AN 532: An SOPC Builder PCI Express Design with GUI Interface

June 2008, ver. 1.0 Application Note 532

This application note teaches you how to build an SOPC Builder system that includes a PCI Express MegaCore function and download it to a development board. This application note builds on the concepts discussed in the design example found in the *PCI Express Compiler User Guide*. The design includes a GUI that you can use to specify reads, writes, and loopback commands to exercise the PCI Express bus. You can also use this GUI to read PCIe configuration registers. You can observe PCIe traffic in the GUI's **Display Window**.

This application describes building the SOPC Builder system and downloading it to your development board. You bring up the demonstration GUI by running a script included with the design files. Inexperienced users of the Quartus® II software and SOPC Builder should begin with step 1 below. If you are an experienced user of the Quartus II software and SOPC Builder, you can use the provided SRAM Object File (**.sof**) and skip to step 12 below.

You can download the files to run this example design from the Altera® website. A hyperlink to the design files appears next to this application note on the Application Notes web page.

SOPC Builder Design Flow

This design example consists of the following steps:

1. Create a Quartus II Project
2. Run SOPC Builder
3. Parameterize the PCI Express MegaCore Function
4. Add the Other Components to the System
5. Complete the Connections in SOPC Builder
6. Generate the SOPC Builder System
7. Integrate the Quartus II Project with the SOPC System
8. Update the Quartus II Settings File
9. Add Files to the Quartus II Project

10. Simulate the SOPC Builder System
11. Compile the Design
12. Program the Development Board

Create a Quartus II Project

To begin, you create a new Quartus II project with the **New Project Wizard**, which helps you specify the working directory for the project, assign the project name, and designate the name of the top-level design entity. To create a new project follow these steps:

1. Choose **Programs > Altera > Quartus II** <*version*> (Windows Start menu) to run the Quartus II software.

For information about installing the Quartus II software, refer to the *Quartus II Installation & Licensing for Windows* or *Quartus II Installation & Licensing for UNIX and Linux Workstations*.

2. On the Quartus II File menu, click **New Project Wizard**.

3. Click **Next** in the **New Project Wizard: Introduction** (The introduction does not display if you previously turned it off.)

4. On the **Directory, Name, Top-Level Entity** page, enter the following information:

   a. The working directory for your project. This design example uses the directory **c:\projects\sopc_s2gx_x4**

   b. The name of the project. This design example uses **sopc_s2gx_x4_example_top**.

   The Quartus II software specifies a top-level design entity that has the same name as the project automatically. Do not change this name.

5. Click **Next** to display the **Add Files** page.

   Click **Yes**, if prompted, to create a new directory.

6. If you installed the MegaCore® IP Library in a different directory from where you installed the Quartus II software, you must add the user libraries:

   a. Click **User Libraries**.

b. Type <*path*>**\ip** in the **Library name** box, where <*path*> is the directory in which you installed the PCI Express Compiler.

c. Click **Add** to add the path to the Quartus II project.

d. Click **OK** to save the library path in the project.

7. Click **Next** to display the **Family & Device Settings** page.

8. On the **Family & Device Settings** page, choose the following target device family and options:

   a. In the **Family** list, select **Stratix II GX**.

   b. In the **Available devices** list, select **EP2SGX90FF1508C3**.

9. Click **Next** to close this page and display the **EDA Tool Settings** page.

10. From the **Simulation** list, select **ModelSim**. From the **Format** list, select the HDL language you intend to use for simulation.

11. Click **Next** to display the **Summary** page.

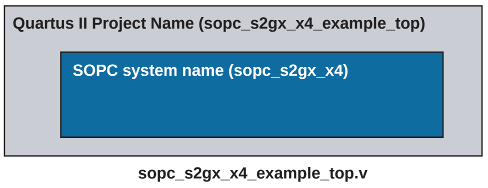
12. Check the **Summary** page to ensure that you have entered all the information correctly.

13. Click **Finish** to create the Quartus II project.

Run SOPC Builder

The SOPC Builder system that you create exists inside of the Quartus II project. Figure 1 illustrates the relationships between the modules. The top-level wrapper file, **sopc_s2gx_x4_example_top.v**, can be written in either Verilog HDL or VHDL, depending on the simulation language you choose. This file is required for system integration. The top-level design entity, sopc_s2xgx_x4_example_top, is the link between the wrapper file and the Quartus II project file. (Refer to Figure 4.)

