# 21154 PCI-to-PCI Bridge

**Datasheet**

## Product Features

- Complies fully with the *PCI Local Bus Specification*, Revision 2.1
- Complies fully with the *PCI Power Management Specification*, Revision 1.0[1]
- Supports 64-bit extension signals on the primary and secondary interfaces
- Implements delayed transactions for all PCI configuration, I/O, and memory read commands–up to three transactions simultaneously in each direction
- Allows 152 bytes of buffering (data and address) for upstream posted memory write commands and 88 bytes of buffering for downstream posted memory write commands—up to nine upstream and five downstream posted write transactions simultaneously
- Allows 152 bytes of read data buffering upstream and 152 bytes of read data buffering downstream
- Provides concurrent primary and secondary bus operation to isolate traffic
- Provides ten secondary clock outputs:
  - —Low skew, permitting direct drive of option slots
  - —Individual clock disables, capable of automatic configuration during reset
- Provides arbitration support for nine secondary bus devices:
  - —A programmable 2-level arbiter
  - —Hardware disable control, permitting use of an external arbiter
- Provides a 4-pin general-purpose I/O interface, accessible through device-specific configuration space
- Provides enhanced address decoding:
  - —A 32-bit I/O address range
  - —A 32-bit memory-mapped I/O address range
  - —A 64-bit prefetchable memory address range
  - —ISA-aware mode for legacy support in the first 64KB of I/O address range
  - — VGA addressing and VGA palette snooping support
- Includes live insertion support
- Supports PCI transaction forwarding for the following commands:
  - —All I/O and memory commands
  - —Type 1 to Type 1 configuration commands
  - —Type 1 to Type 0 configuration commands (downstream only)
  - —All Type 1 to special cycle configuration commands
- Includes downstream lock support
- Supports both 5-V and 3.3-V signaling environments
- Available in both 33 MHz and 66 Mhz versions
- Provides an IEEE standard 1149.1 JTAG interface

[1] For the 21154–AB and later revisions only. The 21154–AA does not implement this feature.

Order Number: 278108-002
July 1999



**Datasheet**

# Contents

# Figures



# Tables

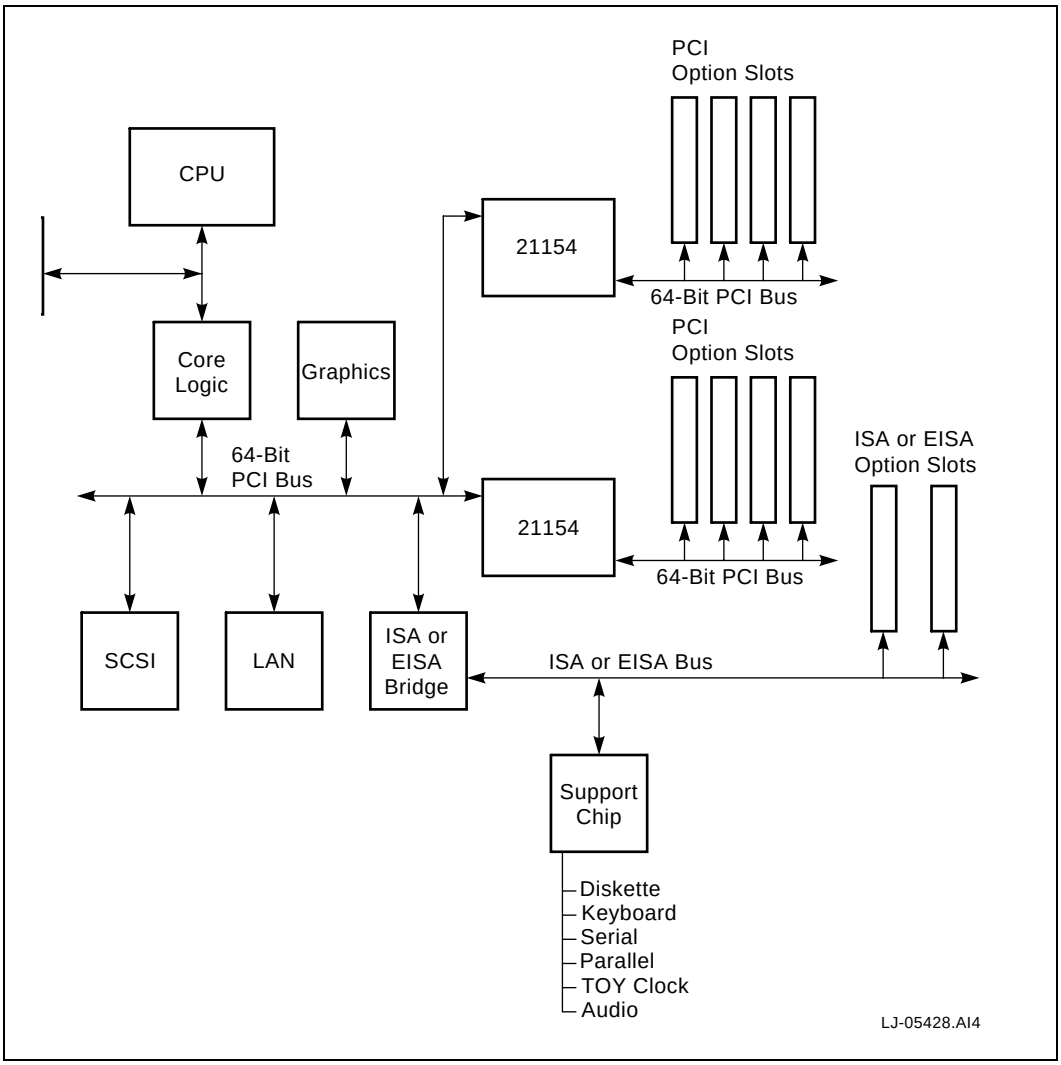
# 1.0 Introduction

The 21154 is a second-generation PCI-to-PCI bridge and is fully compliant with *PCI Local Bus Specification, Revision 2.1*. The 21154 has a 64-bit primary bus interface and a 64-bit secondary interface. The 64-bit interfaces interoperate transparently with either 64-bit or 32-bit devices. The 21154 provides full support for delayed transactions, which enables the buffering of memory read, I/O, and configuration transactions. The 21154 has separate posted write, read data, and delayed transaction queues with the most buffering capability of all Intel's 2115*x* PCI-to-PCI bridge products. In addition, the 21154 supports buffering of simultaneous multiple posted write and delayed transactions in both directions. Among the features provided by the 21154 are: a programmable 2-level secondary bus arbiter, an IEEE standard 1149.1 JTAG interface, live insertion support, a 4-pin general-purpose I/O interface, individual secondary clock disables, and enhanced address decoding. The 21154 has sufficient clock and arbitration pins to support nine PCI bus master devices directly on its secondary interface.

The 21154 allows the two PCI buses to operate concurrently. This means that a master and a target on the same PCI bus can communicate while the other PCI bus is busy. This traffic isolation may increase system performance in applications such as multimedia.

Figure 1 illustrates the use of two 21154 PCI-to-PCI bridges on a system board. Each 21154 that is added to the board creates a new PCI bus that provides support for the additional PCI slots or devices

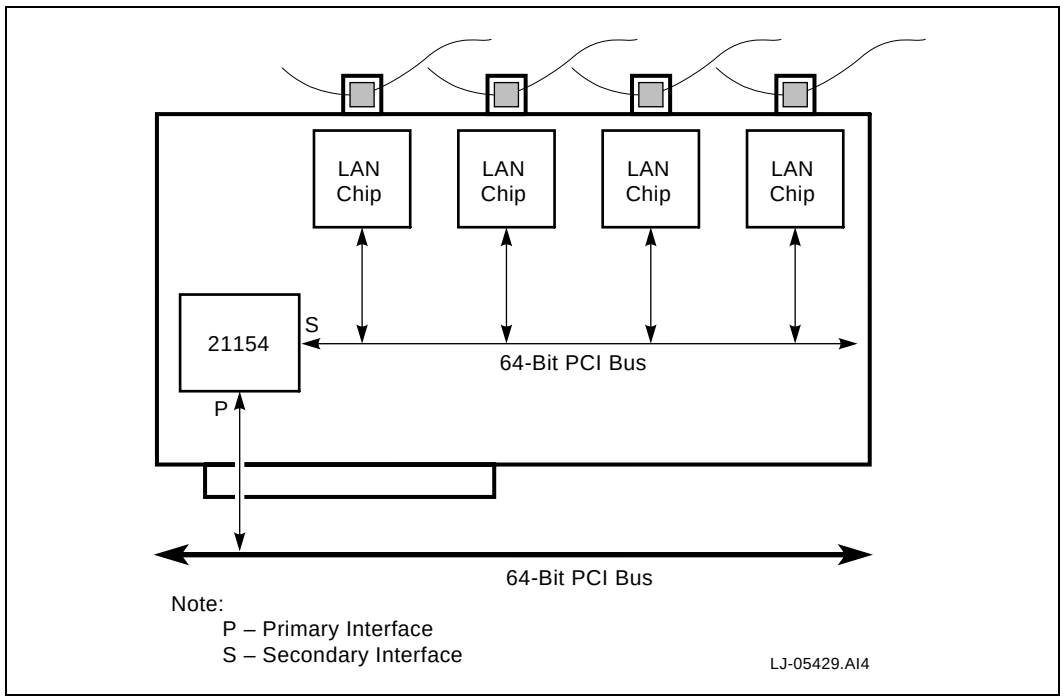
**Figure 1. 21154 on the System Board**




Option card designers can use the 21154 to implement multiple-device PCI option cards. Without a PCI-to-PCI bridge, PCI loading rules would limit option cards to one device. The *PCI Local Bus Specification* loading rules limit PCI option cards to a single connection per PCI signal in the option card connector. However, the 21154 overcomes this restriction by providing, on the option card, an independent PCI bus to which up to nine devices can be attached.

Figure 2 shows how the 21154 enables the design of a multicomponent option card.

**Figure 2. 21154 with Option Cards**




## 1.1 Architecture

The 21154 internal architecture consists of the following major functions:

- PCI interface control logic for the primary and secondary PCI interfaces
- Data path and data path control logic
- Configuration register and configuration control logic
- Secondary bus arbiter

Figure 3 shows the major functional blocks of the 21154.

**Figure 3. 21154 Block Diagram**

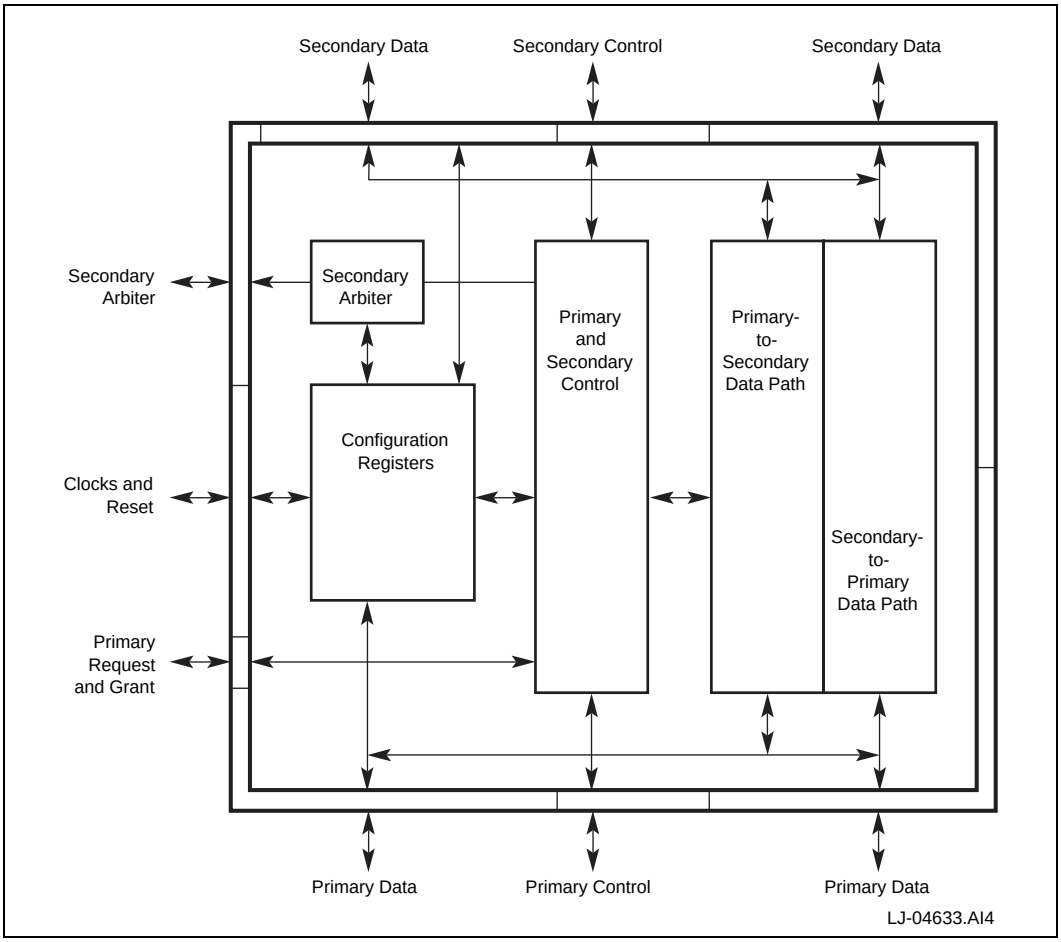



Table 1 describes the major functional blocks of the 21154.

**Table 1. 21154 Function Blocks (Sheet 1 of 2)**

| Function Block | Description |
|---|---|
| Primary and Secondary Control | PCI interface control logic. This block contains state machines and control logic for the primary target interface, the primary master interface, the secondary target interface, and the secondary master interface. This block also contains logic that interfaces to the data path and the configuration block. |
| Primary-to-Secondary Data Path | Data path for data received on the primary interface and driven on the secondary interface. This block is used for write transactions initiated on the primary PCI bus and for returning read data for read transactions initiated on the secondary PCI bus. This block contains logic to store and, for posted write transactions, to increment the address of the current transaction. This block also performs bus command and configuration address format translations. |

**Table 1. 21154 Function Blocks (Sheet 2 of 2)**

| Function Block | Description |
|---|---|
| Secondary-to-Primary Data Path | Data path for data received on the secondary interface and driven on the primary interface. This block is used for write transactions initiated on the secondary PCI bus and for returning read data for read transactions initiated on the primary PCI bus. This block contains logic to store and, for posted write transactions, to increment the address of the current transaction. This block also performs bus command and configuration address format translations |
| Configuration Registers | Configuration space registers and corresponding control logic. These registers are accessible from the primary interface only. |
| Secondary Bus Arbiter Control | Logic for secondary bus arbitration. This block receives s_req_l<8:0>, as well as the 21154 secondary bus request, and drives one of the s_gnt_l<8:0> lines or the 21154 secondary bus grant. |

## 1.2 Data Path

The data path consists of a primary-to-secondary data path for transactions and data flowing in the downstream direction and a secondary-to-primary data path for transactions and data flowing in the upstream direction.

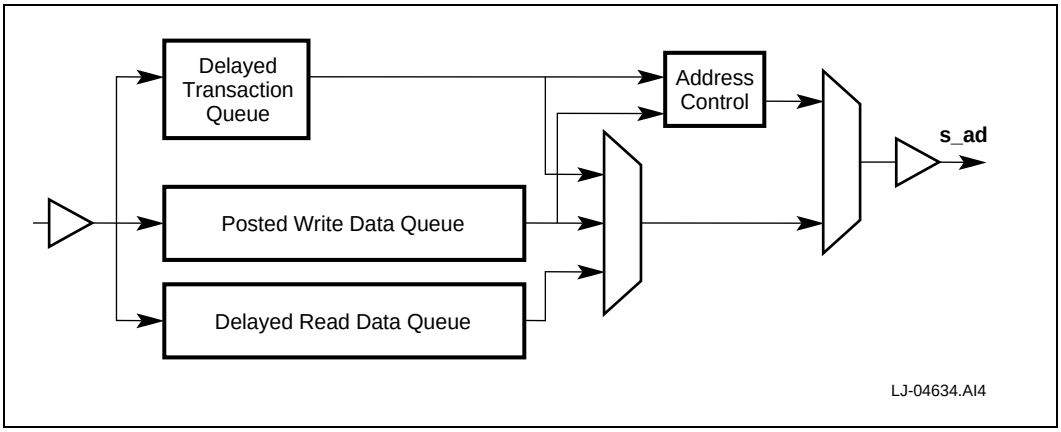
Both data paths have the following queues:

- Posted write queue
- Delayed transaction queue
- Read data queue

To prevent deadlocks and to maintain data coherency, a set of ordering rules is imposed on the forwarding of posted and delayed transactions across the 21154. The queue structure, along with the order in which the transactions in the queues are initiated and completed, supports these ordering requirements. Section 6.0 describes the 21154 ordering rules in detail.

See Section 4.0 for a detailed description of 21154 PCI bus operation. Figure 4 shows the 21154 data path for the downstream direction, and the following sections describe the data path queues.

**Figure 4. 21154 Downstream Data Path**



### 1.2.1 Posted Write Queue

The posted write queue contains the address and data of memory write transactions targeted for the opposite interface. The posted write transaction can consist of an arbitrary number of data phases, subject to the amount of space in the queue and disconnect boundaries. The posted write queue can contain multiple posted write transactions. The number of posted write transactions that can be queued at one time is dependent upon their burst size. The posted write queue consists of 152 bytes in the upstream direction and 88 bytes in the downstream direction.

### 1.2.2 Delayed Transaction Queue

For a delayed write request transaction, the delayed transaction queue contains the address, bus command, 1 Dword of write data, byte enable bits, and parity. When the delayed write transaction is completed on the target bus, the write completion status is added to the corresponding entry.

For a delayed read request transaction, the delayed transaction queue contains the address and bus command, and for nonprefetchable read transactions, the byte enable bits. When the delayed read transaction is completed on the target bus, the read completion status corresponding to that transaction is added to the delayed request entry. Read data is placed in the read data queue.

The delayed transaction queue can hold up to three transactions (any combination of read and write transactions).

### 1.2.3 Read Data Queue

The read data queue contains read data transferred from the target during a delayed read completion. Read data travels in the opposite direction of the transaction. The primary-to-secondary read data queue contains read data corresponding to a delayed read transaction residing in the secondary-to-primary delayed transaction queue. The secondary-to-primary read data queue contains read data corresponding to a delayed read transaction in the primary-to-secondary delayed transaction queue. The amount of read data per transaction depends on the amount of space in the queue and disconnect boundaries.

Read data for up to three transactions, subject to the burst size of the read transactions and available queue space, can be stored. The read data queue for the 21154 consists of 152 bytes pointing upstream and 152 bytes pointing downstream.

# 2.0 Signal Pins

This chapter provides detailed descriptions of the 21154 signal pins, grouped by function.

Table 2 describes the signal pin functional groups, and the following sections describe the signals in each group.

**Table 2. Signal Pin Functional Groups**

| Function | Description |
|---|---|
| Primary PCI bus interface signal pins | All PCI pins required by the *PCI-to-PCI Bridge Architecture Specification.* |
| Primary PCI bus interface 64-bit extension signal pins | All 64-bit extension signal pins defined by the *PCI Local Bus Specification.* |
| Secondary PCI bus interface signal pins | All PCI pins required by the *PCI-to-PCI Bridge Architecture Specification.* |
| Secondary PCI bus interface 64-bit extension signal pins | All 64-bit extension signal pins defined by the *PCI Local Bus Specification.* |
| Secondary PCI bus arbiter signal pins | Nine request/grant pairs of pins for the secondary PCI bus and arbiter enable control pin. |
| General-purpose I/O interface | Four general-purpose pins. |
| Clock signal pins | Two clock inputs (one for each PCI interface). Ten clock outputs (for nine external secondary PCI bus devices and also for the 21154). |
| Reset signal pins | A primary interface reset input and a secondary interface reset output. |
| Miscellaneous signal pins | An input-only pin used to disable secondary clock outputs.<br>Two input voltage signaling level pins.<br>Three pins controlling 66 MHz operation. |
| JTAG signal pins | All JTAG pins required by IEEE standard 1149.1. |

Table 3 defines the signal type abbreviations used in the signal tables:

**Table 3. Signal Type Abbreviations**

| Signal Type | Description |
|---|---|
| I | Standard input only. |
| O | Standard output only. |
| TS | Tristate bidirectional. |
| STS | Sustained tristate. Active low signal must be pulled high for one cycle when deasserting. |
| OD | Standard open drain. |

***Note:*** The **_l** signal name suffix indicates that the signal is asserted when it is at a low voltage level and corresponds to the "#" suffix in the *PCI Local Bus Specification*. If this suffix is not present, the signal is asserted when it is at a high voltage level.

## 2.1 Primary PCI Bus Interface Signals

Table 4 describes the primary PCI bus interface signals.

**Table 4. Primary PCI Bus Interface Signals (Sheet 1 of 3)**

| Signal Name | Type | Description |
|---|---|---|
| p_ad<31:0> | TS | Primary PCI interface address/data. These signals are a multiplexed address and data bus. During the address phase or phases of a transaction, the initiator drives a physical address on p_ad<31:0>. During the data phases of a transaction, the initiator drives write data, or the target drives read data, on p_ad<31:0>. When the primary PCI bus is idle, the 21154 drives p_ad to a valid logic level when p_gnt_l is asserted. |
| p_cbe_l<3:0> | TS | Primary PCI interface command/byte enables. These signals are a multiplexed command field and byte enable field. During the address phase or phases of a transaction, the initiator drives the transaction type on p_cbe_l<3:0>. When there are two address phases, the first address phase carries the dual address command and the second address phase carries the transaction type. For both read and write transactions, the initiator drives byte enables on p_cbe_l<3:0> during the data phases. When the primary PCI bus is idle, the 21154 drives p_cbe_l to a valid logic level when p_gnt_l is asserted. |
| p_par | TS | Primary PCI interface parity. Signal p_par carries the even parity of the 36 bits of p_ad<31:0> and p_cbe_l<3:0> for both address and data phases. Signal p_par is driven by the same agent that has driven the address (for address parity) or the data (for data parity). Signal p_par contains valid parity one cycle after the address is valid (indicated by assertion of p_frame_l), or one cycle after data is valid (indicated by assertion of p_irdy_l for write transactions and p_trdy_l for read transactions). Signal p_par is driven by the device driving read or write data one cycle after p_ad is driven. Signal p_par is tristated one cycle after the p_ad lines are tristated. Devices receiving data sample p_par as an input to check for possible parity errors. When the primary PCI bus is idle, the 21154 drives p_par to a valid logic level when p_gnt_l is asserted (one cycle after the p_ad bus is parked). |
| p_frame_l | STS | Primary PCI interface FRAME#. Signal p_frame_l is driven by the initiator of a transaction to indicate the beginning and duration of an access on the primary PCI bus. Signal p_frame_l assertion (falling edge) indicates the beginning of a PCI transaction. While p_frame_l remains asserted, data transfers can continue. The deassertion of p_frame_l indicates the final data phase requested by the initiator. When the primary PCI bus is idle, p_frame_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |

**Table 4. Primary PCI Bus Interface Signals (Sheet 2 of 3)**

| Signal Name | Type | Description |
|---|---|---|
| p_irdy_l | STS | Primary PCI interface IRDY#. Signal p_irdy_l is driven by the initiator of a transaction to indicate the initiator's ability to complete the current data phase on the primary PCI bus. During a write transaction, assertion of p_irdy_l indicates that valid write data is being driven on the p_ad bus. During a read transaction, assertion of p_irdy_l indicates that the initiator is able to accept read data for the current data phase. Once asserted during a given data phase, p_irdy_l is not deasserted until the data phase completes. When the primary bus is idle, p_irdy_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| p_trdy_l | STS | Primary PCI interface TRDY#. Signal p_trdy_l is driven by the target of a transaction to indicate the target's ability to complete the current data phase on the primary PCI bus. During a write transaction, assertion of p_trdy_l indicates that the target is able to accept write data for the current data phase. During a read transaction, assertion of p_trdy_l indicates that the target is driving valid read data on the p_ad bus. Once asserted during a given data phase, p_trdy_l is not deasserted until the data phase completes. When the primary bus is idle, p_trdy_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| p_devsel_l | STS | Primary PCI interface DEVSEL#. Signal p_devsel_l is asserted by the target, indicating that the device is accepting the transaction. As a target, the 21154 performs positive decoding on the address of a transaction initiated on the primary bus to determine whether to assert p_devsel_l. As an initiator of a transaction on the primary bus, the 21154 looks for the assertion of p_devsel_l within five cycles of p_frame_l assertion; otherwise, the 21154 terminates the transaction with a master abort. When the primary bus is idle, p_devsel_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| p_stop_l | STS | Primary PCI interface STOP#. Signal p_stop_l is driven by the target of the current transaction, indicating that the target is requesting the initiator to stop the current transaction on the primary bus.<br>When p_stop_l is asserted in conjunction with p_trdy_l and p_devsel_l assertion, a disconnect with data transfer is being signaled.<br>When p_stop_l and p_devsel_l are asserted, but p_trdy_l is deasserted, a target disconnect without data transfer is being signaled. When this occurs on the first data phase, that is, no data is transferred during the transaction, this is referred to as a target retry.<br>When p_stop_l is asserted and p_devsel_l is deasserted, the target is signaling a target abort.<br>When the primary bus is idle, p_stop_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| p_lock_l | I | Primary PCI interface LOCK#. Signal p_lock_l is deasserted during the first address phase of a transaction and is asserted one clock cycle later by an initiator attempting to perform an atomic operation that may take more than one PCI transaction to complete. The 21154 samples p_lock_l as a target and can propagate the lock across to the secondary bus. The 21154 does not drive p_lock_l as an initiator; that is, the 21154 does not propagate locked transactions upstream. When released by an initiator, p_lock_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| p_idsel | I | Primary PCI interface IDSEL#. Signal p_idsel is used as the chip select line for Type 0 configuration accesses to 21154 configuration space. When p_idsel is asserted during the address phase of a Type 0 configuration transaction, the 21154 responds to the transaction by asserting p_devsel_l. |

**Table 4. Primary PCI Bus Interface Signals (Sheet 3 of 3)**

| Signal Name | Type | Description |
|---|---|---|
| p_perr_l | STS | Primary PCI interface PERR#. Signal p_perr_l is asserted when a data parity error is detected for data received on the primary interface. The timing of p_perr_l corresponds to p_par driven one cycle earlier and p_ad and p_cbe_l driven two cycles earlier. Signal p_perr_l is asserted by the target during write transactions, and by the initiator during read transactions. When the primary bus is idle, p_perr_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| p_serr_l | OD | Primary PCI interface SERR#. Signal p_serr_l can be driven low by any device on the primary bus to indicate a system error condition. The 21154 can assert p_serr_l for the following reasons:<br>Address parity error<br>Posted write data parity error on target bus<br>Secondary bus s_serr_l assertion<br>Master abort during posted write transaction<br>Target abort during posted write transaction<br>Posted write transaction discarded<br>Delayed write request discarded<br>Delayed read request discarded<br>Delayed transaction master timeout<br>Signal p_serr_l is pulled up through an external resistor. |
| p_req_l | TS | Primary PCI bus REQ#. Signal p_req_l is asserted by the 21154 to indicate to the primary bus arbiter that it wants to start a transaction on the primary bus. When the 21154 receives a target retry or disconnect in response to initiating a transaction, the 21154 deasserts p_req_l for at least two PCI clock cycles before asserting it again. |
| p_gnt_l | I | Primary PCI bus GNT#. When asserted, p_gnt_l indicates to the 21154 that access to the primary bus is granted. The 21154 can start a transaction on the primary bus when the bus is idle and p_gnt_l is asserted. When the 21154 has not requested use of the bus and p_gnt_l is asserted, the 21154 must drive p_ad, p_cbe_l, and p_par to valid logic levels. |

## 2.2 Primary PCI Bus Interface 64-Bit Extension Signals

Table 5 describes the primary PCI bus interface 64-bit extension signals.

**Table 5. Primary PCI Bus Interface 64-Bit Extension Signals**

| Signal Name | Type | Description |
|---|---|---|
| p_ad<63:32> | TS | Primary PCI interface address/data upper 32 bits. This multiplexed address and data bus provides an additional 32 bits to the primary interface. During the address phase or phases of a transaction, when the dual address command is used and p_req64_l is asserted, the initiator drives the upper 32 bits of a 64-bit address; otherwise, these bits are undefined, and the initiator drives a valid logic level onto the pins. During the data phases of a transaction, the initiator drives the upper 32 bits of 64-bit write data, or the target drives the upper 32 bits of 64-bit read data, when p_req64_l and p_ack64_l are both asserted. When not driven, signals p_ad<63:32> are pulled up to a valid logic level through external resistors. |
| p_cbe_l<7:4> | TS | Primary PCI interface command/byte enables upper 4 bits. These signals are a multiplexed command field and byte enable field. During the address phase or phases of a transaction, when the dual address command is used and p_req64_l is asserted, the initiator drives the transaction type on p_cbe_l<7:4>; otherwise, these bits are undefined, and the initiator drives a valid logic level onto the pins. For both read and write transactions, the initiator drives byte enables for the p_ad<63:32> data bits on p_cbe_l<7:4> during the data phases when p_req64_l and p_ack64_l are both asserted. When not driven, signals p_cbe_l<7:4> are pulled up to a valid logic level through external resistors. |
| p_par64 | TS | Primary PCI interface upper 32 bits parity. Signal p_par64 carries the even parity of the 36 bits of p_ad<63:32> and p_cbe_l<7:4> for both address and data phases. Signal p_par64 is driven by the initiator and is valid one cycle after the first address phase when a dual address command is used and p_req64_l is asserted. Signal p_par64 is also valid one clock cycle after the second address phase of a dual address transaction when p_req64_l is asserted. Signal p_par64_l is valid one cycle after valid data is driven (indicated by assertion of p_irdy_l for write data and p_trdy_l for read data) when both p_req64_l and p_ack64_l are asserted for that data phase. Signal p_par64 is driven by the device driving read or write data one cycle after the p_ad lines are driven. Signal p_par64 is tristated one cycle after the p_ad lines are tristated. Devices receive data sample p_par64 as an input to check for possible parity errors during 64-bit transactions. When not driven, p_par64 is pulled up to a valid logic level through external resistors. |
| p_req64_l | STS | Primary PCI interface request 64-bit transfer. Signal p_req64_l is asserted by the initiator to indicate that the initiator is requesting a 64-bit data transfer. Signal p_req64_l has the same timing as p_frame_l. When p_req64_l is asserted low during reset, a 64-bit data path is supported on the board. When p_req64_l is high during reset (indicating that a 64-bit data path is not supported on the board), the 21154 drives p_ad<63:32>, p_cbe_l<7:4>, and p_par64 to valid logic levels. When deasserting, p_req64_l is driven to a deasserted state for one cycle and then sustained by an external pull-up resistor. |
| p_ack64_l | STS | Primary PCI interface acknowledge 64-bit transfer. Signal p_ack64_l is asserted by the target only when p_req64_l is asserted by the initiator, to indicate the target's ability to transfer data using 64 bits. Signal p_ack64_l has the same timing as p_devsel_l. When deasserting, p_ack64_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |

## 2.3 Secondary PCI Bus Interface Signals

Table 6 describes the secondary PCI bus interface signals.

**Table 6. Secondary PCI Bus Interface Signals (Sheet 1 of 2)**

| Signal Name | Type | Description |
|---|---|---|
| s_ad<31:0> | TS | Secondary PCI interface address/data. These signals are a multiplexed address and data bus. During the address phase or phases of a transaction, the initiator drives a physical address on s_ad<31:0>. During the data phases of a transaction, the initiator drives write data, or the target drives read data, on s_ad<31:0>. When the secondary PCI bus is idle, the 21154 drives s_ad to a valid logic level when its secondary bus grant is asserted. |
| s_cbe_l<3:0> | TS | Secondary PCI interface command/byte enables. These signals are a multiplexed command field and byte enable field. During the address phase or phases of a transaction, the initiator drives the transaction type on s_cbe_l<3:0>. When there are two address phases, the first address phase carries the dual address command and the second address phase carries the transaction type. For both read and write transactions, the initiator drives byte enables on s_cbe_l<3:0> during the data phases. When the secondary PCI bus is idle, the 21154 drives s_cbe_l to a valid logic level when its secondary bus grant is asserted. |
| s_par | TS | Secondary PCI interface parity. Signal s_par carries the even parity of the 36 bits of s_ad<31:0> and s_cbe_l<3:0> for both address and data phases. Signal s_par is driven by the same agent that has driven the address (for address parity) or the data (for data parity). Signal s_par contains valid parity one cycle after the address is valid (indicated by assertion of s_frame_l), or one cycle after data is valid (indicated by assertion of s_irdy_l for write transactions and s_trdy_l for read transactions). Signal s_par is driven by the device driving read or write data one cycle after s_ad is driven. Signal s_par is tristated one cycle after the s_ad lines are tristated. Devices receive data sample s_par as an input to check for possible parity errors. When the secondary PCI bus is idle, the 21154 drives s_par to a valid logic level when its secondary bus grant is asserted (one cycle after the s_ad bus is parked). |
| s_frame_l | STS | Secondary PCI interface FRAME#. Signal s_frame_l is driven by the initiator of a transaction to indicate the beginning and duration of an access on the secondary PCI bus. Signal s_frame_l assertion (falling edge) indicates the beginning of a PCI transaction. While s_frame_l remains asserted, data transfers can continue. The deassertion of s_frame_l indicates the final data phase requested by the initiator. When the secondary PCI bus is idle, s_frame_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| s_irdy_l | STS | Secondary PCI interface IRDY#. Signal s_irdy_l is driven by the initiator of a transaction to indicate the initiator's ability to complete the current data phase on the secondary PCI bus. During a write transaction, assertion of s_irdy_l indicates that valid write data is being driven on the s_ad bus. During a read transaction, assertion of s_irdy_l indicates that the initiator is able to accept read data for the current data phase. Once asserted during a given data phase, s_irdy_l is not deasserted until the data phase completes. When the secondary bus is idle, s_irdy_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |

**Table 6. Secondary PCI Bus Interface Signals (Sheet 2 of 2)**

| Signal Name | Type | Description |
|---|---|---|
| s_trdy_l | STS | Secondary PCI interface TRDY#. Signal s_trdy_l is driven by the target of a transaction to indicate the target's ability to complete the current data phase on the secondary PCI bus. During a write transaction, assertion of s_trdy_l indicates that the target is able to accept write data for the current data phase. During a read transaction, assertion of s_trdy_l indicates that the target is driving valid read data on the s_ad bus. Once asserted during a given data phase, s_trdy_l is not deasserted until the data phase completes. When the secondary bus is idle, s_trdy_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| s_devsel_l | STS | Secondary PCI interface DEVSEL#. Signal s_devsel_l is asserted by the target, indicating that the device is accepting the transaction. As a target, the 21154 performs positive decoding on the address of a transaction initiated on the secondary bus in order to determine whether to assert s_devsel_l. As an initiator of a transaction on the secondary bus, the 21154 looks for the assertion of s_devsel_l within five cycles of s_frame_l assertion; otherwise, the 21154 terminates the transaction with a master abort. When the secondary bus is idle, s_devsel_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| s_stop_l | STS | Secondary PCI interface STOP#. Signal s_stop_l is driven by the target of the current transaction, indicating that the target is requesting the initiator to stop the current transaction on the secondary bus.<br>When s_stop_l is asserted in conjunction with s_trdy_l and s_devsel_l assertion, a disconnect with data transfer is being signaled.<br>When s_stop_l and s_devsel_l are asserted, but s_trdy_l is deasserted, a target disconnect without data transfer is being signaled. When this occurs on the first data phase, that is, no data is transferred during the transaction, this is referred to as a target retry.<br>When s_stop_l is asserted and s_devsel_l is deasserted, the target is signaling a target abort.<br>When the secondary bus is idle, s_stop_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| s_lock_l | STS | Secondary PCI interface LOCK#. Signal s_lock_l is deasserted during the first address phase of a transaction and is asserted one clock cycle later by the 21154 when it is propagating a locked transaction downstream. The 21154 does not propagate locked transactions upstream. The 21154 continues to assert s_lock_l until the address phase of the next locked transaction, or until the lock is released. When the lock is released, s_lock_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| s_perr_l | STS | Secondary PCI interface PERR#. Signal s_perr_l is asserted when a data parity error is detected for data received on the secondary interface. The timing of s_perr_l corresponds to s_par driven one cycle earlier and s_ad driven two cycles earlier. Signal s_perr_l is asserted by the target during write transactions, and by the initiator during read transactions. When the secondary bus is idle, s_perr_l is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| s_serr_l | I | Secondary PCI interface SERR#. Signal s_serr_l can be driven low by any device except the 21154 on the secondary bus to indicate a system error condition. The 21154 samples s_serr_l as an input and conditionally forwards it to the primary bus on p_serr_l. The 21154 does not drive s_serr_l. Signal s_serr_l is pulled up through an external resistor. |

## 2.4 Secondary PCI Bus Interface 64-Bit Extension Signals

Table 7 describes the secondary PCI bus interface 64-bit extension signals.

**Table 7. Secondary PCI Bus Interface 64-Bit Extension Signals**

| Signal Name | Type | Description |
|---|---|---|
| s_ad<63:32> | TS | Secondary PCI interface address/data upper 32 bits. This multiplexed address and data bus provides an additional 32 bits to the secondary interface. During the address phase or phases of a transaction, when the Dual Address command is used and s_req64_l is asserted, the initiator drives the upper 32 bits of a 64-bit address; otherwise these bits are undefined and the initiator drives a valid logic level onto these pins. During the data phases of a transaction, the initiator drives the upper 32 bits of 64-bit write data, or the target drives the upper 32 bits of 64-bit read data, when s_req64_l and s_ack64_l are both asserted. When not driven, s_ad<63:32> are pulled up to a valid logic level through external resistors. |
| s_cbe_l<7:4> | TS | Secondary PCI interface command/byte enables upper 4 bits. These signals are a multiplexed command field and byte enable field. During the address phase or phases of a transaction, when the dual address command is used and s_req64_l is asserted, the initiator will drive the transaction type on s_cbe_l<7:4>; otherwise these bits are undefined and the initiator drives a valid logic level onto the pins. For both reads and write transactions, the initiator will drive byte enables for the s_ad<63:32> data bits on s_cbe_l<7:4> during the data phases when s_req64_l and s_ack64_l are both asserted. When not driven, s_cbe_l<7:4> is pulled up to a valid logic level through external resistors. |
| s_par64 | TS | Secondary PCI interface upper 32-bits parity. Signal s_par64 carries the even parity of the 36 bits of s_ad<63:32> and s_cbe_l<7:4> for both address and data phases. Signal s_par64 is driven by the initiator and is valid one cycle after the first address phase when a dual address command is used and s_req64_l is asserted. Signal s_par64 is also valid one clock cycle after the second address phase of a dual address transaction when s_req64_l is asserted. Signal s_par64_l is valid one cycle after valid data is driven (indicated by s_irdy_l assertion for write data and s_trdy_l assertion for read data) and both s_req64_l and s_ack64_l are asserted for that data phase. Signal s_par64 is driven by the device driving read or write data one cycle after the s_ad lines are driven. Signal s_par64 is tristated one cycle after the s_ad lines are tristated. Devices receive data sample s_par64 as an input in order to check for possible parity errors during 64-bit transactions. When not driven, s_par64 is pulled up to a valid logic level through external resistors. |
| s_req64_l | STS | Secondary PCI interface request 64-bit transfer. Signal s_req64_l is asserted by the initiator to indicate that the initiator is requesting a 64-bit data transfer. Signal s_req64_l has the same timing as s_frame_l. The 21154 asserts s_req64_l low during reset, indicating that a 64-bit PCI bus is supported on the board. When deasserting, s_req64_l is driven to a deasserted state for one cycle and then sustained by an external pull-up resistor. |
| s_ack64_l | STS | Secondary PCI interface acknowledge 64-bit transfer. Signal s_ack64_l is asserted by the target only when s_req64_l is asserted by the initiator, to indicate the target's ability to transfer data using 64 bits. Signal s_ack64_l has the same timing as s_devsel_l. When deasserting, s_ack64_l is driven to a deasserted state for one cycle and then sustained by an external pull-up resistor. |

## 2.5 Secondary Bus Arbitration Signals

Table 8 describes the secondary bus arbitration signals.

**Table 8. Secondary PCI Bus Arbitration Signals**

| Signal Name | Type | Description |
|---|---|---|
| s_req_l<8:0> | I | Secondary PCI interface REQ#s. The 21154 accepts nine request inputs, s_req_l<8:0>, into its secondary bus arbiter. The 21154 request input to the arbiter is an internal signal. Each request input can be programmed to be in either a high priority rotating group or a low priority rotating group. An asserted level on an s_req_l pin indicates that the corresponding master wants to initiate a transaction on the secondary PCI bus. If the internal arbiter is disabled (s_cfn_l tied high), s_req_l<0> is reconfigured to be an external secondary grant input for the 21154. In this case, an asserted level on s_req_l<0> indicates that the 21154 can start a transaction on the secondary PCI bus if the bus is idle. |
| s_gnt_l<8:0> | TS | Secondary PCI interface GNT#s. The 21154 secondary bus arbiter can assert one of nine secondary bus grant outputs, s_gnt_l<8:0>, to indicate that an initiator can start a transaction on the secondary bus if the bus is idle. The 21154's secondary bus grant is an internal signal. A programmable 2-level rotating priority algorithm is used. If the internal arbiter, s_cfn_l, is disabled (tied high), s_gnt_l<0> is reconfigured to be an external secondary bus request output for the 21154. The 21154 asserts this signal whenever it wants to start a transaction on the secondary bus. |
| s_cfn_l | I | Secondary PCI central function enable. When tied low, s_cfn_l enables the 21154 secondary bus arbiter. When tied high, s_cfn_l disables the internal arbiter. An external secondary bus arbiter must then be used. Signal s_req_l<0> is reconfigured to be the 21154 secondary bus grant input, and s_gnt_l<0> is reconfigured to be the 21154 secondary bus request output, when an external arbiter is used. Secondary bus parking is done when s_req_l<0> is asserted, the secondary bus is idle, and the 21154 does not want to initiate a transaction. |

## 2.6 General-Purpose I/O Interface Signals

Table 9 describes the general-purpose I/O interface signals.

