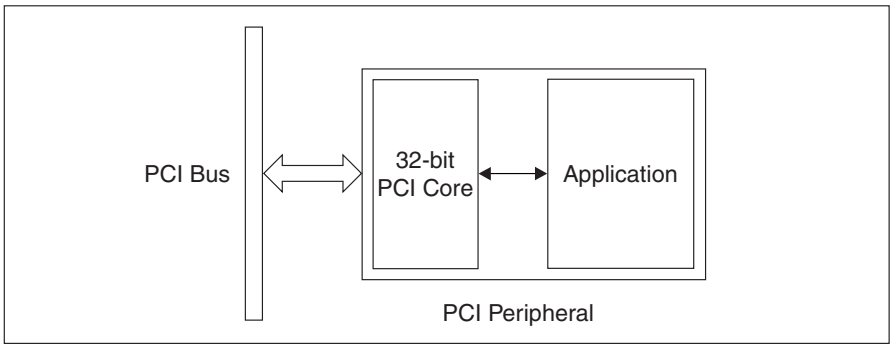
## Features

- **PCI 2.1 Compliant**
- **Supports 33/66 MHz Operation**
- **Master and Slave Support 32-bit Address and Data Transfers**
- **Supports Variable Burst Size Transfers**
- **Performs Zero Wait State Transfers**
- **Master Capable of Performing I/O, Memory and Configuration Types of Transfers**
- **Master Supports Byte Mode Operation**
- **Master Capable of Performing Memory Write Invalidate and Memory Read Line Operations**
- **Performs Back-to-back Transfers**
- **Fully Synchronous Design**
- **Approximately 12K Gates**
- **Slave Supports Up to Six Address Ranges**
- **Verilog-HDL Based Design**
- **Includes Comprehensive Test Environment (complete results listed in this document)**

## System Overview




## Overview

The ATPCI-SSP8000-32 is a fully synthesizable core that can be implemented in any Atmel ASIC library (gate array or standard cell). The core is supported by a comprehensive PCI test environment that can be used to verify the entire design, including the application. The PCI Core can be used in any application that requires a 32-bit master or slave core at either 33 or 66 MHz. The interface to the application consists of a master interface, a slave interface and a configuration interface.

