



EP440 PCI-to-PCI Bridge

Product Summary

FEATURES

- Fully supports PCI bus specification 2.2 and PCI bridge specification 1.1.
- Designed for ASIC and PLD implementations.
- Fully static design with edge triggered flip-flops.
- Independent asynchronous PCI clocks on primary and secondary bus.
- Convert bus transactions between primary bus and secondary bus.
- Combined bus master and target functions on both primary bus and secondary bus.

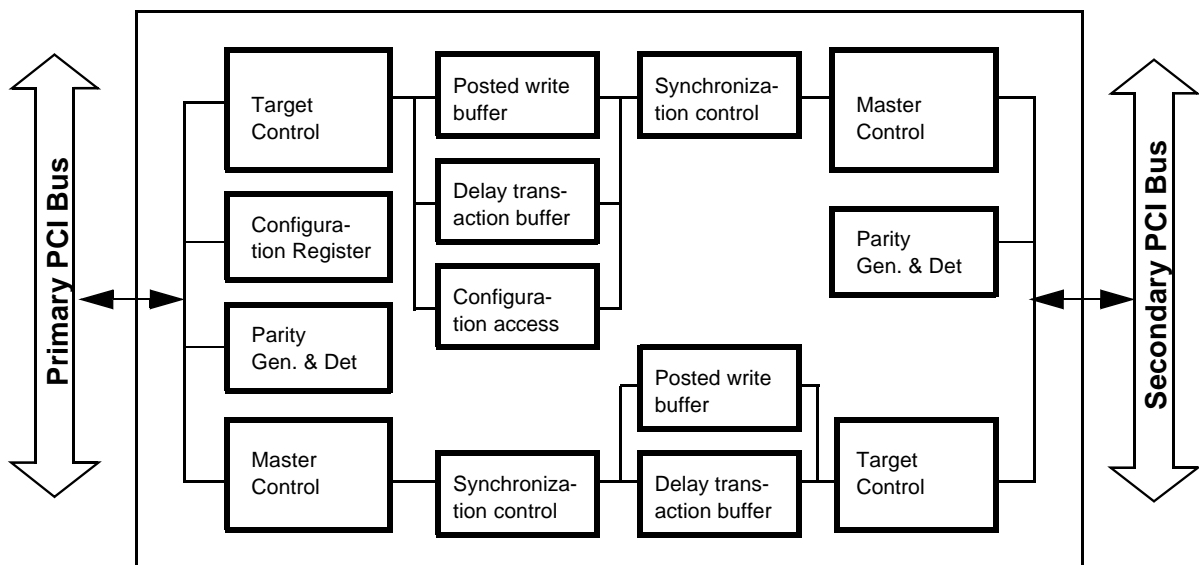
Master function

- Initiate PCI memory and IO read/write.
- Automatic transfer restart on target retry and disconnect.
- Initiate type 0 and type 1 configuration access on secondary bus.

Target function

- Memory or IO read/write.
- Receives type 0 and type 1 configuration access on primary bus.
- Posted memory write transaction and delay transaction on all other transaction types.

- Dual write buffer on each direction supports posted memory write.
- Supports prefetchable and non-prefetchable memory read.
- Delay transaction processes IO read/write, configuration read/write and memory read transactions.
- Supports target retry, disconnect, master abort and target abort terminations.
- Parity generation and parity error detection.
- Includes all PCI-PCI bridge specific configuration registers.
- Supports high speed bus request and bus parking.
- Optional PCI bus arbiter with fix, rotating, and custom priority.





EP440 PCI-to-PCI Bridge

DESCRIPTION

The 32-bit PCI-to-PCI bridge is designed for interfacing between the primary PCI bus and the secondary PCI buses. The PCI bridge consists of bus master, bus target and target functions on the primary PCI bus. It consists of bus master, bus target, and configuration initiation functions on the secondary bus.

The PCI bridge core receives configuration initialization from the host CPU during power-up reset. It also allows the host to configure all the secondary bus and PCI devices reside behind the bridge. Type 0 PCI configuration access, if address to the bridge, is processed and responded by the PCI bridge locally. Type 1 PCI configuration, if address to buses and devices located behind the bridge, is received by the PCI bridge and forwarded to the secondary bus. If the configuration target resides directly on the secondary bus, the PCI bridge converts the type 1 configuration access to type 0 access so that it can be recognized by PCI devices. It preserves the type 1 access if the target is another bridge device on the secondary bus.

The PCI bridge has dual write buffer on each bus interface to post memory write. The all memory write and write invalidate data are posted in the write buffer. The transaction is first completed in the originating bus by the PCI bridge as a target to the transaction. It then writes to the destination bus. The dual write buffer allows the originating bus to post a second write request to the bridge while the first write request is being processed.

The bridge function as a bus master on the destination bus. All different types of transfer termination are handled by the core. If a transfer is retried or disconnected by the target in the destination bus, the bridge re-starts the transfer automatically until all posted data are written. Bus request, bus parking, parity detection and generation all are handled by the core.

All other transaction such as memory read are processed by the PCI bridge as delay transaction. When the bridge becomes the target of such transaction, it terminates the transfer by using target retry. It then starts the transaction on the destination bus. When the read data returns from the destination bus, it stores it in its local read buffer. Until the read data is available, all the subsequent access of the same type to the same destination are terminated with retry. After the read data becomes available, subsequent read from the originating bus for the same address will be completed normally with data from the read buffer.

The bridge distinguish between prefetchable read data and non-prefetchable read data. Prefetchable data read are read one cache line at a time unless it is a single read. Non-prefetchable data are read one data at a time.

Other than read accesses, IO write and configuration write are handled as delay transaction. They are similar to read access except that data flows in the same



EP440 PCI-to-PCI Bridge

direction as the request instead of returning from the destination. The PCI bridge terminates a delayed write normally on the originating bus after it is completed on the destination bus.

The PCI bridge processes one delay transaction at a time. It accepts a second delay transaction only after a previous one is completed on the originating bus.

The PCI bridge preserves the order of posted memory write as defined by the PCI bridge specification 1.1. It also follows the ordering of delayed transaction as required by the specification. It allows posted memory write to pass all delayed transaction while delayed transactions are not allowed to pass posted memory write. This requirement is essential in avoiding deadlock in the bridge between the two buses.

Execution order in both direction of the bridge are general independent of each other. The PCI bridge follows the requirement on the PCI bridge specification 1.1 which requires that before a read transaction can be completed on its originating bus, it must pull out of the bridge any posted write that originated on the opposite side and were posted before the read command completes on the read-destination bus.

OPTIONAL FEATURES

The following table summarizes the optional features which are provided with the PCI bridge as required by user application.

Options	Description
Back-end bus size	32-bit or 64-bit user logic (CPU) interface
Uni-directional PCI bridge	Forward request only from primary bus to secondary bus but not in the opposite direction.
Bus arbiter	Arbitration for the primary and/or secondary PCI bus.
Base address registers	Supports multiple base address registers in the primary bus interface to allow other on-chip register to be mapped to the bridge.
Buffer size	Different buffer size as required by the user