ECE 315 Computer Interfacing

LAB 0: Introduction to Vivado, Xilinx SDK & FreeRTOS

Winter 2024

DATES

Date	Section	Time
23-Jan-2024	D21	2:00-4:50 PM
24-Jan-2024	D31	
25-Feb-2024	D41	
26-Feb-2024	D51	

LEARNING OBJECTIVES

- To gain familiarity with the Digilent Zybo Z7 Zynq-7000-based board.
- To gain experience with the AMD/Xilinx Vivado and Xilinx SDK (Software Development Kit) tools.
- To gain some initial experience with the FreeRTOS real-time kernel.
- Note that this lab exercise does not require a demonstration to a TA nor does it require a lab report.

INTRODUCTION

Xilinx Vivado, like other contemporary Integrated Development Environments (IDEs) for FPGAs and FPGA-based SoCs, is a relatively complex software package. It will take you some time to get familiar with how to use the various commands in the many windows in Vivado and the SDK. This laboratory exercise will give you experience with the main commands that you will need to synthesize VHDL designs and download them to the Zybo Z7 board. Synthesizing a hardware-software design for a target FPGA board requires two main steps: (1) synthesizing the hardware design from a specification, to execute the software on the hardware design the hardware configuration and software must be downloaded to the Zybo Z7. (2) compiling the software for that hardware design.

HARDWARE PLATFORM AND SOFTWARE ENVIRONMENT

The hardware platform that we'll be using for this and the subsequent labs is a Digilent Zybo Z7 development board, which is based on a Xilinx Zynq-7010 System-on-Chip (SoC) silicon chip. The Zynq-7010 contains two 667-MHz ARM Cortex A9 32-bit CPUs, called CPU0 and CPU1, which can simultaneously execute two independent software systems. In the ECE 315 lab, CPU0 will be running the FreeRTOS multi-tasking software system; the second processor, CPU1, will be left disabled.

The Zynq-7010 SoC also contains a variety of other hardware subsystems including:

a. A high-performance Double Date Rate (DDR) memory interface, which the CPUs use to access software and data that is stored in a separate 1-Gbyte DDR3 DRAM memory chip.
b. A Field-Programmable Gate Array (FPGA) fabric, which allows synthesized custom digital logic systems to be implemented and interfaced to the two CPUs.
c. A variety of communication interfaces including two serial ports, two USB 2.0 ports, two Gigabit Ethernet ports, two SPI ports, two CAN ports, and two I2C ports.
d. 54 general-purpose I/O (GPIO) pins. The GPIO pins are software configurable as input or outputs.
e. Two 16-bit 1-Msample/second analog-to-digital converters (ADCs) with 17 software-selectable analog input pins. Note that all of these interface signals are available at the connectors and headers on the Zybo Z7 board.

The Zybo Z7 board houses the Zynq-7010 SoC and provides many connectors, headers, and a variety of simple input/output devices. In particular, the Zybo Z7 has:

a. A microUSB port that allows hardware configuration files and software (such as an operating system and user code) to be downloaded from the PC host. This port also communicates text messages, produced by print statements in software running on the CPUs, to be displayed on the host PC in the console window in Vivado SDK.
b. One RJ45 Ethernet connector, to provide a wired network connection.
c. One microSD flash connector for accessing stored software and data held in an external non-volatile memory.
d. Five Pmod (Peripheral MODule) expansion ports, which can be used to enhance the Zybo Z7 by connecting Pmod devices. Two of the Pmod ports connect to a 2-digit 7-segment LED display (SSD), and a third Pmod port is connected to a 16-button keypad. This leaves two free Pmod ports on the Zybo Z7.
e. Eight pushbuttons, including a reset button, a clear FPGA configuration button, and four user buttons.
f. Four user slide switches.
g. Four user-controlled green Light-Emitting Diodes (LEDs) and one red-green-blue (RGB) LED.