Figure 1. Quartus II Project, SOPC System, and HDL Naming


Quartus II Project Name (sopc_s2gx_x4_example_top)
SOPC system name (sopc_s2gx_x4)
sopc_s2gx_x4_example_top.v

To launch the PCI Express MegaWizard interface in SOPC Builder, follow these steps:

1. On the Tools menu, click **SOPC Builder.** SOPC Builder appears.

Refer to Quartus II Help for more information on how to use SOPC Builder.

2. Type `sopc_s2gx_x4` in the **System Name** box, select your preferred simulation language under **Target HDL** and click **OK**.

3. To build your system by adding modules from the **System Contents** tab, under **Interface Protocols** in the **PCI** folder, double-click the **PCI Express Compiler** MegaCore component.

Parameterize the PCI Express MegaCore Function

The following section gives the parameters that are used in this design example. You may need to modify them for your design.

To parameterize the PCI Express MegaCore function in SOPC Builder, follow these steps:

On the **System Settings** page, specify the parameters in Table 1 and leave all the other parameters at their default values.

Table 1. System Settings

Option	Value
PHY type	**Stratix II GX**
Lanes:	**x4**
PCI Express version	**1.1**
Test out width	**512 bits**

On the **PCI Registers** page, specify the parameters in Table 2 for the PCI Base Address Registers.

Table 2. PCI Registers

BAR	Bar Type	Bar Size	Avalon Base Address
1:0	**64-bit Prefetchable Memory**	**1 MByte - 20 bits**	`0x80000000`
2	**32-bit Non-Prefetchable Memory**	**1 MByte - 20 bits**	`0x80000000`

Specify the values in Table 3 for the PCI Read-Only Registers. (These registers can be set in the MegaWizard Plug-in Manager and are read-only for software.)

Table 3. PCI Registers Read Only Registers

Option	Value
Device ID	**0x0005**
Vendor ID	**0x1172**
Class code	**0xFF0000**
Revision ID	**0x01**
Subsystem ID	**0x0005**
Subsystem vendor ID	**0x1172**

4. Click the **Capabilities** page and specify the settings in Table 4:

Table 4. Capabilities

Option	Value
Link Capabilities	Turn this option on
Link port number	Type `0x01`
Implement advanced error reporting	Turn this option off

Click the **Buffer Setup** page and specify the settings in Table 5.

Table 5. Buffer Setup

Option	Value
Maximum payload size	**128 Bytes**
Auto configure retry buffer size	Turn this option on
Maximum retry packets	**64**
Desired performance for received requests	**High**
Desired performance for received completions	**High**

5. You can keep the default settings on **Power Management** page which are **<64 ns** for **Endpoint LOs acceptable latency** and **<1 us** for E**ndpoint L1 acceptable latency**.

6. Click the **Avalon Configuration** page and specify the settings in Table 6.

Table 6. Avalon Configuration

Option	Value
Avalon Clock Domain	**Use separate clock**
PCIe Peripheral Mode	**Requester/Completer**
Address Translation Table Configuration	**Dynamic translation table**
Number of address pages	**2**
: Size of address pages	**1 MByte - 20 bits**

7. Click **Finish** to close the MegaWizard interface and return to SOPC Builder.

☞ Your system is not yet complete, so you can ignore any error messages generated by SOPC Builder at this stage.

Add the Other Components to the System

In this section you add the DMA controller and on-chip memory to your system.

1. In the **System Contents** tab, double-click **DMA Controller** in the **DMA** subfolder of the **Memories and Memory Controllers** folder. This module contains read and write master ports and a control port slave.

2. In the **DMA Controller** configuration wizard, specify the parameters listed in Table 7.

Table 7. Dma Controller

Option	Value
Width of the DMA length register	Type 13
Enable burst transfers	Turn this option on
Maximum burst size	**1024**
Construct FIFO from embedded memory blocks	Turn this option on

3. Click **Finish**. The DMA Controller module is added to your SOPC Builder system.

4. To specify the interrupt number, in the IRQ column, double-click the box next to the `control_port_slave` and type `1`.

5. In the **System Contents** tab, double-click the **On-Chip Memory (RAM or ROM)** in the **On-Chip** subfolder of the **Memory and Memory Controllers** folder. This module contains a slave port.

