Lab 1.0: PWM Control Software

In this introductory lab, you are going to write a piece of software to control a Pulse Width Modulation (PWM) signal. At this stage, we provide you with the hardware design. Note that this will no longer be the case in future labs. The provided design includes two PWM control interfaces that are programmable. Their outputs are routed to two GPIO pins of the board.

Wiring

Before starting to write the software, you might want to be able to observe your signals. We are going to use a logic analyzer for this matter. The first step is therefore to connect the logic analyzer to the board. The picture below outlines the wiring.

Welcome in the world of short-circuits

Whenever you are physically wiring components in embedded systems, be very cautious. We recommend that you do not connect your FPGA board and your logic analyzer to the power while doing so. Carefully review your wiring before doing so. Always start by connecting the GNDs together.

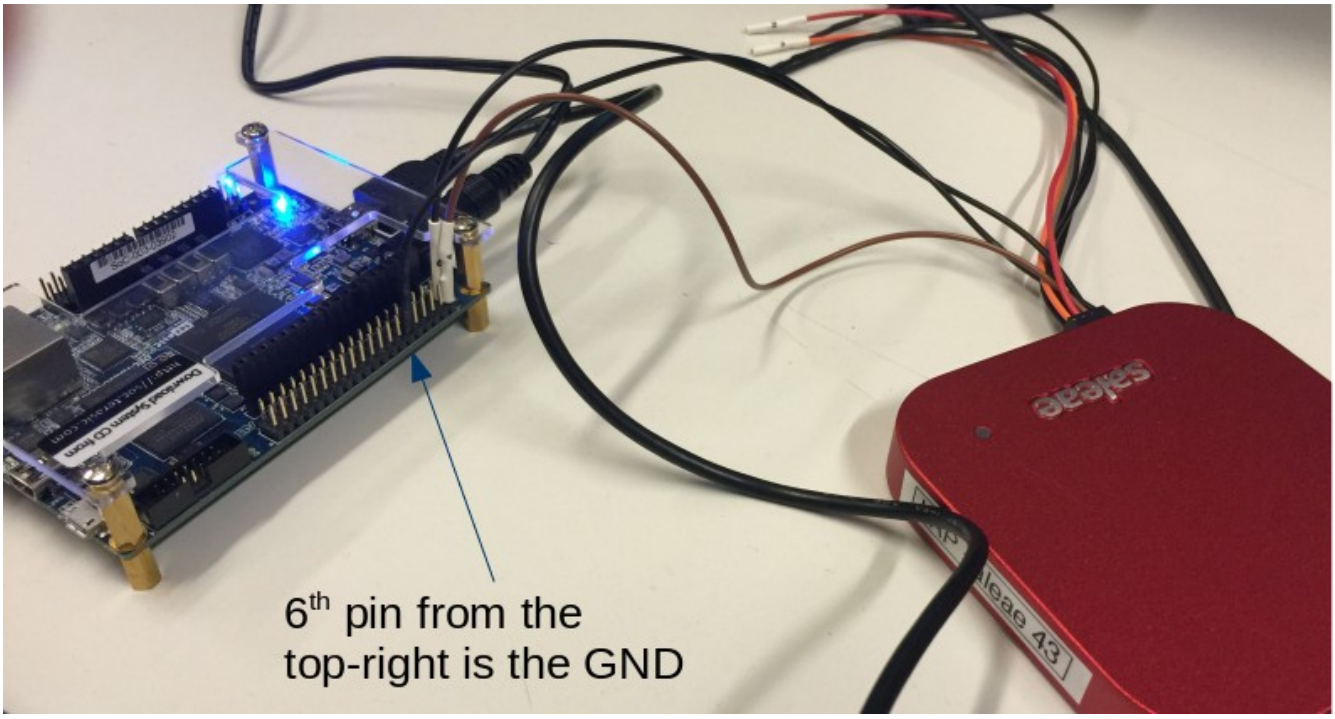


Figure 1: Wiring the FPGA with the logic analyzer.

Launching the SBT

We are using a Nios II CPU in our design, so we need to use the Eclipse-based "Nios II Software Build Tools" (SBT) to program our CPU.

To launch Nios II SBT, we need to first start a Nios II Command Shell. This shell defines some environment variables that are needed for SBT to work correctly.

You can launch the required command shell from the Ubuntu launcher. Once the command shell is open, enter:

eclipse-nios2 &

to launch Nios II SBT.

