PCI IP Core Specification

Authors: Miha Dolenc & Tadej Markovic

mihad@opencores.org

tadej@opencores.org

Rev. 1.2

January 24, 2004

This page has been intentionally left blank.

Revision History

	Date	Author	Description
0.0	5/1/01	Miha Dolenc Tadej Markovic	First Draft
0.1	5/8/01	Miha Dolenc Tadej Markovic	Waveforms added for WISHBONE slave
0.2	5/15/01	Miha Dolenc Tadej Markovic	Detailed description of FIFO added, Operation of PCI target unit added, Waveforms added for PCI target
0.3	5/22/01	Miha Dolenc Tadej Markovic	FIFO structure changed
0.4	10/13/01	Jeanne Wiegelmann	First review
0.5	10/20/01	Miha Dolenc Tadej Markovic	Updated register descriptions and Configuration Space access
0.6	01/28/02	Miha Dolenc Tadej Markovic	Updated descriptions and added Software obligations
0.8	12/09/03	Miha Dolenc	Added new defines descriptions. Added WISHBONE Slave B3 description.
1.0	12/10/2003	Miha Dolenc	Moved and updated Software Obligations to Initialization and reconfiguration chapter.
1.1	12/11/2003	Miha Dolenc	Added CompactPCI Hot Swap support descriptions.
1.2	01/09/2004	Miha Dolenc	Added Serial Power On Configuration interface description.

List of Contents

List of Tables

List of Figures & Examples

1. Introduction

1.1. What is a PCI Bridge?

PCI bridges are used in applications and devices that want to utilize resources provided on a PCI local bus. Systems that have multiple buses must – to enable communication between them – provide an interface that connects the internal buses to the PCI local bus. PCI bridges provide such an interface.

1.2. PCI IP Core Introduction

The PCI IP core (PCI bridge) provides an interface between the WISHBONE SoC bus and the PCI local bus. It consists of two independent units, one handling transactions originating on the PCI bus, the other one handling transactions originating on the WISHBONE bus.

The core has been designed to offer as much flexibility as possible to all kinds of applications.

1.3. PCI IP Core Features

The following lists the main features of the PCI IP core:

- 32-bit PCI interface
- Fully PCI 2.2 compliant (with 66 MHz PCI specification)
- Separated initiator and target functional blocks
- Supported initiator commands and functions:
  - ✓ Memory Read, Memory Write

- ✓ Memory Read Multiple (MRM)
- ✓ Memory Read Line (MRL)
- ✓ I/O Read, I/O Write
- ✓ Configuration Read, Configuration Write
- ✓ Bus Parking
- ✓ Interrupt Acknowledge
- ✓ Host Bridging

- Supported target commands and functions:
  - ✓ Type 0 Configuration Space Header

    (Type 0 is used to configure agents on the same bus segment)

    (Type 1 is used to configure across PCI-to-PCI bridges) Parity Generation (PAR), Parity Error Detection (PERR# and SERR#)
  - ✓ Memory Read, Memory Write
  - ✓ Memory Read Multiple (MRM)
  - ✓ Memory Read Line (MRL)
  - ✓ Memory Write and Invalidate (MWI)
  - ✓ I/O Read, I/O Write
  - ✓ Configuration Read, Configuration Write
  - ✓ Target Abort, Target Retry, Target Disconnect
  - ✓ Fast Back-to-Back Capable Target response
- Full Command/Status registers
- WISHBONE SoC Interconnection Rev. B compliant interface on processor side (master with Target PCI and slave with Initiator PCI interface)
- Configurable on-chip FIFOs

2. Architecture

2.1. Overview

The PCI bridge consists of two units: the PCI target unit and the WISHBONE slave unit. Each holds its own set of functions to support bridging operations from WISHBONE to PCI and from PCI to WISHBONE. The WISHBONE slave unit acts as a slave on the WISHBONE side of the bridge and initiates transactions as a master on the PCI bus. The PCI target unit acts as a target on the PCI side of the bridge and as a master on its WISHBONE side. Both units operate independently of each other. The PCI target unit implements the target interface on the PCI bus and the master interface on the WISHBONE bus, the WISHBONE slave unit implements the slave interface on the WISHBONE bus and the master interface on the PCI bus.