6. In the **On-Chip Memory** wizard, select **RAM (Writeable)**.

7. In the **Block type** list, select **Auto**.

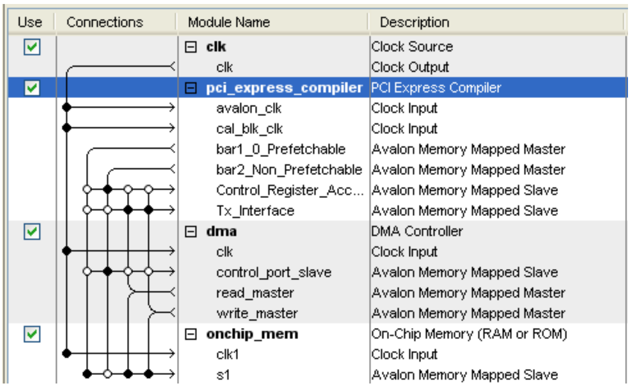
8. Under **Size**, select **64** for **Data width** and **4096 bytes** for **Total memory size**.

9. Click **Finish**. The On-chip Memory module is added to your SOPC Builder system.

Complete the Connections in SOPC Builder

In SOPC Builder, hovering the mouse over the Connections column displays the potential connection points between components, represented as dots connecting wires. A filled dot shows that a connection is made; an open dot shows a potential connection point that is not currently connected. Clicking a dot toggles the connection status. Figure 2 shows the open and closed dots of a connection panel.

Figure 2.



Use	Connections	Module Name	Description
✓		**clk**	Clock Source
		clk	Clock Output
✓		**pci_express_compiler**	PCI Express Compiler
		avalon_clk	Clock Input
		cal_blk_clk	Clock Input
		bar1_0_Prefetchable	Avalon Memory Mapped Master
		bar2_Non_Prefetchable	Avalon Memory Mapped Master
		Control_Register_Acc...	Avalon Memory Mapped Slave
		Tx_Interface	Avalon Memory Mapped Slave
✓		**dma**	DMA Controller
		clk	Clock Input
		control_port_slave	Avalon Memory Mapped Slave
		read_master	Avalon Memory Mapped Master
		write_master	Avalon Memory Mapped Master
✓		**onchip_mem**	On-Chip Memory (RAM or ROM)
		clk1	Clock Input
		s1	Avalon Memory Mapped Slave

To complete this design, create the following connections:

1. Connect the pci_express_compiler `bar1_0_Prefetchable` Avalon master port to the onchip_mem `s1` Avalon slave port using the following procedure:

   a. Click on the `bar1_0_Prefetchable` port then hover in the Connections column to display possible connections.

   b. Click on the open dot at the intersection of the onchip_mem `s1` port and the pci_express_compiler `bar1_0_Prefetchable` to create a connection.

2. Repeat step 1 to make the connections listed in Table 2–1.

Table 2–1. SOPC Builder Connections

Make Connection From:	To:
pci_express_compiler `bar2_Non_Prefetchable` Avalon master port	dma `control_port_slave` Avalon slave port
pci_express_compiler `bar2_Non_Prefetchable` Avalon master port	pci_express_compiler `Control_Register_access` Avalon slave port
dma `read_master` Avalon master port	onchip_mem `s1` Avalon slave port
dma `read_master` Avalon master port	pci_express_compiler `Tx_Interface` Avalon slave port
dma `write_master` Avalon master port	onchip_mem `s1` Avalon slave port
dma `write_master` Avalon master port	pci_express_compiler `Tx_Interface` Avalon slave port

To complete the system, follow these instructions for clock and address assignments:

1. By default, clock names are not displayed. To display clock names in the Module Name column and the clocks in the Clock column in the **System Contents** tab, click **Filter** to display the **Port Type Filters** dialog box. Turn on the **Clock** option and click **OK**.

2. In the Clock column, connect `clk` following these steps:

   a. Click in the Clock column next to the `cal_blk_clk` port. A list of available clock signals appears.

   b. Click `clk` to connect to provide a clock source for the `cal_blk_clk` of the pci_express_compiler.

   c. Under **Clock Settings**, double-click in the **MHz** box, type `125`, and press Enter.

