Lab 23: MicroBlaze processor on Basys-3

In this lab exercise, we will implement a MicroBlaze soft processor on the Artix-7 FPGA of the Basys-3 board, and connect it to some GPIOs (Push Buttons, LEDs).

We will then program the processor, to control the GPIOs, using the Vitis Unified Software Platform.

Exercise 1. Create the HDL Design

Go to ~/labs/lab23 and create a new project for the Basys-3 board, called *MicroBlaze*. In the *Project Type* section, choose RTL Project, and select:

- In the Add Constraints page, add src/basys3.xdc
- Project is an extensible Vitis platform

Once the project is generated, create a new block design, using the command on the left side bar.

Once you are in the Block Design context, click on the *Board* tab on the left window box.

In this window, we can see all the available interfaces for the Basys-3 board. Dragging them into the *Diagram* window, automatically constraints the pins, without the need of constraint file.

	Misse Diana - Tilaama (dalaali Mawli Kana ay uwa tum dala agi shiga dala 2014 Kina Diana (Misse Diana uwa) - Viugda 2022 2	
File Edit Flaur Teola Departe	MicroBiaze - [momerdicien/work/pga-course-tum/labs-solutions/lab2s/microBiaze/microBiaze.xpr] - vivado 2022.2	Roady
		The Default I arout
Flow Navigator $\pm = ?$	BLOCK DESIGN - design_i	£ ×
PROJECT MANAGER	Sources Design Signals Board × ? _ □ □ Platform Setup × Diagram ×	? □ 岱
Settings		<u></u>
Add Sources	Basys3	
Language Templates	✓ □ Clocks (0 out of 1 connected)	
후 IP Catalog	O System Clock	
	External Memory (0 out of 1 connected)	
Create Block Design	GPIO (0 out of 5 connected)	
Create Block Design	€ 4 Push Buttons	
Open Block Design	iO 7 Segment Display - Anodes	
Generate Block Design	ii 7 Segment Display - Segments	
Export Platform	This design is empty. Press the 🕂 butto	n to add IP.
Run Simulation		
itan Simulation	Description: 100 MHz System Clock	
✓ RTL ANALYSIS		
> Open Elaborated Design		
✓ SYNTHESIS	General Properties	
Run Synthesis		
> Open Synthesized Design	Tcl Console × Messages Log Reports Design Runs	? _ 🗆 🖸
✓ IMPLEMENTATION	i start_gui ⊖ create project MicroBlaze /home/dcieri/Work/fpga-course-tum/labs-solutions/lab23/MicroBlaze -part xc7a35tcpg236	-1
Run Implementation	INFO: [IP Flow 19-234] Refreshing IP repositories INFO: [IP Flow 19-1704] No user IP repositories specified	
 Open Implemented Design 	○ INFO: [IP_Flow 19-2313] Loaded Vivado IP repository '/opt/Xilinx/Vivado/2022.2/data/ip'.	
	☆ WARNING: [Board 49-151] The current board 'digilentinc.com::basys3:1.2' is from a local repo. The vivado instal set property platform extensible true [current report]	l has a corresponding boar 🗖
Generate Bitstream	set_property target_language VHDL [current_project]	
Open Hardware Manager	<pre>Oreate_ou_uesign_</pre>	_l/design_l.bd>
2 Open naruware wanayer	update_complie_order -Tileset sources_1	~
	Type a Tel compand here	>
	Liype a recommand nere	

Drag the system clock into the design window. Click then on *Run Connection Automation*, and confirm with OK in the pop-up window.

Double click on the *clk_wiz_0* block, to confirm that the output clock *clk_out1* is 100 MHz.

Click now on the plus button and add the MicroBlaze IP. Click then on *Run Block Automation*.

A pop-up window will appear that let us configure the MicroBlade processor.

utomatically make connections in your desigr	h by checking the boxes of the bl	ocks to connect. Sel	ect a block on the left to disp	ay its configuration options on the	right. 🗼
Q X ♦ ✓ All Automation (1 out of 1 selected) Ø #	Description MicroBlaze connection au MicroBlaze Debug Module Reset are added and conn Information about the opt Options	tomation generates e, Peripheral AXI Int lected as needed. A ions can be found in	local memory of selected siz erconnect, Interrupt Controll preset MicroBlaze configura the tooltips.	e, and caches can be configured. ar, a clock source, Processor Syste tion can also be selected.	
	Preset Local Memory Local Memory ECC Cache Configuration Debug Module Peripheral AXI Port Interrupt Controller Clock Connection	None 8KB None Obug Only Enabled /clk_wiz_0/clk_out	 ✓ 1 (100 MHz) 		
?				ок	Cancel

Set the local memory to 32kB and click OK. Now drag to the diagram the 16 LEDs. This will create an *AXI GPIO* block, called *axi_gpio_0*. Click on the new block, and rename it in the properties box *axi_gpio_led*.

