pci_b PCI Master/Target MegaCore Function

September 1998, ver. 1 Data Sheet

Features...

- Parameterized `pci_b` MegaCore™ function implementing a 32-bit, 33-MHz peripheral component interconnect (PCI) master/target interface
- Fully compliant with the PCI Special Interest Group's (SIG) ***PCI Local Bus Specification, Revision 2.1*** timing and functional requirements
- Vigorously hardware tested using the following hardware and software (see "Verification Summary" on page 5):
  - FLEX® 10K PCI prototype board
  - HP E2925A PCI Bus Exerciser and Analyzer
  - Commonly used Intel host/PCI bridges and DEC PCI/PCI bridges
- Provides easy integration with customer-defined logic
- Includes sample test vectors for user simulation
- Wide range of density and package support; optimized for FLEX 10K architectures
- No-risk OpenCore™ feature allows designers to instantiate, compile, and simulate designs with the MAX+PLUS® II software prior to licensing
- Uses approximately 1,050 FLEX logic elements (LEs)
- Independent master and target operation
- PCI master features:
  - Infinite cycles of zero-wait-state memory read/write transfers (up to 132 Mbytes per second)
  - Initiates most PCI bus commands, including memory read/write, configuration read/write, I/O read/write, memory read multiple (MRM), memory read line (MRL), and memory write and invalidate (MWI)
  - Self configuration feature, allowing master to configure targets (including the `pci_b` function's own target) for host bridge applications
  - Parity error detection
  - Bus parking
- PCI target features:
  - Infinite cycles of zero-wait-state memory read/write transfers (up to 132 Mbytes per second)
  - Type zero configuration space
  - Up to 6 base address registers (BARs) with adjustable memory types and sizes, which achieve adaptability, efficient silicon implementation, and flexible system memory allocation

…and More Features

- Responds to most PCI bus commands, including configuration read/write, memory read/write, I/O read/write, MRM, MRL, and MWI; all other commands are ignored successfully, as required by the PCI specification
- Parity error detection

- Local-initiated target abort, retry, or disconnect
- Configuration registers:
  - Parameterized registers: device ID, vendor ID, class code, revision ID, BAR0 through BAR5, subsystem ID, subsystem vendor ID, maximum latency, and minimum grant
  - Non-parameterized registers: command, status, header type, latency timer, cache line size, interrupt pin, and interrupt line

General Description

Traditionally, PCI local bus applications have been targeted for low- to high-end desktop PCs. Today, the PCI interface is a common, fundamental building block for servers, LAN, SCSI, FDDI, and other high-bandwidth I/O applications. The `pci_b` function is a hardware-tested, high-performance, flexible implementation of the 32-bit, 33-MHz PCI master/target interface (ordering code: PLSM-PCI/B).

Because the `pci_b` function handles the complex PCI protocol and stringent timing requirements internally, designers can focus their engineering efforts on value-added custom development, significantly reducing time to market.

Optimized for Altera® FLEX 10K devices, the `pci_b` function supports configuration, I/O, and memory transactions. With the high density of FLEX devices, designers have ample resources for custom local logic after implementing the PCI interface. The high performance of FLEX devices also enables the `pci_b` function to support unlimited cycles of zero-wait-state memory-burst transactions, achieving 132 Mbytes per second peak and sustained throughput, which is the theoretical maximum for a 32-bit, 33-MHz PCI bus.

In the `pci_b` function, the master and target interfaces can operate independently, providing minimum latency and efficient use of the PCI bus. For instance, while the target interface is accepting zero-wait-state burst write data, the local logic can simultaneously request PCI bus mastership, which minimizes delay. In addition, the `pci_b` function's separate local master and target data paths allow independent data pre-fetching and posting. Depending on the application, different memory and data buffers, such as first-in first-out (FIFO) buffers, in various length, depth, and type can be implemented in the local logic.