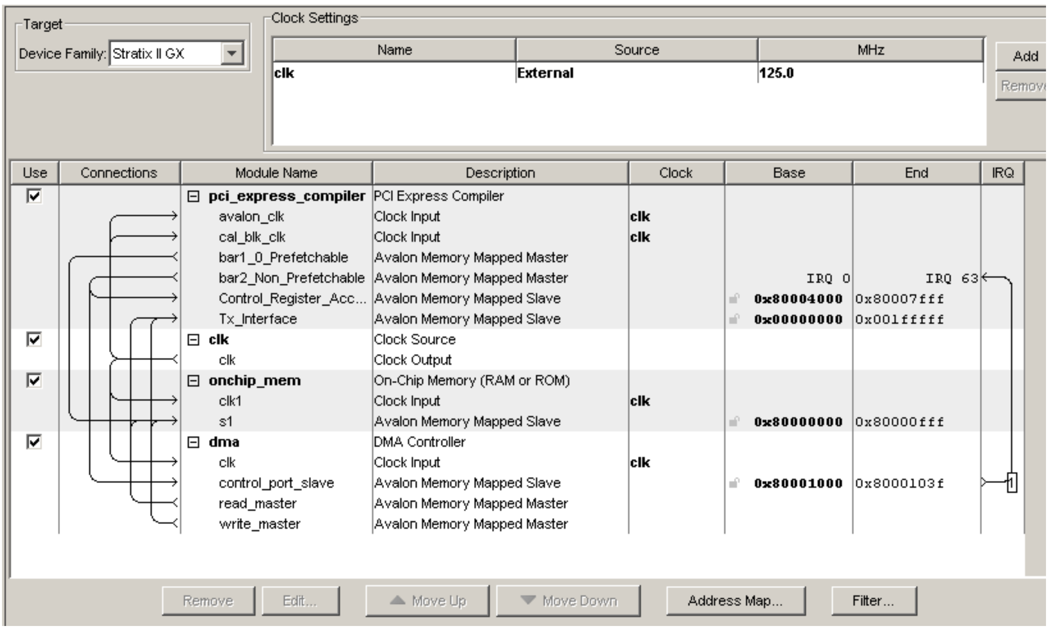
3. In the **Base** column, enter the base addresses in Table 2–2 for all the slaves in your system:

Table 2–2. Base Addresses for Slave Ports

Port	Address
pci_express_compiler `Control_Register_Access`	`0x80004000`
pci_express_compiler `Tx_Interface`	`0x00000000`
dma `control_port_slave`	`0x80001000`
onchip_mem `s1`	`0x80000000`

SOPC Builder generates informational messages indicating the actual PCI BAR settings. Figure 3 illustrates the complete system.

Figure 3. Completed SOPC Builder System



Device Family: Stratix II GX

Clock Settings

Name	Source	MHz
clk	External	125.0

Use	Connections	Module Name	Description	Clock	Base	End	IRQ
✓		**pci_express_compiler**	PCI Express Compiler				
		avalon_clk	Clock Input	clk			
		cal_blk_clk	Clock Input	clk			
		bar1_0_Prefetchable	Avalon Memory Mapped Master				
		bar2_Non_Prefetchable	Avalon Memory Mapped Master		IRQ 0	IRQ 63	
		Control_Register_Acc...	Avalon Memory Mapped Slave		0x80004000	0x80007fff	
		Tx_Interface	Avalon Memory Mapped Slave		0x00000000	0x001fffff	
✓		**clk**	Clock Source				
		clk	Clock Output				
✓		**onchip_mem**	On-Chip Memory (RAM or ROM)				
		clk1	Clock Input	clk			
		s1	Avalon Memory Mapped Slave		0x80000000	0x80000fff	
✓		**dma**	DMA Controller				
		clk	Clock Input	clk			
		control_port_slave	Avalon Memory Mapped Slave		0x80001000	0x8000103f	1
		read_master	Avalon Memory Mapped Master				
		write_master	Avalon Memory Mapped Master				

Generate the SOPC Builder System

1. In SOPC Builder, click **Next**.

2. On the **System Generation** tab, turn on **Simulation** and click **Generate**. After SOPC Builder reports successful system generation, on the File menu click **Save** to save the system.

Integrate the Quartus II Project with the SOPC System

The SOPC Builder system you generated is a subsystem of your Quartus II project. You need to create a simple wrapper file for the SOPC Builder subsystem that only exports the necessary signals to the top level.

Because this project uses the Stratix II GX internal transceiver to send and receive data; the four signals used to control the transceiver and communicate with an external PIPE interface are not used and should not be ports for the Quartus II project. A wrapper file included with this application note, **sopc_s2gx_x4_example_top.v**, ties the three unused inputs to fixed values and leaves the output signal floating, as follows:

```
.reconfig_clk_pci_express_compiler     (1'b0),
.reconfig_fromgxb_pci_express_compiler (),
.reconfig_togxb_pci_express_compiler   (3'b010),
.pipe_mode_pci_express_compiler        (1'b0),
```

You can copy this wrapper file to your project directory or create your own wrapper file if required.

