The Zynq® Book Tutorials

for Zybo and ZedBoard

In association with

XILINX®
ALL PROGRAMMABLE™

University of Strathclyde Glasgow

The Zynq® Book Tutorials

for Zybo and ZedBoard

Louise H. Crockett

Ross A. Elliot

Martin A. Enderwitz

David Northcote

Series Editors: Louise H. Crockett and Robert W. Stewart

Department of Electronic and Electrical Engineering

University of Strathclyde

Glasgow, Scotland, UK

August 2015

This edition first published August 2015 by Strathclyde Academic Media.

Open Source Licence to Use and Reproduce

Tutorial Files

Tutorial files are distributed via the book's companion website: www.zynqbook.com.

Warning and Disclaimer

Trademarks

Acknowledgements

This is a new version of the tutorials accompanying The Zynq Book. It is based in Vivado 2015.1, and now supports both the ZedBoard and the Zybo development boards.

A number of people contributed valuable of feedback on the original set of tutorials, on which these are based. Austin Lesea and Y. C. Wang at Xilinx tested the tutorials at an early stage in their creation, and gave us several useful suggestions. At the University of Strathclyde, Iain Chalmers, Sarunas Kalade, Damien Muir and Craig Ramsay have also been greatly helpful in working through various versions of the tutorials and telling us about their user experiences.

Once again, our sincerest thanks must go to Cathal McCabe of Xilinx University Program, who has not only provided vital feedback and support in the creation of the tutorial material, but has also coordinated the distribution of those materials to others.

Louise Crockett, Ross Elliot, Martin Enderwitz, and David Northcote.

August 2015.

How to Use This Book

Example Files and Ebook Version

In order to follow The Zynq Book Tutorials, you should download a set of prepared files from the book's website:

www.zynqbook.com

An electronic book (non-printable PDF) version of this set of tutorials can also be downloaded from the above link.

Instructions for Zybo and ZedBoard Development Boards

As you read through the tutorials, you will notice that certain procedures have different variations depending on the development board being used. Where a sequence of instructions is board-specific (i.e. relating either to the ZedBoard or the Zybo), the start of the sequence is indicated by a coloured block icon in the left hand margin:

Zed for ZedBoard

Zybo for Zybo

The resumption of instructions common to both boards is marked with another icon:

Resume

Simply pick out the instructions relevant to your board, by identifying either the Zed or Zybo icon, and then look forward to find where to Resume the main flow.

Operating System

The Zynq Book Tutorials have been tested using the Microsoft Windows operating system. It is expected that they will also function on the Linux Kernel OS, although this has not been tested.

Contents

The Zynq Book Tutorials 1

First Designs on Zynq

v1.5, June 2015

Revision History

Date	Version	Changes
14/06/2013	1.0	First release for Vivado Design Suite version 2013.1
19/06/2013	1.1	Updated for changes in Vivado Design Suite version 2013.2
27/01/2014	1.2	Updated for changes in Vivado Design Suite version 2013.4
30/04/2014	1.3	Updated for changes in Vivado Design Suite version 2014.1
1/04/2015	1.4	Updated for changes in Vivado Design Suite version 2014.4
13/04/2015	1.4.1	Updated to include Zybo development board for Vivado Design Suite version 2014.4
18/06/2015	1.5	Updated for changes in Vivado Design Suite Version 2015.1

Introduction

This tutorial will guide you through the process of creating a first Zynq design using the Vivado™ Integrated Development Environment (IDE), and introduce the IP Integrator environment for the generation of a simple Zynq processor design to be implemented on a Zynq development board. The Software Development Kit (SDK) will then be used to create a simple software application which will run on the Zynq's ARM Processing System (PS) to control the hardware that is implemented in the Programmable Logic (PL).

The tutorial is split into three exercises, and is organised as follows:

Exercise 1A - This exercise will guide you through the process of launching Vivado IDE and creating a project for the first time. The various stages of the *New Project Wizard* will be introduced.

Exercise 1B - In this exercise, we will use the project that was created in Exercise 1A to build a simple Zynq embedded system with the graphical tool, IP Integrator, and incorporating existing IP from the Vivado IP Catalog. A number of design aids will be used throughout this exercise, such as the Board Automation feature which automates the customisation of IP modules for a specified device or board. The Designer Assistance feature, which assists with the connections between the Zynq PS and the IP modules in the PL will also be demonstrated.

Once the design is finished, a number of stages will be undertaken to complete the hardware system and generate a bitstream for implementation in the PL. The completed hardware design will then be exported to the Software Development Kit (SDK) for the development of a simple software application in **Exercise 1C.**

Exercise 1C - In this short third exercise, the SDK will be introduced, and a simple software application will be created to allow the Zynq processor to interact with the IP implemented in the PL. A connection to the hardware server that allows the SDK to communicate with the Zynq processors will be established. The software drivers that are automatically created by the Vivado IDE for IP modules will be explored and integrated into the software application, before finally building and executing the software application on the Zynq.

NOTE: Throughout all of the practical tutorial exercise we will be using **C:\Zynq_Book** as the working directory. If this is not suitable, you can substitute it for a directory of your choice, but you should be aware that you will be required to make alterations to some source files in order to complete exercises successfully.

Exercise 1A Creating a First IP Integrator Design

In this exercise we will create a new project in Vivado IDE by moving through the stages of the Vivado IDE *New Project Wizard.*

Zybo The Zybo requires a **one time** additional set-up procedure in order to set the *Default Part* correctly. This is necessary as Vivado 2015.1 does not contain a board part for the Zybo development board. If you have already configured Vivado 2015.1 with the Zybo board part, you can skip this procedure and start from Step (a).

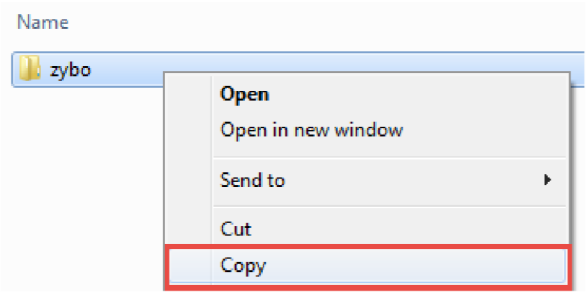
Open windows explorer and navigate to the following location within the Zynq book source files:

C:\Zynq_Book\sources\zybo\setup\board_part

In this directory you will see a file named ***zybo***. This contains the board part for the **Rev. B.3** Zybo development board. You may also check the revision of your Zybo by inspecting the bottom side of your board. Updated board parts can be retrieved from the Digilent Website using the following link:

https://reference.digilentinc.com/vivado:boardfiles

Copy the ***zybo*** file by right clicking on the file and selecting ***copy*** as shown below:



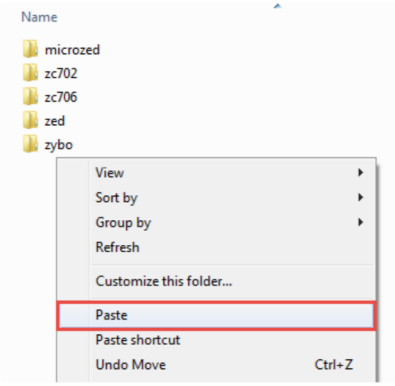
Open a second windows explorer and navigate to the following location in the Vivado 2015.1 installation directory:

{Vivado installation directory}\2015.1\data\boards\board_parts\zynq

This directory is responsible for all the board parts of different Zynq boards that can be used in the Vivado 2015.1 design suite. We will now be adding the Zybo development board to the directory. You may find that a file named ***zybo*** already exists, ignore this and carry on with the

following procedure.

Right click on a blank space in the folder and select ***paste*** as shown below:



A dialogue window may appear asking to *merge* the incoming folder if a ***zybo*** folder currently exists. Click ***Yes***.

You have now successfully added the Zybo board part to the Vivado 2015.1 Design Suite.

Resume We will start by launching the Vivado IDE.

(a) Launch Vivado by double-clicking on the Vivado desktop icon: , or by navigating to **Start > All Programs > Xilinx Design Tools > Vivado 2015.1> Vivado 2015.1**

(b) When Vivado loads, you will be presented with the *Getting Started* screen as in Figure 1.1.

Figure 1.1: Vivado IDE Getting Started Screen

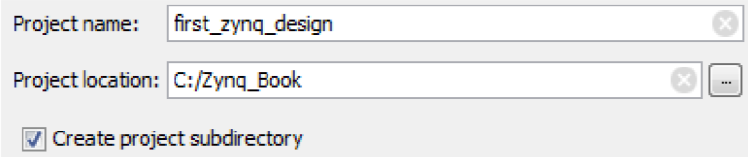
(c) Select the option to ***Create New Project*** and the *New Project Wizard* will open, as in Figure 1.2.

Figure 1.2: New Project Dialogue

Click ***Next***.

(d) At the Project Name dialogue, enter ***first_zynq_design*** as the ***Project name*** and ***C:/Zynq_Book*** as ***Project location***.

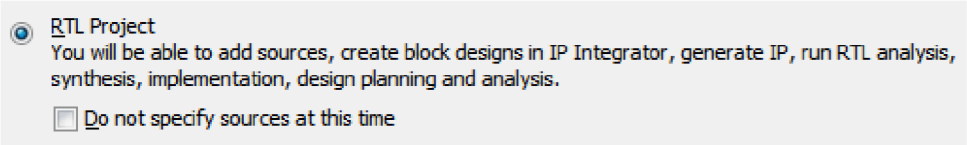
Make sure that you select the option to ***Create project subdirectory***. All options should be the same as shown below:



Click ***Next***.

