Building an Embedded Processor System
on a Xilinx Zync FPGA (Profiling): A Tutorial

Embedded Processor Hardware Design
October 6th 2017.
# Table of Contents

Requirements ................................................................................................................. 3

**Part 1: Building a Zynq-7000 Processor Hardware** ......................................................... 4
Introduction ........................................................................................................................... 4
Step 1: Start the Vivado IDE and Create a Project .............................................................. 4
Step 2: Create an IP Integrator Design ............................................................................... 6
  - Customize Instantiated IP ............................................................................................. 10
  - Use Block Designer Assistance ................................................................................ 11
Step 3: Generate HDL Design Files .................................................................................. 15
Step 4: Implement Design and Generate Bitstream ......................................................... 16
Step 5: Export Hardware to SDK .................................................................................... 18
  - Export to SDK ............................................................................................................. 18

**Part 2: Build Zynq-7000 Processor Software** ............................................................... 19
Step 1: Start SDK and Create a Software Application .................................................... 19
Step 2: Run the Software Application ........................................................................... 22
  - Add a Breakpoint ....................................................................................................... 26
Step 3: Executing the Software ....................................................................................... 27

**Part 3: Profiling the Software** .................................................................................... 29
Step 1: Start SDK and Create a Software Application .................................................... 29
Step 2: Profile the Software Application ....................................................................... 30
  - Producing the GMON Executable ............................................................................. 30
Step 3: Generating statistics
Introduction

This tutorial will guide you through the process of using Vivado and IP Integrator to create a complete Zynq ARM Cortex-A9 based processor system targeting the ZedBoard Zynq development board. You will use the Block Design feature of IP Integrator to configure the Zynq PS and add IP to create the hardware system, and SDK to create an application to verify the design functionality. It will also guide you through the process of profiling an application and analyzing the output.

Objectives

After completing this tutorial, you will be able to:

- Create an embedded system design using Vivado and SDK flow
- Configure the Processing System (PS)
- Add Xilinx standard IP in the Programmable Logic (PL) section
- Use and route the GPIO signal of the PS into the PL using EMIO
- Use SDK to build a software project and verify the functionality in hardware.
- Set up the board support package (BSP) for profiling an application
- Set the necessary compiler directive on an application to enable profiling
- Setup the profiling parameters

Procedure

This lab is separated into steps that consist of general overview statements that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the tutorial.

This tutorial comprises three stages (each consisting of steps): You will create a top-level project using Vivado, create the processor system using the IP Integrator, add two instances of the GPIO IP, validate the design, generate the bitstream, export to the SDK, create an application in the SDK, and, test the design in hardware. You will then be able to profile the application and produce statistics that will help you understand the main bottlenecks of your application.

Requirements

The following is needed in order to follow this tutorial:

- Vivado w/ Xilinx SDK (tested, version 2013.2/version 2014.4). Version 2016.3 could work but with some hiccups.
- Zedboard (tested, version D)
Part 1: Building a Zynq-7000 Processor Hardware

Introduction
In this part of the tutorial you create a Zynq-7000 processor based design and instantiate IP in the processing logic fabric (PL) to complete your design. Then you take the design through implementation, generate a bitstream, and export the hardware to SDK.
If you are not familiar with the Vivado Integrated Development Environment Vivado (IDE), see the Vivado Design Suite User Guide: Using the Vivado IDE (UG893).

Step 1: Start the Vivado IDE and Create a Project
1. Start the Vivado IDE (FIGURE 1) by clicking the Vivado desktop icon or by typing vivado at a terminal command line.

![Figure 1: Getting Started Page](image-url)
2. From the Getting Started page, select **Create New Project**. The New Project wizard opens (FIGURE 2).

3. Click **Next**

![Create New Project Wizard](image)

**Figure 2: Create New Project Wizard**

4. In the **Project Name** dialog box, type the project name and location. Ensure that **Create project subdirectory** is checked, and then click **Next**.