# 32-bit PCI ASIC Core

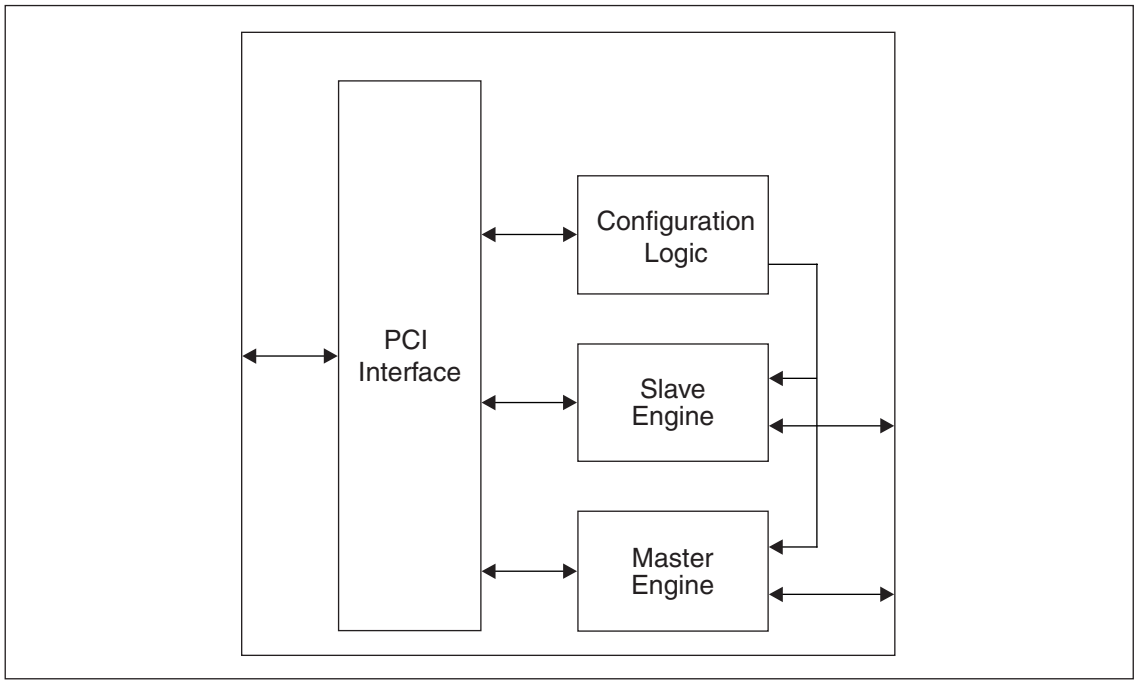
# ATPCI-SSP8000-32

# Summary

Rev. 1665AS–07/01

Note: This is a summary document. A complete document is available under NDA. For more information, please contact your local Atmel sales office.

## Core Block Diagram




### PCI Interface

This block communicates with the slave, configuration and master data paths and address paths to generate appropriate transfers on the PCI bus. Pad control, data multiplexing and parity generation and detection logic are performed in this block.

### Master Engine

This block handles master cycles, retries, command generation and data transfers. It also handles memory write invalidate (MWI) and memory read line (MRL) transfers. It generates handshake signals to communicate with the application.