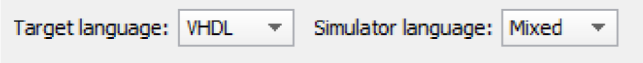
A directory named ***Zynq_Book*** will be created on your ***C drive*** if it did not already exist.

(e) At the *Project Type* dialogue, select ***RTL Project*** and ensure that the option ***Do not specify sources at this time*** is not selected:



Click ***Next***.

(f) Select ***VHDL*** as the ***Target language*** and ***Mixed*** as the ***Simulator Language*** in the *Add Sources* dialogue:



If existing sources, in the form of HDL or netlist files, were to be added to the project they could be imported at this stage.

As we do not have any sources to add to the project, click ***Next***.

(g) The *Add Existing IP (optional)* dialogue will open.

If existing IP sources were to be included in the project, they could be added here.

As we do not have any existing IP to add, click ***Next***.

(h) The *Add Constraints (optional)* dialogue will open.

This is the stage where any physical or timing constraints files could be added to the project.

As we do not have any constraints files to add, click ***Next***.

(i) From the *Default Part* dialogue window,

Zed

Select ***Boards*** from the *Select* dialogue, click ***ZedBoard Zynq Evaluation and Development Kit*** from the *Display Name* list and **All** from the *Board Rev* list, as shown in Figure 1.3. Select the appropriate revision for your board (in this case **Rev. D** has been selected).

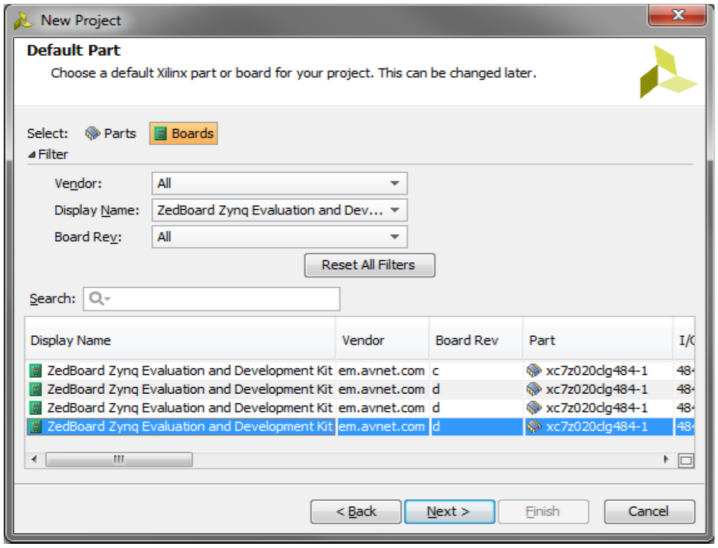


Figure 1.3: Zedboard Default Part Dialogue Options

Click ***Next***.

Zybo Select ***Boards*** from the *Select* dialogue, click ***Zybo*** from the *Display Name* list and **All** from the *Board Rev* list, as shown in Figure 1.4. Select the appropriate revision for your board (in this case **Rev. B.3** has been selected).

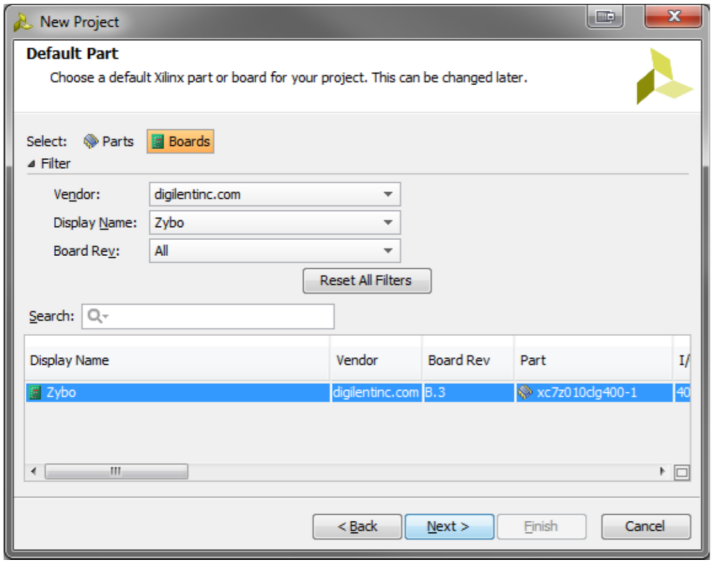


Figure 1.4: Zybo Default Part Dialogue Options

Click ***Next***.

Resume (j) In the *New Project Summary* dialogue, review the specified options, and click ***Finish*** to create the project.

Now that we have created our first project in Vivado IDE, we can now move on to creating our first Zynq embedded system design.

Before doing that, the Vivado IDE tool layout should be introduced. The default Vivado IDE environment layout is shown in Figure 1.5 (other layouts can be chosen by selecting different perspectives). This layout is specifically targeted for the Zedboard. If you are using the Zybo, you will see a slightly different layout.

With reference to the numbered labels in Figure 1.5, the main components of the Vivado IDE environment are:

1. **Menu Bar** - The main access bar gives access to the Vivado IDE commands.
2. **Main Toolbar** - The main toolbar provides easy access to the most commonly used Vivado IDE commands. Tooltips provide information about each command on the toolbar and these can be viewed by hovering the mouse pointer over the buttons, as shown in Figure 1.6.

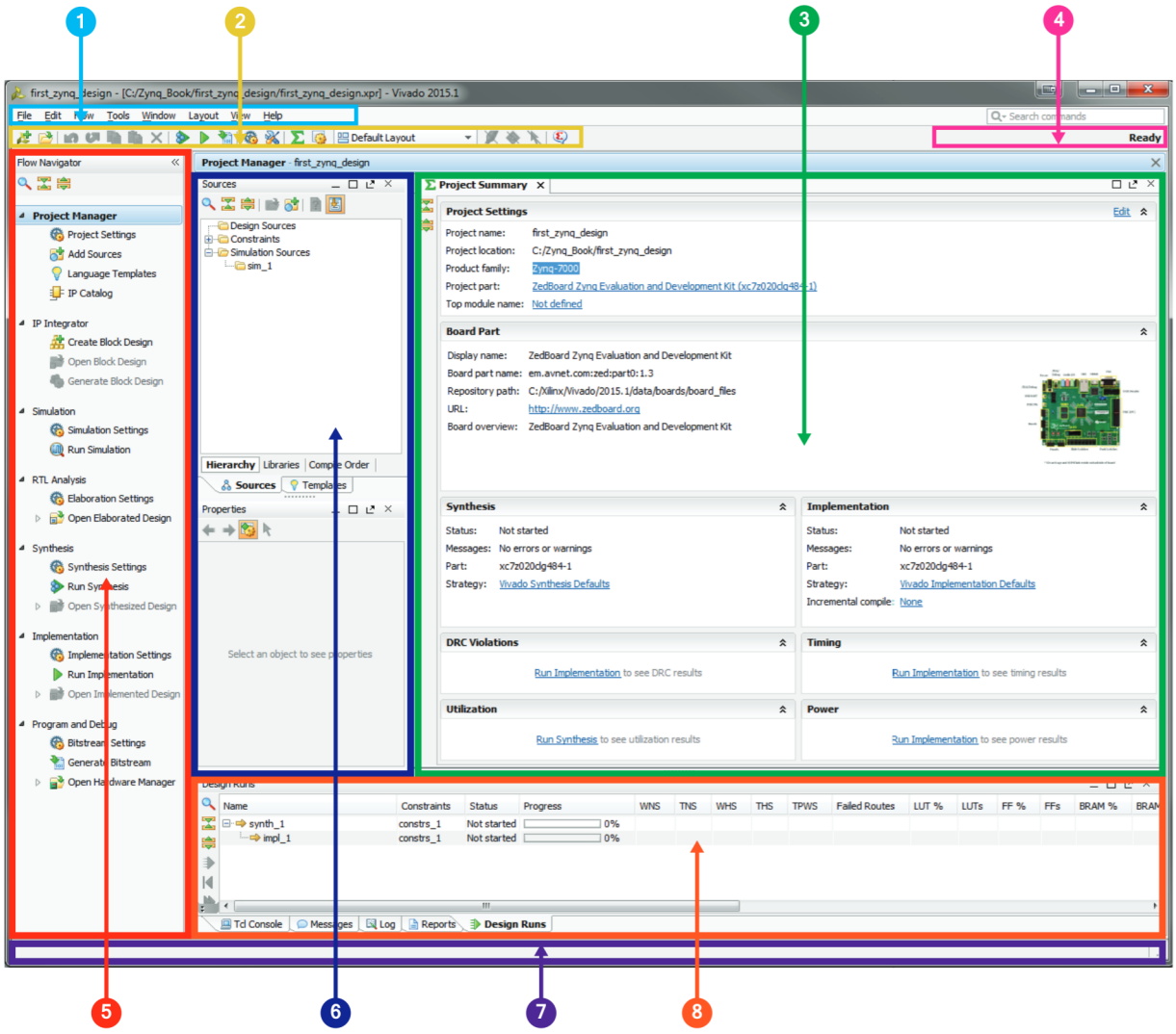


Figure 1.5: Vivado IDE Environment Layout (Zedboard)

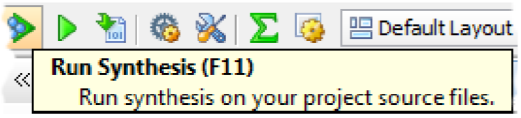


Figure 1.6: Toolbar tooltips

3. **Workspace** - The workspace provides a larger area for panels which require a greater screen space and those with a graphical interface, such as:

   - Schematic panel
   - Device panel
   - Package panel

- Text editor panel

4. **Project Status Bar** - The project status bar displays the status of the currently active design.