5. In the **Project Type** dialog box, select **RTL Project**, then click **Next**.

6. In the **Add Sources** dialog box, ensure that the **Target language** is set to **VHDL**, then click **Next**.

7. In the **Add Existing IP** dialog box, click **Next**.

8. In the **Add Constraints** dialog box, click **Next**.

9. In the **Default Part** dialog box select **Boards** and choose “ZedBoard Zynq Evaluation and Development Kit”. Make sure that you have selected the proper Board Version to match your hardware because multiple versions of hardware are supported in the Vivado IDE. Click **Next**.

10. Review the project summary in the **New Project Summary** dialog box before clicking **Finish** to create the project.
Step 2: Create an IP Integrator Design

1. In the Flow Navigator, select Create Block Design.

   ![Create Block Design from Flow Navigator](Figure3.png)

   Figure 3: Create Block Design from Flow Navigator

2. In the Create Block Design popup menu, specify a name for your IP subsystem design.

   ![Create Block Design dialog box](Figure4.png)

   Figure 4: Create Block Design Dialog Box
3. Right-click in the Vivado IP integrator diagram window, and select Add IP.

![Figure 5: Add IP Option](image)

4. Alternatively, you can click the Add IP link in the IP integrator diagram area.

![Figure 6: Add IP Link in IP Integrator Canvas](image)

The IP Catalog opens.

5. In the search field, type zynq to find the ZYNQ7 Processing System IP, and then press Enter on the keyboard.

![Figure 7: The IP Integrator IP Catalog](image)

Because you selected the ZedBoard when you created the project, the Vivado IP integrator configures the design appropriately.
In the Tcl Console, you see the following message:

```
create_bd_cell -type ip -vlnv xilinx.com:ip:processing_system7:5.2 processing_system7_1
INFO: [PS7-6] Configuring Board Preset zed. Please wait ......
```

There is a corresponding Tcl command for all actions performed in the IP integrator block diagram. Those commands are not shown in this document. See the Tcl Console for information on those commands.

6. In the IP integrator diagram header, click **Run Block Automation**.

![Figure 8: Run Block Automation on Zynq](image)

The **Run Block Automation** dialog box opens, stating that the FIXED_IO and DDR interfaces will be created for the Zynq core.

7. Click **OK**.

![Figure 9: Zync7 Run Block Automation Dialog Box](image)
After running block automation on the Zynq processor, the IP integrator diagram should look as follows:

8. Now you can add peripherals to the processing logic (PL). To do this, right-click in the IP integrator diagram, and select **Add IP**.

9. In the search field, type `gpi` to find the AXI GPIO IP, and then press **Enter** to add the AXI GPIO IP to the design.

10. Repeat the action, typing `axi bram` to find and add AXI BRAM Controller, and typing `block` to find and add Block Memory Generator.

The Block Design window matches FIGURE 11. The relative positions of the IP will vary.
Customize Instantiated IP

1. Double-click the Block Memory Generator IP, or right-click and select **Customize Block** (FIGURE 12).

![Figure 12: Customize Block Option](image)

2. On the Basic tab of the dialog box, set:
   - **Mode** to **BRAM Controller**
   - **Memory Type** to **True Dual Port RAM**

   Click **OK**.

![Figure 13: Set Mode and Memory Type](image)

The AXI BRAM Controller provides an AXI memory map interface to the Block Memory Generator.
3. Connect the Block Memory Generator to the AXI4 BRAM Controller by clicking the connection point and dragging a line between the IP.

The AXI BRAM Controller provides an AXI memory map interface to the Block Memory Generator.

**Use Block Designer Assistance**
Block Designer Assistance helps connect the AXI GPIO and AXI BRAM Controller to the Zynq-7000 PS.

1. Click **Run Connection Automation** and then select `/axi_gpio_1/s_axi` to connect the BRAM controller and GPIO IP to the Zynq PS and to the external pins on the ZedBoard (FIGURE 15).

The Run Connection Automation dialog box opens and states that it will connect the master AXI interface to a slave interface.
In this case, the master is the Zynq Processing System IP (FIGURE 16).

