Implementing the pci_a Master/Target in FLEX 10K Devices

May 1997, ver. 1 Application Note 86

Introduction

A peripheral component interconnect (PCI) function provides an efficient solution for integrating 32-bit PCI peripheral devices, including network adapters, graphic accelerator boards, and embedded control modules. The pci_a MegaCore™ function implements a PCI interface.

This application note explains how to compile and simulate the pci_a function in FLEX® 10K designs using the Altera® MAX+PLUS® II development system.

The instructions in this application note assume the following:

- MAX+PLUS II version 7.2 or higher is installed in the **c:\maxplus2** directory.
- You are using a PC. The information in this application note is also applicable for UNIX users; however, you should alter the steps where appropriate.
- You have installed the pci_a function, and all the files are located in the default directory, **c:\megacore.** If the installation location is different on your system, substitute the appropriate pathname.
- You have purchased a license for the pci_a MegaCore function or are using the Altera OpenCore™ feature.

☞ You can use Altera's OpenCore feature to compile and simulate the pci_a MegaCore function. However, you must obtain a license from Altera before you can generate a Programmer Object File (**.pof**) for device programming.

pci_a

The pci_a function provides a solution for integrating 32-bit PCI peripheral devices, and is fully tested to meet the requirements of the PCI Special Interest Group (SIG) ***PCI Local Bus Specification, Revision 2.1,*** and ***Compliance Checklist, Revision 2.1***. The pci_a function is optimized for the EPF10K30RC240-3 device, reducing the design task and enabling you to focus efforts on the custom logic surrounding the PCI interface. The pci_a function is intended for use in Altera FLEX 10K devices with remaining logic resources available for user-defined local side (DMA control engine) customization.

For more information on the pci_a function, refer to *PCI Master/Target MegaCore Function with DMA Data Sheet*.

Supplied Files

Table 1 describes design and simulation files for this application note supplied with the pci_a MegaCore function.

Table 1. Files Supplied with pci_a MegaCore Function

File Name	Description
pci_top.gdf	Graphic Design File (**.gdf**). This file includes an instance of the pci_a with pins connected to all I/O ports.
pci_top.scf	Simulator Channel File (**.scf**) used to simulate the design

Creating the Project Directory

The following steps explain how to create your project directory.

1. Create a **pci** directory on your computer. This directory will be the project directory. Change to the directory after creating it. For example, type the following at a DOS command prompt:

```
mkdir c:\pci ↵
cd c:\pci ↵
```

2. Copy the **pci_top.gdf** and **pci_top.scf** files from the default directory (**c:\megacore\pci_a\walkthru**) to your **pci** directory.

Compiling the pci_a

The following steps explain how to compile the pci_a function using the MAX+PLUS II development system.

1. Start MAX+PLUS II.

2. In the **User Libraries** dialog box (Options menu), select **c:\megacore\lib** and choose **Add** to add the user library to the **pci** project. The Text Design File (**.tdf**), Symbol File (**.sym**) and Include File (**.inc**) for the pci_a function are stored in the **megacore\lib** directory.

3. Open the **pci_top.gdf** file (from the **pci** working directory) using **Open** (File menu). Figure 1 shows the GDF pin names and pci_a symbol port names. The PCI interface signals (inputs and outputs) are listed on the left side of Figure 1, and the local side signals (inputs and outputs) are listed on the right side.

4. Generate the Assignment & Configuration File (**.acf**). The following steps explain how to generate the ACF using the **make_acf** utility found in the **c:\megacore\bin** directory. For more information on the **make_acf** utility, refer to the documentation in the **c:\megacore\bin** directory.

a. Generate the **pci_top.acf** in the project directory by typing the following at a DOS command prompt:

```
c:\megacore\bin\make_acf ↵
```

b. You are then prompted with questions. Type the following after each question. (The bold text is the prompt text.)

```
Enter the hierarchical name for the PCI MegaCore:

|pci_a:35

Enter the chip name:

pci_top

Type the path and name of the output acf file:

pci_top.acf

Type the path and name of the input acf file:

c:\megacore\pci_a\acf\pci_a.acf
```

For this application note, the hierarchical name of this instance of the `pci_a` function is `|pci_a:35`. However, when the `pci_a` is instantiated in your custom project, you can obtain the hierarchical name by selecting the **pci_a.tdf** icon in the Hierarchy Display. The *Node Name* box in the **Pin/Location/Chip** dialog box (Assign menu) shows the hierarchical name that you should type when prompted for the PCI MegaCore hierarchical name.

5. Open the **pci_top.acf** generated by the **make_acf** utility (File menu). The **pci_top.acf** shows assignments made for the pci_top project. Choose **Save As** to make the assignments current (File menu).

6. If necessary, change the parameter values for the `pci_a` symbol with the **Edit Ports/Parameters** command (Symbol menu).