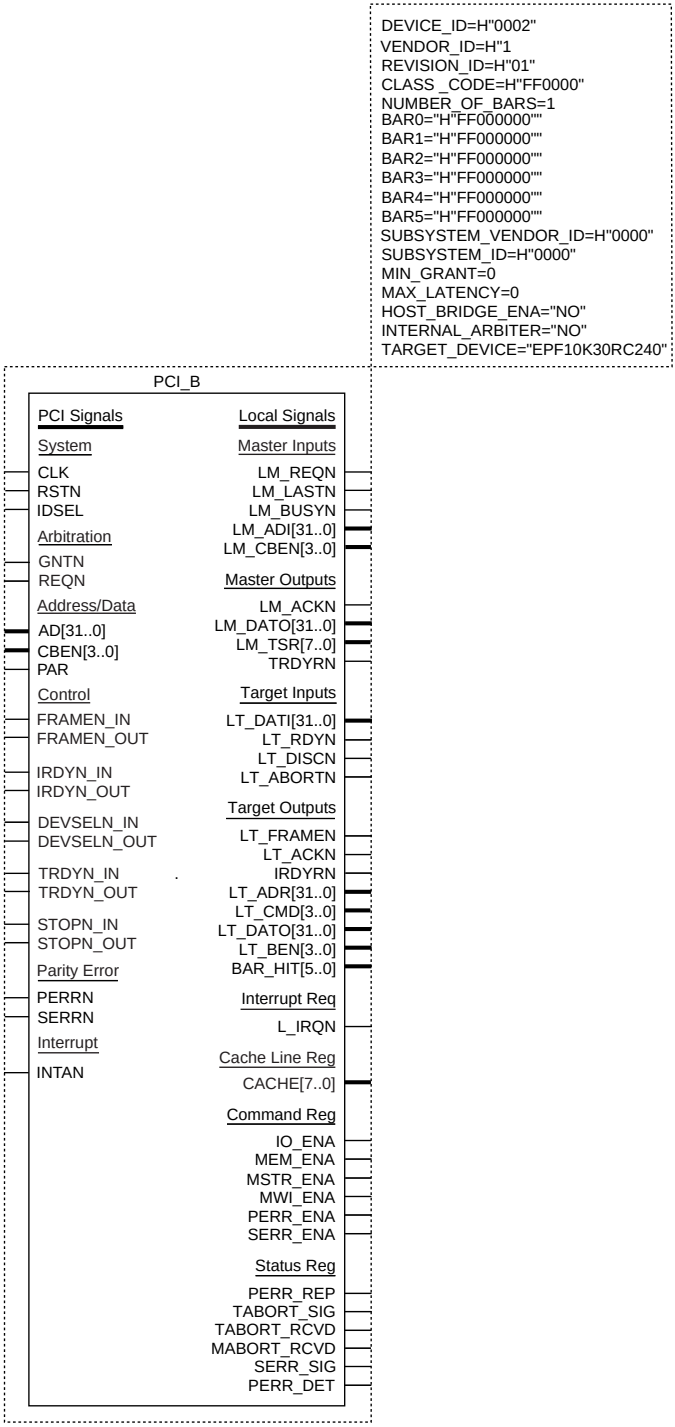
To ensure timing and protocol compliance, the `pci_b` function has been vigorously hardware-tested. See "Verification Summary" on page 5 for more information on the verification tests performed.

As a parameterized function, the pci_b function has configuration registers that can be modified upon instantiation. These features provide scalability, adaptability, and efficient silicon implementation. As a result, the same pci_b function can be used in multiple PCI projects with different requirements.

For example, the pci_b function offers up to six base address registers (BARs) for multiple local-side devices. However, some applications require only one contiguous memory range. PCI designers can choose to instantiate only one BAR, which reduces LE consumption. After designers define the parameter values, the MAX+PLUS II software automatically and efficiently implements the requested logic during compilation.

Figure 1 illustrates the pci_b symbol as used in a MAX+PLUS II Graphic Design File (**.gdf**).

Figure 1. pci_b Symbol



Verification Summary

The pci_b function is compliant with the requirements specified in the PCI SIG's ***PCI Local Bus Specification, Revision 2.1*** and ***PCI Compliance Checklist, Revision 2.1***. The pci_b function is shipped with MAX+PLUS II Simulator Channel Files (**.scf**), which can be used to simulate the function with the MAX+PLUS II software.

To ensure PCI timing and protocol compliance, the pci_b function was vigorously simulated and hardware tested. The simulations included hundreds of PCI cycles to cover the following details:

- All scenarios required by the PCI SIG's ***PCI Compliance Checklist, Revision 2.1***
- Additional applicable rules in Appendix C of the ***PCI Local Bus Specification, Revision 2.1***
- pci_b-specific functionality, including local-side interface operation

In addition to the simulation-based verification, Altera has performed vigorous hardware tests of the pci_b function against the most popular Intel PCI/host bridges, such as the 430NX, 430VX, 430TX, and 430HX bridges. The function was also tested against several DEC PCI/PCI bridges, such as the DEC 21052-AB bridge. The following method was used for each type of hardware testing:

- Implementing the pci_b function with a memory interface in an Altera FLEX 10K PCI prototype board
- Using the HP E2925A PCI Bus Exerciser and Analyzer for exercising the pci_b function, PCI protocol, and checking, as well as for monitoring the pci_b function in runtime transactions. These tests included:
  - Memory read/write of various burst length
  - Configuration read/write
  - I/O read/write
  - Abnormal terminations such as interrupts, target abort, target retry, target disconnect, and master abort
  - Random-wait-state insertions both on the PCI side and the local side