The PCI interface is *PCI Specification 2.2* compliant, whereas the WISHBONE is *WISHBONE SoC Interconnection architecture Specification Rev. B* compliant. The WISHBONE implementation carries out 32-bit bus operations and does not support other bus widths.

Following figure gives an overview of the PCI bridge core architecture.

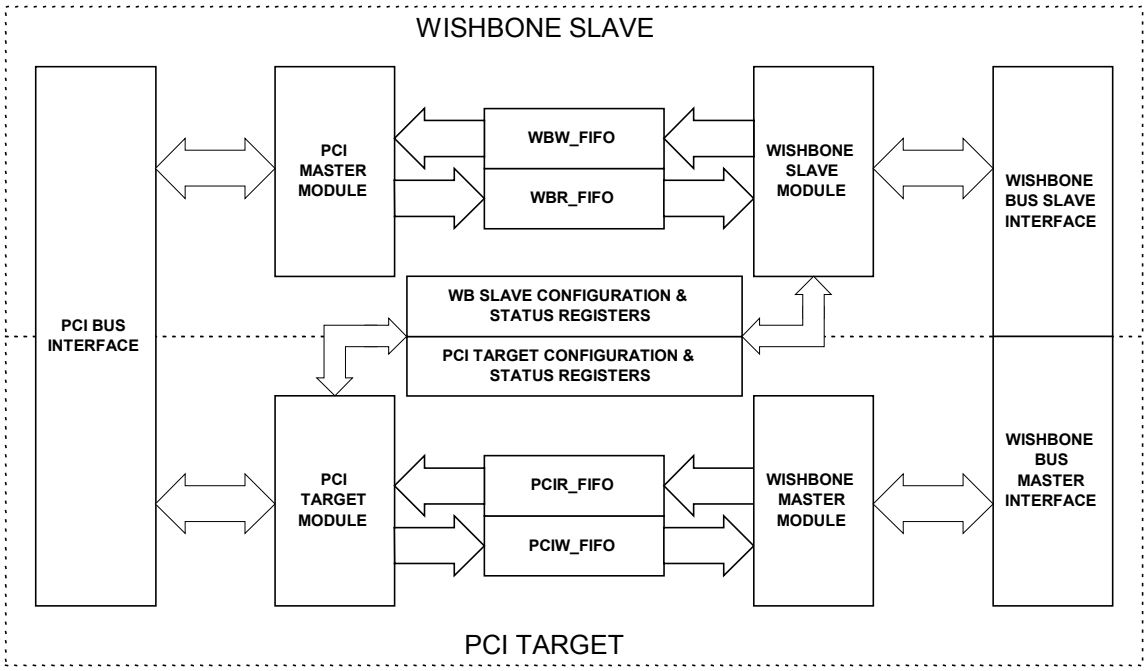


Figure 2.1: PCI bridge core architecture

2.2. WISHBONE Slave Unit

The WISHBONE bus agents can access the PCI bus through the WISHBONE slave unit. One to five configurable images can be used to access the PCI address space.

Each image consists of:

- Base address register
- Address mask register
- Translation address register
- Image control register
- Decoder

The Base address, stored in the Base Address register, is masked with a value stored in the Address Mask register. The decoder compares the masked WISHBONE bus address with the masked base address to identify valid WISHBONE cycles. If needed, the WISHBONE address can be translated to a different value before accessing the PCI bus. The value for an address to be presented on the PCI bus is stored in the Address Translation register. The Image Control register is used to control the behavior of an image.

Each image can be configured to access memory or I/O address space on the PCI bus.

Write cycles through the WB slave unit are processed as Posted Writes and Read cycles as delayed reads. Reads can also be pre-fetched if the image accessed is configured properly. The only exception

to that rule is Configuration Write, which is initiated by a special mechanism and therefore described separately in subsequent chapters.

