PCI Compiler

October 2005, Compiler Version 4.1.0 — Release Notes

These release notes for the PCI Compiler version 4.1.0 contain the following information:

System Requirements

To use the PCI Compiler version 4.1.0, you require the following hardware and software:

- A computer running any of the following operating systems:
  - Windows 2000/XP
  - Red Hat Linux 8.0
  - Red Hat Enterprise Linux 3 WS (with support for 32-bit, AMD64, or Intel EM64T workstations)
  - Solaris 8 or 9 (32-bit or 64-bit)
- Quartus® II software version 5.1 or higher
- Adobe Reader version 5.0 or higher for viewing PDF documentation
- Mozilla Firefox 1.0 or higher on computers running the RedHat Linux or Solaris operating systems for following web links
- For simulation, ModelSim® Altera Edition, SE or PE simulator version 6.0e or higher or any of the Altera-supported, third-party simulation tools.

New Features & Enhancements

The following list outlines new features and enhancements in this release:

- Added preliminary support for Stratix® II GX and HardCopy® II device families.
- Updated support for Stratix II, MAX® II, and Cyclone™ II device families.
- Reduced device utilization for PCI-Avalon Bridge
- Fixed minor bugs

Errata Fixed in This Release

Although there were no errata from the previous release, for known issues fixed in this release, refer to the "Revision History" on page 18 for detailed descriptions these issues.

For existing up-to-date errata, refer to the PCI Compiler version 4.1.0 errata sheet on the errata page of the Altera website:

www.altera.com/literature/es/es_pci_compiler_410.pdf

Obtain & Install the PCI Compiler

Before you can use the PCI Compiler, you must obtain the files and install them on your computer. Altera MegaCore® functions can be installed from the MegaCore IP Library CD-ROM during or after Quartus II installation, or downloaded individually from the Altera website and installed separately.

☞ The following instructions describe downloading and installing the PCI Compiler. If you already have installed the PCI Compiler from the MegaCore IP Library CD-ROM, skip to "Directory Structure" on page 4.

If you have Internet access, you can download PCI Compiler from Altera's website at www.altera.com. Follow the instructions below to obtain the PCI Compiler from the Internet. If you do not have Internet access, contact your local Altera representative to obtain the MegaCore IP Library CD-ROM.

1. Point your web browser to **www.altera.com/ipmegastore**

2. Type **PCI** in the IP MegaSearch box.

3. Click **Go**.

4. Choose **PCI Compiler** <*PCI MegaCore function*> from the search results page, where <*PCI MegaCore function*> is the name of the PCI MegaCore function you want to use. The search results display the product description web page.

5. Click **Download Free Evaluation** on the product description web page.

6. Complete the registration form and click **Submit Request**.

7. Read the Altera MegaCore license agreement, turn on the I have read the license agreement check box, and click **Proceed to Download Page**.

8. Follow the instructions on the PCI Compiler download and installation page to download the appropriate PCI Compiler installation file.

Install the PCI Compiler

The following instructions describe how you install the PCI Compiler on computers running the Windows, Linux, or Solaris operating systems.

Windows

Follow these steps to install PCI Compiler on a PC running a supported version of the Windows operating system:

1. Choose `Run` (Start menu).

2. Type <*path*>`\pci_compiler-v4.1.0.exe`, where <*path*> is the location of the downloaded PCI Compiler installation executable.

3. Click **OK**. The **PCI Compiler Installation** dialog box appears. Follow the on-screen instructions to finish installation.

Solaris & Linux

Follow these steps to install PCI Compiler on computers running supported versions of the Solaris or Linux operating systems:

1. Move the compressed files to the desired installation directory and make that directory your current directory.

2. Decompress the package by typing the following command:

   `gunzip pci_compiler-v4.1.0_`*<operating system>*.tar.gz ↵

   where *<operating system>* is either `solaris` or `linux`.