ECE 315 Computer Interfacing

LAB 0: Introduction to Vivado, Xilinx SDK & FreeRTOS

The Vivado SDK tool is an Integrated Development Environment (IDE) for software development that is based on the widely used open-source Eclipse framework. In order to produce an executable binary image file, SDK needs to know details of the target board (such as the CPU types, the memory maps the interface hardware, etc.) upon which the software will be running. The required details of the hardware, together with the software drivers that control the hardware, are contained in the board support package (BSP) for the Zybo Z7, which was provided by Digilent to Xilinx.

The demonstration project and source files are downloadable from the Lab 0 section of the lab eClass site. This contains the FreeRTOS kernel functions as well as the application code in C that you will be studying and then modifying.

Handling Precautions for the Zybo z7 Board:

The Zybo Z7 board contains sensitive electronic components that can be easily damaged by electrostatic discharge (ESD) when touched. The situation is worse in a cold climate, such as Edmonton's, where the indoor air is heated and thus the relative humidity of the air becomes very low. This fact, coupled with the fact that many people wear rubber-based, electrically insulating shoes and/or their floors are covered with static-creating carpets, mean that it is easy for your body to build up static charges of many hundreds and even thousands of volts. When your body gets charged up with significant amounts of static charge, and you then go and touch a connector on a circuit board (say a copper trace on the board, or a pin on a connector or header), it is easy to cause that charge to discharge through the conductor and punch its way through the fragile transistor gates in the input circuits. It is standard practice to provide protection diodes at the pins of integrated circuits so that the diodes will provide a safe path for discharging any deposited static electricity. However, serious damage can still occur when you touch conductors since the ESD protection circuits can't provide perfect protection. **As a standard precaution, be very careful when you handle circuit boards.** If possible, wear a grounded wrist strap when you are handling a circuit board. Avoid wearing rubber soled shoes. Avoid touching exposed conductors and the pins in connectors and headers. Get in the habit of touching a nearby grounded conductor before you start handling a circuit board or any semiconductor component.

ECE 315 Computer Interfacing

LAB 0: Introduction to Vivado, Xilinx SDK & FreeRTOS

PART 1

In this part we will go over the general steps of creating a new application project that uses one of the two CPUs on the FPGA board to interact with a specific hardware design for an embedded system. The main steps to do this are:

1. Create a hardware design (you will be provided with the necessary hardware platform for each lab).
2. Generate a bitstream for the hardware design and export it to SDK.
3. Create a new application using Xilinx SDK with the exported hardware platform.
4. Set up the runtime configuration for the project.
5. Run the project. This step runs a script that downloads the software image to the Zybo Z7 and then starts the execution of the software system.

Now we'll go over each of these steps in more detail and explain some of the important aspects of the application development.

1. Start by downloading the resources (a zip folder) from the Tutorial Lab section of the Laboratory eClass site and unzip the folder at a convenient location (say a lab_0 subdirectory). From the unzipped folder, open the ECE_315_lab_0 project file using Vivado 2019.1.

Note: Make sure that the path to your Vivado project files does not contain any whitespace character on it since this will cause critical errors when loading your projects.

2. Click on the **Open Block Design** command in the **IP Integrator** section of the **Flow Navigator** window at the left side of the Vivado interface. This action will cause a block diagram of the pre-designed Zynq-7010 hardware configuration for the tutorial lab to be displayed.