The WISHBONE Write FIFO (WBW_FIFO) is used to post writes performed on the WISHBONE bus; the WISHBONE Read FIFO (WBR_FIFO) accumulates pre-fetched reads. The WISHBONE slave unit connects to WISHBONE masters by acting as a slave.

This section describes the architecture of a WISHBONE slave unit and is divided into subsections.

2.2.1. WISHBONE Slave Unit Architecture

The WISHBONE slave unit consists of a few functional parts allowing the WISHBONE master to perform Read/Write access to the PCI bus. The following sections provide detailed descriptions.

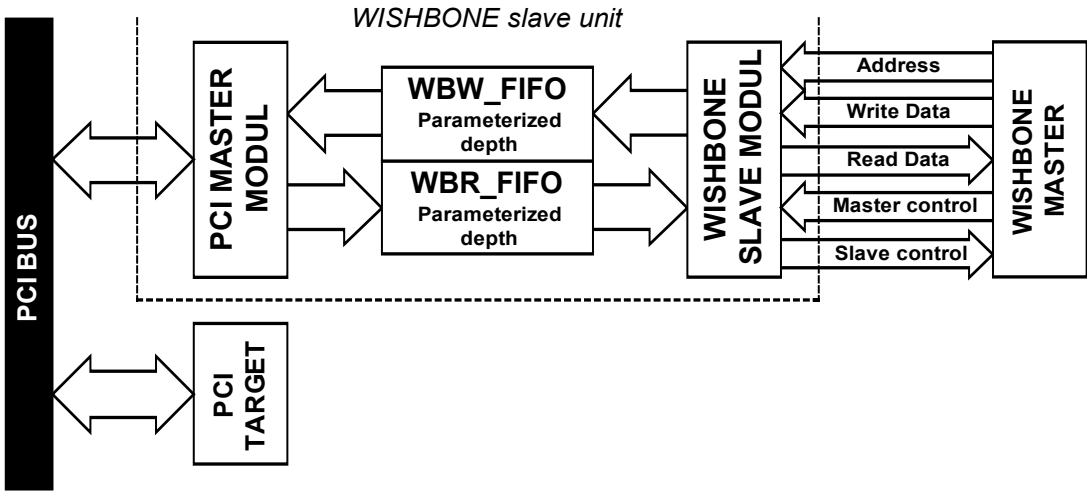


Figure 2.2: WISHBONE slave unit architecture

2.2.1.1 WISHBONE Slave Module

The WISHBONE slave module, which includes one to six image units, is a 32-bit WISHBONE slave interface as defined in *WISHBONE Specification Rev. 1B*. It handles Read/Write cycles to images of PCI address space and configuration space accesses.

2.2.1.2 WBW_FIFO

The WISHBONE slave module uses WBW_FIFO (WISHBONE Write FIFO) for posting memory and I/O Write cycles performed by the WISHBONE master. Parameterized depth provides the option to define the WBW_FIFO with regard to application specific needs for posting more or less Write cycles.

The WISHBONE bus determines the speed of Write cycles to the WBW_FIFO, whereas the PCI bus regulates the speed of Write cycles from the WBW_FIFO.

2.2.1.3 WBR_FIFO

The WISHBONE slave module uses WBR_FIFO (WISHBONE Read FIFO) for storing data read from PCI targets.

The PCI bus determines the speed of Read cycles to the WBR_FIFO, and the WISHBONE bus regulates the speed of Read cycles from the WBR_FIFO.

2.2.1.4 PCI Master Module

The PCI master module uses information provided by the WISHBONE slave module to perform PCI bus cycles. It is a 32-bit/66MHz (33MHz in FPGA), PCI Local Bus Specification Rev. 2.2 compliant initiator interface.

