Contents

[Educational Objective 2](#_Toc475604566)

[Technical Objective 2](#_Toc475604567)

[Part 1 – Create directory structure and repositories. 2](#_Toc475604568)

[Part 2 – Create Quartus project using the given VHDL files. 3](#_Toc475604569)

[Part 3 – Create the dual port block RAM in Quartus 3](#_Toc475604570)

[Part 4 – Instantiate dual port block RAM and add code 3](#_Toc475604571)

[Part 5 – Create Nios (QSYS) project with bus bridge 4](#_Toc475604572)

[Part 6 - Create Software Project in Eclipse using given C-code 5](#_Toc475604573)

[Part 7 – Use SignalTap Logic Analyzer to Look at a Bus Bridge Transaction 5](#_Toc475604574)

# Educational Objective

After completing this lab, you will

1. Learn how to use a bus bridge to connect the Avalon bus to your top-level VHDL code. This allows Nios to read/write registers or memory that is external to the Nios system, in the FPGA logic fabric.
2. Learn how to use block RAM to create memory components in Quartus.
3. Learn how to use a dual-port ram.
4. Learn how to use the SignalTap tool with Quartus to debug a design.
5. Learn how to use Tortoise SVN to control the source files for your project.

# Technical Objective

The technical objective of this lab is to create a project which plays a waveform through the audio codec, which can then be heard on headphones. Pressing Key1 on the DE1 board will incrementally increase the frequency at which the waveform is played back. Additionally, the Nios processor will be able to write any arbitrary waveform, allowing experimentation with what various wave-shapes sound like.

# Part 1 – Create directory structure and repositories.

First get all the given source files from MyCourses and put them in a temporary directory. You will need the VHDL, C, and the audio\_wvfrm\_init.mif file.

Then Create the following directory structure…

lab4 -----> lab4\_quartus

-----> lab4\_src

1. Then right-click on lab4\_src and select **create repository here** option from the Tortoise menus.
2. Tell Tortoise to create the default file structure, and then click OK.
3. Then again right-clock on lab4\_src and select **open repo browser** from the Tortoise menus.
4. In the repo-browser, navigate to the trunk directory.
5. In the trunk directory, right-click on **create directory**, and create directory lab4\_hdl\_src in the trunk directory
6. Enter the newly created lab4\_hdl\_src directory, right-click, and select **add file**. Use this to add each of the VHDL files from MyCourses, (which you stored in a temporary directory).
7. Now go back to the trunk directory in repo-browser, right-click, and select create directory. Create the directory lab4\_software\_src, also in the trunk directory. As before, navigate to the newly created lab4\_software\_src, and right-click to add the provide c source code.
8. At this point if you go to your lab4\_src repo directory, you will not yet see lab4\_hdl\_src or lab4\_software\_src. You need to check them out of the repository. Still in the repo-browser, right-click on lab4\_software\_src, and select checkout. Do the same for lab4\_hdl\_src. You now should see these directories in your repository.

# Part 2 – Create Quartus project using the given VHDL files.

Create a new Quartus project just as you have in the past. Add all the VHDL source files given in MyCourses in your project. These source files should be located in your repository, in the lab4\_hdl\_src directory. Don’t forget to import the lights\_import.qsf file with the pin assignments for the DE1 board. Also you will need to have the audio\_wvfrm\_init.mif file in your Quartus project directory. The audio\_wvfrm\_init.mif contains a default waveform that will be loaded into the RAM block you will create.

# Part 3 – Create the dual port block RAM in Quartus

In this part you will create a RAM IP component. You will instantiate this component into your VHDL in Part 4 below.

The dual port RAM is a RAM with two independent sets of data/address/control signals. This enables you to access two different parts of RAM simultaneously. In some cases it can also be used when the two ports will have different clocks, but that is not the case for the this lab.

The Quartus tool will create this component, along with a template you use to instantiate the component. Follow the following steps to create this:

1. In your Quartus project look at the **IP Catalog** on the right side of the window.
2. Navigate to following IP component:

Installed IP->Library->Basic Functions->On Chip Memory->RAM 2 Port

1. Double click on this component to open the wizard and create one. Configure it as indicated in the screen captures on MyCourses under Dual Port RAM Config. Make sure you have the file audio\_wvfrm\_init.mif. in your Quartus project directory. This contains a default waveform that will be loaded into the RAM.

# Part 4 – Instantiate dual port block RAM and add code

You now need to instantiate the dual port RAM, along with any required additional code, in the peripheral\_on\_external\_bus.vhd file. This part of the code is not given, you will write it. Port A of the dual port RAM is to be connected to the Avalon bus, which is exported from the Nios system by the bridge. Port A will therefore enable Nios to read and write the waveform RAM. Port B will be connected to the **codec\_dac\_interface** component which serializes audio data and sends it to the CODEC.

