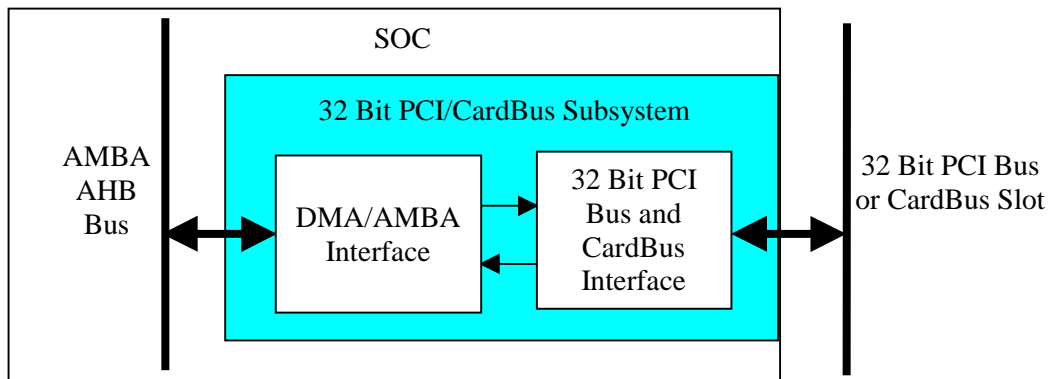


## **AU-PB7032: 32 Bit PCI/CardBus AMBA Subsystem Core** **AMBA AHB Bus 32 Bit PCI/CardBus Interface with DMA**

The AU-PB7032 32 Bit PCI/CardBus AMBA Subsystem provides a 32 Bit PCI Bus and CardBus peripheral subsystem for AMBA based SOCs. It contains a 32 Bit PCI Bus and CardBus Interface with PCI/CardBus master and slave functionality. The 32 bit PCI/CardBus AMBA Subsystem connects seamlessly to the AMBA AHB Bus. A DMA Engine is included to move data of PCI/CardBus master memory and I/O transactions. PCI/CardBus master configuration transaction data is transferred by direct AMBA Bus reads and writes. PCI/CardBus slave transactions are translated into AMBA Bus transactions by the 32 Bit PCI/CardBus AMBA Subsystem and driven onto the AMBA Bus. The figure below shows the 32 Bit PCI/CardBus AMBA Subsystem within an SOC. The 32 Bit PCI/CardBus AMBA Subsystem Core is available as a synthesizable Verilog model from Aurora VLSI, Inc. Contact [CustomerService@auroravlsi.com](mailto:CustomerService@auroravlsi.com).



PCI Bus speeds of 33MHz and 66MHz are supported. The CardBus speed is 33MHz as described by the "CardBus Specification- PC Card Standard Release 8.0". The 32 Bit PCI/CardBus Interface block of the PCI/CardBus Subsystem provides 32 bit PCI/CardBus master and slave functionality. An AMBA Bus master can initiate a PCI/CardBus transaction through this PCI/CardBus Interface block in master mode. In PCI/CardBus slave mode, the PCI/CardBus data is written to or read from an AMBA Bus slave. The PCI/CardBus Interface block contains the standard PCI/CardBus configuration registers including four base address registers.

Internal to an SOC, the PCI/CardBus Subsystem is a peripheral on the AMBA AHB Bus. Direct Memory Access- DMA, between the DMA/AMBA Interface block of the PCI/CardBus Subsystem, and AMBA Bus targets, is used to transfer data of PCI/CardBus master memory and I/O transactions over the AMBA Bus to/from the PCI/CardBus Subsystem. When doing DMA, the PCI/CardBus Subsystem is an AMBA Bus master.

When the PCI/CardBus Subsystem is functioning as a PCI/CardBus slave, a PCI/CardBus target address is provided by the PCI/CardBus transaction. This address is mapped into an AMBA Bus address by the PCI/CardBus Subsystem. PCI/CardBus data is driven onto the AMBA Bus for PCI/CardBus slave writes, and read from the AMBA Bus for PCI/CardBus slave reads. When doing PCI/CardBus slave transactions, the PCI/CardBus Subsystem is an AMBA Bus master.

The PCI/CardBus Subsystem is an AMBA Bus slave for control and status register accesses, and for PCI/CardBus master configuration transactions. All control and status register accesses and PCI/CardBus master configuration transactions originate in an AMBA Bus master outside of the PCI/CardBus AMBA Subsystem, such as a host processor.