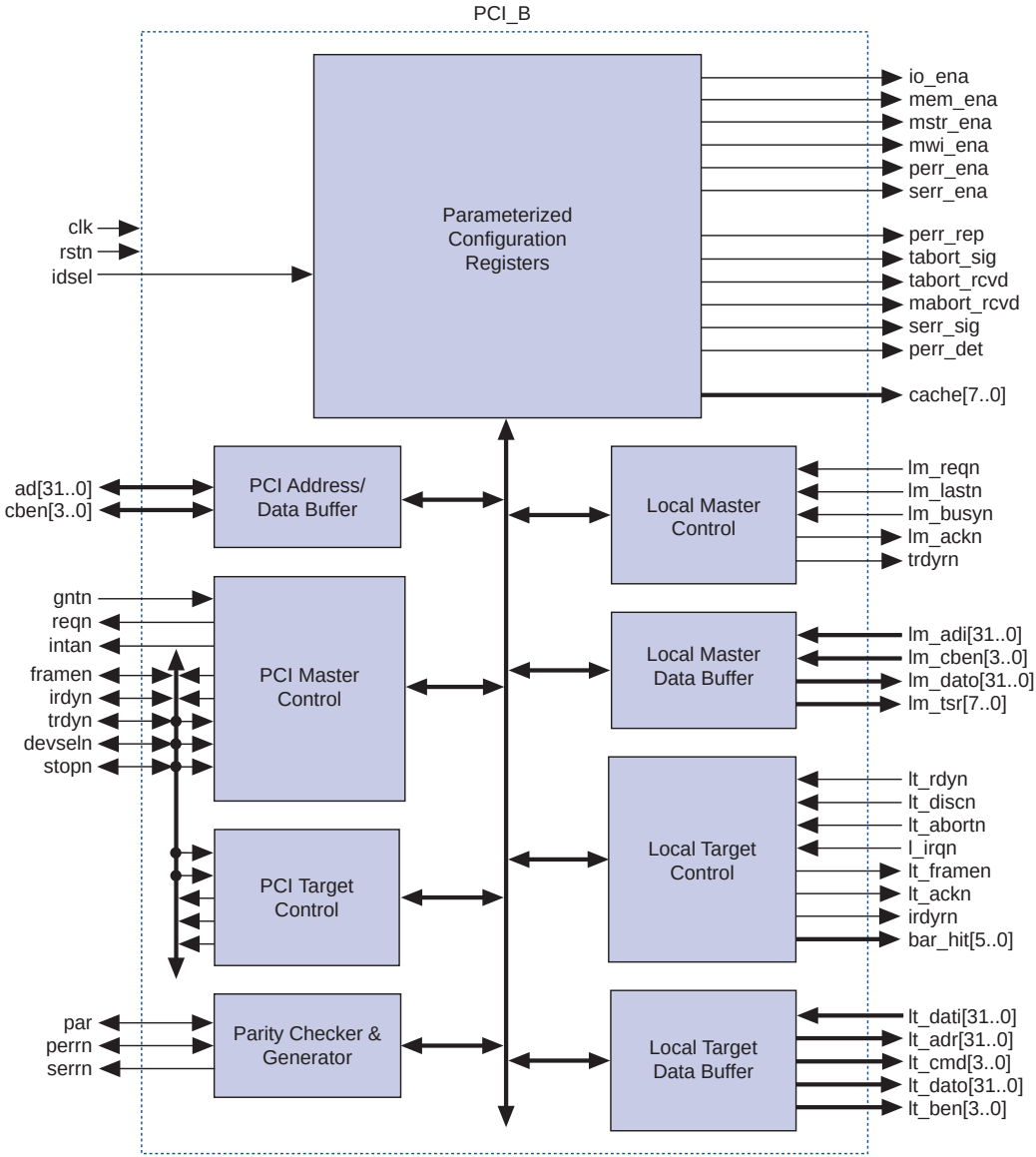
Functional Description

The pci_b function consists of four main components:

- Parameterized PCI bus configuration space
- Target interface control logic and data path
- Master interface control logic and data path
- Parity checker and generator

Figure 2 shows the pci_b functional block diagram.

Figure 2. pci_b Functional Block Diagram



Bus Commands

In target mode, the pci_b function responds to standard memory read/write, cache memory read/write, I/O read/write, and configuration read/write commands. In master mode, the pci_b function can initiate standard memory read/write, cache memory read/write, I/O read/write, and configuration read/write commands. When the master interface initiates MRM, MRL, and MWI, the PCI specification requires the master to keep track of the number of words transferred and to not initiate and terminate cache transaction except at cache line boundaries. It is the responsibility of the local logic to ensure this requirement is met.

Table 1 shows the PCI bus commands that can be initiated or responded to by the pci_b function. In accordance with the PCI SIG's ***PCI Local Bus Specification, Revision 2.1***, all unsupported and reserved commands are ignored by the pci_b function.