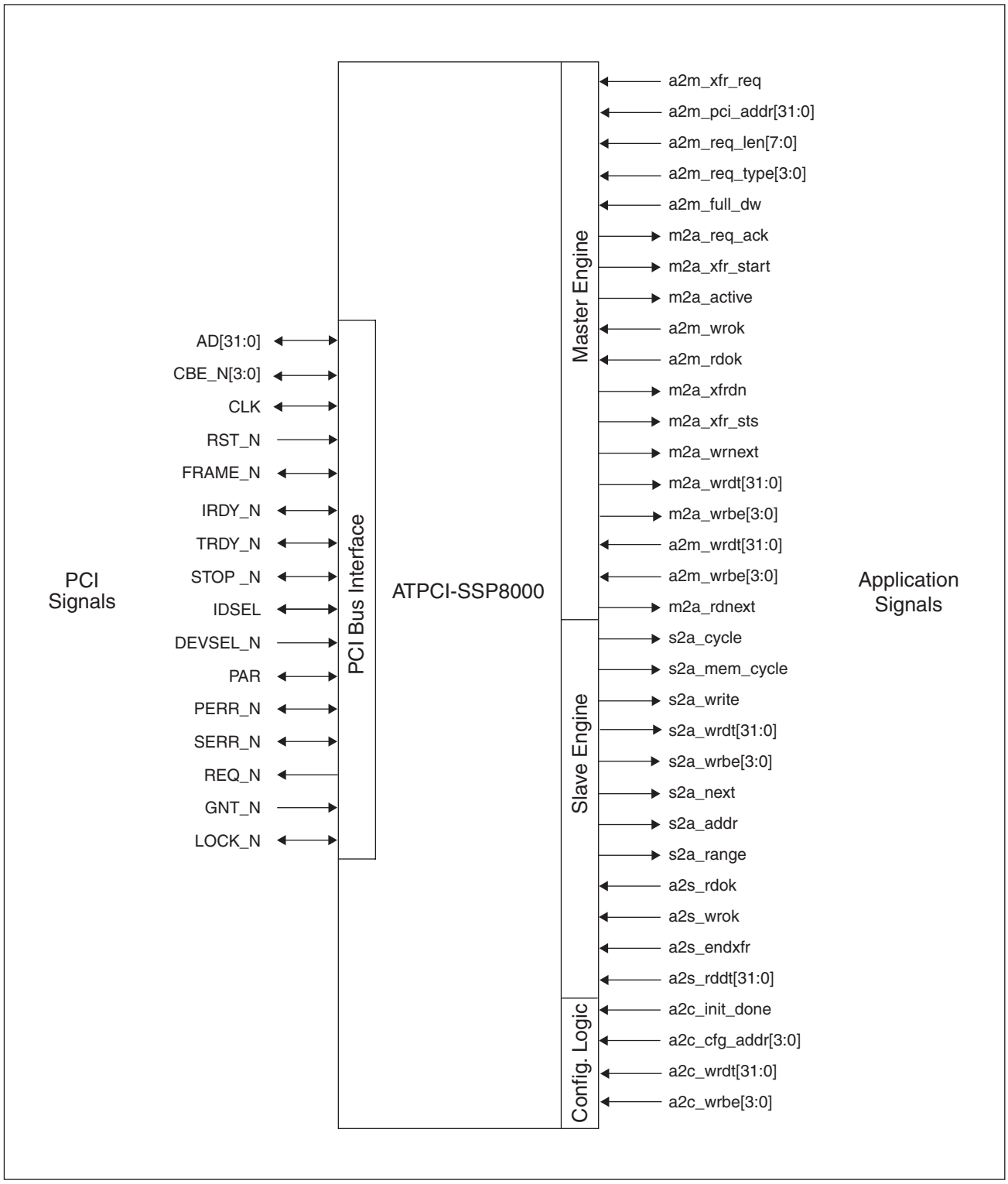
### Slave Engine

This block handles address decode, command decode and generation of slave PCI cycles. It is capable of performing burst transfers. Up to six address ranges can be configured.

### Configuration

This block has the configuration logic of the PCI Core. It is programmable to accommodate multiple base address registers. It can be addressed from both application and slave interfaces.

## Pin Diagram


PCI Signals
AD[31:0]
CBE_N[3:0]
CLK
RST_N
FRAME_N
IRDY_N
TRDY_N
STOP _N
IDSEL
DEVSEL_N
PAR
PERR_N
SERR_N
REQ_N
GNT_N
LOCK_N
PCI Bus Interface
ATPCI-SSP8000
Master Engine
a2m_xfr_req
a2m_pci_addr[31:0]
a2m_req_len[7:0]
a2m_req_type[3:0]
a2m_full_dw
m2a_req_ack
m2a_xfr_start
m2a_active
a2m_wrok
a2m_rdok
m2a_xfrdn
m2a_xfr_sts
m2a_wrnext
m2a_wrdt[31:0]
m2a_wrbe[3:0]
a2m_wrdt[31:0]
a2m_wrbe[3:0]
m2a_rdnext
Slave Engine
s2a_cycle
s2a_mem_cycle
s2a_write
s2a_wrdt[31:0]
s2a_wrbe[3:0]
s2a_next
s2a_addr
s2a_range
a2s_rdok
a2s_wrok
a2s_endxfr
a2s_rddt[31:0]
Config. Logic
a2c_init_done
a2c_cfg_addr[3:0]
a2c_wrdt[31:0]
a2c_wrbe[3:0]
Application Signals

## PCI – Bus Signals

**Table 1.** PCI Interface

| Pin Name | Direction | Description |
|---|---|---|
| AD [31:0] | I/O | PCI address and data are multiplexed on the same pins. A bus transaction consists of an address phase followed by one or more data phases. During data phase, AD[7:0] contains the least significant byte, and AD[31:28] contains the most significant byte. These are tristate signals. |
| CBE_N[3:0] | I/O | Command and byte enables are multiplexed on the same pins. Commands during the address phase of the PCI transfer and byte enables during the data phase of the transfer are provided on this bus. These are bidirectional, tristate signals. |
| CLK | Input | This is the PCI CLK. This determines the clock frequency of the PCI system. A 30 ns period is used for 33 MHz frequency. All other PCI signals except RST_N, INTA_N, INTB_N, INTC_N and INTD_N are sampled on the rising edge of CLK. |
| RST_N | Input | This is an active-low signal to indicate a reset condition on the PCI system. All PCI-specific registers, sequencers and signals are reset. |
| FRAME_N | I/O | This is an active-low signal indicating the start of a master transfer. FRAME continues to be asserted during the duration of the master cycle. When frame is deasserted, the transaction is in the final data phase or has completed. |
| IRDY_N | I/O | InitiatorReady indicates the initiating agent's (bus master's) ability to complete the current data phase of the transaction. A data phase is completed on any clock when both IRDY_N and TRDY_N are asserted. |
| TRDY_N | I/O | TargetReady indicates the target device's ability to complete the current data phase of the transaction. A data phase is completed on any clock when both IRDY_N and TRDY_N are asserted. |
| STOP_N | I/O | This signal is an active-low signal driven by the current target. It indicates to the requesting master to terminate the current transaction. |
| IDSEL | Input | InitializationDeviceSelect is used as chip select during configuration read and write transactions. |
| DEVSEL_N | I/O | This active-low signal is asserted when the target has decoded the current address on the bus and has found a match for that address. As input, DEVSEL_N indicates whether any device on the bus has been selected. |
| PAR | I/O | This active-high signal is driven by the device that has driven the address bus in the previous cycle. It indicates the even parity of address and command bits for an address phase or even parity of data and byte enables for a data phase. Parity is always valid for the transfer that occurred the previous clock cycle. |
| PERR_N | I/O | ParityError is for reporting of data parity errors during all PCI transactions except a special cycle. This is a sustained tristate signal. |
| SERR_N | I/O | This is an active-low signal for reporting address parity errors and data parity errors on the special cycle command or any other system error where the result will be catastrophic. |
| REQ_N | Output | This is an active-low output from the PCI Core to indicate request for ownership for the PCI bus. |
| GNT_N | Input | This is an active-low input signal to the PCI Core to indicate grant of ownership of the PCI bus after the completion of the current transfer on the bus. |
| LOCK_N | I/O | LOCK is an active-low signal indicating an atomic operation that may require multiple transactions to complete. When LOCK_N is asserted, non-exclusive transactions may proceed to an address that is not currently locked. The PCI Core does not support the lock feature. |

