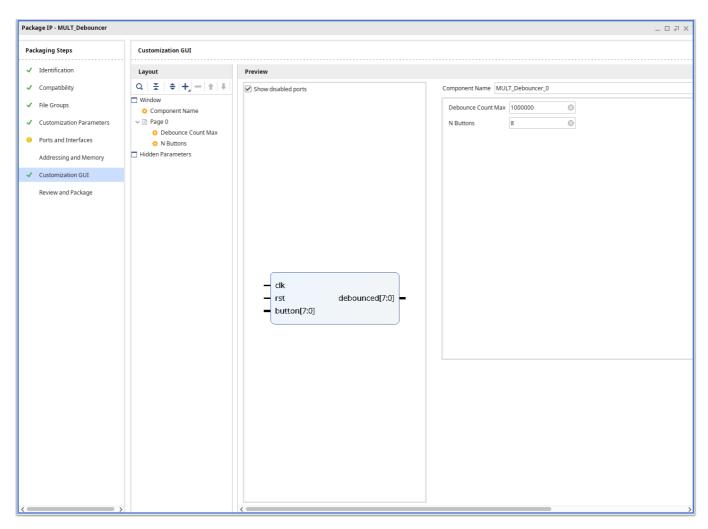
To fix this, go back to the Ports and Interfaces page, and double click on rst, under *Clock and Reset Signals* 

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	- •
Ise the tabs and fields below to modify the Bus	Interface on your IP.	
General Port Mapping Parameters		
Choose Parameters to Override	Q   X   \$   +   C	0
<ul> <li>Auto-calculated</li> <li>PortWidth</li> </ul>	Name Descript	ion Value
INSERT_VIP	POLARITY	ACTIVE_HIGH
Requires User Setting > Optional	User Set	
+ + 11		he table above will override the e software. It is recommended that for the parameters in the Required
		OK Cancel

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.



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: <pre>src/top_adder.vhd. Select VHDL as target language.</pre>
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.

•

	Settings –		>
Q-	IP > Repository		
Project Settings	Add directories to the list of repositories. You may then add additional IP to a		
General	selected repository. If an IP is disabled then a tool-tip will alert you to the reason.		
Simulation			•••
Elaboration	IP Repositories		
Dataflow			
Synthesis			
Implementation	/home/dcieri/Work/fpga-course-tum/labs/lab16/UserIPs (Project)		
Bitstream			
∼ IP			
Repository			
Packager	Refresh All		
ool Settings	Refrest All		
Project			
IP Defaults			
> Vivado Store			
Source File			
Display			
Help			
> Text Editor			
3rd Party Simulators			
> Colors			
Selection Rules			
Shortcuts			
> Strategies			
> Remote Hosts			
> Window Behavior			
?	OK Cancel <u>A</u> pply Re	<u>s</u> tore	_

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.

IP Catalog	2
Cores   Interfaces	
	6
Search: Q-	
Name ^1 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	
N The Karnals	
Details	
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).

	Customize IP	- D X
MULT_Debouncer_v1_0 (1.0)		1
ODcumentation 🕞 IP Location C Switch to Defaults		
Show disabled ports	Component Name Rst_Debouncer Debounce Count Max 1000000 @ N Buttons 1 @	©
- clk - rst debounced[0:0] - button[0:0]	-	
		OK Cancel

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.

			Customize IP	- • ×
Adder/Subtra	acter (12.0)			4
1 Documentatio	n 📄 IP Location 😋 Switch to Defa	aults		
IP Symbol	Information		Component Name c_addsub_0	⊗
Show disab	led ports		Basic Control	
			Implement using Fabric V	
			S = A +/- B	
	A[7:0]		Input Type Unsigned V	
-	B[7:0]		Input Width 8 (1,256) 8 (1,256) Add Mode Add V	
	CLK		Output Width 8 (8 - 9)	
			Latency Configuration Manual V	
	ADD		Latency 1 (0 - 8)	
	C_IN	C_OUT	Constant Input	
-	CE	S[7:0]	Constant Value (Bin) 00000000	
-	BYPASS			
	SCLR			
-	SSET			
-	SINIT			
				K Cancel