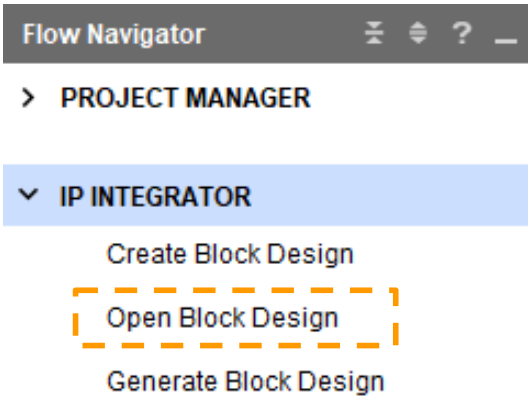

Figure 1: Open Block Design

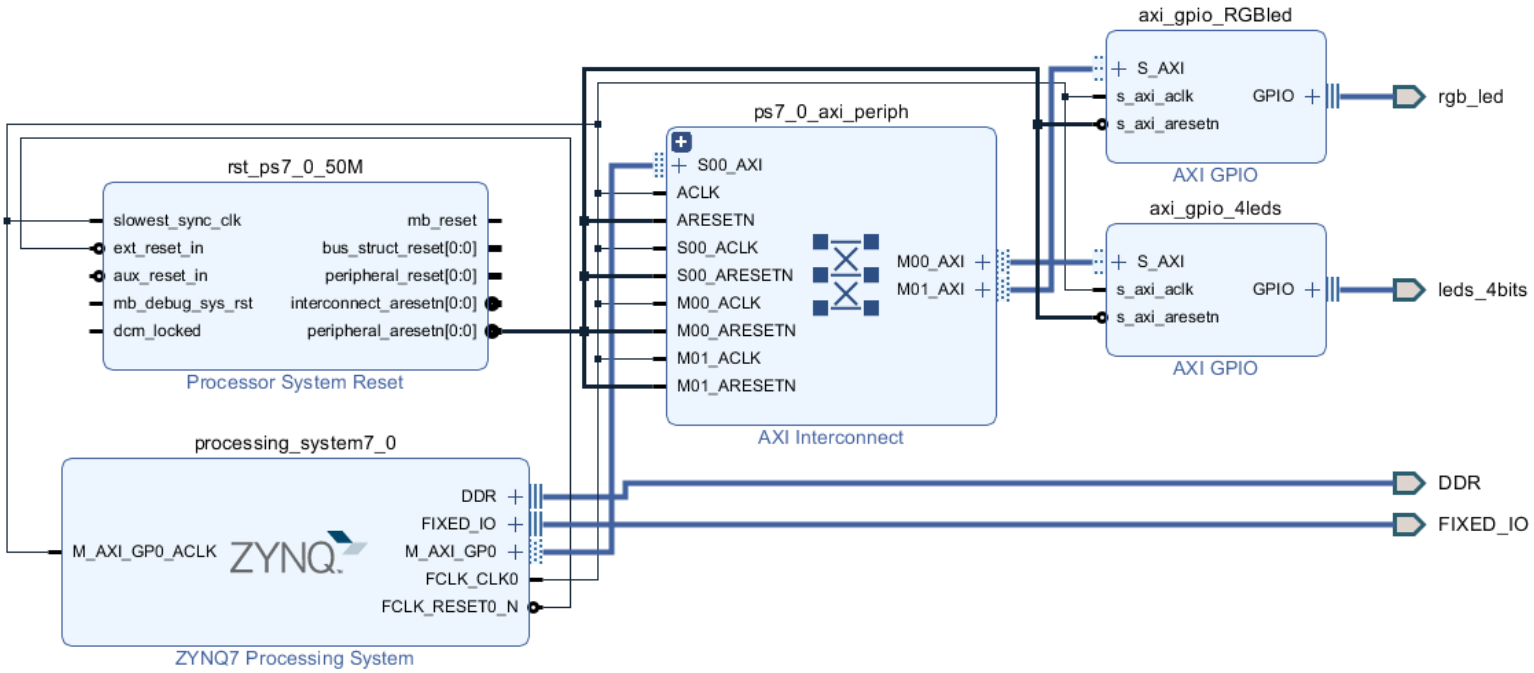

Figure 2: Block Design Example

3. Next, click on the **Generate Bitstream** command in the **PROGRAM AND DEBUG** section of the **Flow Navigator.** Select **Yes** in the pop-up window for Synthesis. Next click **OK** in the Launch Runs pop-up window.
   On the top right-hand side of the window, you can watch the progress of the synthesis process. Synthesis causes the given high-level specification the Zynq-7000 SoC hardware configuration to be converted into a bitstream configuration file. This bitstream file, when downloaded into the SoC, will cause the desired hardware configuration to be set up in the SoC to create the embedded system upon which you will run the FreeRTOS kernel and your application software.