2.3. PCI Target Unit

PCI agents can access the WISHBONE bus through the PCI target unit of the bridge, which provides one to six images of the WISHBONE side memory space. Each image is selected by an address provided during the address phase on the PCI bus. It is compared to the base address masked with a mask value stored in PCI Configuration registers and can be mapped into the memory or I/O space. An address can also be translated to a value stored in the Translation Address register if the image is properly configured.

Write cycles through the PCI target unit are handled as Posted Writes. Read cycles and can be pre-fetched.

The PCIW_FIFO stores Posted Write cycles; the PCIR_FIFO saves pre-fetched Read cycles.

2.3.1. PCI Target Unit Architecture

This part describes the architecture of the PCI target unit. The following sections provide detailed descriptions.

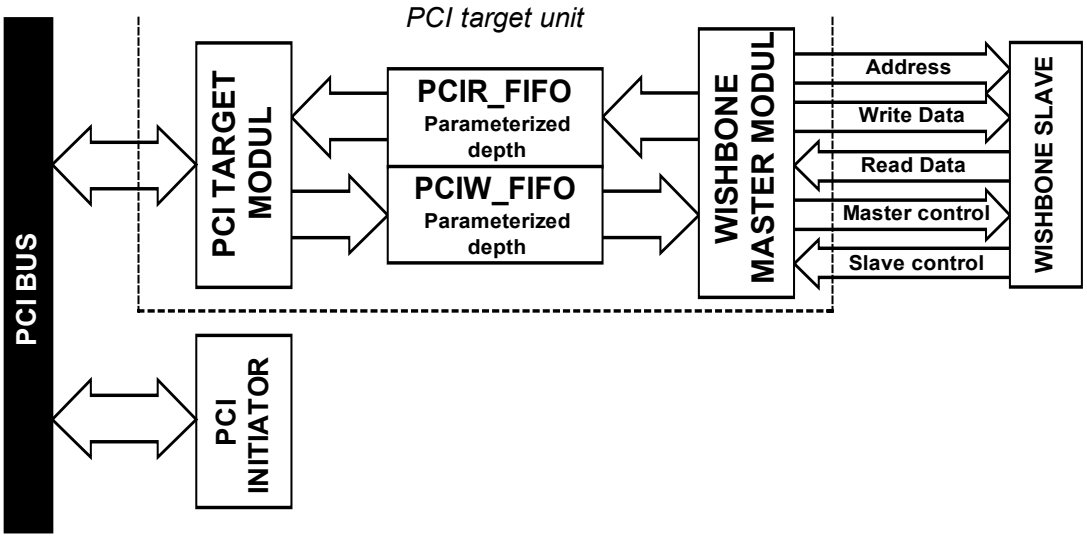


Figure 2.3: PCI target unit architecture overview

The PCI target unit consists of a few functional parts allowing PCI initiators to perform Read/Write accesses to the WISHBONE bus.

The PCI target module is a 32-bit/66MHz (33MHz in FPGA), *PCI Local Bus Specification Rev. 2.2* compliant target interface that includes two to six image units for address translation from the PCI bus. Therefore, it handles Read/Write cycles to images of WISHBONE address space and configuration space accesses.

2.3.1.1 PCI Target Module

The PCI target module uses PCIW_FIFO (PCI Write FIFO) for posting memory and I/O Write cycles performed by the PCI initiator. Parameterized depth provides the option to define the PCIW_FIFO with regard to application specific needs for posting more or less Write cycles.

The PCI bus determines the speed of Write cycles to the PCIW_FIFO, whereas the WISHBONE bus regulates the speed of Write cycles from the PCIW_FIFO.

2.3.1.2 PCIR_FIFO

The WISHBONE master module uses PCIR_FIFO (PCI Read FIFO) for storing data read from WISHBONE slaves.

The WISHBONE bus determines the speed of Read cycles to PCIR_FIFO, and the PCI bus regulates the speed of Read cycles from the PCIR_FIFO.