**Table 9. General-Purpose I/O Interface Signals**

| Signal Name | Type | Description |
|---|---|---|
| gpio<3:0> | TS | General-purpose I/O data. These four general-purpose signals are programmable as either input-only or bidirectional signals by writing the gpio output enable control register in configuration space. The value on these signals is reflected in a gpio input data configuration register when read. Levels to be driven on gpio pins configured as bidirectional are derived from the value written in the gpio output data configuration register.<br>During the first 23 clock cycles (46 cycles when s_clk operates at 66 MHz) while p_rst_l is deasserted and s_rst_l is asserted, the gpio signals are used to control an external shift register that can shift in a serial clock disable mask into the msk_in input. The gpio pins should not be driven by software during these 23 clock cycles. The mask can then be read and modified in the secondary clock control register in configuration space. |

## 2.7 Clock Signals

Table 10 describes the clock signals.

**Table 10. Clock Signals**

| Signal Name | Type | Description |
|---|---|---|
| p_clk | I | Primary interface PCI CLK. Provides timing for all transactions on the primary PCI bus. All primary PCI inputs are sampled on the rising edge of p_clk, and all primary PCI outputs are driven from the rising edge of p_clk. Frequencies supported by the 21154 range from 0 MHz to 33 MHz, or 0 MHz to 66 MHz for a 66 MHz capable 21154. |
| s_clk | I | Secondary interface PCI CLK. Provides timing for all transactions on the secondary PCI bus. All secondary PCI inputs are sampled on the rising edge of s_clk, and all secondary PCI outputs are driven from the rising edge of s_clk. Frequencies supported by the 21154 range from 0 MHz to 33 MHz, or 0 MHz to 66 MHz for a 66 MHz capable 21154 . |
| s_clk_o<9:0> | O | Secondary interface PCI CLK outputs. Signals s_clk_o<9:0> are 10 clock outputs generated from the primary interface clock input, p_clk. These clocks operate at the same frequency of p_clk, or at half the frequency when the primary bus frequency is 66 MHz and the secondary bus frequency is 33 MHz.<br>When these clocks are used, one of the clock outputs must be fed back to the secondary clock input, s_clk. Unused clock outputs can be disabled either by using the serial disable mask mechanism (using the gpio pins and msk_in), or by writing the secondary clock disable bits in configuration space; otherwise, terminate them electrically. |

## 2.8 Reset Signals

Table 11 describes the reset signals.

**Table 11. Reset Signals (Sheet 1 of 2)**

| Signal Name | Type | Description |
|---|---|---|
| bpcce[1] | I | Bus/power clock control management pin. When signal bpcce is tied high and the 21154 is placed in the $D3_{hot}$ power state, the 21154 places the secondary bus in the B2 power state. The 21154 disables the secondary clocks and drives them to 0. When tied low, placing the 21154 in the $D3_{hot}$ power state has no effect on the secondary bus clocks. |
| p_rst_l | I | Primary PCI bus RST#. Signal p_rst_l forces the 21154 to a known state. All register state is cleared, and all primary PCI bus outputs are tristated. The 21154 samples p_req64_l during p_rst_l assertion to determine whether the 64-bit extension is supported on the board. Signal p_rst_l is asynchronous to p_clk. |

**Table 11. Reset Signals (Sheet 2 of 2)**

| Signal Name | Type | Description |
|---|---|---|
| s_rst_l | O | Secondary PCI bus RST#. Signal s_rst_l is driven by the 21154 and acts as the PCI reset for the secondary bus. The 21154 asserts s_rst_l when any of the following conditions is met:<br>• Signal p_rst_l is asserted.<br>• The secondary reset bit in the bridge control register in configuration space is set.<br>• The chip reset bit in the diagnostic control register in configuration space is set.<br>When the 21154 asserts s_rst_l, it tristates all secondary control signals and drives zeros on s_ad, s_cbe_l, s_par, and s_par64. The 21154 also asserts s_req64_l during reset, indicating that a 64-bit bus is supported on the secondary interface. Signal s_rst_l remains asserted until p_rst_l is deasserted, the gpio serial clock mask has been shifted in, and the secondary reset bit is clear. Assertion of s_rst_l by itself does not clear register state, and configuration registers are still accessible from the primary PCI interface. |

[1.] For 21154–AB and later revisions only.

## 2.9 Miscellaneous Signals

Table 12 describes the miscellaneous signals.

**Table 12. Miscellaneous Signals (Sheet 1 of 2)**

| Signal Name | Type | Description |
|---|---|---|
| msk_in | I | Secondary clock disable serial input. This input-only signal is used by the hardware mechanism to disable secondary clock outputs. The serial stream is received by msk_in, starting when p_rst is detected deasserted and s_rst_l is detected asserted. This serial data is used for selectively disabling secondary clock outputs and is shifted into the secondary clock control configuration register. This input can be tied low to enable all secondary clock outputs, or tied high to drive all secondary clock outputs high. |
| p_vio | I | Primary interface I/O voltage. This signal must be tied to either 3.3 V or 5 V, corresponding to the signaling environment of the primary PCI bus as described in the *PCI Local Bus Specification*, *Revision 2.1*. When any device on the primary PCI bus uses 5-V signaling levels, tie p_vio to 5 V. Signal p_vio is tied to 3.3 V only when all the devices on the primary bus use 3.3-V signaling levels. |
| s_vio | I | Secondary interface I/O voltage. This signal must be tied to either 3.3 V or 5 V, corresponding to the signaling environment of the secondary PCI bus as described in the *PCI Local Bus Specification, Revision 2.1*. When any device on the secondary PCI bus uses 5-V signaling levels, tie s_vio to 5 V. Signal s_vio is tied to 3.3 V only when all the devices on the secondary bus use 3.3-V signaling levels. |

**Table 12. Miscellaneous Signals (Sheet 2 of 2)**

| Signal Name | Type | Description |
|---|---|---|
| config66 | I | Configure 66 MHz operation. This input only pin is used to specify if the 21154 is capable of running at 66 MHz. If the pin is tied high, then the device can be run at 66 MHz. If the pin is tied low, then the 21154 can only function under the 33 MHz PCI specification. |
| p_m66ena | I | Primary interface 66 MHz enable. This input-only signal pin is used to designate the primary interface bus speed. This signal should be pulled low for 33 MHz operation on the primary bus. In this case, the s_m66ena pin will be driven low, forcing the secondary interface to also run at 33 MHz. For 66 MHz operation on the primary bus, this signal should be pulled high. |
| s_m66ena | I/OD | Secondary interface 66 MHz enable. This signal pin is used to designate the secondary interface bus speed. If the primary bus is operating at 33 MHz (i.e. if p_m66ena is low), then the s_m66ena pin will be driven low by the 21154 forcing the secondary bus to operate at 33 MHz. If the primary bus is operating at 66 MHz, then the s_m66ena pin is an input and should be externally pulled high for the secondary bus to operate at 66 MHz or low for the secondary bus to operate at 33 MHz. |

## 2.10 JTAG Signals

Table 13 describes the JTAG signals.

**Table 13. JTAG Signals**

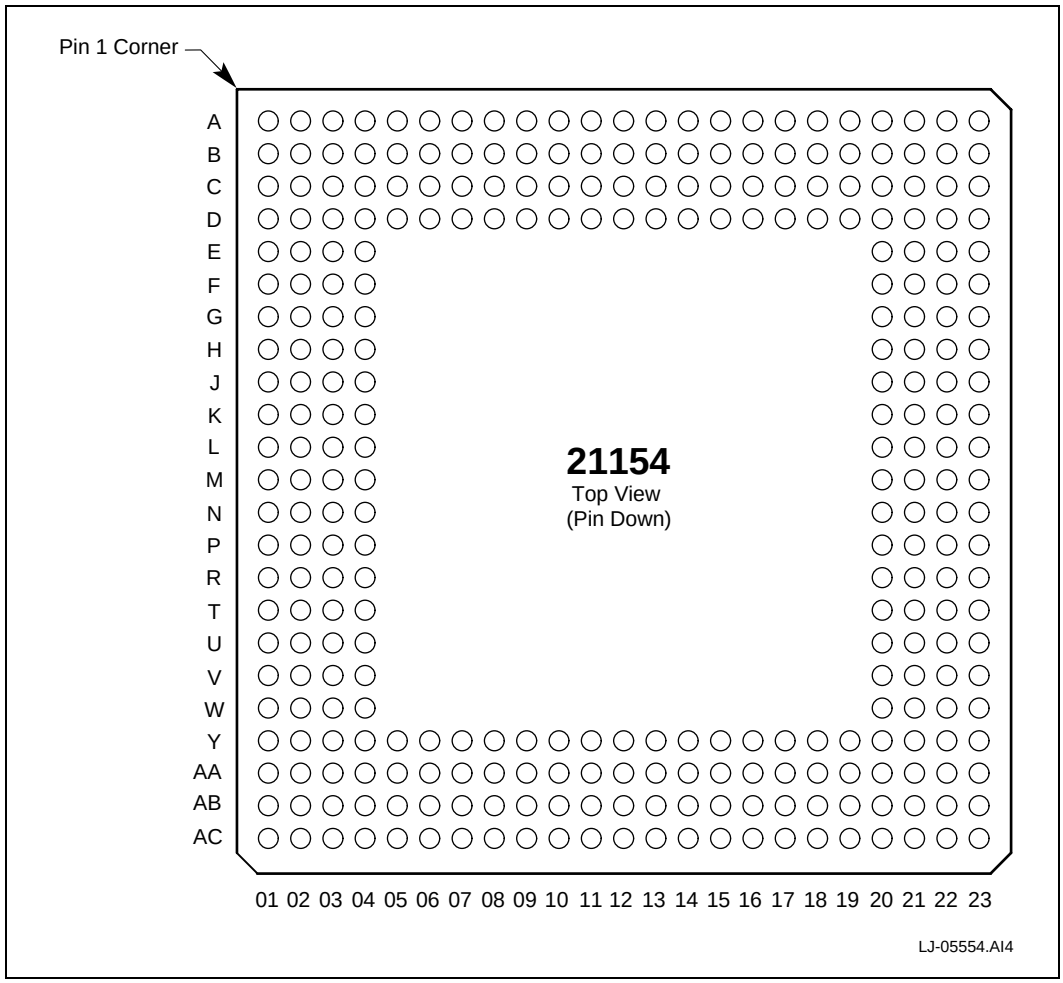
| Signal Name | Type | Description |
|---|---|---|
| tdi | I | JTAG serial data in. Signal tdi is the serial input through which JTAG instructions and test data enter the JTAG interface. The new data on tdi is sampled on the rising edge of tck. An unterminated tdi produces the same result as if tdi were driven high. |
| tdo | O | JTAG serial data out. Signal tdo is the serial output through which test instructions and data from the test logic leave the 21154. |
| tms | I | JTAG test mode select. Signal tms causes state transitions in the test access port (TAP) controller. An undriven tms has the same result as if it were driven high. |
| tck | I | JTAG boundary-scan clock. Signal tck is the clock controlling the JTAG logic. |
| trst_l | I | JTAG TAP reset. When asserted low, the TAP controller is asynchronously forced to enter a reset state, which in turn asynchronously initializes other test logic. An unterminated trst_l produces the same result as if it were driven high. The TAP controller must be reset before the chip can function in normal operating mode. |

# 3.0 Pin Assignment

This chapter describes the 21154 pin assignment and lists the pins according to location and in alphabetic order.

Figure 5 shows the 21154 304-point ball grid array, representing the pins in vertical rows labeled alphabetically, and horizontal rows labeled numerically. Table 14 and Table 15 use this location to identify pin assignments.

**Figure 5. 21154 PBGA Cavity Down View**



## 3.1 Numeric Pin Assignment

Table 14 lists the 21154 pins in order of location, showing the location code, name, and signal type of each pin.

Figure 5 provides the map for identifying the pin location codes, listed in alphabetic order in the PBGA Location column in Table 14.

The following table defines the signal type abbreviations used in the Type column in Table 14.

| Signal Type | Description |
|---|---|
| I | Standard input only. |
| O | Standard output only. |
| P | Power. |
| TS | Tristate bidirectional. |
| STS | Sustained tristate. Active low signal must be pulled high for one cycle when deasserting. |
| OD | Standard open drain. |

**Table 14. 21154 PBGA Pin List (Sheet 1 of 6)**

| PBGA Location | Pin Name | Type | PBGA Location | Pin Name | Type |
|---|---|---|---|---|---|
| A1 | vss | P | A2 | vdd | P |
| A3 | s_ad<30> | TS | A4 | s_ad<27> | TS |
| A5 | vss | P | A6 | s_ad<23> | TS |
| A7 | s_ad<22> | TS | A8 | s_ad<19> | TS |
| A9 | s_ad<16> | TS | A10 | s_trdy_l | STS |
| A11 | s_lock_l | STS | A12 | vss | P |
| A13 | s_ad<13> | TS | A14 | s_m66ena | OD |
| A15 | s_cbe_l<0> | TS | A16 | vss | P |
| A17 | s_ad<2> | TS | A18 | s_ad<0> | TS |
| A19 | s_cbe_l<7> | TS | A20 | s_cbe_l<5> | TS |
| A21 | s_ad<62> | TS | A22 | vdd | P |
| A23 | vss | P | — | — | — |
| B1 | vdd | P | B2 | vss | P |
| B3 | s_ad<29> | TS | B4 | s_ad<26> | TS |
| B5 | s_ad<24> | TS | B6 | vddvdd | P |
| B7 | s_ad<20> | TS | B8 | s_ad<18> | TS |
| B9 | s_frame_l | STS | B10 | s_devsel_l | STS |
| B11 | s_serr_l | I | B12 | s_par | TS |
| B13 | s_ad<14> | TS | B14 | s_ad<10> | TS |
| B15 | s_ad<8> | TS | B16 | s_ad<6> | TS |

**Table 14. 21154 PBGA Pin List (Sheet 2 of 6)**

| PBGA Location | Pin Name | Type | PBGA Location | Pin Name | Type |
|---|---|---|---|---|---|
| B17 | s_ad<4> | TS | B18 | s_ad<1> | TS |
| B19 | s_req64_l | STS | B20 | vdd | P |
| B21 | vss | P | B22 | vss | P |
| B23 | vdd | P | — | — | — |
| C1 | s_req_l<1> | I | C2 | s_req_l<2> | I |
| C3 | s_ad<31> | TS | C4 | s_ad<28> | TS |
| C5 | s_ad<25> | TS | C6 | s_cbe_l<3> | TS |
| C7 | vss | P | C8 | s_ad<17> | TS |
| C9 | s_irdy_l | STS | C10 | s_stop_l | STS |
| C11 | s_perr | STS | C12 | s_cbe_l<1> | TS |
| C13 | s_ad<15> | TS | C14 | s_ad<11> | TS |
| C15 | s_ad<9> | TS | C16 | s_ad<7> | TS |
| C17 | s_ad<5> | TS | C18 | s_ack64_l | STS |
| C19 | s_cbe_l<6> | TS | C20 | s_ad<63> | TS |
| C21 | s_ad<60> | TS | C22 | s_ad<58> | TS |
| C23 | s_ad<59> | TS | — | — | — |
| D1 | s_req_l<5> | I | D2 | s_req_l<6> | I |
| D3 | s_req_l<3> | I | D4 | s_req_l<0> | I |
| D5 | vdd | P | D6 | vdd | P |
| D7 | s_ad<21> | TS | D8 | vss | P |
| D9 | s_cbe_l<2> | TS | D10 | vdd | P |
| D11 | vdd | P | D12 | vss | P |
| D13 | s_ad<12> | TS | D14 | vdd | P |
| D15 | vdd | P | D16 | vss | P |
| D17 | s_ad<3> | TS | D18 | vdd | P |
| D19 | s_cbe_l<4> | TS | D20 | s_ad<61> | TS |
| D21 | s_ad<57> | TS | D22 | s_ad<55> | TS |
| D23 | vss | P | — | — | — |
| E1 | s_req_l<8> | I | E2 | s_gnt_l<0> | TS |
| E3 | s_req_l<7> | I | E4 | s_req_l<4> | I |
| — | — | — | E20 | s_ad<56> | TS |
| E21 | s_ad<54> | TS | E22 | vdd | P |
| E23 | s_ad<53> | TS | — | — | — |
| F1 | s_gnt_l<2> | TS | F2 | s_gnt_l<3> | TS |
| F3 | s_gnt_l<1> | TS | F4 | vss | P |
| — | — | — | F20 | vss | P |
| F21 | s_ad<52> | TS | F22 | s_ad<50> | TS |

**Table 14. 21154 PBGA Pin List (Sheet 3 of 6)**

| PBGA Location | Pin Name | Type | PBGA Location | Pin Name | Type |
|---|---|---|---|---|---|
| F23 | s_ad<51> | TS | — | — | — |
| G1 | s_gnt_l<4> | TS | G2 | s_gnt_l<6> | TS |
| G3 | s_gnt_l<7> | TS | G4 | s_gnt_l<5> | TS |
| — | | | G20 | s_ad<49> | TS |
| G21 | s_ad<47> | TS | G22 | s_ad<48> | TS |
| G23 | vss | P | — | — | — |
| H1 | s_gnt_l<8> | TS | H2 | s_rst_l | O |
| H3 | vss | P | H4 | vdd | P |
| — | — | — | H20 | vdd | P |
| H21 | s_ad<44> | TS | H22 | s_ad<45> | TS |
| H23 | s_ad<46> | TS | — | — | — |
| J1 | vdd | P | J2 | vss | P |
| J3 | vdd | P | J4 | s_clk | I |
| — | — | — | J20 | s_ad<42> | TS |
| J21 | vdd | P | J22 | s_ad<41> | TS |
| J23 | s_ad<43> | TS | — | — | — |
| K1 | s_cfn_l | I | K2 | gpio<3> | TS |
| K3 | gpio<2> | TS | K4 | vss | P |
| — | — | — | K20 | vss | P |
| K21 | s_ad<38> | TS | K22 | s_ad<39> | TS |
| K23 | s_ad<40> | TS | — | — | — |
| L1 | gpio<0> | TS | L2 | s_clk_o<0> | O |
| L3 | s_clk_o<1> | O | L4 | gpio<1> | TS |
| — | — | — | L20 | vss | P |
| L21 | s_ad<36> | TS | L22 | s_ad<35> | TS |
| L23 | s_ad<37> | TS | — | — | — |
| M1 | s_clk_o<3> | O | M2 | s_clk_o<4> | O |
| M3 | s_clk_o<2> | O | M4 | vdd | P |
| — | — | — | M20 | vdd | P |
| M21 | s_ad<32> | TS | M22 | s_ad<34> | TS |
| M23 | s_ad<33> | TS | — | — | — |
| N1 | s_clk_o<6> | O | N2 | vss | P |
| N3 | s_clk_o<5> | O | N4 | vdd | P |
| — | — | — | N20 | tck | I |
| N21 | s_par64 | TS | N22 | s_vio | I |
| N23 | trst_l | I | — | — | — |
| P1 | s_clk_o<9> | O | P2 | s_clk_o<8> | O |

**Table 14. 21154 PBGA Pin List (Sheet 4 of 6)**

| PBGA Location | Pin Name | Type | PBGA Location | Pin Name | Type |
|---|---|---|---|---|---|
| P3 | s_clk_o<7> | O | P4 | vss | P |
| — | — | — | P20 | vss | P |
| P21 | tms | I | P22 | tdo | O |
| P23 | tdi | I | — | — | — |
| R1 | vdd | P | R2 | p_gnt_l | I |
| R3 | p_rst_l | I | R4 | bpcce[1] | I |
| — | — | — | R20 | p_vio | I |
| R21 | msk_in | I | R22 | config66 | I |
| R23 | vdd | P | — | — | — |
| T1 | vdd | P | T2 | vss | P |
| T3 | p_clk | I | T4 | vdd | P |
| — | — | — | T20 | vdd | P |
| T21 | p_par64 | TS | T22 | p_ad<32> | TS |
| T23 | p_ad<33> | TS | — | — | — |
| U1 | p_ad<29> | TS | U2 | p_ad<31> | TS |
| U3 | p_req_l | TS | U4 | p_ad<30> | TS |
| — | — | — | U20 | p_ad<35> | TS |
| U21 | vss | P | U22 | p_ad<34> | TS |
| U23 | p_ad<36> | TS | — | — | — |
| V1 | p_ad<27> | TS | V2 | p_ad<28> | TS |
| V3 | p_ad<26> | TS | V4 | vss | P |
| — | — | — | V20 | vss | P |
| V21 | p_ad<39> | TS | V22 | p_ad<37> | TS |
| V23 | p_ad<38> | TS | — | — | — |
| W1 | p_ad<24> | TS | W2 | p_ad<25> | TS |
| W3 | vdd | P | W4 | p_ad<23> | TS |
| — | — | — | W20 | p_ad<44> | TS |
| W21 | p_ad<42> | TS | W22 | p_ad<40> | TS |
| W23 | p_ad<41> | TS | — | — | — |
| Y1 | p_idsel | I | Y2 | p_cbe_l<3> | TS |
| Y3 | p_ad<22> | TS | Y4 | p_ad<19> | TS |
| Y5 | p_ad<16> | TS | Y6 | vdd | P |
| Y7 | p_serr_l | OD | Y8 | vss | P |
| Y9 | vss | P | Y10 | vdd | P |
| Y11 | p_ad<8> | TS | Y12 | vss | P |
| Y13 | p_ad<1> | TS | Y14 | vdd | P |
| Y15 | p_cbe_l<5> | TS | Y16 | vss | P |

**Table 14. 21154 PBGA Pin List (Sheet 5 of 6)**

| PBGA Location | Pin Name | Type | PBGA Location | Pin Name | Type |
|---|---|---|---|---|---|
| Y17 | p_ad<59> | TS | Y18 | vdd | P |
| Y19 | p_ad<52> | TS | Y20 | p_ad<47> | TS |
| Y21 | p_ad<45> | TS | Y22 | vdd | P |
| Y23 | p_ad<43> | TS | — | — | — |
| AA1 | p_ad<21> | TS | AA2 | vss | P |
| AA3 | p_ad<20> | TS | AA4 | p_ad<17> | TS |
| AA5 | p_frame_l | STS | AA6 | p_devsel_l | STS |
| AA7 | p_cbe_l<1> | TS | AA8 | p_ad<14> | TS |
| AA9 | p_ad<11> | TS | AA10 | p_ad<9> | TS |
| AA11 | p_ad<6> | TS | AA12 | p_ad<5> | TS |
| AA13 | p_ad<2> | TS | AA14 | p_ad<0> | TS |
| AA15 | p_cbe_l<7> | TS | AA16 | p_ad<63> | TS |
| AA17 | p_ad<61> | TS | AA18 | p_ad<56> | TS |
| AA19 | p_ad<54> | TS | AA20 | p_ad<51> | TS |
| AA21 | p_ad<48> | TS | AA22 | vss | P |
| AA23 | p_ad<46> | TS | — | — | — |
| AB1 | vdd | P | AB2 | vss | P |
| AB3 | p_ad<18> | TS | AB4 | p_cbe_l<2> | TS |
| AB5 | p_trdy_l | STS | AB6 | p_lock_l | I |
| AB7 | p_par | TS | AB8 | p_ad<15> | TS |
| AB9 | p_ad<12> | TS | AB10 | p_m66ena | I |
| AB11 | p_ad<7> | TS | AB12 | p_ad<4> | TS |
| AB13 | p_ad<3> | TS | AB14 | p_ack64_l | STS |
| AB15 | p_cbe_l<6> | TS | AB16 | p_ad<62> | TS |
| AB17 | p_ad<60> | TS | AB18 | p_ad<58> | TS |
| AB19 | vdd | P | AB20 | p_ad<53> | TS |
| AB21 | p_ad<50> | TS | AB22 | vss | P |
| AB23 | vdd | P | — | — | — |
| AC1 | vss | P | AC2 | vdd | P |
| AC3 | vdd | P | AC4 | vss | P |
| AC5 | p_irdy_l | STS | AC6 | p_stop_l | STS |
| AC7 | p_perr_l | STS | AC8 | vdd | P |
| AC9 | p_ad<13> | TS | AC10 | p_ad<10> | TS |
| AC11 | p_cbe_l<0> | TS | AC12 | vdd | P |
| AC13 | vss | P | AC14 | p_req64_l | STS |
| AC15 | p_cbe_l<4> | TS | AC16 | vdd | P |
| AC17 | vss | P | AC18 | p_ad<57> | TS |

**Table 14. 21154 PBGA Pin List (Sheet 6 of 6)**

| PBGA Location | Pin Name | Type | PBGA Location | Pin Name | Type |
|---|---|---|---|---|---|
| AC19 | p_ad<55> | TS | AC20 | vss | P |
| AC21 | p_ad<49> | TS | AC22 | vdd | P |
| AC23 | vss | P | — | — | — |

1. Pertains to the 21154–AB and later revisions only. For the 21154–AA, this pin is vss.

## 3.2 Pins Listed in Alphabetic Order

Table 15 lists the 21154 pins in alphabetic order, showing the name, location code, and signal type of each pin.

Figure 5 provides the map for identifying the pin location codes.

The following table defines the signal type abbreviations used in the Type column in Table 15.

| Signal Type | Description |
|---|---|
| I | Standard input only. |
| O | Standard output only. |
| P | Power. |
| TS | Tristate bidirectional. |
| STS | Sustained tristate. Active low signal must be pulled high for one cycle when deasserting. |
| OD | Standard open drain. |

**Table 15. 21154 PBGA Pin List (Sheet 1 of 5)**

| Pin Name | PBGA Location | Type | Pin Name | PBGA Location | Type |
|---|---|---|---|---|---|
| bpcce[1] | R4 | I | p_ad<4> | AB12 | TS |
| config66 | R22 | I | p_ad<5> | AA12 | TS |
| gpio<0> | L1 | TS | p_ad<6> | AA11 | TS |
| gpio<1> | L4 | TS | p_ad<7> | AB11 | TS |
| gpio<2> | K3 | TS | p_ad<8> | Y11 | TS |
| gpio<3> | K2 | TS | p_ad<9> | AA10 | TS |
| msk_in | R21 | I | p_ad<10> | AC10 | TS |
| p_ack64_l | AB14 | STS | p_ad<11> | AA9 | TS |
| p_ad<0> | AA14 | TS | p_ad<12> | AB9 | TS |
| p_ad<1> | Y13 | TS | p_ad<13> | AC9 | TS |
| p_ad<2> | AA13 | TS | p_ad<14> | AA8 | TS |
| p_ad<3> | AB13 | TS | p_ad<15> | AB8 | TS |

**Table 15. 21154 PBGA Pin List (Sheet 2 of 5)**

| Pin Name | PBGA Location | Type | Pin Name | PBGA Location | Type |
|---|---|---|---|---|---|
| p_ad<16> | Y5 | TS | p_ad<53> | AB20 | TS |
| p_ad<17> | AA4 | TS | p_ad<54> | AA19 | TS |
| p_ad<18> | AB3 | TS | p_ad<55> | AC19 | TS |
| p_ad<19> | Y4 | TS | p_ad<56> | AA18 | TS |
| p_ad<20> | AA3 | TS | p_ad<57> | AC18 | TS |
| p_ad<21> | AA1 | TS | p_ad<58> | AB18 | TS |
| p_ad<22> | Y3 | TS | p_ad<59> | Y17 | TS |
| p_ad<23> | W4 | TS | p_ad<60> | AB17 | TS |
| p_ad<24> | W1 | TS | p_ad<61> | AA17 | TS |
| p_ad<25> | W2 | TS | p_ad<62> | AB16 | TS |
| p_ad<26> | V3 | TS | p_ad<63> | AA16 | TS |
| p_ad<27> | V1 | TS | p_cbe_l<0> | AC11 | TS |
| p_ad<28> | V2 | TS | p_cbe_l<1> | AA7 | TS |
| p_ad<29> | U1 | TS | p_cbe_l<2> | AB4 | TS |
| p_ad<30> | U4 | TS | p_cbe_l<3> | Y2 | TS |
| p_ad<31> | U2 | TS | p_cbe_l<4> | AC15 | TS |
| p_ad<32> | T22 | TS | p_cbe_l<5> | Y15 | TS |
| p_ad<33> | T23 | TS | p_cbe_l<6> | AB15 | TS |
| p_ad<34> | U22 | TS | p_cbe_l<7> | AA15 | TS |
| p_ad<35> | U20 | TS | p_clk | T3 | I |
| p_ad<36> | U23 | TS | p_devsel_l | AA6 | STS |
| p_ad<37> | V22 | TS | p_frame_l | AA5 | STS |
| p_ad<38> | V23 | TS | p_gnt_l | R2 | I |
| p_ad<39> | V21 | TS | p_idsel | Y1 | I |
| p_ad<40> | W22 | TS | p_irdy_l | AC5 | STS |
| p_ad<41> | W23 | TS | p_lock_l | AB6 | STS |
| p_ad<42> | W21 | TS | p_m66ena | AB10 | I |
| p_ad<43> | Y23 | TS | p_par | AB7 | TS |
| p_ad<44> | W20 | TS | p_par64 | T21 | TS |
| p_ad<45> | Y21 | TS | p_perr_l | AC7 | STS |
| p_ad<46> | AA23 | TS | p_req_l | U3 | TS |
| p_ad<47> | Y20 | TS | p_req64_l | AC14 | STS |
| p_ad<48> | AA21 | TS | p_rst_l | R3 | I |
| p_ad<49> | AC21 | TS | p_serr_l | Y7 | OD |
| p_ad<50> | AB21 | TS | p_stop_l | AC6 | STS |
| p_ad<51> | AA20 | TS | p_trdy_l | AB5 | STS |
| p_ad<52> | Y19 | TS | p_vio | R20 | I |

**Table 15. 21154 PBGA Pin List (Sheet 3 of 5)**

| Pin Name | PBGA Location | Type | Pin Name | PBGA Location | Type |
|---|---|---|---|---|---|
| s_ack64_l | C18 | STS | s_ad<36> | L21 | TS |
| s_ad<0> | A18 | TS | s_ad<37> | L23 | TS |
| s_ad<1> | B18 | TS | s_ad<38> | K21 | TS |
| s_ad<2> | A17 | TS | s_ad<39> | K22 | TS |
| s_ad<3> | D17 | TS | s_ad<40> | K23 | TS |
| s_ad<4> | B17 | TS | s_ad<41> | J22 | TS |
| s_ad<5> | C17 | TS | s_ad<42> | J20 | TS |
| s_ad<6> | B16 | TS | s_ad<43> | J23 | TS |
| s_ad<7> | C16 | TS | s_ad<44> | H21 | TS |
| s_ad<8> | B15 | TS | s_ad<45> | H22 | TS |
| s_ad<9> | C15 | TS | s_ad<46> | H23 | TS |
| s_ad<10> | B14 | TS | s_ad<47> | G21 | TS |
| s_ad<11> | C14 | TS | s_ad<48> | G22 | TS |
| s_ad<12> | D13 | TS | s_ad<49> | G20 | TS |
| s_ad<13> | A13 | TS | s_ad<50> | F22 | TS |
| s_ad<14> | B13 | TS | s_ad<51> | F23 | TS |
| s_ad<15> | C13 | TS | s_ad<52> | F21 | TS |
| s_ad<16> | A9 | TS | s_ad<53> | E23 | TS |
| s_ad<17> | C8 | TS | s_ad<54> | E21 | TS |
| s_ad<18> | B8 | TS | s_ad<55> | D22 | TS |
| s_ad<19> | A8 | TS | s_ad<56> | E20 | TS |
| s_ad<20> | B7 | TS | s_ad<57> | D21 | TS |
| s_ad<21> | D7 | TS | s_ad<58> | C22 | TS |
| s_ad<22> | A7 | TS | s_ad<59> | C23 | TS |
| s_ad<23> | A6 | TS | s_ad<60> | C21 | TS |
| s_ad<24> | B5 | TS | s_ad<61> | D20 | TS |
| s_ad<25> | C5 | TS | s_ad<62> | A21 | TS |
| s_ad<26> | B4 | TS | s_ad<63> | C20 | TS |
| s_ad<27> | A4 | TS | s_cbe_l<0> | A15 | TS |
| s_ad<28> | C4 | TS | s_cbe_l<1> | C12 | TS |
| s_ad<29> | B3 | TS | s_cbe_l<2> | D9 | TS |
| s_ad<30> | A3 | TS | s_cbe_l<3> | C6 | TS |
| s_ad<31> | C3 | TS | s_cbe_l<4> | D19 | TS |
| s_ad<32> | M21 | TS | s_cbe_l<5> | A20 | TS |
| s_ad<33> | M23 | TS | s_cbe_l<6> | C19 | TS |
| s_ad<34> | M22 | TS | s_cbe_l<7> | A19 | TS |
| s_ad<35> | L22 | TS | s_cfn_l | K1 | I |

**Table 15. 21154 PBGA Pin List (Sheet 4 of 5)**

| Pin Name | PBGA Location | Type | Pin Name | PBGA Location | Type |
|---|---|---|---|---|---|
| s_clk | J4 | I | s_req64_l | B19 | STS |
| s_clk_o<0> | L2 | O | s_rst_l | H2 | O |
| s_clk_o<1> | L3 | O | s_serr_l | B11 | I |
| s_clk_o<2> | M3 | O | s_stop_l | C10 | STS |
| s_clk_o<3> | M1 | O | s_trdy_l | A10 | STS |
| s_clk_o<4> | M2 | O | s_vio | N22 | I |
| s_clk_o<5> | N3 | O | tck | N20 | I |
| s_clk_o<6> | N1 | O | tdi | P23 | I |
| s_clk_o<7> | P3 | O | tdo | P22 | O |
| s_clk_o<8> | P2 | O | tms | P21 | I |
| s_clk_o<9> | P1 | O | trst_l | N23 | I |
| s_devsel_l | B10 | STS | vdd | A2 | P |
| s_frame_l | B9 | STS | vdd | A22 | P |
| s_gnt_l<0> | E2 | TS | vdd | B1 | P |
| s_gnt_l<1> | F3 | TS | vdd | B6 | P |
| s_gnt_l<2> | F1 | TS | vdd | B20 | P |
| s_gnt_l<3> | F2 | TS | vdd | B23 | P |
| s_gnt_l<4> | G1 | TS | vdd | D5 | P |
| s_gnt_l<5> | G4 | TS | vdd | D6 | P |
| s_gnt_l<6> | G2 | TS | vdd | D10 | P |
| s_gnt_l<7> | G3 | TS | vdd | D11 | P |
| s_gnt_l<8> | H1 | TS | vdd | D14 | P |
| s_irdy_l | C9 | STS | vdd | D15 | P |
| s_lock_l | A11 | STS | vdd | D18 | P |
| s_m66ena | A14 | OD | vdd | E22 | P |
| s_par | B12 | TS | vdd | H4 | P |
| s_par64 | N21 | TS | vdd | H20 | P |
| s_perr_l | C11 | STS | vdd | J1 | P |
| s_req_l<0> | D4 | I | vdd | J3 | P |
| s_req_l<1> | C1 | I | vdd | J21 | P |
| s_req_l<2> | C2 | I | vdd | M4 | P |
| s_req_l<3> | D3 | I | vdd | M20 | P |
| s_req_l<4> | E4 | I | vdd | N4 | P |
| s_req_l<5> | D1 | I | vdd | R1 | P |
| s_req_l<6> | D2 | I | vdd | R23 | P |
| s_req_l<7> | E3 | I | vdd | T1 | P |
| s_req_l<8> | E1 | I | vdd | T4 | P |

**Table 15. 21154 PBGA Pin List (Sheet 5 of 5)**

| Pin Name | PBGA Location | Type | Pin Name | PBGA Location | Type |
|---|---|---|---|---|---|
| vdd | T20 | P | vss | F4 | P |
| vdd | W3 | P | vss | F20 | P |
| vdd | Y6 | P | vss | G23 | P |
| vdd | Y10 | P | vss | H3 | P |
| vdd | Y14 | P | vss | J2 | P |
| vdd | Y18 | P | vss | K4 | P |
| vdd | Y22 | P | vss | K20 | P |
| vdd | AB1 | P | vss | L20 | P |
| vdd | AB19 | P | vss | N2 | P |
| vdd | AB23 | P | vss | P4 | P |
| vdd | AC2 | P | vss | P20 | P |
| vdd | AC3 | P | vss | T2 | P |
| vdd | AC8 | P | vss | U21 | P |
| vdd | AC12 | P | vss | V4 | P |
| vdd | AC16 | P | vss | V20 | P |
| vdd | AC22 | P | vss | Y8 | P |
| vss | A1 | P | vss | Y9 | P |
| vss | A5 | P | vss | Y12 | P |
| vss | A12 | P | vss | Y16 | P |
| vss | A16 | P | vss | AA2 | P |
| vss | A23 | P | vss | AA22 | P |
| vss | B2 | P | vss | AB2 | P |
| vss | B21 | P | vss | AB22 | P |
| vss | B22 | P | vss | AC1 | P |
| vss | C7 | P | vss | AC4 | P |
| vss | D8 | P | vss | AC13 | P |
| vss | D12 | P | vss | AC17 | P |
| vss | D16 | P | vss | AC20 | P |
| vss | D23 | P | vss | AC23 | P |

[1.] Pertains to the 21154–AB and later revisions only. For the 21154–AA, this pin is **vss**.

# 4.0 PCI Bus Operation

This chapter presents detailed information about PCI transactions, transaction forwarding across the 21154, and transaction termination. Section 4.1 through Section 4.7 describe 32-bit transaction operation. Section 4.8 describes special considerations for 64-bit transaction operation.

## 4.1 Types of Transactions

This section provides a summary of PCI transactions performed by the 21154. Table 16 lists the command code and name of each PCI transaction. The Master and Target columns indicate 21154 support for each transaction when the 21154 initiates transactions as a master, on the primary bus and on the secondary bus, and when the 21154 responds to transactions as a target, on the primary bus and on the secondary bus.

**Table 16. 21154 PCI Transactions**

| Type of Transaction | 21154 Initiates As Master | | 21154 Responds As Target | |
|---|---|---|---|---|
| | Primary | Secondary | Primary | Secondary |
| 0000 Interrupt acknowledge | No | No | No | No |
| 0001 Special cycle | Yes | Yes | No | No |
| 0010 I/O read | Yes | Yes | Yes | Yes |
| 0011 I/O write | Yes | Yes | Yes | Yes |
| 0100 Reserved | No | No | No | No |
| 0101 Reserved | No | No | No | No |
| 0110 Memory read | Yes | Yes | Yes | Yes |
| 0111 Memory write | Yes | Yes | Yes | Yes |
| 1000 Reserved | No | No | No | No |
| 1001 Reserved | No | No | No | No |
| 1010 Configuration read | No | Yes | Yes | No |
| 1011 Configuration write | Type 1 | Yes | Yes | Type 1 |
| 1100 Memory read multiple | Yes | Yes | Yes | Yes |
| 1101 Dual address cycle | Yes | Yes | Yes | Yes |
| 1110 Memory read line | Yes | Yes | Yes | Yes |
| 1111 Memory write and invalidate | Yes | Yes | Yes | Yes |

As indicated in Table 16, the following PCI commands are not supported by the 21154:

- The 21154 never initiates a PCI transaction with a reserved command code and, as a target, the 21154 ignores reserved command codes.
- The 21154 never initiates an interrupt acknowledge transaction and, as a target, the 21154 ignores interrupt acknowledge transactions. Interrupt acknowledge transactions are expected to reside entirely on the primary PCI bus closest to the host bridge.
- The 21154 does not respond to special cycle transactions. The 21154 cannot guarantee delivery of a special cycle transaction to downstream buses because of the broadcast nature of the special cycle command and the inability to control the transaction as a target. To generate

special cycle transactions on other PCI buses, either upstream or downstream, a Type 1 configuration command must be used.

- The 21154 does not generate Type 0 configuration transactions on the primary interface, nor does it respond to Type 0 configuration transactions on the secondary PCI interface. The *PCI-to-PCI Bridge Architecture Specification* does not support configuration from the secondary bus.

## 4.2 Address Phase

The standard PCI transaction consists of one or two address phases, followed by one or more data phases. An address phase always lasts one PCI clock cycle. The first address phase is designated by an asserting (falling) edge on the FRAME# signal.

The number of address phases depends on whether the address is 32 bits or 64 bits.

### 4.2.1 Single Address Phase

A 32-bit address uses a single address phase. This address is driven on AD<31:0>, and the bus command is driven on C/BE#<3:0>.

The 21154 supports the linear increment address mode only, which is indicated when the low 2 address bits are equal to 0. If either of the low 2 address bits is nonzero, the 21154 automatically disconnects the transaction after the first data transfer.

### 4.2.2 Dual Address Phase

Dual address transactions are PCI transactions that contain two address phases specifying a 64-bit address.

The first address phase is denoted by the asserting edge of FRAME#.

The second address phase always follows on the next clock cycle.

For a 32-bit interface, the first address phase contains the dual address command code on the C/BE#<3:0> lines, and the low 32 address bits on the AD<31:0> lines. The second address phase consists of the specific memory transaction command code on the C/BE#<3:0> lines, and the high 32 address bits on the AD<31:0> lines. In this way, 64-bit addressing can be supported on 32-bit PCI buses.

The *PCI-to-PCI Bridge Architecture Specification* supports the use of dual address transactions in the prefetchable memory range only. See Section 5.3.3 for a discussion of prefetchable address space. The 21154 supports dual address transactions in both the upstream and the downstream direction. The 21154 supports a programmable 64-bit address range in prefetchable memory for downstream forwarding of dual address transactions. Dual address transactions falling outside the prefetchable address range are forwarded upstream, but not downstream. Prefetching and posting are performed in a manner consistent with the guidelines given in this specification for each type of memory transaction in prefetchable memory space.