5. **Flow Navigator** - The Flow Navigator provides easy access to the tools and commands that are necessary to guide your design from start to finish, starting in the *Project Manager* section with design entry and ending with bitstream generation in the *Program and Debug* section. Run commands are available in the *Simulation*, *Synthesis* and *Implementation* sections to simulate, synthesise and implement the active design.

6. **Data Windows Pane** -The Data Windows pane, by default, displays information that relates to design data and sources, including:

   - **Properties window** - Shows information about selected logic objects or device resources.
   - **Netlist window** - Provides a hierarchical view of the synthesised or elaborated logic design.
   - **Sources window** - Shows IP Sources, Hierarchy, Libraries and Compile Order views.

7. **Status Bar** - The status bar displays a variety of information, including:

   - Detailed information regarding menu bar and toolbar commands will be shown in the lower left side of the status bar when the command is accessed.
   - When hovering over an object in the Schematic window with the mouse pointer, the object details appear in the status bar.
   - During constraint and placement creation in the Device and Package windows, validity and constraint type will be shown on the left side of the status bar. Site coordinates and type will be shown in the right side.
   - The task progress of a running task will be relocated to the right side of the status bar when the **Background** button is selected.

8. **Results Window Area** -The Results Window displays the status and results of commands in a set of windows grouped in the bottom of the Vivado IDE environment. As commands progress, messages are generated and log files and reports are created. The related information is shown here. The default windows are:

   - **Messages** - Displays all messages for the active design.
   - **Tcl Console** - Tcl commands can be entered here and a history of previous commands and outputs are also available.
   - **Reports** - Quick access is provided to the reports generated throughout the design flow.

- **Log** -Displays the log files generated by the simulation, synthesis and implementation processes.
- **Design Runs** -Manages runs for the current project.

Additional windows that can appear in this area as required are: Find Results window, Timing Results window and Package Pins window.

With the layout of the Vivado IDE environment introduced, we can now move on to creating the Zynq system.

Exercise 1B Creating a Zynq System in Vivado

In this exercise we will create a simple Zynq embedded system which implements a General Purpose Input/Output (GPIO) controller in the PL of the Zynq device. The GPIO controller will connect to the LEDs. It will also be connected to the Zynq processor via an AXI bus connection, allowing the LEDs to be controlled by a software application which we will create in Exercise 1C.

A graphical representation of the Zynq embedded design is provided in Figure 1.7.

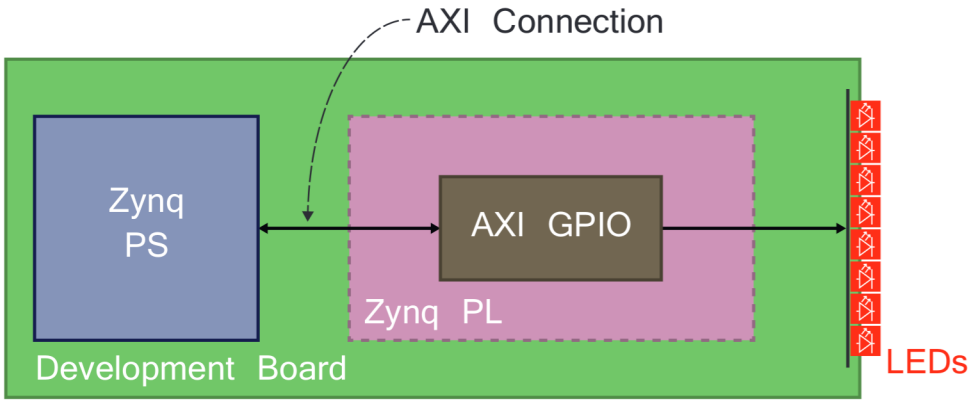


Figure 1.7: Zynq Embedded Design for Exercise 1B

We will begin by creating a new Block Design in Vivado IDE.

(a) In the *Flow Navigator* window, select ***Create Block Design*** from the *IP Integrator* section, as in Figure 1.8:

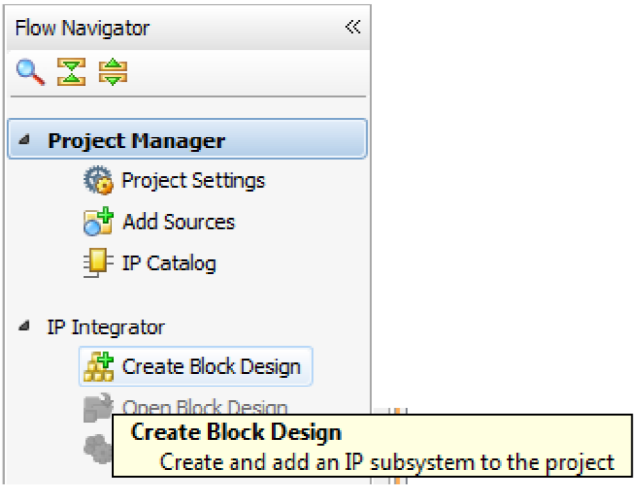


Figure 1.8: Creating a new Block Design in Flow Navigator

The *Create Block Design* dialogue will open.

(b) Enter ***first_zynq_system*** in the Design name box, as in Figure 1.9:

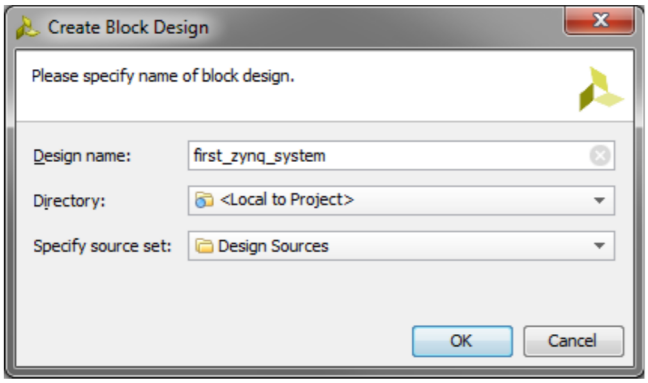


Figure 1.9: Create Block Design dialogue

Click ***OK***. The *Vivado IP Integrator Diagram* canvas will open in the *Workspace*.

The first block that we will add to our design will be a Zynq Processing System.

(c) In the *Vivado IP Integrator Diagram* canvas, right-click anywhere and select ***Add IP***, as in Figure 1.10.

Alternatively, select the ***Add IP*** button in the toolbar at the left of the canvas, shown in Figure 1.11.

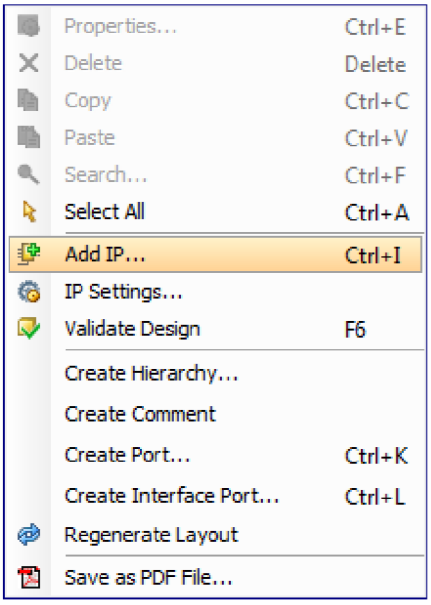


Figure 1.10: Add IP Option

Figure 1.11: Add IP option in IP Integrator canvas information message

The pop-up IP Catalog window will open, as in Figure 1.12.

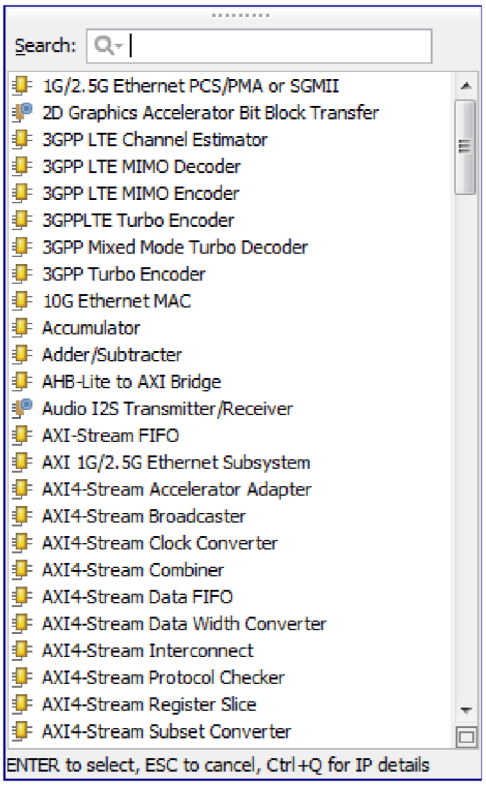


Figure 1.12: Pop-up IP Catalog Window

(d) Enter ***zynq*** in the search field and select the ***ZYNQ7 Processing System***, as shown in Figure 1.13. Be careful not to select the **BFM** version and press the ***Enter*** key on your keyboard.

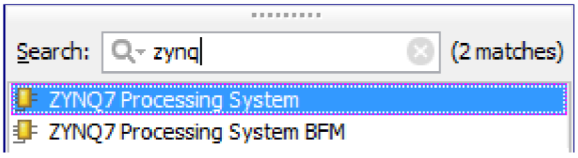


Figure 1.13: Adding ZYNQ7 Processing System from IP CatalogCatalog

You should see a similar message to the following in the *Tcl Console* window to confirm that the processing system has indeed been added to the design correctly:

```
create_bd_cell -type ip -vlnv xilinx.com:ip:processing_system7:5.5
processing_system7_0
```

Messages like this will be displayed in the *Tcl Console* window for all actions carried out on IP Integrator blocks.