Here are some other hints on how to write this code:

* While the B port can be written, for our purposes it is read only.
* A write enable can be generated from the Avalon bus as follows:

wren <= (not i\_rw\_n) and i\_bus\_enable;

* To generate the acknowledge signal, back to the Avalon bus, you should output a single clock wide pulse, generated on the rising edge of **bus\_enable**.
* The address bus from the Avalon bus bridge used byte addressing, therefore you will need to drop the lower two bits before passing this signal into the RAM IP component you created using the Quartus IP library.
* You may need to pad the addresses with leading zeroes to make the vector sized match.
* Figure 1 below shows typical read/write transactions on the Avalon Bus Bridge.



Figure 1: Avalon Bus Bridge Timing Diagram

# Part 5 – Create Nios (QSYS) project with bus bridge

The bus bridge is located in University Program->Bridges->Avalon to External Bus Bridge. Add the bridge and configure it to be 2kBytes, with data width of 32. Lock the base address of the bus bridge at 0x00000, as you did in the last lab, before assigning addresses.

You also need to add two single bit pio ports, as shown in Figure 2 below. The clock port is an **output**, and the data port is **bi-dir**. These will be used for the IIC serial port, used to configure the CODEC.

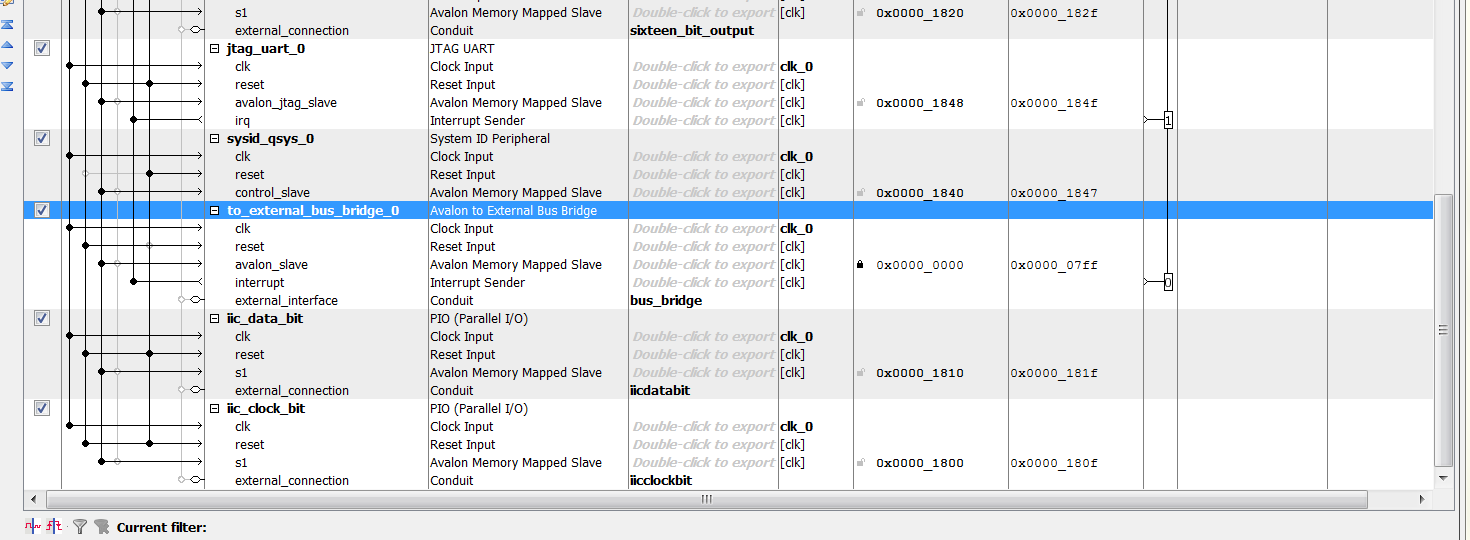


Figure 2: Nios Processor system in QSYS, with bus bridge and the two required single bit PIOs for the IIC serial bus.

# Part 6 - Create Software Project in Eclipse using given C-code

You will need to run this c-code to initialize and configure the codec through the serial IIC interface. We will talk more about IIC later in the course. You may need to make modifications to get this code to work, since the defines created in your project may have different names.

# Part 7 – Use SignalTap Logic Analyzer to Look at a Bus Bridge Transaction

We will discuss this in class, and I will give a demo of how to set it up.