32 Bit PCI/CardBus AMBA Subsystem features are summarized:

#### 32 Bit PCI/CardBus Interface

- PCI 2.1 and 2.2 compliant
- CardBus compliant
- Master and slave support 32 bit address and data transfers
- Supports variable burst size transfers
- Performs zero wait state transfers
- Master is capable of performing I/O, Memory, and Configuration types of PCI/CardBus transfers
- Master supports byte mode operation
- Master can perform Memory Write Invalidate and Memory Read Line operations
- Performs back to back transfers
- PCI/CardBus master transaction status captured in eight entry PCI/CardBus Master Status FIFO
- Includes CardBus STSCHG signal
- CardBus power management states D0, D1, D2, and D3 supported

#### DMA/AMBA Interface

- AMBA AHB Bus interface
- Single channel DMA Engine for PCI/CardBus master memory and I/O transactions
- Physical DMA addresses
- Programmable DMA starting address
- Programmable DMA transfer count- up to 64 Kbytes
- Programmable DMA AMBA Bus interface transaction size- 8 to 1024 bytes
- Programmable DMA AMBA Bus data transfer size- 4 or 8 bytes
- Locked DMA operation optional (software programmable)
- Direct software writes or information extracted from descriptors in memory, to program DMA control information
- Dedicated AMBA Bus master interface for the DMA channel
- Dedicated AMBA Bus master interface for PCI/CardBus slave transactions
- Unique PCI/CardBus slave address to AMBA Bus address mapping for each PCI/CardBus slave address space
- AMBA Bus slave interface for register reads and writes, and PCI/CardBus master configuration transactions
- Interrupts:
  - PCI/CardBus master DMA completed
  - Software interrupt
  - PCI/CardBus reset interrupt
  - CardBus soft reset interrupt
  - CardBus power state change interrupt
  - CardBus stopped/slowed clock interrupt
- PCI/CardBus interrupt INTA

The core is delivered as a synthesizable RTL Verilog model. Deliverables include:

- RTL Verilog source code model of the core
- Verilog testbench and test cases
- Synthesis scripts examples
- Complete detailed documentation and training class notes

### **32 Bit PCI/CardBus Interface**

The PCI/CardBus Subsystem includes the AU-P7032 32 Bit PCI/CardBus Interface Core. Additional logic at the application interface of the 32 Bit PCI/CardBus Interface provides a DMA Engine, control and status registers, the AMBA Bus interface, and host processor interrupts for the 32 Bit PCI/CardBus Subsystem.

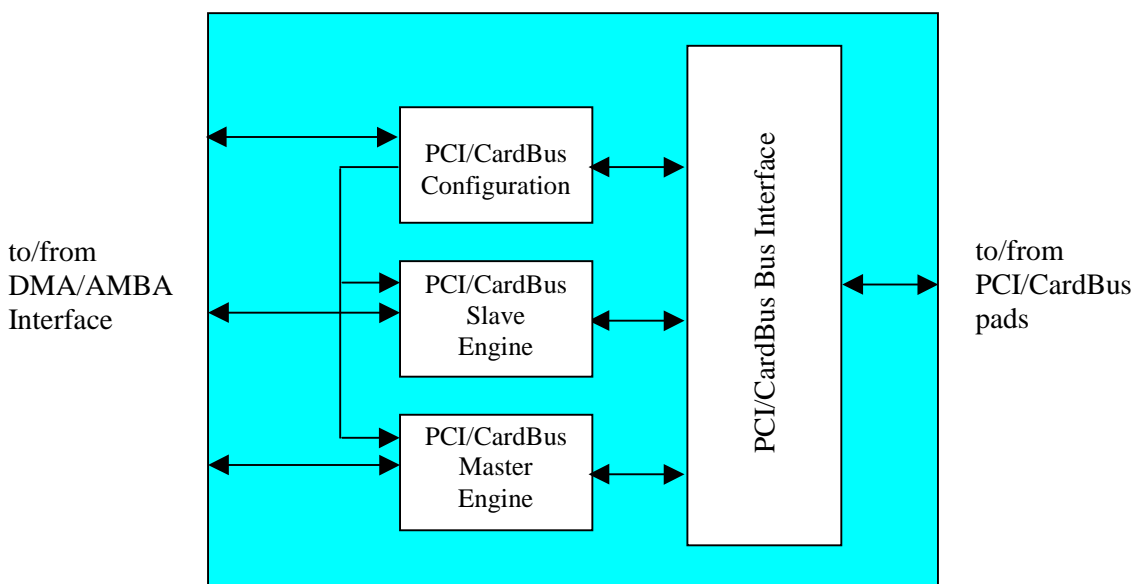
A block diagram of the PCI/CardBus Interface is shown below.

The PCI/CardBus Master Engine 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 DMA/AMBA Interface block.

The PCI/CardBus Slave Engine handles address decode, command decode, and generation of slave PCI/CardBus cycles. It is capable of performing burst transfers.

The PCI/CardBus Configuration block has the configuration address space of the PCI/CardBus Interface. It is programmable to accommodate multiple base address registers. It is accessed from the DMA/AMBA Interface block and by PCI/CardBus slave transactions from the PCI/CardBus Interface block.

