APPENDIX TO EXERCISE 1
CONTROLLING AN LED ON THE ZEDBOARD

This is an extension of Exercise 1. In this second part, we will use the Zedboard to execute the driver code. The board has eight switches, which we will use for the eight inputs, and has eight LEDs that we can use as outputs.

The mapping of signals to hardware in the Zedboard is made through a constraint file. You can find a sample file in this link:

https://github.com/Digilent/Zedboard-old/blob/master/Resources/XDC/zedboard_master.xdc

Also, remember to install the board drivers so that you can choose the Zedboard when creating a project. Follow the instructions here:

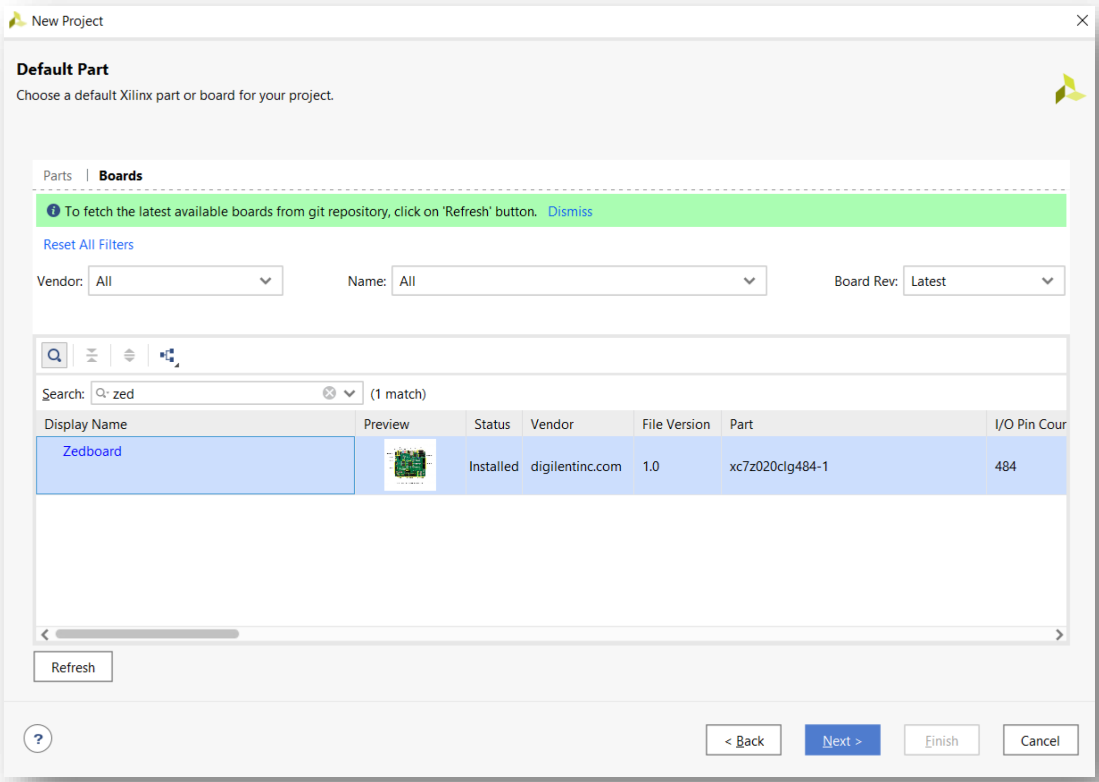
https://reference.digilentinc.com/reference/software/vivado/board-files

The rest of the document contains an example that shows how to get input from the eight switches in the Zedboard and control its LEDs using VHDL code. You can **use this example as a base to write your own code and implement Exercise 1** using switches as inputs.

For colors that are represented by RGB values different than 0 or 255, you will need to **use pulse width modulation** to emulate an intermediate value. To do that, create a new top module that maps the three outputs of your driver (which take values from 0 to 255) into a squared signal with the corresponding duty cycle.