2.3.1.3 WISHBONE Master Module

The WISHBONE master module is a 32-bit WISHBONE master interface as defined in *WISHBONE Specification Rev. 1B*. Through its WISHBONE master module, the core sends requests to the WISHBONE bus. Chapter WISHBONE Slave Interface, provides detailed information on the WISHBONE interface of the core.

2.4. Clocks

The PCI core has two clock domains, one from the PCI bus, the other one from the WISHBONE bus. With its interconnection logic, the FIFO adjusts the different bus clocks. There is no difference between all four FIFOs, because it is not decisive which bus operates on higher frequency.

2.5. FIFO

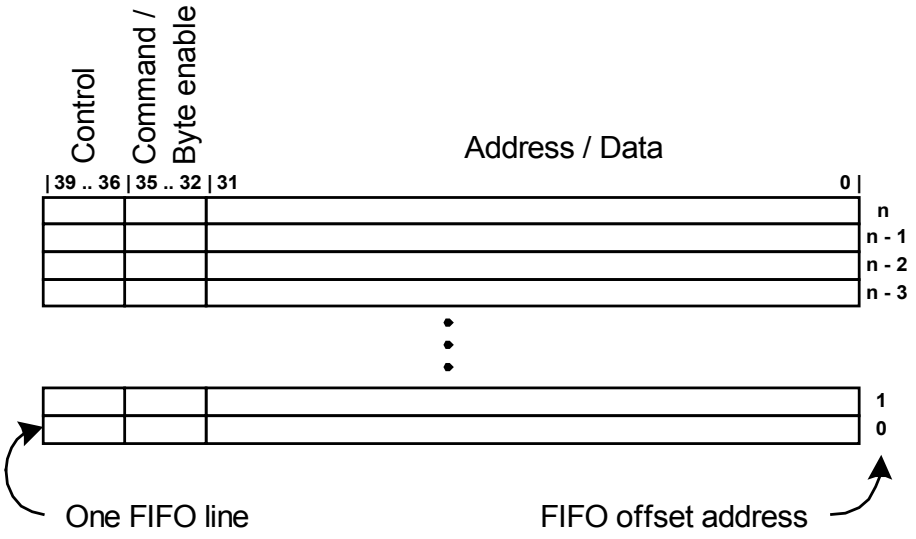


Figure 2.4: Detailed Description of FIFO Register Lines

The FIFO is structured by more than one line. The number of FIFO lines, which is configurable, determines the depth number (the *Design Document and Implementation Notes* discuss in detail how FIFO depth is defined). Figure 2.4 describes the structure of one FIFO line, which consists of 4 control bits (the *Design Document* describes in detail how they are used—e.g. one bit is used to sign the last data of the burst transfer etc.), 4 command or byte enable bits (coding will be described in detail in the *Design Document*), and 32 address or data bits.

FIFOs are implemented as circular data buffers between WISHBONE and PCI interfaces (Figure 2.5) and adapt to different bus speeds with their interconnection logic. The input bus clock, which is also connected to FIFO registers, writes data to the input side of the FIFO. The input pointer (input counter), which has the same clock frequency as the input bus side, stores the value of the input offset address of the first free FIFO line.