Update the Quartus II Settings File

The Quartus II Settings File (**.qsf**) contains project-wide and entity-level assignments and settings for your project. This file includes the FPGA pin assignments for top-level signals. To prevent the four signals at the top level of the SOPC Builder project from being routed to pins, you must assign these signals to a `VIRTUAL_PIN`, as shown in the following example:

```
set_instance_assignment -name VIRTUAL_PIN ON -to reconfig_clk_pci_express_compiler
set_instance_assignment -name VIRTUAL_PIN ON -to reconfig_togxb_pci_express_compiler
set_instance_assignment -name VIRTUAL_PIN ON -to reconfig_fromgxb_pci_express_compiler
set_instance_assignment -name VIRTUAL_PIN ON -to pipe_mode_pci_express_compiler
```

The QAR file included with this application note includes these assignments in **sopc_s2gx_x4_example_top.qsf**. If you want to use this file, make sure the Quartus II library setting matches the version of Quartus II you are using. You can set the `USER_LIBRARIES` parameter by updating the <*ver*> in the following line:

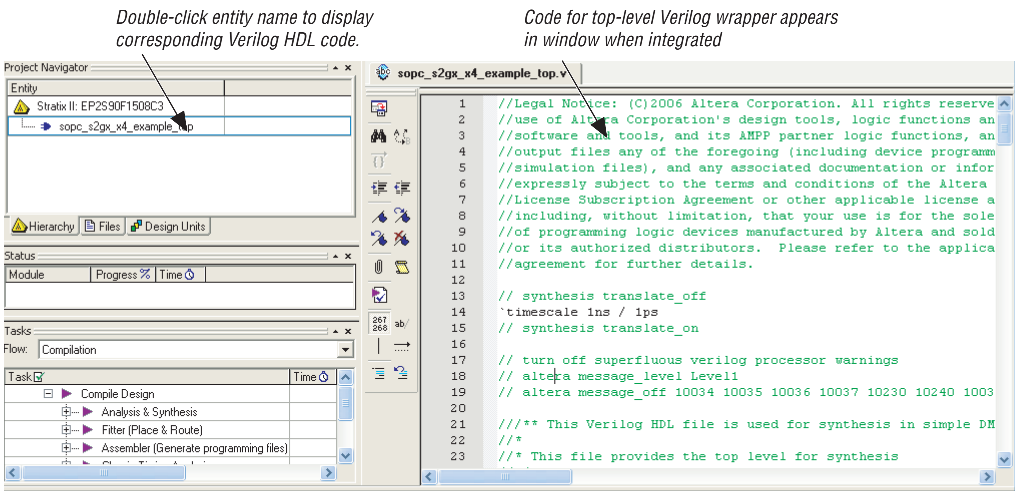
```
set_global_assignment -name USER_LIBRARIES"C:/altera/<ver>/ip/pci_express_compiler/lib;"
```

Add Files to the Quartus II Project

To add **sopc_s2gx_x4_example_top.v**, **sopc_s2gx_x4_example_top.qsf**, and **sopc_s2gx_x4_example_topc.sdc** (for timing constraints) to your Quartus II project, complete the following steps:

1. Copy these files from the **.zip** file that accompanies this application note to your project directory.
2. On the Quartus II Assignments menu, click **Settings**.
3. Click **Files**.
4. Browse to each file and click **Add**.
5. Click **OK**.
6. To verify that the top-level Verilog HDL file is linked to your Quartus II project double-click the entity name, **sopc_s2gx_x4_example_top**, in the Project Navigator and confirm that the corresponding code displays in the Quartus II Text Editor.

Figure 4. Verify Linkage between Top-Level Verilog HDL and Entity Files


Double-click entity name to display corresponding Verilog HDL code.
Code for top-level Verilog wrapper appears in window when integrated
Project Navigator
Entity
Stratix II: EP2S90F1508C3
sopc_s2gx_x4_example_top
Hierarchy
Files
Design Units
Status
Module
Progress %
Time
Tasks
Flow: Compilation
Task
Time
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
sopc_s2gx_x4_example_top.v

```
//Legal Notice: (C)2006 Altera Corporation. All rights reserve
//use of Altera Corporation's design tools, logic functions an
//software and tools, and its AMPP partner logic functions, an
//output files any of the foregoing (including device programm
//simulation files), and any associated documentation or infor
//expressly subject to the terms and conditions of the Altera
//License Subscription Agreement or other applicable license a
//including, without limitation, that your use is for the sole
//of programming logic devices manufactured by Altera and sold
//or its authorized distributors.  Please refer to the applica
//agreement for further details.

// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 1003

///** This Verilog HDL file is used for synthesis in simple DM
//*
//* This file provides the top level for synthesis
```

Simulate the SOPC Builder System

SOPC Builder automatically sets up the simulation environment for the generated system. SOPC Builder creates the **sopc_s2gx_x4_sim** subdirectory in your project directory and generates the required files and models to simulate your PCI Express system.