In this example, we will show how to connect the 8 switches in the Zedboard to control the LEDs and also control some of the JA1 pin outputs.. Follow these steps:

- ✓ Create a new project, and choose Zedboard as the default Xilinx board



New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | **Boards**

To fetch the latest available boards from git repository, click on 'Refresh' button. Dismiss

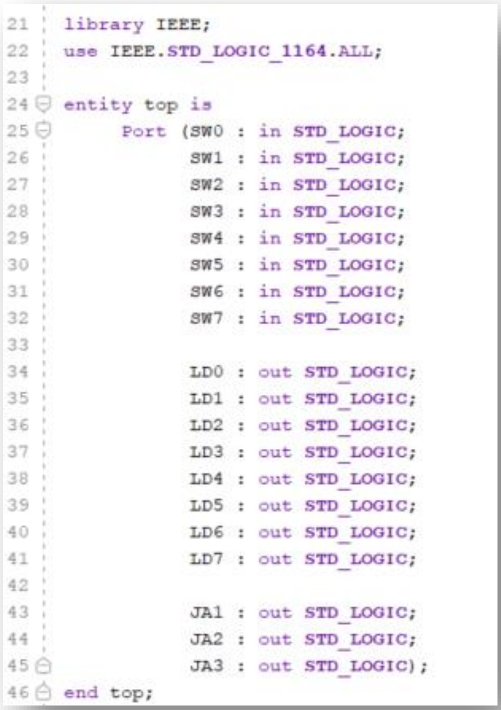
Reset All Filters

Vendor: All   Name: All   Board Rev: Latest

Search: zed (1 match)

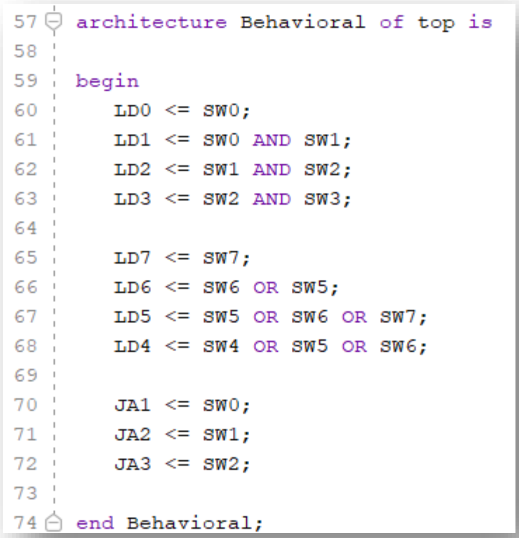
Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Cour
Zedboard		Installed	digilentinc.com	1.0	xc7z020clg484-1	484

- ✓ Create a new source file, top.vhdl. Using the same names as in the constraint file, we will declare the inputs and outputs which will be automatically mapped to the hardware based on the constraint file:



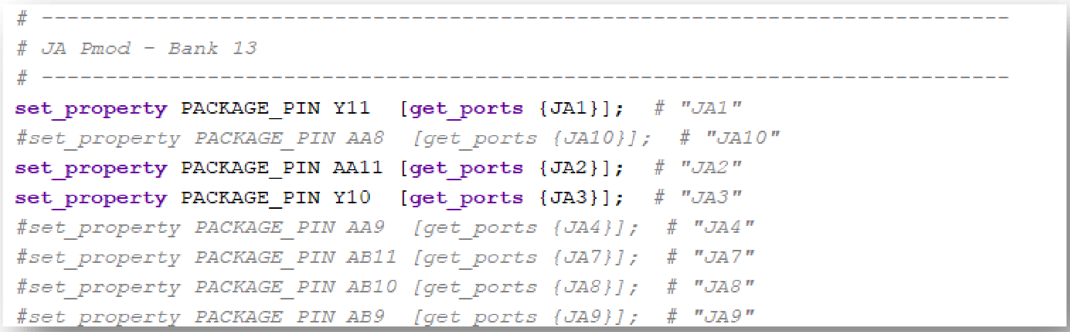
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top is
        Port (SW0 : in STD_LOGIC;
              SW1 : in STD_LOGIC;
              SW2 : in STD_LOGIC;
              SW3 : in STD_LOGIC;
              SW4 : in STD_LOGIC;
              SW5 : in STD_LOGIC;
              SW6 : in STD_LOGIC;
              SW7 : in STD_LOGIC;