4. Once the bitstream is generated, click on **File → Export → Export Hardware** (1 on the figure below), as shown below. The pop-up Export Hardware window will open. Make sure to check the **Include bitstream** option and leave the remaining settings with their default values (2 on the figure below). Click **OK**. The hardware design files that are required for software development in the Xilinx SDK tool are now generated and the hardware design is now exported into Xilinx SDK.

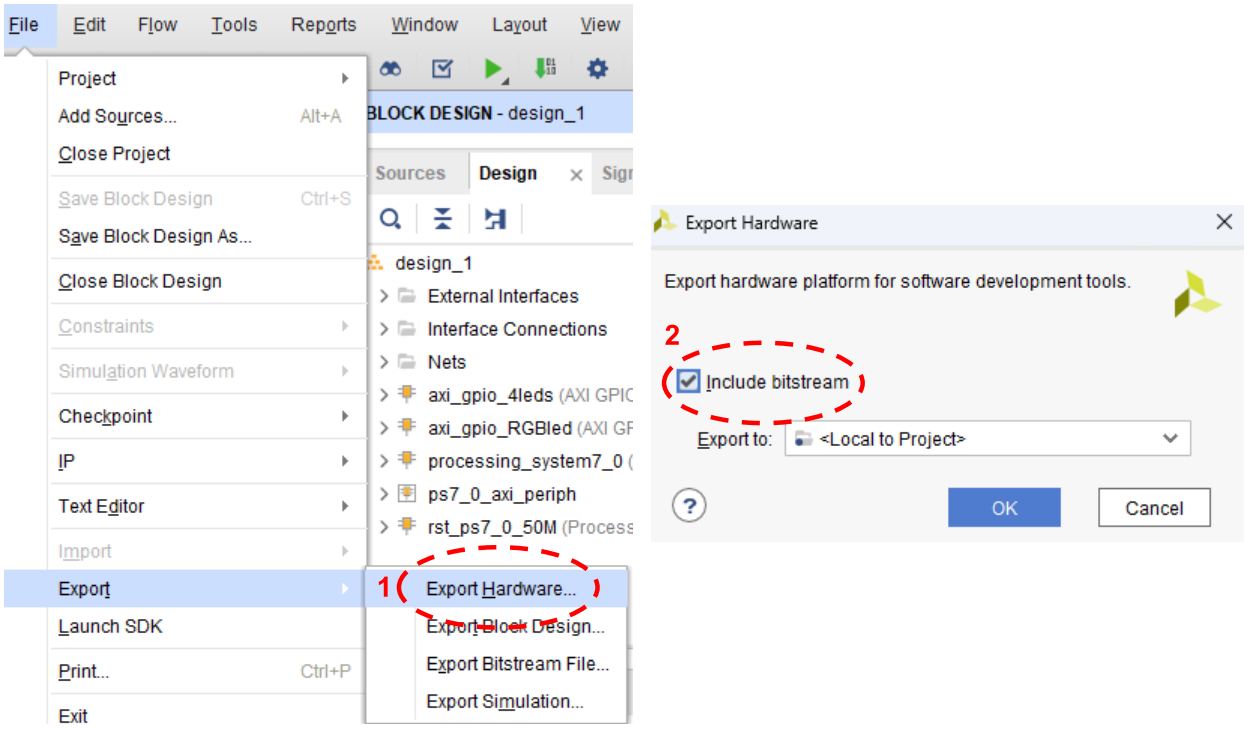

Figure 3: Export Hardware and bitstream

5. Start the Xilinx SDK by selecting **File → Launch SDK**. The Xilinx SDK desktop window will now open and **you must wait for the initialization process to finish**. You will soon see the populated Xilinx SDK windows appear. The exported hardware design will appear in the Project Explorer pane at the left side under the name of "**design_1_wrapper_hw_platform_0**". You will recognize a few **.c** files and **.h** header files in this folder. The Tcl script file **ps7_init.tcl** is used to initialize and configure the Zynq Processing System, which is called a ps7 (one of the two processing system CPUs in the Zynq7000 SoC). Tcl (pronounced "tickle") is a high-level scripting language that was used to create Xilinx Vivado. After gaining more experience with the tools, you can investigate how to create your own Tcl scripts, with file type .tcl, that contain useful sequences of Vivado and SDK commands.

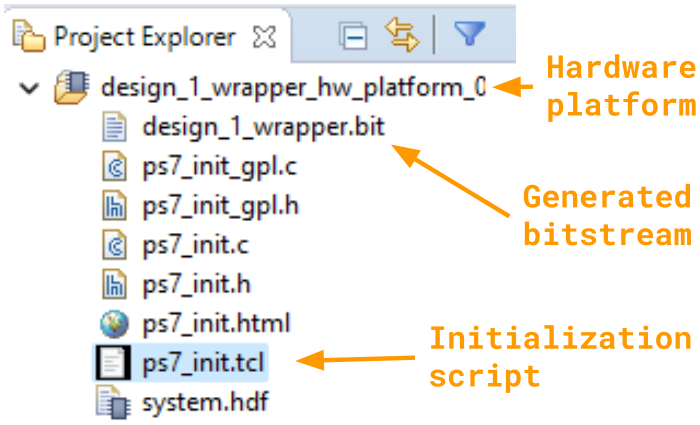

Figure 4: Hardware Platform

6. Click on the **File → New → Application Project** command. Enter the project name that you want to use and **remember to not include whitespace characters in the name**. Under the OS Platform drop-down menu, make sure to select **freertos_10_xilinx** to add the FreeRTOS kernel to the software system. Leave the other selected choices at their default values and click **Next**. Leave the default template as **FreeRTOS Hello World** and click **Finish**.