Now click on the plus button, and add another *AXI GPIO* block. Double click on it, to open the IP configuration page. Go to the *IP Configuration* tab, select *All Inputs*, set the Width to 4, and disable the dual Channel. Confirm with OK. Then rename the IP, *axi_gpio_btn*.

*	Re-customi	ze IP		\vee \wedge
XI GPIO (2.0)				4
Documentation 🛛 📄 IP Location				
Show disabled ports	Component Name axi_gpio_button			
	Board IP Configuration			
	GPIO			
	All Inputs			
	All Outputs			
	GPIO Width 4	8	[1 - 32]	
	Default Output Value 0x000000	00 🔞	[0x00000000,0xFFFFFFFF]	
	Default Tri State Value 0xFFFFFF	F 😗	[0x0000000,0xFFFFFFF]	
:: + S_AXI - s_axi_aclk	Enable Dual Channel			
• s_axi_aresetn	GPIO 2			
	All Inputs			
	All Outputs			
	GPIO Width 32		[1 - 32]	
	Default Output Value 0x000000	00 6	[0x0000000,0xFFFFFFF]	
	Default Tri State Value 0xFFFFFF	F 😗	[0x00000000,0xFFFFFFFF]	
	Enable Interrupt			
				OK Cancel

Back on the Block Diagram, select the GPIO port of the *axi_gpio_button*, right click and select *Make External*. This should connect the port with an external interface (GPIO_0 by default). Click on the interface, and rename it BTN.

Click again on *Run Block Automation* and select all the blocks. Now open the *Platform Setup* tab. You should see a couple of Warning point in the left sidebar.

Platform Setup ? _ 🗆 🗟 X						
Settings Clock						
Name	Enabled	ID	Is Default	Proc Sys	Status	Frequency
External Interfaces						
sys_clock sys_clock sys_clock sys_clock						
clk_out1						
 Error: There must be at least one enabl Error: There must be a default clock. 	ied clock interface.					
	Clock Xame Sys_clock Clk_wiz_0 (Clocking Wizard:6.0) Clk_out1 Clk Clk Clk Clk	Clock Image: Signal Content of Signal	Clock Image: Signal Interfaces Image: Signal Interfaces <th>Clock Image: Sys_clock Image: Sys_clock</th> <th>Clock Image: Systemal Interfaces Image: Systemal Interf</th> <th>Clock Xame Enabled D Is Default Proc Sys Status > > External Interfaces </th>	Clock Image: Sys_clock Image: Sys_clock	Clock Image: Systemal Interfaces Image: Systemal Interf	Clock Xame Enabled D Is Default Proc Sys Status > > External Interfaces

Select Clock, then enable the clk_out1 signal, and set it as default. Ignore the error on the AXI port, since we don't have any AXI ports that can be made available in our design.

Save the design, and click on the *Validate* icon 🗹 to verify.

Open now the *Sources* tab, from the project manager. Right click on the block design file (by default design 1), and select *Create HDL Wrapper*. Once the wrapper is generated, click on Generate Bitstream.

Once the bitstream is generated, we have to export the hardware platform files. Click on *File > Export > Export Platform*. This tool will export the platform, that could be later used inside Vitis, to program our embedded processor.

*	Export Hardware Platform V A					
	Export Hardware Platform					
MLEditions	This wizard will guide you through the export of a hardware platform for use in the Vitis or PetaLinux software tools.					
	To export a hardware platform, you will need to provide a name and location for the exported file and specify the platform properties.					
E XILINX.						

Configure it as follow:

- In the *Platform Type* Window, select Hardware
- In the *Platform State* window, select *Pre-synthesis, and tick the *Include bitstream* box
- Give a reasonable name in the *Platform Properties* window, and keep the rest as default
- Continue and click on Finish

Exercise 2. Program the MicroBlaze processor with Vitis

Click now on the toolbar of Vivado **Tools > Launch Vitis**. If this is the first time you launch Vitis, you should be prompted with the welcome page.