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



## **DMA/AMBA Interface**

The DMA/AMBA Interface includes a single channel DMA Engine. The DMA Engine is used to transfer PCI/CardBus master memory and I/O write data from the data's source, over the AMBA Bus, to the PCI/CardBus Master Engine in the PCI/CardBus Interface block. PCI/CardBus master memory and I/O read data is sent from the PCI/CardBus Master Engine, over the AMBA Bus, to the received data's destination, by the DMA Engine.

Block moves of up to 64K bytes are supported by the DMA Engine. The exact transfer count of each DMA operation is the size of the data block that is being transferred, and is set by software. DMA operations are done as a series of data transfers to/from the PCI/CardBus Master Engine, and AMBA Bus transactions to move the data block. The length of each AMBA Bus transaction is software programmable so that it can be optimized according to system characteristics. Each individual AMBA Bus transaction is from 8 bytes to 1024 bytes according to a value programmed into a DMA channel's control register. Additionally, the data size of each data transfer on the AMBA Bus is software programmable to be four or eight bytes.

The series of accesses that make up a complete DMA operation may be locked together so that no other device gets the AMBA Bus until the DMA operation is finished. This is under software control.

The DMA starting address is set by software. This is the write data source starting address in memory for write data, and the read data destination starting address in memory for read data. These starting addresses are incremented by the DMA Engine to form the AMBA Bus transaction addresses as the DMA operation progresses. All addresses are physical addresses.

The DMA control information that is set by software- starting address, transfer count, AMBA Bus transaction size, data transfer size, and lock flag, can be set by direct software writes to PCI/CardBus Subsystem registers that hold this DMA control information. Alternatively, this DMA control information can be set from descriptors in memory that hold the DMA control information. Scatter/gather DMA is done using a chained descriptor list as the DMA control information source. When using descriptors to set the DMA control information, the descriptors are initialized by software. To support DMA configuration from descriptors, the DMA/AMBA Interface contains logic to read the descriptors from memory, and load the appropriate DMA/AMBA Interface registers with the DMA control information.

A DMA operation begins when the DMA channel is enabled, after the starting address, transfer count, AMBA Bus transaction size, data transfer size, and lock flag are configured. The DMA channel moves the data block, and the DMA operation ends when the entire data block has been moved.

The PCI/CardBus Subsystem can also function as a PCI/CardBus master in direct access (non-DMA) mode. To do a direct access PCI/CardBus master write, the PCI/CardBus address, byte write enables, and write data are written to registers within the PCI/CardBus Subsystem. A direct access PCI/CardBus master read is done by writing the PCI/CardBus address to a PCI/CardBus Subsystem register, and then reading the PCI/CardBus read data and byte read enables from PCI/CardBus Subsystem registers. Optionally, the direct access address register can be automatically incremented upon completion of each PCI/CardBus data transfer. There are six sets of PCI/CardBus master direct access registers; one set for each PCI/CardBus master bus transaction type/direction- PCI/CardBus memory read, memory write, I/O read, I/O write,

configuration read, and configuration write. PCI/CardBus master direct access mode can be used to provide a bus master AMBA/PCI/CardBus bridge.

Status is captured for each PCI/CardBus master transaction. It is held in an eight entry Master Status FIFO. The host processor may read this FIFO or optionally in DMA mode, the PCI/CardBus Subsystem automatically reports the status by sending it out over the AMBA Bus.

As a PCI/CardBus slave, the PCI/CardBus Subsystem accepts PCI/CardBus transactions, translates the PCI/CardBus address into an AMBA Bus address, and generates the appropriate read or write AMBA Bus transaction. On PCI/CardBus slave reads, when the read data is returned to the PCI/CardBus Subsystem over the AMBA Bus, the PCI/CardBus Subsystem provides it to the PCI/CardBus Interface block that then drives the read data onto the PCI/CardBus. The PCI/CardBus Subsystem supports four PCI/CardBus slave address spaces. Each PCI/CardBus slave address space has a unique software programmable AMBA Bus base address.

The PCI/CardBus Subsystem contains several registers. These registers are accessed using AMBA Bus single transactions initiated by host processor reads and writes. In addition to the AU-P7032 registers, there are registers that hold:

- General control and status information
- PCI/CardBus slave mode control and status information
- PCI/CardBus slave mode AMBA Bus base addresses
- PCI/CardBus master DMA addresses, control, and status information
- PCI/CardBus master direct access control and status information
- PCI/CardBus master direct access addresses, data, and byte enables

The DMA/AMBA Interface generates interrupts to notify the host processor of events that are important to driver software. These interrupts include:

- Interrupt upon a completed PCI/CardBus master DMA operation or chain of PCI/CardBus master DMA operations
- Software interrupt
- Interrupt upon a PCI/CardBus reset
- Interrupt upon a CardBus soft reset
- Interrupt upon a CardBus power state change
- Interrupt upon the CardBus clock stopping or slowing

The PCI/CardBus Subsystem can assert the PCI/CardBus interrupt INTA through software.