![Figure 16: Run Connection Automation Message](image)

Click **OK**.

This action instantiates an AXI Interconnect IP as well as a Proc Sys Reset IP and makes the interconnection between the AXI interface of the GPIO and the Zynq-7000 PS.

2. Select **Run Connection Automation** again, and the `/axi_gpio_1/gpio` shown in FIGURE 17.

![Figure 17: axi_gpio Selection](image)

The Run Connection Automation dialog box includes options to hook up to the GPIO port.

3. Select `leds_8bits` (FIGURE 18).

![Figure 18: Select Board Interface Options](image)
4. Click **OK**. This step also configures the IP so that during netlist generation, the IP creates the necessary Xilinx Design Constraints (XDC).

5. Click **Run Connection Automation** again, and select the remaining option `/axi_bram_ctrl_1/S_AXI` (FIGURE 19).

![Figure 19: axi_bram_ctrl Selection](image)

This completes the connection between the Zynq7 Processing System and the AXI BRAM Controller.

The IP integrator subsystem looks like FIGURE 20. Again, the relative positions of the IP can differ slightly.

![Figure 20: Zynq Processor System](image)

6. Click the Address Editor tab to show the memory map of all the IP in the design.

In this case, there are two IP: the AXI GPIO and the AXI BRAM Controller. The IP integrator assigns the memory maps for these IP automatically. You can change them if necessary.
7. Change the range of the AXI BRAM Controller to **64K**, as shown in FIGURE 21.

![Figure 21: axi_bram_ctrl to 64k Range](image)

8. Save your design by pressing Ctrl-S, or select File > Save Block Design.

9. Click the Address Editor tab to make sure that the memory mappings for the GPIO and BRAM controller have been auto populated.

10. From the toolbar, run Design-Rules-Check (DRC) by clicking the **Validate Design** button (FIGURE 22). Alternatively, you can do the same from the menu by:

   - Selecting **Tools > Validate Design** from the menu.
   - Right-clicking in the Diagram window and selecting **Validate Design**.

![Figure 22: Validate Design Button](image)

The Validate Design Successful dialog box opens (FIGURE 23).

![Figure 23: Validate Design Message](image)

11. Click **OK**.
Step 3: Generate HDL Design Files
You now generate the HDL files for the design.

1. In the Source window, right-click the top-level subsystem design and select **Generate Output Products** (FIGURE 24). This generates the source files for the IP used in the block diagram and the relevant constraints file.

   ![Figure 24: Generate Output Products Option](image)

2. The **Manage Output Products** dialog box opens. Click **OK**.

   ![Figure 25: Create HDL Wrapper](image)

3. In the Sources window, select the top-level subsystem source, and select **Create HDL Wrapper** to create an example top-level HDL file (FIGURE 25).

4. Click **OK** when the **Create HDL Wrapper** dialog box opens.
Step 4: Implement Design and Generate Bitstream

1. In Flow Navigator, click **Generate Bitstream** to implement the design and generate a BIT file.

   *Note:* If the system requests to re-synthesize the design before implementing, click **No**. The previous step of saving the constraints caused the flow to mark synthesis out-of-date. Ordinarily, you might want to re-synthesize the design if you manually changed the constraints, but for this tutorial, it is safe to ignore this condition (FIGURE 26).

![Generate Bitstream](image)

Figure 26: Generate Bitstream

You might see a dialog box stating no implementation results are available.

2. Click **Yes**.

![No Implementation Results Available Dialog Box](image)

Figure 27: No Implementation Results Available Dialog Box
3. After the design implementation, click **Open Implemented Design**, (FIGURE 28).

![Bitstream Generation Completed](image)

**Figure 28: Bitstream Generation Completed**

4. You might get a warning that the implementation is out of date. Click **Yes**.

![Implementation Is Out-of-Date](image)

**Figure 29: Implementation Is Out-of-Date Dialog Box**
**Step 5: Export Hardware to SDK**

In this step, you export the hardware description to SDK. You use this in Part 2. The IP integrator block diagram, and the Implemented design, must be open to export the design to SDK.