This section of the design example uses the following components:

- The system you created using SOPC Builder
- Simulation scripts created by SOPC Builder in the **projects\sopc_s2gx_x4\sopc_s2gx_x4_sim** directory
- The ModelSim software

You can also use any other supported third-party simulator to simulate your design.

The PCI Express testbench files are located in the **\projects\sopc_s2gx_x4\pci_express_compiler_examples\sopc\testbench** directory.

SOPC Builder creates IP functional simulation models for all the system components. The IP functional simulation models are the **.vo** or **.vho** files generated by SOPC Builder in your project directory.

For more information on IP functional simulation models, refer to the *Simulating Altera IP in Third-Party Simulation Tools* chapter in ***Volume 3: Verification*** of the *Quartus II Handbook*.

The SOPC Builder-generated top-level file also integrates the simulation modules of the system components and testbenches (if available), including the PCI Express testbench. The Altera-provided PCI Express testbench simulates a single link at a time. You can use it to verify the basic functionality of your PCI Express compiler system. The default configuration of the PCI Express testbench is predefined to run basic PCI Express configuration transactions to the PCI Express device in your SOPC Builder generated system. You can edit the PCI Express testbench **altpcietb_bfm_driver.v** or **altpcietb_bfm_driver.vhd** file to add other PCI Express transactions, such as memory read (`MRd`) and memory write (`MWr`).

Before modifying the **altpcietb_bfm_driver** module for your own project you should make your own copy of it in a different directory specific to your project. If you need to regenerate the SOPC Builder system containing the PCI Express MegaCore function variation, it overwrites all of the example subdirectories and files.

To simulate this design example, perform the following steps:

1. If you are running the Verilog HDL design example, edit the **altpcietb_bfm_driver.v** file in the **\projects\sopc_s2gx_x4\pci_express_compiler_examples\sopc\testbench** directory to enable target and DMA tests. Set the following parameters in the file to 1:

   - `parameter RUN_TGT_MEM_TST = 1;`
   - `parameter RUN_DMA_MEM_TST = 1;`

   If you are running the VHDL design example, edit the **altpcietb_bfm_driver.vhd** in the **\projects\sopc_s2gx_x4\pci_express_compiler_examples\sopc\testbench** directory to set the following parameters to 1.

   - `RUN_TGT_MEM_TST : std_logic := '1';`
   - `RUN_DMA_MEM_TST : std_logic := '1';`

The target memory and DMA memory tests in the **altpcietb_bfm_driver.v/.vhd** file enabled by these parameters only work with the SOPC Builder system as specified in this design example procedure. When designing an application, modify these tests to match your system.

2. Choose **Programs > ModelSim_**`<ver>`**> ModelSim** (Windows Start menu) to start the ModelSim simulator. In ModelSim, change your working directory to **\projects\sopc_s2gx_x4\sopc_s2gx_x4_sim**.

3. To run the script, type the following command at the simulator command prompt:

   ```
   source setup_sim.do↵
   ```

4. To generate waveform output for the simulation, type the following command at the simulator command prompt:

   ```
   do wave_presets.do↵
   ```

5. To compile all the files and load the design, type the following command at the simulator prompt:

   ```
   s↵
   ```

6. To simulate the design, type the following command at the simulator prompt:

   ```
   run -all↵
   ```

   The PCI Express test driver performs the following transactions with the status of the transactions displayed in the ModelSim simulation message window:

   - Various configuration accesses to the PCI Express MegaCore function in your system after the link is initialized
   - Setup of the address translation table for requests that are coming from the DMA component
   - Setup of the DMA controller to read 4 KBytes of data from the root port BFM's shared memory
   - Setup of the DMA controller to write the same 4 KBytes of data back to the root port BFM's shared memory
   - Data comparison and report of any mismatch

7. Exit the ModelSim tool after it reports successful completion.

Compile the Design

To compile your design, on the Processing menu, click **Start Compilation**.

