CorePCI 5.32 Release Notes

This document describes the new features and enhancements in the CorePCI5.32 release. CorePCI5.32 is a patch release that MUST be installed on top of the CorePCI5.3 or CorePCI5.31 release; it provides a complete new set of VHDL source files that resolve four key issues.

1. PCI compliance relating to FRAME de-assertion when STOP is asserted by a target while the core is in carrying out master transfers.
2. The number of logic levels on the PCI inputs has been reduced to allow the place and route to easily meet the PCI setup requirements. This change is recommended to meet PCI timing for master functions in the RTSX-S family.
3. A second release structure is included to allow the core to be easily loaded into the Libero Environment.
4. A simplified user testbench is provided to reduce simulation run times within the Libero Environment.

WARNING: This patch only updates the VHDL source files; customers using Verilog source code should contact Actel customer support.

Installation Instructions

You must install the CorePCI5.3 or CorePCI5.31 release before installing the CorePCI5.32 patch release. The CorePCI5.3 software and installation instructions are included on the CorePCI5.32 CD.

If you have already installed the CorePCI5.3 or CorePCI 5.31 software, you are ready to proceed with the installation of CorePCI5.32. To do so:

1. **Copy the CorePCI5.32 update zip file from the CorePCI5.32 CD or electronic delivery (ftp or email) to a temporary location on the hard disk and unzip the files.**
2. **In the *CorePCI5.3/vhdl* directory rename the *src* directory to *src_old*.**
3. **In the *CorePCI5.3/vhdl* directory rename the *wrappers* directory to *wrappers_old*.**

4. **In the *CorePCI5.3* directory rename the *tbench* directory to *tbench_old*.**
5. **Copy the unzipped *vhdl/src* directory to *CorePCI5.3/vhdl/src*.**
6. **Copy the unzipped *vhdl/wrappers* directory to *CorePCI5.3/vhdl/wrappers*.**
7. **Copy the unzipped *tbench* directory to *CorePCI5.3/tbench***
8. **Copy the unzipped Libero directory to CorePCI5.3/Libero**

Other Changes Required

CorePCI5.32 uses an additional low-level library file cm8dxe.vhd. The Synplicity project files must be updated manually to add this file to the synthesis project.

You have now updated the *vhdl/src*, *vhdl/wrappers* and *tbench* directories to the 5.32 release. You can now perform simulation and synthesis using the new VHDL files (as described in the *CorePCI5.3 Users Guide*).

Supported Tool Flows

Use Libero 2.3-SP2 or Designer R1-2003 SP2 (or later) with the CorePCI5.32 release.

Resolved Issues

Table 1 lists all the updates and fixes for the CorePCI5.32 software release.

Table 1. Software Action Requests (SARs) Resolved in the CorePCI5.32 Release

SAR No	Description
23711	lib_xxx directories are inconsistent
24027	Target/Master Fails timing in Designer R1-2003
24542	Core does not disconnect at Memory (BAR) Boundaries

Table 1. Software Action Requests (SARs) Resolved in the CorePCI5.32 Release (Continued)

SAR No	Description
25059	PCI Circuitry can fail when GRANT removed for a single cycle (Illegal)
25061	No reset on the signal IRDYND1 in ad_phase64.vhd
25139	If a PCI target ends in an illegal way the core may carry on driving the bus
25466	tdma64 non SDRAM wrapper is incorrect
25582	Wrapper files have missing ports on internal blocks
25960	Latency Timer violates the PCI Specification
26030	BE_REQ input is not honored whilst PCI bus is busy to other devices
27141	Need to create Libero data structures in the release
27143	Core violates STOP to FRAME de-assertion specification
27956	Code uses inefficient if else-if structures in configuration logic
28369	Watchdog not reset if a disconnect after 7-cycles occurs
28370	PCI input signals have to many logic levels for RTSX silicon
28390	PCI core testbench to have a resolution of 1ps. (LiberoSP2 Compatibility)

Known Issues

Table 2 lists unresolved SARs in the CorePCI5.32 release.

