Lab 01: First look at Vivado

Lab Goals

• Take a simple HDL design through synthesis, implementation and onto the development board, using AMD Vivado

Starting Vivado and opening the project

Open a terminal window and launch vivado

vivado &

You will be then prompted with the launch screen.

Vivado 2024	k1	- • ×
Elle Flow Iools Window Help Q. Quick Access AMDDI Vivado ML Edition		
<section-header></section-header>	Pacent Projects Image: Control and a Basys 3-GPD-hws.ptr/html Image: Control and a Basys 3-Keyboard-hws.ptr/html Image: Control and Control and Basys 3-Keyboard-hws.ptr/html Image: Control and Con	
Tcl Console		

Click on **Open Project**, and navigate to your home area, and open the *labs* directory.

Go inside the lab01 folder, and open the Vivado project file lab01.xpr.

We are now in the Vivado design Environment

The left sidebar shows the **Flow Navigator**, which can be used to launch most of the flow steps. Some of the options are grayed out, since they require some steps in the flow to be run beforehand, e.g. you cannot open the synthesized design, before having run the synthesis.

The panel on the right shows the Project Summary. On the top, you can see the main settings of the project. Check for example, that the *Product Family* is Artix-7 and that the *Project Part* is xc7a35tcpg236-1.

Above the pad, you will see the current context of the project, which is **PROJECT MANAGER** – lab01. The context affects the commands you can run, as we'll se later.

The left pad in the context shows the Sources window. Sources are subdivided in four categories.

- Design Sources: these are the files actually used by Vivado to synthesise the design
- *Constraints*: these are the files that maps your source file to the actual pins of the FPGA.
- Simulation Sources: files used to simulate the design
- Utility Sources: any other file in the project, not belonging to the above category. E.g. a python script.

Expand the *Design Sources* folder, if not yet done.

You should see a hierarchical view of modules in our project, using the following syntax.

<Module Name>(<Architecture Name>) (<Filename>) (<No. of submodules>)

The top module of the design is highlighted in bold. In our case we have,

LED_Counter (Behavioral) (LED_Counter.vhd) (3)

If you single click on this source file, you will see its properties pop up in the directly below panel (*Source File Properties*).

Some of this properties are just file information (E.g. the location), but other are important for the project, like the file type. For LED_Counter.vhd, we set a file type to VHDL, which is fine in this case. Please, note that Vivado refers to the 93 version of VHDL when saying VHDL. To use VHDL 2008, please change the file type.

Now double-click on LED Counter. This will open the file in an editor window on the right.



The Vivado editor provides syntax highlighting and some other simple editor features.

The Example Design (An LED Counter)

The design we are using for the first two labs is a simple 4-bit binary counter, that can be setup to to count up, down and be paused and reset.

The bits are then represented on the Basys3 board LD0-LD3

The functions are controlled by four buttons, that go high when pressed.

- BTC: Pause
- BTU: Up Counter
- BTD: Down Counter
- BTR: Reset Counter

There is a counting mechanism implemented that increases the LED counters every second, since using the system clock of the board (100 MHz) will be to fast for us to see any LED blinking.

RTL Analysis

Vivado allows you to compile your code before actually starting the synthesis and implementation flow. On the left sidebar, click on *Open Elaborated Design* and click OK in the pop-up window.

You can then click on *Schematic* on the sidebar. It should open a block diagram scheme of our design. This is useful a useful tool to check that our design is correctly interpreted by Vivado.



Synthesis

Run now the synthesis, clicking on the Run Synthesis command on the right.

You can follow the synthesis flow, in the Log window on the bottom. Once the synthesis is completed, a pop-up window will appear.



Select Open Synthesized Design and click OK.

You are now in the SYNTHESIZED DESIGN Context as you can see from the top bar. Here you can see two new tabs appearing on the right pad.

- Package: This is the I/O planning where you can see the pins of the FPGA and its configurations
- Device: Here you can see the actual layout of the device.



Let's have a look at the layout. The six blocks (X0Y0->X1Y2) are the six clock regions of the FPGA. The components on the FPGAs are painted in different colors.

- In blue, you have the logic slices.
- The green columns are DSP slices.
- Red are block RAMs.
- In orange are the clock resources.
- The colored blocks on the sides of the devices are the chip I/O.

You can click on any area to see its properties (in the left box), and zoom in to select individual blocks.

There is now another pad in the window showing the Netlist. This shows the nets and leaf cells in your design. You can expand each category and click on individual objects to see its property.

