Education Objective

The educational objective of this demo is to become familiar with the creation of custom components that can interface with the NIOS II through the Avalon Switch Fabric.

Technical Objective

The technical objective of this laboratory is to design a custom component for the NIOS II that stores eight 32-bit values. Conduit interfaces to the DE1-SoC board allow for an address to be entered through the switches and the value stored at that location to be displayed on the LEDs. The component generates an interrupt to the NIOS II if the address on the switches is greater than 7.

Background

The QSYS System Builder allows a designer to quickly create digital system by interconnecting selected QSYS components, such as processors, memory controllers, and serial ports, etc. The QSYS System Builder includes many pre-designed components that may be may be used in a system. It is also possible for users to create their own custom Qsys components.

A QSYS ready component is a hardware design that is available as a library component for use in the QSYS System Builder. Typically, the component contains two parts. The first part is internal hardware modules that implement the desired functionality of the component. The second part of a component is the Avalon Interfaces that allows the component to communicate with other components that exist in the system, such as the Nios II processor.

In this demonstration we will create a custom component written in VHDL that will be a slave to the Nios II processor. In addition, the component will also be capable of interfacing to other hardware peripherals or logic in the FPGA that is located outside of the QSYS system.

Once a custom component is in VHDL, the component editor in the QSYS System Builder is used to create the necessary files and make the component part of the QSYS component library. Once a component is part of the library, it can be added to the system and linked to a processor in the same manner as the other components provided by Altera.

When the QSYS system is generated, the hardware description for the custom component becomes part of the nios_system.vhd. If the component has signals that interface external to the system (**Avalon Conduit Interface**) those signals will be ports on the system entity.

Custom Component Demonstration

For this demonstration, we will design a VHDL module that will create a QSYS ready custom component. A block diagram of the custom component is provided in below.

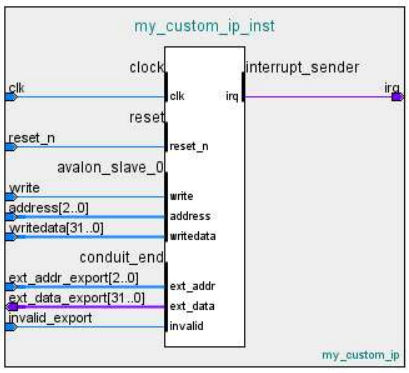

Figure 1: High-level Block Diagram of Custom Component

The data array in the custom component allows the processor to store 8 different 32-bit values within it. The external, or conduit, interface allows for an address from an external source (the switches on the DE1 board in our case). The data stored in the data array at the address specified will be displayed on the ext_data port. In our case this will be the LEDs on the DE1-SoC board. If the invalid signal (asserted if the address selected is greater than 7) is active, the custom component will interrupt the processor.

Building the Hardware

1. Download the Custom_Component_Suport_Files zip file located on MyCourses and unzip it.
2. Open the file custom_ip.vhd and read through it so you understand it.
3. Create a new Quartus II project for your system and name it custom_component_demo.
4. Use QSYS System Builder to create a system named nios_system, which includes the following components:

    - Nios II/e processor
    - 32K On-chip memory
    - JTAG Uart
    - Sysid

5. At this point your system should look like this:

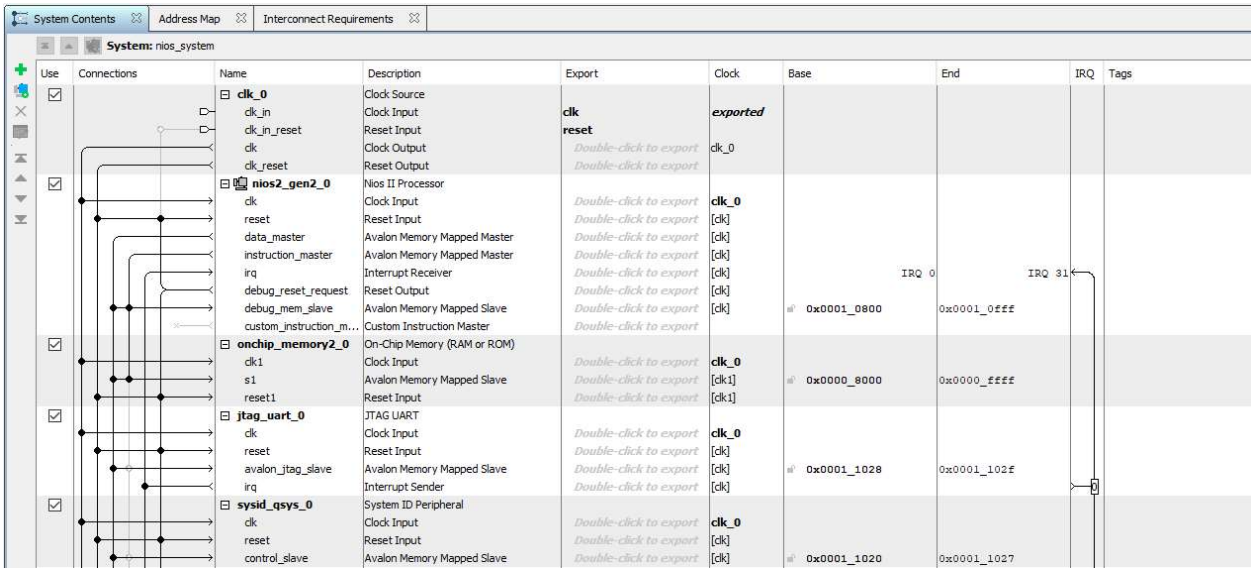

Figure 2: nios_system in Platform Designer

6. Now we will create our custom component. Choose **File > New Component** or click on clicking on **New component** in the Component Library window of QSYS.

7. The first step in creating a component is to specify where in the Component Library our new component will appear. In the **Component Type** tab, change the **Name** to my_custom_ip, the **Display name** to my_custom_ip, and provide a name for the **Group** setting, such as ESDI_IP. The group allows you to place custom components into specific libraries. Add your name by **Create By**. Your Component tab should similar to Figure 3. Click **Next** when done.