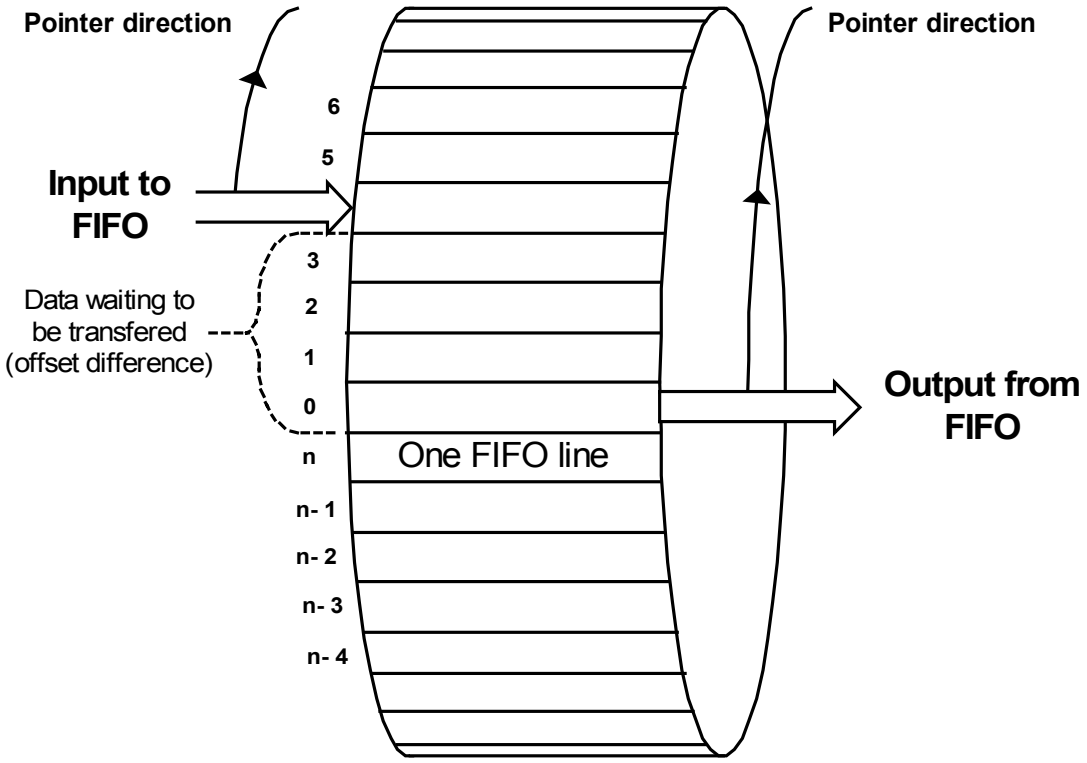


Figure 2.5: FIFO Architecture

The output pointer (output counter) stores the output offset address value of the first FIFO line from which data is to be read. It has the same clock frequency as the output bus side that reads the data.

The comparator between both pointers (counters) validates if any data is waiting in the FIFO to be read (the *Design Document* describes in detail the exact counter/comparator operation). Another comparator is between the counter, which has the value of an input pointer incremented to one, and the output pointer. When both variables are equal, the FIFO is full.

2.6. Address Translation Logic

WISHBONE slave unit and PCI target unit incorporate several address space images. If address translation is implemented, each image can have address translation enabled via the Image Control Register and Translation Address set in the Translation Address register.

2.6.1. Description of Address Translation Logic

For a description of the address translation logic, see Figure 2.6. All AND blocks and OR blocks are bit-oriented operators that stand for logic operations between bits of the same weight (e.g. logic function between bit[n-2] of bus A and bit[n-2] of bus B).

The base address is written into the Base Address register. The Address Mask register, which also defines the size of an image, decides how many most significant bits are masked and replaced by translation address bits. There is a rule how to set the Address Mask register: Address bits that can be masked must start with the MS bit (bit[31]) and continue to the twelfth bit (bit[11]). All bits allowed to be masked define the smallest size of 4KB that can be assigned. No zeros must be between mask bits; otherwise this image will have two base addresses but only one Base Address register—a situation that does not comply with the *PCI Specification*.

To find out if an address falls into the correct address range, the masked bits of input address and base address must be compared (the number of masked bits defines the unchanging address of the current address range and thereby the size of this image).