Table 1. PCI Bus Commands Supported by the pci_b Function

cben[3..0] Value	Bus Command Cycle	Master	Target
0000	Interrupt acknowledge	Ignored	Ignored
0001	Special cycle	Ignored	Ignored
0010	I/O read	Yes	Yes
0011	I/O write	Yes	Yes
0100	Reserved	Ignored	Ignored
0101	Reserved	Ignored	Ignored
0110	Memory read	Yes	Yes
0111	Memory write	Yes	Yes
1000	Reserved	Ignored	Ignored
1001	Reserved	Ignored	Ignored
1010	Configuration read	Yes	Yes
1011	Configuration write	Yes	Yes
1100	Memory read multiple (MRM)	Yes	Yes
1101	Dual address cycle	Ignored	Ignored
1110	Memory read line (MRL)	Yes	Yes
1111	Memory write and invalidate (MWI)	Yes	Yes

Target Operation

When responding to a memory transaction, the pci_b function supports an infinite cycle of zero-wait-state burst data transfers. Therefore, both memory read and write transfers achieve a sustained throughput of 132 Mbytes per second, which is the maximum bandwidth attainable in a 32-bit, 33-MHz PCI bus system. However, if the local logic cannot handle burst data during any data cycle, it may insert one or more wait states to stall data exchange on the PCI bus via the lt_rdyn port. For slower applications, it is usually desirable to implement a data buffer, such as a FIFO buffer, at the local side of the interface to post or pre-fetch data. The pci_b function provides separate buses for the input and output data and address. When coupled with FLEX 10K devices' unique embedded memory, high density, and flexibility, this feature makes it easy to add data buffers of various width and depth to the local logic.

In addition to high-performance data operation, the pci_b function also supports local-initiated abnormal termination. Under most conditions, data is successfully exchanged between master and target. However, when the local logic is unable to complete the request, it can use the lt_abortn or lt_discn signal to terminate the current PCI cycles. Depending on the inputs to the local logic and PCI bus, the pci_b function drives a combination of the control signals devseln, trdyn, and stopn to terminate the transaction with either a retry, disconnect, or abort.

Master Operation

As a PCI master, the pci_b function may initiate infinite cycles of zero-wait-state burst memory transfers, achieving a sustained bandwidth of 132 Mbytes per second. However, if local logic cannot process data during any data cycles, it may initiate one or more wait states to stall data exchange on the PCI bus via the lm_busyn port. In addition, the pci_b function can initiate the configuration and I/O transactions listed in Table 1 on page 7.

The pci_b function can act as a host bridge. When the HOST_BRIDGE_ENA parameter is set to "ON", the pci_b function can act as a host bridge to the PCI bus. As a host/PCI bridge, the pci_b function allows the local-side intelligent host to configure other agents on the PCI system. The local intelligent host can scan for devices present on the bus, construct a consistent memory map, handle interrupts and abnormal terminations, and even configure the pci_b function's own target.

During most cycles, data is exchanged successfully. When abnormal errors occur, the pci_b function automatically processes terminations (i.e., retry, abort, and disconnect), abstracting users from such errors. The pci_b function can also terminate a transaction when a transaction is not claimed by a target. In this scenario, the mabort_rcvd signal is driven.

For slower applications, it is usually desirable to implement a data buffer at the local side of the interface to post or pre-fetch data. The pci_b function provides separate address and data ports to allow the most optimal implementation of any type of buffer.

The master and target interfaces can operate independently of each other. For example, when the target interface is bursting memory write data through the lt_dato[31..0] port, the local logic can initiate a separate transfer by requesting bus mastership via the lm_reqn pin. This operation increases performance and reduces local latency.

Configuration Registers

Each logical PCI bus device includes a block of 256 bytes reserved for the implementation of its configuration registers. The format of the first 16 DWORDS, known as the configuration header, is defined by the PCI SIG's ***PCI Compliance Checklist, Revision 2.1***, which defines two header formats, type one and type zero. Header type one is used for PCI-to-PCI bridges; header type zero is used for all other devices, including the pci_b function.

Table 2 displays the defined 64-byte configuration space. The registers within this range are used to identify the device, control PCI bus functions, and provide PCI bus status. The shaded areas indicate registers that are supported by the pci_b function.

Table 2. PCI Bus Configuration Registers

Address	Byte			
	3	2	1	0
00H	Device ID		Vendor ID	
04H	Status Register		Command Register	
08H	Class Code			Revision ID
0CH	BIST	Header Type	Latency Timer	Cache Line Size
10H	Base Address Register 0			
14H	Base Address Register 1			
18H	Base Address Register 2			
1CH	Base Address Register 3			
20H	Base Address Register 4			
24H	Base Address Register 5			
28H	Card Bus CIS Pointer			
2CH	Subsystem ID		Subsystem Vendor ID	
30H	Expansion ROM Base Address Register			
34H	Reserved			
38H	Reserved			
3CH	Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line

Parameters

The pci_b parameters provide the flexibility to implement features that set PCI bus configuration registers. Modifying the INT_ARBITER_ENA, HOST_BRIDGE_ENA, NUMBER_OF_BARS, and BAR0 through BAR5 parameters automatically customizes the function's logic during compilation. Other parameters set read-only PCI configuration registers. See "Configuration Registers" on page 9 for more information on these registers. Table 3 describes the parameters of the pci_b function.