3. Extract the package contents by typing the following command:

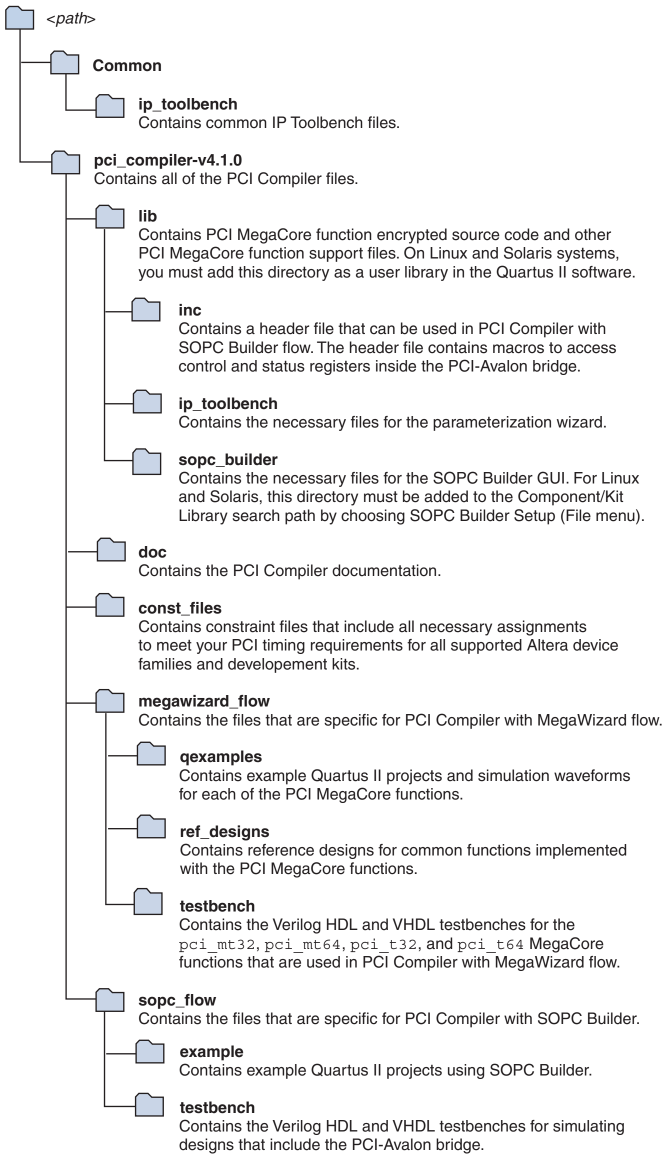
   `tar -xvf pci_compiler-v4.1.0_`*<operating system>*`.tar` ↵

   where *<operating system>* is either `solaris` or `linux`.

Directory Structure

Figure 1 shows the directory structure for the PCI Compiler, where <*path*> is the PCI Compiler installation directory. The default Windows installation directory is c:\altera\megacore.

Figure 1. PCI Compiler Directory Structure



- *<path>*
  - **Common**
    - **ip_toolbench**
      Contains common IP Toolbench files.
  - **pci_compiler-v4.1.0**
    Contains all of the PCI Compiler files.
    - **lib**
      Contains PCI MegaCore function encrypted source code and other PCI MegaCore function support files. On Linux and Solaris systems, you must add this directory as a user library in the Quartus II software.
      - **inc**
        Contains a header file that can be used in PCI Compiler with SOPC Builder flow. The header file contains macros to access control and status registers inside the PCI-Avalon bridge.
      - **ip_toolbench**
        Contains the necessary files for the parameterization wizard.
      - **sopc_builder**
        Contains the necessary files for the SOPC Builder GUI. For Linux and Solaris, this directory must be added to the Component/Kit Library search path by choosing SOPC Builder Setup (File menu).
    - **doc**
      Contains the PCI Compiler documentation.
    - **const_files**
      Contains constraint files that include all necessary assignments to meet your PCI timing requirements for all supported Altera device families and developement kits.
    - **megawizard_flow**
      Contains the files that are specific for PCI Compiler with MegaWizard flow.
      - **qexamples**
        Contains example Quartus II projects and simulation waveforms for each of the PCI MegaCore functions.
      - **ref_designs**
        Contains reference designs for common functions implemented with the PCI MegaCore functions.
      - **testbench**
        Contains the Verilog HDL and VHDL testbenches for the `pci_mt32`, `pci_mt64`, `pci_t32`, and `pci_t64` MegaCore functions that are used in PCI Compiler with MegaWizard flow.
    - **sopc_flow**
      Contains the files that are specific for PCI Compiler with SOPC Builder.
      - **example**
        Contains example Quartus II projects using SOPC Builder.
      - **testbench**
        Contains the Verilog HDL and VHDL testbenches for simulating designs that include the PCI-Avalon bridge.