   It is common to include a **"Hello World"** tutorial example in software design environments such as the Xilinx SDK. Such an example demonstrates to new users how a minimal software program can be compiled, downloaded to the target hardware, and executed to produce a simple output message. More complicated software programs can then be developed by modifying and growing the Hello World example. Trying to create a brand-new working **"Hello World"** example from the documentation would typically take a long time for a new user.

7. After the workspace build is done, two new folders will appear in the Project Explorer Window. Note the appearance of **tutorial_lab_0** and **tutorial_lab_0 _bsp**. Expand the project folder and the **src** subfolder and open the source file **freertos_hello_world.c**. Replace this code with the **lab_0_part_1.c** file that was provided to you on eClass and save the changes. When executed, this tutorial lab code will print the message "**Welcome to ECE315 Lab.**" in the console window (at the bottom of the desktop) four times at an interval of 1 second.

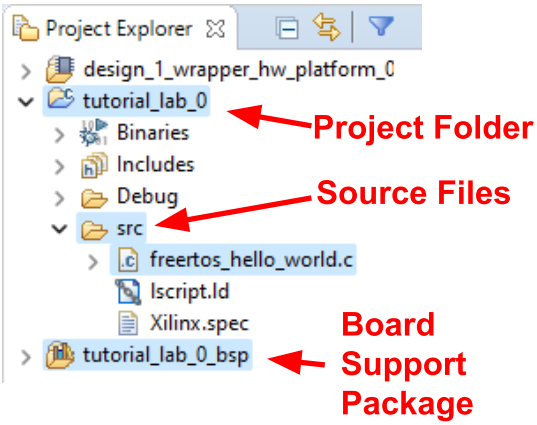

Figure 5: Project folder structure

8. Turn ON the power switch on the board and you should see a red light on **LD13**. Now, to establish serial communication with the board, click on the **SDK Terminal** tab in the bottom window of the Xilinx SDK. Click on the icon on the top border of the SDK Terminal window and a pop-up **Connect to serial port** window should open. In the Port pulldown menu of the pop-up window, select the COM port on your PC to which the micro-USB cable from your Zybo Z7 board is connected. Leave the other settings at their default values, including the baud rate of **115200** and click **OK**.