Table 3. pci_b Parameters

Name	Format	Default Value	Description
NUMBER_OF_BARS	Decimal	1	Number of BARs used. This parameter controls the number of BARs instantiated in the pci_b function at compile time. BARs are instantiated in sequential order, starting with BAR0.
INT_ARBITER_ENA	String	"OFF"	Internal arbiter enabled. When set to "ON", the reqn and gntn signals become internal signals, reducing the number of I/Os used by the pci_b function; when set to "OFF", the reqn and gntn signals become tri-stated signals.
HOST_BRIDGE_ENA	String	"OFF"	Host bridge enabled. When set to "ON", the master bit is tied to high.
TARGET_DEVICE	String	"EPF10K30RC240"	The device for which the pci_b function is targeted.
BAR0	Hexadecimal	H"FF000000"	BAR0
BAR1	Hexadecimal	H"FF000000"	BAR1
BAR2	Hexadecimal	H"FF000000"	BAR2
BAR3	Hexadecimal	H"FF000000"	BAR3
BAR4	Hexadecimal	H"FF000000"	BAR4
BAR5	Hexadecimal	H"FF000000"	BAR5
CLASS_CODE	Hexadecimal	H"FF0000"	Class code register
DEVICE_ID	Hexadecimal	H"0001"	Device ID register
VENDOR_ID	Hexadecimal	H"1172"	Device vendor ID register
REVISION_ID	Hexadecimal	H"01"	Revision ID register
SUBSYSTEM_ID	Hexadecimal	H"0000"	Subsystem ID register
SUBSYSTEM_VENDOR_ID	Hexadecimal	H"0000"	Subsystem vendor ID register
MIN_GRANT	Hexadecimal	H"0"	Minimum grant register
MAX_LATENCY	Hexadecimal	H"0"	Maximum latency register

The NUMBER_OF_BARS parameter defines the number of BARs to be instantiated in the pci_b function at compile time. Depending on the value of the NUMBER_OF_BARS parameter, some of the BAR0 through BAR5 parameters are ignored by the MAX+PLUS II software. For example, setting NUMBER_OF_BARS to 1 causes the BAR1 through BAR5 parameter values to be ignored because the corresponding BARs are not implemented.

The BAR0 through BAR5 parameters control the following properties of the corresponding BAR:

- Type of address space reserved (i.e., memory or I/O)
- Amount of memory or I/O space that is reserved
- Whether the memory space is pre-fetchable
- Whether the memory space can be located anywhere in the 32-bit address space, or if it must be mapped below 1 Mbyte

For example, if BAR0 = "H"FE000008"", the seven 1s in the most significant bits (MSBs) instantiate seven registers. This parameter value thus specifies that the pci_b function uses the following amount of memory space:

$2^{(32-7)}$ Bytes = 132 Mbytes

In addition, the least significant four bits specify that the function's memory is pre-fetchable, and can be located anywhere in the 32-bit address space.

pci_b Signals

The pci_b function uses the following PCI bus signals:

- *Input*—Standard input-only signal.
- *Output*—Standard output-only signal.
- *Bidirectional*—Tri-state input/output signal.
- *Sustained tri-state (STS)*—Signal that is driven by one agent at a time (e.g., device or host operating on the PCI bus). An agent that drives a sustained tri-state pin low must actively drive it high for one clock cycle before tri-stating it. Another agent cannot drive a sustained tri-state signal any sooner than one clock cycle after it is released by the previous agent.
- *Open-drain*—Signal that is wire-ORed with other agents. The signaling agent asserts the open-drain signal, and a weak pull-up resistor deasserts the open-drain signal. The pull-up resistor may take two or three PCI bus clock cycles to restore the open-drain signal to its inactive state.

PCI Bus Interface Signals

Table 4 describes the PCI bus signals that connect the pci_b function to the PCI bus.