The next step is to connect the **DDR** and **FIXED_IO** interface ports on the Zynq PS to the top-level interface ports on the design.

(e) Click the ***Run Block Automation*** option from the *Designer Assistance* message at the top of the Diagram window, as shown in Figure 1.14.

Figure 1.14: Run Block Automation - Processing System

Zed

In the *Run Block Automation* dialogue, ensure that the option to ***Apply Board Preset*** is selected and click ***OK***. The external connections for both the **DDR** and **FIXED_IO** interfaces will now be generated.

Your block diagram should now resemble Figure 1.15.

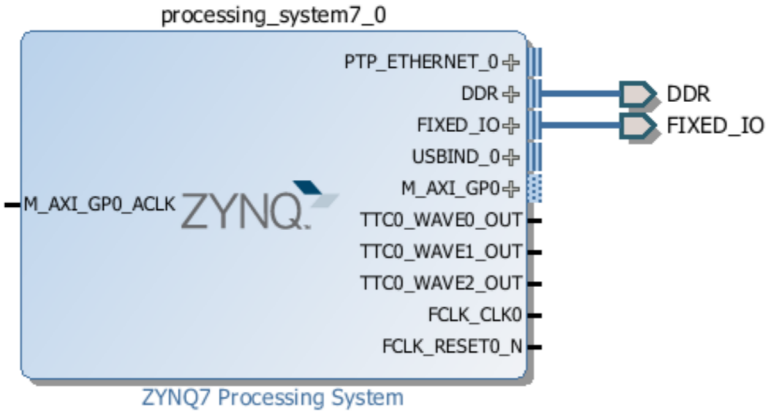


Figure 1.15: Zedboard ZYNQ7 Processing System External Connections

As the ***ZedBoard*** platform is the target development board, and this was specified on creation of the project, Vivado will configure the Zynq processor block accordingly.

Zybo

In the *Run Block Automation* dialogue, ensure that the option to ***Apply Board Preset*** is selected and click ***OK***. The external connections for both the **DDR** and **FIXED_IO** interfaces will now be generated.

Your block diagram should now resemble Figure 1.16.

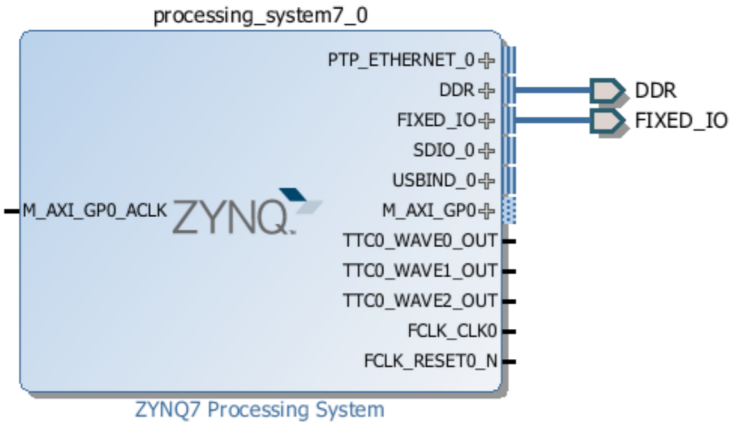


Figure 1.16: Zybo ZYNQ7 Processing System External Connections

As the ***Zybo*** platform is the target development board, and this was specified on creation of the project, Vivado will configure the Zynq processor block accordingly.

Resume Now that the main Zynq PS has been added to our design and configured, we can now add further blocks which will be placed in the PL to add functionality to the system. In this case we will only be adding a single block, **AXI GPIO**, to allow us to access the **LEDs** on the development board.

(f) Right-click in an empty area of the *Diagram* window and select ***Add IP***. Enter ***GPIO*** in the search field and add an instance of the ***AXI GPIO*** IP.

We will now use the *IP Integrator Designer Assistance* tool to automate the connection of the **AXI GPIO** block to the **ZYNQ7 Processing System**.

(g) Click ***Run Connection Automation*** from the *Designer Assistance* message at the top of the *Diagram* window and select ***/axi_gpio_0/S_AXI***, as shown Figure 1.17.

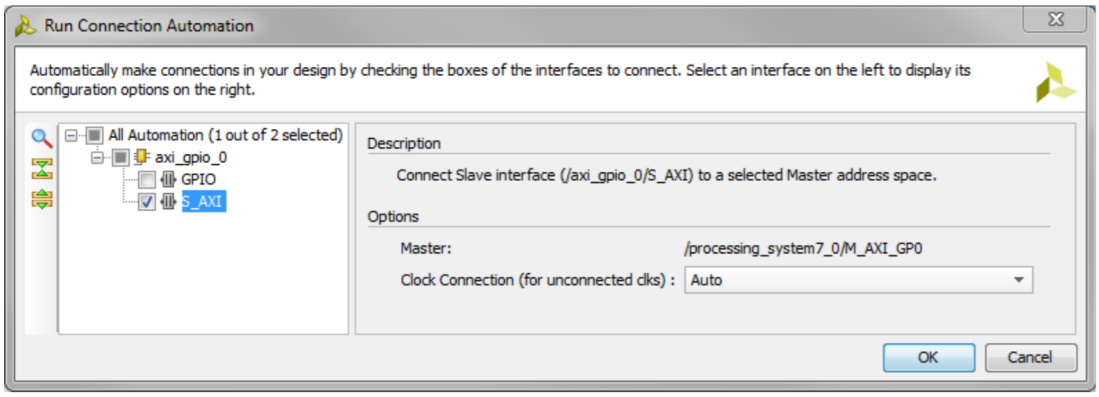


Figure 1.17: Run Block Automation - GPIO Block

This will automate the process of connecting the GPIO to an AXI port, and will automatically instantiate two further IP blocks:

- **Processor System Reset Module** - This provides customised resets for an entire processing system, including the peripherals, interconnect and the processor itself.
- **AXI Interconnect** - Provides an AXI interconnect for the system, allowing further IP and peripherals in the PL to communicate with the main processing system.

Click ***OK***.

All connections between the blocks should be made automatically.

(h) One final connection is required to connect the **AXI GPIO** block to the **LEDs** on the development board. This can also be completed using *Designer Assistance*.

Zed

Click ***Run Connection Automation*** from the *Designer Automation* message at the top of the *Diagram* window. The Run Connection Automation dialogue will open, as shown in Figure 1.18.

Select ***/axi_gpio_0/GPIO***.

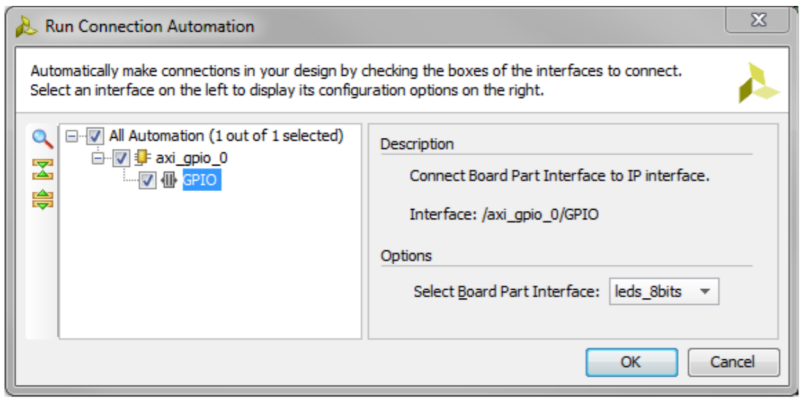


Figure 1.18: Zedboard Run Connection Automation Dialogue - GPIO Block

Select ***leds_8bits*** from the *Select Board Part Interface* drop-down menu, and click ***OK***.

The gpio interface of the AXI GPIO block will now be connected to the LEDs on the development board, and your complete design should resemble Figure 1.19.

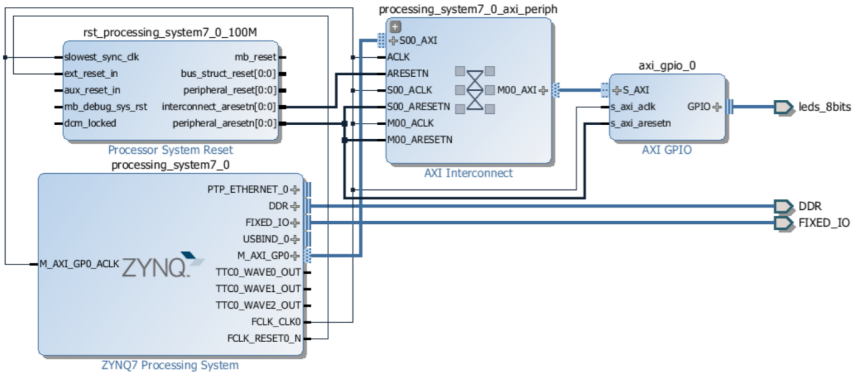


Figure 1.19: Zedboard, Zynq Processor System

The positions of the individual IP blocks in your design may vary slightly from Figure 1.19, but the blocks and their connections should be the same.

Zybo

Click ***Run Connection Automation*** from the *Designer Automation* message at the top of the *Diagram* window. The Run Connection Automation dialogue will open, as in Figure 1.20.

Select ***/axi_gpio_0/GPIO***.