PCI Timing Support

Designs that use an Altera PCI MegaCore function must use an Altera-provided PCI constraint file. A PCI constraint file does the following:

- Constrains Quartus II compilations so that your design meets PCI timing requirements
- Specifies the required PCI pin assignments for your board layout

The PCI Compiler installation includes PCI constraint files in the form of Tcl scripts that allow you to meet the PCI timing requirements in the Quartus II software. Refer to the"Directory Structure" on page 4 for the location of the constraint files.

Table 1 shows the list of constraint files shipped with this PCI Compiler Release.

Table 1. Constraint Files for Supported Devices

Constraint Filename	Family Supported
pci410_q51_stratixii.tcl	Stratix II
pci410_q51_stratixiigx.tcl	Stratix II GX
pci410_q51_stratix.tcl	Stratix
pci410_q51_stratixgx.tcl	Stratix GX
pci410_q51_cycloneii.tcl	Cyclone II
pci410_q51_cyclone.tcl	Cyclone
pci410_q51_maxii.tcl	MAX II
pci410_q51_stratix_board.tcl	Stratix PCI Kit and Stratix PCI Professional Kit

Refer to Appendix A in the *PCI Compiler User Guide* for detailed information about using PCI constraint files.

The constraint files use the following naming convention:

*<PCI Compiler Version>*_*<Quartus II Version>*
*<Family Supported>*`cf.tcl`

These constraint files have been tested against PCI Compiler 4.1.0 and Quartus II 5.1 and meet PCI timing.

To use the constraint file, follow these steps:

1. Copy the constraint file you need into your project directory. If you are using the SOPC Builder flow, this step is already done.

2. Open your project in the Quartus II software.

3. In the Quartus II software, choose Tcl Console (View > Utility Windows menu).

4. To source the constraint file, type the following in the Quartus II Tcl console:

```
source <Tcl filename>.tcl

add_pci_constraints [-speed "33" | "66"]
[-no_compile] [no_pinouts] [-help]
```

Upgrading from a Previous PCI Compiler Version

If you are upgrading from PCI Compiler version 3.0.0 or an earlier version, refer to "Upgrading from PCI Compiler Version 3.0.0 or Earlier" on page 8, then perform the procedure below. This procedure applies only to the MegaWizard® Plug-In Manager flow and **does not** apply to the SOPC Builder flow.

Perform the following steps to upgrade from a previous version of the PCI Compiler:

1. Install the new version of PCI Compiler in a directory of your choice.

   See the section, "Obtain & Install the PCI Compiler" on page 2.

2. Remove <*path*>`\pci_compiler-v`<*previous version*>`\lib` from the user libraries in the Quartus II software.

3. Add the <*path*>`\pci_compiler-v4.1.0\lib` directory of the new PCI Compiler installation as a user library in the Quartus II 5.1 software.

4. Choose **MegaWizard Plug-In Manger** (Tools menu) in the Quartus II software version 5.1.

5. In the MegaWizard Plug-In Manager dialog box choose **Edit an existing custom megafunction variation** and click **Next**.

6. Select the PCI wrapper file that you generated using a previous version of PCI Compiler.

   If you have uninstalled the previous version of PCI Compiler you get a warning indicating that Megafunction PCI Compiler is not listed in any **wizard.lst** in the specified library path. Wizard launch aborted. Click **OK** for the warning message.

7. In the **MegaWizard Plug-In Manger** dialog box, select **PCI Compiler v4.1.0** in the **Megafunction name** list and click **Next**.

   A dialog box displays reporting that the current file is based on a previous version of PCI Compiler, but that you have selected PCI Compiler version 4.1.0. Click **OK** to launch IP Toolbench.

8. IP Toolbench generates a PCI Compiler warning indicating the change in the local side interface. Click **Yes** to continue. This warning message will be displayed only if you are upgrading from PCI Compiler version 3.0.0 or an earlier version.