Table 4. PCI Signals Connecting the pci_b Function to the PCI Bus (Part 1 of 2)

Name	Type/Direction	Polarity	Description
clk	Input	–	Clock. The clk input provides the reference signal for all other PCI interface signals, except rstn and intan.
rstn	Input	Low	Reset. The rstn input initializes the FLEX 10K PCI interface circuitry, and can be asserted asynchronously to the PCI bus clk edge. When active, the PCI output signals are tri-stated and the open-drain signals, such as serrn, float.
gntn	Input	Low	Grant. The gntn input indicates to the master device that it has control of the PCI bus. Each master device has a pair of arbitration lines (gntn and reqn) that connect directly to the arbiter.
reqn	Output	Low	Request. The reqn output indicates to the arbiter that the master wants to gain control of the PCI bus to perform a transaction.
ad[31..0]	Tri-State/ Bidirectional	–	Address/data bus. The ad[31..0] bus is a time-multiplexed address/data bus; each bus transaction consists of an address phase followed by one or more data phases. The data phases occur when irdyn and trdyn are both asserted.
cben[3..0]	Tri-State/ Bidirectional	Low	Command/byte enable. The cben[3..0] bus is a time-multiplexed command/byte enable bus. During the address phase, this bus indicates the command; during the data phase, this bus indicates byte enables.
par	Tri-State/ Bidirectional	–	Parity. The par signal has even parity across ad[31..0] and cben[3..0] (i.e., the number of 1s on ad[31..0], cben[3..0], and par equal an even number). On the clock following the data phase, the parity of a data phase is presented on the bus.
idsel	Input	High	Initialization device select. The idsel input is a chip select for configuration transactions.
framen, *Note (1)*	STS/ Bidirectional	Low	Frame. The framen signal is an output from the current bus master that indicates the beginning and duration of a bus operation. When framen is initially asserted, the address and command signals are present on the ad[31..0] and cben[3..0] buses. The framen signal remains asserted during the data operation and is deasserted to identify the end of a transaction.

Table 4. PCI Signals Connecting the pci_b Function to the PCI Bus (Part 2 of 2)

Name	Type/Direction	Polarity	Description
irdyn, *Note (1)*	STS/ Bidirectional Master: Output Target: Input	Low	Initiator ready. The irdyn signal is an output from a bus master to its target and indicates that the bus master can complete a data transaction. In a write transaction, irdyn indicates that valid data is on the ad[31..0] bus. In a read transaction, irdyn indicates that the master is ready to accept data on the ad[31..0] bus.
devseln, *Note (1)*	STS/ Bidirectional	Low	Device select. Target asserts devseln to indicate that the target has decoded its own address, and it accepts the transaction.
trdyn, *Note (1)*	STS/ Bidirectional	Low	Target ready. The trdyn signal is a target output, indicating that the target can complete the current data transaction. In a read operation, trdyn indicates that the target is providing data on the ad[31..0] bus. In a write operation, trdyn indicates that the target is ready to accept data on the ad[31..0] bus.
stopn, *Note (1)*	STS/ Bidirectional	Low	Stop. The stopn signal is a target device request that indicates to the bus master to terminate the current transaction. The stopn signal is used in conjunction with trdyn and devseln to indicate the type of termination initiated by the target.
perrn	STS/ Bidirectional	Low	Parity error. The perrn signal indicates a parity error. The perrn signal is asserted one clock following the par signal or two clocks following a data phase with a parity error.
serrn	Open-Drain/Output	Low	System error. The serrn signal indicates a system error and address parity error. The pci_b function asserts serrn if a parity error is detected during an address phase and the required bits in the PCI command register are set up accordingly.
intan	Open-Drain/Output	Low	Interrupt A. The intan signal is an active-low interrupt to the host, and must be used for any single-function device requiring an interrupt capability.

Note:

(1) In the pci_b function, these signals are actually split into two components: input and output. For example, framen has the input framen_in and the output framen_out. Splitting these signals allows designers to use devices that do not meet set up times for these signals.

The PCI bus and FLEX 10K devices allow IEEE Std. 1149.1 Joint Test Action Group (JTAG) boundary-scan testing. To use JTAG boundary-scan testing, designers should connect the PCI bus JTAG pins to the FLEX 10K device JTAG pins.

Table 5 summarizes the optional JTAG signals.

Table 5. Optional IEEE Std. 1149.1-1990 JTAG Signals

Name	Type	Polarity	Description
TCK	Input	High	Test clock. The TCK input is used to clock the test mode and test data in and out of the device.
TMS	Input	High	Test mode select. The TMS input is used to control the state of the Test Access Port (TAP) control in the device.
TDI	Input	High	Test data. The TDI input is used to shift the test data and instructions into the device.
TDO	Output	High	Test data. The TDO output is used to shift the test data and instructions out of the device.

Target Local-Side Signals

Table 6 summarizes the pci_b target interface signals that connect the function to the local-side peripheral device(s) during target transactions.

Table 6. pci_b Target Signals Connecting to the Local Side (Part 1 of 2)

Name	Direction	Polarity	Description
lt_dati[31..0]	Input	–	Local target data bus input. The lt_dati[31..0] bus is driven active by the local-side peripheral device during target read transactions.
lt_rdyn	Input	Low	Local target ready. The local side asserts lt_rdyn to indicate a valid data input during target read, or ready to accept data during a target write. During a target read, lt_rdyn deassertion suspends the current transfer (i.e., a wait state is inserted by the local side). During a target write, an inactive lt_rdyn signal directs pci_b to insert wait states on the PCI bus. The only time pci_b inserts wait states during a burst is when lt_rdyn inserts wait states on the local side.
lt_abortn	Input	Low	Local target abort request. A local device asserts lt_abortn when it encounters a fatal error and cannot complete the current transaction. Therefore, it is requesting the pci_b function to issue a target abort to the PCI master.
lt_discn	Input	Low	Local target disconnect request. The lt_discn input is used to signal a request for both a retry and a disconnect depending on when the signal is asserted during a transaction.