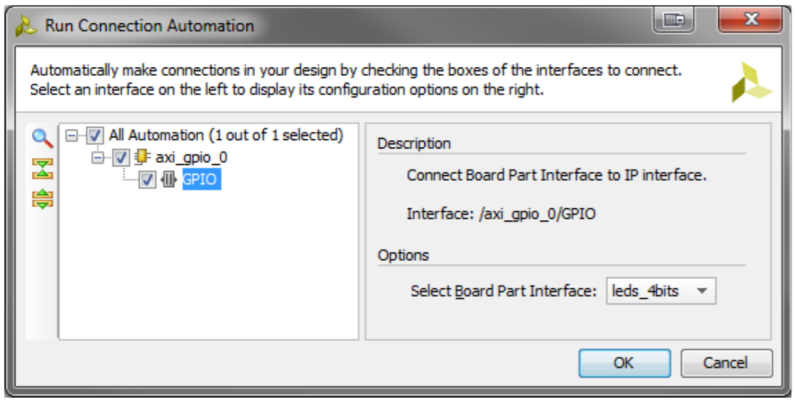


Figure 1.20: Zybo Run Connection Automation Dialogue - GPIO Block

Select ***leds_4bits*** from the *Select Board Part Interface* drop-down menu, and click ***OK***.

The gpio interface of the AXI GPIO block will now be connected to the LEDs on the development board, and your complete design should resemble Figure 1.21.

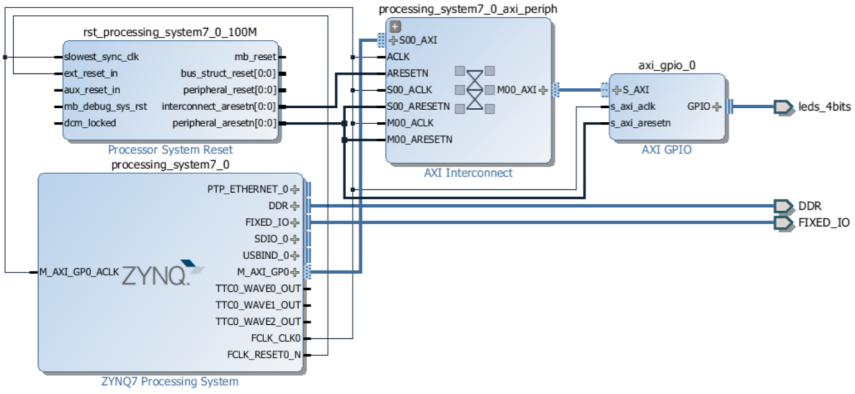


Figure 1.21: Zybo, Zynq Processor System

The positions of the individual IP blocks in your design may vary slightly from Figure 1.21, but the blocks and their connections should be the same.

Resume IP Integrator will automatically assign a memory map for all IP that is present in the design. We will not be changing the memory map in this tutorial, but for future reference we will take a look at the Address Editor.

(i) Select the Address Editor tab from the top of the Workspace window, as shown in Figure 1.22, and expand the Data group.

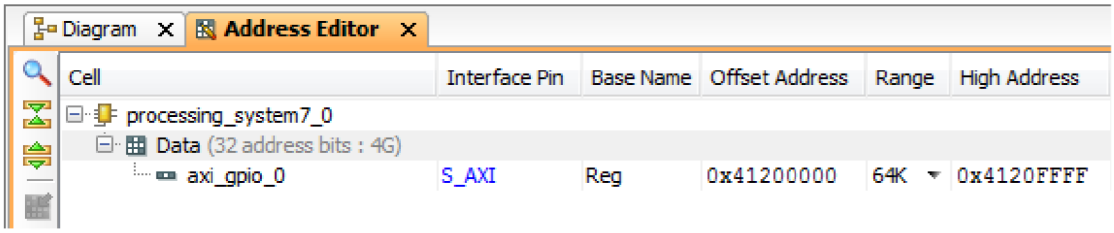


Figure 1.22: Address Editor Tab

You can see that IP Integrator has already assigned a memory map (the mapping of specific sections of memory to the memory-mapped registers of the IP blocks in the PL) to the AXI GPIO interface, and that it has a range of **64K**.

Now that our system is complete, we must first validate the design before generating the HDL design files.

(j) Save your design by selecting ***File > Save Block Design*** from the *Menu Bar*.

(k) Validate the design by selecting ***Tools > Validate Design*** from the *Menu Bar*. This will run a Design-Rule-Check (DRC).
Alternatively, select the Validate Design button, , from the Main Toolbar, or right-cick anywhere in the *Diagram* canvas and select ***Validate Design***.

(l) A *Validate Design* dialogue should appear to confirm that validation of the design was successful. Click ***OK*** to dismiss the message.

With the design successfully validated, we can now move on to generating the HDL design files for the system.

(m) Switch to the **Sources Tab** by selecting ***Window > Sources*** from the *Menu Bar*.

(n) In the **Sources** window, right-click on the top-level system design, which in this case is **first_zynq_system**, and select ***Create HDL Wrapper***, as shown in Figure 1.23.

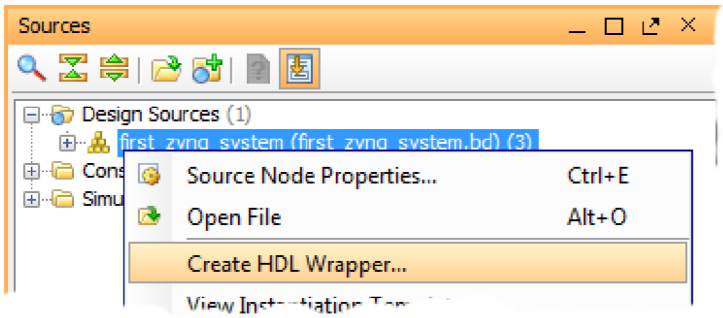


Figure 1.23: Create HDL Wrapper

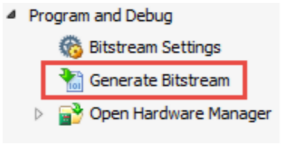
The *Create HDL Wrapper* dialogue window will open. Select *Let Vivado manage wrapper and auto-update*, and click ***OK***.

This will generate the top level HDL wrapper for our system.

All of the source files for the IP blocks that were used in the IP Integrator block diagram, as well as any relevant constraints files, will be generated during the synthesis process. As we specified VHDL as the target language when creating the project in Exercise 1A, all generated source files will be VHDL.

With all HDL design files generated, the next step in Vivado is to implement our design and generate a bitstream file.

(o) In *Flow Navigator*, click ***Generate Bitstream*** from the *Program and Debug* section. If a dialogue window appears prompting you to save your design, click ***Save***.



(p) A dialogue window will open requesting that you launch synthesis and implementation before starting the *Generate Bitstream* process. Click ***Yes*** to accept.

The combination of running the synthesis, implementation and bitstream generation processes back-to-back may take a few minutes, depending on the power of your computer system.

(q) Once the bitstream generation is finished, a dialogue window will open to inform you that the process has been completed successfully, as in Figure 1.24.

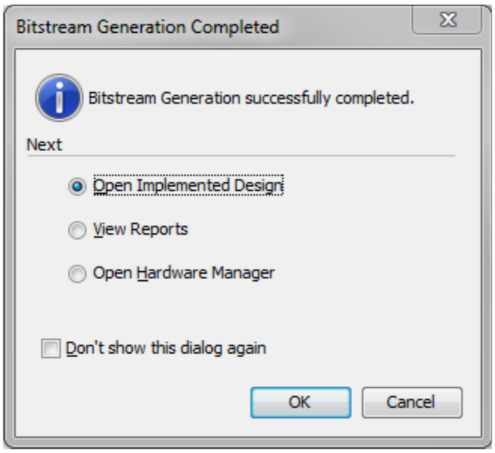


Figure 1.24: Bitstream Generation Completed Dialogue Window

Select ***Open Implemented Design***, and click ***OK***.

At this point you will be presented with the *Device* view, where you can see the PL resources that are utilised by the design. With the default colour scheme, these are shown in light blue.

With the bitstream generated, the building of the hardware image is complete. It must now be exported to a software environment where we will build a software application to control and interact with the custom hardware.

The final step in Vivado is to export the design to the SDK, where we will create the software application that will allow the Zynq PS to control the LEDs on the development board.

(r) Select ***File > Export > Export Hardware...*** from the *Menu Bar*.

(s) The *Export Hardware* dialogue window will open. Ensure that the option to ***Include bitstream*** is selected, as in Figure 1.25, and Click ***OK***.

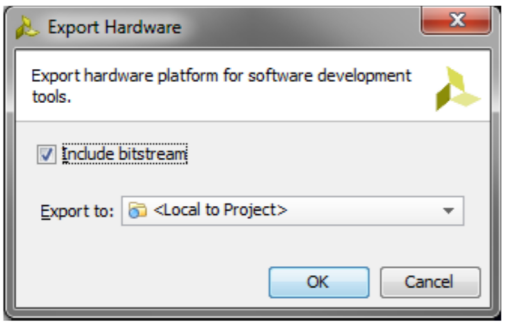


Figure 1.25: Export Hardware for SDK

NOTE: For the option to *Include bitstream* to be enabled, an implemented design must be active. This is the reason that we opened the implemented design in Step (q).

(t) Launch the SDK in Vivado by selecting ***File > Launch SDK*** from the *Menu Bar* and Click ***OK***.

This concludes the steps that are required in Vivado IDE. All hardware components of the system have been configured and generated. In the next exercise we will move on to creating a simple software component which will control the system.

Exercise 1C Creating a Software Application in the SDK

In this exercise we will create a simple software application which will control the LEDs on the Zynq development board. The software application will run on the Zynq processing system and communicate with the AXI GPIO block which is implemented in the PL. We will take a look at the software drivers that are created by IP Integrator for each of the IP modules, before building and executing the software on the development board.

The SDK should have opened after the conclusion of Exercise 1B. If it did not open, you can open the SDK by navigating to **Start > All Programs > Xilinx Design Tools > SDK 2015.1 > Xilinx SDK 2015.1**

When launching the SDK from the start menu, you will need to specify the workspace that was created when the Vivado IP Integrator design was exported in Exercise 1B. It should be:

C:\Zynq_Book\first_zynq_design\first_zynq_design.sdk

Enter this in the *Workspace* field of the *Workspace Launcher* dialogue window, as shown in Figure 1.26.