              LD0 : out STD_LOGIC;
              LD1 : out STD_LOGIC;
              LD2 : out STD_LOGIC;
              LD3 : out STD_LOGIC;
              LD4 : out STD_LOGIC;
              LD5 : out STD_LOGIC;
              LD6 : out STD_LOGIC;
              LD7 : out STD_LOGIC;

              JA1 : out STD_LOGIC;
              JA2 : out STD_LOGIC;
              JA3 : out STD_LOGIC);
end top;
```

- ✓ Write a simple architecture

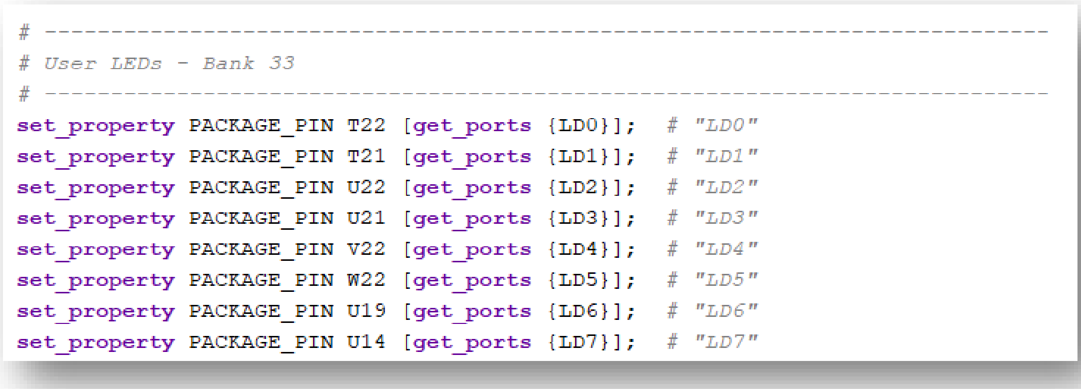


```
architecture Behavioral of top is

begin
   LD0 <= SW0;
   LD1 <= SW0 AND SW1;
   LD2 <= SW1 AND SW2;
   LD3 <= SW2 AND SW3;

   LD7 <= SW7;
   LD6 <= SW6 OR SW5;
   LD5 <= SW5 OR SW6 OR SW7;
   LD4 <= SW4 OR SW5 OR SW6;

   JA1 <= SW0;
   JA2 <= SW1;
   JA3 <= SW2;

end Behavioral;
```

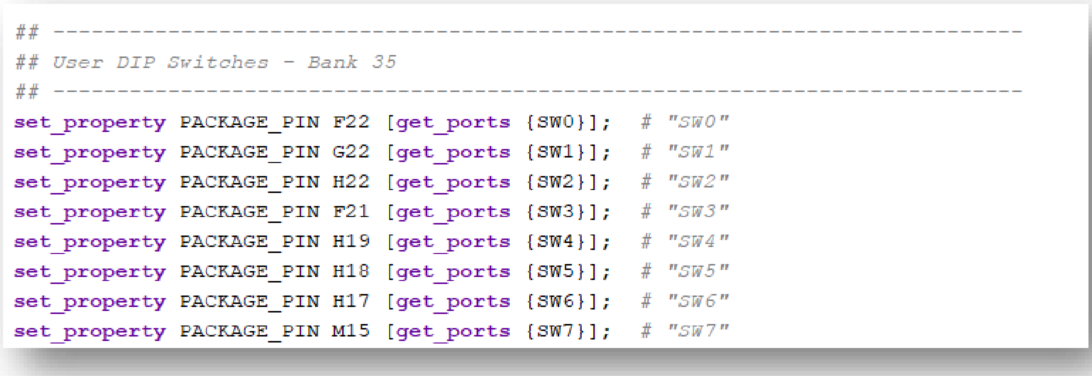
- ✓ Create a new constraint file. Paste the content from the sample you downloaded earlier and uncomment the lines mapping the eight switches, eight LEDs and the RGB LED 5.