**IMPORTANT:** For the Digilent driver to install, you must power on and connect the board to the host PC before launching SDK.

**Export to SDK**

1. In the Flow Navigator, click **Open Block** to invoke the IP integrator design (FIGURE 30).

![Figure 30: IP Integrator - Open Block Design](image)

Now you are ready to export your design to SDK.

2. From the main Vivado File menu, select Export Hardware for SDK (FIGURE 31).

![Figure 31: Export Hardware for SDK](image)

The Export Hardware for SDK dialog box opens, ensure that Export Hardware, Include Bitstream, and Launch SDK are checked (FIGURE 32).

![Figure 32: Export Hardware for SDK](image)
Part 2: Build Zynq-7000 Processor Software

In this portion of the tutorial you will build an embedded software project that prints “Hello World” to the serial port.

**Step 1: Start SDK and Create a Software Application**

1. If you are doing this lab as a continuation of Part 1 then SDK should have launched in a separate window (if you checked the Launch SDK option while exporting hardware). You can also start SDK from the Windows Start menu by clicking on Start > All Programs > Xilinx Design Tools > Vivado 2013.2 > SDK > Xilinx SDK 2013.2. When starting SDK in this manner you need to ensure that you in the correct workspace.

2. You can do that by clicking on File > Switch Workspace > Other in SDK. In the Workspace Launcher dialog box in the Workspace field, point to the SDK_Export folder where you had exported your hardware. Usually, this is located at ..\project_name\project_name.sdk\SDK\SDK_Export.

Now you can create a hello world application.

3. Select File > New > Application Project (FIGURE 33).

New Project dialog box opens
4. In the Project Name field, type **Zync_Design**, and click **Next** (FIGURE 34).
5. From the Available Templates, select **Hello World** (FIGURE 35) and click **Finish**.

When the program finish compiling, you will see the following (FIGURE 36).

![Figure 35: SDK New Project Template](image)

![Figure 36: SDK Message](image)
Step 2: Run the Software Application

Now, you must run the hello world application on the ZedBoard. Make sure that your hardware is powered on and a USB Cable is connected to the host PC. Also, ensure that you have a USB cable connected to the UART port of the ZedBoard.

1. Download the bitstream into the FPGA by selecting Xilinx Tools > Program FPGA (FIGURE 37).

   ![Program FPGA](image)

   Figure 37: Program FPGA

   This opens the Program FPGA dialog box.

2. Ensure that the path to the bitstream that you created in this tutorial is correct and then click Program.

   Note: The DONE LED on the board turns blue if the programming is successful.

3. Select and right-click the Zynq_Design application.

4. Select Debug As and Debug Configurations (FIGURE 38).
5. In the Debug Configurations dialog box, right-click **Xilinx C/C++ Application (GDB)** and select **New**.
6. In the Debug Configurations dialog box, click **Debug**.

7. The Confirm Perspective Switch dialog box opens. Click **Yes**.
8. Set the terminal by selecting the Terminal 1 tab and clicking the Settings button (FIGURE 42).

![Figure 42: Settings Button](image)

9. Use the following settings for the ZedBoard (FIGURE 43). Click OK.

![Figure 43: Terminal Settings](image)

The Port should be the port for the Cypress USB-to-Serial.

10. Verify the Terminal connection by checking the status at the top of the tab (FIGURE 44).

![Figure 44: Terminal Connection Verification](image)
11. In the **Debug** tab, expand the tree, and select the processor core on which the program is to be run (FIGURE 45).

![Figure 45: Processor Core to Debug](image)

12. If it is not already open, select `./src/helloworld.c`, line 41, and double click that line to open the source file.

**Add a Breakpoint**

You add a breakpoint on line 43.

1. Select **Navigate > Go To Line** (FIGURE 46).

![Figure 46: Go to Line](image)
2. In the Go To Line dialog box, type 43.