Table 6. pci_b Target Signals Connecting to the Local Side (Part 2 of 2)

Name	Direction	Polarity	Description
lt_framen	Output	Low	Local target frame. The lt_framen output is asserted while pci_b is engaged in a PCI transaction. It is asserted one clock before pci_b asserts devseln, and it is released after the last data phase of the transaction is completed on the PCI bus.
lt_ackn	Output	Low	Local target acknowledge. The pci_b function asserts lt_ackn to indicate a valid data output during a target write, or that it is ready to accept data during a target read. During a target read, an inactive lt_ackn signal indicates that pci_b is not ready to accept data, and local logic should hold off the bursting operation. During a target write, lt_ackn de-assertion suspends the current transfer (i.e., a wait state is inserted by the PCI master). The only time lt_ackn goes inactive during a burst is when the PCI bus master inserts a wait state.
irdyrn	Output	Low	Registered irdyn. This signal is a registered version of the PCI irdyn signal and is delayed by one clock.
lt_dato[31..0]	Output	–	Local target data bus output. The lt_dato[31..0] bus is driven active by the local-side peripheral device during target write transactions.
lt_adr[31..0]	Output	–	Local target address bus. The lt_adr[31..0] bus represents the target memory address for the current local-side data phase. The pci_b function increments lt_adr[31..0] after a successful data transfer is completed on the local side (i.e., lt_rdyn and lt_ackn are active during the same clock).
lt_cmd[3..0]	Output	–	Local target command. The lt_cmd[3..0] bus represents the PCI command for the current claimed transaction. The lt_cmd[3..0] bus uses the same encoding scheme as the cben[3..0] bus.
lt_ben[3..0]	Output	Low	Local target byte enable bus. The lt_ben[3..0] bus represents the byte enable requests from the PCI master during data phases.
bar_hit[5..0]	Output	High	Base address register hit. The bar_hit[5..0] bus assertion indicates that the PCI address matches that of a BAR, and the pci_b function has claimed the transaction. Each bit in the bar_hit[5..0] bus is used for the corresponding BAR. Therefore, the bar_hit[0] signal is used for BAR0. The bar_hit[5..0] bus has the same timing as the lt_framen signal.
l_irqn	Input	Low	Local interrupt request. The local-side peripheral device asserts l_irqn to signal a PCI bus interrupt. Asserting this signal forces pci_b to assert the intan signal for as long as l_irqn is asserted.

Master Local-Side Signals

Table 7 summarizes the pci_b master interface signals that connect the function to the local-side peripheral device(s) during master transactions.

Table 7. pci_b Master Signals Connecting to the Local Side (Part 1 of 2)

Name	Type	Polarity	Description
lm_reqn	Input	Low	Local master request. The local side asserts this signal to request ownership of the PCI bus for a master transaction. The local-side device must supply the PCI bus address and command in the same clock when lm_reqn goes from high to low. To request a master transaction, it is sufficient for the local-side device to assert lm_reqn and drive the address and command for one clock.
lm_lastn	Input	Low	Local master last. This signal is driven by the local-side application to request the pci_b master interface to end the current transaction. When the local side asserts this signal, the pci_b master interface deasserts framen as soon as possible and asserts irdyn to indicate that the last data phase has begun. It is sufficient for the local side to assert this signal for one clock cycle any time during the master transaction.
lm_busyn	Input	Low	Local master busy. The local side asserts this signal to request a wait state for the pci_b burst from/to the local side. Asserting this signal generally results in pci_b deasserting irdyn on the PCI bus to request wait states. A local data transfer occurs only if lm_ackn is asserted. Therefore, asserting lm_busyn results in deasserting lm_ackn.
lm_adi[31..0]	Input	–	Local master address/data bus. The local side must drive the transaction address at the same time as it asserts lm_reqn to request the master transaction. In all other cases, lm_adi[31..0] carries data from the local-side application for write transactions. A local-side data transfer is complete when lm_ackn is asserted. If the local side is unable to transfer data, it must assert lm_busyn. This situation results in lm_ackn being deasserted, indicating that a data transfer did not take place.
lm_cben[3..0]	Input	Low	Local master command/byte enable bus. This bus is a local-side time-multiplexed command/byte enable bus. The local side must drive the transaction command at the same time it asserts lm_reqn to request the master transaction. In all other cases, lm_cben[3..0] carries byte enable information. During a burst transaction, it is often not possible to maintain synchronization between data transferred on the PCI bus and local-side byte enables. Therefore, the pci_b function only clocks the byte enable signal on the first data phase.