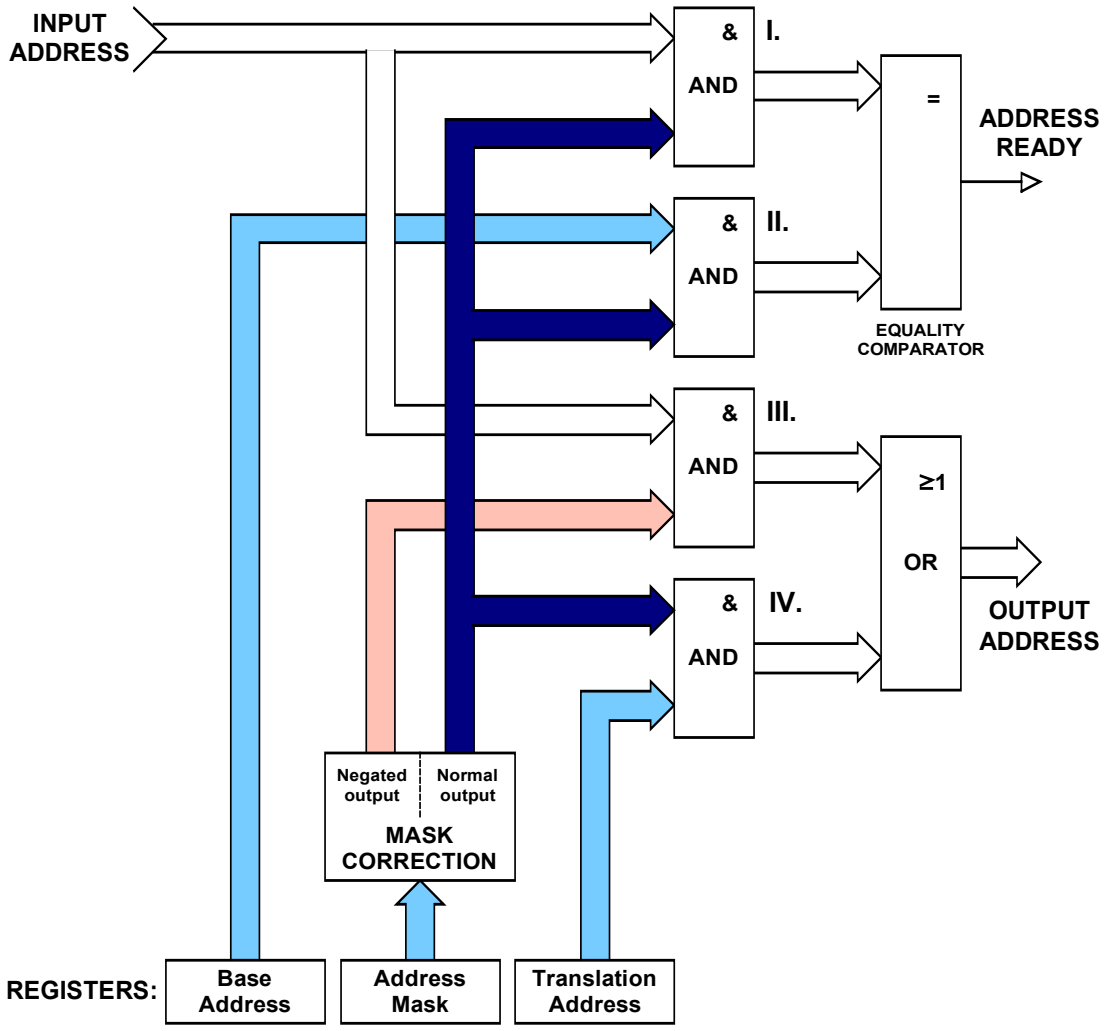


Figure 2.6: Address Translation Logic

3. Operation

3.1. Initialization

3.1.1. Initialization for GUEST implementation

The GUEST implementation of the PCI Bridge will be connected into the host system with PCI Bus support. At power up, the device independent software scans the PCI slots in the host system for any connected devices. The host system performs configuration cycles to achieve that. The PCI Bridge and all other PCI devices connected to the host system respond to configuration cycle transactions only after power up reset has been applied and released. The following steps must be performed in the correct order, to assure proper PCI Bridge operation:

1. Configure all PCI Base Address registers.
2. Enable PCI Master operation and PCI Target's response to Memory and I/O spaces.
3. Configure other PCI Configuration Space Type00 Header registers as appropriate.
4. After the first three steps are done, all configuration registers can be accessed using Memory Write and Read commands within the address range of PCI Image0. Now all other registers can be configured as appropriate for the application.
5. When all PCI Bridge registers are configured, the software can start accessing slave devices on the WISHBONE bus through the PCI Target unit and enable master devices on the WISHBONE bus to access the PCI bus through the WISHBONE Slave unit.