9. If you want to change any of the existing parameters, click **Step 2: Parameterize**.

10. After changing the required parameters, click **Finish**.

11. Click **Step 2: Set Up Simulation**.

12. Click **Generate Simulation Model** and select the language in the **Language Combo** box.

13. Click **Step 3: Generate** to generate the modified wrapper file and updated IP functional Simulation Model.

14. Update the HDL code that instantiates the variation file and IP functional Simulation model.

Upgrading from PCI Compiler Version 3.0.0 or Earlier

The PCI MegaCore functions included in PCI Compiler version 4.1.0 include a change to the local-side interface. The PCI MegaCore functions implement an interrupt disable register (command register bit 10) and an interrupt status register (status register bit 3) in the configuration space.

Refer to the *PCI Compiler User Guide* for more information about these registers.

These additional registers are reflected on the local-side interface on `cmd_reg[6]` and `stat_reg[6]`. After upgrading, the existing HDL code that instantiates the variation file needs to be modified to account for the change in the bit-width of `cmd_reg[5..0]` to `cmd_reg[6..0]` and `stat_reg[5..0]` to `stat_reg[6..0]`. This change does not affect the SOPC Builder flow.

Meeting PCI Timing Requirements With a New Version of the PCI Compiler

If you already have a constraint file that meets all PCI timing requirements with the previous version of PCI Compiler, attempt to compile your project using your existing constraint file and the new user libraries. If PCI timing requirements are not met, use the PCI Constraint file included in the PCI Compiler installation.

Refer to Appendix A in the *PCI Compiler User Guide* for detailed information about using PCI constraint files.

PCI Compiler v4.1.0 SOPC Builder Example

This section describes how to compile and simulate the SOPC Builder example project in Quartus II software.

Follow these steps to compile the SOPC Builder example:

1. Open the project file **chip_top.qpf** in Quartus II software

2. Open SOPC Builder in the Quartus II software 'Tools' menu by selecting **SOPC Builder...**

   For Solaris and Linux Operating system, follow these steps:

   a. Choose SOPC Builder Setup (File menu)

   b. Add <*path*>`/pci_compiler-v4.1.0/lib/sopc_builder` to the **Component/Kit Library Search Path** box

3. Generate the SOPC Builder system.

   In SOPC Builder click the **Generate** button, and then exit back to Quartus when generation is complete.

   The project shipped in **sopc_builder_example** directory has PCI Constraints specified in **.qsf** (quartus setting file). Therefore, you are not required to reapply PCI constraints.

Steps for Simulating the SOPC Builder Example

To simulate the project, perform the following steps:

☞ The PCI testbench `mstr_tranx` located in the **pci_sim/verilog/mt32** directory has the command specified in the `USER COMMAND` section that will exercise target transaction to on-chip memory and DMA transactions to transfer data from PCI testbench `trgt_tranx` to on-chip memory, and also from the on-chip memory to the PCI testbench `trgt_tranx`.

1. Start the Modelsim-Altera simulator. In the simulator change your working directory to the directory below

   <path>`/pci_compiler-v4.1.0/sopc_builder_example`
   `/chip_top_sim`

2. Run the script. When prompted, type the following command in the simulator command prompt

   ```
   >source setup_sim.do
   ```

3. Compile all the files and load the design.

4. Type the following command in the simulator command prompt:

   ```
   >s
   ```

5. To see all the signals in the wave window, type the following command in the simulator command prompt:

   ```
   >do wave_presets.do
   ```

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

   ```
   >run -all
   ```

Compiling the Project in Quartus II Software

To Compile the project in Quartus II software perform the following steps

1. Choose **Start Compilation** (processing menu) in the Quartus II software.

2. After compilation, expand the Timing Analyzer folder in the compilation report by clicking the **+**symbol next to the folder name. Note the values of `Clock Setup`, `tsu`, `th`, and `tco` report sections.

pci_mt32 MegaCore Function Reference Design

The pci_mt32 MegaCore Function Reference Design example illustrates how to interface local logic to the pci_mt32 MegaCore function. The reference design includes a target and a master interface to the pci_mt32 function and the SDRAM memory. The DMA engine is implemented in the local logic to enable the pci_mt32 function to operate as a bus master. The design implements a FIFO interface to solve latency issues when data is transferred between the PCI bus and the SDRAM.

pci_mt32 MegaCore Function Reference Design Software Requirements

The pci_mt32 MegaCore Function Reference Design requires the Quartus II software version 5.1 or higher.

pci_mt32 MegaCore Function Reference Design Directory Structure

Table 2 on page 12 describes the directory structure of the pci_mt32 MegaCore Function Reference Design. The directory names are relative to the following path:

```
<path>/pci_compiler-v4.1.0/megawizard_flow
/ref_designs/ref_designs/pci_mt32/vhdl
```

where *<path>* is the directory in which you installed the PCI Compiler.