Program the Development Board

Programming the development board includes the following steps:

1. Configure the Hardware
2. Install the Development Board
3. Download the Programming Files to the Development Board
4. Run the Application

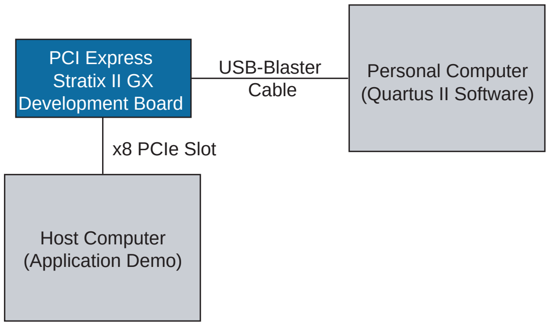
Configure the Hardware

The hardware configuration for this designs includes the PCI Express Development Board, Stratix II GX Edition and two computers, a host computer and your personal computer.

Refer to the product page for the PCI Express Development Board, Stratix II GX Edition for more information about this board.

You plug the PCI Express Development Board into the host computer which includes a x8 PCI Express slot. You use a USB-Blaster™ cable to connect your PC to the development board to program the board and download the software application. Figure 5 illustrates this hardware configuration.

Figure 5. Hardware Configuration for PCI Express Application


PCI Express Stratix II GX Development Board
USB-Blaster Cable
Personal Computer (Quartus II Software)
x8 PCIe Slot
Host Computer (Application Demo)

Install Drivers and Application

1. If you have not done so earlier, unzip the design files that accompany this application note.

This application only runs on 32-bit Windows XP operating system and uses the WinDriver version 8.11 from Jungo.

You can download the files to run this example design from the Altera website. A hyperlink to the design files appears next to this application note on Literature: Application Notes web page.

2. Copy the **windriver_811** directory to the host computer. If the host computer is not on your network, you can transfer the files using a USB flash drive.

3. On the host computer, open the **windriver_811** directory and double-click the **install.bat** file.

If you get a warning message saying, "The software you are installing for the hardware: Altera PCI Express Megafunction Beta has not passed Windows Logo testing to verify its compatibility with Windows XP," click **Continue anyway**.

4. After the installation completes, shut down the host computer.

Install the Development Board

1. Install the PCI Express development board in an x8 PCIe slot.

2. Restart the host computer.

3. The **Found New Hardware Wizard** appears.

4. Follow the instructions to install the Stratix II GX device.

Download the Programming Files to the Development Board

The Stratix II GX PCI Express development board is preprogrammed with a PCI Express design example. Complete the following steps to replace the original design:

1. Connect the USB-Blaster cable 10-pin female plug to the JTAG header (J5) on the development board. Note that pin 1 of J5 is marked on the PCB. On the cable, pin 1 is marked by a white line and the text, "PIN 1."

2. Connect the other end of the USB-Blaster cable to a USB port on your PC.

3. To open the programming GUI, on the Tools menu, click **Programmer**.

4. To display the devices in JTAG chain on your board, click **Auto Detect.**

5. To specify the programming file, complete the following steps:

   a. Right-click on the **EP2SGX90FF1508** and click **Change File**.

   b. Browse to your project directory and click **sopc_s2gx_x4_example_top.sof**.

6. For the host PC to ensure that the BIOS reliably enumerates the PCI Express development board, you must program the FPGA on that board before the boot sequence completes. Consequently, you must interrupt the boot sequence to program the FPGA. There are two ways to interrupt the boot sequence:

   - Press the F8 key to start the Windows boot manager while the host PC is booting. Then you can resume the boot sequence after you have programmed the FPGA.

or:

- Alternatively, if the operating system on your host PC recognizes the add-in card and displays a message that says, "Alert! Error initializing PCI Express slot 1," you can program the FPGA before acknowledging this notifier.