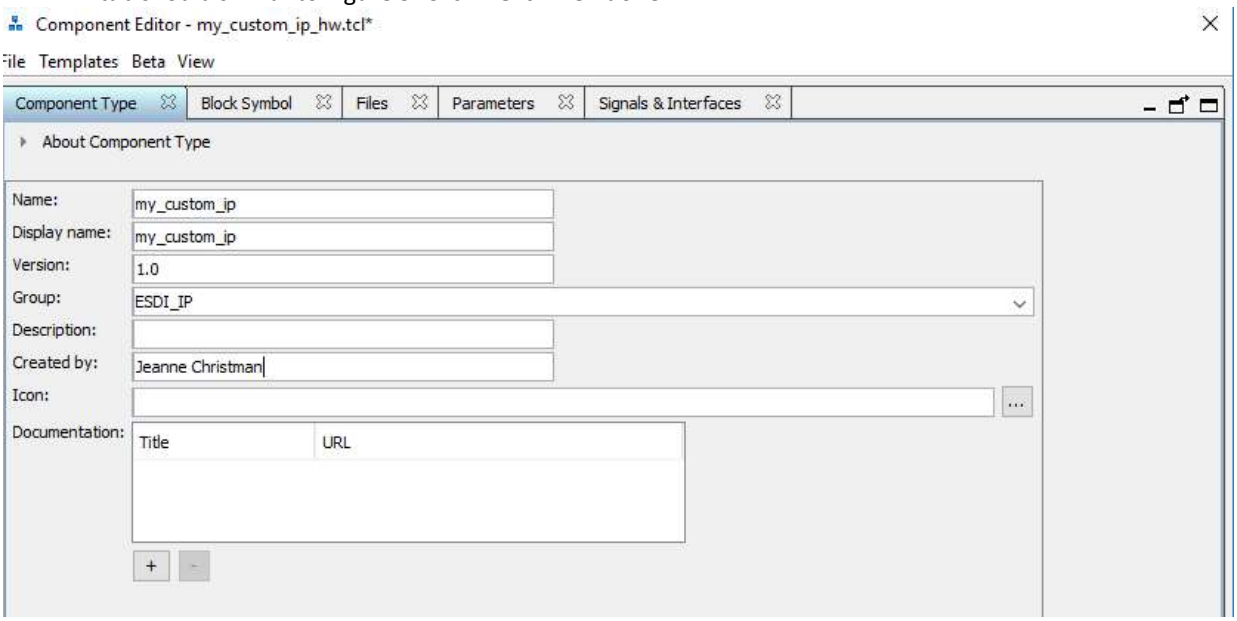

Figure 3: Component Type Tab of Component Editor

8. Select the **Files** tab and add the file custom_ip.vhd that was provided by clicking on Add File…

9. Once the file(s) have been added to the Component Editor, click on the **Analyze Synthesis Files**. This will cause the Component Editor to analyze the file for any problems. You will get some errors.

10. Click on the **Signals & Interfaces** tab. In the left panel on the bottom, click on *<<add interface>>* and choose conduit. Choose **reset** in the **associated reset box**. Select **ext_addr_export** in the signal list and drag it under **conduit_end.** Do the same for **ext_data_export** and **invalid_export.**

11. Click on **ext_addr_export** and change the **Signal Type** to **ext_addr.** Do the same for **ext_data_export** and **invalid_export**, changing their Signal Types to **ext_data** and **invalid** respectively.

12. Choose *<<add interface>>* and choose **interrupt sender**. Choose reset in the **associated reset** box. Select **irq** in the signal list and drag it under **interrupt_sender**. Change **Signal Type** to **irq.** Click on interrupt_sender. In the **Parameters** area, change **Associated addressable interface** to **Avalon_slave**

13. Click on **Avalon_slave_0** and set the **Associated Reset** to **reset**.

14. At this point your **Signal & Interfaces** tab should look like this:

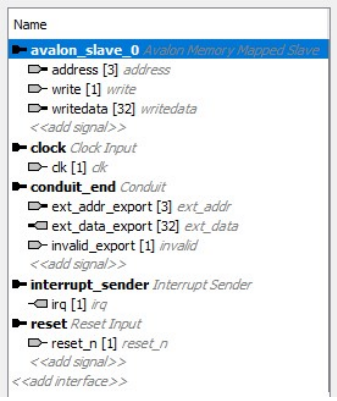

Figure 4: Signal Tab of Component Editor

15. Click on Finish… and save.

16. In the IP Catalog in Qsys, click on ESDI_IP and double click on my_custom_ip. Click Finish in the pop-up box and then complete the signal connections. Export the conduit and name the export **custom_ip**. Make the custom_ip component the highest priority interrupt.

17. Generate the VHDL.

18. The top-level VHDL file (custom_component_demo.vhd) is provided for you. Add this file to the Quartus II project and set it as the Top-Level entity. Remember to also add the IP variation file (.**qip**) to your Quartus II project and specify the pin assignments.

19. Compile your Quartus II Project and program the Cyclone V FPGA on the DE1-SoC board.

Building the Software

1. Open the Nios II Software Build Tools for Eclipse. Create a NIOS II App and BSP from template. Name the project **custom_component** and choose **Blank Project.**

2. The C program for the demonstration is provided for you in the downloaded ZIP file. Add the file custom_component_demo.c file to your custom_component App folder.

3. Generate the BSP, copy system.h from the bsp to the app folder and build the project. Choose Debug as > NIOS II Hardware.

4. Click on the resume icon and change switches 2-0 on the DE1-SoC board to view the data in the custom IP.

5. Now raise SW3 and notice that an error message appears in the console window of the EDS. This error occurs because when SW3 = 1, an address 8 or greater is being accessed, but the component only has 7 locations.

Demo: Submit a video in the dropbox for steps 4 and 5 from the Building the Software section. Be sure to include an explanation in your video of where the LED values are coming from.
Please submit ONLY the video and document, and not a zipped project.