See Table 2, the pci_mt32 MegaCore Function Reference Design Directory Structure for more details regarding the directory structure.

Table 2. Directory Structure of pci_mt32 MegaCore Reference Design

Directory Name	Description
chip_top	This directory contains a top-level design file that instantiates the following modules: ● pci_mt32 MegaCore function variation file ● Local interface logic ● SDR SDRAM interface ● SDR SDRAM controller
pci_top	This directory contains a pci_mt32 MegaCore function top-level wrapper file. This wrapper file was generated using IP Toolbench with the following parameters selected using the **Parameterize - PCI Compiler Wizard**: ● BAR0 reserves 1MB of memory space ● BAR1 reserves 32MB of memory space
pci_local	This directory contains local interface logic files. For more information on these files, refer to FS 12: pci_mt32 MegaCore Function Reference Design.
sdr_intf	This directory contains files for the interface logic between the PCI local interface logic and the SDR SDRAM controller.
sdr_cntrl	This directory contains files for the SDR SDRAM controller.

Quartus II Synthesis & Compilation Instructions for the pci_mt32 MegaCore Function Reference Design

You can compile this design targeting the following device families:

- Stratix II
- Stratix GX
- Stratix
- Cyclone II
- Cyclone

To compile the pci_mt32 MegaCore Function Reference Design in the Quartus II software, perform the following steps:

1. Create a new project in the Quartus II software, specifying <*path*>`/pci_compiler-v4.1.0/megawizard_flow/ref_designs/pci_mt2/vhdl/chip_top.vhd` as the top-level design file.

2. Add the following directories as user libraries in the Quartus II software:

```
<path>/pci_compiler-v4.1.0/lib

<path>/pci_compiler-v4.1.0/megawizard_flow/
ref_designs/pci_mt32/vhdl/chip_top

<path>/pci_compiler-v4.1.0/megawizard_flow
/ref_designs/pci_mt32/vhdl/pci_local

<path>/pci_compiler-v4.1.0/megawizard_flow
/ref_designs/pci_mt32/vhdl/sdr_intf

<path>/pci_compiler-v4.1.0/megawizard_flow
/ref_designs/pci_mt32/vhdl/sdr_cntrl
```

Refer to Quartus II help for information on how to add user libraries in the Quartus II software.

3. Include the following files in your Quartus II project:

```
<path>/pci_compiler-v4.1.0/megawizard_flow
/ref_designs/pci_mt32/vhdl/chip_top
/vhdl_components.vhd

<path>/pci_compiler-v4.1.0/megawizard_flow
/ref_designs/pci_mt32/vhdl/pci_top/pci_top.vhd
```

4. Select the appropriate Altera device for your project.

   Use an Altera-provided PCI constraint file for the device you have selected

Refer to Appendix A of the *PCI Compiler User Guide* for more information on using PCI constraint files.

5. Compile your project.

pci_mt64 MegaCore Function Reference Design

The pci_mt64 MegaCore Function Reference Design is an example that shows how to connect the local-side signals of the Altera pci_mt64 MegaCore function to local-side applications when the MegaCore function is used as a master or target on the PCI bus. The reference design consists of the following elements:

- Master control logic
- Target control logic
- DMA engine
- Data path FIFO buffer functions
- SDRAM interface

pci_mt64 MegaCore Function Reference Design Software Requirements

The pci_mt64 MegaCore Function Reference Design requires the Quartus II software version 5.1 or higher.

pci_mt64 MegaCore Function Reference Design Directory Structure

Table 3 describes the directory structure of the pci_mt64 MegaCore Function Reference Design. The directory names are relative to the following path:

```
<path>/pci_compiler-v4.1.0megawizard_flow
/ref_designs/pci_mt64/vhdl
```

where *<path>* is the directory in which you installed the PCI Compiler.