Table 2. Unresolved Software Action Requests (SARs) in the CorePCI5.32 Release

SAR No	Description
12131	The verilog_setup.do script for MTI does not work on PC

Table 2. Unresolved Software Action Requests (SARs) in the CorePCI5.32 Release

SAR No	Description
12160	SX32A-BG329 with JTAG pins restricted causes an error with the pin file
22631	APA FG456 Pin out, INTAN at wrong end of device
24553	MTI scripts do not compile the non DMA wrappers
24755	Pin definitions for A54SXA devices should be upward compatible to SX72A
24964	64 bit cores do not operate correctly when plugged into 32 bit buses
25205	Users Guide should mention the break set on the pad enable pins
27825	RTSXS Implementations can violate PCI hold times
28076	Disconnect occurs instead of target abort when ERROR asserted for a single clock cycle

SAR24964 is a PCI compliance issue, and only effects 64 bit versions of the core when plugged into 32 bit PCI buses. All other issues relate either to the Verilog source files (not provided with this patch), pin location files (not provided with this patch), are enhancements to the simulation environment, or are corrections to the full release structure.

Important Notes

Timing Driven layout - Clock to Out timing fails after layout

Occasionally the core fails to meet the PCI clock-to-output delays. The constraints provided in the user guide do not take into account that the enables for the I/O buffers enable the buffers and clock cycle early and therefore the path is non-critical. If the Timer shows the clock-to-out timing failing through the I/O pad ENABLE pin then disable the path. The turnoff time is also non-critical since the PCI bus allows a complete clock cycle for the bus turnaround cycle.

In the Timer GUI go to the Breaks tab and select Global Stops. In the Filter box type “*PAD*:E” and click Set, followed by Select All and then Add. This adds all the I/O pad enable pins to the breaks, allowing the Timer to ignore the timing through the I/O pad enable pins.

SAR24542: Core does not disconnect at Memory (BAR) Boundaries

The core maintains the previous non-compliant behavior and does not disconnect at memory boundaries. However, if the EN_BAR_OVERFLOW constant in ADD_PHASE.VHD (line 111) is changed to '1' the core becomes PCI complaint and disconnect at memory boundaries. When enabled the core may also disconnect at address locations within 4 dwords of the actual address boundary (this may lower the PCI bus throughput).

SAR24976: Testbench clocking may cause non-SDF netlist simulation to fail

If you perform gate-level simulations then you must back-annotate SDF timing onto the netlist or the simulation will fail (due to clock skew within the testbench).

SAR24964: 64 Bit Cores

The core does not support 32 bit operations when in 64-bit mode. If a 32-bit cycle is carried out then the core treats it as 64-bit operation, even though no data is provided on the upper 32 data bits. This implies that a 64-bit core should be used only in systems that are known to operate exclusively in 64-bit mode.

SAR28390: PCI testbench to have a resolution of 1ps. (LiberoSP2 Compatibility)

The testbench generates an internal clock that is delayed to match the delay on the clock buffer that is instantiated in the core. In the Verilog and Vital libraries before Libero2.3SP2 this delay was set to 1 ns for all supported families. In Libero 2.3SP2 this delay was changed to 0.1 ns for the SX-A, APA and AX families. The testbench is now set to 0.1 ns to match these families. If the SX or A500K families are being used then this delay needs to change back to 1 ns. This can easily be accomplished by changing the generic DELAY1 in system32.vhd or system64.vhd to TRUE.

SAR27825: RTSXS Implementations can violate hold times

When implemented in RTSXS devices the core may violate the PCI hold times (0 ns), in particular on the AD inputs. This can be resolved by inserting BUFD cells between the I/O pads and the AD registers either in the RTL code or gate level netlists. Alternatively, this can also be corrected by moving the registers away from the I/O pad post layout using the chip edit tool within Designer. Increasing the delay from PAD to register will remove the hold violations.

Note: To verify the external setup and hold times the Timer Report in Tools->Reports->Timer can be used within Designer; the reported hold times should be less than 0 ns.

Libero Support

The 5.32 release includes an additional release directory structured to allow the Core to be easily implemented using the Libero environment.