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 - [/ł	nome/dcieri/Work/fpga	-course-	tum/lab	os/lab1	I6/pro	ject_2/	project	_2.xpr] - Viv	ado 2022.2						-	
<u>F</u> ile <u>E</u> dit Flow <u>T</u> ools Rep <u>o</u> rts	<u>W</u> indow La <u>y</u> out <u>V</u> iew <u>H</u> elp	Q- Quick Access															R	Ready
🕒 🔸 🕗 🖬 🐘 X 🕨	👫 🔅 ∑ % 🕅 🔀														-	Default La	ayout	~
Flow Navigator 🗧 🗧 🗧 🗕	PROJECT MANAGER - project_2																	?
✓ PROJECT MANAGER	Sources		? _ O Ľ X	Proi	ect Summ	narv	× to	o adder:	vhd >	IP Catalog	× top add	er.vhd (2) * >	c addsub (	0.vho ×			2	200
Settings	Q ≚ ≑ +																	
Add Sources	✓         →         →         +         +           ✓         □         IP (3)         - </td <td></td> <td>\$</td> <td>/hor</td> <td>ne/dcieri/</td> <td>Work/tp</td> <td>ga-cou</td> <td>rse-tum/</td> <td>labs/lab1</td> <td>6/project_2/pr</td> <td>oject_2.gen/sou</td> <td>ces_1/ip/c_adds</td> <td>ib_0/c_addsub</td> <td>_0.vho</td> <td></td> <td></td> <td></td> <td>×</td>		\$	/hor	ne/dcieri/	Work/tp	ga-cou	rse-tum/	labs/lab1	6/project_2/pr	oject_2.gen/sou	ces_1/ip/c_adds	ib_0/c_addsub	_0.vho				×
Language Templates	v ⊕ i (5) v ⊕ i c_addsub_0 (14)									< // 🛛	0						Read-or	only 🌣
<ul> <li>IP Catalog</li> <li>IP INTEGRATOR</li> </ul>	Instantiation Template ( c_addsub_0.vho c_addsub_0.veo	2)		31 32 33	safe perf	inx pro e, or f formanc	oducts For us ce, su	are no e in an ch as l	y appli ife-sup	cation requ port or saf	nded to be fa iring fail-sa aty devices o mar facilitie	fe r						
Create Block Design	> Synthesis (3)			35	appl	licatio	ons re	lated t	o the d	leployment o	<sup>f</sup> airbags, or	any						
-	> 🗁 Simulation (3)			37	inju	ury, or	seve	re prop	erty or	environmen	ath, personal tal damage							
Open Block Design	> 🗁 Change Log (1)			39	Appl	licatio	ons").	Custom	er assu	y, "Critica mes the sol	e risk and							
Generate Block Design	c_addsub_0.dcp c addsub 0 sim netlist:	1								x products . applicable								
✓ SIMULATION	<ul> <li>c_addsub_0_sim_netiist;</li> <li>c_addsub_0_sim_netiist;</li> </ul>			42 43	regu	ulation	ns gov	erning	limitat	ions on pro	duct liabilit	γ.						
Run Simulation	c_addsub_0_stub.vhdl			44	THIS	S COPYR	RIGHT	NOTICE	AND DIS	CLAIMER MUS	BE RETAINED	AS						
har sinducion	c_addsub_0_stub.v			46						5.								
✓ RTL ANALYSIS	> 🖓 🔳 MULT_Debouncer_0 (16)			48	DO N IP V	VLNV: x	cilinx	.com:ip	c_adds:	ub:12.0								
> Open Elaborated Design	> 守 🔳 Rst_Debouncer (16)			50	IP F The				t appea	r in the VH	DL architectu	re header.						
✓ SYNTHESIS				53	COMPONE	ENT c o	Begi	n Cut h	ere for	COMPONENT	Declaration -	COMP_TA						
Run Synthesis	Hierarchy IP Sources Librarie	s Compile Order		55	PORT	(			00/7 00									
> Open Synthesized Design	,			56 57	B :	: IN ST	ID LOG	IC_VECT	0R(7 D0 0R(7 D0	WNTO 0); WNTO 0);								
	Source File Properties		? _ 🗆 🖒 ×		CE	K : IN : IN S	STD_EO	GIC;										
✓ IMPLEMENTATION	C_addsub_0.vho		←   →   Ø	60 61	S: );	: OUT S	STD_LO	eic_vec	TOR(7 D	OWNTO 0)								
Run Implementation	C Enabled			62				En	d COMPO	NENT Declar	ation							
> Open Implemented Design				64							)L architectu							
		/tpga-course-tum/lab	s/lab16/project_2/project_2.g	66 67	body	y. Subs	titut	e your	own ins	tance name	and net names							
<ul> <li>PROGRAM AND DEBUG</li> </ul>	Type: VHDL Template			68						INSTANTIAT	ION Template	INST_TA						
👫 Generate Bitstream	Size: 3.2 KB			70	your_in PORT	MAP (	e_name	: c_ad	dsub_0									
> Open Hardware Manager	Modified: Today at 14:13:49	PM		71 72	В =	=> A, => B,												
	Copied to: <project directory<="" td=""><td>&gt;/project_2.gen/sourc</td><td>es_1/ip/c_addsub_0</td><td>73 74</td><td>CLK CE</td><td>K =&gt; CL =&gt; CE,</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></project>	>/project_2.gen/sourc	es_1/ip/c_addsub_0	73 74	CLK CE	K => CL => CE,												
	Read-only: Yes			75 76	S =	=> S												
	Encrypted: No			77	INS7	T_TAG_E	ND	En	d INSTA	NTIATION Te	nplate							
	Core Container: No			79	You	must c	ompil	e the w	rapper	file c_adds	ub_0.vhd when wrapper file	simulating						
	core container. No			80 81						npiling the library.	wrapper rite	, be sure to						
				82														
	<		;	84 85														
	General Properties				<													>
	Tcl Console Messages Log	Reports Design R	tuns ×														? _	
	Q   ★   ♦     (   ≪   ▶	» <b>+</b> %																
	Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF BR	AM L	URAM
	v > synth_1 (active)	constrs_1	Not started															
	⊳ impl_1	constrs_1	Not started															
	> 📄 Out-of-Context Module Runs																	
	<			_	_			0										