Table 3. Directory Structure of pci_mt64 MegaCore Reference Design

Directory Name	Description
chip_top	This directory contains a top-level design file that instantiates the following modules: ● pci_mt64 MegaCore function variation file ● Local interface logic ● SDR SDRAM interface ● SDR SDRAM controller
pci_top	This directory contains a pci_mt64 MegaCore function top-level wrapper file. This wrapper file was generated using IP Toolbench with the following parameters selected using the **Parameterize - PCI Compiler Wizard**: ● BAR0 reserves 1MB of memory space ● BAR1 reserves 32MB of memory space

Table 3. Directory Structure of pci_mt64 MegaCore Reference Design

Directory Name	Description
pci_local	This directory contains local interface logic files. For more information on these files, refer to FS 10: pci_mt64 MegaCore Function Reference Design.
sdr_intf	This directory contains files for the interface logic between the PCI local interface logic and the SDR SDRAM controller.
sdr_cntrl	This directory contains files for the SDR SDRAM controller.

Quartus II Synthesis & Compilation Instructions for pci_mt64 MegaCore Function Reference Design

You can compile the pci_mt64 MegaCore Function Reference Design targeting the following device families:

- Stratix II
- Stratix
- Stratix GX
- Cyclone
- Cyclone II

To compile the pci_mt64 MegaCore Function Reference Design in the Quartus II software, perform the following steps:

1. Create a new project in the Quartus II software, specifying the top-level design file as follows:

   ```
   <path>/pci_compilerv-4.1.0/megawizard_flow
   /ref_designs/pci_mt64/vhdl/chip_top.vhd
   ```

2. Add the following directories as user libraries in the Quartus II software:

   <path>`/pci_compiler-v4.1.0/lib`

   <path>`/pci_compiler-v4.1.0/megawizard_flow`
   `/ref_designs/pci_mt64/vhdl/chip_top`

   <path>`/pci_compiler-v4.1.0/megawizard_flow`
   `/ref_designs/pci_mt64/vhdl/pci_local`

   <path>`/pci_compiler-v4.1.0/megawizard_flow`
   `/ref_designs/pci_mt64/vhdl/sdr_intf`

   <path>`/pci_compiler-v4.1.0/megawizard_flow`
   `/ref_designs/pci_mt64/vhdl/sdr_cntrl`

Refer to Quartus II help for information on how to add user libraries in the Quartus II software.

3. Include the following files in your Quartus II project:

   <path>`/pci_compiler-v4.1.0/megawizard_flow`
   `/ref_designs/pci_mt64/vhdl/chip_top`
   `/vhdl_components.vhd`

   <path>`/pci_compiler-v4.1.0/megawizard_flow`
   `/ref_designs/pci_mt64/vhdl/pci_top/pci_top.vhd`

4. Select the appropriate Altera device for your project.

5. Use an Altera-provided PCI constraint file for the device you have selected.

Refer to Appendix A of the *PCI Compiler User Guide* for more information on using PCI constraint files.

6. Compile your project.

Contact Altera

Although every effort has been made to ensure that this version of the PCI Compiler works correctly, if problems occur, use the following contact information to communicate issues to the appropriate Altera representative.

For technical support or other information about Altera products, go to the Altera website at **www.altera.com**. You can also contact Altera through your local sales representative or any of the sources listed in Table 4.

Table 4. Contacting Altera

Information Type	USA & Canada	All Other Locations
Technical support	www.altera.com/mysupport/	www.altera.com/mysupport/
	800-800-EPLD (3753) 7:00 a.m. to 5:00 p.m. Pacific Time	+1 408-544-8767 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	www.altera.com	www.altera.com
Altera literature services	literature@altera.com	literature@altera.com
Non-technical customer service	800-767-3753	+ 1 408-544-7000 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
FTP site	ftp.altera.com	ftp.altera.com

Revision History

Table 5 shows the revision history for the PCI Compiler version 4.1.0.