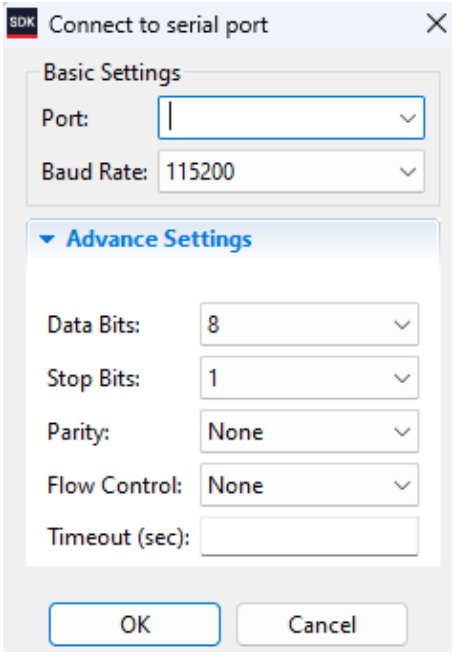

Figure 6: Connect to serial port

Note: to check the available COM ports on a computer running windows open the start menu and type "**Device Manager**" then select it from the list that appears, once it's open expand the **Ports (COM & LPT)** dropdown menu. The FPGA board port will appear as "**USB Serial Port**"

9. To run the application, right click on the project folder inside the Project Explorer pane. Select **Run As → Run Configurations**. The configuration window will open.

   Double-click on the **Xilinx C/C++ application (System Debugger)** command. In the same window, on the right side check the boxes for **Reset entire system** and **Program FPGA**. Make sure that the remaining settings are left with their default values. Under the "**Application"** tab, verify that the **ps7_cortexa9_0** checkbox is selected, the project name is the same that you chose in Step 6 and the application field has the same name with an **.elf** file.