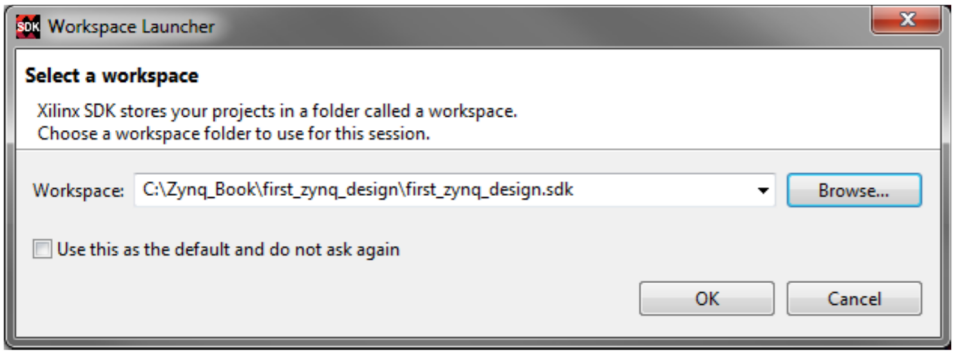


Figure 1.26: SDK Workspace Launcher Dialogue Window

With the SDK open, we can begin the creation of our software application. You will already be able to see the Hardware Platform Project, which will be automatically created and opened. It is now necessary to add an Application Project and a Board Support Package.

(a) Select **File > New > Application Project** from the *Menu bar*.

(b) The *New Project* dialogue window will open. Enter ***LED_test*** in the *Project name* field, as shown in Figure 1.27, keeping all other options with the default settings. Click ***Next*** (Be careful not to select *Finish*).

.

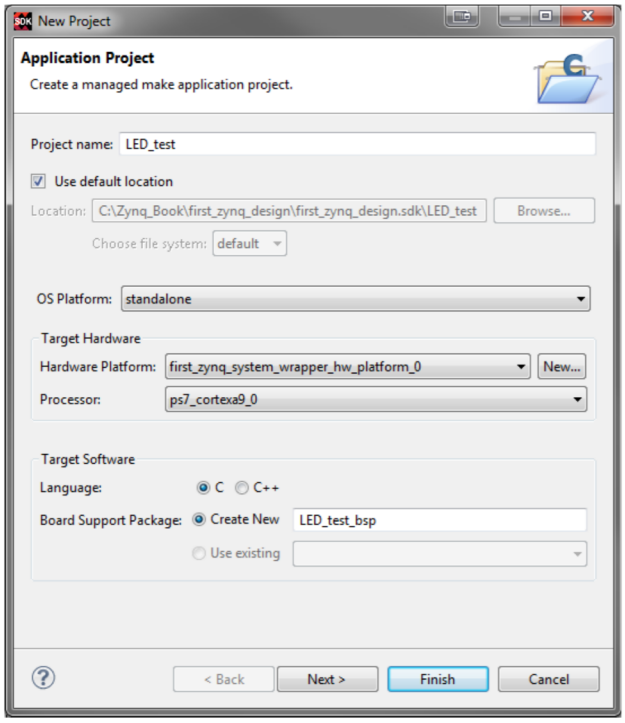


Figure 1.27: New Application Project Dialogue

(c) At the *New Project Templates* screen, select ***Empty Application***, as in Figure 1.28, and click ***Finish*** to create the project.

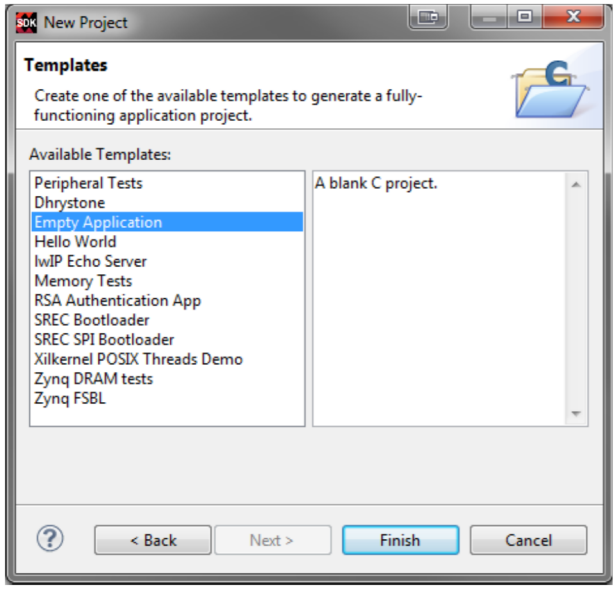


Figure 1.28: New Project Template Dialogue

NOTE: the new project should open automatically. If it doesn't, you may need to close the Welcome tab in order to view the project.

With the new Application Project created, we can now import some pre-prepared source code for the application.

(d) In the *Project Explorer* panel, expand **LED_test** and highlight the *src* directory. Right-click and select ***Import...***, as shown in Figure 1.29.

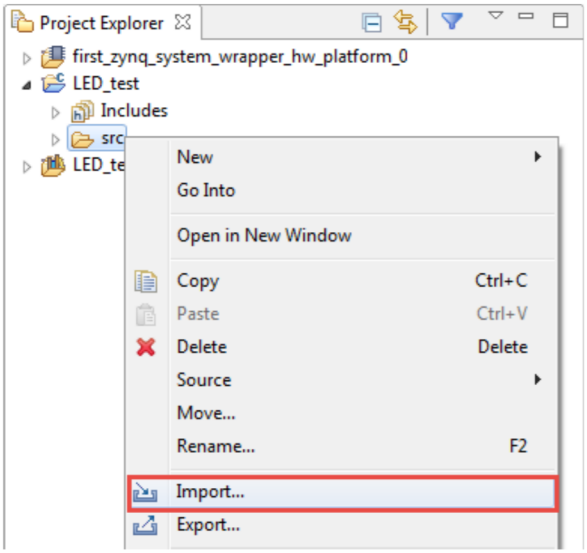


Figure 1.29: Import Source Files to Project

(e) The *Import* window will open. Expand the **General** option and highlight ***File System***, as in Figure 1.30, and click ***Next***.

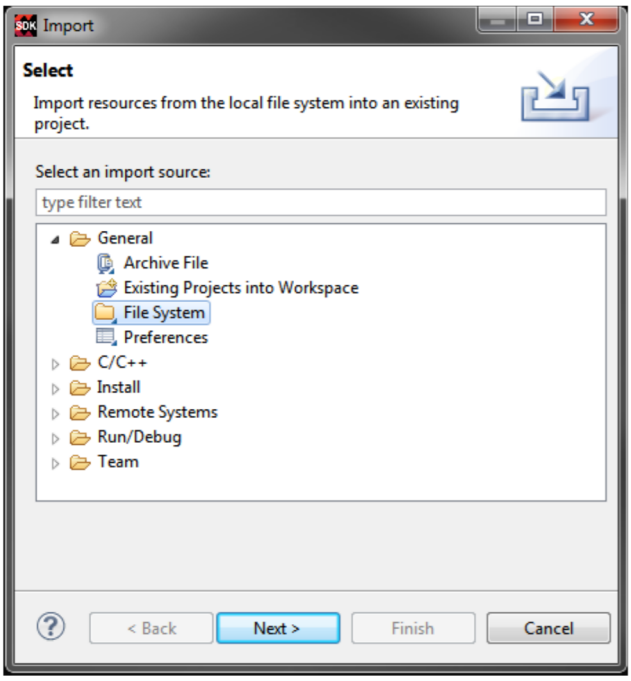


Figure 1.30: Import File System

(f) In the *Import File System* window, click the ***Browse...*** button.

The source file directory will depend on the Zynq development board that is in use. If you are using the **Zedboard**, navigate to: **C:\Zynq_Book\sources\zedboard\first_zynq_design.** If you are using the **Zybo**, navigate to **C:\Zynq_Book\sources\zybo\first_zynq_design**

Click ***OK.***

(g) Select the file **LED_test_tut_1C.c**, as shown in Figure 1.31, and click ***Finish***.

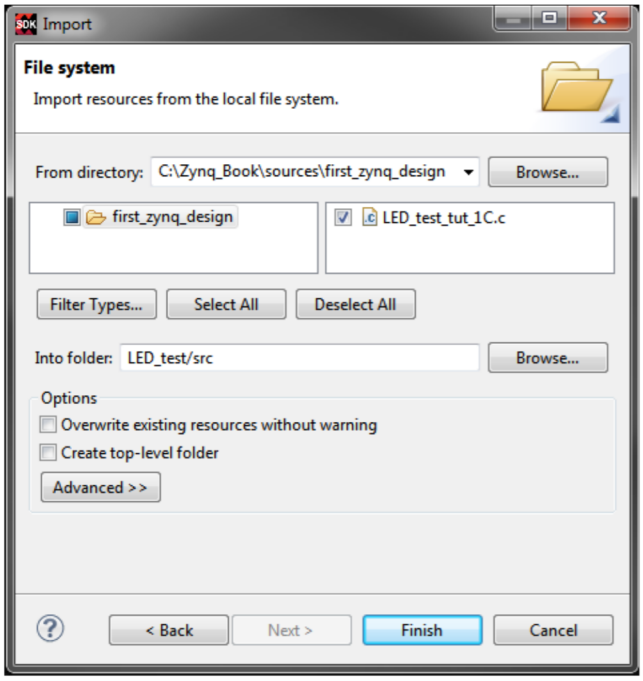


Figure 1.31: Import C Source File

The C source file will be imported and the project should automatically build. You should see a similar message to Figure 1.32 in the *Console* window.