The 21154 responds only to dual address transactions that use the following transaction command codes:

- Memory write

- Memory write and invalidate
- Memory read
- Memory read line
- Memory read multiple

Use of other transaction codes may result in a master abort.

Any memory transactions addressing the first 4GB space should use a single address phase; that is, the high 32 bits of a dual address transaction should never be 0.

## 4.3 Device Select (DEVSEL#) Generation

The 21154 always performs positive address decoding when accepting transactions on either the primary or secondary buses. The 21154 never subtractively decodes. Medium DEVSEL# timing is used on both interfaces.

## 4.4 Data Phase

The address phase or phases of a PCI transaction are followed by one or more data phases. A data phase is completed when IRDY# and either TRDY# or STOP# are asserted. A transfer of data occurs only when both IRDY# and TRDY# are asserted during the same PCI clock cycle. The last data phase of a transaction is indicated when FRAME# is deasserted and both TRDY# and IRDY# are asserted, or when IRDY# and STOP# are asserted. See Section 4.10 for further discussion of transaction termination.

Depending on the command type, the 21154 can support multiple data phase PCI transactions. For a detailed description of how the 21154 imposes disconnect boundaries, see Section 4.5.4 for a description of write address boundaries and Section 4.6.3 for a description of read address boundaries.

## 4.5 Write Transactions

Write transactions are treated as either posted write or delayed write transactions.

Table 17 shows the method of forwarding used for each type of write operation.

**Table 17. Write Transaction Forwarding**

| Type of Transaction | Type of Forwarding |
|---|---|
| Memory write | Posted |
| Memory write and invalidate | Posted |
| I/O write | Delayed |
| Type 1 configuration write | Delayed |

### 4.5.1 Posted Write Transactions

Posted write forwarding is used for memory write and for memory write and invalidate transactions.

When the 21154 determines that a memory write transaction is to be forwarded across the bridge, the 21154 asserts DEVSEL# with medium timing and TRDY# in the same cycle, provided that enough buffer space is available in the posted data queue for the address and at least 8 Dwords of data. This enables the 21154 to accept write data without obtaining access to the target bus. The 21154 can accept 1 Dword of write data every PCI clock cycle; that is, no target wait states are inserted. This write data is stored in internal posted write buffers and is subsequently delivered to the target.

The 21154 continues to accept write data until one of the following events occurs:

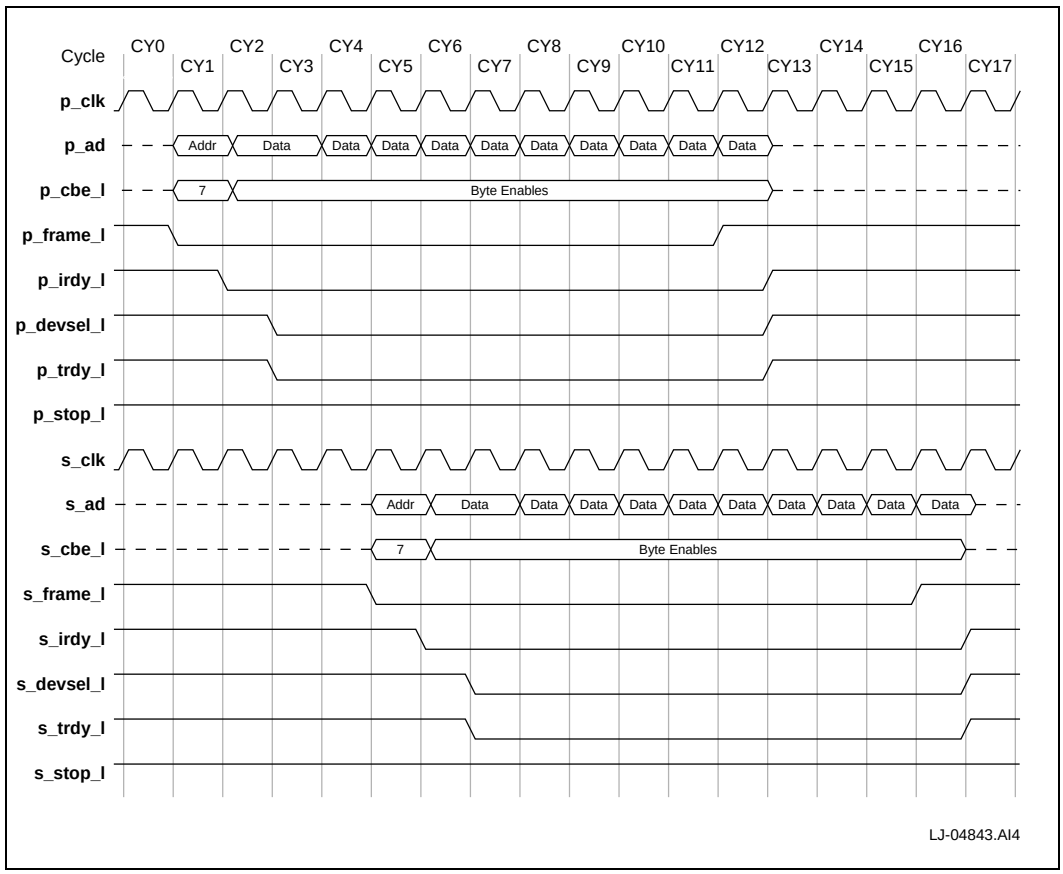
- The initiator terminates the transaction by deasserting FRAME# and IRDY#.
- An internal write address boundary is reached, such as a cache line boundary or an aligned 4KB boundary, depending on the transaction type.
- The posted write data buffer fills up.

When one of the last two events occurs, the 21154 returns a target disconnect to the requesting initiator on this data phase to terminate the transaction.

Once the posted write data moves to the head of the posted data queue, the 21154 asserts its request on the target bus. This can occur while the 21154 is still receiving data on the initiator bus. When the grant for the target bus is received and the target bus is detected in the idle condition, the 21154 asserts FRAME# and drives the stored write address out on the target bus. On the following cycle, the 21154 drives the first Dword of write data and continues to transfer write data until all write data corresponding to that transaction is delivered, or until a target termination is received. As long as write data exists in the queue, the 21154 can drive 1 Dword of write data each PCI clock cycle; that is, no master wait states are inserted. If write data is flowing through the 21154 and the initiator stalls, the 21154 may have to insert wait states on the target bus if the queue empties.

Figure 6 shows a memory write transaction in flow-through mode, where data is being removed from buffers on the target interface while more data is being transferred into the buffers on the master interface.

**Figure 6. Flow-Through Posted Memory Write Transaction**




The 21154 ends the transaction on the target bus when one of the following conditions is met:

- All posted write data has been delivered to the target.
- The target returns a target disconnect or target retry (the 21154 starts another transaction to deliver the rest of the write data).
- The target returns a target abort (the 21154 discards remaining write data).
- The master latency timer expires, and the 21154 no longer has the target bus grant (the 21154 starts another transaction to deliver remaining write data).

Section 4.10.3.2 provides detailed information about how the 21154 responds to target termination during posted write transactions.

## 4.5.2 Memory Write and Invalidate Transactions

Posted write forwarding is used for memory write and invalidate transactions.

Memory write and invalidate transactions guarantee transfer of entire cache lines. If the write buffer fills before an entire cache line is transferred, the 21154 disconnects the transaction and converts it to a memory write transaction.

The 21154 disconnects memory write and invalidate commands at aligned cache line boundaries. The cache line size value in the 21154 cache line size register gives the number of Dwords in a cache line. For the 21154 to generate memory write and invalidate transactions, this cache line size value must be written to a value that is a nonzero power of 2 and less than or equal to 16 (that is, 1, 2, 4, 8, or 16 Dwords).

If the cache line size does not meet the memory write and invalidate conditions, that is, the value is 0, or is not a power of 2, or is greater than 16 Dwords, the 21154 treats the memory write and invalidate command as a memory write command. In this case, when the 21154 forwards the memory write and invalidate transaction to the target bus, it converts the command code to a memory write code and does not observe cache line boundaries.

If the value in the cache line size register does meet the memory write and invalidate conditions, that is, the value is a nonzero power of 2 less than or equal to 16 Dwords, the 21154 returns a target disconnect to the initiator either on a cache line boundary or when the posted write buffer fills. For a cache line size of 16 Dwords, the 21154 disconnects a memory write and invalidate transaction on every cache line boundary. When the cache line size is 1, 2, 4, or 8 Dwords, the 21154 accepts another cache line if at least 8 Dwords of empty space remains in the posted write buffer. If less than 8 Dwords of empty space remains, the 21154 disconnects on that cache line boundary.

When the memory write and invalidate transaction is disconnected before a cache line boundary is reached, typically because the posted write buffer fills, the transaction is converted to a memory write transaction.

### 4.5.3 Delayed Write Transactions

Delayed write forwarding is used for I/O write transactions and for Type 1 configuration write transactions.

A delayed write transaction guarantees that the actual target response is returned back to the initiator without holding the initiating bus in wait states. A delayed write transaction is limited to a single Dword data transfer.

When a write transaction is first detected on the initiator bus, and the 21154 forwards it as a delayed transaction, the 21154 claims the access by asserting DEVSEL# and returns a target retry to the initiator. During the address phase, the 21154 samples the bus command, address, and address parity one cycle later. After IRDY# is asserted, the 21154 also samples the first data Dword, byte enable bits, and data parity. This information is placed into the delayed transaction queue. The transaction is queued only if no other existing delayed transactions have the same address and command, and if the delayed transaction queue is not full. When the delayed write transaction moves to the head of the delayed transaction queue and all ordering constraints with posted data are satisfied (see Section 6.0), the 21154 initiates the transaction on the target bus. The 21154 transfers the write data to the target.

If the 21154 receives a target retry in response to the write transaction on the target bus, it continues to repeat the write transaction until the data transfer is completed, or until an error condition is encountered.

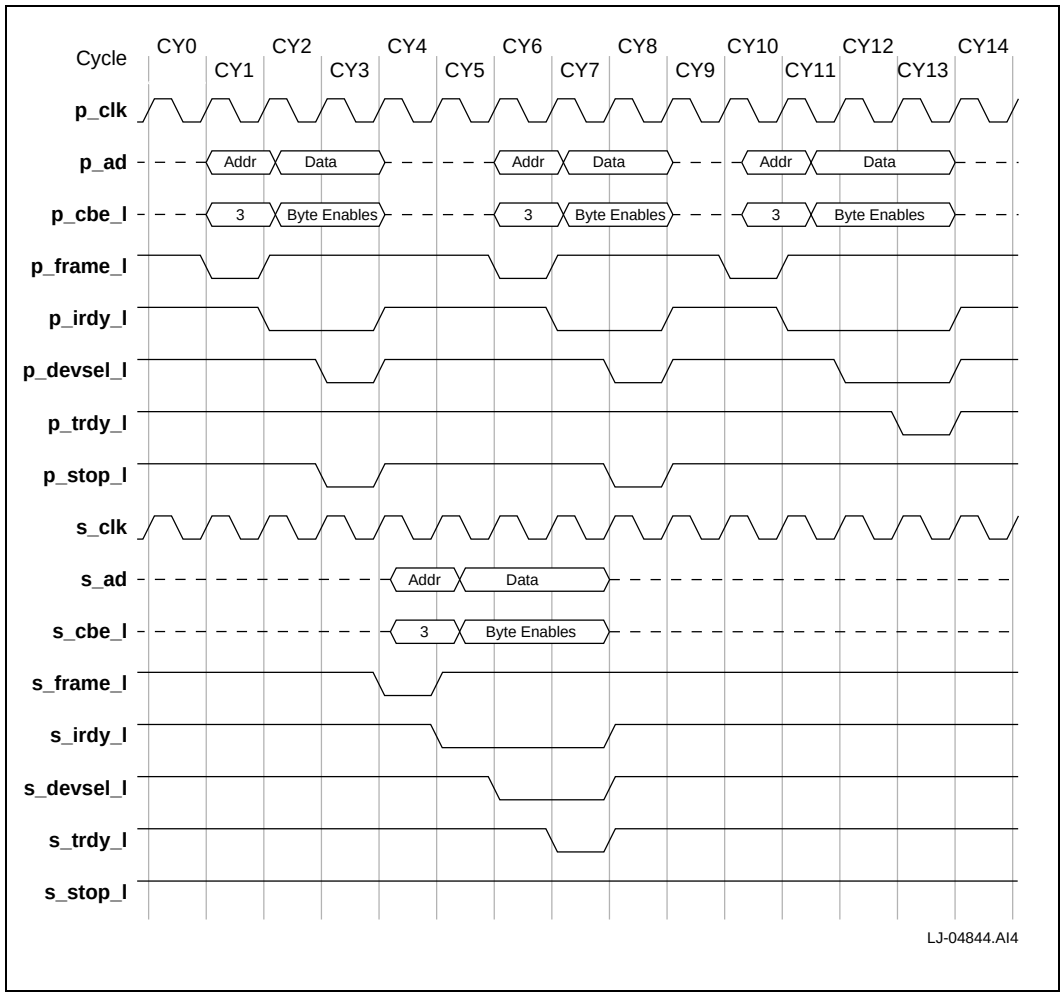
If the 21154 is unable to deliver write data after $2^{24}$ attempts, the 21154 ceases further write attempts and returns a target abort to the initiator. The delayed transaction is removed from the delayed transaction queue. The 21154 also asserts p_serr_l if the primary SERR# enable bit is set in the command register. See Section 7.4 for information on the assertion of p_serr_l.

When the initiator repeats the same write transaction (same command, address, byte enable bits, and data), and the completed delayed transaction is at the head of the queue, the 21154 claims the access by asserting DEVSEL# and returns TRDY# to the initiator, to indicate that the write data was transferred. If the initiator requests multiple Dwords, the 21154 also asserts STOP# in conjunction with TRDY# to signal a target disconnect. Note that only those bytes of write data with valid byte enable bits are compared. If any of the byte enable bits are turned off (driven high), the corresponding byte of write data is not compared.

If the initiator repeats the write transaction before the data has been transferred to the target, the 21154 returns a target retry to the initiator. The 21154 continues to return a target retry to the initiator until write data is delivered to the target, or until an error condition is encountered. When the write transaction is repeated, the 21154 does not make a new entry into the delayed transaction queue. Section 4.10.3.1 provides detailed information about how the 21154 responds to target termination during delayed write transactions.

Figure 7 shows a delayed write transaction forwarded downstream across the 21154.

**Figure 7. Downstream Delayed Write Transaction**



The 21154 implements a discard timer that starts counting when the delayed write completion is at the head of the delayed transaction queue. The initial value of this timer can be set to one of two values, selectable through both the primary and secondary master timeout bits in the bridge control register. If the initiator does not repeat the delayed write transaction before the discard timer expires, the 21154 discards the delayed write transaction from the delayed transaction queue. The 21154 also conditionally asserts p_serr_l (see Section 7.4).

### 4.5.4 Write Transaction Address Boundaries

The 21154 imposes internal address boundaries when accepting write data. The aligned address boundaries are used to prevent the 21154 from continuing a transaction over a device address boundary and to provide an upper limit on maximum latency. The 21154 returns a target disconnect to the initiator when it reaches the aligned address boundaries under the conditions shown in Table 18.

**Table 18. Write Transaction Disconnect Address Boundaries**

| Type of Transaction | Condition | Aligned Address Boundary |
|---|---|---|
| Delayed write | All | Disconnects after one data transfer |
| Posted memory write | Memory write disconnect control bit = 0[1] | 4KB aligned address boundary |
| Posted memory write | Memory write disconnect control bit = 1[1] | Disconnects at cache line boundary |
| Posted memory write and invalidate | Cache line size ≠ 1, 2, 4, 8, 16 | 4KB aligned address boundary |
| Posted memory write and invalidate | Cache line size = 1, 2, 4, 8 | *n*th cache line boundary, where a cache line boundary is reached and less than 8 free Dwords of posted write buffer space remains |
| Posted memory write and invalidate | Cache line size = 16 | 16-Dword aligned address boundary |

1. The memory write disconnect control bit is located in the chip control register at offset 40h in configuration space.

### 4.5.5 Buffering Multiple Write Transactions

The 21154 continues to accept posted memory write transactions as long as space for at least 8 Dwords of data in the posted write data buffer remains. If the posted write data buffer fills before the initiator terminates the write transaction, the 21154 returns a target disconnect to the initiator.

Delayed write transactions are posted as long as at least one open entry in the 21154 delayed transaction queue exists. Therefore, several posted and delayed write transactions can exist in data buffers at the same time.

See Section 6.0 for information about how multiple posted and delayed write transactions are ordered.

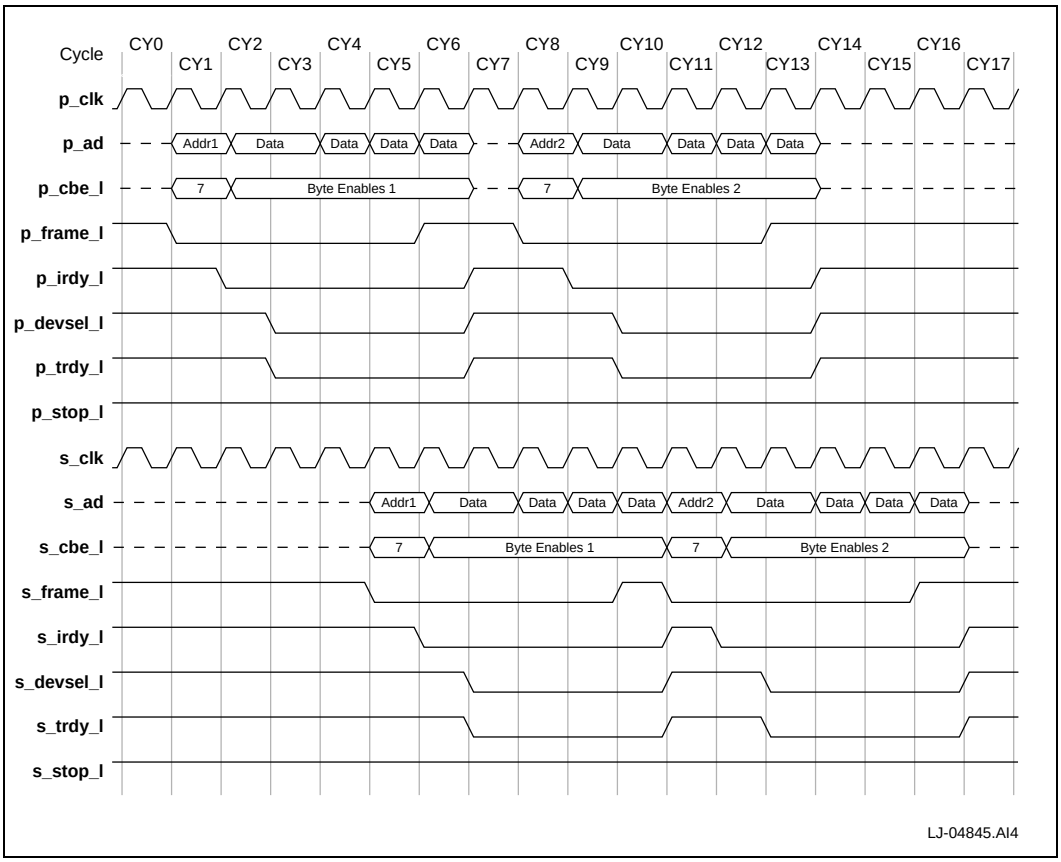
### 4.5.6 Fast Back-to-Back Write Transactions

The 21154 can recognize and post fast back-to-back write transactions. When the 21154 cannot accept the second transaction because of buffer space limitations, it returns a target retry to the initiator.

When the 21154 has posted multiple write transactions, it can initiate fast back-to-back write transactions if the fast back-to-back enable bit is set in the command register for upstream write transactions, and in the bridge control register for downstream write transactions. The 21154 does not perform write combining or merging.

Figure 8 shows how multiple memory write transactions can be posted and then initiated as fast back-to-back transactions on the target bus.

**Figure 8. Fast Back-to-Back Transactions on the Target Bus**



## 4.6 Read Transactions

Delayed read forwarding is used for all read transactions crossing the 21154.

Delayed read transactions are treated as either prefetchable or nonprefetchable.

Table 19 shows the read behavior, prefetchable or nonprefetchable, for each type of read operation.

**Table 19. Read Transaction Prefetching**

| Type of Transaction | Read Behavior |
|---|---|
| I/O read | Prefetching never done |
| Configuration read | Prefetching never done |
| Memory read | Downstream: Prefetching used if address in prefetchable space<br>Upstream: Prefetching used if prefetch disable is off (default) |
| Memory read line | Prefetching always used |
| Memory read multiple | Prefetching always used |

See Section 5.3 for detailed information about prefetchable and nonprefetchable address spaces.

### 4.6.1 Prefetchable Read Transactions

A prefetchable read transaction is a read transaction where the 21154 performs speculative Dword reads, transferring data from the target before it is requested from the initiator. This behavior allows a prefetchable read transaction to consist of multiple data transfers. However, byte enable bits cannot be forwarded for all data phases as is done for the single data phase of the nonprefetchable read transaction. For prefetchable read transactions, the 21154 forces all byte enable bits to be turned on for all data phases.

Prefetchable behavior is used for memory read line and memory read multiple transactions, as well as for memory read transactions that fall into prefetchable memory space.

The amount of data that is prefetched depends on the type of transaction. The amount of prefetching may also be affected by the amount of free buffer space available in the 21154, and by any read address boundaries encountered.

Prefetching should not be used for those read transactions that have side effects in the target device, that is, control and status registers, FIFOs, and so on. The target device's base address register or registers indicate if a memory address region is prefetchable.

### 4.6.2 Nonprefetchable Read Transactions

A nonprefetchable read transaction is a read transaction where the 21154 requests 1, and only 1, Dword from the target and disconnects the initiator after delivery of the first Dword of read data. Unlike prefetchable read transactions, the 21154 forwards the read byte enable information for the data phase.

Nonprefetchable behavior is used for I/O and configuration read transactions, as well as for memory read transactions that fall into nonprefetchable memory space.

If extra read transactions could have side effects, for example, when accessing a FIFO, use nonprefetchable read transactions to those locations. Accordingly, if it is important to retain the value of the byte enable bits during the data phase, use nonprefetchable read transactions. If these locations are mapped in memory space, use the memory read command and map the target into nonprefetchable (memory-mapped I/O) memory space to utilize nonprefetching behavior.

### 4.6.3 Read Prefetch Address Boundaries

The 21154 imposes internal read address boundaries on read prefetching. When a read transaction reaches one of these aligned address boundaries, the 21154 stops prefetching data, unless the target signals a target disconnect before the read prefetch boundary is reached. When the 21154 finishes transferring this read data to the initiator, it returns a target disconnect with the last data transfer, unless the initiator completes the transaction before all prefetched read data is delivered. Any leftover prefetched data is discarded.

Prefetchable read transactions in flow-through mode prefetch to the nearest aligned 4KB address boundary, or until the initiator deasserts FRAME#. Section 4.6.6 describes flow-through mode during read operations.

Table 20 shows the read prefetch address boundaries for read transactions during non-flow-through mode.

**Table 20. Read Prefetch Address Boundaries**

| Type of Transaction | Address Space | Cache Line Size | Prefetch Aligned Address Boundary |
|---|---|---|---|
| Configuration read | — | — | 1 Dword (no prefetch) |
| I/O read | — | — | 1 Dword (no prefetch) |
| Memory read | Nonprefetchable | — | 1 Dword (no prefetch) |
| Memory read | Prefetchable | CLS ≠ 1, 2, 4, 8 | 16-Dword aligned address boundary |
| Memory read | Prefetchable | CLS = 1, 2, 4, 8 | Cache line address boundary |
| Memory read line | — | CLS ≠ 1, 2, 4, 8 | 16-Dword aligned address boundary |
| Memory read line | — | CLS = 1, 2, 4, 8 | Cache line boundary |
| Memory read multiple | — | CLS ≠ 1, 2, 4, 8 | Queue full |
| Memory read multiple | — | CLS = 1, 2, 4, 8 | Second cache line boundary |

### 4.6.4 Delayed Read Requests

The 21154 treats all read transactions as delayed read transactions, which means that the read request from the initiator is posted into a delayed transaction queue. Read data from the target is placed in the read data queue directed toward the initiator bus interface and is transferred to the initiator when the initiator repeats the read transaction.

When the 21154 accepts a delayed read request, it first samples the read address, read bus command, and address parity. When IRDY# is asserted, the 21154 then samples the byte enable bits for the first data phase. This information is entered into the delayed transaction queue. The

21154 terminates the transaction by signaling a target retry to the initiator. Upon reception of the target retry, the initiator is required to continue to repeat the same read transaction until at least one data transfer is completed, or until a target response other than a target retry (target abort, or master abort) is received.

### 4.6.5 Delayed Read Completion with Target

When the delayed read request reaches the head of the delayed transaction queue, and all previously queued posted write transactions have been delivered, the 21154 arbitrates for the target bus and initiates the read transaction, using the exact read address and read command captured from the initiator during the initial delayed read request. If the read transaction is a nonprefetchable read, the 21154 drives the captured byte enable bits during the next cycle. If the transaction is a prefetchable read transaction, it drives all byte enable bits to 0 for all data phases. If the 21154 receives a target retry in response to the read transaction on the target bus, it continues to repeat the read transaction until at least one data transfer is completed, or until an error condition is encountered. If the transaction is terminated via normal master termination or target disconnect after at least one data transfer has been completed, the 21154 does not initiate any further attempts to read more data.

If the 21154 is unable to obtain read data from the target after $2^{24}$ attempts, the 21154 ceases further read attempts and returns a target abort to the initiator. The delayed transaction is removed from the delayed transaction queue. The 21154 also asserts p_serr_l if the primary SERR# enable bit is set in the command register. See Section 7.4 for information on the assertion of p_serr_l.

Once the 21154 receives DEVSEL# and TRDY# from the target, it transfers the data read to the opposite direction read data queue, pointing toward the opposite interface, before terminating the transaction. For example, read data in response to a downstream read transaction initiated on the primary bus is placed in the upstream read data queue. The 21154 can accept 1 Dword of read data each PCI clock cycle; that is, no master wait states are inserted. The number of Dwords transferred during a delayed read transaction depends on the conditions given in Table 20 (assuming no disconnect is received from the target).

### 4.6.6 Delayed Read Completion on Initiator Bus

When the transaction has been completed on the target bus, and the delayed read data is at the head of the read data queue, and all ordering constraints with posted write transactions have been satisfied, the 21154 transfers the data to the initiator when the initiator repeats the transaction. For memory read transactions, the 21154 aliases the memory read, memory read line, and memory read multiple bus commands when matching the bus command of the transaction to the bus command in the delayed transaction queue. The 21154 returns a target disconnect along with the transfer of the last Dword of read data to the initiator. If the initiator terminates the transaction before all read data has been transferred, the remaining read data left in data buffers is discarded.

Figure 9 shows a nonprefetchable delayed read transaction.

**Figure 9. Nonprefetchable Delayed Read Transaction**

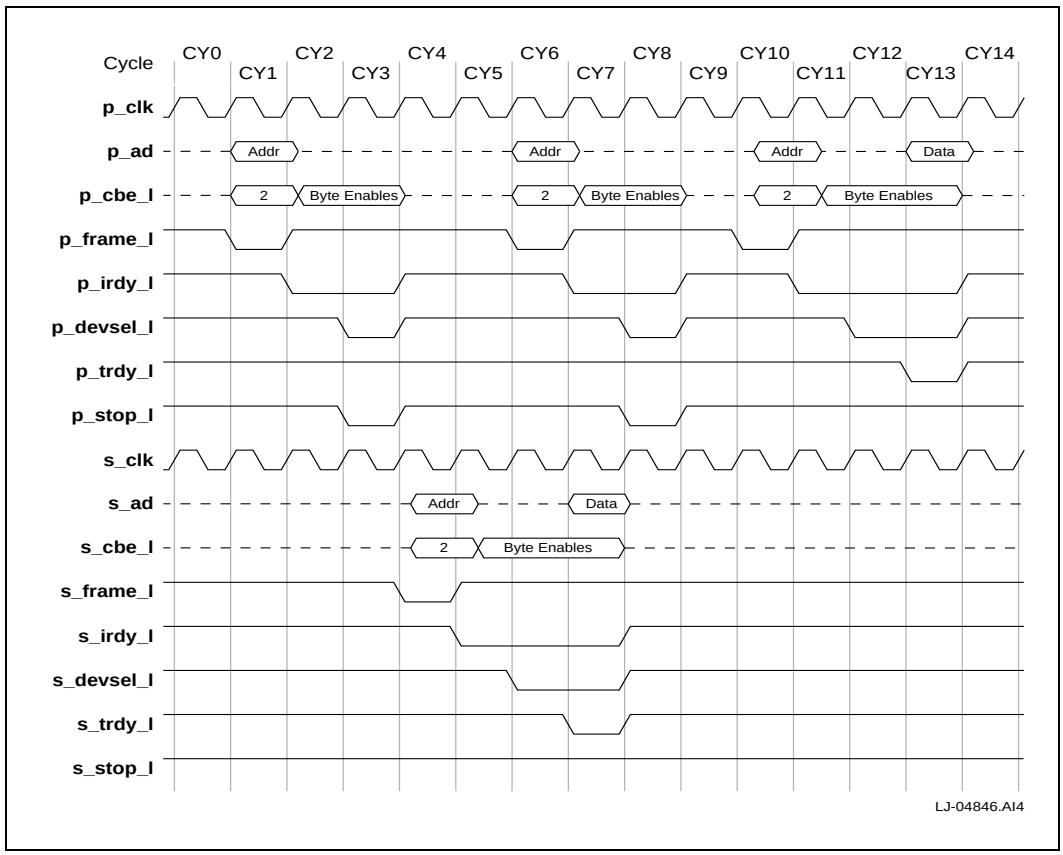
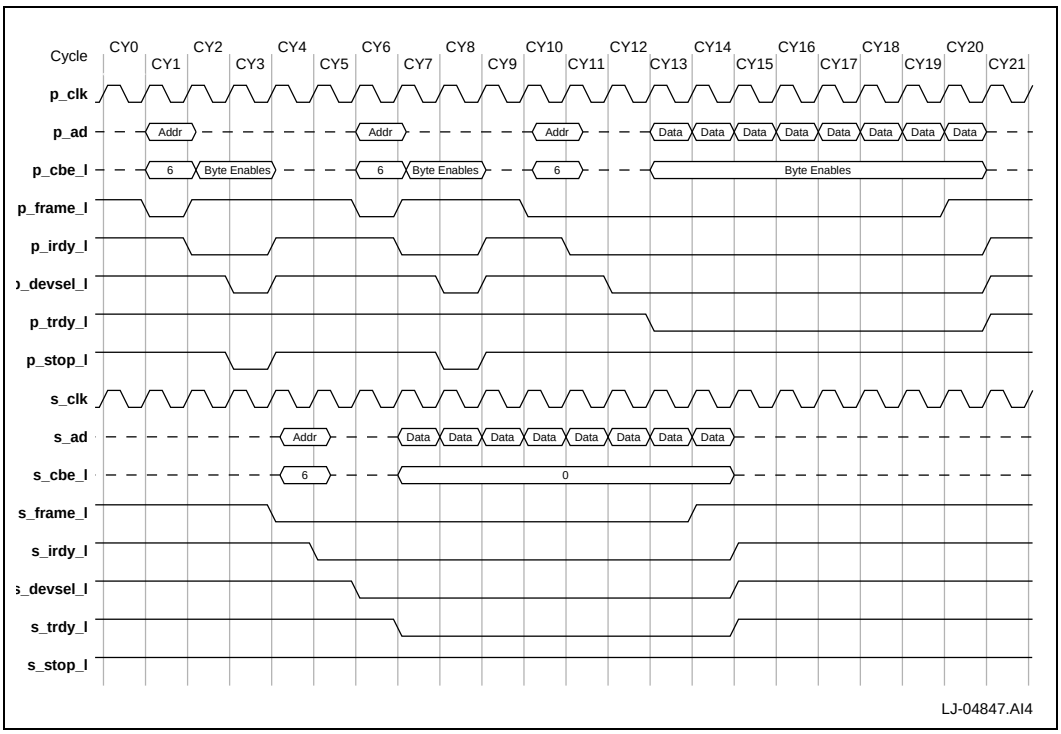


Figure 10 shows a prefetchable delayed read transaction.

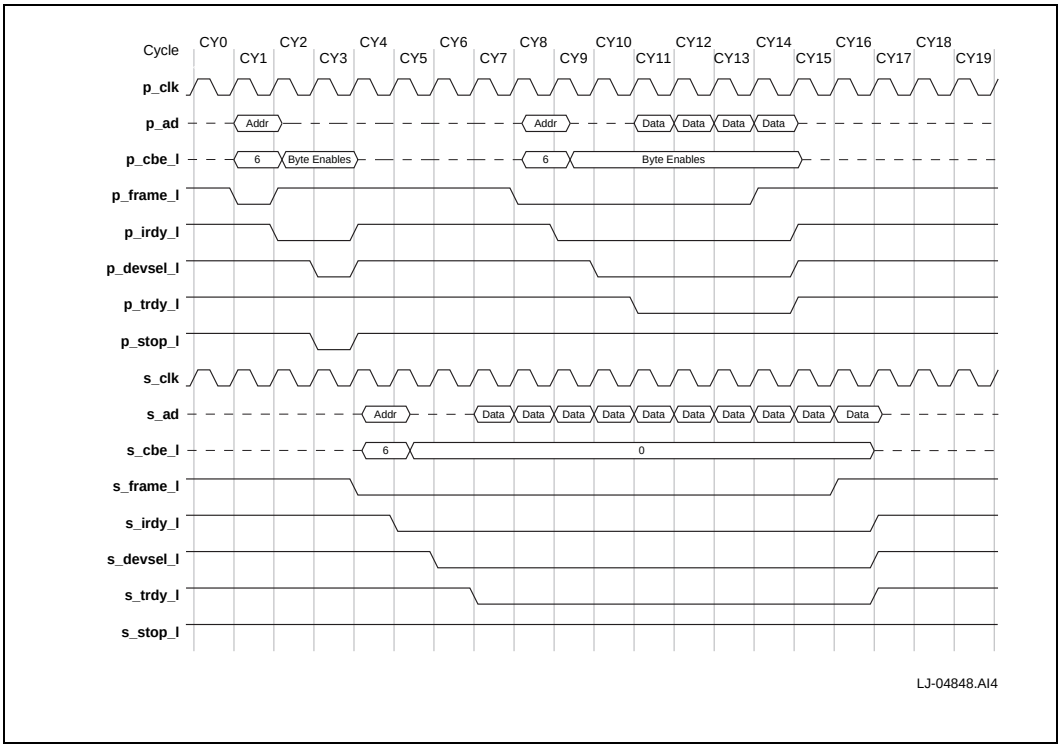
**Figure 10. Prefetchable Delayed Read Transaction**




When the master repeats the transaction and starts transferring prefetchable read data from 21154 data buffers while the read transaction on the target bus is still in progress and before a read boundary is reached on the target bus, the read transaction starts operating in flow-through mode. Because data is flowing through the data buffers from the target to the initiator, long read bursts can then be sustained. In this case, the read transaction is allowed to continue until the initiator terminates the transaction, or until an aligned 4KB address boundary is reached, or until the buffer fills, whichever comes first. When the buffer empties, the 21154 reflects the stalled condition to the initiator by deasserting TRDY# until more read data is available; otherwise, the 21154 does not insert any target wait states. When the initiator terminates the transaction, the deassertion of FRAME# on the initiator bus is forwarded to the target bus. Any remaining read data is discarded.

Figure 11 shows a flow-through prefetchable read transaction.

**Figure 11. Flow-Through Prefetchable Read Transaction**




The 21154 implements a discard timer that starts counting when the delayed read completion is at the head of the delayed transaction queue, and the read data is at the head of the read data queue. The initial value of this timer can be set to one of two values, selectable through both the primary and secondary master timeout value bits in the bridge control register. If the initiator does not repeat the read transaction before the discard timer expires, the 21154 discards the read transaction and the read data from its queues. The 21154 also conditionally asserts p_serr_l (see Section 7.4).

The 21154 has the capability to post multiple delayed read requests, up to a maximum of three in each direction. If an initiator starts a read transaction that matches the address and read command of a read transaction that is already queued, the current read command is not posted as it is already contained in the delayed transaction queue.

See Section 6.0 for a discussion of how delayed read transactions are ordered when crossing the 21154.

## 4.7 Configuration Transaction

Configuration transactions are used to initialize a PCI system. Every PCI device has a configuration space that is accessed by configuration commands. All 21154 registers are accessible in configuration space only.

In addition to accepting configuration transactions for initialization of its own configuration space, the 21154 also forwards configuration transactions for device initialization in hierarchical PCI systems, as well as for special cycle generation.

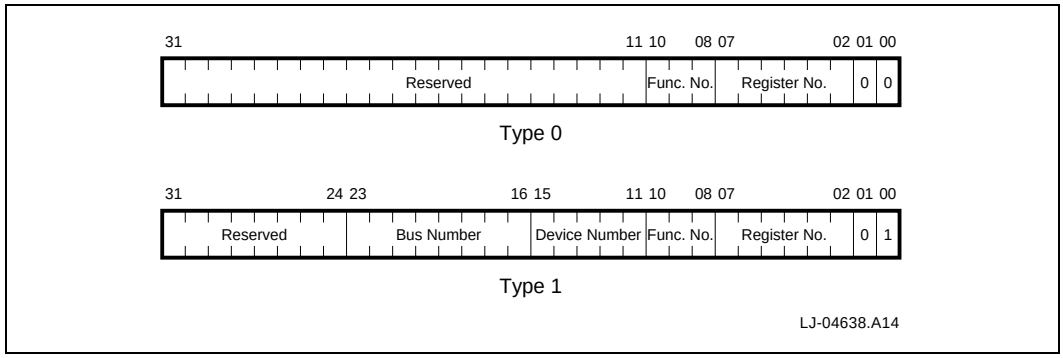
To support hierarchical PCI bus systems, two types of configuration transactions are specified: Type 0 and Type 1.

Type 0 configuration transactions are issued when the intended target resides on the same PCI bus as the initiator. A Type 0 configuration transaction is identified by the configuration command and the lowest 2 bits of the address set to 00b.

Type 1 configuration transactions are issued when the intended target resides on another PCI bus, or when a special cycle is to be generated on another PCI bus. A Type 1 configuration command is identified by the configuration command and the lowest 2 address bits set to 01b.

Figure 12 shows the address formats for Type 0 and Type 1 configuration transactions.

**Figure 12. Configuration Transaction Address Formats**




The register number is found in both Type 0 and Type 1 formats and gives the Dword address of the configuration register to be accessed. The function number is also included in both Type 0 and Type 1 formats and indicates which function of a multifunction device is to be accessed. For single-function devices, this value is not decoded. Type 1 configuration transaction addresses also include a 5-bit field designating the device number that identifies the device on the target PCI bus that is to be accessed. In addition, the bus number in Type 1 transactions specifies the PCI bus to which the transaction is targeted.

### 4.7.1 Type 0 Access to the 21154

The 21154 configuration space is accessed by a Type 0 configuration transaction on the primary interface. The 21154 configuration space cannot be accessed from the secondary bus. The 21154 responds to a Type 0 configuration transaction by asserting p_devsel_l when the following conditions are met during the address phase:

- The bus command is a configuration read or configuration write transaction.

- Low 2 address bits p_ad<1:0> must be 00b.
- Signal p_idsel must be asserted.

The function code is ignored because the 21154 is a single-function device.

The 21154 limits all configuration accesses to a single Dword data transfer and returns a target disconnect with the first data transfer if additional data phases are requested. Because read transactions to 21154 configuration space do not have side effects, all bytes in the requested Dword are returned, regardless of the value of the byte enable bits.

Type 0 configuration write and read transactions do not use 21154 data buffers; that is, these transactions are completed immediately, regardless of the state of the data buffers.

The 21154 ignores all Type 0 transactions initiated on the secondary interface.

### 4.7.2 Type 1 to Type 0 Translation

Type 1 configuration transactions are used specifically for device configuration in a hierarchical PCI bus system. A PCI-to-PCI bridge is the only type of device that should respond to a Type 1 configuration command. Type 1 configuration commands are used when the configuration access is intended for a PCI device that resides on a PCI bus other than the one where the Type 1 transaction is generated.

The 21154 performs a Type 1 to Type 0 translation when the Type 1 transaction is generated on the primary bus and is intended for a device attached directly to the secondary bus. The 21154 must convert the configuration command to a Type 0 format so that the secondary bus device can respond to it. Type 1 to Type 0 translations are performed only in the downstream direction; that is, the 21154 generates a Type 0 transaction only on the secondary bus, and never on the primary bus.

The 21154 responds to a Type 1 configuration transaction and translates it into a Type 0 transaction on the secondary bus when the following conditions are met during the address phase:

- The low 2 address bits on p_ad<1:0> are 01b.
- The bus number in address field p_ad<23:16> is equal to the value in the secondary bus number register in 21154 configuration space.
- The bus command on p_cbe_l<3:0> is a configuration read or configuration write transaction.

When the 21154 translates the Type 1 transaction to a Type 0 transaction on the secondary interface, it performs the following translations to the address:

- Sets the low 2 address bits on s_ad<1:0> to 00b
- Decodes the device number and drives the bit pattern specified in Table 21 on s_ad<31:16> for the purpose of asserting the device's IDSEL signal
- Sets s_ad<15:11> to 0
- Leaves unchanged the function number and register number fields

The 21154 asserts a unique address line based on the device number. These address lines may be used as secondary bus IDSEL signals. The mapping of the address lines depends on the device number in the Type 1 address bits p_ad<15:11>.

Table 21 presents the mapping that the 21154 uses.