Clicking on the object, will also highlight it in the device view, if placed. Since implementation hasn't been run yet, no object has been placed.

Netlist	? _ D @ X
X H	¢
N LED_Counter	<u>^</u>
🗸 🚍 Nets (169)	
> _[clk_div (32)	
> data0 (31)	
> 🗊 leds (4)	
> _[leds_OBUF (4)	
_ <const0></const0>	
_ <const1></const1>	
n ∫ clk	
_ clk_div[31]_i_2_n_0	
clk_div[31]_i_4_n_0	
clk_div[31]_i_5_n_0	
clk_div[31]_i_6_n_0	
clk_div[31]_i_7_n_0	
clk_div[31]_i_8_n_0	
clk_div[31]_i_9_n_0	
_ clk_div[31]_i_10_n_0	
clk_div_reg[4]_i_2_n_0	
clk_div_reg[4]_i_2_n_1	
_ clk_div_reg[4]_i_2_n_2	
E clk div reat/1 i 2 n 3	~

On the left panel, you can see the device constraints. Here, you can see which I/O bank on the device has constraints to the actual device pin.



If you click for example on I/O Bank 14, you should see the I/O Bank Properties window opening in the directly below pad. Here you should see the names of the port of your design, connected their relative pins. You can click on the Float button, on the top right of the window, to have a better look.

I/O Bank	Proper	ties						?.	2	×
为 I/O Bar	nk 14							+	$ \Rightarrow $	¢
Name	Av	^1	Prohibit	Ports		I/C) Std		Dir	
🔎 E19		0		leds[1]	~	LV	CMOS33*		Output	^
🔎 U19		0		leds[2]	~	LV	CMOS33*		Output	
🔎 V19		0		leds[3]	~	LV	CMOS33*		Output	
🔎 T17		0		reset	~	LV	CMOS33*		Input	
🔎 T18		0		count_up	~	LV	CMOS33*		Input	
🔎 U17		0		count_dowr	n 🗸	LV	CMOS33*	Ŧ	Input	
🔎 U18		0		count_paus	e 🗸	LV	CMOS33*		Input	
🔎 U16		0		leds[0]	~	LV	CMOS33*	Ŧ	Output	
,ື F17		0								
, К12		0								
"© К1З		0								
≫ L12		0								
» L13		0								
≫ M12		0								
» M17		0								
🔊 R17		0								
0 1113		0								~
General	Prop	pertie	s Port S	Summary	Package	Pins	I/O Ports	S	ites \land 🕨	Ξ

Click now on Report Utilization in the sidebar. This will create a estimate report of the resources that our

design will use on the FPGA. You can also click on the percentage icon $\,\%\,$, to have get a relative usage with respect to the available resources on the device.

l Console Messages Log	Repo	orts Design Run	Utilizati	on × Package	Pins I/O Po	orts										
¥ 🔶	4	Q. <u>∓</u> ♦ %	Hierarc	hy												
Hierarchy Summary	î	Name 1	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)	ιL.									
Slice Logic	10	N LED_Counter	32	42	9	1	1									
✓ Slice LUTs (<1%)																
LUT as Logic (<1%)																
✓ Slice Registers (<1%)																
Register as Flip Flop (<19	6)															
Memory	~															
tilization_1	2							 	 	 	 					
												V7 (IOB	X1Y11)	U۶	er IO	

Once you finished exploring, close the Synthesized design context, clicking on the x symbol on the blue bar. You should get a warning message. Just click OK to confirm.

Implementation

Run now the implementation by clicking on Run Implementation on the sidebar. Similarly to the synthesis, you can follow the progress in the log on the bottom panel.

Once it completes, you'll receive another pop-up message.

Open now the *Implemented Design*, and go to the Device tab.

To show the routing click on the Routing Icon



Now you can select a leaf cell in the Netlist window, and you should see where it has been placed on the device.

Select the Auto-fit selection Icon , and select a leaf cell in the netlist. You should now see the inside of the slice where the leaf cell has been stored.



In this view, we use the following color scheme (which is user-configurable)

- Grey: not used
- Cyan: Placed leaf cell
- Green: Placed net.

You can finally run the Report Utilization, to see what resources are actually used by the implemented design. Do you see differences with respect to the Report obtained in the synthesized design?

Close now the implemented Design context window.

Bistream Generation

Generate the bitstream, by clicking on Generate Bistream on the left sidebar. Again, once it completes, you will see a pop-up window.