Table 7. pci_b Master Signals Connecting to the Local Side (Part 2 of 2)

Name	Type	Polarity	Description
lm_ackn	Output	Low	Local master acknowledge. The pci_b master interface asserts this signal when a local-side data transfer occurs. During a write transaction, pci_b asserts this signal when it internally latches data from the local side. In a read transaction, the pci_b function asserts this signal when it transfers data to the local side. When the local side is not ready to receive/send data, it must assert lm_busyn. Therefore, during a master transaction, the pci_b function deasserts lm_ackn if the lm_busyn signal was asserted or if the PCI target deasserts its trdyn signal. The operation of lm_ackn differs from the operation of lt_ackn.
trdyrn	Output	Low	Registered trdyn. This signal is a registered version of the PCI trdyn signal and is delayed by one clock.
lm_dato[31..0]	Output	–	Local master data bus. The pci_b function drives data on this bus during a master read transaction. Successful data transfer occurs when pci_b asserts lm_ackn. If the local side is unable to transfer data, it must assert lm_busyn.
lm_tsr[7..0]	Output	–	Local master transaction status register bus. These signals inform the local interface of the progress of the transaction. See Table 8 for a detailed description of these bits.

Table 8 summarizes the bits for the local master transaction status register.

Table 8. Local Master Transaction Status Register Bits (Part 1 of 2)

Bit Number	Bit Name	Description
0	tsr_req	Request. This bit indicates that the pci_b function is requesting control of the PCI bus (i.e., it asserted the reqn signal).
1	tsr_gnt	Grant. This signal is active after the pci_b function detects that gntn is asserted and while the pci_b function is in the transaction address phase.
2	tsr_dat_xfr	Data transfer. This signal is active while the pci_b function is in data transfer mode. It is active after the address phase and remains active until the turn around state begins.
3	tsr_lat_exp	Latency timer expired. This signal indicates that pci_b terminated the master transaction because the latency timer counter expired.
4	tsr_ret	Retry detected. This signal indicates that pci_b terminated the master transaction because the target issued a retry. The PCI specification requires that a transaction that ends in a retry must be retried at a later time.
5	tsr_disc_wod	Disconnect without data detected. This signal indicates that pci_b terminated the master transaction because the target issued a disconnect without data.

Table 8. Local Master Transaction Status Register Bits (Part 2 of 2)

Bit Number	Bit Name	Description
6	tsr_disc_wd	Disconnect with data detected. This signal indicates that pci_b terminated the master transaction because the target issued a disconnect with data.
7	tsr_dat_phase	Data phase. This signal indicates that a successful data transfer has occurred on the PCI side in the prior clock. This signal is useful for the local side to keep track of how much data was actually transferred on the PCI side.

Configuration Space Output Signals

Table 9 summarizes the pci_b configuration signals, which are useful for local-side applications. For a detailed description of the registers, refer to the ***PCI Local Bus Specification, Revision 2.1***.

Table 9. pci_b Configuration Space Output Signals

Name	Polarity	Description
cache[7..0]	–	PCI cache line register. The local-side application must use this signal when using MWI or MRL commands.
io_ena	High	I/O space enable. PCI command register bit 0.
mem_ena	High	Memory space enable. PCI command register bit 1.
mstr_ena	High	Master enable. PCI command register bit 2.
mwi_ena	High	Memory write and invalidate enable. PCI command register bit 4.
perr_ena	High	Parity error response enable. PCI command register bit 6.
serr_ena	High	System error enable. PCI command register bit 8.
perr_rep	High	The perrn signal was detected during a master write transaction. PCI status register bit 8.
tabort_sig	High	The pci_b function signaled target abort. PCI status register bit 11.
tabort_rcvd	High	The pci_b function received a target abort. PCI status register bit 12.
mabort_rcvd	High	The pci_b function received a master abort. PCI status register bit 13.
serr_sig	High	Signaled system error. PCI status register bit 14.
perr_det	High	The pci_b function detected a data or address parity error. PCI status register bit 15.

References

Reference documents for the pci_b function include:

- ***PCI MegaCore Function User Guide.*** San Jose, California: Altera Corporation, September 1998.
- ***PCI Local Bus Specification***. ***Revision 2.1***. Portland, Oregon: PCI Special Interest Group, June 1995.
- ***PCI Compliance Checklist***. ***Revision 2.1.*** Portland, Oregon.
- ***1998 Data Book***. San Jose, California: Altera Corporation, January 1998.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com
Applications Hotline:
(800) 800-EPLD
Customer Marketing:
(408) 544-7104
Literature Services:
(888) 3-ALTERA
lit_req@altera.com