**Table 21. Device Number to IDSEL s_ad Pin Mapping**

| Device Number | p_ad<15:11> | Secondary IDSEL s_ad<31:16> | s_ad Bit |
|---|---|---|---|
| 0h | 00000 | 0000 0000 0000 0001 | 16 |
| 1h | 00001 | 0000 0000 0000 0010 | 17 |
| 2h | 00010 | 0000 0000 0000 0100 | 18 |
| 3h | 00011 | 0000 0000 0000 1000 | 19 |
| 4h | 00100 | 0000 0000 0001 0000 | 20 |
| 5h | 00101 | 0000 0000 0010 0000 | 21 |
| 6h | 00110 | 0000 0000 0100 0000 | 22 |
| 7h | 00111 | 0000 0000 1000 0000 | 23 |
| 8h | 01000 | 0000 0001 0000 0000 | 24 |
| 9h | 01001 | 0000 0010 0000 0000 | 25 |
| Ah | 01010 | 0000 0100 0000 0000 | 26 |
| Bh | 01011 | 0000 1000 0000 0000 | 27 |
| Ch | 01100 | 0001 0000 0000 0000 | 28 |
| Dh | 01101 | 0010 0000 0000 0000 | 29 |
| Eh | 01110 | 0100 0000 0000 0000 | 30 |
| Fh | 01111 | 1000 0000 0000 0000 | 31 |
| 10h–1Eh | 10000—11110 | 0000 0000 0000 0000 | — |
| 1Fh | 11111 | Generate special cycle (p_ad<7:2> = 00h) 0000 0000 0000 0000 ( p_ad<7:2> ≠ 00h) | — |

The 21154 can assert up to 16 unique address lines to be used as IDSEL signals for up to 16 devices on the secondary bus, for device numbers ranging from 0 through 15. Because of electrical loading constraints of the PCI bus, more than 16 IDSEL signals should not be necessary. However, if device numbers greater than 15 are desired, some external method of generating IDSEL lines must be used, and no upper address bits are then asserted. The configuration transaction is still translated and passed from the primary bus to the secondary bus. If no IDSEL pin is asserted to a secondary device, the transaction ends in a master abort.

The 21154 forwards Type 1 to Type 0 configuration read or write transactions as delayed transactions. Type 1 to Type 0 configuration read or write transactions are limited to a single 32-bit data transfer.

### 4.7.3 Type 1 to Type 1 Forwarding

Type 1 to Type 1 transaction forwarding provides a hierarchical configuration mechanism when two or more levels of PCI-to-PCI bridges are used.

When the 21154 detects a Type 1 configuration transaction intended for a PCI bus downstream from the secondary bus, the 21154 forwards the transaction unchanged to the secondary bus. Ultimately, this transaction is translated to a Type 0 configuration command or to a special cycle transaction by a downstream PCI-to-PCI bridge. Downstream Type 1 to Type 1 forwarding occurs when the following conditions are met during the address phase:

- The low 2 address bits are equal to 01b.
- The bus number falls in the range defined by the lower limit (exclusive) in the secondary bus number register and the upper limit (inclusive) in the subordinate bus number register.
- The bus command is a configuration read or write transaction.

The 21154 also supports Type 1 to Type 1 forwarding of configuration write transactions upstream to support upstream special cycle generation. A Type 1 configuration command is forwarded upstream when the following conditions are met:

- The low 2 address bits are equal to 01b.
- The bus number falls outside the range defined by the lower limit (inclusive) in the secondary bus number register and the upper limit (inclusive) in the subordinate bus number register.
- The device number in address bits AD<15:11> is equal to 11111b.
- The function number in address bits AD<10:8> is equal to 111b.
- The bus command is a configuration write transaction.

The 21154 forwards Type 1 to Type 1 configuration write transactions as delayed transactions. Type 1 to Type 1 configuration write transactions are limited to a single data transfer.

### 4.7.4 Special Cycles

The Type 1 configuration mechanism is used to generate special cycle transactions in hierarchical PCI systems. Special cycle transactions are ignored by a PCI-to-PCI bridge acting as a target and are not forwarded across the bridge. Special cycle transactions can be generated from Type 1 configuration write transactions in either the upstream or the downstream direction.

The 21154 initiates a special cycle on the target bus when a Type 1 configuration write transaction is detected on the initiating bus and the following conditions are met during the address phase:

- The low 2 address bits on AD<1:0>are equal to 01b.
- The device number in address bits AD<15:11> is equal to 11111b.
- The function number in address bits AD<10:8> is equal to 111b.
- The register number in address bits AD<7:2> is equal to 000000b.
- The bus number is equal to the value in the secondary bus number register in configuration space for downstream forwarding or equal to the value in the primary bus number register in configuration space for upstream forwarding.
- The bus command on C/BE# is a configuration write command.

When the 21154 initiates the transaction on the target interface, the bus command is changed from configuration write to special cycle. The address and data are forwarded unchanged. Devices that use special cycles ignore the address and decode only the bus command. The data phase contains the special cycle message. The transaction is forwarded as a delayed transaction, but in this case the target response is not forwarded back (because special cycles result in a master abort). Once the transaction is completed on the target bus, through detection of the master abort condition, the 21154 responds with TRDY# to the next attempt of the configuration transaction from the initiator. If more than one data transfer is requested, the 21154 responds with a target disconnect operation during the first data phase.

## 4.8 64-Bit Operation

The 21154 provides 64-bit extension support on the primary and secondary interfaces. Both 64-bit and 32-bit operation are supported on both interfaces.

This section describes how to use the 64-bit extensions. It describes the conditions under which a transaction can be treated as a 64-bit transaction and includes information about how the transaction is forwarded.

### 4.8.1 64-Bit and 32-Bit Transactions Initiated by the 21154

The 21154 requests a 64-bit transaction on the primary or secondary bus 64-bit PCI extension by asserting p_req64_l on the primary bus or s_req64_l on the secondary bus respectively, during the address phase.

The 21154 asserts and deasserts REQ64# during the same cycles in which it asserts and deasserts FRAME#, respectively.

Under certain circumstances, the 21154 does not use the 64-bit extension when initiating transactions and therefore does not assert REQ64#.

The 21154 does not assert REQ64# when initiating a transaction, and the transaction is therefore initiated as a 32-bit wide transaction, when any of the following is true:

- Signal p_req64_l was not asserted by the primary bus central function during reset (64-bit extension not supported on primary PCI bus), for upstream transactions only.
- The 21154 is initiating an I/O transaction.
- The 21154 is initiating a configuration transaction.
- The 21154 is initiating a nonprefetchable memory read transaction.
- The 21154 is initiating a special cycle transaction.
- The address is not quadword aligned (AD<2> = 1).
- A 1- or 2-Dword memory write transaction is being performed.
- The 21154 is resuming a memory write transaction after a target disconnect, and ACK64# was not asserted by the target in the previous transaction—does not apply when the previous target termination was a target retry.
- A single Dword read transaction is being performed.
- The address is near the top of a cache line (AD<3> = 1)—applies to prefetchable read transactions.

### 4.8.2 Address Phase of 64-Bit Transactions

When a transaction using the primary bus 64-bit extension is a single address cycle (SAC)—that is, the address falls below the 4GB boundary, and the upper 32 bits of the address are assumed to be zero—AD<63:32> and C/BE#<7:4> are not defined but are driven to valid logic level during the address phase.

When the transaction is a dual address cycle (DAC)—that is, the address falls above the 4GB boundary, and the upper 32 bits of the address are nonzero—signals AD<63:32> contain the upper 32 bits of the address for both address phases. Signals C/BE#<7:4> contain the memory bus command during both address phases. A 64-bit target then has the opportunity to decode the entire 64-bit address and bus command after the first address phase. A 32-bit target needs both address phases to decode the full address and bus command.

### 4.8.3 Data Phase of 64-Bit Transactions

During memory write transactions, when the 21154 has driven REQ64# to indicate it is initiating a 64-bit transfer, during the data phase the 21154 drives the following:

- The low 32 bits of data on AD<31:0>
- The low four byte enable bits on C/BE#<3:0>
- The high 32 bits of data on AD<63:32>
- The high four byte enable bits on C/BE#<7:4>

When the 21154 detects ACK64# asserted by the target at the same time that it detects DEVSEL# asserted, every data phase then consists of 64 bits and eight byte enable bits.

For write transactions, when the 21154 does not detect ACK64# asserted at the same time that it detects DEVSEL# asserted, the 21154 redirects the write data that it has on the AD<63:32> bus to AD<31:0> during the second data phase. Similarly, the upper four byte enable bits are redirected to C/BE#<3:0> during the second data phase. All data phases then consist of 32 bits.

For 64-bit memory write transactions that end at an odd Dword boundary, the 21154 drives the byte enable bits to 1 during the last data phase. Signals AD<63:32> are then unpredictable but are driven to a valid logic level.

For read transactions, when the 21154 has asserted REQ64#, it drives 8 bits of byte enables on C/BE#<7:0>. Because the only read transactions that use the 64-bit extension are prefetchable memory read transactions, the byte enable bits are always zero. Therefore, no special redirection is needed based on the target's assertion or lack of assertion of ACK64#. When the target asserts ACK64# at the same time that it asserts DEVSEL#, all read data transfers then consist of 64 bits and the target drives PAR64, which covers AD<63:32> and C/BE#<7:4>. When the target does not assert ACK64# when it asserts DEVSEL#, all data phases then consist of 32 bits.

### 4.8.4 64-Bit Transactions Received by the 21154

When the 21154 is the target of a transaction and the 21154 detects REQ64# asserted during a memory transaction to be forwarded across the bridge, the 21154 either asserts ACK64# at the same time that it asserts DEVSEL# to indicate its ability to perform 64-bit data transfers or, under certain circumstances, the 21154 does not use the 64-bit extension as a target and therefore does not assert ACK64#.

The 21154 does not assert ACK64# when any of the following is true:

- Signal REQ64# was not asserted by the initiator.
- The 21154 is responding to a nonprefetchable memory read transaction.
- The 21154 is responding to an I/O transaction.
- The 21154 is responding to a configuration transaction.

- Only 1 Dword of data was read from the target.

When the 21154 is the target of a 64-bit memory write transaction, it is able to accept 64 bits of data during each data phase.

When the 21154 is the target of a 64-bit prefetchable memory read transaction, it supplies 64 bits of read data during each data phase and drives PAR64 corresponding to AD<63:32> and C/BE#<7:4>, for each data phase. If an odd number of Dwords was read from the target and the 21154 has asserted ACK64# when returning read data to the initiator, the 21154 disconnects before the last odd Dword is returned. The 21154 may have read an odd number of Dwords because of either a target disconnect or a master latency timer expiration during 32-bit data transfers on the opposite interface.

### 4.8.5 64-Bit Extension Support During Reset

When the 21154 supports a 64-bit interface on its primary bus, it samples p_req64_l while p_rst_l is asserted to determine whether the PCI 64-bit extension signals are connected on the board. If p_req64_l is high, the 64-bit extension signals are not connected and the 21154 then always drives the 64-bit extension outputs to have valid logic levels on the inputs. The 21154 treats all transactions on the primary interface as 32-bit transactions. If p_req64_l is low, the 64-bit signals are connected to pull-up resistors on the board and the 21154 does not perform any input biasing. In this case, the 21154 can treat memory write and prefetchable memory read transactions as 64-bit transactions on the primary interface, as discussed previously.

The 21154 always asserts s_req64_l low during s_rst_l assertion to indicate that the 64-bit extension is supported on the secondary bus. Individual pull-up resistors must always be supplied for s_ad<63:32>, s_cbe_l<7:4>, and s_par64

## 4.9 Transaction Flow Through

Transaction flow occurs when data is removed from a 21154 read or write buffer at the same time that data for that transaction is still being written to the buffer.

For writes, flow-through occurs when the 21154 is able to arbitrate for the target bus, initiate the transaction and receive a TRDY# from the target, while still receiving data from that same transaction on the initiator bus. Writes that were previously posted in that same direction must be delivered before flow-through can occur.

For reads, flow-through occurs when the initiator repeats the delayed transaction during a timing window when some read data is in the buffer, but the transaction is still ongoing on the target bus. For read flow-through to occur, there can be no other reads or writes previously posted in the same direction.

The 21154 allows flow-through only when a queue empty condition cannot occur during the middle of a transaction (in the absence of wait states). Therefore, a transaction can flow-through only when the bus bandwidth for data coming into the 21154 is the same or greater than the bus bandwidth for data delivered by the 21154.

For example, when the 21154 accepts a memory write transaction as a 32-bit transaction but the 21154 initiates it as a 64-bit transaction, the entire write transaction must be posted in the 21154 write buffers before the 21154 can initiate the transaction. Otherwise, if the transaction were to be initiated on the target bus while write data was still being accepted by the 21154 on the initiator bus, write data could be removed from the posted write buffer at a rate greater than write data is

posted, even in the absence of master wait states on the initiator bus. It would then be possible for the posted write buffer to empty during the write transaction. As a result, additional wait states would be introduced on the target bus. To avoid introducing additional wait states, all the data must be posted and the write transaction completed on the initiator bus before the transaction can be initiated on the target bus.

Similarly, when read data is accepted from the target using 32-bit transfers, but is supplied to the initiator using 64-bit transfers, the 21154 continues to return target retry to the initiator until all read data is buffered and the transaction on the target bus has been terminated.

The previous examples apply when both interfaces are operating at the same frequency. However, bus frequency as well as bandwidth must be considered when calculating whether flow-through can occur. Thus, a write transaction initiated on a 32-bit, 66 MHz bus and directed to a 64-bit, 33 MHz bus can still flow-through the 21154, because the bandwidths of the two buses are the same.

## 4.10 Transaction Termination

This section describes how the 21154 returns transaction termination conditions back to the initiator.

The initiator can terminate transactions with one of the following types of termination:

- Normal termination—occurs when the initiator deasserts FRAME# at the beginning of the last data phase, and deasserts IRDY# at the end of the last data phase in conjunction with either TRDY# or STOP# assertion from the target.
- Master abort—occurs when no target response is detected. When the initiator does not detect a DEVSEL# from the target within five clock cycles after asserting FRAME#, the initiator terminates the transaction with a master abort. If FRAME# is still asserted, the initiator deasserts FRAME# on the next cycle, and then deasserts IRDY# on the following cycle. IRDY# must be asserted in the same cycle in which FRAME# deasserts. If FRAME# is already deasserted, IRDY# can be deasserted on the next clock cycle following detection of the master abort condition.

The target can terminate transactions with one of the following types of termination:

- Normal termination—TRDY# and DEVSEL# asserted in conjunction with FRAME# deasserted and IRDY# asserted.
- Target retry—STOP# and DEVSEL# asserted without TRDY# during the first data phase. No data transfers occur during the transaction. This transaction must be repeated.
- Target disconnect with data transfer—STOP# and DEVSEL# asserted with TRDY#. Signals that this is the last data transfer of the transaction.
- Target disconnect without data transfer—STOP# and DEVSEL# asserted without TRDY# after previous data transfers have been made. Indicates that no more data transfers will be made during this transaction.
- Target abort—STOP# asserted without DEVSEL# and without TRDY#. Indicates that the target will never be able to complete this transaction. DEVSEL# must be asserted for at least one cycle during the transaction before the target abort is signaled.

### 4.10.1 Master Termination Initiated by the 21154

The 21154, as an initiator, uses normal termination if DEVSEL# is returned by the target within five clock cycles of the 21154's assertion of FRAME# on the target bus. As an initiator, the 21154 terminates a transaction when the following conditions are met:

- During a delayed write transaction, a single Dword is delivered.
- During a nonprefetchable read transaction, a single Dword is transferred from the target.
- During a prefetchable read transaction, a prefetch boundary is reached.
- For a posted write transaction, all write data for the transaction is transferred from 21154 data buffers to the target.
- For a burst transfer, with the exception of memory write and invalidate transactions, the master latency timer expires and the 21154's bus grant is deasserted.
- The target terminates the transaction with a retry, disconnect, or target abort.

If the 21154 is delivering posted write data when it terminates the transaction because the master latency timer expires, it initiates another transaction to deliver the remaining write data. The address of the transaction is updated to reflect the address of the current Dword to be delivered.

If the 21154 is prefetching read data when it terminates the transaction because the master latency timer expires, it does not repeat the transaction to obtain more data.

### 4.10.2 Master Abort Received by the 21154

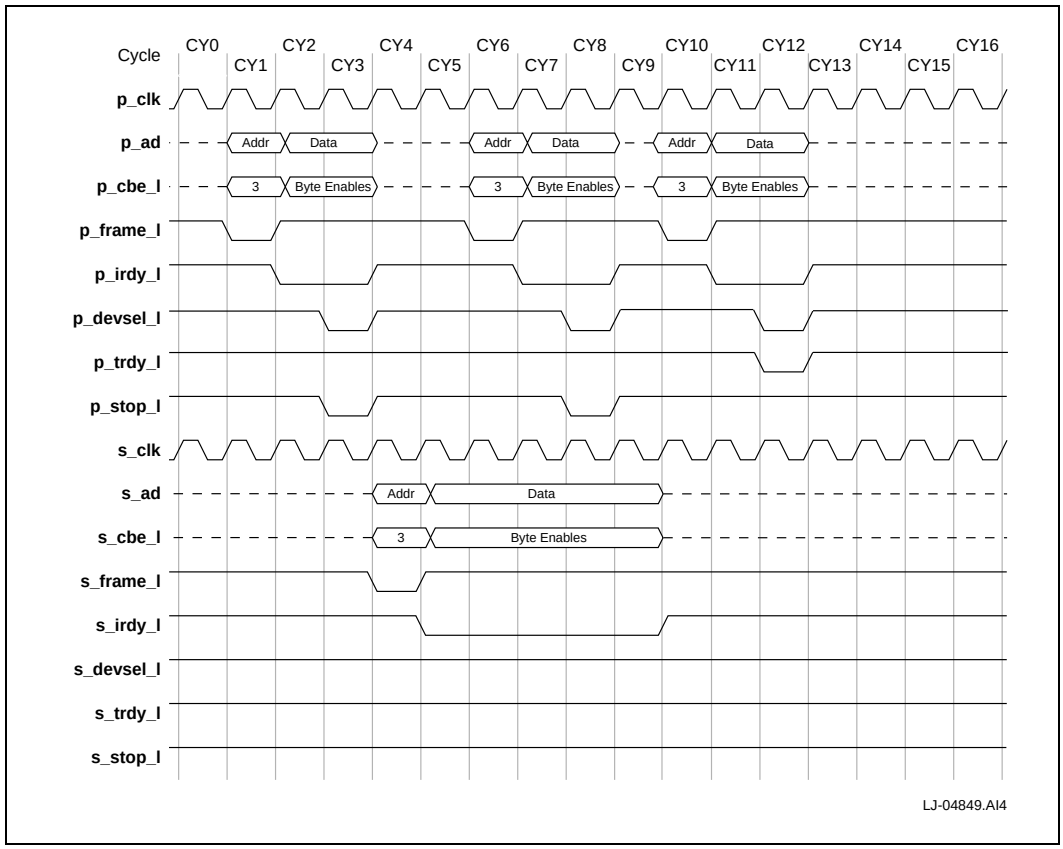
If the 21154 initiates a transaction on the target bus and does not detect DEVSEL# returned by the target within five clock cycles of the 21154's assertion of FRAME#, the 21154 terminates the transaction with a master abort. The 21154 sets the received master abort bit in the status register corresponding to the target bus.

For delayed read and write transactions, when the master abort mode bit in the bridge control register is 0, the 21154 returns TRDY# on the initiator bus and, for read transactions, returns FFFF FFFFh as data.

When the master abort mode bit is 1, the 21154 returns target abort on the initiator bus. The 21154 also sets the signaled target abort bit in the register corresponding to the initiator bus.

Figure 13 shows a delayed write transaction that is terminated with a master abort.

**Figure 13. Delayed Write Transaction Terminated with Master Abort**




When a master abort is received in response to a posted write transaction, the 21154 discards the posted write data and makes no more attempts to deliver the data. The 21154 sets the received master abort bit in the status register when the master abort is received on the primary bus, or it sets the received master abort bit in the secondary status register when the master abort is received on the secondary interface. When a master abort is detected in response to a posted write transaction and the master abort bit is set, the 21154 also asserts p_serr_l if enabled by the SERR# enable bit in the command register and if not disabled by the device-specific p_serr_l disable bit for master abort during posted write transactions (that is, master abort mode = 1; SERR# enable bit = 1; and p_serr_l disable bit for master aborts = 0).

***Note:*** When the 21154 performs a Type 1 to special cycle translation, a master abort is the expected termination for the special cycle on the target bus. In this case, the master abort received bit is *not* set, and the Type 1 configuration transaction is disconnected after the first data phase.

### 4.10.3 Target Termination Received by the 21154

When the 21154 initiates a transaction on the target bus and the target responds with DEVSEL#, the target can end the transaction with one of the following types of termination:

- Normal termination (upon deassertion of FRAME#)
- Target retry

- Target disconnect
- Target abort

The 21154 handles these terminations in different ways, depending on the type of transaction being performed.

#### 4.10.3.1 Delayed Write Target Termination Response

When the 21154 initiates a delayed write transaction, the type of target termination received from the target can be passed back to the initiator. Table 22 shows the 21154 response to each type of target termination that occurs during a delayed write transaction.

**Table 22. 21154 Response to Delayed Write Target Termination**

| Target Termination | 21154 Response |
|---|---|
| Normal | Return disconnect to initiator with first data transfer only if multiple data phases requested. |
| Target retry | Return target retry to initiator. Continue write attempts to target. |
| Target disconnect | Return disconnect to initiator with first data transfer only if multiple data phases requested. |
| Target abort | Return target abort to initiator.<br>Set received target abort bit in target interface status register.<br>Set signaled target abort bit in initiator interface status register. |

The 21154 repeats a delayed write transaction until one of the following conditions is met:

- The 21154 completes at least one data transfer.
- The 21154 receives a master abort.
- The 21154 receives a target abort.
- The 21154 makes $2^{24}$ write attempts resulting in a response of target retry.

After the 21154 makes $2^{24}$ attempts of the same delayed write transaction on the target bus, the 21154 asserts p_serr_l if the primary SERR# enable bit is set in the command register and the implementation-specific p_serr_l disable bit for this condition is not set in the p_serr_l event disable register. The 21154 stops initiating transactions in response to that delayed write transaction. The delayed write request is discarded. Upon a subsequent write transaction attempt by the initiator, the 21154 returns a target abort. See Section 7.4 for a description of system error conditions.

#### 4.10.3.2 Posted Write Target Termination Response

When the 21154 initiates a posted write transaction, the target termination cannot be passed back to the initiator. Table 23 shows the 21154 response to each type of target termination that occurs during a posted write transaction.

**Table 23. 21154 Response to Posted Write Target Termination**

| Target Termination | 21154 Response |
|---|---|
| Normal | No additional action. |
| Target retry | Repeat write transaction to target. |
| Target disconnect | Initiate write transaction to deliver remaining posted write data. |
| Target abort | Set received target abort bit in the target interface status register. Assert **p_serr_l** if enabled, and set the signaled system error bit in the primary status register. |

Note that when a target retry or target disconnect is returned and posted write data associated with that transaction remains in the write buffers, the 21154 initiates another write transaction to attempt to deliver the rest of the write data. In the case of a target retry, the exact same address will be driven as for the initial write transaction attempt. If a target disconnect is received, the address that is driven on a subsequent write transaction attempt is updated to reflect the address of the current Dword. If the initial write transaction is a memory write and invalidate transaction, and a partial delivery of write data to the target is performed before a target disconnect is received, the 21154 uses the memory write command to deliver the rest of the write data because less than a cache line will be transferred in the subsequent write transaction attempt.

After the 21154 makes $2^{24}$ write transaction attempts and fails to deliver all the posted write data associated with that transaction, the 21154 asserts p_serr_l if the primary SERR# enable bit is set in the command register and the device-specific p_serr_l disable bit for this condition is not set in the p_serr_l event disable register. The write data is discarded. See Section 7.4 for a discussion of system error conditions.

#### 4.10.3.3 Delayed Read Target Termination Response

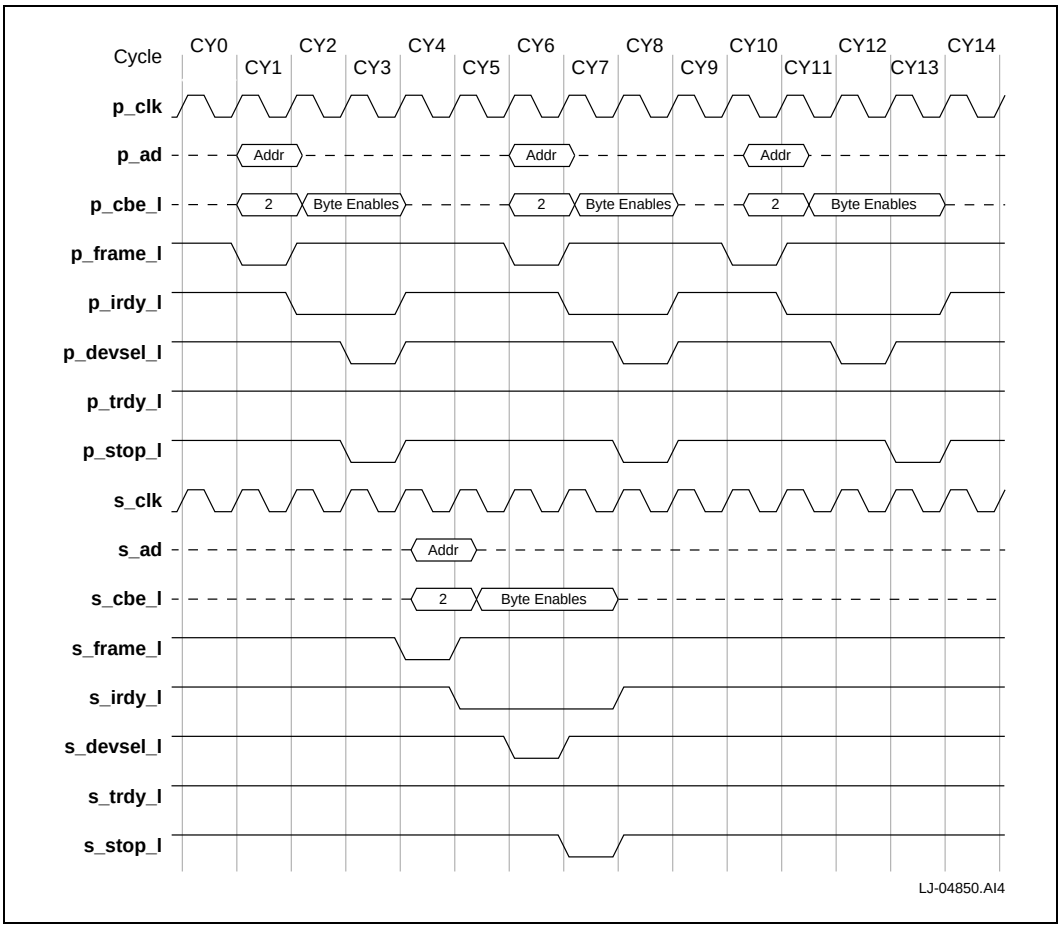
When the 21154 initiates a delayed read transaction, the abnormal target responses can be passed back to the initiator. Other target responses depend on how much data the initiator requests. Table 24 shows the 21154 response to each type of target termination that occurs during a delayed read transaction.

**Table 24. 21154 Response to Delayed Read Target Termination**

| Target Termination | 21154 Response |
|---|---|
| Normal | If prefetchable, target disconnect only if initiator requests more data than read from target. If nonprefetchable, target disconnect on first data phase. |
| Target retry | Reinitiate read transaction to target. |
| Target disconnect | If initiator requests more data than read from target, return target disconnect to initiator. |
| Target abort | Return target abort to initiator.<br>Set received target abort bit in the target interface status register.<br>Set signaled target abort bit in the initiator interface status register. |

Figure 14 shows a delayed read transaction that is terminated with a target abort.

**Figure 14. Delayed Read Transaction Terminated with Target Abort**



The 21154 repeats a delayed read transaction until one of the following conditions is met:

- The 21154 completes at least one data transfer.
- The 21154 receives a master abort.
- The 21154 receives a target abort.
- The 21154 makes $2^{24}$ read attempts resulting in a response of target retry.

After the 21154 makes $2^{24}$ attempts of the same delayed read transaction on the target bus, the 21154 asserts p_serr_l if the primary SERR# enable bit is set in the command register and the implementation-specific p_serr_l disable bit for this condition is not set in the p_serr_l event disable register. The 21154 stops initiating transactions in response to that delayed read transaction. The delayed read request is discarded. Upon a subsequent read transaction attempt by the initiator, the 21154 returns a target abort. See Section 7.4 for a description of system error conditions.

### 4.10.4 Target Termination Initiated by the 21154

The 21154 can return a target retry, target disconnect, or target abort to an initiator for reasons other than detection of that condition at the target interface.

#### 4.10.4.1 Target Retry

The 21154 returns a target retry to the initiator when it cannot accept write data or return read data as a result of internal conditions. The 21154 returns a target retry to an initiator when any of the following conditions is met:

- For delayed write transactions:
  - — The transaction is being entered into the delayed transaction queue.
  - — The transaction has already been entered into the delayed transaction queue, but target response has not yet been received.
  - — Target response has been received but has not progressed to the head of the return queue.
  - — The delayed transaction queue is full, and the transaction cannot be queued.
  - — A transaction with the same address and command has been queued.
  - — A locked sequence is being propagated across the 21154, and the write transaction is not a locked transaction.
- For delayed read transactions:
  - — The transaction is being entered into the delayed transaction queue.
  - — The read request has already been queued, but read data is not yet available.
  - — Data has been read from the target, but it is not yet at the head of the read data queue, or a posted write transaction precedes it.
  - — The delayed transaction queue is full, and the transaction cannot be queued.
  - — A delayed read request with the same address and bus command has already been queued.
  - — A locked sequence is being propagated across the 21154, and the read transaction is not a locked transaction.
  - — The 21154 is currently discarding previously prefetched read data.
- For posted write transactions:

— The posted write data buffer does not have enough space for address and at least 8 Dwords of write data.

— A locked sequence is being propagated across the 21154, and the write transaction is not a locked transaction.

When a target retry is returned to the initiator of a delayed transaction, the initiator must repeat the transaction with the same address and bus command as well as the data if this is a write transaction, within the time frame specified by the master timeout value; otherwise, the transaction is discarded from the 21154 buffers.

#### 4.10.4.2 Target Disconnect

The 21154 returns a target disconnect to an initiator when one of the following conditions is met:

- The 21154 hits an internal address boundary
- The 21154 cannot accept any more write data
- The 21154 has no more read data to deliver

See Section 4.5.4 for a description of write address boundaries, and Section 4.6.3 for a description of read address boundaries.

#### 4.10.4.3 Target Abort

The 21154 returns a target abort to an initiator when one of the following conditions is met:

- The 21154 is returning a target abort from the intended target.
- The 21154 is unable to obtain delayed read data from the target or to deliver delayed write data to the target after $2^{24}$ attempts.

When the 21154 returns a target abort to the initiator, it sets the signaled target abort bit in the status register corresponding to the initiator interface.

# 5.0 Address Decoding

The 21154 uses three address ranges that control I/O and memory transaction forwarding. These address ranges are defined by base and limit address registers in the 21154 configuration space.

This chapter describes these address ranges, as well as ISA-mode and VGA-addressing support.

## 5.1 Address Ranges

The 21154 uses the following address ranges that determine which I/O and memory transactions are forwarded from the primary PCI bus to the secondary PCI bus, and from the secondary bus to the primary bus:

- One 32-bit I/O address range
- One 32-bit memory-mapped I/O (nonprefetchable memory)
- One 64-bit prefetchable memory address range

Transactions falling within these ranges are forwarded downstream from the primary PCI bus to the secondary PCI bus. Transactions falling outside these ranges are forwarded upstream from the secondary PCI bus to the primary PCI bus.

The 21154 uses a flat address space; that is, it does not perform any address translations. The address space has no “gaps”—addresses that are not marked for downstream forwarding are always forwarded upstream.

## 5.2 I/O Address Decoding

The 21154 uses the following mechanisms that are defined in the 21154 configuration space to specify the I/O address space for downstream and upstream forwarding:

- I/O base and limit address registers
- The ISA enable bit
- The VGA mode bit
- The VGA snoop bit

This section provides information on the I/O address registers and ISA mode. Section 5.4 provides information on the VGA modes.

To enable downstream forwarding of I/O transactions, the I/O enable bit must be set in the command register in 21154 configuration space. If the I/O enable bit is not set, all I/O transactions initiated on the primary bus are ignored. To enable upstream forwarding of I/O transactions, the master enable bit must be set in the command register. If the master enable bit is not set, the 21154 ignores all I/O and memory transactions initiated on the secondary bus. Setting the master enable bit also allows upstream forwarding of memory transactions.

***Caution:*** If any 21154 configuration state affecting I/O transaction forwarding is changed by a configuration write operation on the primary bus at the same time that I/O transactions are ongoing on the secondary bus, the 21154 response to the secondary bus I/O transactions is not predictable.

Configure the I/O base and limit address registers, ISA enable bit, VGA mode bit, and VGA snoop bit before setting the I/O enable and master enable bits, and change them subsequently only when the primary and secondary PCI buses are idle.

### 5.2.1 I/O Base and Limit Address Registers

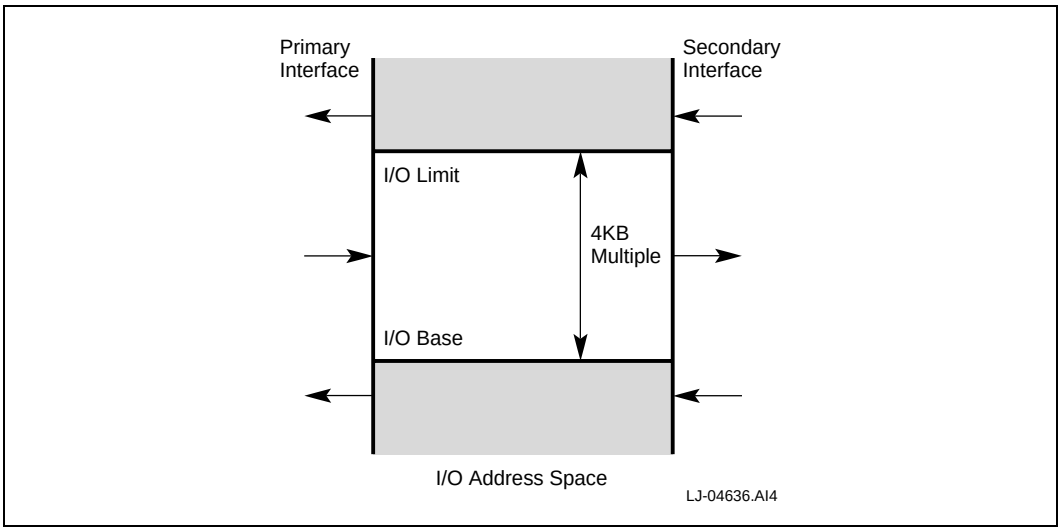
The 21154 implements one set of I/O base and limit address registers in configuration space that define an I/O address range for downstream forwarding. The 21154 supports 32-bit I/O addressing, which allows I/O addresses downstream of the 21154 to be mapped anywhere in a 4GB I/O address space.

I/O transactions with addresses that fall inside the range defined by the I/O base and limit registers are forwarded downstream from the primary PCI bus to the secondary PCI bus. I/O transactions with addresses that fall outside this range are forwarded upstream from the secondary PCI bus to the primary PCI bus.

The I/O range can be turned off by setting the I/O base address to a value greater than that of the I/O limit address. When the I/O range is turned off, all I/O transactions are forwarded upstream, and no I/O transactions are forwarded downstream.

Figure 15 illustrates transaction forwarding within and outside the I/O address range.

**Figure 15. I/O Transaction Forwarding Using Base and Limit Addresses**




The 21154 I/O range has a minimum granularity of 4KB and is aligned on a 4KB boundary. The maximum I/O range is 4GB in size.

The I/O base register consists of an 8-bit field at configuration address 1Ch, and a 16-bit field at address 30h. The top 4 bits of the 8-bit field define bits <15:12> of the I/O base address. The bottom 4 bits read only as 1h to indicate that the 21154 supports 32-bit I/O addressing. Bits <11:0> of the base address are assumed to be 0, which naturally aligns the base address to a 4KB boundary. The 16 bits contained in the I/O base upper 16 bits register at configuration offset 30h define AD<31:16> of the I/O base address. All 16 bits are read/write. After primary bus reset or chip reset, the value of the I/O base address is initialized to 0000 0000h.

The I/O limit register consists of an 8-bit field at configuration offset 1Dh and a 16-bit field at offset 32h. The top 4 bits of the 8-bit field define bits <15:12> of the I/O limit address. The bottom 4 bits read only as 1h to indicate that 32-bit I/O addressing is supported. Bits <11:0> of the limit address are assumed to be FFFh, which naturally aligns the limit address to the top of a 4KB I/O address block. The 16 bits contained in the I/O limit upper 16 bits register at configuration offset 32h define AD<31:16> of the I/O limit address. All 16 bits are read/write. After primary bus reset or chip reset, the value of the I/O limit address is reset to 0000 0FFFh.

***Note:*** The initial states of the I/O base and I/O limit address registers define an I/O range of 0000 0000h to 0000 0FFFh, which is the bottom 4KB of I/O space. Write these registers with their appropriate values before setting either the I/O enable bit or the master enable bit in the command register in configuration space.

### 5.2.2 ISA Mode

The 21154 supports ISA mode by providing an ISA enable bit in the bridge control register in configuration space. ISA mode modifies the response of the 21154 inside the I/O address range in order to support mapping of I/O space in the presence of an ISA bus in the system. This bit only affects the response of the 21154 when the transaction falls inside the address range defined by the I/O base and limit address registers, and only when this address also falls inside the first 64KB of I/O space (address bits <31:16> are 0000h).

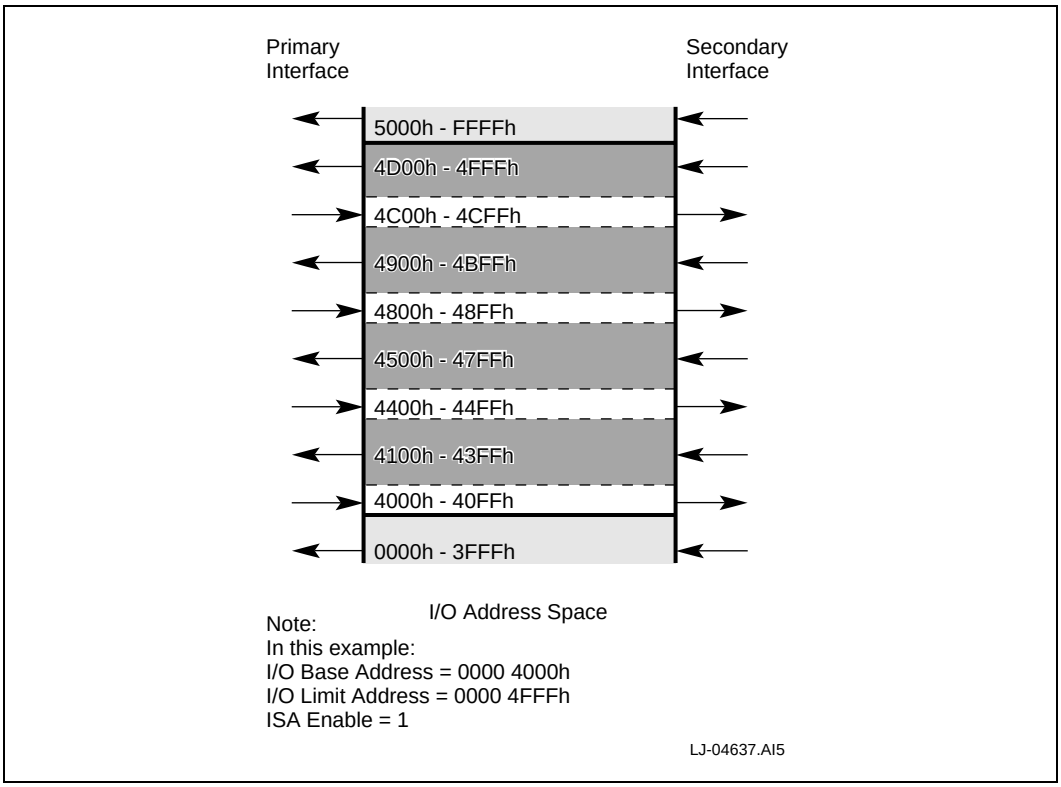
When the ISA enable bit is set, the 21154 does not forward downstream any I/O transactions addressing the top 768 bytes of each aligned 1KB block. Only those transactions addressing the bottom 256 bytes of an aligned 1KB block inside the base and limit I/O address range are forwarded downstream. Transactions above the 64KB I/O address boundary are forwarded as defined by the address range defined by the I/O base and limit registers.

Accordingly, if the ISA enable bit is set, the 21154 forwards upstream those I/O transactions addressing the top 768 bytes of each aligned 1KB block within the first 64KB of I/O space. The master enable bit in the command configuration register must also be set to enable upstream forwarding. All other I/O transactions initiated on the secondary bus are forwarded upstream only if they fall outside the I/O address range.

When the ISA enable bit is set, devices downstream of the 21154 can have I/O space mapped into the first 256 bytes of each 1KB chunk below the 64KB boundary, or anywhere in I/O space above the 64KB boundary.

Figure 16 illustrates I/O forwarding when the ISA enable bit is set.

**Figure 16. I/O Transaction Forwarding in ISA Mode**




## 5.3 Memory Address Decoding

The 21154 has three mechanisms for defining memory address ranges for forwarding of memory transactions:

- Memory-mapped I/O base and limit address registers
- Prefetchable memory base and limit address registers
- VGA mode

This section describes the first two mechanisms. Section 5.4.1 describes VGA mode.

To enable downstream forwarding of memory transactions, the memory enable bit must be set in the command register in 21154 configuration space. To enable upstream forwarding of memory transactions, the master enable bit must be set in the command register. Setting the master enable bit also allows upstream forwarding of I/O transactions.