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

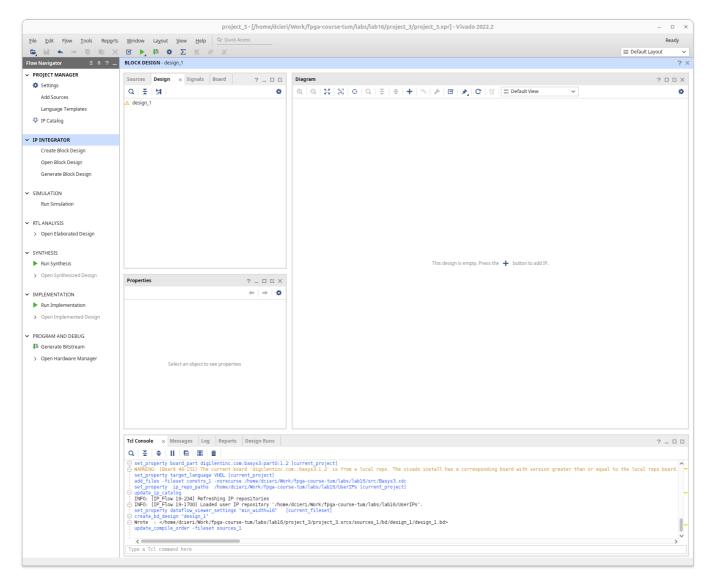
We'll try now to generate the same design, using the Vivado IP Integrator tool. Create a new Vivado project with the following settings.

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.



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.