   The `pci_a` is a parameterized function. For example, the default value of the class code register is `FF0000` hexadecimal, but you can change the value by setting the `CLASS_CODE` parameter. Refer to the PCI-SIG's ***PCI Local Bus Specification, Revision 2.1*** for more information.

When changing a parameter value, be sure to change the number only, i.e., leave the hexadecimal H and quotation marks. If the H or quotation marks are deleted, a compilation error will result. Also, when setting register values, MAX+PLUS II may send several warning messages indicating that one or more registers are stuck at ground. These warning messages should be ignored.

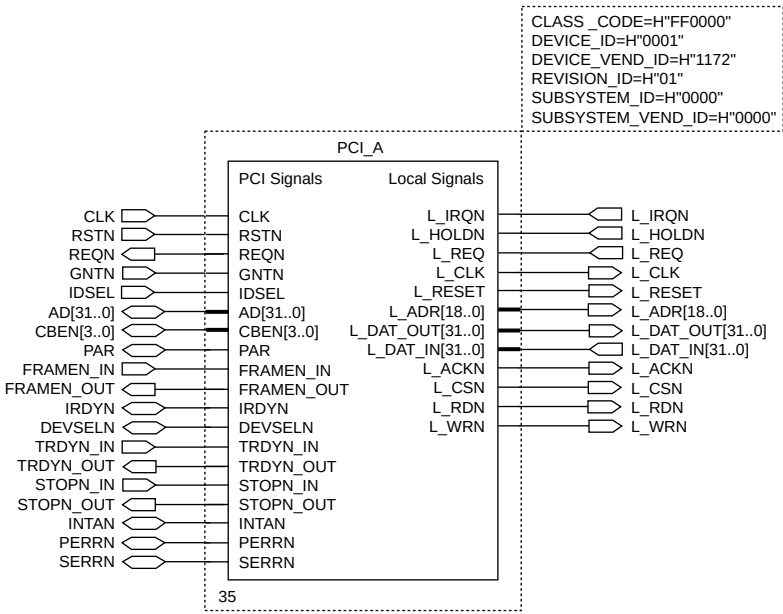
Table 2 shows the default parameter values.

Table 2. Default Parameter Values

Name	Default Value (Hexadecimal)	Description
CLASS_CODE	FF0000	Class code register
DEVICE_ID	0001	Device ID register
DEVICE_VEND_ID	1172	Device vendor ID register
REVISION_ID	01	Revision ID register
SUBSYSTEM_ID	0000	Subsystem ID register
SUBSYSTEM_VEND_ID	0000	Subsystem vendor ID register

Figure 1 shows the **pci_top.gdf** file with the default values.

Figure 1. pci_top.gdf File



7. If you changed the parameters, save your file by choosing **Save** (File menu).

8. Set your project to the current file by choosing the **Project Set Project to Current File** command (File menu).

   ☞ The ACF generated with the **make_acf** utility contains all project assignments necessary to meet PCI timing requirements, i.e., pin assignments for the PCI signals, location assignments for internal logic cells, and clique resource assignments. The `pci_a` is targeted for the EPF10K30RC240-3.

9. Compile the design.

Simulating pci_a

The following steps explain how to simulate the **pci_a** function using the MAX+PLUS II development system.

1. In the **Open** dialog box (File menu), select *Waveform Editor files* and choose **pci_top.scf** from the **pci** directory.

2. In the MAX+PLUS II Simulator, turn on the *Check Outputs* option.

3. Choose Inputs/Outputs (File Menu), specify **pci_top.scf** in the *Input* box, and choose OK.

4. Start the simulation.

   ☞ The example simulation file (**pci_top.scf**) provided illustrates a few typical simulation cycles. In your design, you will probably want to simulate the test bench scenarios provided with the `pci_a` MegaCore function as well as develop simulations for your customized local side. Also, with a MegaCore license, you will be able to generate a POF to download the design into the Altera PCI prototyping board for hardware testing.

Timing Analysis

The MAX+PLUS II Timing Analyzer can analyze the timing performance of a project after it has been optimized by the Compiler. To begin a timing analysis of the `pci_a`, choose **Registered Performance** (Analysis menu) in the Timing Analyzer and then choose **Start**.

Conclusion

The pci_a function provides a time-saving solution for integrating 32-bit PCI peripheral devices, which dramatically shortens design cycles and allows you to focus your efforts on the custom logic surrounding the PCI interface.

If you have any comments on the pci_a MegaCore function or supporting documentation, please send e-mail to **pci@altera.com**.

Notes:

2610 Orchard Parkway
San Jose, CA 95134-2020
(408) 544-7000
http://www.altera.com
Applications Hotline:
(800) 800-EPLD
Customer Marketing:
(408) 894-7104
Literature Services:
(888) 3-ALTERA
lit_services@altera.com