***Caution:*** If any 21154 configuration state affecting memory transaction forwarding is changed by a configuration write operation on the primary bus at the same time that memory transactions are ongoing on the secondary bus, the 21154 response to the secondary bus memory transactions is not

predictable. Configure the memory-mapped I/O base and limit address registers, prefetchable memory base and limit address registers, and VGA mode bit before setting the memory enable and master enable bits, and change them subsequently only when the primary and secondary PCI buses are idle.

### 5.3.1 Memory-Mapped I/O Base and Limit Address Registers

Memory-mapped I/O is also referred to as nonprefetchable memory. Memory addresses that cannot automatically be prefetched but that can conditionally prefetch based on command type should be mapped into this space. Read transactions to nonprefetchable space may exhibit side effects; this space may have non-memory-like behavior. The 21154 prefetches in this space only if the memory read line or memory read multiple commands are used; transactions using the memory read command are limited to a single data transfer.

The memory-mapped I/O base address and memory-mapped I/O limit address registers define an address range that the 21154 uses to determine when to forward memory commands. The 21154 forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the memory-mapped I/O address range. The 21154 ignores memory transactions initiated on the secondary interface that fall into this address range. Any transactions that fall outside this address range are ignored on the primary interface and are forwarded upstream from the secondary interface (provided that they do not fall into the prefetchable memory range or are not forwarded downstream by the VGA mechanism).

The memory-mapped I/O range supports 32-bit addressing only. The *PCI-to-PCI Bridge Architecture Specification* does not provide for 64-bit addressing in the memory-mapped I/O space. The memory-mapped I/O address range has a granularity and alignment of 1MB. The maximum memory-mapped I/O address range is 4GB.

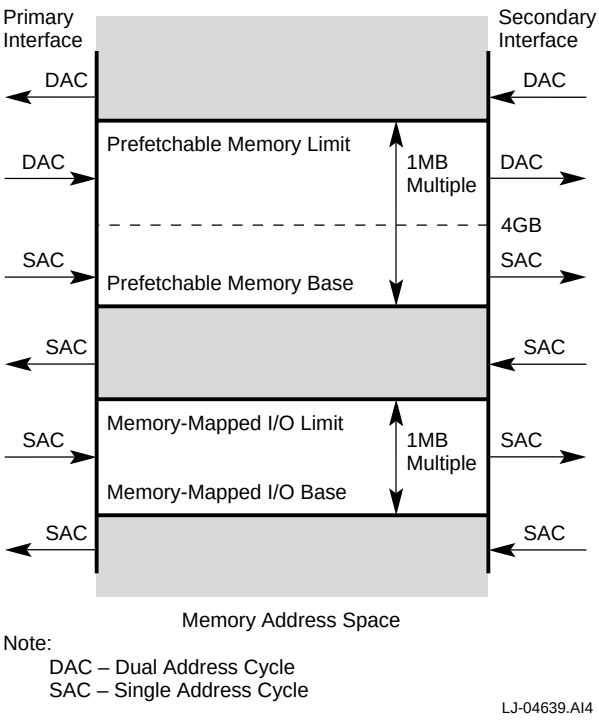
The memory-mapped I/O address range is defined by a 16-bit memory-mapped I/O base address register at configuration offset 20h and by a 16-bit memory-mapped I/O limit address register at offset 22h. The top 12 bits of each of these registers correspond to bits <31:20> of the memory address. The low 4 bits are hardwired to 0. The low 20 bits of the memory-mapped I/O base address are assumed to be 0 0000h, which results in a natural alignment to a 1MB boundary. The low 20 bits of the memory-mapped I/O limit address are assumed to be F FFFFh, which results in an alignment to the top of a 1MB block.

***Note:*** The initial state of the memory-mapped I/O base address register is 0000 0000h. The initial state of the memory-mapped I/O limit address register is 000F FFFFh. Note that the initial states of these registers define a memory-mapped I/O range at the bottom 1MB block of memory. Write these registers with their appropriate values before setting either the memory enable bit or the master enable bit in the command register in configuration space.

To turn off the memory-mapped I/O address range, write the memory-mapped I/O base address register with a value greater than that of the memory-mapped I/O limit address register.

Figure 17 shows how transactions are forwarded using both the memory-mapped I/O range and the prefetchable memory range.

**Figure 17. Memory Transaction Forwarding Using Base and Limit Registers**




## 5.3.2 Prefetchable Memory Base and Limit Address Registers

Locations accessed in the prefetchable memory address range must have true memory-like behavior and must not exhibit side effects when read. This means that extra reads to a prefetchable memory location must have no side effects. The 21154 prefetches for all types of memory read commands in this address space.

The prefetchable memory base address and prefetchable memory limit address registers define an address range that the 21154 uses to determine when to forward memory commands. The 21154 forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the prefetchable memory address range. The 21154 ignores memory transactions initiated on the secondary interface that fall into this address range. The 21154 does not respond to any transactions that fall outside this address range on the primary interface and forwards those transactions upstream from the secondary interface (provided that they do not fall into the memory-mapped I/O range or are not forwarded by the VGA mechanism).

The prefetchable memory range supports 64-bit addressing and provides additional registers to define the upper 32 bits of the memory address range, the prefetchable memory base address upper 32 bits register, and the prefetchable memory limit address upper 32 bits register. For address comparison, a single address cycle (32-bit address) prefetchable memory transaction is treated like a 64-bit address transaction where the upper 32 bits of the address are equal to 0. This upper 32-bit value of 0 is compared to the prefetchable memory base address upper 32 bits register and the

prefetchable memory limit address upper 32 bits register. The prefetchable memory base address upper 32 bits register must be 0 in order to pass any single address cycle transactions downstream. Section 5.3.3 further describes 64-bit addressing support.

The prefetchable memory address range has a granularity and alignment of 1MB. The maximum memory address range is 4GB when 32-bit addressing is used, and $2^{64}$ bytes when 64-bit addressing is used.

The prefetchable memory address range is defined by a 16-bit prefetchable memory base address register at configuration offset 24h and by a 16-bit prefetchable memory limit address register at offset 28h. The top 12 bits of each of these registers correspond to bits <31:20> of the memory address. The low 4 bits are hardwired to 1h, indicating 64-bit address support. The low 20 bits of the prefetchable memory base address are assumed to be 0 0000h, which results in a natural alignment to a 1MB boundary. The low 20 bits of the prefetchable memory limit address are assumed to be F FFFFh, which results in an alignment to the top of a 1MB block.

***Note:*** The initial state of the prefetchable memory base address register is 0000 0000h. The initial state of the prefetchable memory limit address register is 000F FFFFh. Note that the initial states of these registers define a prefetchable memory range at the bottom 1MB block of memory. Write these registers with their appropriate values before setting either the memory enable bit or the master enable bit in the command register in configuration space.

To turn off the prefetchable memory address range, write the prefetchable memory base address register with a value greater than that of the prefetchable memory limit address register. The entire base value must be greater than the entire limit value, meaning that the upper 32 bits must be considered. Therefore, to disable the address range, the upper 32 bits registers can both be set to the same value, while the lower base register is set greater than the lower limit register; otherwise, the upper 32-bit base must be greater than the upper 32-bit limit.

### 5.3.3 Prefetchable Memory 64-Bit Addressing Registers

The 21154 supports 64-bit memory address decoding for forwarding of dual address memory transactions. The dual address cycle is used to support 64-bit addressing. The first address phase of a dual address transaction contains the low 32 address bits, and the second address phase contains the high 32 address bits. During a dual address cycle transaction, the upper 32 bits must never be 0—use the single address cycle commands for transactions addressing the first 4GB of memory space.

The 21154 implements the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register to define a prefetchable memory address range greater than 4GB. The prefetchable address space can then be defined in three different ways:

- Residing entirely in the first 4GB of memory
- Residing entirely above the first 4GB of memory
- Crossing the first 4GB memory boundary

If the prefetchable memory space on the secondary interface resides entirely in the first 4GB of memory, both upper 32 bits registers must be set to 0. The 21154 ignores all dual address cycle transactions initiated on the primary interface and forwards all dual address transactions initiated on the secondary interface upstream.

If the secondary interface prefetchable memory space resides entirely above the first 4GB of memory, both the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register must be initialized to nonzero values. The 21154 ignores all single address memory transactions initiated on the primary interface and forwards all single address memory transactions initiated on the secondary interface upstream (unless they fall within the memory-mapped I/O or VGA memory range). A dual address memory transaction is forwarded downstream from the primary interface if it falls within the address range defined by the prefetchable memory base address, prefetchable memory base address upper 32 bits, prefetchable memory limit address, and prefetchable memory limit address upper 32 bits registers. If the dual address transaction initiated on the secondary interface falls outside this address range, it is forwarded upstream to the primary interface. The 21154 does not respond to a dual address transaction initiated on the primary interface that falls outside this address range, or to a dual address transaction initiated on the secondary interface that falls within the address range.

If the secondary interface prefetchable memory space straddles the first 4GB address boundary, the prefetchable memory base address upper 32 bits register is set to 0, while the prefetchable memory limit address upper 32 bits register is initialized to a nonzero value. Single address cycle memory transactions are compared to the prefetchable memory base address register only. A transaction initiated on the primary interface is forwarded downstream if the address is greater than or equal to the base address. A transaction initiated on the secondary interface is forwarded upstream if the address is less than the base address. Dual address transactions are compared to the prefetchable memory limit address and the prefetchable memory limit address upper 32 bits registers. If the address of the dual address transaction is less than or equal to the limit, the transaction is forwarded downstream from the primary interface and is ignored on the secondary interface. If the address of the dual address transaction is greater than this limit, the transaction is ignored on the primary interface and is forwarded upstream from the secondary interface.

The prefetchable memory base address upper 32 bits register is located at configuration Dword offset 28h, and the prefetchable memory limit address upper 32 bits register is located at configuration Dword offset 2Ch. Both registers are reset to 0. See Figure 17 for an illustration of how transactions are forwarded using both the memory-mapped I/O range and the prefetchable memory range.

## 5.4 VGA Support

The 21154 provides two modes for VGA support:

- VGA mode, supporting VGA-compatible addressing
- VGA snoop mode, supporting VGA palette forwarding

### 5.4.1 VGA Mode

When a VGA-compatible device exists downstream from the 21154, set the VGA mode bit in the bridge control register in configuration space to enable VGA mode. When the 21154 is operating in VGA mode, it forwards downstream those transactions addressing the VGA frame buffer memory and VGA I/O registers, regardless of the values of the 21154 base and limit address registers. The 21154 ignores transactions initiated on the secondary interface addressing these locations.

The VGA frame buffer consists of the following memory address range:

```
000A 0000h-000B FFFFh
```

Read transactions to frame buffer memory are treated as nonprefetchable. The 21154 requests only a single data transfer from the target, and read byte enable bits are forwarded to the target bus.

The VGA I/O addresses consist of the following I/O addresses:

- 3B0h–3BBh
- 3C0h–3DFh

These I/O addresses are aliased every 1KB throughout the first 64KB of I/O space. This means that address bits <15:10> are not decoded and can be any value, while address bits <31:16> must be all 0s.

VGA BIOS addresses starting at C0000h are not decoded in VGA mode.

### 5.4.2 VGA Snoop Mode

The 21154 provides VGA snoop mode, allowing for VGA palette write transactions to be forwarded downstream. This mode is used when a graphics device downstream from the 21154 needs to snoop or respond to VGA palette write transactions. To enable the mode, set the VGA snoop bit in the command register in configuration space.

Note that the 21154 claims VGA palette write transactions by asserting DEVSEL# in VGA snoop mode.

When the VGA snoop bit is set, the 21154 forwards downstream transactions with the following I/O addresses:

- 3C6h
- 3C8h
- 3C9h

Note that these addresses are also forwarded as part of the VGA compatibility mode previously described. Again, address bits <15:10> are not decoded, while address bits <31:16> must be equal to 0, which means that these addresses are aliased every 1KB throughout the first 64KB of I/O space.

***Note:*** If both the VGA mode bit and the VGA snoop bit are set, the 21154 behaves in the same way as if only the VGA mode bit were set.

# 6.0 Transaction Ordering

To maintain data coherency and consistency, the 21154 complies with the ordering rules set forth in the *PCI Local Bus Specification, Revision 2.1*, for transactions crossing the bridge.

This chapter describes the ordering rules that control transaction forwarding across the 21154. For a more detailed discussion of transaction ordering, see Appendix E of the *PCI Local Bus Specification, Revision 2.1*.

## 6.1 Transactions Governed by Ordering Rules

Ordering relationships are established for the following classes of transactions crossing the 21154:

- Posted write transactions, comprised of memory write and memory write and invalidate transactions

  Posted write transactions complete at the source before they complete at the destination; that is, data is written into intermediate data buffers before it reaches the target.

- Delayed write request transactions, comprised of I/O write and configuration write transactions

  Delayed write requests are terminated by target retry on the initiator bus and are queued in the delayed transaction queue. A delayed write transaction must complete on the target bus before it completes on the initiator bus.

- Delayed write completion transactions, also comprised of I/O write and configuration write transactions

  Delayed write completion transactions have been completed on the target bus, and the target response is queued in the 21154 buffers. A delayed write completion transaction proceeds in the direction opposite that of the original delayed write request; that is, a delayed write completion transaction proceeds from the target bus to the initiator bus.

- Delayed read request transactions, comprised of all memory read, I/O read, and configuration read transactions

  Delayed read requests are terminated by target retry on the initiator bus and are queued in the delayed transaction queue.

- Delayed read completion transactions, comprised of all memory read, I/O read, and configuration read transactions

  Delayed read completion transactions have been completed on the target bus, and the read data has been queued in the 21154 read data buffers. A delayed read completion transaction proceeds in the direction opposite that of the original delayed read request; that is, a delayed read completion transaction proceeds from the target bus to the initiator bus.

The 21154 does not combine or merge write transactions:

- The 21154 does not combine separate write transactions into a single write transaction—this optimization is best implemented in the originating master.
- The 21154 does not merge bytes on separate masked write transactions to the same Dword address—this optimization is also best implemented in the originating master.
- The 21154 does not collapse sequential write transactions to the same address into a single write transaction—the *PCI Local Bus Specification* does not permit this combining of transactions.

## 6.2 General Ordering Guidelines

Independent transactions on the primary and secondary buses have a relationship only when those transactions cross the 21154.

The following general ordering guidelines govern transactions crossing the 21154:

- The ordering relationship of a transaction with respect to other transactions is determined when the transaction completes, that is, when a transaction ends with a termination other than target retry.
- Requests terminated with target retry can be accepted and completed in any order with respect to other transactions that have been terminated with target retry. If the order of completion of delayed requests is important, the initiator should not start a second delayed transaction until the first one has been completed. If more than one delayed transaction is initiated, the initiator should repeat all the delayed transaction requests, using some fairness algorithm. Repeating a delayed transaction cannot be contingent on completion of another delayed transaction; otherwise, a deadlock can occur.
- Write transactions flowing in one direction have no ordering requirements with respect to write transactions flowing in the other direction. The 21154 can accept posted write transactions on both interfaces at the same time, as well as initiate posted write transactions on both interfaces at the same time.
- The acceptance of a posted memory write transaction as a target can never be contingent on the completion of a nonlocked, nonposted transaction as a master. This is true of the 21154 and must also be true of other bus agents; otherwise, a deadlock can occur.
- The 21154 accepts posted write transactions, regardless of the state of completion of any delayed transactions being forwarded across the 21154.

## 6.3 Ordering Rules

Table 25 shows the ordering relationships of all the transactions and refers by number to the ordering rules that follow.

**Table 25. Summary of Transaction Ordering**

| ↓ Pass→ | Posted Write | Delayed Read Request | Delayed Write Request | Delayed Read Completion | Delayed Write Completion |
|---|---|---|---|---|---|
| Posted write | No[1] | Yes[5] | Yes[5] | Yes[5] | Yes[5] |
| Delayed read request | No[2] | No | No | Yes | Yes |
| Delayed write request | No[4] | No | No | Yes | Yes |
| Delayed read completion | No[3] | Yes | Yes | No | No |
| Delayed write completion | Yes | Yes | Yes | No | No |

***Note:*** The superscript accompanying some of the table entries refers to any applicable ordering rule listed in this section. Many entries are not governed by these ordering rules; therefore, the

implementation can choose whether or not the transactions pass each other. The entries without superscripts reflect the 21154's implementation choices.

The following ordering rules describe the transaction relationships. Each ordering rule is followed by an explanation, and the ordering rules are referred to by number in Table 25. These ordering rules apply to posted write transactions, delayed write and read requests, and delayed write and read completion transactions crossing the 21154 in the same direction. Note that delayed completion transactions cross the 21154 in the direction opposite that of the corresponding delayed requests.

1. Posted write transactions must complete on the target bus in the order in which they were received on the initiator bus.

   The subsequent posted write transaction can be setting a flag that covers the data in the first posted write transaction; if the second transaction were to complete before the first transaction, a device checking the flag could subsequently consume stale data.

2. A delayed read request traveling in the same direction as a previously queued posted write transaction must push the posted write data ahead of it. The posted write transaction must complete on the target bus before the delayed read request can be attempted on the target bus.

   The read transaction can be to the same location as the write data, so if the read transaction were to pass the write transaction, it would return stale data.

3. A delayed read completion must "pull" ahead of previously queued posted write data traveling in the same direction. In this case, the read data is traveling in the same direction as the write data, and the initiator of the read transaction is on the same side of the 21154 as the target of the write transaction. The posted write transaction must complete to the target before the read data is returned to the initiator.

   The read transaction can be to a status register of the initiator of the posted write data and therefore should not complete until the write transaction is complete.

4. Delayed write requests cannot pass previously queued posted write data.

   As in the case of posted memory write transactions, the delayed write transaction can be setting a flag that covers the data in the posted write transaction; if the delayed write request were to complete before the earlier posted write transaction, a device checking the flag could subsequently consume stale data.

5. Posted write transactions must be given opportunities to pass delayed read and write requests and completions.

   Otherwise, deadlocks may occur when bridges that support delayed transactions are used in the same system with bridges that do not support delayed transactions. A fairness algorithm is used to arbitrate between the posted write queue and the delayed transaction queue.

## 6.4 Data Synchronization

Data synchronization refers to the relationship between interrupt signaling and data delivery. The *PCI Local Bus Specification, Revision 2.1*, provides the following alternative methods for synchronizing data and interrupts.

- The device signaling the interrupt performs a read of the data just written (software).
- The device driver performs a read operation to any register in the interrupting device before accessing data written by the device (software).
- System hardware guarantees that write buffers are flushed before interrupts are forwarded.

The 21154 does not have a hardware mechanism to guarantee data synchronization for posted write transactions. Therefore, all posted write transactions must be followed by a read operation, either from the device to the location just written (or some other location along the same path), or from the device driver to one of the device registers.

# 7.0 Error Handling

The 21154 checks, forwards, and generates parity on both the primary and secondary interfaces. To maintain transparency, the 21154 always tries to forward the existing parity condition on one bus to the other bus, along with address and data. The 21154 always attempts to be transparent when reporting errors, but this is not always possible, given the presence of posted data and delayed transactions.

To support error reporting on the PCI bus, the 21154 implements the following:

- PERR# and SERR# signals on both the primary and secondary interfaces
- Primary status and secondary status registers
- The device-specific p_serr_l event disable register
- The device-specific p_serr_l status register

This chapter provides detailed information about how the 21154 handles errors. It also describes error status reporting and error operation disabling.

## 7.1 Address Parity Errors

The 21154 checks address parity for all transactions on both buses, for all address and all bus commands.

When the 21154 detects an address parity error on the primary interface, the following events occur:

- If the parity error response bit is set in the command register, the 21154 does not claim the transaction with p_devsel_l; this may allow the transaction to terminate in a master abort.

  If the parity error response bit is not set, the 21154 proceeds normally and accepts the transaction if it is directed to or across the 21154.
- The 21154 sets the detected parity error bit in the status register.
- The 21154 asserts p_serr_l and sets the signaled system error bit in the status register, if both of the following conditions are met:
  - — The SERR# enable bit is set in the command register.
  - — The parity error response bit is set in the command register.

When the 21154 detects an address parity error on the secondary interface, the following events occur:

- If the parity error response bit is set in the bridge control register, the 21154 does not claim the transaction with s_devsel_l; this may allow the transaction to terminate in a master abort.

  If the parity error response bit is not set, the 21154 proceeds normally and accepts the transaction if it is directed to or across the 21154.
- The 21154 sets the detected parity error bit in the secondary status register.
- The 21154 asserts p_serr_l and sets the signaled system error bit in the status register, if both of the following conditions are met:
  - — The SERR# enable bit is set in the command register.

— The parity error response bit is set in the bridge control register.

## 7.2 Data Parity Errors

When forwarding transactions, the 21154 attempts to pass the data parity condition from one interface to the other unchanged, whenever possible, to allow the master and target devices to handle the error condition.

The following sections describe, for each type of transaction, the sequence of events that occurs when a parity error is detected and the way in which the parity condition is forwarded across the 21154.

### 7.2.1 Configuration Write Transactions to 21154 Configuration Space

When the 21154 detects a data parity error during a Type 0 configuration write transaction to 21154 configuration space, the following events occur:

- If the parity error response bit is set in the command register, the 21154 asserts p_trdy_l and writes the data to the configuration register. The 21154 also asserts p_perr_l.

  If the parity error response bit is not set, the 21154 does not assert p_perr_l.
- The 21154 sets the detected parity error bit in the status register, regardless of the state of the parity error response bit.

### 7.2.2 Read Transactions

When the 21154 detects a parity error during a read transaction, the target drives data and data parity, and the initiator checks parity and conditionally asserts PERR#.

For downstream transactions, when the 21154 detects a read data parity error on the secondary bus, the following events occur:

- The 21154 asserts s_perr_l two cycles following the data transfer, if the secondary interface parity error response bit is set in the bridge control register.
- The 21154 sets the detected parity error bit in the secondary status register.
- The 21154 sets the data parity detected bit in the secondary status register, if the secondary interface parity error response bit is set in the bridge control register.
- The 21154 forwards the bad parity with the data back to the initiator on the primary bus.

  If the data with the bad parity is prefetched and is not read by the initiator on the primary bus, the data is discarded and the data with bad parity is not returned to the initiator.
- The 21154 completes the transaction normally.

For upstream transactions, when the 21154 detects a read data parity error on the primary bus, the following events occur:

- The 21154 asserts p_perr_l two cycles following the data transfer, if the primary interface parity error response bit is set in the command register.
- The 21154 sets the detected parity error bit in the primary status register.

- The 21154 sets the data parity detected bit in the primary status register, if the primary interface parity error response bit is set in the command register.
- The 21154 forwards the bad parity with the data back to the initiator on the secondary bus.

  If the data with the bad parity is prefetched and is not read by the initiator on the secondary bus, the data is discarded and the data with bad parity is not returned to the initiator.
- The 21154 completes the transaction normally.

The 21154 returns to the initiator the data and parity that was received from the target. When the initiator detects a parity error on this read data and is enabled to report it, the initiator asserts PERR# two cycles after the data transfer occurs. It is assumed that the initiator takes responsibility for handling a parity error condition; therefore, when the 21154 detects PERR# asserted while returning read data to the initiator, the 21154 does not take any further action and completes the transaction normally.

### 7.2.3 Delayed Write Transactions

When the 21154 detects a data parity error during a delayed write transaction, the initiator drives data and data parity, and the target checks parity and conditionally asserts PERR#.

For delayed write transactions, a parity error can occur at the following times:

- During the original delayed write request transaction
- When the initiator repeats the delayed write request transaction
- When the 21154 completes the delayed write transaction to the target

When a delayed write transaction is normally queued, the address, command, address parity, data, byte enable bits, and data parity are all captured and a target retry is returned to the initiator. When the 21154 detects a parity error on the write data for the initial delayed write request transaction, the following events occur:

- If the parity error response bit corresponding to the initiator bus is set, the 21154 asserts TRDY# to the initiator and the transaction is not queued. If multiple data phases are requested, STOP# is also asserted to cause a target disconnect. Two cycles after the data transfer, the 21154 also asserts PERR#.

  If the parity error response bit is not set, the 21154 returns a target retry and queues the transaction as usual. Signal PERR# is not asserted. In this case, the initiator repeats the transaction.
- The 21154 sets the detected parity error bit in the status register corresponding to the initiator bus, regardless of the state of the parity error response bit.

***Note:*** If parity checking is turned off and data parity errors have occurred for queued or subsequent delayed write transactions on the initiator bus, it is possible that the initiator's reattempts of the write transaction may not match the original queued delayed write information contained in the delayed transaction queue. In this case, a master timeout condition may occur, possibly resulting in a system error (p_serr_l assertion).

For downstream transactions, when the 21154 is delivering data to the target on the secondary bus and s_perr_l is asserted by the target, the following events occur:

- The 21154 sets the secondary interface data parity detected bit in the secondary status register, if the secondary parity error response bit is set in the bridge control register.

- The 21154 captures the parity error condition to forward it back to the initiator on the primary bus.

Similarly, for upstream transactions, when the 21154 is delivering data to the target on the primary bus and p_perr_l is asserted by the target, the following events occur:

- The 21154 sets the primary interface data parity detected bit in the status register, if the primary parity error response bit is set in the command register.
- The 21154 captures the parity error condition to forward it back to the initiator on the secondary bus.

A delayed write transaction is completed on the initiator bus when the initiator repeats the write transaction with the same address, command, data, and byte enable bits as the delayed write command that is at the head of the posted data queue. Note that the parity bit is not compared when determining whether the transaction matches those in the delayed transaction queues.

Two cases must be considered:

- When parity error is detected on the initiator bus on a subsequent reattempt of the transaction and was not detected on the target bus
- When parity error is forwarded back from the target bus

For downstream delayed write transactions, when the parity error is detected on the initiator bus and the 21154 has write status to return, the following events occur:

- The 21154 first asserts p_trdy_l and then asserts p_perr_l two cycles later, if the primary interface parity error response bit is set in the command register.
- The 21154 sets the primary interface parity error detected bit in the status register.
- Because there was not an exact data and byte enable match, the write status is not returned and the transaction remains in the queue.

Similarly, for upstream delayed write transactions, when the parity error is detected on the initiator bus and the 21154 has write status to return, the following events occur:

- The 21154 first asserts s_trdy_l and then asserts s_perr_l two cycles later, if the secondary interface parity error response bit is set in the bridge control register.
- The 21154 sets the secondary interface parity error detected bit in the secondary status register.
- Because there was not an exact data and byte enable match, the write status is not returned and the transaction remains in the queue.

For downstream transactions, in the case where the parity error is being passed back from the target bus and the parity error condition was not originally detected on the initiator bus, the following events occur:

- The 21154 asserts p_perr_l two cycles after the data transfer, if both of the following are true:
  - — The primary interface parity error response bit is set in the command register.
  - — The secondary interface parity error response bit is set in the bridge control register.
- The 21154 completes the transaction normally.

  For upstream transactions, in the case where the parity error is being passed back from the target bus and the parity error condition was not originally detected on the initiator bus, the following events occur:
- The 21154 asserts s_perr_l two cycles after the data transfer, if both of the following are true:

  - The primary interface parity error response bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.
- The 21154 completes the transaction normally.

### 7.2.4 Posted Write Transactions

During downstream posted write transactions, when the 21154, responding as a target, detects a data parity error on the initiator (primary) bus, the following events occur:

- The 21154 asserts p_perr_l two cycles after the data transfer, if the primary interface parity error response bit is set in the command register.
- The 21154 sets the primary interface parity error detected bit in the status register.
- The 21154 captures and forwards the bad parity condition to the secondary bus.
- The 21154 completes the transaction normally.

Similarly, during upstream posted write transactions, when the 21154, responding as a target, detects a data parity error on the initiator (secondary) bus, the following events occur:

- The 21154 asserts s_perr_l two cycles after the data transfer, if the secondary interface parity error response bit is set in the bridge control register.
- The 21154 sets the secondary interface parity error detected bit in the secondary status register.
- The 21154 captures and forwards the bad parity condition to the primary bus.
- The 21154 completes the transaction normally.

During downstream write transactions, when a data parity error is reported on the target (secondary) bus by the target's assertion of s_perr_l, the following events occur:

- The 21154 sets the data parity detected bit in the secondary status register, if the secondary interface parity error response bit is set in the bridge control register.
- The 21154 asserts p_serr_l and sets the signaled system error bit in the status register, if all of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The device-specific p_serr_l disable bit for posted write parity errors is not set.
  - The secondary interface parity error response bit is set in the bridge control register.
  - The primary interface parity error response bit is set in the command register.
  - The 21154 did not detect the parity error on the primary (initiator) bus; that is, the parity error was not forwarded from the primary bus.

During upstream write transactions, when a data parity error is reported on the target (primary) bus by the target's assertion of p_perr_l, the following events occur:

- The 21154 sets the data parity detected bit in the status register, if the primary interface parity error response bit is set in the command register.
- The 21154 asserts p_serr_l and sets the signaled system error bit in the status register, if all of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.

— The primary interface parity error response bit is set in the command register.

— The 21154 did not detect the parity error on the secondary (initiator) bus; that is, the parity error was not forwarded from the secondary bus.

The assertion of p_serr_l is used to signal the parity error condition in the case where the initiator does not know that the error occurred. Because the data has already been delivered with no errors, there is no other way to signal this information back to the initiator.

If the parity error was forwarded from the initiating bus to the target bus, p_serr_l is not asserted.

## 7.3 Data Parity Error Reporting Summary

In the previous sections, the 21154's responses to data parity errors are presented according to the type of transaction in progress. This section organizes the 21154's responses to data parity errors according to the status bits that the 21154 sets and the signals that it asserts.

Table 26 shows setting the detected parity error bit in the status register, corresponding to the primary interface. This bit is set when the 21154 detects a parity error on the primary interface.

**Table 26. Setting the Primary Interface Detected Parity Error Bit**

| Primary Detected Parity Error Bit | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 0 | Read | Downstream | Primary | x/x |
| 0 | Read | Downstream | Secondary | x/x |
| 1 | Read | Upstream | Primary | x/x |
| 0 | Read | Upstream | Secondary | x/x |
| 1 | Posted write | Downstream | Primary | x/x |
| 0 | Posted write | Downstream | Secondary | x/x |
| 0 | Posted write | Upstream | Primary | x/x |
| 0 | Posted write | Upstream | Secondary | x/x |
| 1 | Delayed write | Downstream | Primary | x/x |
| 0 | Delayed write | Downstream | Secondary | x/x |
| 0 | Delayed write | Upstream | Primary | x/x |
| 0 | Delayed write | Upstream | Secondary | x/x |

1. x — don't care.

Table 27 shows setting the detected parity error bit in the secondary status register, corresponding to the secondary interface. This bit is set when the 21154 detects a parity error on the secondary interface

**Table 27. Setting the Secondary Interface Detected Parity Error Bit**

| Secondary Detected Parity Error Bit | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 0 | Read | Downstream | Primary | x/x |
| 1 | Read | Downstream | Secondary | x/x |
| 0 | Read | Upstream | Primary | x/x |
| 0 | Read | Upstream | Secondary | x/x |
| 0 | Posted write | Downstream | Primary | x/x |
| 0 | Posted write | Downstream | Secondary | x/x |
| 0 | Posted write | Upstream | Primary | x/x |
| 1 | Posted write | Upstream | Secondary | x/x |
| 0 | Delayed write | Downstream | Primary | x/x |
| 0 | Delayed write | Downstream | Secondary | x/x |
| 0 | Delayed write | Upstream | Primary | x/x |
| 1 | Delayed write | Upstream | Secondary | x/x |

1. x — don't care.

Table 28 shows setting the data parity detected bit in the status register, corresponding to the primary interface. This bit is set under the following conditions:

- The 21154 must be a master on the primary bus.
- The parity error response bit in the command register, corresponding to the primary interface, must be set.
- The p_perr_l signal is detected asserted or a parity error is detected on the primary bus.

**Table 28. Setting the Primary Interface Data Parity Detected Bit**

| Primary Data Parity Detected Bit | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 0 | Read | Downstream | Primary | x/x |
| 0 | Read | Downstream | Secondary | x/x |
| 1 | Read | Upstream | Primary | 1/x |
| 0 | Read | Upstream | Secondary | x/x |
| 0 | Posted write | Downstream | Primary | x/x |
| 0 | Posted write | Downstream | Secondary | x/x |
| 1 | Posted write | Upstream | Primary | 1/x |
| 0 | Posted write | Upstream | Secondary | x/x |
| 0 | Delayed write | Downstream | Primary | x/x |
| 0 | Delayed write | Downstream | Secondary | x/x |
| 1 | Delayed write | Upstream | Primary | 1/x |
| 0 | Delayed write | Upstream | Secondary | x/x |

1. x — don't care.

Table 29 shows setting the data parity detected bit in the secondary status register, corresponding to the secondary interface. This bit is set under the following conditions:

- The 21154 must be a master on the secondary bus.
- The parity error response bit in the bridge control register, corresponding to the secondary interface, must be set.
- The s_perr_l signal is detected asserted or a parity error is detected on the secondary bus.

**Table 29. Setting the Secondary Interface Data Parity Detected Bit (Sheet 1 of 2)**

| Secondary Data Parity Detected Bit | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 0 | Read | Downstream | Primary | x/x |
| 1 | Read | Downstream | Secondary | x/1 |
| 0 | Read | Upstream | Primary | x/x |
| 0 | Read | Upstream | Secondary | x/x |
| 0 | Posted write | Downstream | Primary | x/x |
| 1 | Posted write | Downstream | Secondary | x/1 |
| 0 | Posted write | Upstream | Primary | x/x |
| 0 | Posted write | Upstream | Secondary | x/x |
| 0 | Delayed write | Downstream | Primary | x/x |

**Table 29. Setting the Secondary Interface Data Parity Detected Bit (Sheet 2 of 2)**

| Secondary Data Parity Detected Bit | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 1 | Delayed write | Downstream | Secondary | x/1 |
| 0 | Delayed write | Upstream | Primary | x/x |
| 0 | Delayed write | Upstream | Secondary | x/x |

1. x — don't care.

Table 30 shows assertion of p_perr_l. This signal is set under the following conditions:

- The 21154 is either the target of a write transaction or the initiator of a read transaction on the primary bus.
- The parity error response bit in the command register, corresponding to the primary interface, must be set.
- The 21154 detects a data parity error on the primary bus or detects s_perr_l asserted during the completion phase of a downstream delayed write transaction on the target (secondary) bus.

**Table 30. Assertion of p_perr_l**

| p_perr_l | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 1 (deasserted) | Read | Downstream | Primary | x/x |
| 1 | Read | Downstream | Secondary | x/x |
| 0 (asserted) | Read | Upstream | Primary | 1/x |
| 1 | Read | Upstream | Secondary | x/x |
| 0 | Posted write | Downstream | Primary | 1/x |
| 1 | Posted write | Downstream | Secondary | x/x |
| 1 | Posted write | Upstream | Primary | x/x |
| 1 | Posted write | Upstream | Secondary | x/x |
| 0 | Delayed write | Downstream | Primary | 1/x |
| 0[2] | Delayed write | Downstream | Secondary | 1/1 |
| 1 | Delayed write | Upstream | Primary | x/x |
| 1 | Delayed write | Upstream | Secondary | x/x |

1. x — don't care.
2. The parity error was detected on the target (secondary) bus but not on the initiator (primary) bus.

Table 31 shows assertion of s_perr_l. This signal is set under the following conditions:

- The 21154 is either the target of a write transaction or the initiator of a read transaction on the secondary bus.
- The parity error response bit in the bridge control register, corresponding to the secondary interface, must be set.
- The 21154 detects a data parity error on the secondary bus or detects **p_perr_l** asserted during the completion phase of an upstream delayed write transaction on the target (primary) bus.

**Table 31. Assertion of s_perr_l**

| s_perr_l | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 1 (deasserted) | Read | Downstream | Primary | x/x |
| 0 (asserted) | Read | Downstream | Secondary | x/1 |
| 1 | Read | Upstream | Primary | x/x |
| 1 | Read | Upstream | Secondary | x/x |
| 1 | Posted write | Downstream | Primary | x/x |
| 1 | Posted write | Downstream | Secondary | x/x |
| 1 | Posted write | Upstream | Primary | x/x |
| 0 | Posted write | Upstream | Secondary | x/1 |
| 1 | Delayed write | Downstream | Primary | x/x |
| 1 | Delayed write | Downstream | Secondary | x/x |
| 0[2] | Delayed write | Upstream | Primary | 1/1 |
| 0 | Delayed write | Upstream | Secondary | x/1 |

1. x — don't care.
2. The parity error was detected on the target (primary) bus but not on the initiator (secondary) bus.

Table 32 shows assertion of p_serr_l. This signal is set under the following conditions:

- The 21154 has detected p_perr_l asserted on an upstream posted write transaction or s_perr_l asserted on a downstream posted write transaction.
- The 21154 did not detect the parity error as a target of the posted write transaction.
- The parity error response bit on the command register and the parity error response bit on the bridge control register must both be set.
- The SERR# enable bit must be set in the command register.

**Table 32. Assertion of p_serr_l for Data Parity Errors**

| p_serr_l | Transaction Type | Direction | Bus Where Error Was Detected | Primary/Secondary Parity Error Response Bits[1] |
|---|---|---|---|---|
| 1 (deasserted) | Read | Downstream | Primary | x/x |
| 1 | Read | Downstream | Secondary | x/x |
| 1 | Read | Upstream | Primary | x/x |
| 1 | Read | Upstream | Secondary | x/x |
| 1 | Posted write | Downstream | Primary | x/x |
| 0[2] (asserted) | Posted write | Downstream | Secondary | 1/1 |
| 0[3] | Posted write | Upstream | Primary | 1/1 |
| 1 | Posted write | Upstream | Secondary | x/x |
| 1 | Delayed write | Downstream | Primary | x/x |
| 1 | Delayed write | Downstream | Secondary | x/x |
| 1 | Delayed write | Upstream | Primary | x/x |
| 1 | Delayed write | Upstream | Secondary | x/x |

1. x — don't care.
2. The parity error was detected on the target (secondary) bus but not on the initiator (primary) bus.
3. The parity error was detected on the target (primary) bus but not on the initiator (secondary) bus.

## 7.4 System Error (SERR#) Reporting

The 21154 uses the p_serr_l signal to report conditionally a number of system error conditions in addition to the special case parity error conditions described in Section 7.2.3.

Whenever the assertion of p_serr_l is discussed in this document, it is assumed that the following conditions apply:

- For the 21154 to assert p_serr_l for any reason, the SERR# enable bit must be set in the command register.
- Whenever the 21154 asserts p_serr_l, the 21154 must also set the signaled system error bit in the status register.

In compliance with the *PCI-to-PCI Bridge Architecture Specification*, the 21154 asserts p_serr_l when it detects the secondary SERR# input, s_serr_l, asserted and the SERR# forward enable bit is set in the bridge control register. In addition, the 21154 also sets the received system error bit in the secondary status register.

The 21154 also conditionally asserts p_serr_l for any of the following reasons:

- Target abort detected during posted write transaction
- Master abort detected during posted write transaction
- Posted write data discarded after $2^{24}$ attempts to deliver ($2^{24}$ target retries received)
- Parity error reported on target bus during posted write transaction (see previous section)
- Delayed write data discarded after $2^{24}$ attempts to deliver ($2^{24}$ target retries received)

- Delayed read data cannot be transferred from target after $2^{24}$ attempts ($2^{24}$ target retries received)
- Master timeout on delayed transaction

The device-specific p_serr_l status register reports the reason for the 21154's assertion of p_serr_l.

Most of these events have additional device-specific disable bits in the p_serr_l event disable register that make it possible to mask out p_serr_l assertion for specific events. The master timeout condition has a SERR# enable bit for that event in the bridge control register and therefore does not have a device-specific disable bit.

# 8.0 Exclusive Access

This chapter describes the use of the LOCK# signal to implement exclusive access to a target for transactions that cross the 21154.

## 8.1 Concurrent Locks

The primary and secondary bus lock mechanisms operate concurrently except when a locked transaction crosses the 21154. A primary master can lock a primary target without affecting the status of the lock on the secondary bus, and vice versa. This means that a primary master can lock a primary target at the same time that a secondary master locks a secondary target.

## 8.2 Acquiring Exclusive Access Across the 21154

For any PCI bus, before acquiring access to the LOCK# signal and starting a series of locked transactions, the initiator must first check that both of the following conditions are met:

- The PCI bus must be idle.
- The LOCK# signal must be deasserted.

The initiator leaves the LOCK# signal deasserted during the address phase (only the first address phase of a dual address transaction) and asserts LOCK# one clock cycle later. Once a data transfer is completed from the target, the target lock has been achieved.

Locked transactions can cross the 21154 only in the downstream direction, from the primary bus to the secondary bus.

When the target resides on another PCI bus, the master must acquire not only the lock on its own PCI bus but also the lock on every bus between its bus and the target's bus. When the 21154 detects, on the primary bus, an initial locked transaction intended for a target on the secondary bus, the 21154 samples the address, transaction type, byte enable bits, and parity, as described in Section 4.6.4. It also samples the lock signal. Because a target retry is signaled to the initiator, the initiator must relinquish the lock on the primary bus, and therefore the lock is not yet established.

The first locked transaction must be a read transaction. Subsequent locked transactions can be read or write transactions. Posted memory write transactions that are a part of the locked transaction sequence are still posted. Memory read transactions that are a part of the locked transaction sequence are not prefetched.

When the locked delayed read request is queued, the 21154 does not queue any more transactions until the locked sequence is finished. The 21154 signals a target retry to all transactions initiated subsequent to the locked read transaction that are intended for targets on the other side of the 21154. The 21154 allows any transactions queued before the locked transaction to complete before initiating the locked transaction.

When the locked delayed read request transaction moves to the head of the delayed transaction queue, the 21154 initiates the transaction as a locked read transaction by deasserting s_lock_l on the secondary bus during the first address phase, and by asserting s_lock_l one cycle later. If s_lock_l is already asserted (used by another initiator), the 21154 waits to request access to the secondary bus until s_lock_l is sampled deasserted when the secondary bus is idle. Note that the

existing lock on the secondary bus could not have crossed the 21154; otherwise, the pending queued locked transaction would not have been queued. When the 21154 is able to complete a data transfer with the locked read transaction, the lock is established on the secondary bus.