**Table 2.** Application/Master Interface Signals

| Signal Name | Direction | Description |
|---|---|---|
| a2m_xfr_req | Input | An active-high signal asserted by the master to indicate a new transfer requested by the application. This is an input to the PCI Core. |
| a2m_pci_addr[31:0] | Input | The PCI address of the next request is provided on the bus. This is valid before the a2m_xfr_req to indicate the PCI address to which the data transfer needs to be performed. This is an input to the PCI Core. |
| a2m_req_len[7:0] | Input | This is valid along with a2m_xfr_req indicating the number of double words to be transferred for the request. This is an input to the PCI Core. |
| a2m_req_type[3:0] | Input | It indicates the type of transfer that needs to be performed on the bus for the request. It indicates whether it is a configuration, I/O or memory transfer. It also indicates whether it is a read or write transfer. |
| a2m_full_dw | Input | This is asserted by the application to indicate whether all the bytes in the current request are valid. This signal enables the PCI Core to determine whether it can perform a MWI transfer. The PCI Core does not rely on the byte enables once it has initiated a MWI. It is necessary that a master drive all its byte enables to “0” once it has initiated a MWI. This is an input to the PCI Core. |
| m2a_req_ack | Output | It indicates that the current request from the application has been accepted. Upon receipt of acknowledge, the application may choose to post another request to the PCI Core. |
| m2a_xfr_start | Output | It indicates the start of a new transfer by the master for the previous application request. |
| m2a_active | Output | It indicates that the PCI Core is the current master on the bus. It is deasserted once the master transfer is completed. |
| a2m_wrok | Input | It indicates that the application is ready to accept data into its FIFO. Deassertion of this signal indicates that the PCI Core needs to insert a wait state on the bus for the next data transfer. |
| a2m_rdok | Input | It indicates that the application has data in its FIFO. Deassertion of this signal indicates that the PCI Core needs to insert a wait state on the bus for the next data transfer. |
| m2a_xfrdn | Output | This signal is asserted by the PCI Core upon completion of the transfer on the bus. |
| m2a_xfr_sts | Output | This signal is asserted by the PCI Core along with m2a_xfrdn indicating the status of the last master transfer. |
| m2a_wrnext | Output | When the PCI Core is in the master mode, the data from the current PCI master read transaction is written into the application. This is asserted by the PCI Core to indicate that the application should advance its data pointer to the next location. |
| m2a_wrdt[31:0] | Output | The data from the PCI bus is written into the application through the bus in the master mode. |
| m2a_wrbe[3:0] | Output | This indicates the byte enables for the current transfer. |
| a2m_wrdt[31:0] | Input | This is the data provided by the application. This data is latched into the PCI Core staging register with every `m2a_rdnext` assertion. During PCI master write transactions, this data is transmitted on the bus. |
| a2m_wrbe[3:0] | Input | The byte enables to a PCI write transaction in the master mode are provided by the application through this bus. This byte enable is provided by the application for every data transfer. |
| m2a_rdnext | Output | This indicates that the PCI Core has latched the data from the application for the current PCI master write transaction and the application needs to increment its data pointer to point to the next data. |