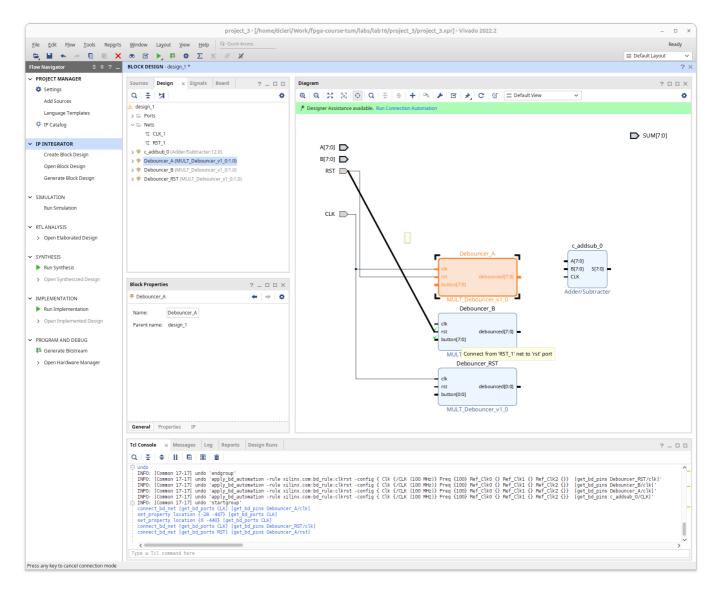
Diag	ram																		? _	
Ð,	Q,	20	$\mathbb{N}$	Ò	 \$	+	$\square_{\eta}$	P	ß	*	С	्र	🔲 Default	View		~				4
			stance ava																	
										[	Debo	uncer	A							
									- clk	¢										
									- rst		-01	deb	ounced[7:0]	•						
										itton[7:			J							
													er_v1_0							
										[	Debo	uncer	В			c_ado	lsub_0	)		
									- clk						•	A[7:0]				
									<ul> <li>rst</li> <li>but</li> </ul>	t itton[7:	:01	deb	ounced[7:0]	•		B[7:0] CLK	S[7:0]	-		
																		)		
												ounc ncer_	er_v1_0		/	Adder/S	ubtracte	er		
										U	ebou	ncer_	.K51							
									- clk			dah	ounced[0:0]	_						
										tton[0:	:0]	uebi	ounceu[0.0]	•						
										MUUT	Dob		er_v1_0							
										WULI	_Den	ounc	er_v1_0							

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

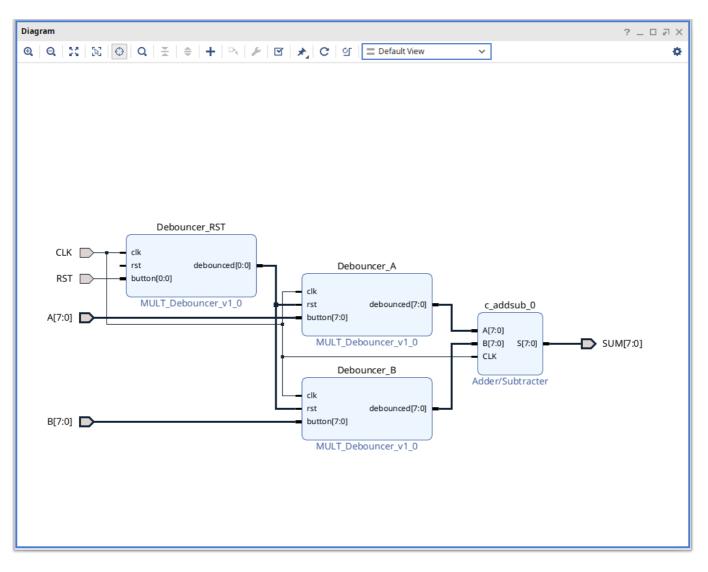
Let's first add our interfaces. Right click on an empty space in the diagram, or type CTRL-K. You should see a pop-up window, that will create our ports. Create the port of our module as described at the beginning of the exercise, selecting the right type.

	Create Port									
Create port and connect it to selected pins and ports										
<u>P</u> ort name:	CLK		$\otimes$							
Direction:	Input 🗸									
T <u>y</u> pe:	Clock 🗸									
Create vector:	from	31 🌲 to	0 🌲							
Frequency (MHz):	100		8							
Interrupt type:	<u>L</u> evel	◯ <u>E</u> dge								
Sensitivity:	Active High	O Ac <u>t</u> ive Low								
Connect to mat	ching selected ports									
?		ОК	Cancel							

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.



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.