Figure 1.32: Build Finished Console Message

(h) Open the imported source file by expanding the *src* folder and double-clicking on **LED_test_tut_1C.c**, and explore the code.

Note the command `XGpio_Initialize(&Gpio, GPIO_DEVICE_ID);` This is a function provided by the GPIO device driver in the file `xgpio.h`. It initialises the XGpio instance, *Gpio*, with the unique ID of the device specified by `GPIO_DEVICE_ID`.

If you look toward the top of the source file you will see that `GPIO_DEVICE_ID` is defined as `XPAR_AXI_GPIO_0_DEVICE_ID`. The value of `XPAR_AXI_GPIO_0_DEVICE_ID` can be found by opening the file, `xparameters.h`, which is automatically generated by Vivado IDE when exporting a hardware design to the SDK. It contains definitions of all the hardware parameters of the system.

The function, `XGpio_SetDataDirection(&Gpio, LED_CHANNEL, 0xFF);` is also provided by the GPIO device driver, and sets the direction of the specified GPIO port. As we are specifying the LEDs in this case, it is specifying an output. Bits set to '0' are output, and bits set to '1' are input. As there are 8 LEDs on the **Zedboard**, by setting the LED channel direction to a value of `0x00`, or `00000000` in binary, we are setting all 8 LEDs as outputs. Similarly as there are 4 LEDs on the **Zybo** board, by setting the LED channel direction a value of `0x0` or `0000` in binary, we are setting all 4 LEDs as outputs.

Further information on the peripheral drivers can be found by selecting the **system.mss** tab. A list of all the peripherals in the system is provided, along with links to available documentation and examples, as shown in Figure 1.33.

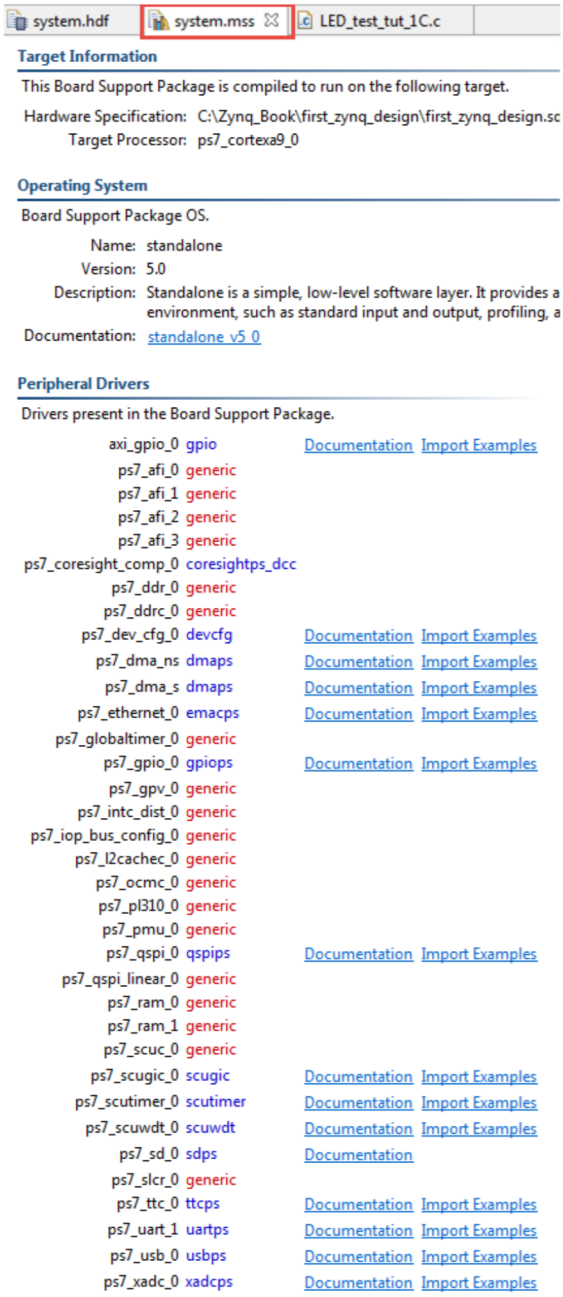


Figure 1.33: Peripheral Documentation and Drivers in system.mss tab

The next step is to program the Zynq PL with the bitstream file that we generated in Exercise 1B.

(i) Ensure that the Zynq development board is powered on and that the JTAG port is connected to the PC via the provided USB-A to USB-B cable. Additionally the board jumpers must also be correctly set so that to enable JTAG mode, which allows the hardware to be programmed and access for system debugging tools.

Zed The **Zedboard** requires five jumpers to be set as shown in Figure 1.34. This configuration will enable JTAG mode.

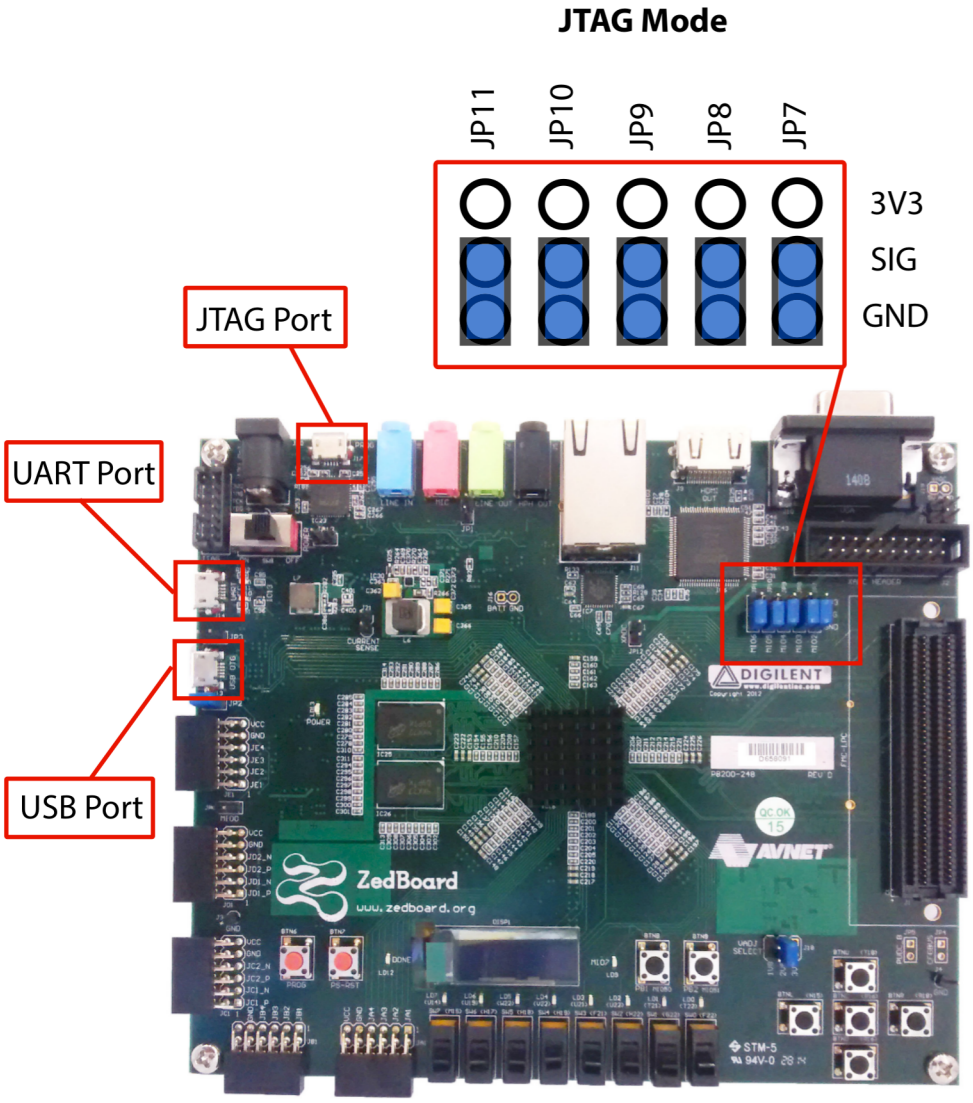


Figure 1.34: Zedboard JTAG Jumper Configuration

Zybo

One jumper is set to enable JTAG mode on the **Zybo** development board. Additionally the board's power supply is set using a jumper with the possibility of receiving power from USB, wall or battery. Both JTAG and power jumper configurations are set in Figure 1.35. The board has been set to receive power from USB.