**Table 3.** Application/Slave Interface Signals

| Signal Name | Direction | Description |
|---|---|---|
| s2a_cycle | Output | This indicates that the PCI Core has detected a valid slave cycle. |
| s2a_mem_cycle | Output | This signal indicates that the current cycle is a memory cycle when “1” and I/O cycle when “0”. This signal is valid when s2a_cycle is asserted. |
| s2a_write | Output | This signal indicates that the current cycle is a write cycle when “1” and read cycle when “0”. This signal is valid when s2a_cycle is asserted. |
| s2a_wrdt [31:0] | Output | This is the data that is transferred on the PCI bus during the current PCI slave write cycle. The PCI Core asserts TRDY_N for further PCI slave write transfers based on s2a_wrok. If s2a_wrok is deasserted, then it inserts wait states on the bus. If the s2a_wrok is deasserted for more than eight clocks, the PCI Core automatically performs a disconnect on the bus. |
| s2a_wrbe [3:0] | Output | The byte enables that were transferred on the PCI bus for the current PCI slave write cycle to the PCI Core are provided on this bus. |
| s2a_next | Output | The PCI Core in the slave mode will assert this signal to the application to advance to the next location in the case of a read or write transaction. |
| s2a_addr | Output | This indicates the current address from which data is needed in the case of a slave read cycle and the location to which data is written in the case of a slave write cycle. This address is incremented by the PCI Core every time a data transfer occurs on the bus. The signal s2a_range indicates the valid bits for that range of decoded addresses. |
| s2a_range[2:0] | Output | This indicates the range of addresses that were decoded for the current slave cycle. This is an output from the PCI Core. The PCI Core may have up to six address ranges as a slave. This indicates to which of the decoded addresses the current slave cycle belongs. |
| a2s_rdok | Input | This is an input to the PCI Core. This signal indicates whether the application data to slave is valid. If this signal is deasserted, the PCI Core inserts wait states on the TRDY_N signal during a PCI slave read cycle. |
| a2s_wrok | Input | This is an input to the PCI Core. This signal indicates that the application is ready to accept data from slave. Deassertion of a2s_wrok forces the PCI Core to insert wait states on the PCI bus for a PCI slave write cycle. |
| a2s_endxfr | Input | This is an input to the PCI Core. This signal indicates that the application intends to end the current slave transaction and results in a disconnect. |
| a2s_rddt[31:0] | Input | The data to be read by the PCI Core in the slave read mode is provided on this bus. The signal s2a_addr addresses the location in the application to read the data. |

**Table 4.** Application/Configuration Interface Signals

| Signal Name | Direction | Description |
|---|---|---|
| a2c_init_done | Input | This must be deasserted upon reset. It is asserted by the application after initializing the configuration space. This is an input to the PCI Core. In devices where there is no initialization required, it should be tied high. During initialization, the application can write to the configuration registers. The address is provided on the a2c_cfg_addr bus and the configuration data is provided on the a2c_wrdt bus. The byte enables for the data are on the a2c_wrbe bus. |
| a2c_cfg_addr[3:0] | Input | The application addresses the configuration space through this bus. |
| a2c_wrdt[31:0] | Input | The configuration data from the application is written through this bus. The a2c_cfg_addr bus addresses the configuration registers. |
| a2c_wrbe[3:0] | Input | The byte enables for the configuration data from the application are provided on this bus. |

## Atmel Headquarters

*Corporate Headquarters*
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

*Europe*
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

*Asia*
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

*Japan*
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

## Atmel Operations

*Atmel Colorado Springs*
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

*Atmel Rousset*
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

*Atmel Smart Card ICs*
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

*Atmel Grenoble*
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

*Fax-on-Demand*
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

*e-mail*
literature@atmel.com

*Web Site*
http://www.atmel.com

*BBS*
1-(408) 436-4309



1665AS–07/01