When the initiator repeats the locked read transaction on the primary bus with the same address, transaction type, and byte enable bits, the 21154 transfers the read data back to the initiator, and the lock is then also established on the primary bus.

For the 21154 to recognize and respond to the initiator, the initiator's subsequent attempts of the read transaction must use the locked transaction sequence (deassert p_lock_l during address phase, and assert p_lock_l one cycle later). If the LOCK# sequence is not used in subsequent attempts, a master timeout condition may result. When a master timeout condition occurs, p_serr_l is conditionally asserted (see Section 7.4), the read data and queued read transaction are discarded, and the s_lock_l signal is deasserted on the secondary bus.

Once the intended target has been locked, any subsequent locked transactions initiated on the primary bus that are forwarded by the 21154 are driven as locked transactions on the secondary bus.

When the 21154 receives a target abort or a master abort in response to the delayed locked read transaction, a target abort is returned to the initiator, and no locks are established on either the target or the initiator bus. The 21154 resumes forwarding unlocked transactions in both directions.

When the 21154 detects, on the secondary bus, a locked delayed transaction request intended for a target on the primary bus, the 21154 queues and forwards the transaction as an unlocked transaction. The 21154 ignores s_lock_l for upstream transactions and initiates all upstream transactions as unlocked transactions.

## 8.3 Ending Exclusive Access

After the lock has been acquired on both the primary and secondary buses, the 21154 must maintain the lock on the secondary (target) bus for any subsequent locked transactions until the initiator relinquishes the lock.

The only time a target retry causes the lock to be relinquished is on the first transaction of a locked sequence. On subsequent transactions in the sequence, the target retry has no effect on the status of the lock signal.

An established target lock is maintained until the initiator relinquishes the lock. The 21154 does not know whether the current transaction is the last one in a sequence of locked transactions until the initiator deasserts the p_lock_l signal at the end of the transaction.

When the last locked transaction is a delayed transaction, the 21154 has already completed the transaction on the secondary bus. In this case, as soon as the 21154 detects that the initiator has relinquished the p_lock_l signal by sampling it in the deasserted state while p_frame_l is deasserted, the 21154 deasserts the s_lock_l signal on the secondary bus as soon as possible. Because of this behavior, s_lock_l may not be deasserted until several cycles after the last locked transaction has been completed on the secondary bus. As soon as the 21154 has deasserted s_lock_l to indicate the end of a sequence of locked transactions, it resumes forwarding unlocked transactions.

When the last locked transaction is a posted write transaction, the 21154 deasserts s_lock_l on the secondary bus at the end of the transaction because the lock was relinquished at the end of the write transaction on the primary bus.

When the 21154 receives a target abort or a master abort in response to a locked delayed transaction, the 21154 returns a target abort when the initiator repeats the locked transaction. The initiator must then deassert p_lock_l at the end of the transaction. The 21154 sets the appropriate status bits, flagging the abnormal target termination condition (see Section 4.10). Normal forwarding of unlocked posted and delayed transactions is resumed.

When the 21154 receives a target abort or a master abort in response to a locked posted write transaction, the 21154 cannot pass back that status to the initiator. The 21154 asserts p_serr_l when a target abort or a master abort is received during a locked posted write transaction, if the SERR# enable bit is set in the command register. Signal p_serr_l is asserted for the master abort condition if the master abort mode bit is set in the bridge control register (see Section 7.4).

# 9.0 PCI Bus Arbitration

The 21154 must arbitrate for use of the primary bus when forwarding upstream transactions, and for use of the secondary bus when forwarding downstream transactions. The arbiter for the primary bus resides external to the 21154, typically on the motherboard. For the secondary PCI bus, the 21154 implements an internal arbiter. This arbiter can be disabled, and an external arbiter can be used instead.

This chapter describes primary and secondary bus arbitration.

## 9.1 Primary PCI Bus Arbitration

The 21154 implements a request output pin, p_req_l, and a grant input pin, p_gnt_l, for primary PCI bus arbitration. The 21154 asserts p_req_l when forwarding transactions upstream; that is, it acts as initiator on the primary PCI bus. As long as at least one pending transaction resides in the queues in the upstream direction, either posted write data or delayed transaction requests, the 21154 keeps p_req_l asserted. However, if a target retry, target disconnect, or a target abort is received in response to a transaction initiated by the 21154 on the primary PCI bus, the 21154 deasserts p_req_l for two PCI clock cycles.

When p_gnt_l is asserted low by the primary bus arbiter after the 21154 has asserted p_req_l, the 21154 initiates a transaction on the primary bus during the next PCI clock cycle. When p_gnt_l is asserted to the 21154 when p_req_l is not asserted, the 21154 parks p_ad, p_cbe_l, and p_par by driving them to valid logic levels. When the primary bus is parked at the 21154 and the 21154 then has a transaction to initiate on the primary bus, the 21154 starts the transaction if p_gnt_l was asserted during the previous cycle.

## 9.2 Secondary PCI Bus Arbitration

The 21154 implements an internal secondary PCI bus arbiter. This arbiter supports nine external masters in addition to the 21154. The internal arbiter can be disabled, and an external arbiter can be used instead for secondary bus arbitration.

### 9.2.1 Secondary Bus Arbitration Using the Internal Arbiter

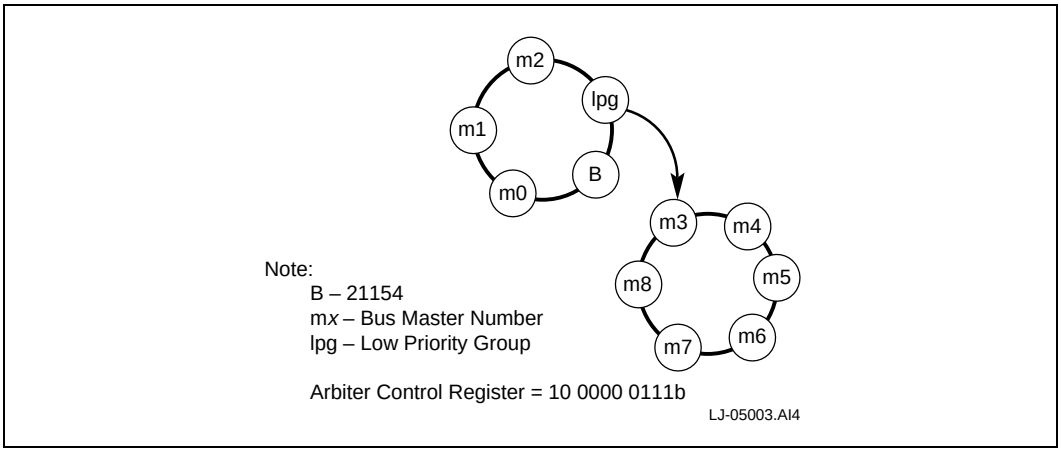
To use the internal arbiter, the secondary bus arbiter enable pin, s_cfn_l, must be tied low. The 21154 has nine secondary bus request input pins, s_req_l<8:0>, and nine secondary bus output grant pins, s_gnt_l<8:0>, to support external secondary bus masters. The 21154 secondary bus request and grant signals are connected internally to the arbiter and are not brought out to external pins when s_cfn_l is low.

The secondary arbiter supports a programmable 2-level rotating algorithm. Two groups of masters are assigned, a high priority group and a low priority group. The low priority group as a whole represents one entry in the high priority group; that is, if the high priority group consists of *n* masters, then in at least every *n* +1 transactions the highest priority is assigned to the low priority group. Priority rotates evenly among the low priority group. Therefore, members of the high priority group can be serviced *n* transactions out of *n* +1, while one member of the low priority group is serviced once every *n* +1 transactions. Figure 18 shows an example of an internal arbiter where four masters, including the 21154, are in the high priority group, and six masters are in the

low priority group. Using this example, if all requests are always asserted, the highest priority rotates among the masters in the following fashion (high priority members are given in italics, low priority members, in boldface type):

*B*, *m*0, *m*1, *m*2, **m3**, *B*, *m*0, *m*1, *m*2, **m4**, *B*, *m*0, *m*1, *m*2, **m5**, *B*, *m*0, *m*1, *m*2, **m6**, *B*, *m*0, *m*1, and so on.

**Figure 18. Secondary Arbiter Example**




Each bus master, including the 21154, can be configured to be in either the low priority group or the high priority group by setting the corresponding priority bit in the arbiter control register in device-specific configuration space. Each master has a corresponding bit. If the bit is set to 1, the master is assigned to the high priority group. If the bit is set to 0, the master is assigned to the low priority group. If all the masters are assigned to one group, the algorithm defaults to a straight rotating priority among all the masters. After reset, all external masters are assigned to the low priority group, and the 21154 is assigned to the high priority group. The 21154 receives highest priority on the target bus every other transaction, and priority rotates evenly among the other masters.

Priorities are reevaluated every time s_frame_l is asserted, that is, at the start of each new transaction on the secondary PCI bus. From this point until the time that the next transaction starts, the arbiter asserts the grant signal corresponding to the highest priority request that is asserted. If a grant for a particular request is asserted, and a higher priority request subsequently asserts, the arbiter deasserts the asserted grant signal and asserts the grant corresponding to the new higher priority request on the next PCI clock cycle. When priorities are reevaluated, the highest priority is assigned to the next highest priority master relative to the master that initiated the previous transaction. The master that initiated the last transaction now has the lowest priority in its group.

If the 21154 detects that an initiator has failed to assert s_frame_l after 16 cycles of both grant assertion and a secondary idle bus condition, the arbiter deasserts the grant. That master does not receive any more grants until it deasserts its request for at least one PCI clock cycle.

To prevent bus contention, if the secondary PCI bus is idle, the arbiter never asserts one grant signal in the same PCI cycle in which it deasserts another. It deasserts one grant, and then asserts the next grant, no earlier than one PCI clock cycle later. If the secondary PCI bus is busy, that is, either s_frame_l or s_irdy_l is asserted, the arbiter can deassert one grant and assert another grant during the same PCI clock cycle.

### 9.2.2 Secondary Bus Arbitration Using an External Arbiter

The internal arbiter is disabled when the secondary bus central function control pin, s_cfn_l, is pulled high. An external arbiter must then be used.

When s_cfn_l is tied high, the 21154 reconfigures two pins to be external request and grant pins. The s_gnt_l<0> pin is reconfigured to be the 21154's external request pin because it is an output. The s_req_l<0> pin is reconfigured to be the external grant pin because it is an input. When an external arbiter is used, the 21154 uses the s_gnt_l<0> pin to request the secondary bus. When the reconfigured s_req_l<0> pin is asserted low after the 21154 has asserted s_gnt_l<0>, the 21154 initiates a transaction on the secondary bus one cycle later. If s_req_l<0> is asserted and the 21154 has not asserted s_gnt_l<0>, the 21154 parks the s_ad<31:0>, s_cbe_l<3:0>, and s_par pins by driving them to valid logic levels.

The unused secondary bus grant outputs, s_gnt_l<8:1>, are driven high. Unused secondary bus request inputs, s_req_l<8:1>, should be pulled high.

### 9.2.3 Bus Parking

Bus parking refers to driving the AD, C/BE#, and PAR lines to a known value while the bus is idle. In general, the device implementing the bus arbiter is responsible for parking the bus or assigning another device to park the bus. A device parks the bus when the bus is idle, its bus grant is asserted, and the device's request is not asserted. The AD and C/BE# signals should be driven first, with the PAR signal driven one cycle later. The 64-bit extension signals are not parked, because those signals are connected to external pull-up resistors.

The 21154 parks the primary bus only when p_gnt_l is asserted, p_req_l is deasserted, and the primary PCI bus is idle. When p_gnt_l is deasserted, the 21154 tristates the p_ad, p_cbe_l, and p_par signals on the next PCI clock cycle. If the 21154 is parking the primary PCI bus and wants to initiate a transaction on that bus, then the 21154 can start the transaction on the next PCI clock cycle by asserting p_frame_l if p_gnt_l is still asserted.

If the internal secondary bus arbiter is enabled, the secondary bus is always parked at the last master that used the PCI bus. That is, the 21154 keeps the secondary bus grant asserted to a particular master until a new secondary bus request comes along. After reset, the 21154 parks the secondary bus at itself until transactions start occurring on the secondary bus. If the internal arbiter is disabled, the 21154 parks the secondary bus only when the reconfigured grant signal, s_req_l<0>, is asserted and the secondary bus is idle.

# 10.0 General-Purpose I/O Interface

The 21154 implements a 4-pin general-purpose I/O gpio interface. During normal operation, the gpio interface is controlled by device-specific configuration registers. In addition, the gpio interface can be used for the following functions:

- During secondary interface reset, the gpio interface can be used to shift in a 16-bit serial stream that serves as a secondary bus clock disable mask.
- A live insertion bit can be used, along with the gpio<3> pin, to bring the 21154 gracefully to a halt through hardware, permitting live insertion of option cards behind the 21154.

## 10.1 gpio Control Registers

During normal operation, the gpio interface is controlled by the following device-specific configuration registers:

- The gpio output data register
- The gpio output enable control register
- The gpio input data register

These registers consist of five 8-bit fields:

- Write-1-to-set output data field
- Write-1-to-clear output data field
- Write-1-to-set signal output enable control field
- Write-1-to-clear signal output enable control field
- Input data field

The bottom 4 bits of the output enable fields control whether each gpio signal is input only or bidirectional. Each signal is controlled independently by a bit in each output enable control field. If a 1 is written to the write-1-to-set field, the corresponding pin is activated as an output. If a 1 is written to the write-1-to-clear field, the output driver is tristated, and the pin is then input only. Writing zeros to these registers has no effect. The reset state for these signals is input only.

The input data field is read only and reflects the current value of the gpio pins. A type 0 configuration read operation to this address is used to obtain the values of these pins. All pins can be read at any time, whether configured as input only or as bidirectional.

The output data fields also use the write-1-to-set and write-1-to-clear method. If a 1 is written to the write-1-to-set field and the pin is enabled as an output, the corresponding gpio output is driven high. If a 1 is written to the write-1-to-clear field and the pin is enabled as an output, the corresponding gpio output is driven low. Writing zeros to these registers has no effect. The value written to the output register will be driven only when the gpio signal is configured as bidirectional. A type 0 configuration write operation is used to program these fields. The reset value for the output is 0.

## 10.2 Secondary Clock Control

The 21154 uses the gpio pins and the msk_in signal to input a 16-bit serial data stream. This data stream is shifted into the secondary clock control register and is used for selectively disabling secondary clock outputs.

The serial data stream is shifted in as soon as p_rst_l is detected deasserted and the secondary reset signal, s_rst_l, is detected asserted. The deassertion of s_rst_l is delayed until the 21154 completes shifting in the clock mask data, which takes 23 clock cycles (46 cycles if operating at 66 MHz). After that, the gpio pins can be used as general purpose I/O pins.

An external shift register should be used to load and shift the data. The gpio pins are used for shift register control and serial data input. Table 33 shows the operation of the gpio pins.

**Table 33. gpio Operation**

| gpio Pin | Operation |
|---|---|
| gpio<0> | Shift register clock output at 33 MHz maximum frequency |
| gpio<1> | Not used |
| gpio<2> | Shift Register control:<br>0—Load<br>1—Shift |
| gpio<3> | Not used |

The data is input through the dedicated input signal, msk_in.

The shift register circuitry is not necessary for correct operation of the 21154. The shift registers can be eliminated, and msk_in can be tied low to enable all secondary clock outputs or tied high to force all secondary clock outputs high.

Table 34 shows the format of the serial stream.

**Table 34. gpio Serial Data Format**

| Bit | Description | s_clk_o Output |
|---|---|---|
| <1:0> | Slot 0 PRSNT#<1:0> or device 0 | 0 |
| <3:2> | Slot 1 PRSNT#<1:0> or device 1 | 1 |
| <5:4> | Slot 2 PRSNT#<1:0> or device 2 | 2 |
| <7:6> | Slot 3 PRSNT#<1:0> or device 3 | 3 |
| <8> | Device 4 | 4 |
| <9> | Device 5 | 5 |
| <10> | Device 6 | 6 |
| <11> | Device 7 | 7 |
| <12> | Device 8 | 8 |
| <13> | 21154 **s_clk** input | 9 |
| <14> | Reserved | Not applicable |
| <15> | Reserved | Not applicable |

The first eight bits contain the PRSNT#<1:0> signal values for four slots, and these bits control the s_clk_o<3:0> outputs. If one or both of the PRSNT#<1:0> signals are 0, that indicates that a card is present in the slot and therefore the secondary clock for that slot is not masked. If these clocks are connected to devices and not to slots, one or both of the bits should be tied low to enable the clock.

The next five bits are the clock mask for devices; each bit enables or disables the clock for one device. These bits control the s_clk_o<8:4> outputs: 0 enables the clock, and 1 disables the clock.

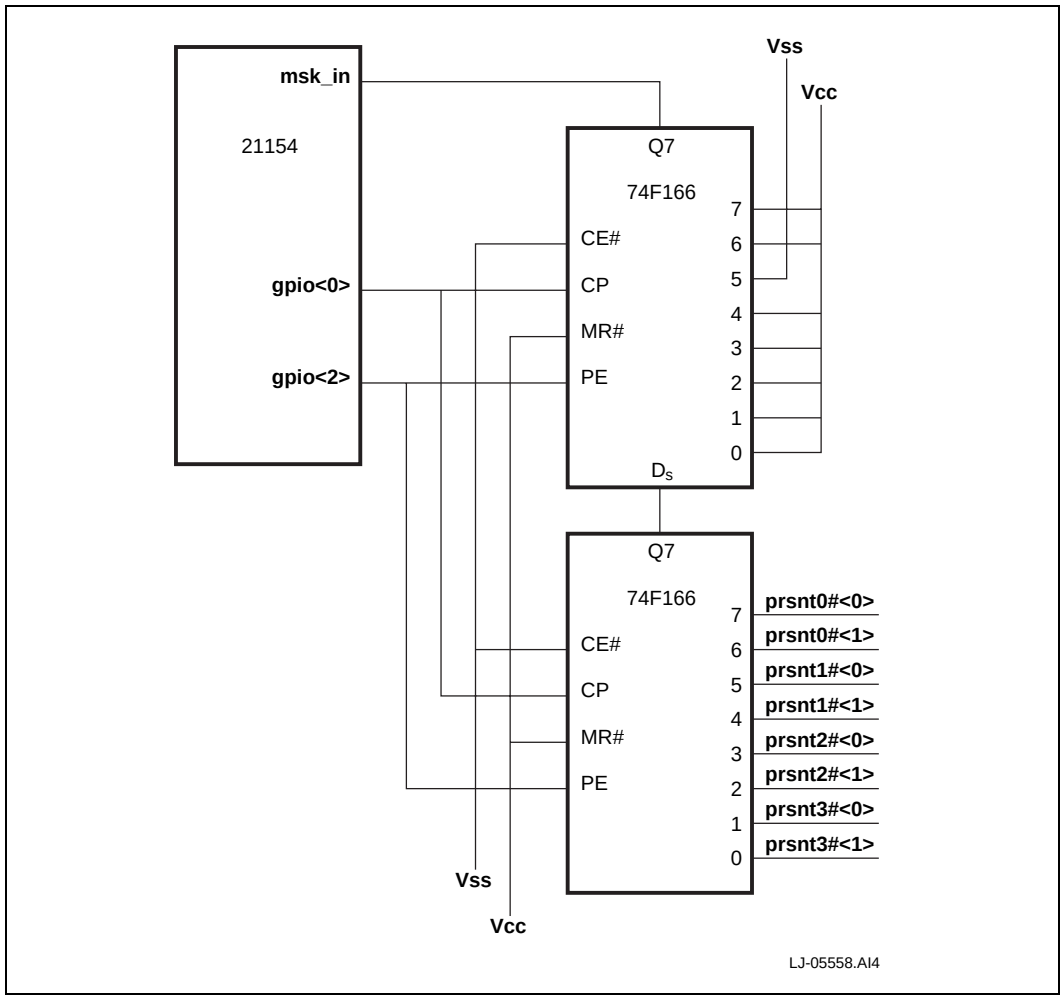
Bit 13 is the clock enable bit for s_clk_o<9>, which is connected to the 21154's s_clk input.

If desired, the assignment of s_clk_o clock outputs to slots, devices, and the 21154's s_clk input can be rearranged from the assignment shown here. However, it is important that the serial data stream format match the assignment of s_clk_o outputs.

The gpio pin serial protocol is designed to work with two 74F166 8-bit shift registers.

Figure 19 shows how the serial mask circuitry may be implemented for a motherboard with four slots.

**Figure 19. Example of gpio Clock Mask Implementation on the System Board**

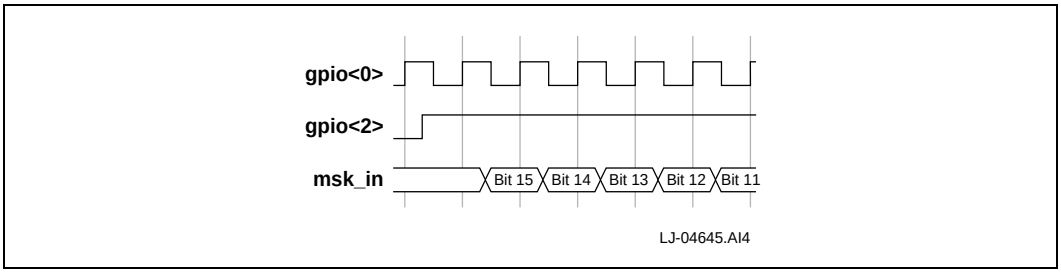


The eight least significant bits are connected to the PRSNT# pins for the slots. The next five bits are tied high to disable their respective secondary clocks because those clocks are not connected to anything. The next bit is tied low because that secondary clock output is connected to the 21154 s_clk input.

When the secondary reset signal, s_rst_l, is detected asserted and the primary reset signal, p_rst_l, is detected deasserted, the 21154 drives gpio<2> low for one cycle to load the clock mask inputs into the shift register. On the next cycle, the 21154 drives gpio<2> high to perform a shift operation. This shifts the clock mask into msk_in; the most significant bit is shifted in first, and the least significant bit is shifted in last.

Figure 20 shows a timing diagram for the load and for the beginning of the shift operation.

**Figure 20. Clock Mask and Load Shift Timing**




After the shift operation is complete, the 21154 tristates the gpio signals and can deassert s_rst_l if the secondary reset bit is clear. The 21154 then ignores msk_in. Control of the gpio signal now reverts to the 21154 gpio control registers. The clock disable mask can be modified subsequently through a configuration write command to the secondary clock control register in device-specific configuration space.

## 10.3 Live Insertion

The gpio<3> pin can be used, along with a live insertion mode bit, to disable transaction forwarding.

To enable live insertion mode, the live insertion mode bit in the chip control register must be set to 1, and the output enable control for gpio<3> must be set to input only in the gpio output enable control register. When live insertion mode is enabled, whenever gpio<3> is driven to a value of 1, the I/O enable, the memory enable, and the master enable bits are internally masked to 0. This means that, as a target, the 21154 no longer accepts any I/O or memory transactions, on either interface. When read, the register bits still reflect the value originally written by a configuration write command; when gpio<3> is deasserted, the internal enable bits return to their original value (as they appear when read from the command register). When this mode is enabled, as a master, the 21154 completes any posted write or delayed request transactions that have already been queued.

Delayed completion transactions are not returned to the master in this mode because the 21154 is not responding to any I/O or memory transactions during this time.

Note that the 21154 continues to accept configuration transactions in live insertion mode.

Once live insertion mode brings the 21154 to a halt and queued transactions are completed, the secondary reset bit in the bridge control register can be used to assert s_rst_l, if desired, to reset and tristate secondary bus devices, and to enable any live insertion hardware.

# 11.0 Clocks

This chapter provides information about the 21154 clocks.

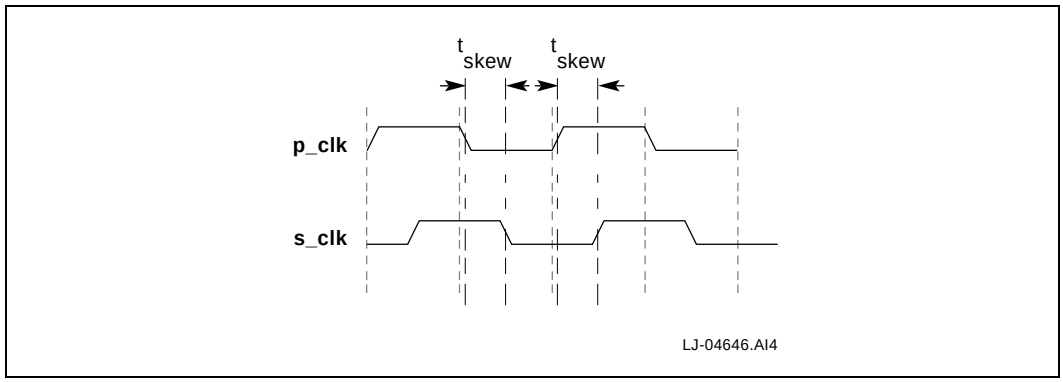
## 11.1 Primary and Secondary Clock Inputs

The 21154 implements a separate clock input for each PCI interface. The primary interface is synchronized to the primary clock input, p_clk, and the secondary interface is synchronized to the secondary clock input, s_clk.

The 21154 operates at a maximum frequency of 33 MHz, or 66 MHz if the 21154 is 66 MHz capable. s_clk operates either at the same frequency or at half the frequency as p_clk.

The primary and secondary clock inputs must always maintain a synchronous relationship to each other; that is, their edge relationships to each other are well defined. The maximum skew between s_clk rising edges is 7 ns, as is the maximum skew between p_clk and s_clk falling edges. The minimum skew between p_clk and s_clk edges is 0 ns. The secondary clock edge must never precede the primary clock edge. Figure 21 illustrates the timing relationship between the primary and the secondary clock inputs.

**Figure 21. p_clk and s_clk Relative Timing**




## 11.2 Secondary Clock Outputs

The 21154 has five secondary clock outputs, s_clk_o<9:0>, that can be used as clock inputs for up to nine external secondary bus devices and for the 21154 secondary clock input.

The s_clk_o outputs are derived from p_clk. The s_clk_o edges are delayed from p_clk edges by a minimum of 0 ns and a maximum of 5 ns. The maximum skew between s_clk_o edges is 500 ps. Therefore, to meet the p_clk and s_clk requirements stated in Section 11.1, no more than 2 ns of delay is allowed for secondary clock etch returning to the device secondary clock inputs.

The rules for using secondary clocks are:

- Each secondary clock output is limited to one load.
- One of the secondary clock outputs must be used for the 21154 s_clk input.
- Intel recommends using an equivalent amount of etch on the board for all secondary clocks, to minimize skew between them, and a maximum delay of the etch of 2 ns.
- Intel recommends terminating or disabling unused secondary clock outputs to reduce power dissipation and noise in the system.

### 11.2.1 Disabling Unused Secondary Clock Outputs

When secondary clock outputs are not used, both gpio<3:0> and msk_in can be used to clock in a serial mask that selectively tristates secondary clock outputs. Section 10.2 describes how the 21154 uses the gpio pins and the msk_in signal to input this data stream.

After the serial mask has been shifted into the 21154, the value of the mask is readable and modifiable in the secondary clock disable mask register. When the mask is modified by a configuration write operation to this register, the new clock mask disables the appropriate secondary clock outputs within a few cycles. This feature allows software to disable or enable secondary clock outputs based on the presence of option cards, and so on.

The 21154 delays deasserting the secondary reset signal, s_rst_l, until the serial clock mask has been completely shifted in and the secondary clocks have been disabled or enabled, according to the mask. The delay between p_rst_l deassertion and s_rst_l deassertion is approximately 23 cycles (46 cycles if s_clk is operating at 66 MHz).

# 12.0 66-Mhz Operation

Some versions of the 21154 support 66 MHz operation.

Signal config66 must be tied high on the board to enable 66 MHz operation and to set the 66 MHz Capable bit in the Status register and Secondary Status register in configuration space. If the 21154 version is not 66MHz capable, then config66 should be tied low. Signals p_m66ena and s_m66ena should never be pulled high unless config66 is also high.

Signals p_m66ena and s_m66ena indicate whether the primary and secondary interfaces, respectively, are operating at 66 MHz[1]. This information is needed to control the frequency of the secondary bus. Note that the *PCI Local Bus Specification, Revision 2.1* restricts clock frequency changes above 33 MHz to during PCI reset only.

The 66Mhz capable 21154 supports the following primary and secondary bus frequency combinations:

- 66 MHz primary bus, 66 MHz secondary bus
- 66 MHz primary bus, 33 MHz secondary bus
- 33 MHz primary bus, 33 MHz secondary bus

The 21154 does not support 33 MHz primary/66 MHz secondary bus operation, where the secondary bus is operating at twice the frequency of the primary bus. If config66 is high and p_m66ena is low (66 MHz capable, primary bus at 33MHz), then the 21154 pulls down s_m66ena to indicate that the secondary bus is operating at 33 MHz.

The 21154 generates the clock signals (s_clk_o<9:0>) for the secondary bus devices and its own secondary interface. The 21154 divides the primary bus clock p_clk by two to generate the secondary bus clock outputs whenever the primary bus is operating at 66 MHz and the secondary bus is operating at 33 MHz. The bridge detects this condition when p_m66ena is high and s_m66ena is low.

[1] In general, 66-MHz operation means operation ranging from 33 MHz up to 66 MHz.

# 13.0 PCI Power Management

The 21154[1] incorporates functionality that meets the requirements of the *PCI Power Management Specification, Revision 1.0*. These features include:

- PCI power management registers using the enhanced capabilities port (ECP) address mechanism
- Support for D0, $D3_{hot}$, and $D3_{cold}$ power management states
- Support for D0, D1, D2, $D3_{hot}$, and $D3_{cold}$ power management states for devices behind the bridge
- Support of the B2 secondary bus power state when in the $D3_{hot}$ power management state

Table 35 shows the states and related actions that the 21154 performs during power management transitions. (No other transactions are permitted.)

**Table 35. Power Management Transitions**

| Current State | Next State | Action |
|---|---|---|
| D0 | $D3_{cold}$ | Power has been removed from the 21154. A power-up reset must be performed to bring the 21154 to D0. |
| D0 | $D3_{hot}$ | If enabled to do so by the **bpcce** pin, the 21154 will disable the secondary clocks and drive them low. |
| D0 | D2 | Unimplemented power state. The 21154 will ignore the write to the power state bits (power state remains at D0). |
| D0 | D1 | Unimplemented power state. The 21154 will ignore the write to the power state bits (power state remains at D0). |
| $D3_{hot}$ | D0 | The 21154 enables secondary clock outputs and performs an internal chip reset. Signal **s_rst_l** will not be asserted. All registers will be returned to the reset values and buffers will be cleared. |
| $D3_{hot}$ | D3cold | Power has been removed from the 21154. A power-up reset must be performed to bring the 21154 to D0. |
| $D3_{cold}$ | D0 | Power-up reset. The 21154 performs the standard power-up reset functions as described in Section 11.0. |

PME# signals are routed from downstream devices around PCI-to-PCI bridges. PME# signals do not pass through PCI-to-PCI bridges.

1. The 21154-AA does not include these features.

# 14.0 Reset

This chapter describes the primary interface, secondary interface, and chip reset mechanisms.

## 14.1 Primary Interface Reset

The 21154 has one reset input, p_rst_l. When p_rst_l is asserted, the following events occur:

- The 21154 immediately tristates all primary and secondary PCI interface signals.
- The 21154 performs a chip reset.
- Registers that have default values are reset. Section 15.3 lists the values of all configuration space registers after reset.
- The 21154 samples p_req64_l to determine whether the 64-bit extension is enabled on the primary bus (see Section 4.8.5).

The p_rst_l asserting and deasserting edges can be asynchronous to p_clk and s_clk.

## 14.2 Secondary Interface Reset

The 21154 is responsible for driving the secondary bus reset signal, s_rst_l. The 21154 asserts s_rst_l when any of the following conditions is met:

- Signal p_rst_l is asserted.

  Signal s_rst_l remains asserted as long as p_rst_l is asserted and does not deassert until p_rst_l is deasserted and the secondary clock serial disable mask has been shifted in (23 or 46 clock cycles after p_rst_l deassertion).
- The secondary reset bit in the bridge control register is set.

  Signal s_rst_l remains asserted until a configuration write operation clears the secondary reset bit and the secondary clock serial mask has been shifted in.
- The chip reset bit in the diagnostic control register is set.

  Signal s_rst_l remains asserted until a configuration write operation clears the secondary reset bit and the secondary clock serial mask has been shifted in.

When s_rst_l is asserted, all secondary PCI interface control signals, including the secondary grant outputs, are immediately tristated. Signals s_ad<31:0>, s_cbe_l<3:0>, and s_par are driven low for the duration of s_rst_l assertion. Signal s_req64_l is asserted low indicating 64-bit extension support on the secondary interface. All posted write and delayed transaction data buffers are reset; therefore, any transactions residing in 21154 buffers at the time of secondary reset are discarded.

When s_rst_l is asserted by means of the secondary reset bit, the 21154 remains accessible during secondary interface reset and continues to respond to accesses to its configuration space from the primary interface.

## 14.3 Chip Reset

The chip reset bit in the diagnostic control register can be used to reset the 21154 and the secondary bus.

When the chip reset bit is set, all registers and chip state are reset and all signals are tristated. In addition, s_rst_l is asserted, and the secondary reset bit is automatically set. Signal s_rst_l remains asserted until a configuration write operation clears the secondary reset bit and the serial clock mask has been shifted in.

As soon as chip reset completes, within 20 PCI clock cycles after completion of the configuration write operation that sets the chip reset bit, the chip reset bit automatically clears and the chip is ready for configuration.

During chip reset, the 21154 is inaccessible.

# 15.0 Configuration Space Registers

This chapter provides a detailed description of the 21154 configuration space registers. The chapter is divided into three sections: Section 15.1 describes the standard 21154 PCI-to-PCI bridge configuration registers, Section 15.2 describes the 21154 device-specific configuration registers, and Section 15.3 describes the configuration register values after reset.

The 21154 configuration space uses the PCI-to-PCI bridge standard format specified in the *PCI-to-PCI Bridge Architecture Specification*. The header type at configuration address 0Eh reads as 01h, indicating that this device uses the PCI-to-PCI bridge format.

The 21154 also contains device-specific registers, starting at address 40h. Use of these registers is not required for standard PCI-to-PCI bridge implementations.

The configuration space registers can be accessed only from the primary PCI bus. To access a register, perform a Type 0 format configuration read or write operation to that register. During the Type 0 address phase, p_ad<7:2> indicates the Dword offset of the register. During the data phase, p_cbe_l<3:0> selects the bytes in the Dword that is being accessed.

***Caution:*** Software changes the configuration register values that affect 21154 behavior only during initialization. Change these values subsequently only when both the primary and secondary PCI buses are idle, and the data buffers are empty; otherwise, the behavior of the 21154 is unpredictable.

Figure 22 shows a summary of configuration space:

**Figure 22. 21154 Configuration Space Map**

| 31 — 16 | 15 — 00 | |
|---|---|---|
| Device ID | Vendor ID | 00h |
| Primary Status | Primary Command | 04h |
| Class Code | Revision ID | 08h |
| Reserved / Header Type | Primary Latency Timer / Cache Line Size | 0Ch |
| Reserved | | 10h |
| Reserved | | 14h |
| Secondary Latency Timer / Subordinate Bus Number | Secondary Bus Number / Primary Bus Number | 18h |
| Secondary Status | I/O Limit Address / I/O Base Address | 1Ch |
| Memory Limit Address | Memory Base Address | 20h |
| Prefetchable Memory Limit Address | Prefetchable Memory Base Address | 24h |
| Prefetchable Memory Base Address Upper 32 Bits | | 28h |
| Prefetchable Memory Limit Address Upper 32 Bits | | 2Ch |
| I/O Limit Address Upper 16 Bits | I/O Base Address Upper 16 Bits | 30h |
| Reserved* | ECP Pointer* | 34h |
| Reserved | | 38h |
| Bridge Control | Interrupt Pin / Reserved | 3Ch |
| Arbiter Control | Diagnostic Control / Chip Control | 40h |
| Reserved | | 44h-60h |
| **gpio** Input Data / **gpio** Output Enable Control | **gpio** Output Data / **p_serr_l** Event Disable | 64h |
| Reserved / **p_serr_l** Status | Secondary Clock Control | 68h |
| Reserved | | 6Ch-DBh |
| Power Management Capabilities** | Next Item Ptr** / Capability ID** | DCh |
| Data** / PPB Support Extensions** | Power Management CSR** | E0h |
| Reserved | | E4h-FFh |

* For the 21154-AA, these registers are R/W Subsystem ID and Subsystem Vendor ID.

** These are reserved for the 21154-AA.

## 15.1 PCI-to-PCI Bridge Standard Configuration Registers

This section provides a detailed description of the PCI-to-PCI bridge standard configuration registers.

Each field has a separate description.

Fields that have the same configuration Dword address are selectable by turning on (driving low) the appropriate byte enable bits on p_cbe_l during the data phase. To select all fields of a configuration address, drive all byte enable bits low.

All reserved fields and registers are read only and always return 0.

### 15.1.1 Vendor ID Register—Offset 00h

This section describes the vendor ID register.

Dword address = 00h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:0 | Vendor ID | R | Identifies Intel as the vendor of this device. Internally hardwired to be 1011h. |

### 15.1.2 Device ID Register—Offset 02h

This section describes the device ID register.

Dword address = 00h

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:16 | Device ID | R | Identifies this device as the 21154. Internally hardwired to be 26h. |

### 15.1.3 Primary Command Register—Offset 04h

This section describes the primary command register.

These bits affect the behavior of the 21154 primary interface, except where noted. Some of the bits are repeated in the bridge control register, to act on the secondary interface.

This register must be initialized by configuration software.

Dword address = 04h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 0 | I/O space enable | R/W | Controls the 21154's response to I/O transactions on the primary interface.<br>When 0: The 21154 does not respond to I/O transactions initiated on the primary bus.<br>When 1: The 21154 response to I/O transactions initiated on the secondary bus is enabled.<br>Reset value: 0. |
| 1 | Memory space enable | R/W | Controls the 21154's response to memory transactions on the 21154 primary interface.<br>When 0: The 21154 does not respond to memory transactions initiated on the primary bus.<br>When 1: The 21154 response to memory transactions initiated on the primary bus is enabled.<br>Reset value: 0. |
| 2 | Master enable | R/W | Controls the 21154's ability to initiate memory and I/O transactions on the primary bus on behalf of an initiator on the secondary bus. Forwarding of configuration transactions is not affected.<br>When 0: The 21154 does not respond to I/O or memory transactions on the secondary interface and does not initiate I/O or memory transactions on the primary interface.<br>When 1: The 21154 is enabled to operate as an initiator on the primary bus and responds to I/O and memory transactions initiated on the secondary bus.<br>Reset value: 0. |
| 3 | Special cycle enable | R | The 21154 ignores special cycle transactions, so this bit is read only and returns 0. |
| 4 | Memory write and invalidate enable | R | The 21154 generates memory write and invalidate transactions only when operating on behalf of another master whose memory write and invalidate transaction is crossing the 21154.<br>This bit is read only and returns 0. |
| 5 | VGA snoop enable | R/W | Controls the 21154's response to VGA-compatible palette write transactions. VGA palette write transactions correspond to I/O transactions whose address bits are as follows:<br>• p_ad<9:0> are equal to 3C6h, 3C8h, and 3C9h.<br>• p_ad<15:10> are not decoded.<br>• p_ad<31:16> must be 0.<br>When 0: VGA palette write transactions on the primary interface are ignored unless they fall inside the 21154's I/O address range.<br>When 1: VGA palette write transactions on the primary interface are positively decoded and forwarded to the secondary interface.<br>Reset value: 0. |

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 6 | Parity error response | R/W | Controls the 21154's response when a parity error is detected on the primary interface.<br>When 0: The 21154 does not assert p_perr_l, nor does it set the data parity reported bit in the status register. The 21154 does not report address parity errors by asserting p_serr_l.<br>When 1: The 21154 drives p_perr_l and conditionally sets the data parity reported bit in the status register when a data parity error is detected (see Section 7.0). The 21154 allows p_serr_l assertion when address parity errors are detected on the primary interface.<br>Reset value: 0. |
| 7 | Wait cycle control | R | Reads as 0 to indicate that the 21154 does not perform address or data stepping. |
| 8 | SERR# enable | R/W | Controls the enable for p_serr_l on the primary interface.<br>When 0: Signal p_serr_l cannot be driven by the 21154.<br>When 1: Signal p_serr_l can be driven low by the 21154 under the conditions described in Section 7.4.<br>Reset value: 0. |
| 9 | Fast back-to-back enable | R/W | Controls the ability of the 21154 to generate fast back-to-back transactions on the primary bus.<br>When 0: The 21154 does not generate back-to-back transactions on the primary bus.<br>When 1: The 21154 is enabled to generate back-to-back transactions on the primary bus.<br>Reset value: 0. |
| 15:10 | Reserved | R | Reserved. Returns 0 when read. |

### 15.1.4 Primary Status Register—Offset 06h

This section describes the primary status register.

These bits affect the status of the 21154 primary interface. Bits reflecting the status of the secondary interface are found in the secondary status register. W1TC indicates that writing 1 to a bit sets that bit to 0. Writing 0 has no effect.