Table 5. PCI Compiler Revision History

Version	Date	Revision
4.1.0	October 2005	● Added preliminary support for Stratix II GX and HardCopy II device families. ● Updated support for Stratix II, MAX II, and Cyclone II device families. ● Reduced device utilization for PCI-Avalon Bridge ● Fixed the following Megawizard Plugin Manager flow problems: - The CIS pointer default value was made to be 0x00000000 instead of 0x00000001 ● Fixed the following SOPC Builder flow problems: -The prefetchable write transactions were made to disconnect at BAR boundary. - Fixed a configuration address translation bug that prevented the selection of `idsel=AD[31]` - Fixed the issue in which for 32-bit master read transactions, byte enables from the Avalon side were not being passed to the PCI side. - Corrected the issue where `intan` was asserted after `reset` if the Control register module is not implemented. ● Fixed the following issues that affect both flows: - The pci_mt64 did not end the PCI master transactions immediately following the `lm_lastn` assertion in the case of a 32-bit PCI target response to a 64-bit write request and the PCI Target is asserting random wait states. - In the case of an Address parity error detected by the PCI core, the PCI core used to also enable the perrn drivers although it does not assert `devseln`. - Fixed protocol violation where `stopn` was asserted after the end of the transaction in the case of wait states on `irdyn`. - Fixed protocol violation where `framen` was not deasserted immediately after detecting `stopn` asserted during 64-bit master write transaction to 32-bit target with target wait states. - Fixed a protocol violation where `framen` was asserted for one cycle during single cycle DAC memory read transaction.
4.0.0	April 2005	● Updated user guide of the PCI Compiler. Divided the user guide into two sections. The first section contains the previous version of the user guide, which describes the PCI Compiler with the MegaWizard flow. The second section describes the PCI Compiler with the SOPC Builder flow, new feature added in v4.0.0 of the PCI Compiler. ● Provided an SOPC Builder ready PCI component. ● PCI MegaCore function fixes ● Corrected the problem when `framen` and `req64n` are not asserted properly during dual address cycle transactions if `lm_lastn` signal is asserted during the address phase in the single-cycle master read transaction.
3.2.0	June 2004	● Added preliminary support for Cyclone II device family ● Added support for the *PCI Local Bus Specification, Revision 3.0*

Table 5. PCI Compiler Revision History

Version	Date	Revision
3.1.0	April 2004	● Added preliminary support for MAX II device family Added support for the *PCI Local Bus Specification, Revision 2.3* Concatenated the constraint files of all the device densities and packages for a given device family to a single constraint file Tcl script. For example, the **pci310_q40sp1_cyclone_cf.tcl** Tcl script includes PCI constraints for all the PCI supported devices and packages in the Cyclone family. Includes constraint files for all supported device families and development boards Modified the installation directory structure to include example Quartus II projects for all PCI MegaCore functions in the **/qexample** directory PCI MegaCore function fixes. ● Corrected the problem that had previously caused the PCI bus to behave badly when a single cycle master write transaction, in which `lm_rdyn` was asserted in the same clock cycle as `lm_adr_ackn` on the local side interface, resulted in the `framen` signal on the PCI bus to stay low.
3.0.0	February 2004	● Added preliminary support for Stratix II device family ● Added OpenCore® Plus evaluation feature ● Updated the pci_mt64 and pci_mt32 reference designs to support Stratix II, Stratix, Stratix GX, and Cyclone devices ● Made PCI MegaCore function fixes: - Corrected the master's handling retry in the case where the local logic does not assert `lm_rdyn` signal. - Corrected the target read data output when the `lt_rdyn` signal from the previous target write transaction is overly extended to the current transaction. - Fixed the problem where incorrect data was transferred to the local side when `lm_req32n` or `lm_req64n` was asserted by the local master logic and the PCI MegaCore function also was performing a target write transaction.
2.4.0	September 2003	● Improved timing for Stratix and Cyclone device families. ● Added IP Toolbench, a toolbar from which you can quickly and easily view documentation, specify the MegaCore function (pci_mt64, pci_mt32, pci_t64, or pci_t32), customize the MegaCore function, and generate all of the files necessary for integrating the parameterized MegaCore function variation into your design.You also can use this to generate an IP functional simulation model that you can use to verify your custom PCI MegaCore function in any Altera-supported simulator. ● PCI MegaCore function fixes: ● Corrected the issue with pci_mt64 and pci_mt32 where `irdy`, under rare conditions, is not asserted properly during the last data phase of a master memory burst write transaction. The conditions include cases where the PCI target issues a disconnect with data and the local side de-asserts `lm_rdyn` in the same clock cycle.
2.3.0	February 2003	● Improved timing for Stratix device family. ● PCI MegaCore function fixes.