```
# ----------------------------------------------------------------------------
# JA Pmod - Bank 13
# ----------------------------------------------------------------------------
set_property PACKAGE_PIN Y11  [get_ports {JA1}];  # "JA1"
#set_property PACKAGE_PIN AA8  [get_ports {JA10}];  # "JA10"
set_property PACKAGE_PIN AA11 [get_ports {JA2}];  # "JA2"
set_property PACKAGE_PIN Y10  [get_ports {JA3}];  # "JA3"
#set_property PACKAGE_PIN AA9  [get_ports {JA4}];  # "JA4"
#set_property PACKAGE_PIN AB11 [get_ports {JA7}];  # "JA7"
#set_property PACKAGE_PIN AB10 [get_ports {JA8}];  # "JA8"
#set_property PACKAGE_PIN AB9  [get_ports {JA9}];  # "JA9"
```

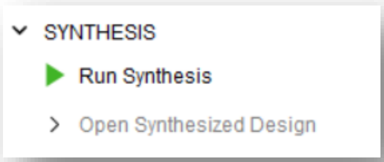


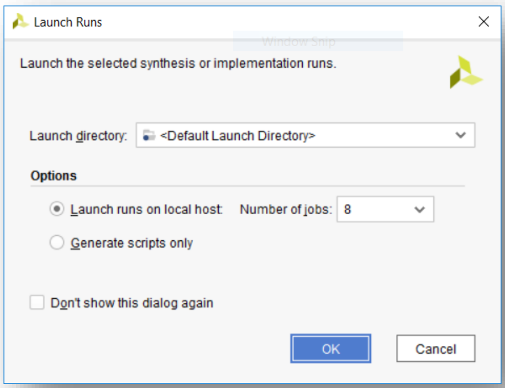
```
# ----------------------------------------------------------------------------
# User LEDs - Bank 33
# ----------------------------------------------------------------------------
set_property PACKAGE_PIN T22 [get_ports {LD0}];  # "LD0"
set_property PACKAGE_PIN T21 [get_ports {LD1}];  # "LD1"
set_property PACKAGE_PIN U22 [get_ports {LD2}];  # "LD2"
set_property PACKAGE_PIN U21 [get_ports {LD3}];  # "LD3"
set_property PACKAGE_PIN V22 [get_ports {LD4}];  # "LD4"
set_property PACKAGE_PIN W22 [get_ports {LD5}];  # "LD5"
set_property PACKAGE_PIN U19 [get_ports {LD6}];  # "LD6"
set_property PACKAGE_PIN U14 [get_ports {LD7}];  # "LD7"
```