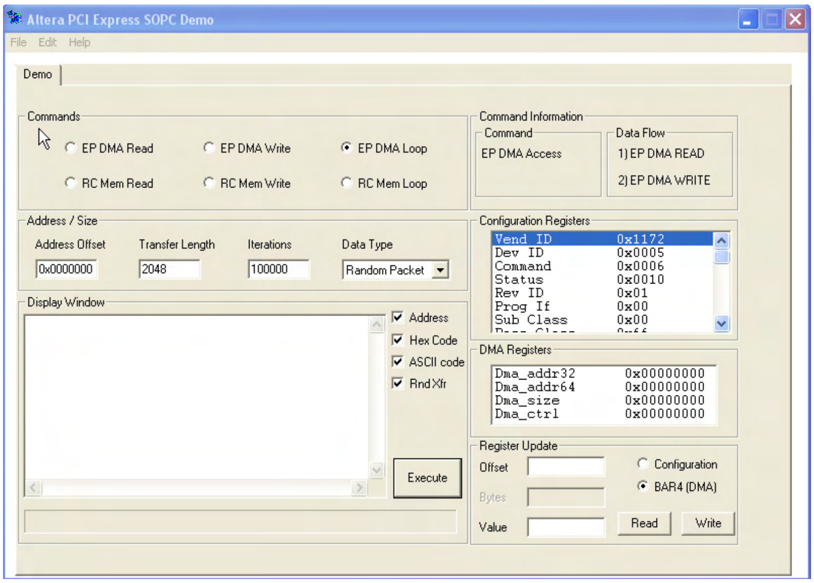
7. While the boot sequence is interrupted, program the device by turning on the **Program/Configure** option and clicking **Start**.

8. To complete the installation, you must reboot the host PC by typing Ctrl+Alt+Del to restart your computer without powering down.

Run the Application

To run the application demo GUI, click the **pcie.exe** file in the **\projects\sopc_s2gx_x4\windriver_811** directory. Figure 6 shows the PCI Express SOPC Builder Demo interface.

Figure 6. PCI Express SOPC Demo Application Window



Commands: ( ) EP DMA Read ( ) EP DMA Write [✓ EP DMA Loop] ( ) RC Mem Read ( ) RC Mem Write ( ) RC Mem Loop

Command Information — Command: EP DMA Access; Data Flow: 1) EP DMA READ, 2) EP DMA WRITE

Address / Size — Address Offset: 0x0000000; Transfer Length: 2048; Iterations: 100000; Data Type: Random Packet

Display Window options: [✓] Address [✓] Hex Code [✓] ASCII code [✓] Rnd Xfr

Configuration Registers:

```
Vend ID     0x1172
Dev ID      0x0005
Command     0x0006
Status      0x0010
Rev ID      0x01
Prog If     0x00
Sub Class   0x00
```

DMA Registers:

```
Dma_addr32   0x00000000
Dma_addr64   0x00000000
Dma_size     0x00000000
Dma_ctrl     0x00000000
```

Register Update — Offset: ; Bytes: ; Value: ; ( ) Configuration [✓ BAR4 (DMA)]

By default, the Demo interface is set up to run a DMA loopback test. You can run this test by clicking the **Execute** button. You can also use this interface to perform tests that include memory read and write transactions or to read the configuration registers.

Using the GUI Loop Commands

This section describes sequence transactions that occur when you run **EP DMA Loop** and **RC MEM Loop** commands.

RC MEM Loop

When you run the **RC Mem Loop** command, the PCI Express root complex on the host PC initiates a memory write to the PCIe endpoint on the development board. The root complex then reads the same location from the endpoint. You set the **Address Offset**, **Transfer Length**, **Iterations**, and **Data Type** then click the **Execute** button to start the test. After each write and read pair, the host PC compares the data to verify that it was transmitted correctly. The data is also printed to the **Display Window**. If the entire test finishes without error, the host CPU prints " * * * SUCCESS * * * " to the **Display Window**.

EP DMA Loop

When you run the **EP DMA Loop** command, the endpoint initiates a DMA read to the root complex. The root complex returns the read data with a read completion transaction. Then, the endpoint requests a DMA write to transfer the same data to the host PC. You set the **Address Offset**, **Transfer Length**, **Iterations**, and **Data Type** then click the **Execute** button to start the DMA loop test. After each DMA pair completes, the host PC compares the DMA data to verify that it was transmitted correctly. The data is also printed to the **Display Window**. If the entire test finishes without error, the host CPU prints " * * * SUCCESS * * * " to the **Display Window**.

References

This application note references the following documents and software:

- *PCI Express Compiler User Guide*
- *Quartus II Installation & Licensing for UNIX and Linux Workstations*
- *Quartus II Installation & Licensing for Windows*
- *Simulating Altera IP in Third-Party Simulation Tools* chapter in ***Volume 3: Verification*** of the *Quartus II Handbook*
- WinDriver information (**www.jungo.com**)

Document Revision History

Table 3 shows the revision history for this application note.

Table 3. Document Revision History

Date	Changes Made	Summary of Changes
June 2008 v1.0	Initial release	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/

Copyright © 2008 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. ModelSim is a registered trademark of Mentor Graphics. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001