Select Open Hardware Manager and click OK.

Hardware Manager

The hardware manager allows us to program the FPGA on our Basys-3 board. First of all connect the board to your laptop, using the provided microUSB cable, if not yet done. Switch on the board, using the POWER switch on the top left corner of the board.

The board contains a default firmware which is loaded on the FPGA, that exercises most of the I/Os. We want now to load our design.

In the hardware manager, click on Open Target on the left sidebar. Since we have just one board connected, it is enough to click on auto-connect.

If everything went well you should see the board appearing in the Hardware panel.

	lab01 - [/home/dcieri/Work/fpga-course-tum/labs/lab01/lab01.xpr] - Vivado 2024.1	- • ×
Eile Edit Flow Tools Reports	Window Layout View Help Q- Quick Access	write_bitstream Complete 🛛 🚽
	III Or Σ. M // M Dashboard →	🗮 Default Layout 🗸 🗸
Flow Navigator 🗧 🚊 ?	HARDWARE MANAGER - localhost/xilinx tcf/Dioilent/210183B81E8EA	2 X
V PROJECT MANAGER	There are no debug cores. Program device Refresh device	
Settings		
Add Sources	Hardware ? _ D Ľ ×	
Language Tamplatan		
	Name Status	
· P Catalog	v 🛙 localhost(1) Connected	
	V e vilinx_tcf/Digilent/210183B81E8E Open	
Create Block Design	V @ xc7a35_0(1) Programmed	
Create Block Design	1 XADC (System Monitor)	
Open Block Design		
Generate Block Design		
SIMULATION		
Run Simulation		
✓ RTLANALYSIS		
Run Linter		
> Onen Elaborated Decign		
y open claborated besign		
✓ SYNTHESIS		
Run Synthesis		
Open Synthesized Design	Properties ? _ D 🖾 ×	
y open synchesized besign		
✓ IMPLEMENTATION		
Run Implementation		
> Open Implemented Design		
,,		
✓ PROGRAM AND DEBUG		
👫 Generate Bitstream		
> Open Hardware Manager		
Open Target	Select an object to see properties	
Program Device		
Add Configuration Memory Davi		
Add configuration memory been		
	Internet in the second s	
	Ici Unisole X Messages Senai J/O Links Senai J/O Scans	? = 0 8
	open_hw_target NP0: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210183B81E8EA	^
	<pre>set_property PROGRAM.FILE {/home/dciari/Work/fpga-course-tum/labs/lab01/lab01.runs/impl_1/LED_Counter.bit} [get_hw_devices xc7a35t_0] current hw device [get hw devices xc7a35t_0]</pre>	
	Grefresh hw device -update hw probes false (lindex (get hw devices xc7a35t 0) 0) TNEO: (labtools 27-1424) Device xc7a35t (ITAG device index = 0) is programmed with a design that has no supported debug core/e) in it	
	Lie of Leonore E. Fiel, contee Across (Jino defice index - of is programmed with a design that has no supported debug CDE(3) in it.	
		× *
	Type a Tcl command here	

Here you can see already some information on the board, like its ID and the part number of the FPGA, which is the same as in our design.

The FPGA contains also an XADC system monitor, which monitors the temperature on the device. If you double-click on XADC, a window should pop-up asking to create a New Dashboard. Leave the default and click OK.

	New Dashboard	_		×
Specify the name an	d contents for a new dasl	nboard.	Z	
<u>N</u> ame: dashboard	d_1		8	
Contents				
Q ¥ ♦				
✓ 🖌 xc7a35t_0	System Monitor)			
?	OK	Ca	incel]

You should now see the temperature on the board as a function of time.



Let's program now our FPGA, clicking on Program Device on the sidebar. It should select automatically the bistream we just generated. Keep the default values and click Program.

	Program Device	_		×
Select a bitstream prog optionally select a debu bitstream programmin	gramming file and download it to your hardware device. You car ug probes file that corresponds to the debug cores contained in g file.	n the	Ż	1
Bitstre <u>a</u> m file: Debu <u>g</u> probes file: 🕑 <u>E</u> nable end of st	ork/fpga-course-tum/labs/lab01/lab01.runs/impl_1/LED_Count	er.bit ⊗		
?	<u>P</u> rogram	Ca	ncel	

/

Lab1.md

You should see now on the board Led 0 to 3 going up and down following the counter we implemented. You can push the buttons, to try the other feature of our design.