Dword address = 04h

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 19:16 | Reserved | R | Reserved. Returns 0 when read. |
| 20 | ECP | R | Enhanced capabilities port (ECP) enable. Reads as 1 in the 21154–AB and later revisions to indicate that the 21154–AB supports an enhanced capabilities list. The 21154–AA reads as 0 to show that this capability is not supported. |

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 21 | 66-MHz capable | R | Indicates whether the primary interface is 66-MHz capable.<br>Reads as 0 when pin config66 is tied low to indicate that the 21154 is not 66 MHz capable.<br>Reads as 1 when pin config66 is tied high to indicate that the primary bus is 66 MHz capable |
| 22 | Reserved | R | Reserved. Returns 0 when read. |
| 23 | Fast back-to-back capable | R | Reads as 1 to indicate that the 21154 is able to respond to fast back-to-back transactions on the primary interface. |
| 24 | Data parity detected | R/W1TC | This bit is set to 1 when all of the following are true:<br>• The 21154 is a master on the primary bus.<br>• Signal p_perr_l is detected asserted, or a parity error is detected on the primary bus.<br>• The parity error response bit is set in the command register.<br>Reset value: 0. |
| 26:25 | DEVSEL# timing | R | Indicates slowest response to a nonconfiguration command on the primary interface.<br>Reads as 01b to indicate that the 21154 responds no slower than with medium timing. |
| 27 | Signaled target abort | R/W1TC | This bit is set to 1 when the 21154 is acting as a target on the primary bus and returns a target abort to the primary master.<br>Reset value: 0. |
| 28 | Received target abort | R/W1TC | This bit is set to 1 when the 21154 is acting as a master on the primary bus and receives a target abort from the primary target.<br>Reset value: 0. |
| 29 | Received master abort | R/W1TC | This bit is set to 1 when the 21154 is acting as a master on the primary bus and receives a master abort. |
| 30 | Signaled system error | R/W1TC | This bit is set to 1 when the 21154 has asserted p_serr_l.<br>Reset value: 0. |
| 31 | Detected parity error | R/W1TC | This bit is set to 1 when the 21154 detects an address or data parity error on the primary interface.<br>Reset value: 0. |

### 15.1.5 Revision ID Register—Offset 08h

This section describes the revision ID register.

Dword = 08h

Byte enable p_cbe_l<3:0> = xxx0b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 7:0 | Revision ID | R | Indicates the revision number of this device. |

### 15.1.6 Programming Interface Register—Offset 09h

This section describes the programming interface register.

Dword address = 08h

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:8 | Programming interface | R | No programming interfaces have been defined for PCI-to-PCI bridges.<br>Reads as 0. |

### 15.1.7 Subclass Code Register—Offset 0Ah

This section describes the subclass code register.

Dword address = 08h

Byte enable p_cbe_l<3:0> = x0xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 23:16 | Subclass code | R | Reads as 04h to indicate that this bridge device is a PCI-to-PCI bridge. |

### 15.1.8 Base Class Code Register—Offset 0Bh

This section describes the base class code register.

Dword address = 08h

Byte enable p_cbe_l<3:0> = 0xxxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:24 | Base class code | R | Reads as 06h to indicate that this device is a bridge device. |

### 15.1.9 Cache Line Size Register—Offset 0Ch

This section describes the cache line size register.

Dword address = 0Ch

Byte enable p_cbe_l<3:0**>** = xxx0b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 7:0 | Cache line size | R/W | Designates the cache line size for the system in units of 32-bit Dwords. Used for prefetching memory read transactions and for terminating memory write and invalidate transactions.<br>The cache line size should be written as a power of 2. If the value is not a power of 2 or is greater than 16, the 21154 behaves as if the cache line size were 0.<br>Reset value: 0. |

### 15.1.10 Primary Latency Timer Register—Offset 0Dh

This section describes the primary latency timer register.

Dword address = 0Ch

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:8 | Master latency timer | R/W | Master latency timer for the primary interface. Indicates the number of PCI clock cycles from the assertion of p_frame_l to the expiration of the timer when the 21154 is acting as a master on the primary interface. All bits are writable, resulting in a granularity of one PCI clock cycle.<br>When 0: The 21154 relinquishes the bus after the first data transfer when the 21154's primary bus grant has been deasserted, with the exception of memory write and invalidate transactions.<br>Reset value: 0. |

### 15.1.11 Header Type Register—Offset 0Eh

This section describes the header type register.

Dword address = 0Ch

Byte enable p_cbe_l<3:0> = x0xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 23:16 | Header type | R | Defines the layout of addresses 10h through 3Fh in configuration space.<br>Reads as 01h to indicate that the register layout conforms to the standard PCI-to-PCI bridge layout. |

### 15.1.12 Primary Bus Number Register—Offset 18h

This section describes the primary bus number register.

This register must be initialized by configuration software.

Dword address = 18h

Byte enable p_cbe_l<3:0> = xxx0b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 7:0 | Primary bus number | R/W | Indicates the number of the PCI bus to which the primary interface is connected. The 21154 uses this register to decode Type 1 configuration transactions on the secondary interface that should either be converted to special cycle transactions on the primary interface or passed upstream unaltered.<br>Reset value: 0. |

### 15.1.13 Secondary Bus Number Register—Offset 19h

This section describes the secondary bus number register.

This register must be initialized by configuration software.

Dword address = 18h

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:8 | Secondary bus number | R/W | Indicates the number of the PCI bus to which the secondary interface is connected. The 21154 uses this register to determine when to respond to and forward Type 1 configuration transactions on the primary interface, and to determine when to convert them to Type 0 or special cycle transactions on the secondary interface.<br>Reset value: 0. |

### 15.1.14 Subordinate Bus Number Register—Offset 1Ah

This section describes the subordinate bus number register.

This register must be initialized by configuration software.

Dword address = 18h

Byte enable p_cbe_l<3:0> = x0xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 23:16 | Subordinate bus number | R/W | Indicates the number of the highest numbered PCI bus that is behind (or subordinate to) the 21154. Used in conjunction with the secondary bus number to determine when to respond to Type 1 configuration transactions on the primary interface and pass them to the secondary interface as a Type 1 configuration transaction.<br>Reset value: 0. |

### 15.1.15 Secondary Latency Timer Register—Offset 1Bh

This section describes the secondary latency timer register.

Dword address = 18h

Byte enable p_cbe_l<3:0> = 0xxxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:24 | Secondary latency timer | R/W | Master latency timer for the secondary interface. Indicates the number of PCI clock cycles from the assertion of s_frame_l to the expiration of the timer when the 21154 is acting as a master on the secondary interface. All bits are writable, resulting in a granularity of one PCI clock cycle.<br>When 0: The 21154 ends the transaction after the first data transfer when the 21154's secondary bus grant has been deasserted, with the exception of memory write and invalidate transactions.<br>Reset value: 0. |

## 15.1.16 I/O Base Address Register—Offset 1Ch

This section describes the I/O base address register.

This register must be initialized by configuration software.

Dword address = 1Ch

Byte enable p_cbe_l<3:0> = xxx0b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 3:0 | 32-bit indicator | R | The low 4 bits of this register read as 1h to indicate that the 21154 supports 32-bit I/O address decoding. |
| 7:4 | I/O base address <15:12> | R/W | Defines the bottom address of an address range used by the 21154 to determine when to forward I/O transactions from one interface to the other. The upper 4 bits are writable and correspond to address bits <15:12>. The lower 12 bits of the address are assumed to be 0. The upper 16 bits corresponding to address bits <31:16> are defined in the I/O base address upper 16 bits register. The I/O address range adheres to 4KB alignment and granularity.<br>Reset value: 0. |

## 15.1.17 I/O Limit Address Register—Offset 1Dh

This section describes the I/O limit address register.

This register must be initialized by configuration software.

Dword address = 1Ch

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 11:8 | 32-bit indicator | R/W | The low 4 bits of this register read as 1h to indicate that the 21154 supports 32-bit I/O address decoding. |

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:12 | I/O limit address <15:12> | R/W | Defines the top address of an address range used by the 21154 to determine when to forward I/O transactions from one interface to the other. The upper 4 bits are writable and correspond to address bits <15:12>. The lower 12 bits of the address are assumed to be FFFh. The upper 16 bits corresponding to address bits <31:16> are defined in the I/O limit address upper 16 bits register. The I/O address range adheres to 4KB alignment and granularity.<br>Reset value: 0. |

## 15.1.18 Secondary Status Register—Offset 1Eh

This section describes the secondary status register.

These bits reflect the status of the the 21154 secondary interface. W1TC indicates that writing 1 to that bit sets the bit to 0. Writing 0 has no effect.

Dword address = 1Ch

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description (Sheet 1 of 2) |
|---|---|---|---|
| 20:16 | Reserved | R | Reserved. Returns 0 when read. |
| 21 | 66-MHz capable | R | Indicates whether the secondary interface is 66-MHz capable.<br>Reads as 0 when pin config66 is tied low to indicate that the 21154 is not 66 MHz capable.<br>Reads as 1 when pin config66 is tied high to indicate that the secondary bus is 66 MHz capable |
| 22 | Reserved | R | Reserved. Returns 0 when read. |
| 23 | Fast back-to-back capable | R | Reads as 1 to indicate that the 21154 is able to respond to fast back-to-back transactions on the secondary interface. |
| 24 | Data parity detected | R/W1TC | This bit is set to 1 when all of the following are true:<br>• The 21154 is a master on the secondary bus.<br>• Signal s_perr_l is detected asserted, or a parity error is detected on the secondary bus.<br>• The parity error response bit is set in the bridge control register.<br>Reset value: 0. |
| 26:25 | s_devsel_l timing | R | Indicates slowest response to a command on the secondary interface.<br>Reads as 01b to indicate that the 21154 responds with medium (or faster) timing. |
| 27 | Signaled target abort | R/W1TC | This bit is set to 1 when the 21154 is acting as a target on the secondary bus and returns a target abort to the secondary bus master.<br>Reset value: 0. |

| Dword Bit | Name | R/W | Description (Sheet 2 of 2) |
|---|---|---|---|
| 28 | Received target abort | R/W1TC | This bit is set to 1 when the 21154 is acting as a master on the secondary bus and receives a target abort from the secondary bus target.<br>Reset value: 0. |
| 29 | Received master abort | R/W1TC | This bit is set to 1 when the 21154 is acting as an initiator on the secondary bus and receives a master abort.<br>Reset value: 0. |
| 30 | Received system error | R/W1TC | This bit is set to 1 when the 21154 detects the assertion of s_serr_l on the secondary interface.<br>Reset value: 0. |
| 31 | Detected parity error | R/W1TC | This bit is set to 1 when the 21154 detects an address or data parity error on the secondary interface.<br>Reset value: 0. |

### 15.1.19 Memory Base Address Register—Offset 20h

This section describes the memory base address register.

This register must be initialized by configuration software.

Dword address = 20h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 3:0 | Reserved | R | The low 4 bits of this register are read only and return 0. |
| 15:4 | Memory base address <31:20> | R/W | Defines the bottom address of an address range used by the 21154 to determine when to forward memory transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be 0. The memory address range adheres to 1MB alignment and granularity.<br>Reset value: 0. |

### 15.1.20 Memory Limit Address Register—Offset 22h

This section describes the memory limit address register.

This register must be initialized by configuration software.

Dword address = 20h

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 19:16 | Reserved | R | The low 4 bits of this register are read only and return 0. |
| 31:20 | Memory limit address <31:20> | R/W | Defines the top address of an address range used by the 21154 to determine when to forward memory transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be FFFFFh. The memory address range adheres to 1MB alignment and granularity.<br>Reset value: 0. |

### 15.1.21 Prefetchable Memory Base Address Register—Offset 24h

This section describes the prefetchable memory base address register.

This register must be initialized by configuration software.

Dword address = 24h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 3:0 | 64-bit indicator | R | The low 4 bits of this register are read only and return 1h to indicate that this range supports 64-bit addressing. |
| 15:4 | Prefetchable memory base address <31:20> | R/W | Defines the bottom address of an address range used by the 21154 to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be 0. The memory base register upper 32 bits contains the upper half of the base address. The memory address range adheres to 1MB alignment and granularity.<br>Reset value: 0. |

### 15.1.22 Prefetchable Memory Limit Address Register—Offset 26h

This section describes the prefetchable memory limit address register.

This register must be initialized by configuration software.

Dword address = 24h

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 19:16 | 64-bit indicator | R | The low 4 bits of this register are read only and return 1h to indicate that this range supports 64-bit addressing. |
| 31:20 | Prefetchable memory limit address <31:20> | R/W | Defines the top address of an address range used by the 21154 to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be FFFFFh. The memory limit upper 32 bits register contains the upper half of the limit address. The memory address range adheres to 1MB alignment and granularity.<br>Reset value: 0. |

### 15.1.23 Prefetchable Memory Base Address Upper 32 Bits Register—Offset 28h

This section describes the prefetchable memory base address upper 32 bits register.

This register must be initialized by configuration software.

Dword address = 28h

Byte enable p_cbe_l<3:0> = 0000b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:0 | Upper 32 prefetchable memory base address <63:32> | R/W | Defines the upper 32 bits of a 64-bit bottom address of an address range used by the 21154 to determine when to forward memory read and write transactions from one interface to the other. The memory address range adheres to 1MB alignment and granularity.<br>Reset value: 0. |

### 15.1.24 Prefetchable Memory Limit Address Upper 32 Bits Register—Offset 2Ch

This section describes the prefetchable memory limit address upper 32 bits register.

This register must be initialized by configuration software.

Dword address = 2Ch

Byte enable p_cbe_l<3:0> = 0000b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:0 | Upper 32 prefetchable memory limit address <63:32> | R/W | Defines the upper 32 bits of a 64-bit top address of an address range used by the 21154 to determine when to forward memory read and write transactions from one interface to the other. Extra read transactions should have no side effects. The memory address range adheres to 1MB alignment and granularity.<br>Reset value: 0. |

### 15.1.25 I/O Base Address Upper 16 Bits Register—Offset 30h

This section describes the I/O base address upper 16 bits register.

Dword address = 30h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:0 | I/O base address upper 16 bits <31:16> | R/W | Defines the upper 16 bits of a 32-bit bottom address of an address range used by the 21154 to determine when to forward I/O transactions from one interface to the other. The I/O address range adheres to 4KB alignment and granularity.<br>Reset value: 0. |

### 15.1.26 I/O Limit Address Upper 16 Bits Register—Offset 32h

This section describes the I/O limit address upper 16 bits register.

This register must be initialized by configuration software.

Dword address = 32h

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:16 | I/O limit address upper 16 bits <31:16> | R/W | Defines the upper 16 bits of a 32-bit top address of an address range used by the 21154 to determine when to forward I/O transactions from one interface to the other. The I/O address range adheres to 4KB alignment and granularity.<br>Reset value: 0. |

### 15.1.27 Subsystem Vendor ID Register—Offset 34h

This section describes the subsystem vendor ID register.

Dword address = 34h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:0 | Subsystem vendor ID | R/W | Provides a mechanism allowing add-in cards to distinguish their cards from one another. The 21154 provides a writable subsystem vendor ID that can be initialized during POST. This register is implemented only in the 21154–AA.<br>Reset to 0. |

## 15.1.28 ECP Pointer Register—Offset 34h

This section describes the ECP pointer register.

Dword address = 34h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 7:0 | ECP_PTR | R | Enhanced capabilities port (ECP) offset pointer. Reads as DCh in the 21154–AB and later revisions to indicate that the first item, which corresponds to the power management registers, resides at that configuration offset. This is a R/W register with no side effects in the 21154–AA. |
| 31:8 | Reserved | R | Reserved. The 21154–AB and later revisions return to 0 when read. This is a R/W register with no side effects in the 21154–AA. |

## 15.1.29 Subsystem ID Register—Offset 36h

This section describes the subsystem ID register.

Dword address = 34h

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:16 | Subsystem ID | R/W | Provides a mechanism allowing add-in cards to distinguish their cards from one another. The 21154 provides a writable subsystem ID that can be initialized during POST. This register is implemented only in the 21154–AA<br>Reset to 0. |

## 15.1.30 Interrupt Pin Register—Offset 3Dh

This section describes the interrupt pin register.

Dword address = 3Ch

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:8 | Interrupt pin | R | Reads as 0 to indicate that the 21154 does not have an interrupt pin. |

### 15.1.31 Bridge Control Register—Offset 3Eh

This section describes the bridge control register.

This register must be initialized by configuration software.

Dword address = 3Eh

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 16 | Parity error response | R/W | Controls the 21154's response when a parity error is detected on the secondary interface.<br>When 0: The 21154 does not assert s_perr_l, nor does it set the data parity reported bit in the secondary status register. The 21154 does not report address parity errors by asserting p_serr_l.<br>When 1: The 21154 drives s_perr_l and conditionally sets the data parity reported bit in the secondary status register when a data parity error is detected on the secondary interface (see Section 7.0.) Also must be set to 1 to allow p_serr_l assertion when address parity errors are detected on the secondary interface.<br>Reset value: 0. |
| 17 | SERR# forward enable | R/W | Controls whether the 21154 asserts p_serr_l when it detects s_serr_l asserted.<br>When 0: The 21154 does not drive p_serr_l when it detects s_serr_l asserted.<br>When 1: The 21154 asserts p_serr_l when s_serr_l is detected asserted (the primary SERR# driver enable bit must also be set).<br>Reset value: 0. |
| 18 | ISA enable | R/W | Modifies the 21154's response to ISA I/O addresses. Applies only to those addresses falling within the I/O base and limit address registers and within the first 64KB of PCI I/O space.<br>When 0: The 21154 forwards all I/O transactions downstream that fall within the I/O base and limit address registers.<br>When 1: The 21154 ignores primary bus I/O transactions within the I/O base and limit address registers and within the first 64KB of PCI I/O space that address the last 768 bytes in each 1KB block. Secondary bus I/O transactions are forwarded upstream if the address falls within the last 768 bytes in each 1KB block.<br>Reset value: 0. |

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 19 | VGA enable | R/W | Modifies the 21154's response to VGA-compatible addresses.<br>When 0: VGA transactions are ignored on the primary bus unless they fall within the I/O base and limit address registers and the ISA mode is 0.<br>When 1: The 21154 positively decodes and forwards the following transactions downstream, regardless of the values of the I/O base and limit registers, ISA mode bit, or VGA snoop bit:<br>• Memory transactions addressing 000A0000h–000BFFFFh<br>• I/O transaction addressing:<br>— p_ad<9:0> = 3B0h–3BBh and 3C0h–3DFh<br>— p_ad<15:10> are not decoded.<br>— p_ad<31:16> = 0000h.<br>I/O and memory space enable bits must be set in the command register.<br>The transactions listed here are ignored by the 21154 on the secondary bus.<br>Reset value: 0. |
| 20 | Reserved | R | Reserved. Returns 0 when read. |
| 21 | Master abort mode | R/W | Controls the 21154's behavior when a master abort termination occurs in response to a transaction initiated by the 21154 on either the primary or secondary PCI interface.<br>When 0: The 21154 asserts TRDY# on the initiator bus for delayed transactions, and FFFF FFFFh for read transactions. For posted write transactions, p_serr_l is not asserted.<br>When 1: The 21154 returns a target abort on the initiator bus for delayed transactions. For posted write transactions, the 21154 asserts p_serr_l if the SERR# enable bit is set in the command register.<br>Reset value: 0. |
| 22 | Secondary bus reset | R/W | Controls s_rst_l on the secondary interface.<br>When 0: The 21154 deasserts s_rst_l.<br>When 1: The 21154 asserts s_rst_l. When s_rst_l is asserted, the data buffers and the secondary interface are initialized back to reset conditions. The primary interface and configuration registers are not affected by the assertion of s_rst_l.<br>Reset value: 0. |
| 23 | Fast back-to-back enable | | Controls the ability of the 21154 to generate fast back-to-back transactions on the secondary interface.<br>When 0: The 21154 does not generate fast back-to-back transactions on the secondary PCI bus.<br>When 1: The 21154 is enabled to generate fast back-to-back transactions on the secondary PCI bus.<br>Reset value: 0. |

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 24 | Primary master timeout | R/W | Sets the maximum number of PCI clock cycles that the 21154 waits for an initiator on the primary bus to repeat a delayed transaction request. The counter starts once the delayed transaction completion is at the head of the queue. If the master has not repeated the transaction at least once before the counter expires, the 21154 discards the transaction from its queues.<br>When 0: The primary master timeout value is $2^{15}$ PCI clock cycles, or 0.983 ms for a 33-MHz bus.<br>When 1: The value is $2^{10}$ PCI clock cycles, or 30.7 µs for a 33-MHz bus.<br>Reset value: 0. |
| 25 | Secondary master timeout | R/W | Sets the maximum number of PCI clock cycles that the 21154 waits for an initiator on the secondary bus to repeat a delayed transaction request. The counter starts once the delayed transaction completion is at the head of the queue. If the master has not repeated the transaction at least once before the counter expires, the 21154 discards the transaction from its queues.<br>When 0: The primary master timeout value is $2^{15}$ PCI clock cycles, or 0.983 ms for a 33-MHz bus.<br>When 1: The value is $2^{10}$ PCI clock cycles, or 30.7 µs for a 33-MHz bus.<br>Reset value: 0. |
| 26 | Master timeout status | R/W1TC | This bit is set to 1 when either the primary master timeout counter or the secondary master timeout counter expires and a delayed transaction is discarded from the 21154's queues. Write 1 to clear.<br>Reset value: 0. |
| 27 | Master timeout SERR# enable | R/W | Controls assertion of p_serr_l during a master timeout.<br>When 0: Signal p_serr_l is not asserted as a result of a master timeout.<br>When 1: Signal p_serr_l is asserted when either the primary master timeout counter or the secondary master timeout counter expires and a delayed transaction is discarded from the 21154's queues. The SERR# enable bit in the command register must also be set.<br>Reset value: 0. |
| 31:28 | Reserved | R | Reserved. Returns 0 when read. |

## 15.1.32 Capability ID Register—Offset DCh

This section describes the capability ID register. (Implemented in the 21154–AB and later revisions only. In the 21154–AA, this register is reserved.)

Dword address = DCh

Byte enable p_cbe_l<3:0> = xxx0b

| Dword Bits | Name | R/W | Description |
|---|---|---|---|
| 7:0 | CAP_ID | R | Enhanced capabilities ID. Reads only as 01h to indicate that this is the power management enhanced capability register. |

### 15.1.33 Next Item Ptr Register—Offset DDh

This section describes the next item ptr register. (Implemented in the 21154–AB and later revisions only. In the 21154–AA, this register is reserved.)

Dword address = DCh

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 15:8 | NEXT_ITEM | R | Next item pointer. Reads as 0 to indicate that there is no other ECP register. |

### 15.1.34 Power Management Capabilities Register—Offset DEh

This section describes the power management capabilities register. (Implemented in the 21154–AB and later revisions only. In the 21154–AA, this register is reserved.)

Dword address = DCh

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 18:16 | PM_VER | R | Power Management Revision. Reads as 001 to indicate that this device is compliant to Revision 1.0 of the *PCI Power Management Interface Specification*. |
| 19 | PME#Clock | R | PME# Clock Required. Reads as 0 because this device does not support the PME# pin. |
| 20 | AUX | R | Auxiliary Power Support. Reads as 0 because this device does not have PME# support or an auxiliary power source. |
| 21 | DSI | R | Device-Specific Initialization. Reads as 0 to indicate that this device does not have device-specific initialization requirements. |
| 24:22 | Reserved | R | Reserved. Read as 000b. |
| 25 | D1 | R | D1 Power State Support. Reads as 0 to indicate that this device does not support the D1 power management state. |
| 26 | D2 | R | D2 Power State Support. Reads as 0 to indicate that this device does not support the D2 power management state. |
| 31:27 | PME_SUP | R | PME# Support. Reads as 0 to indicate that this device does not support the PME# pin. |

### 15.1.35 Power Management Control and Status Register—Offset E0h

This section describes the power management control and status register. (Implemented in the 21154–AB and later revisions only. In the 21154–AA, this register is reserved.)

Dword address = E0h

Byte enable p_cbe_l = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 1:0 | PWR_STATE | R/W | Power State. Reflects the current power state of this device. If an unimplemented power state is written to this register, the 21154 completes the write transaction, ignores the write data, and does not change the value of this field. Writing a value of D0 when the previous state was D3 will cause a chip reset to occur (without asserting s_rst_l).<br>• 00b: D0<br>• 01b: D1 (not implemented)<br>• 10b: D2 (not implemented)<br>• 11b: D3<br>Reset value: 00b |
| 7:2 | Reserved | R | Reserved. Reads as 00000b. |
| 8 | PME_EN | R | PME# Enable. Reads as 0 because the PME# pin is not implemented. |
| 12:9 | DATA_SEL | R | Data Select. Reads as 0000b because the data register is not implemented. |
| 14:13 | DATA_ SCALE | R | Data Scale. Reads as 00b because the data register is not implemented. |
| 15 | PME_STAT | R | PME Status. Reads as 0 because the PME# pin is not implemented. |

### 15.1.36 PPB Support Extensions Registers—Offset E2h

This section describes the PPB support extensions registers. (Implemented in the 21154–AB and later revisions only. In the 21154–AA, these registers are reserved.)

Dword address = E0h

Byte enable p_cbe_l<3:0> = x0xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 21:16 | Reserved | R | Reserved. Read only as 000000b. |
| 22 | B2_B3 | R | B2_B3 Support for $D3_{hot.}$ When the BPCC_En (bit 23) reads as 1, this bit reads as 1 to indicate that the secondary bus clock outputs will be stopped and driven low when this device is placed in $D3_{hot}$. This bit is not defined when the BPCC_En bit reads as 0. |
| 23 | BPCC_En | R | Bus Power/Clock Control Enable. When the bpcce pin is tied high, this bit reads as a 1 to indicate that the bus power/clock control mechanism is enabled, as described in B2_B3 (bit 23). When the bpcce pin is tied low, this bit reads as a 0 to indicate that the bus power/clock control mechanism is disabled (secondary clocks are not disabled when this device is placed in $D3_{hot}$). |

### 15.1.37 Data Register — Offset E3h

This section describes the data register.

Dword address = E0h

Byte enable p_cbe_l<3:0> = 0xxxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 31:24 | Data | R | Data register. This register is not implemented and reads 00h. |

## 15.2 Device-Specific Configuration Registers

This section provides a detailed description of the 21154 device-specific configuration registers.

Each field has a separate description.

Fields that have the same configuration address are selectable by turning on (driving low) the appropriate byte enable bits on p_cbe_l during the data phase. To select all fields of a configuration address, drive all byte enable bits low.

All reserved fields and registers are read only and always return 0.

### 15.2.1 Chip Control Register—Offset 40h

This section describes the chip control register.

Dword address = 40h

Byte enable p_cbe_l<3:0> = xxx0b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 0 | Reserved | R | Reserved. Returns 0 when read. |
| 1 | Memory write disconnect control | R/W | Controls when the 21154, as a target, disconnects memory write transactions.<br>When 0: The 21154 disconnects on queue full or on a 4KB boundary.<br>When 1: The 21154 disconnects on a cache line boundary, as well as when the queue fills or on a 4KB boundary.<br>Reset value: 0. |
| 3:2 | Reserved | R | Reserved. Returns 0 when read. |
| 4 | Secondary bus prefetch disable | R/W | Controls the 21154's ability to prefetch during upstream memory read transactions.<br>When 0: The 21154 prefetches and does not forward byte enable bits during memory read transactions.<br>When 1: The 21154 requests only one Dword from the target during memory read transactions and forwards read byte enable bits. The 21154 returns a target disconnect to the requesting master on the first data transfer. Memory read line and memory read multiple transactions are still prefetchable.<br>Reset value: 0. |
| 5 | Live insertion mode | R/W | Enables hardware control of transaction forwarding in the 21154.<br>When 0: Pin gpio<3> has no effect on the I/O, memory, and master enable bits.<br>When 1: If the output enable control for gpio<3> is set to input only in the gpio output enable control register, this bit enables gpio<3> to mask the I/O enable, memory enable, and master enable bits to 0. These enable bits are masked when gpio<3> is driven high. When this occurs, the 21154 stops accepting I/O and memory transactions.<br>Reset value: 0. |
| 7:6 | Reserved | R | Reserved. Returns 0 when read. |

### 15.2.2 Diagnostic Control Register—Offset 41h

This section describes the diagnostic control register.

W1TR indicates that writing 1 in this bit position causes a chip reset to occur. Writing 0 has no effect.

Dword address = 40h

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 8 | Chip reset | R/W1TR | Chip and secondary bus reset control.<br>When 1: Causes the 21154 to perform a chip reset. Data buffers, configuration registers, and both the primary and secondary interfaces are reset to their initial state. The 21154 clears this bit once chip reset is complete. The 21154 can them be reconfigured.<br>Secondary bus reset s_rst_l is asserted and the secondary reset bit in the bridge control register is set when this bit is set. The secondary reset bit in the bridge control register must be cleared in order to deassert s_rst_l. |
| 10:9 | Test mode | R/W | Controls the testability of the 21154's internal counters. These bits are used for chip test only. The value of these bits controls which bytes of the counters are exercised:<br>• 00b = Normal functionality—all bits are exercised.<br>• 01b = Byte 1 is exercised.<br>• 10b = Byte 2 is exercised.<br>• 11b = Byte 0 is exercised.<br>Reset value: 00b. |
| 15:11 | Reserved | R | Reserved. Returns 0 when read. |

### 15.2.3 Arbiter Control Register—Offset 42h

This section describes the arbiter control register.

Dword address = 40h

Byte enable p_cbe_l<3:0> = 00xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 25:16 | Arbiter control | R/W | Each bit controls whether a secondary bus master is assigned to the high priority arbiter group or the low priority arbiter group. Bits <24:16> correspond to request inputs **s_req_l<8:0>**, respectively. Bit <25> corresponds to the 21154 as a secondary bus master.<br>When 0: Indicates that the master belongs to the low priority group.<br>When 1: Indicates that the master belongs to the high priority group.<br>Reset value: 10 0000 0000b. |
| 31:26 | Reserved | R | Reserved. Returns 0 when read. |

### 15.2.4 p_serr_l Event Disable Register—Offset 64h

This section describes the p_serr_l event disable register.

Dword address = 64h

Byte enable p_cbe_l<3:0> = xxx0b

| Dword Bit | Name | R/W | Description (Sheet 1 of 2) |
|---|---|---|---|
| 0 | Reserved | R | Reserved. Returns 0 when read. |
| 1 | Posted write parity error | R/W | Controls the 21154's ability to assert p_serr_l when a data parity error is detected on the target bus during a posted write transaction.<br>When 0: Signal p_serr_l is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>When 1: Signal p_serr_l is not asserted if this event occurs.<br>Reset value: 0. |
| 2 | Posted write nondelivery | R/W | Controls the 21154's ability to assert p_serr_l when it is unable to deliver posted write data after $2^{24}$ attempts.<br>When 0: Signal p_serr_l is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>When 1: Signal p_serr_l is not asserted if this event occurs.<br>Reset value: 0. |
| 3 | Target abort during posted write | R/W | Controls the 21154's ability to assert p_serr_l when it receives a target abort when attempting to deliver posted write data.<br>When 0: Signal p_serr_l is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>When 1: Signal p_serr_l is not asserted if this event occurs.<br>Reset value: 0. |
| 4 | Master abort on posted write | R/W | Controls the 21154's ability to assert p_serr_l when it receives a master abort when attempting to deliver posted write data.<br>When 0: Signal p_serr_l is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>When 1: Signal p_serr_l is not asserted if this event occurs.<br>Reset value: 0. |

| Dword Bit | Name | R/W | Description (Sheet 2 of 2) |
|---|---|---|---|
| 5 | Delayed write nondelivery | R/W | Controls the 21154's ability to assert p_serr_l when it is unable to deliver delayed write data after $2^{24}$ attempts.<br>When 0: Signal p_serr_l is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>When 1: Signal p_serr_l is not asserted if this event occurs.<br>Reset value: 0. |
| 6 | Delayed read—no data from target | R/W | Controls the 21154's ability to assert p_serr_l when it is unable to transfer any read data from the target after $2^{24}$ attempts.<br>When 0: Signal p_serr_l is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>When 1: Signal p_serr_l is not asserted if this event occurs.<br>Reset value: 0. |
| 7 | Reserved | R | Reserved. Returns 0 when read. |

## 15.2.5 gpio Output Data Register—Offset 65h

This section describes the gpio output data register.

Dword Address = 64h

Byte enable p_cbe_l<3:0> = xx0xb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 11:8 | GPIO output write-1-to-clear | R/W1TC | The gpio<3:0> pin output data write-1-to-clear. Writing 1 to any of these bits drives the corresponding bit low on the gpio<3:0> bus if it is programmed as bidirectional. Data is driven on the PCI clock cycle following completion of the configuration write to this register. Bit positions corresponding to gpio pins that are programmed as input only are not driven. Writing 0 to these bits has no effect. When read, reflects the last value written.<br>Reset value: 0. |
| 15:12 | GPIO output write-1-to-set | R/W1TS | The gpio<3:0> pin output data write-1-to-set. Writing 1 to any of these bits drives the corresponding bit high on the gpio<3:0> bus if it is programmed as bidirectional. Data is driven on the PCI clock cycle following completion of the configuration write to this register. Bit positions corresponding to gpio pins that are programmed as input only are not driven. Writing 0 to these bits has no effect. When read, reflects the last value written.<br>Reset value: 0. |

### 15.2.6 gpio Output Enable Control Register—Offset 66h

This section describes the gpio output enable control register.

Dword Address = 64h

Byte enable p_cbe_l<3:0> = x0xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 19:16 | GPIO output enable write-1-to-clear | R/W1TC | The gpio<3:0> pin output data write-1-to-enable. Writing 1 to any of these bits drives the corresponding gpio<3:0> pin as input only; that is, the output driver is tristated. Writing 0 to this register has no effect. When read, reflects the last value written.<br>Reset value: 0 (all pins are input only). |
| 23:20 | GPIO output enable write-1-to-set | R/W1TS | The gpio<3:0> pin output enable control write-1-to-set. Writing 1 to any of these bits configures the corresponding gpio<3:0> pin as bidirectional, that is, enables the output driver and drives the value set in the output data register (65h). Writing 0 to these bits has no effect. When read, reflects the last value written.<br>Reset value: 0 (all pins are input only). |

### 15.2.7 gpio Input Data Register—Offset 67h

This section describes the gpio input data register.

Dword address = 64h

Byte enable p_cbe_l<3:0> = 0xxxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 27:24 | Reserved | R | Reserved. Returns 0 when read. |
| 31:28 | GPIO input | R | This read-only register reads the state of the gpio<3:0> pins. This state is updated on the PCI clock cycle following a change in the gpio pins. |

### 15.2.8 Secondary Clock Control Register—Offset 68h

This section describes the secondary clock control register.

Dword address = 68h

Byte enable p_cbe_l<3:0> = xx00b

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 1:0 | Slot 0 clock disable | R/W | If either bit is 0: Signal s_clk_o<0> is enabled.<br>When both bits are 1: Signal s_clk_o<0> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. These bits are assigned to correspond to the PRSNT# pins for slot 0. |
| 3:2 | Slot 1 clock disable | R/W | If either bit is 0: Signal s_clk_o<1> is enabled.<br>When both bits are 1: Signal s_clk_o<1> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. These bits are assigned to correspond to the PRSNT# pins for slot 1. |
| 5:4 | Slot 2 clock disable | R/W | If either bit is 0: Signal s_clk_o<2> is enabled.<br>When both bits are 1: Signal s_clk_o<2> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. These bits are assigned to correspond to the PRSNT# pins for slot 2. |
| 7:6 | Slot 3 clock disable | R/W | If either bit is 0: Signal s_clk_o<3> is enabled.<br>When both bits are 1: Signal s_clk_o<3> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. These bits are assigned to correspond to the PRSNT# pins for slot 3. |
| 8 | Device 1 clock disable | R/W | When 0: Signal s_clk_o<4> is enabled.<br>When 1: Signal s_clk_o<4> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. |
| 9 | Device 2 clock disable | R/W | When 0: Signal s_clk_o<5> is enabled.<br>When 1: Signal s_clk_o<5> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. |
| 10 | Device 3 clock disable | R/W | When 0: Signal s_clk_o<6> is enabled.<br>When 1: Signal s_clk_o<6> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. |
| 11 | Device 4 clock disable | R/W | When 0: Signal s_clk_o<7> is enabled.<br>When 1: Signal s_clk_o<7> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. |

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 12 | Device 5 clock disable | R/W | When 0: Signal s_clk_o<8> is enabled.<br>When 1: Signal s_clk_o<8> is disabled and driven high.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. |
| 13 | The 21154 clock disable | R/W | When 1: Signal s_clk_o<9> is disabled and driven high.<br>When 0: Signal s_clk_o<9> is enabled.<br>Upon secondary bus reset, this bit is initialized by shifting in a serial data stream. This bit is assigned to correspond to the 21154 secondary clock input, s_clk. |
| 15:14 | Reserved | R | Reserved. Returns 1 when read. |

### 15.2.9 p_serr_l Status Register—Offset 6Ah

This section describes the p_serr_l status register.

This status register indicates the reason for the 21154's assertion of p_serr_l.

Dword address = 68h

Byte enable p_cbe_l<3:0> = x0xxb

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 0 | Address parity error | R/W1TC | When 1: Signal p_serr_l was asserted because an address parity error was detected on either the primary or secondary PCI bus.<br>Reset value: 0. |
| 1 | Posted write data parity error | R/W1TC | When 1: Signal p_serr_l was asserted because a posted write data parity error was detected on the target bus.<br>Reset value: 0. |
| 2 | Posted write nondelivery | R/W1TC | When 1: Signal p_serr_l was asserted because the 21154 was unable to deliver posted write data to the target after $2^{24}$ attempts.<br>Reset value: 0. |
| 3 | Target abort during posted write | R/W1TC | When 1: Signal p_serr_l was asserted because the 21154 received a target abort when delivering posted write data.<br>Reset value: 0. |
| 4 | Master abort during posted write | R/W1TC | When 1: Signal p_serr_l was asserted because the 21154 received a master abort when attempting to deliver posted write data.<br>Reset value: 0. |

| Dword Bit | Name | R/W | Description |
|---|---|---|---|
| 5 | Delayed write nondelivery | R/W1TC | When 1: Signal p_serr_l was asserted because the 21154 was unable to deliver delayed write data after $2^{24}$ attempts.<br>Reset value: 0. |
| 6 | Delayed read—no data from target | R/W1TC | When 1: Signal p_serr_l was asserted because the 21154 was unable to read any data from the target after $2^{24}$ attempts.<br>Reset value: 0. |
| 7 | Delayed transaction master timeout | R/W1TC | When 1: Signal p_serr_l was asserted because a master did not repeat a read or write transaction before the master timeout counter expired on the initiator's PCI bus.<br>Reset to 0. |

## 15.3 Configuration Register Values After Reset

Table 36 lists the value of the 21154 configuration registers after reset. Reserved registers are not listed and are always read only as 0.

**Table 36. Configuration Register Values After Reset (Sheet 1 of 2)**

| Byte Address | Register Name | Reset Value |
|---|---|---|
| 00—01h | Vendor ID | 1011h |
| 02—03h | Device ID | 0026h |
| 04—05h | Primary command | 0000h |
| 06—07h | Primary status | 0280h–21154-AA only<br>0290h–33 MHz 21154<br>02B0h–66 MHz capable 21154 |
| 08h | Revision ID | xxh[1] |
| 09—0Bh | Class code | 060400h |
| 0Ch | Cache line size | 00h |
| 0Dh | Primary latency timer | 00h |
| 0Eh | Header type | 01h |
| 18h | Primary bus number | 00h |
| 19h | Secondary bus number | 00h |
| 1Ah | Subordinate bus number | 00h |
| 1Bh | Secondary latency timer | 00h |
| 1Ch | I/O base address | 01h |
| 1Dh | I/O limit address | 01h |
| 1E—1Fh | Secondary status | 0280h–33 MHz 21154<br>02A0h–66 MHz capable 21154 |
| 20—21h | Memory base address | 0000h |
| 22—23h | Memory limit address | 0000h |
| 24—25h | Prefetchable memory base address | 0001h |

**Table 36. Configuration Register Values After Reset (Sheet 2 of 2)**

| Byte Address | Register Name | Reset Value |
|---|---|---|
| 26—27h | Prefetchable memory limit address | 0001h |
| 28—2Bh | Prefetchable memory base address upper 32 bits | 00000000h |
| 2C—2Fh | Prefetchable memory limit address upper 32 bits | 00000000h |
| 30—31h | I/O base address upper 16 bits | 0000h |
| 32—33h | I/O limit address upper 16 bits | 0000h |
| 34—35h | Subsystem vendor ID–21154-AA only<br>ECP pointer | 0000h<br>00DCh |
| 36—37h | Subsystem ID–21154-AA only<br>Reserved | 0000h |
| 3Dh | Interrupt pin | 00h |
| 3E—3Fh | Bridge control | 0000h |
| 40h | Chip control | 00h |
| 41h | Diagnostic control | 00h |
| 42—43h | Arbiter control | 0200h |
| 64h | **p_serr_l** event disable | 00h |
| 65h | gpio output data | 00h |
| 66h | **gpio** output enable control | 00h |
| 67h | **gpio** input data | 00h |
| 68—69h | Secondary clock control[2] | |
| 6Ah | **p_serr_l** status | 00h |
| DCh | Power management capability ID[3] | 01h |
| DDh | Next item pointer[3] | 00h |
| DEh–DFh | Power management capabilities register[3] | 0001h |
| E0h–E1h | Power management control and status[3] | 0000h |
| E2h | PPB support extensions[3] | 00h (**bpcce**=0)<br>C0h (bpcce = 1) |
| E3h | Power management data register[3] | 00h |

1. Dependent on revision of device.
2. The value of this register is dependent upon the serial clock disable shift function that occurs during secondary bus reset.
3. In the 21154–AA, these registers are reserved.

# 16.0 JTAG Test Port

This chapter describes the 21154's implementation of a joint test action group (JTAG) test port according to IEEE Std 1149.1, IEEE Standard Test Access Port and Boundary-Scan Architecture.

## 16.1 Overview

The 21154 contains a serial-scan test port that conforms to IEEE standard 1149.1. The JTAG test port consists of the following:

- A 5-wire test access port
- A test access port controller
- An instruction register
- A bypass register
- A boundary-scan register

***Note:*** The JTAG test access port is to be used only while the 21154 is not operating.

## 16.2 JTAG Signal Pins

This chapter describes the JTAG pins listed in Table 37.