✓ ★ File Edit Search Villey Dreiset Window Hala	workspace - mbla/src/helloworld	d.c - Vitis IDE	~ ^ X
Welcome 3 XILINX VITIS.			- <i>6</i>
	VITIS IDE Project	PLATFORM	RESOURCES
	Create Application Project Create Platform Project Create Library Project Import Project	Add Custom Platform	Vitis Documentation Xilinx Developer

Select *Create Application Project*. This will open another wizard window, that guides you in the creation of an application. Configure it with the following instructions:

- 1. Click Next in the *Create a New Application Project* Page.
- 2. In the *Platform* page, select the tab *Create a new platform from hardware (XSA)*, click on Browse, and select the xsa file, contained in the Vivado project folder in ~/labs/lab23.
- 3. In Application Project Details, provide a name for your project, and click Next
- 4. In Domain keep the default an click Next
- 5. In the Templates, choose *Hello World* and click Finish.

You should now be in the main project window of Vitis.

◀ *	workspace - microblaze_Sw/m	nicroblaze_Sw.prj - Vitis IDE	~ ^ X
File Edit Search Xilinx Project Window H	lelp		
📑 🕶 🗐 💿 💌 🔦 🕶 🔄 💋 🧰 🐇 🕶 (D • 🔗 • 🏷 🗢 • -> •		Q 📝 Design 🕸 Debug
Explorer 🛿 🕒 🗟 😐 🕴 🗖 🗖	メ microblaze_Sw 🛙	- 8	🗄 Outline 🗱 📃 🗆
> design_1_wrapper_1	🛠 Application Project Settings	Active build configuration: Debug 🗸 📎	8 There is no active editor
✓ imicroblaze_Sw[standalone_microblaze_0]	General	Options	that provides an outline.
> இ`Includes >	Project name: microblaze Sw Platform: design 1 wrapper 1 Runtime: cpp Domain: standalone_microblaze_0 . CPU: microblaze_0 . OS: standalone Hardware Specification: View processors.memory	View current BSP settings, or configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/ remove libraries, assign drivers to peripherals, change versions of OS/libraries/drivers etc. Navigate to BSP Settings	
Assistant ⊠	Console 🛿 🖺 Problems 🗐 Vitis Log 🛈 Gu	idance □ □ 1 Set Include Browser Set	earch 🕱 🔽 🗆 🖸
K Emulation-HW K Hardware	CDT Global Build Console	No search results available <u>dialog</u>	. Start a search from the <u>search</u>

In the Explorer box on the left, you should see our hardware platform, imported from Vivado, and the new application we just created.

If you expand the src/ folder of your application, you should see the actual source files. Open the helloworld.c file and have a look at its content.

This is a simple C code, that will print "Hello World" on screen.

Now connect the board to your pc, and open a minicom terminal.



Push a button on the board, to verify that the connection works. The default firmware should print something.

Keep it the terminal open and visible, and go back to Vitis.

In the bottom left box (Assistant), expand your application, right click on Hardware, and select Build.

Once it is done, click with the right button on your application, and select *Run > Launch Hardware*.

If everything went well, you should see the Hello World string appearing in the minicom terminal.

If you get an error, close the pop-up window, power-down and power-up the board, and try again.



Exercise 3. Drive the GPIO with MicroBlaze

In Vitis, click on *File->New->Application Project* to create another application.

- In the *Platform* page, select the same platform from before
- Give it another name in the Application Project Details page
- In the *Templates* page, select Empty Application C

Now select the src folder of your new Application in the *Explorer* window, right click and select Import Sources.

Select then the folder lab23/src, and import the gpio.c file.

Have a look, at this file, to understand what we are doing.

The script initialises the LED / Button GPIOs, using the addresses in the AXI memory, which have been defined in our Vivado Block Diagram and exported to Vitis. (XGpio CfgInitialize function)

With the XGpio SetDataDirection function, we declare if the GPIO is an input (1) or output interface (0).

In the while loop, we continuously check for the values of the buttons, and if any of them is pushed, we switch on the first four LEDs.

Build now the project, as we did before, and run it on the hardware, to verify that works.

Lab23.md

Modify now the code, to implement the following behaviour:

- If button 0 (BTN UP) is pushed, switch on the first 4 leds
- If button 1 (BTN RIGHT), leds 4-7
- If button 2 (BTN DOWN), leds 8-11
- If button 3 (BTN LEFT), leds 12-15