Setting up a Libero Project

1. **Create a new Libero project**
2. **In the Libero File Manager pane import all the files in the *$corepci532/Libero/HDL_pack* directory into the VHDL Packages Files section**
3. **In the Libero File Manager pane import all the files in the *$corepci532/Libero/hdl_xxx* directory into the HDL files section. "xxx" should be SX-A, APA, RTSX-S or AX depending on your target library**
4. **In the Libero File Manager pane import all the files in the *$corepci532/Libero/Stimulus* directory into the Stimulus Files section.**
5. **Using Windows Explorer copy the four files from *$corepci532/Libero/Simulation* directory to the *simulation* directory in the Libero project.**
6. **If necessary, edit the simulation scripts copied in the previous step.** Instructions are provided in the script files describing the simple edits that are required. The compile.do file is set up for 32 bit cores using the SX, SX-A, RTSXS & AX families. For 64 bit cores and other

FPGA families this file needs to be edited. The run.do script is set up for SX-A, APA, RTSXS, and AX families, for other families this needs to be edited.

7. **Alter the Libero Simulation Options in Tools->Options->Simulation, disable the Use Automatic Do File option.**

Running the Simulation

1. **In the Libero Design Hierarchy pane set the design root as required.** For TDMA32_WRP, you should see 16 possible design roots, the four variations of the core, 32 and 64 bit versions and additional versions that include the SDRAM controller. (For simulation it does not matter what the root is set to as long as it is set).
2. **In the Libero Design Hierarchy pane select the design root and right-click and select Run Pre-Synthesis Simulation. I**f a pop-up window appears select Start Modelsim without loading stimulus and click OK.
3. **When the Modelsim window appears type "do compile.do".** This compiles all the source files, no ERRORS or WARNING should occur.
4. **When the compilation finishes, type "do run.do".** This runs the complete simulation.

Running Synthesis

1. **In the Libero Design Hierarchy pane set the design root as required.** For TDMA32_WRP, you should see 16 possible design roots, the four variations of the core, 32 and 64 bit versions and additional versions that include the SDRAM controller.
2. **In the Libero Design Hierarchy pane select the design root and right-click and select Synthesize.** When Synplicity appears click on RUN.

Simplified Testbench

Included with the CorePCI5.32 release is a BETA version of a simplified testbench. This testbench is not intended to verify core compliance against the PCI specification but instead to provide an easy way for you to verify integration of the core into the end system. The source for this testbench is provided in the *$corePCI532/Libero/BetaSim* directory. To use this testbench, follow the steps below.

Set up a Libero Project

1. **Create a new Libero project**
2. **In the Libero File Manager pane import all the files in the *$corepci532/Libero/HDL_pack* directory into the VHDL Packages Files section.**
3. **In the Libero File Manager pane import all the files in the *$corepci532/Libero/hdl_xxx* directory into the HDL files section. "xxx" should be SX-A, APA, RTSX-S or AX depending on your target library. Remove all nine "*sdram*" files from the Libero project.**
4. **In the Libero File Manager pane import all the files from the *$corepci532/Libero/BetaStimulus* directory into the Stimulus Files section**
5. **Using Windows Explorer copy the three files from *$corepci532/Libero/BetaSimulation* directory to the simulation directory in the Libero project.**
6. **If necessary, edit the simulation scripts copied in the previous step.** The compile.do file is set up for 32 bit cores using the SX, SX-A, RTSX-S & AX families, for 64 bit cores and other FPGA families this file needs to be edited. The run.do script is set up for SX-A, APA, RTSX-S, and AX families, for other families this needs to be edited. Instructions are provided in the script files describing the simple edits that are required.
7. **Alter the Libero Simulation Options in Tools->Options->Simulation, disable the Use Automatic Do File option.**

Run the Simulation

1. **In the Libero Design Hierarchy pane set the design root as required. For TDMA32_WRP.** (For simulation it does not matter what the root is set to as long as it is set.)
2. **In the Libero Design Hierarchy pane select the design root and right click and select Run Pre-Synthesis Simulation.** If a pop-up window appears select Start Modelsim without loading stimulus and click on OK.
3. **When the Modelsim window appears type “do compile.do”.** This compiles all the source files, no ERRORS or WARNING should occur.
4. **When the compilation finishes, type “do run.do”.** This runs the complete simulation.

Updating from 5.21

The top-level generic types on the 5.3 and 5.31 core use INTEGER types, previous versions (5.21 and earlier) used std_logic. The instantiation of the core in your design needs to be updated. This change was made to make the code compatible with Synopsys Design Compiler.