**Table 37. JTAG Pins**

| Signal Name | Type | Description |
|---|---|---|
| tdi | Input | Serial boundary-scan data in |
| tdo | Output | Serial boundary-scan data |
| tms | Input | JTAG test mode select |
| tck | Input | Boundary-scan clock |
| trst_l | Input | JTAG test access port reset |

## 16.3 Test Access Port Controller

The test access port controller is a finite state machine that interprets IEEE 1149.1 protocols received through the tms line.

The state transitions in the controller are caused by the tms signal on the rising edge of tck. In each state, the controller generates appropriate clock and control signals that control the operation of the test features. After entry into a state, test feature operations are initiated on the rising edge of tck.

## 16.4 Instruction Register

The 5-bit instruction register selects the test modes and features. The instruction register bits are interpreted as instructions, as shown in Table 38. The instructions select and control the operation of the boundary-scan and bypass registers.

Table 38 describes the 21154's instructions.

**Table 38. JTAG Instruction Register**

| Instruction Register Contents | Instruction Name (Test Mode or State) | Test Register Selected | Operation |
|---|---|---|---|
| 00000 | EXTEST | Boundary-scan | External test (drives pins from the boundary-scan register) |
| 00001 | SAMPLE | Boundary-scan | Samples I/O |
| 00010 | BSROSC | Boundary-scan | Ring oscillates the boundary-scan register |
| 00011 | BSRDLY | Boundary-scan | Configures the boundary-scan register for propagation delay measurement |
| 00100 | CLAMP | Bypass | Drives pins from the boundary-scan register and selects the bypass register for shifts |
| 00101 | HIGHZ | Bypass | Tristates all output and I/O pins except the **tdo** pin |
| 00110–11111 | BYPASS | Bypass | Selects the bypass register for shifts |

The instruction register is loaded through the tdi pin. The instruction register has a shift-in stage from which the instruction is then loaded in parallel.

## 16.5 Bypass Register

The bypass register is a 1-bit shift register that provides a means for effectively bypassing the JTAG test logic through a single-bit serial connection through the chip from tdi to tdo. At board-level testing, this helps reduce overall length of the scan ring.

## 16.6 Boundary-Scan Register

The boundary-scan register is a single-shift register-based path formed by boundary-scan cells placed at the chip's signal pins. The register is accessed through the JTAG port's tdi and tdo pins.

### 16.6.1 Boundary-Scan Register Cells

Each boundary-scan cell operates in conjunction with the current instruction and the current state in the test access port controller state machine. The function of the BSR cells is determined by the associated pins, as follows:

- Input-only pins—The boundary-scan cell is basically a 1-bit shift register. The cell supports sample and shift functions.
- Output-only pins—The boundary-scan cell comprises a 1-bit shift register and an output multiplexer. The cell supports the sample, shift, and drive output functions.
- Bidirectional pins—The boundary-scan cell is identical to the output-only pin cell, but it captures test data from the incoming data line. The cell supports sample, shift, drive output, and hold output functions. It is used at all I/O pins.

### 16.6.2 21154 Boundary-Scan Order

Table 39 lists the boundary-scan register order and the group disable controls. The group disable control either enables or tristates its corresponding group of bidirectional drivers. When the value of a group disable control bit is 0, the output driver is enabled. When the value is 1, the driver is tristated. There are nine groups of bidirectional drivers, and therefore nine group disable control bits.

The Group Disable Number column in Table 39 shows which group disable bit controls the corresponding output driver. Group disable bits do not affect input-only pins, so those pins have a blank rather than a group number in the By Group Disable column. The group disable control wire can control pins on either side of where the group disable boundary-scan register is placed. The group disable boundary-scan registers have a boundary-scan register order number entry, but they do not have a corresponding pin number or signal name.

Data shifts from tdi into the most significant bit of the boundary-scan register, and from the least significant bit of the boundary-scan register out to tdo.

**Table 39. Boundary Scan Order (Sheet 1 of 7)**

| Pin Number | Signal Name | Boundary-Scan Order | By Group Disable | Group Disable Cell |
|---|---|---|---|---|
| D4 | s_req_l<0> | 88 | — | |
| C1 | s_req_l<1> | 89 | — | |
| C2 | s_req_l<2> | 90 | — | |
| D3 | s_req_l<3> | 91 | — | |
| E4 | s_req_l<4> | 92 | — | |
| D1 | s_req_l<5> | 93 | — | |
| D2 | s_req_l<6> | 94 | — | |
| E3 | s_req_l<7> | 95 | — | |
| E1 | s_req_l<8> | 96 | — | |
| E2 | s_gnt_l<0> | 97 | 5 | |
| F3 | s_gnt_l<1> | 98 | 5 | |
| F1 | s_gnt_l<2> | 99 | 5 | |

**Table 39. Boundary Scan Order (Sheet 2 of 7)**

| Pin Number | Signal Name | Boundary-Scan Order | By Group Disable | Group Disable Cell |
|---|---|---|---|---|
| F2 | s_gnt_l<3> | 100 | 5 | |
| — | Vss | 101 | — | Group Disable 5 |
| G1 | s_gnt_l<4> | 102 | 5 | |
| G4 | s_gnt_l<5> | 103 | 5 | |
| G2 | s_gnt_l<6> | 104 | — | |
| G3 | s_gnt_l<7> | 105 | 5 | |
| H1 | s_gnt_l<8> | 106 | 5 | |
| H2 | s_rst_l | 107 | 4 | |
| J4 | s_clk | 108 | — | |
| K1 | s_cfn_l | 109 | — | |
| K2 | gpio<3> | 110 | 4 | |
| K3 | gpio<2> | 111 | 4 | |
| L4 | gpio<1> | 112 | 4 | |
| L1 | gpio<0> | 113 | 4 | |
| L2 | s_clk_o<0> | 114 | 4 | |
| — | Vss | 115 | — | Group Disable 4 |
| L3 | s_clk_o<1> | 116 | 4 | |
| M3 | s_clk_o<2> | 117 | 4 | |
| M1 | s_clk_o<3> | 118 | 4 | |
| M2 | s_clk_o<4> | 119 | 4 | |
| N3 | s_clk_o<5> | 120 | 4 | |
| N1 | s_clk_o<6> | 121 | 4 | |
| P3 | s_clk_o<7> | 122 | 4 | |
| P2 | s_clk_o<8> | 123 | 4 | |
| P1 | s_clk_o<9> | 124 | 4 | |
| R3 | p_rst_l | 125 | — | |
| R2 | p_gnt_l | 126 | — | |
| T3 | p_clk | 127 | — | |
| T2 | Vss | 128 | — | Group Disable 3 |
| U3 | p_req_l | 129 | 3 | |
| U2 | p_ad<31> | 130 | 2 | |
| U4 | p_ad<30> | 131 | 2 | |
| U1 | p_ad<29> | 132 | 2 | |
| V2 | p_ad<28> | 133 | 2 | |
| V1 | p_ad<27> | 134 | 2 | |
| V3 | p_ad<26> | 135 | 2 | |
| W2 | p_ad<25> | 136 | 2 | |

**Table 39. Boundary Scan Order (Sheet 3 of 7)**

| Pin Number | Signal Name | Boundary-Scan Order | By Group Disable | Group Disable Cell |
|---|---|---|---|---|
| W1 | p_ad<24> | 137 | 2 | |
| Y2 | p_cbe_l<3> | 138 | 2 | |
| Y1 | p_idsel | 139 | — | |
| W4 | p_ad<23> | 140 | 2 | |
| Y3 | p_ad<22> | 141 | 2 | |
| AA1 | p_ad<21> | 142 | 2 | |
| AA3 | p_ad<20> | 143 | 2 | |
| Y4 | p_ad<19> | 144 | 2 | |
| AB3 | p_ad<18> | 145 | 2 | |
| AA4 | p_ad<17> | 146 | 2 | |
| Y5 | p_ad<16> | 147 | 2 | |
| AC4 | Vss | 148 | — | Group Disable 2 |
| AB4 | p_cbe_l<2> | 149 | 2 | |
| AA5 | p_frame_l | 150 | 1 | |
| AC5 | p_irdy_l | 151 | 1 | |
| AB5 | p_trdy_l | 152 | 1 | |
| AA6 | p_devsel_l | 153 | 1 | |
| AC6 | p_stop_l | 154 | 1 | |
| AB6 | p_lock_l | 155 | 1 | |
| — | Vss | 156 | — | Group Disable 1 |
| AC7 | p_perr_l | 157 | 1 | |
| Y7 | p_serr_l | 158 | 1 | |
| AB7 | p_par | 159 | 0 | |
| AA7 | p_cbe_l<1> | 160 | 0 | |
| AB8 | p_ad<15> | 161 | 0 | |
| AA8 | p_ad<14> | 162 | 0 | |
| AC9 | p_ad<13> | 163 | 0 | |
| AB9 | p_ad<12> | 164 | 0 | |
| AA9 | p_ad<11> | 165 | 0 | |
| AC10 | p_ad<10> | 166 | 0 | |
| AB10 | p_m66ena | 167 | 0 | |
| AA10 | p_ad<9> | 168 | 0 | |
| Y11 | p_ad<8> | 169 | 0 | |
| AC11 | p_cbe_l<0> | 170 | 0 | |
| AB11 | p_ad<7> | 171 | 0 | |
| AA11 | p_ad<6> | 172 | 0 | |
| AA12 | p_ad<5> | 173 | 0 | |

**Table 39. Boundary Scan Order (Sheet 4 of 7)**

| Pin Number | Signal Name | Boundary-Scan Order | By Group Disable | Group Disable Cell |
|---|---|---|---|---|
| AB12 | p_ad<4> | 174 | 0 | |
| AB13 | p_ad<3> | 175 | 0 | |
| AA13 | p_ad<2> | 176 | 0 | |
| Y13 | p_ad<1> | 177 | 0 | |
| AA14 | p_ad<0> | 178 | 0 | |
| AB14 | p_ack64_l | 179 | 0 | |
| AC14 | p_req64_l | 180 | 0 | |
| AA15 | p_cbe_l<7> | 181 | 0 | |
| AB15 | p_cbe_l<6> | 182 | 0 | |
| Y15 | p_cbe_l<5> | 183 | 0 | |
| AC15 | p_cbe_l<4> | 184 | 0 | |
| AA16 | p_ad<63> | 185 | 0 | |
| AB16 | p_ad<62> | 186 | 0 | |
| AA17 | p_ad<61> | 187 | 0 | |
| AB17 | p_ad<60> | 188 | 0 | |
| Y17 | p_ad<59> | 189 | 0 | |
| AB18 | p_ad<58> | 190 | 0 | |
| AC18 | p_ad<57> | 191 | 0 | |
| AA18 | p_ad<56> | 192 | 0 | |
| AC19 | p_ad<55> | 193 | 0 | |
| AA19 | p_ad<54> | 194 | 0 | |
| AB20 | p_ad<53> | 195 | 0 | |
| Y19 | p_ad<52> | 196 | 0 | |
| AA20 | p_ad<51> | 197 | 0 | |
| AB21 | p_ad<50> | 198 | 0 | |
| AC21 | p_ad<49> | 199 | 0 | |
| AA21 | p_ad<48> | 200 | 0 | |
| Y20 | p_ad<47> | 201 | 0 | |
| AA23 | p_ad<46> | 202 | 0 | |
| Y21 | p_ad<45> | 203 | 0 | |
| W20 | p_ad<44> | 204 | 0 | |
| Y23 | p_ad<43> | 205 | 0 | |
| W21 | p_ad<42> | 206 | 0 | |
| W23 | p_ad<41> | 207 | 0 | |
| W22 | p_ad<40> | 208 | 0 | |
| V21 | p_ad<39> | 209 | 0 | |
| V23 | p_ad<38> | 210 | 0 | |

**Table 39. Boundary Scan Order (Sheet 5 of 7)**

| Pin Number | Signal Name | Boundary-Scan Order | By Group Disable | Group Disable Cell |
|---|---|---|---|---|
| V22 | p_ad<37> | 211 | 0 | |
| U23 | p_ad<36> | 212 | 0 | |
| U20 | p_ad<35> | 213 | 0 | |
| U22 | p_ad<34> | 214 | 0 | |
| U21 | Vss | 215 | — | Group Disable 0 |
| T23 | p_ad<33> | 216 | 0 | |
| T22 | p_ad<32> | 217 | 0 | |
| T21 | p_par64 | 218 | 0 | |
| R22 | config66 | 219 | — | |
| R21 | msk_in | 220 | — | |
| P23 | tdi | — | — | |
| P22 | tdo | — | — | |
| N21 | s_par64 | 0 | 8 | |
| M21 | s_ad<32> | 1 | 8 | |
| M23 | s_ad<33> | 2 | 8 | |
| M22 | s_ad<34> | 3 | 8 | |
| L22 | s_ad<35> | 4 | 8 | |
| L21 | s_ad<36> | 5 | 8 | |
| L23 | s_ad<37> | 6 | 8 | |
| K21 | s_ad<38> | 7 | 8 | |
| K22 | s_ad<39> | 8 | 8 | |
| K23 | s_ad<40> | 9 | 8 | |
| J22 | s_ad<41> | 10 | 8 | |
| J20 | s_ad<42> | 11 | 8 | |
| J23 | s_ad<43> | 12 | 8 | |
| H21 | s_ad<44> | 13 | 8 | |
| H22 | s_ad<45> | 14 | 8 | |
| H23 | s_ad<46> | 15 | 8 | |
| G21 | s_ad<47> | 16 | 8 | |
| G22 | s_ad<48> | 17 | 8 | |
| G20 | s_ad<49> | 18 | 8 | |
| F22 | s_ad<50> | 19 | 8 | |
| F23 | s_ad<51> | 20 | 8 | |
| F21 | s_ad<52> | 21 | 8 | |
| E23 | s_ad<53> | 22 | 8 | |
| E21 | s_ad<54> | 23 | 8 | |
| D22 | s_ad<55> | 24 | 8 | |

**Table 39. Boundary Scan Order (Sheet 6 of 7)**

| Pin Number | Signal Name | Boundary-Scan Order | By Group Disable | Group Disable Cell |
|---|---|---|---|---|
| E20 | s_ad<56> | 25 | 8 | |
| D21 | s_ad<57> | 26 | 8 | |
| C22 | s_ad<58> | 27 | 8 | |
| C23 | s_ad<59> | 28 | 8 | |
| C21 | s_ad<60> | 29 | 8 | |
| D20 | s_ad<61> | 30 | 8 | |
| A21 | s_ad<62> | 31 | 8 | |
| C20 | s_ad<63> | 32 | 8 | |
| D19 | s_cbe_l<4> | 33 | 8 | |
| A20 | s_cbe_l<5> | 34 | 8 | |
| C19 | s_cbe_l<6> | 35 | 8 | |
| A19 | s_cbe_l<7> | 36 | 8 | |
| B19 | s_req64_l | 37 | 8 | |
| C18 | s_ack64_l | 38 | 8 | |
| A18 | s_ad<0> | 39 | 8 | |
| B18 | s_ad<1> | 40 | 8 | |
| A17 | s_ad<2> | 41 | 8 | |
| D17 | s_ad<3> | 42 | 8 | |
| B17 | s_ad<4> | 43 | 8 | |
| C17 | s_ad<5> | 44 | 8 | |
| B16 | s_ad<6> | 45 | 8 | |
| C16 | s_ad<7> | 46 | 8 | |
| A15 | s_cbe_l<0> | 47 | 8 | |
| B15 | s_ad<8> | 48 | 8 | |
| C15 | s_ad<9> | 49 | 8 | |
| A14 | s_m66ena | 50 | 8 | |
| B14 | s_ad<10> | 51 | 8 | |
| C14 | s_ad<11> | 52 | 8 | |
| D13 | s_ad<12> | 53 | 8 | |
| A13 | s_ad<13> | 54 | 8 | |
| B13 | s_ad<14> | 55 | 8 | |
| C13 | s_ad<15> | 56 | 8 | |
| C12 | s_cbe_l<1> | 57 | 8 | |
| A12 | Vss | 58 | — | Group Disable 8 |
| B12 | s_par | 59 | 8 | |
| B11 | s_serr_l | 60 | — | |
| C11 | s_perr_l | 61 | 7 | |

**Table 39. Boundary Scan Order (Sheet 7 of 7)**

| Pin Number | Signal Name | Boundary-Scan Order | By Group Disable | Group Disable Cell |
|---|---|---|---|---|
| A11 | s_lock_l | 62 | 7 | |
| C10 | s_stop_l | 63 | 7 | |
| B10 | s_devsel_l | 64 | 7 | |
| A10 | s_trdy_l | 65 | 7 | |
| C9 | s_irdy_l | 66 | 7 | |
| — | Vss | 67 | — | Group Disable 7 |
| B9 | s_frame_l | 68 | 7 | |
| D9 | s_cbe_l<2> | 69 | 6 | |
| A9 | s_ad<16> | 70 | 6 | |
| C8 | s_ad<17> | 71 | 6 | |
| B8 | s_ad<18> | 72 | 6 | |
| A8 | s_ad<19> | 73 | 6 | |
| B7 | s_ad<20> | 74 | 6 | |
| D7 | s_ad<21> | 75 | 6 | |
| A7 | s_ad<22> | 76 | 6 | |
| A6 | s_ad<23> | 77 | 6 | |
| C6 | s_cbe_l<3> | 78 | 6 | |
| B5 | s_ad<24> | 79 | 6 | |
| C5 | s_ad<25> | 80 | 6 | |
| B4 | s_ad<26> | 81 | 6 | |
| A4 | s_ad<27> | 82 | 6 | |
| C4 | s_ad<28> | 83 | 6 | |
| B3 | s_ad<29> | 84 | 6 | |
| A3 | s_ad<30> | 85 | 6 | |
| — | Vss | 86 | — | Group Disable 6 |
| C3 | s_ad<31> | 87 | 6 | |

## 16.7 Initialization

The test access port controller and the instruction register output latches are initialized when the trst_l input is asserted. The test access port controller enters the test-logic reset state. The instruction register is reset to hold the bypass register instruction. During test-logic reset state, all JTAG test logic is disabled, and the chip performs normal functions. The test access port controller leaves this state only when an appropriate JTAG test operation sequence is sent on the tms and tck pins.

# 17.0 Electrical Specifications

This chapter specifies the following electrical behavior of the 21154:

- PCI electrical conformance
- Absolute maximum ratings
- dc specifications
- ac timing specifications

## 17.1 PCI Electrical Specification Conformance

The 21154 PCI pins conform to the basic set of PCI electrical specifications in the *PCI Local Bus Specification, Revision 2.1*. See that document for a complete description of the PCI I/O protocol and pin ac specifications.

## 17.2 Absolute Maximum Ratings

The 21154 is specified to operate at a maximum frequency of 33 MHz, or 66 MHz if 66 MHz capable, at a junction temperature ($T_j$) not to exceed 125C. Table 40 lists the absolute maximum ratings for the 21154. These are stress ratings only; stressing the device beyond the absolute maximum ratings may cause permanent damage. Operating beyond the functional operating range (see Table 41) is not recommended and extended exposure beyond the functional operating range may affect reliability.

**Table 40. Absolute Maximum Ratings**

| Parameter | Minimum | Maximum |
|---|---|---|
| Junction temperature,$T_j$ | — | 125°C |
| Maximum voltage applied to signal pins | — | 5.5 V |
| Supply voltage, $V_{cc}$ | — | 3.9 V |
| Maximum Power, $P_{WC}$ | — | 2.2 W |
| Storage temperature range, $T_{sg}$ | –55°C | 125°C |

**Table 41. Functional Operating Range**

| Parameter | Minimum | Maximum |
|---|---|---|
| Supply voltage, $V_{cc}$ | 3.0 V | 3.6 V |
| Operating ambient temperature, $T_a$ | 0°C | 70°C |

## 17.3 DC Specifications

Table 42 defines the dc parameters met by all 21154 signals under normal operating conditions.

**Table 42. DC Parameters**

| Symbol | Parameter | Condition | Minimum | Maximum | Unit |
|---|---|---|---|---|---|
| $V_{cc}$ | Supply voltage | — | 3.0 | 3.6 | V |
| $V_{il}$ | Low-level input voltage[1] | — | –0.5 | 0.3 $V_{cc}$ | V |
| $V_{ih}$ | High-level input voltage[1] | — | 0.5 $V_{cc}$ | $V_{cc}$ + 0.5 V | V |
| $V_{ol}$ | Low-level output voltage[2] | $I_{out}$ = 1500 µA | — | 0.1 $V_{cc}$ | V |
| $V_{ol5V}$ | Low-level output voltage[3] | $I_{out}$ = 6 mA | — | 0.55 | V |
| $V_{oh}$ | High-level output voltage[2] | $I_{out}$ = –500 µA | 0.9 $V_{cc}$ | — | V |
| $V_{oh5V}$ | High-level output voltage[3] | $I_{out}$ = –2 mA | 2.4 | — | V |
| $I_{il}$ | Low-level input leakage current[1,4] | 0 < $V_{in}$ < $V_{cc}$ | — | ±10 | µA |
| $C_{in}$ | Input pin capacitance | — | — | 10.0 | pF |
| $C_{IDSEL}$ | p_idsel pin capacitance | — | — | 8.0 | pF |
| $C_{clk}$ | p_clk, s_clk pin capacitance | — | 5.0 | 12.0 | pF |

[1] Guarantees meeting the specification for the 5-V signaling environment.
[2] For 3.3-V signaling environment.
[3] For 5-V signaling environment.
[4] Input leakage currents include high-Z output leakage for all bidirectional buffers with tristate outputs.

***Note:*** In Table 42, currents into the chip (chip sinking) are denoted as positive (+) current. Currents from the chip (chip sourcing) are denoted as negative (–) current.

## 17.4 AC Timing Specifications

The next sections specify the following:

- Clock timing specifications
- PCI signal timing specifications
- Reset timing specifications
- gpio timing specifications
- JTAG timing specifications

### 17.4.1 Clock Timing Specifications

The ac specifications consist of input requirements and output responses. The input requirements consist of setup and hold times, pulse widths, and high and low times. Output responses are delays from clock to signal. The ac specifications are defined separately for each clock domain within the 21154.

Figure 23 shows the ac parameter measurements for the p_clk and s_clk signals, and Table 43 and Table 44 specify p_clk and s_clk parameter values for clock signal ac timing. See also Figure 24 for a further illustration of signal timing. Unless otherwise noted, all ac parameters are guaranteed when tested within the functional operating range of Table 41.

**Figure 23. PCI Clock Signal AC Parameter Measurements**

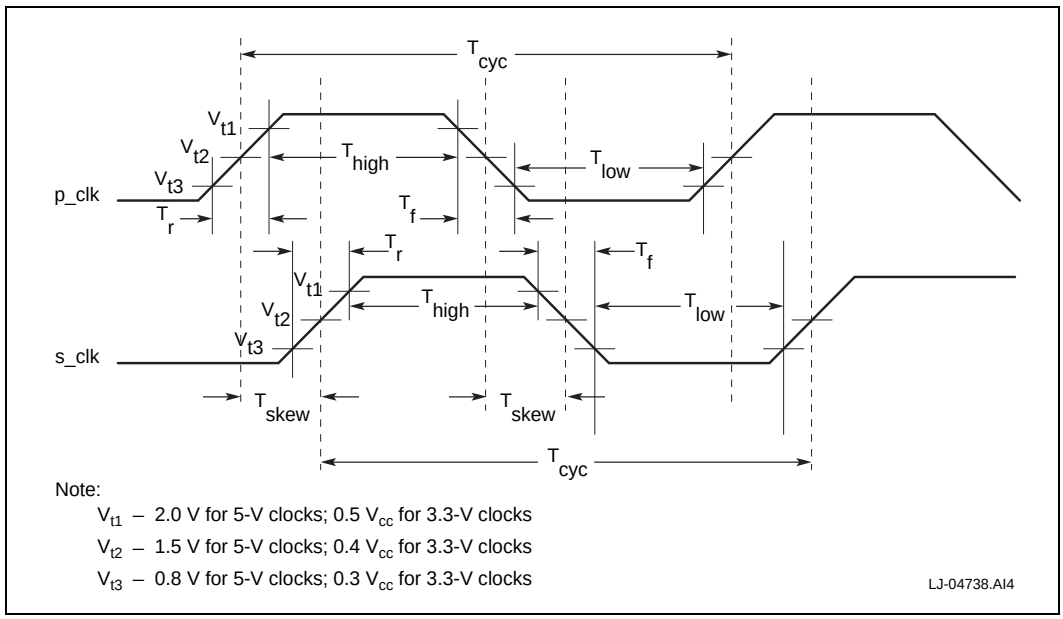



**Table 43. 33 MHz PCI Clock Signal AC Parameters**

| Symbol | Parameter | Minimum | Maximum | Unit |
|---|---|---|---|---|
| $T_{cyc}$ | p_clk,s_clk cycle time | 30 | ∞ | ns |
| $T_{high}$ | p_clk, s_clk high time | 11 | — | ns |
| $T_{low}$ | p_clk, s_clk low time | 11 | — | ns |
| | p_clk, s_clk slew rate[1] | 1 | 4 | V/ns |
| $T_{sclk}$ | Delay from p_clk to s_clk | 0 | 7 | ns |
| $T_{sclkr}$ | p_clk rising to s_clk_o rising | 0 | 5 | ns |
| $T_{sclkf}$ | p_clk falling to s_clk_o falling[2] | 0 | 5 | ns |
| $T_{dskew}$ | s_clk_0 duty cycle skew from p_clk duty cycle[2] | — | 0.750 | ns |
| $T_{skew}$ | s_clk_0<x> to s_clk_0<y> | — | 0.500 | ns |

[1] 0.2 $V_{cc}$ to 0.6 $V_{cc}$.
[2] Measured with 30-pF lumped load.

**Table 44. 66 MHz PCI Clock Signal AC Parameters**

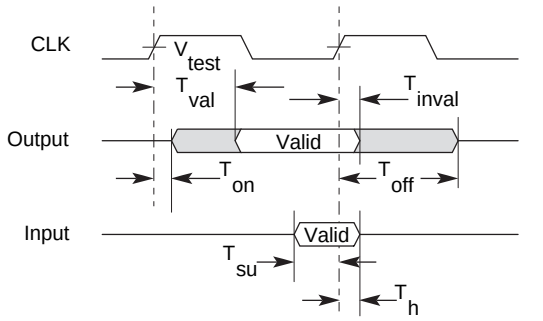
| Symbol | Parameter | Minimum | Maximum | Unit |
|---|---|---|---|---|
| $T_{cyc}$ | p_clk,s_clk cycle time | 15 | 30 | ns |
| $T_{high}$ | p_clk, s_clk high time | 6 | — | ns |
| $T_{low}$ | p_clk, s_clk low time | 6 | — | ns |
| | p_clk, s_clk slew rate[1] | 1.5 | 4 | V/ns |
| $T_{sclk}$ | Delay from p_clk to s_clk | 0 | tbd[2] | ns |
| $T_{sclkr}$ | p_clk rising to s_clk_o rising | 0 | 5 | ns |
| $T_{sclkf}$ | p_clk falling to s_clk_o falling[3] | 0 | 5 | ns |
| $T_{dskew}$ | s_clk_0 duty cycle skew from p_clk duty cycle | — | 0.750 | ns |
| $T_{skew}$ | s_clk_0<x> to s_clk_0<y> | — | 0.500 | ns |

1. 0.2 $V_{cc}$ to 0.6 $V_{cc}$.
2. To be determined.
3. Measured with 30-pF lumped load.

## 17.4.2 PCI Signal Timing Specifications

Figure 24 and Table 45 show the PCI signal timing specifications.

**Figure 24. PCI Signal Timing Measurement Conditions**




Note:
$V_{test}$ – 1.5 V for 5-V signals; 0.4 $V_{cc}$ for 3.3-V signals

LJ-04739.AI4

**Table 45. 33 MHz PCI Signal Timing (Sheet 1 of 2)**

| Symbol | Parameter | Minimum | Maximum | Unit |
|---|---|---|---|---|
| $T_{val}$ | CLK to signal valid delay — bused signals[1,2,3] | 2 | 11 | ns |
| $T_{val(ptp)}$ | CLK to signal valid delay — point-to-point[1,2,3] | 2 | 12 | ns |
| $T_{on}$ | Float to active delay[1,2] | 2 | — | ns |
| $T_{off}$ | Active to float delay[1,2] | — | 28 | ns |

**Table 45. 33 MHz PCI Signal Timing (Sheet 2 of 2)**

| Symbol | Parameter | Minimum | Maximum | Unit |
|---|---|---|---|---|
| $T_{su}$ | Input setup time to CLK — bused signals[1,2,3] | 7 | — | ns |
| $T_{su(ptp)}$ | Input setup time to CLK—point-to-point[1,2,3] | 10, 12 | — | ns |
| $T_h$ | Input signal hold time from CLK[1,2] | 0 | — | ns |

1 See Figure 24.
2 All primary interface signals are synchronized to p_clk. All secondary interface signals are synchronized to s_clk.
3 Point-to-point signals are p_req_l, s_req_l<8:0>, p_gnt_l, and s_gnt_l<8:0>. Bused signals are p_ad, p_cbe_l, p_par, p_perr_l, p_serr_l, p_frame_l, p_irdy_l, p_trdy_l, p_lock_l, p_devsel_l, p_stop_l, p_idsel, p_req64_l, p_ack64_l, p_par64, s_ad, s_cbe_l, s_par, s_perr_l, s_serr_l, s_frame_l, s_irdy_l, s_trdy_l, s_lock_l, s_devsel_l, s_stop_l, s_req64_l, s_ack64_l, and s_par64.

**Table 46. 66 MHz PCI Signal Timing**

| Symbol | Parameter | Minimum | Maximum | Unit |
|---|---|---|---|---|
| $T_{val}$ | CLK to signal valid delay — bused signals[1,2,3] | 2 | 6 | ns |
| $T_{val(ptp)}$ | CLK to signal valid delay — point-to-point[1,2,3] | 2 | 6 | ns |
| $T_{on}$ | Float to active delay[1,2] | 2 | — | ns |
| $T_{off}$ | Active to float delay[1,2] | — | 14 | ns |
| $T_{su}$ | Input setup time to CLK — bused signals[1,2,3] | 3 | — | ns |
| $T_{su(ptp)}$ | Input setup time to CLK—point-to-point[1,2,3] | 5 | — | ns |
| $T_h$ | Input signal hold time from CLK[1,2] | 0 | — | ns |

1 See Figure 24.
2 All primary interface signals are synchronized to p_clk. All secondary interface signals are synchronized to s_clk.
3 Point-to-point signals are p_req_l, s_req_l<8:0>, p_gnt_l, and s_gnt_l<8:0>. Bused signals are p_ad, p_cbe_l, p_par, p_perr_l, p_serr_l, p_frame_l, p_irdy_l, p_trdy_l, p_lock_l, p_devsel_l, p_stop_l, p_idsel, p_req64_l, p_ack64_l, p_par64, s_ad, s_cbe_l, s_par, s_perr_l, s_serr_l, s_frame_l, s_irdy_l, s_trdy_l, s_lock_l, s_devsel_l, s_stop_l, s_req64_l, s_ack64_l, and s_par64.

### 17.4.3 Reset Timing Specifications

Table 47 shows the reset timing specifications for p_rst_l and s_rst_l.

**Table 47. Reset Timing Specifications**

| Symbol | Parameter | Minimum | Maximum | Unit |
|---|---|---|---|---|
| $T_{rst}$ | p_rst_l active time after power stable | 1 | — | µs |
| $T_{rst—clk}$ | p_rst_l active time after p_clk stable | 100 | — | µs |
| $T_{rst—off}$ | p_rst_l active-to-output float delay | — | 40 | ns |
| $T_{srst}$ | s_rst_l active after p_rst_l assertion | — | 40 | ns |
| $T_{srst—on}$ | s_rst_l active time after s_clk stable | 100 | — | µs |
| $T_{dsrst}$ | s_rst_l deassertion after p_rst_l deassertion | 20 | 25 | Cycles |
| | p_rst_l slew rate[1] | 50 | — | mV/ns |
| $T_{rrsu}$ | p_req64_l to p_rst_l setup time | $10*T_{cyc}$ | — | ns |
| $T_{rrh}$ | p_rst_l to p_req64_l hold time | 0 | 50 | ns |
| $T_{rval}$ | s_rst_l rising to s_req64_l rising | 2 | 52 | ns |

1. Applies to rising (deasserting) edge only.

### 17.4.4 gpio Timing Specifications

Table 48 and Table 49 show the gpio timing specifications. See also Figure 24.

**Table 48. 33 MHz gpio Timing Specifications**

| Symbol | Parameter | Maximum | Minimum | Unit |
|---|---|---|---|---|
| $T_{vgpio}$ | s_clk to gpio output valid | 2 | 12 | ns |
| $T_{gon}$ | gpio float-to-active delay | 2 | — | ns |
| $T_{goff}$ | gpio active to float delay | — | 28 | ns |
| $T_{gsu}$ | gpio-to-s_clk setup time | 7 | — | ns |
| $T_{gh}$ | gpio hold time after s_clk | 0 | — | ns |
| $T_{gcval}$ | s_clk-to-gpio<0> shift clock output valid | — | 13.5 | ns |
| $T_{gcyc}$ | gpio<0> cycle time | 30 | ∞ | ns |
| $T_{gsval}$ | gpio<0>-to-gpio<2> shift control output valid | — | 8 | ns |
| $T_{msu}$ | msk_in setup time to gpio<0> | 15 | — | ns |
| $T_{mh}$ | msk_in hold time after gpio<0> | 0 | — | ns |

**Table 49. 66 MHz gpio Timing Specifications**

| Symbol | Parameter | Maximum | Minimum | Unit |
|---|---|---|---|---|
| $T_{vgpio}$ | s_clk to gpio output valid | 2 | 12 | ns |
| $T_{gon}$ | gpio float-to-active delay | 2 | — | ns |
| $T_{goff}$ | gpio active to float delay | — | 28 | ns |
| $T_{gsu}$ | gpio-to-s_clk setup time | 7 | — | ns |
| $T_{gh}$ | gpio hold time after s_clk | 0 | — | ns |
| $T_{gcval}$ | s_clk-to-gpio<0> shift clock output valid | — | 13.5 | ns |
| $T_{gcyc}$ | gpio<0> cycle time | 30 | ∞ | ns |
| $T_{gsval}$ | gpio<0>-to-gpio<2> shift control output valid | — | 8 | ns |
| $T_{msu}$ | msk_in setup time to gpio<0> | 15 | — | ns |
| $T_{mh}$ | msk_in hold time after gpio<0> | 0 | — | ns |

## 17.4.5 JTAG Timing Specifications

Table 50 shows the JTAG timing specifications.

**Table 50. JTAG Timing Specifications**

| Symbol | Parameter | Maximum | Minimum | Unit |
|---|---|---|---|---|
| $T_{if}$ | tck frequency | 0 | 10 | MHz |
| $T_{jp}$ | tck period | 100 | ∞ | ns |
| $T_{jht}$ | tck high time | 45 | — | ns |
| $T_{jlt}$ | tck low time | 45 | — | ns |
| $T_{jrt}$ | tck rise time[1] | — | 10 | ns |
| $T_{jft}$ | tck fall time[2] | — | 10 | ns |
| $T_{je}$ | tdi, tms setup time to tck rising edge | 10 | — | ns |
| $T_{jh}$ | tdi, tms hold time from tck rising edge | 25 | — | ns |
| $T_{jd}$ | tdo valid delay from tck falling edge[3] | — | 30 | ns |
| $T_{jfd}$ | tdo float delay from tck falling edge | — | 30 | ns |

1. Measured between 0.8 V and 2.0 V.
2. Measured between 2.0 V and 0.8 V.
3. $C_1$ = 50 pF.

# 18.0 Mechanical Specifications

The 21154 is contained in an industry-standard 304-point 2-layer plastic ball grid array (PBGA) package, shown in Figure 25.

**Figure 25. 304-Point 2-Layer PBGA Package**

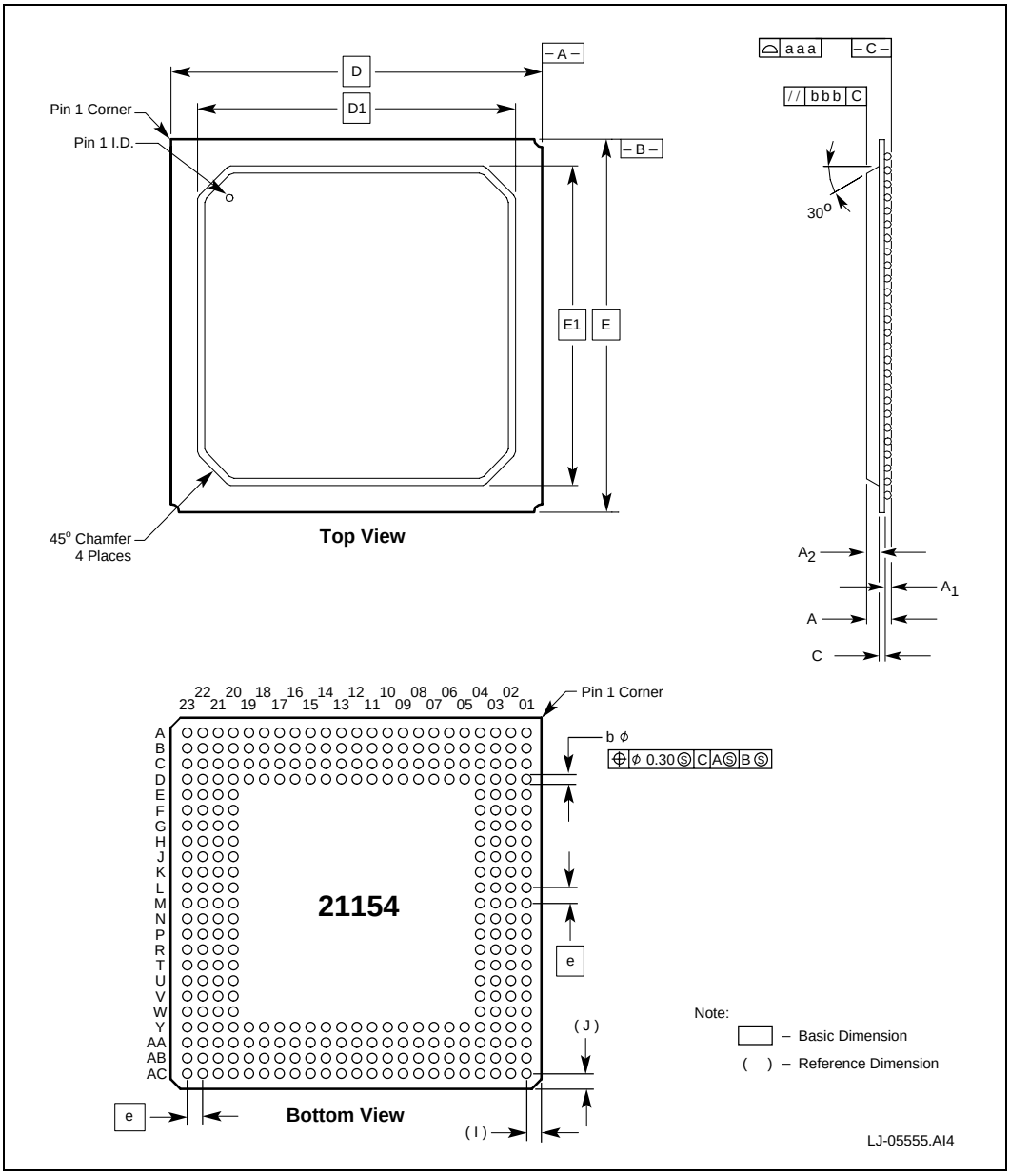


Table 51 lists the package dimensions in millimeters.

**Table 51. 304-Point 2-Layer PBGA Package Dimensions**

| Symbol | Dimension | Minimum Value | Nominal Value | Maximum Value |
|---|---|---|---|---|
| e | Ball Pitch | — | 1.27 BSC[1] | — |
| A | Overall package height | 2.12 | 2.33 | 2.54 |
| $A_1$ | Package standoff height | 0.50 | 0.60 | 0.70 |
| $A_2$ | Encapsulation thickness | 1.12 | 1.17 | 1.22 |
| b | Ball diameter | 0.60 | 0.76 | 0.90 |
| c | Substrate thickness | 0.50 | 0.56 | 0.62 |
| aaa | Coplanarity | — | — | 0.15 |
| bbb | Overall package planarity | — | — | 0.15 |
| D | Overall package width | 30.80 | 31.00 | 31.20 |
| $D_1$ | Overall encapsulation width | — | 26.00 | 26.70 |
| E | Overall package width | 30.80 | 31.00 | 31.20 |
| $E_1$ | Overall encapsulation width | — | 26.00 | 26.70 |
| I | Location of first row (*x*-direction) | — | 1.53 reference[2] | — |
| J | Location of first row (*y*-direction) | — | 1.53 reference[2] | — |

1. ANSI Y14.5M-1982 American National Standard Dimensioning and Tolerancing, Section 1.3.2, defines Basic Dimension (BSC) as: A numerical value used to describe the theoretically exact size, profile, orientation, or location of a feature or datum target. It is the basis from which permissible variations are established by tolerances on other dimensions, in notes, or in feature control frames.
2. The value for this measurement is for reference only.

# Support, Products, and Documentation

If you need technical support, a *Product Catalog*, or help deciding which documentation best meets your needs, visit the Intel World Wide Web Internet site:
`http://www.intel.com`

Copies of documents that have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling **1-800-332-2717** or by visiting Intel's website for developers at:
`http://developer.intel.com`

You can also contact the Intel Massachusetts Information Line or the Intel Massachusetts Customer Technology Center. Please use the following information lines for support:

| **For documentation and general information:** | |
|---|---|
| Intel Massachusetts Information Line | |
| United States: | 1–800–332–2717 |
| Outside United States: | 1–303-675-2148 |
| Electronic mail address: | techdoc@intel.com |

| **For technical support:** | |
|---|---|
| **Intel Massachusetts Customer Technology Center** | |
| Phone (U.S. and international): | 1–978–568–7474 |
| Fax: | 1–978–568–6698 |
| Electronic mail address: | techsup@intel.com |