Programming the FPGA

It is time to download the hardware design on the FPGA. Here are the steps:

1. Plug your FPGA into your computer.
2. Open the Quartus Programmer from **Nios II > Quartus Prime Programmer...** in the menu bar.
3. Click on the "Auto Detect" button on the left-hand side of the Quartus Programmer.
4. Choose 5CSEMA4.
5. Once you get back in the Quartus Programmer's main window you will see 2 devices listed in the JTAG scan chain. One of them corresponds to the HPS (ARM cpu), and the other to the FPGA.
6. Right-click on the FPGA entry, and go to **Edit > Change File**
7. Select the compiled .sof file in the "output_files" directory in your quartus project's directory.
8. Enable the "Program/Configure" checkbox for the FPGA entry, then click on the "Start" button on the left-side menu.

Creating the software project

Now that the FPGA is programmed, we can create a software project for our design.

1. Go to **File > New > Nios II Application and BSP from Template**.
2. Select *<project_dir>/hw/quartus/soc_system.sopcinfo* as the SOPC Information File name.
3. Name your software project "lab_pwm"
4. We invite you to uncheck the "Use default location" checkbox and to choose *<project_dir>/sw/nios/application* instead. We encourage this practice to properly separate sotware from hardware design files.
5. Choose "Blank Project" as the Project Template.
6. Click **Finish**.
7. Right-click on *app.c* and *pwm.c* in the **Project Explorer** and select **Add to Nios II Build**.
8. You can now write/compile/run your software.

PWM Control Interface

The provided system includes two PWM control interfaces. Both are 32-bit Avalon Memory-Mapped Slave interfaces. They are mapped in memory at addresses *PWM_0_BASE* and *PWM_1_BASE*, respectively. These macros can be found in the auto-generated *system.h* header file that you need to include (already done in the provided files).

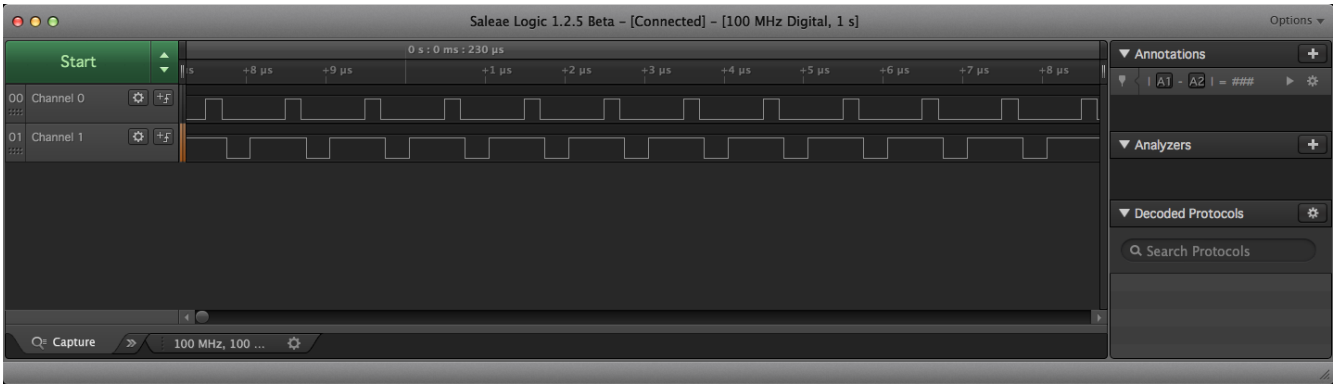
The register mapping in memory of those control interface looks as follows:

Offset (as viewed by the NIOS)	**Name**	**Access**	**Description**
0	PERIOD	R/W	The period of the PWM. The period unit is 50-Mhz clock period. For instance, writing 2 in this register will generate a 25-Mhz PWM signal.
4	DUTY	R/W	A value between 0 and PERIOD indicating the duty cycle of the PWM.
8	CTRL	WO	Writing 0 (resp. 1) to the register stops (resp. starts) the PWM.

To write these registers you will want to use the IOWR_32DIRECT(BASE, OFFSET, DATA) macro available in the *io.h* header file.

Using the logic analyzer

You can use the Ubuntu launcher to start the Saleae Logic program. Press the **Start** button. After launching your software on the FPGA, you should obtain something looking as follows:



Exercise

Fill in the gaps in *pwm.c*. We provide you with a *main* function to test your code.