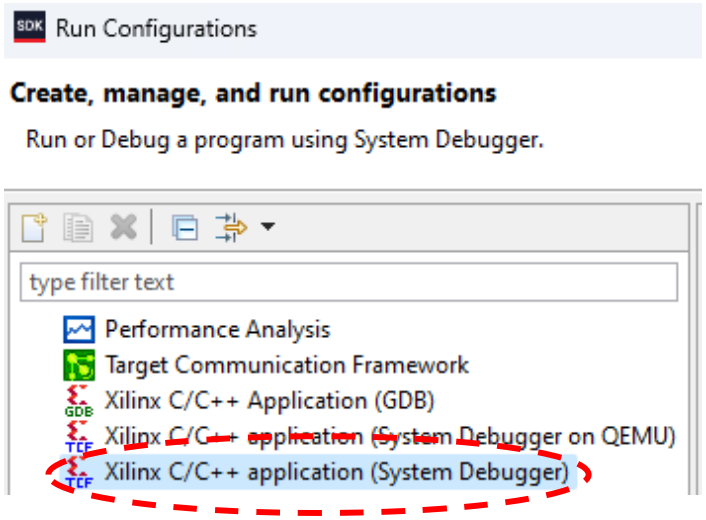

Figure 7: Run Configurations

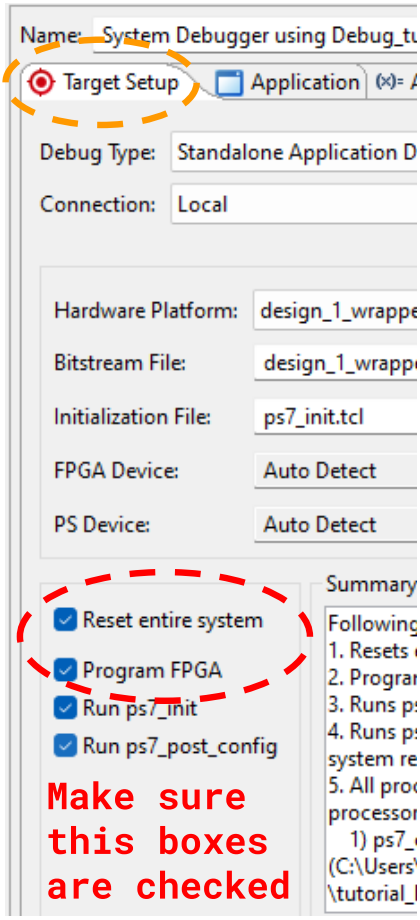

Figure 8:Target Setup

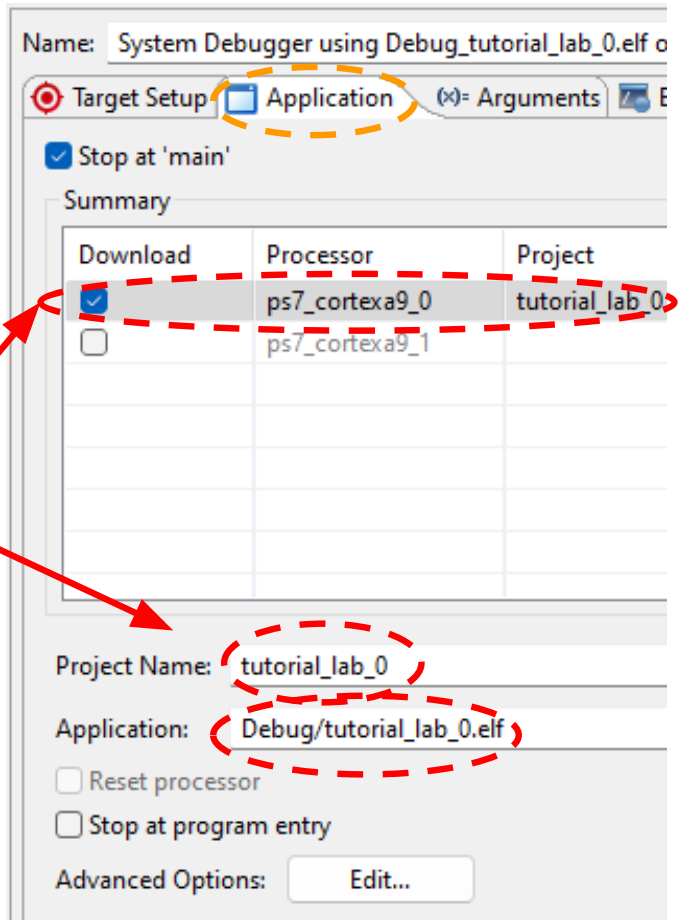

Figure 9: Application Setup

The file type **.elf** stands for Executable and Linkable Format, which is a widely used file format for encoding binary files for embedded systems. Click on **Apply → Run** to produce the following sequence of actions:

(1) A board reset
(2) Downloading the hardware configuration to the Zynq7010 SoC
(3) Downloading the compiled software to the DDR3 memory on the Zybo Z7 board
(4) Executing the software on the Cortex A9 processor that is designated "**design_1_wrapper_hw_platform_0**".

You should now see that once the application begins to run, the DONE LED (LD12) turns green on the board, and serial messages produced by the xil_printf() statement in the software will be displayed in the SDK terminal window.

