Adding A Custom Core Tutorial

September 25, 2007

1 Objective

The goal of this tutorial is to explain how to create a custom hardware core. Do not confuse Hardware Core with Hard IP Core. The custom cores created are actually Soft IP Cores, but since they are running in hardware (in the FPGA fabric) they are referred to as Hardware Cores. A Hard IP Core refers to a core that is embedded onto the FPGA which cannot be reconfigured (PowerPC, BRAM etc...). The process discussed here is how to use Xilinx's *Create or Import Peripheral Wizard* to create a custom core, but does not cover how to modify the core to perform any specific functionality. The functionality is application specific and requires hardware description language knowledge that is beyond the scope of this simple tutorial.

2 Adding a Custom Core Tutorial

This tutorial assumes you have already built a base system. From XPS click on the **Hardware** menu option and select **Create or Import Peripheral...** This will launch the Wizard we will use to create a template core. This is similar to how we created a base system, except now we are specifying the hardware core's configuration. To begin:

1. Welcome Screen - [Page 1]

   Click Next

2. Peripheral Flow - [Page 2]

   We will be creating our own NEW peripheral (hardware core)

   Click Next

3. Repository or Project - [Page 3]

   We want to add our hardware core to our project directory rather than a Repository. For larger systems it may be advantagous to create a repository and add your cores to it, but for class we will stick *To an XPS project*

   Click Next

4. Name and Version - [Page 4]

   We name the core some specific name (use detailed names)

   Certain characters and keywords cannot be used and will turn the text red, before you click Next make sure the name is a valid name (not red). The version can be 1.00.a or whatever you want.

   Click Next

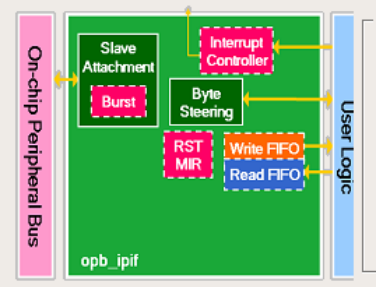

Figure 1: Xilin'x IP Interface

5. Bus Interface - [Page 5]

   Select which bus you want to connect your core

   Click Next

6. IPIF Services - [Page 6]

   Probably the most confusing page out of the entire wizard. The IPIF stands for *Intellectual Property Interface*. If you look at the **Figure 1** you will see the IPIF acts as a middle man between the bus and your user logic. The specific functionality of your core will be in the *User Logic*. Xilinx created an IPIF to interface your core to the bus without requiring you to handle the complex bus transactions. That being said here is the important take away from this page:

   - To make your core a Slave on the Bus select **User Logic S/W Register Support**
   - You can uncheck *S/W reset and MIR* and *User Logic Interrupt Support* as they are not necessary for slave designs in our simple cores.
   - To make your core a Master on the Bus select **User Logic Master Support**
   - It is possible for a core to be both a Master and a Slave
   - Any more complex functionality (checking other boxes) will be covered in future tutorials.

   Click Next

7. User S/W Register - [Page 7]

   Select the Number and Data Width of registers that can be read from or written to by other cores (these are slave registers)

   This is not BRAM, but actual registers that are addressable by other cores in the system.

   Leave *Enable posted write behavior* checked

   Click Next

8. IP Interconnect (IPIC) - [Page 8]

   Add any additional signals you want your core to have access to from the IPIF. Typically if anything I add Bus2IP_Addr, but it is not necessary for most basic designs to add any signals.

   Click Next

9. (Optional) Peripheral Simulation Support - [Page 9]

   To simulate the Bus with ModelSim you can click this box

   This requires the package to be installed (currently it is not)

   Click Next

10. (Optional) Peripheral Implementation Support - [Page 10]

    We like VHDL in our lab, not Verilog (don't check / uncheck anything)

    Click Next

11. Click Finish (Congratualtions)

At this point you have created a hardware core. Select **IP Catalog** and look for *Project Local pcores*. [You may need to click the Icon right under *Project* to change between *Flat View* and *Hiearchical View*] Expand *Project Local pcores* and double click on your core to add it to your project. Finally, you need to connect your Project to the bus.

3 Modifying VHDL

Whether or not you connect your hardware core to a particular bus, you can go in and modify the User Logic and specifiy functionality. The hardware core created is located in the *pcores* directory of your base system. For this example, let's pretend you created a core called my_core_v1_00_a the location of the core is:

```
lab/build/pcores/mycore_v1_00_a
```

We can now modify the User Logic and add functionality. The User Logic file is located at:

```
mycore_v1_00_a/hdl/vhdl/user_logic.vhd
```

Modifications at this point are application specific. I highly recommend making a backup copy of your user_logic.vhd file before making any modifications. That way if you really mess up you can revert back without having to re-generate a new core.

```
cp user_logic.vhd user_logic.vhd_bkup
```

Take some time to look at the user_logic.vhd file. To modify the core there is a **Simple VHDL Tutorial** that you can follow next.