```
## ----------------------------------------------------------------------------
## User DIP Switches - Bank 35
## ----------------------------------------------------------------------------
set_property PACKAGE_PIN F22 [get_ports {SW0}];  # "SW0"
set_property PACKAGE_PIN G22 [get_ports {SW1}];  # "SW1"
set_property PACKAGE_PIN H22 [get_ports {SW2}];  # "SW2"
set_property PACKAGE_PIN F21 [get_ports {SW3}];  # "SW3"
set_property PACKAGE_PIN H19 [get_ports {SW4}];  # "SW4"
set_property PACKAGE_PIN H18 [get_ports {SW5}];  # "SW5"
set_property PACKAGE_PIN H17 [get_ports {SW6}];  # "SW6"
set_property PACKAGE_PIN M15 [get_ports {SW7}];  # "SW7"
```

- ✓ Run Synthesis. The higher the number of jobs, the faster *Vivado* will synthesize your design.




Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

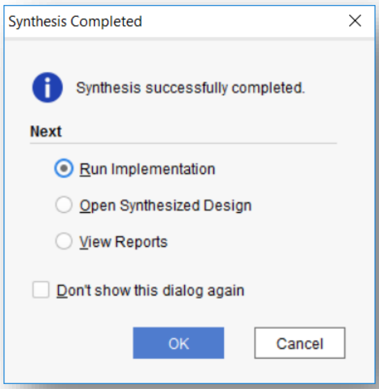
Options

[✓ Launch runs on local host] Number of jobs: 8

Generate scripts only

Don't show this dialog again

✓ Run implementation. Again, choose the highest number of jobs.


Synthesis Completed

Synthesis successfully completed.

Next

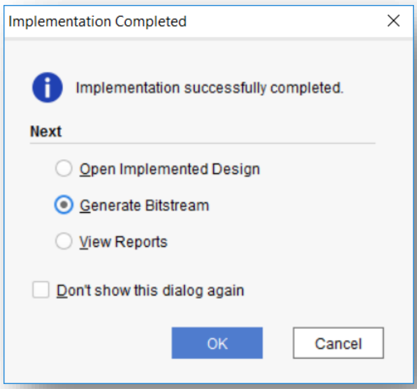
[✓ Run Implementation]

Open Synthesized Design

View Reports

Don't show this dialog again

✓ Choose *Generate Bitstream*.


Implementation Completed

Implementation successfully completed.

Next

Open Implemented Design

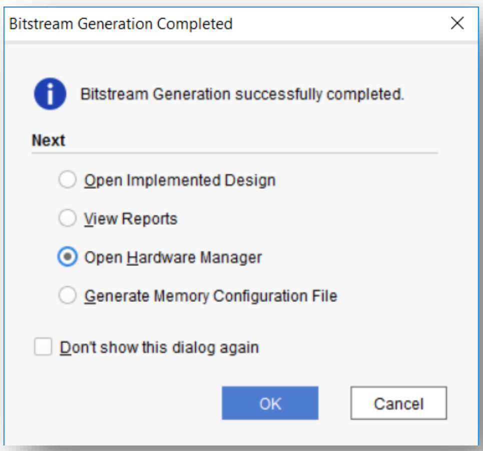
[✓ Generate Bitstream]

View Reports

Don't show this dialog again

✓ Connect the Zedboard. First, make sure the switch is set to OFF. Then connect the power supply and the USB cable to the PROG port. Turn on the board.

✓ Open *Hardware Manager*.



Bitstream Generation Completed

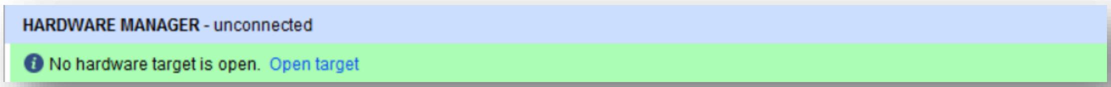
Bitstream Generation successfully completed.

Next

- Open Implemented Design
- View Reports
- [✓ Open Hardware Manager]
- Generate Memory Configuration File

Don't show this dialog again

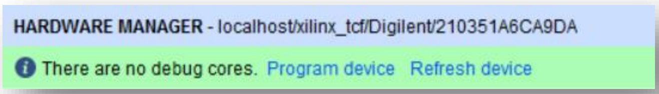
✓ On the top of the window, choose **Open target**. Click in *Auto Connect*.



HARDWARE MANAGER - unconnected

No hardware target is open. Open target

✓ Finally, click on *Program device*.



HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210351A6CA9DA

There are no debug cores. Program device Refresh device

At this point, you should be able to control the LEDs with the switches and also the RGB LED if you connect it to the corresponding JA1 pins.