Table 5. PCI Compiler Revision History

Version	Date	Revision
2.2.0	September 2002	● Added Cyclone Device Support Updated pci_mt32 and pci_mt64 reference designs for pci_mt64 and pci_mt32 MegaCore functions v2.2.0. ● Consolidated documentation and reduced number of PDF documents. ● 66-MHz support for FLEX® 10KE and ACEX® 1K devices is supported in Quartus II software. ● 66-MHz support in MAX+PLUS II is discontinued. ● Corrected PCI MegaWizard issues in Solaris platform. Updated PCI Behavioral models to use latest simulator and VIP software. ● Updated PCI Reference designs to the latest pci_mt64 and pci_mt32 revisions. ● PCI MegaCore function fixes: ● Corrected behavior of `l_ldat_ackn` and `l_hdat_ackn` when an incomplete master read transaction if followed by target read transaction. ● Corrected behavior during bus parking while the bus is not idle. ● Master will indicate master abort even during configuration transactions. ● Corrected behavior of `ack64n` getting asserted for one additional clock during rare master local wait cycles during a target retry condition

Table 5. PCI Compiler Revision History

Version	Date	Revision
2.0.0	August 2001	● Improved the look and feel of the PCI Compiler wizard and added support for new PCI MegaCore features. Updated the PCI MegaCore functions to v2.0.0. See the revision history in the readme files for each individual PCI MegaCore function for details on the changes. Included behavioral simulation models for all PCI MegaCore functions to support VHDL and Verilog HDL simulators. Included an open source PCI testbench in VHDL and Verilog HDL. Updated pci_mt32 and pci_mt64 reference designs for pci_mt64 and pci_mt32 MegaCore functions v2.0.0. Due to enhanced support in the PCI Compiler wizard for constraint files, the command line utility set_constraint was removed from the PCI Compiler. Use the PCI Compiler MegaWizard to create project specific constraint files for your design. **Added PCI MegaCore function features:** ● Added the **Add Internal Data Steering Logic for 32/64-Bit Systems** option, which provides improved logic optimization to let you attain 66 MHz performance more easily. When disabled, this option removes the internal data steering logic from `l_dato[63..32]` and `l_beno[7..4]`, providing valid data on `l_dato[31..0]` and `l_beno[3..0]` during a 32-bit transaction. When enabled, valid data is available on `l_dato[63..0]` and `l_beno[7..0]` during a 32-bit transaction and the operation is fully compatible with previous versions of pci_mt64. The Add Internal Data Steering Logic for 32/64-Bit Systems option is available in the PCI Compiler wizard; the option sets bit 14 of the `ENABLE_BITS` parameter. ● Added the **Host Bridge Enable** option to allow the pci_mt64 function to be used as a system host. When enabled, the option causes the pci_mt64 MegaCore function to power up with the master enable bit in the command register hardwired to 1 and allows the master interface to initiate configuration read and write transactions to the internal configuration space. The Host Bridge Enable option is available in the PCI Compiler wizard; the option sets bit 13 of the `ENABLE_BITS` parameter. ● Added the **64-Bit Only Devices option** to provide enhanced functionality for 64-bit only systems. When enabled, the option improves the latency of the pci_mt64 master PCI signals (especially `irdyn`) to be the same as those in 32-bit transactions. This feature also enables single-cycle master writes and adds the functionality previously offered through the `PCI_64BIT_SYSTEM` parameter. The 64-Bit Only Devices option is available in the PCI Compiler wizard; the option sets bit 16 of the `ENABLE_BITS` parameter. ● The **PCI_64BIT_SYSTEM parameter** has been disabled. You must enable this option in designs that previously used the `PCI_64BIT_SYSTEM` parameter. ● Added the ability to permanently **disable the pci_mt64 master latency timer registerm** which prevents the latency timer from expiring. The option is available in the PCI Compiler wizard; the option sets bit 15 of the `ENABLE_BITS` parameter. ● **Enabled the master** to keep the PCI bus when `gntn` is asserted and the latency timer expires. Additionally, the local side signal `lm_tsr[4]`, which indicates that the latency timer has expired, is not set until `gnt` is removed.

Table 5. PCI Compiler Revision History

Version	Date	Revision
1.3.0	February 2001	● First release ● PCI MegaCore functions included the pci_mt64, pci_mt32, pci_t64, and pci_t32 functions.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
literature@altera.com

I.S. EN ISO 9001