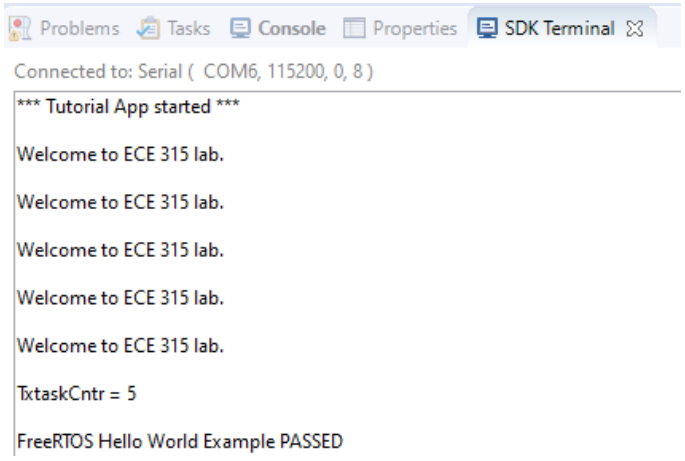

Figure 10:Serial Console Output

10. Examine and modify the code to print a new message. For instance, change it to include your name. Re-run the application by clicking on **Apply → Run** and confirm that the updated serial message is now being printed.
11. Modify the timer to expire after 10 seconds and confirm the change in behavior.
12. Set the timer to expire after 3 seconds and notice the difference between the new behavior and the previous step.
13. What happens if you delete the timer entirely?

PART 2 - BLINKING LEDS

This part creates two FreeRTOS tasks, one responsible for turning the RGB LED to BLUE and the other responsible for turning it to YELLOW using the RGB LED (LD6 on board). The BLUE task runs and goes into a delay (i.e. it is deliberately blocked and is waiting for the delay period to finish) for 1.25 seconds. The YELLOW task runs and then goes into a delay period for 0.65 seconds. If there is a case where both the tasks come out of the blocked state at the same instant, the BLUE task will run first before the YELLOW task as the BLUE task has the higher priority. In this exercise, you will be responsible for changing the RGB LED colors on the board. By default, one task is assigned a blue color and the other is assigned a yellow color. Download the file named **gpio_tasks_leds.c** from the eclass. The remaining color definitions are provided at the top of the code file. Modify the C code under the commented section of the file to change the color of your choice.

1. Start by creating a new application project on the Xilinx SDK **File ➞ New ➞ Application Project**. Follow the same procedure except that under the project name, specify "**gpio_leds**" but this time replace the existing code with the code from **gpio_tasks_leds.c**. Save the changed source file.
2. Take a moment to predict the outcome based on your understanding of task priorities in a real-time operating system. Consider the following questions to guide your prediction:
    a. Which task do you expect to run first and why?
    b. Which color will be ON most of the time given the task priorities and time delays?
3. Turn ON the Zybo Z7 board and establish a serial connection as described in the above steps. Run this project in the same manner as before. You should now see the RGB LED display and the blinking of the colors blue and yellow. On the serial terminal you will see the messages being printed that identify the task function that is currently being executed.
    a. Use the FreeRTOS function **xTaskGetTickCount()** to get the number of ticks that have passed since **vTaskStartScheduler** was called and modify the **xil_printf** statement, included at the beginning of every task, to display this number.
    b. Create a new task to add a third color to the RGB LED blinking pattern and set a delay of 2.2 seconds. Modify the priorities of the tasks so that the blue task keeps the highest priority and the yellow task has the lowest priority, leaving the new task with a priority in the middle.
    c. Notice the changes in the system's behavior after adding the third task. Experiment with different priority values for each task in order to understand better how the system works and how the structure of the code can give you hints of the system's behavior.

ECE 315 Computer Interfacing

LAB 0: Introduction to Vivado, Xilinx SDK & FreeRTOS

Winter 2024

RESOURCES

- ➔ Digilent Zybo Z7 Reference Manual
- ➔ The FreeRTOS Reference Manual

REPORT REQUIREMENTS

No report is required for this tutorial lab exercise.

MARKING SCHEME

This lab is not worth any marks. However, it is important that you complete this lab exercise if you wish to succeed on the subsequent lab exercises, which are all graded and worth marks.