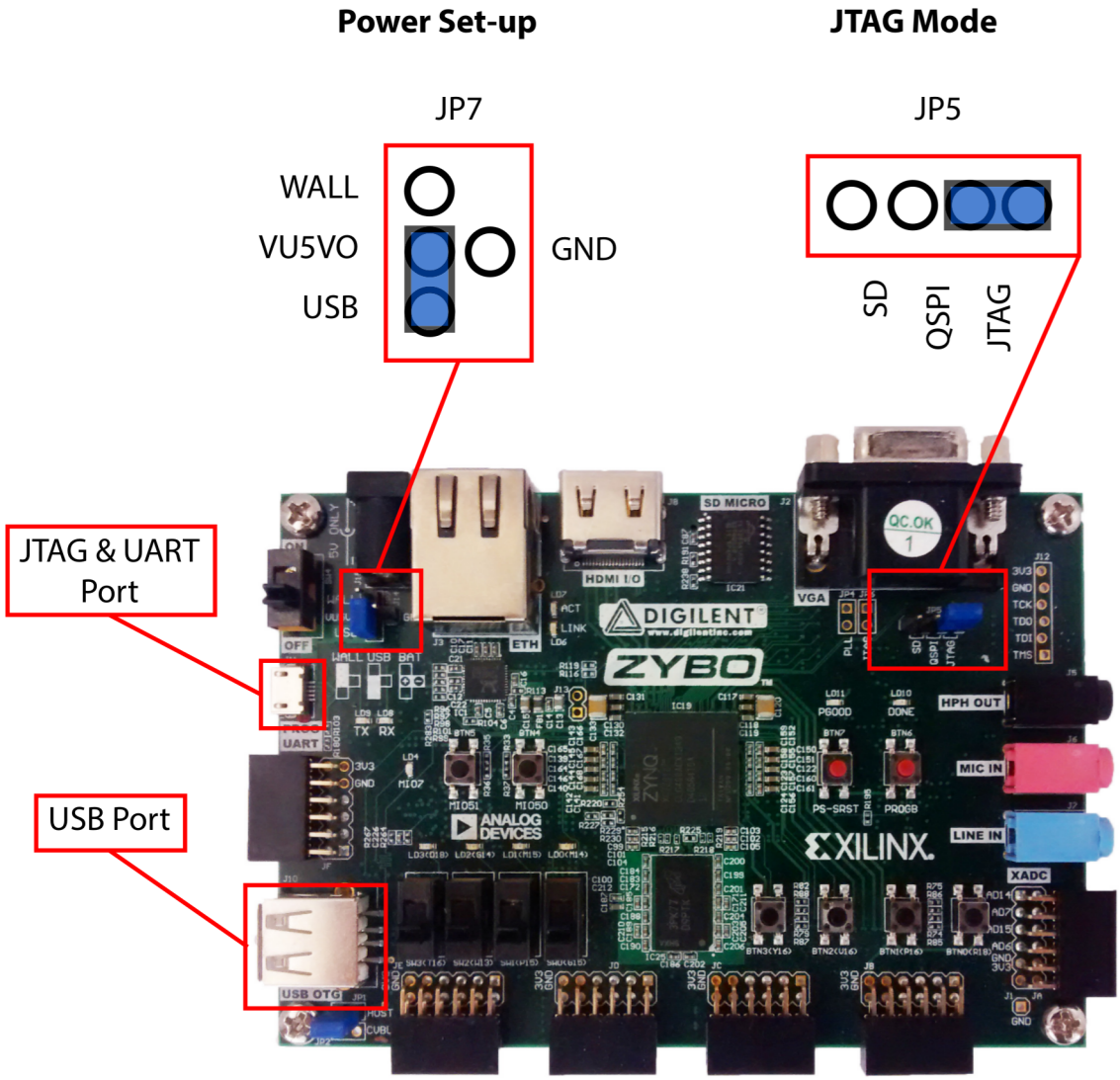


Figure 1.35: Zybo JTAG and Power Jumper Configurations

Resume (j) Download the bitstream to the Zynq PL by selecting **Xilinx Tools > Program FPGA** from the *Menu bar*. The *Program FPGA* window will appear. The Bitstream field should already be populated with the correct bitstream file, as in Figure 1.36.

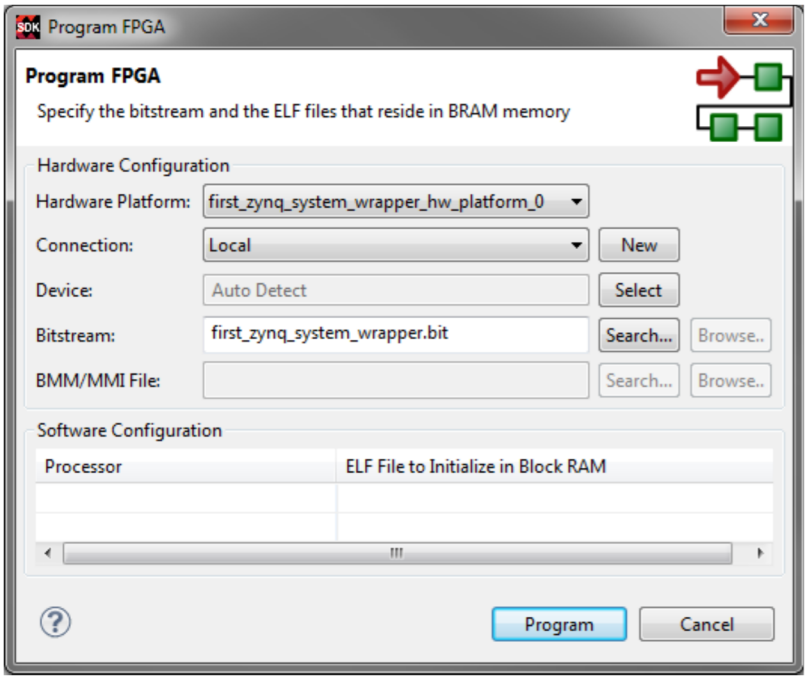


Figure 1.36: Program FPGA dialogue Window

NOTE: Once the device has successfully been programmed, the *DONE LED* on the ***ZedBoard*** will turn blue. Similarly the *DONE LED* on the ***Zybo*** will turn green.

With the Zynq PL successfully configured with the bitstream file, we can now launch our software application on the Zynq PS.

(k) Select the project ***LED_test*** in *Project Explorer*. Right-click and select **Run As > Launch on Hardware (GDB)** as in Figure 1.37.

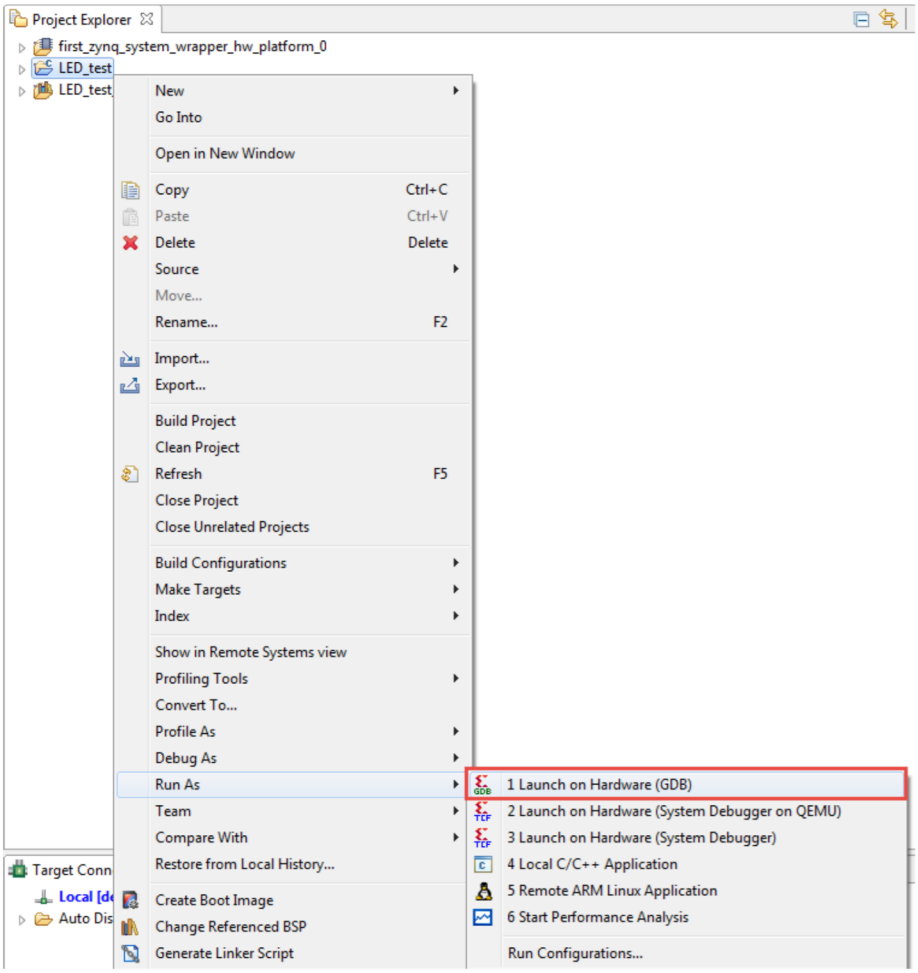


Figure 1.37: Launch Application onto the Zynq Development Board

Zed

After a few seconds the LEDs on the **ZedBoard** should begin to flash between the states highlighted in Figure 1.38.

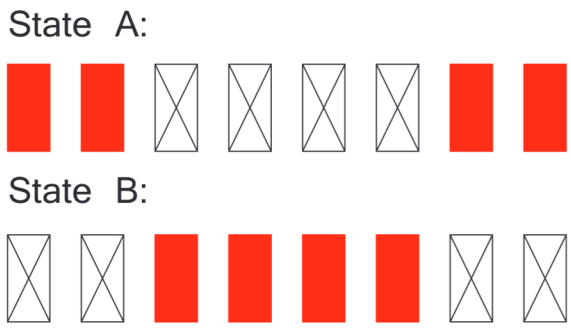


Figure 1.38: Zedboard LED Flashing States

Zybo The LEDs on the **Zybo** should begin to flash between the states highlighted in Figure 1.39

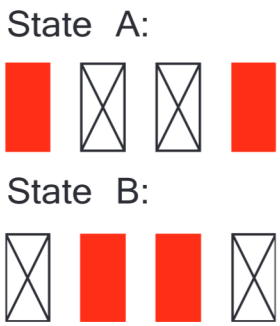


Figure 1.39: Zybo LED Flashing States

Resume You have successfully created and executed your first software application on the Zynq processing system.

In summary a GPIO controller has been successfully implemented in the FPGA fabric of the Zynq device, forming a connection between the Zynq Processing System and the development board LEDs via an AXI interface. The Zynq Processing System was then programmed to control the LEDs by means of a standalone software application with the capability to interface with the GPIO controller in the FPGA fabric.