3. Double click on the left pane of line 43, which adds a breakpoint on that line of source code (Figure 47).

![Figure 47: Add a Breakpoint]

**Step 3: Executing the Software**

This step will take you through executing the code up to and past the break point.

1. Click the **Resume** button or press **F8**

2. Click the **Step Over** button or press **F6**

3. You should see “Hello World” in the terminal if everything worked correctly (FIGURE 48).

![Figure 48: Terminal Output]
Part 3: Profiling an Application

Export the Design to the SDK

Step 1

1-1. Export the design to the SDK, create the software BSP using the standalone operating system; Enable the profiling options.

1-1-1. Export the hardware configuration by clicking File > Export > Export Hardware

1-1-2. Tick the box to Include Bitstream, and click OK

1-1-3. Launch SDK by clicking File > Launch SDK and click OK


1-1-5. Notice Standalone_bsp_0 in the Project name field and click Finish with default settings.

A Board Support Package Settings window will appear.

1-1-6. Select the Overview > standalone entry in the left pane, click on the drop-down arrow of the enable_sw_intrusive_profiling Value field and select true.

![Figure 3.1 Enable profiling in the board support package](image)

1-1-7. Select the Overview > drivers > cpu_cortexa9 and add –pg in addition to the –g in the extra_compiler_flags Value field.
1-1-8. Click OK to accept the settings and create the BSP.

Create the Application

Step 2

2-1. Create the tutorial application.

2-1-1. Select File > New > Application Project.

2-1-2. Enter tutorial-profile as the project name, select the Use existing standalone_bsp_0 option, and click Next.


2-1-4. Replace the Hello World C program with the intended application you have.

2-2. A snippet of the source code is shown in the following figure.

![Figure 3.3 Adding profiling switch](image)
2-2-1. Save the program and it should compile successfully and generate the tutorial-profile.elf file.

Run the Application and Profile  

Step 3

3-1. Place the board into the JTAG boot up mode. Program the PL section and run the application.

3-1-1. Place the board in the JTAG boot up mode.

3-1-2. Power ON the board.

3-1-3. Select Xilinx Tools > Program FPGA and click on Program.

3-1-4. Right click on the tutorial-profile directory, and select C/C++ Build Settings.

3-1-5. Under the ARM gcc compiler group, select the Profiling sub-group, then check the Enable Profiling box, and click OK.

![Figure 3.4 Compiler setting for enabling profiling](image)

3-1-6. From the menu bar, Select Run > Run Configurations… and double click on Xilinx C/C++ application to create a new configuration.

3-1-7. Click on the newly created tutorial-profile Debug configuration, and select the Profile Options tab.

3-1-8. Click on the Enable Profiling check box, enter 100000 (100 kHz) in the Sampling Frequency field, enter 0x10000000 in the scratch memory address field, and click Apply.
3-1-9. Click the **Run** button to download the application and execute it.

The program will run, and when execution has completed, a message will be displayed indicating that the profiling results are being saved in gmon.out file at the tutorial-profile\Debug directory.

3-1-10. Click **OK**.

3-2. **Invoke gprof and analyze the results.**

3-2-1. Expand the *Debug* folder under the **tutorial-profile** project in the Project Explorer view, and double click on the **gmon.out** entry.

3-2-2. The Gmon File Viewer dialog box will appear showing **tutorial-profile.elf** as the corresponding binary file. Click **OK**.
3-2-3. Click on the **Sort samples per function** button.

3-2-4. Click in the **%Time** column to sort in the descending order (See Figure 3.7).

![Figure 3.7 Sorting results](image)

3-2-5. Go back to the **Run Configuration**, and change the sampling frequency to 1000000 (1 MHz) and profile the application again.

3-2-6. Invoke **gprof**, select the **Sorts samples per function** output, and sort the **%Time** column.

Notice that the output has better resolution and reports more functions and more samples per function calls.

3-2-7. Close the SDK and Vivado programs by selecting **File > Exit** in each program.

3-2-8. Turn OFF the power on the board.

**Conclusion**

This Tutorial led you through enabling the software BSP and the application settings for the profiling.