The device independent software running on the host system usually performs the first three steps and device dependent software (or driver) performs steps 4 and 5.

3.1.2. Initialization for HOST implementation

The HOST implementation of the PCI Bridge is intended for applications that will take full control over and responsibility for configuration of the PCI devices connected to the PCI bus. The software running on the WISHBONE side of the PCI Bridge should perform the following steps to assure proper operation:

1. Enable the PCI Bridge's PCI Master operation.
2. Scan the PCI bus for connected PCI devices using Type0 and Type1 configuration read cycles. This is described in Generating Configuration Cycles.
3. Determine all connected PCI devices' address space requirements and configure their Configuration Space Header registers.
4. Configure the PCI Bridge registers using the WISHBONE Image0, as described in Configuration Space Access for Host Bus Bridges.
5. When the PCI Bridge is configured and all devices on the PCI bus have their address ranges assigned, the software can start loading device specific device drivers and applications, which start accessing the PCI bus through the WISHBONE Slave unit.
6. The devices on the PCI bus can also access the WISHBONE bus through the PCI Target unit.

3.1.3. Changing the configuration in operational mode

For majority of applications the initial configuration of the PCI Bridge will suffice and will not be changed once the PCI Bridge is configured and operational. Some applications might need to do that however. You must not change the values in the PCI Bridge registers while the PCI Bridge is servicing accesses through the WISHBONE Slave or the PCI Target unit. The following steps should be performed in the correct order, to assure proper operation during and after re-configuration of the PCI Bridge:

1. Disable all devices and applications that are accessing WISHBONE address space through the PCI Target unit.
2. Disable all devices and applications that are accessing PCI address space through the WISHBONE Slave unit.
3. The configuration software initiates one valid write transaction through the PCI Target unit (PCI to WISHBONE Write Cycles) for GUEST implementation, or through WISHBONE Slave unit (WISHBONE to PCI Write Cycles) for HOST implementation.
4. The configuration software initiates one valid read transaction through the PCI Target unit (PCI to WISHBONE Read Cycles) for GUEST implementation, or through WISHBONE Slave unit (WISHBONE to PCI Read Cycles) for HOST implementation.
5. The configuration software is now free to reconfigure or disable the PCI Bridge.

Steps 3 and 4 in the above procedure are necessary to assure that none of four FIFO memories contains any data. These two operations force all data out from the FIFOs, due to transaction ordering requirements.

3.2. Configuration Space

Depending on core implementation, either the PCI or the WISHBONE agents have full access to configuration space. If the core is implemented as a host bus bridge, the WISHBONE slave unit has exclusive access to this space, whereas the PCI target unit has read-only access (this image can be canceled or changed to normal PCI to WB image). If the core is implemented as a guest (expansion bus bridge), exclusive access to configuration space lies with the PCI target unit and the WISHBONE slave unit has read-only access (this image can also be canceled).

Configuration space has a configurable block size and is divided into two parts—one intended for Configuration, Control, and Status registers of the WB slave unit, the other one for PCI Target Unit registers. If the core is implemented as a host bus bridge, accessing specific registers in the configuration space from the WISHBONE bus can generate PCI configuration cycles; otherwise, another agent on the PCI bus must perform these cycles. Configuration space is accessible only with Single Read and Single Write cycles (e.g. it cannot be accessed with bursts).

All registers in the configuration space of a core are 32-bits wide with 8-bit granularity. All accesses from the PCI bus are DWORD aligned (e.g. two LS bits of address must be 00). The PCI standard defines special encoding for those two bits used for PCI bus memory access. To access individual bytes, the BE# signals for PCI bus access and the SEL_I signals for WISHBONE bus access must carry an appropriate value.

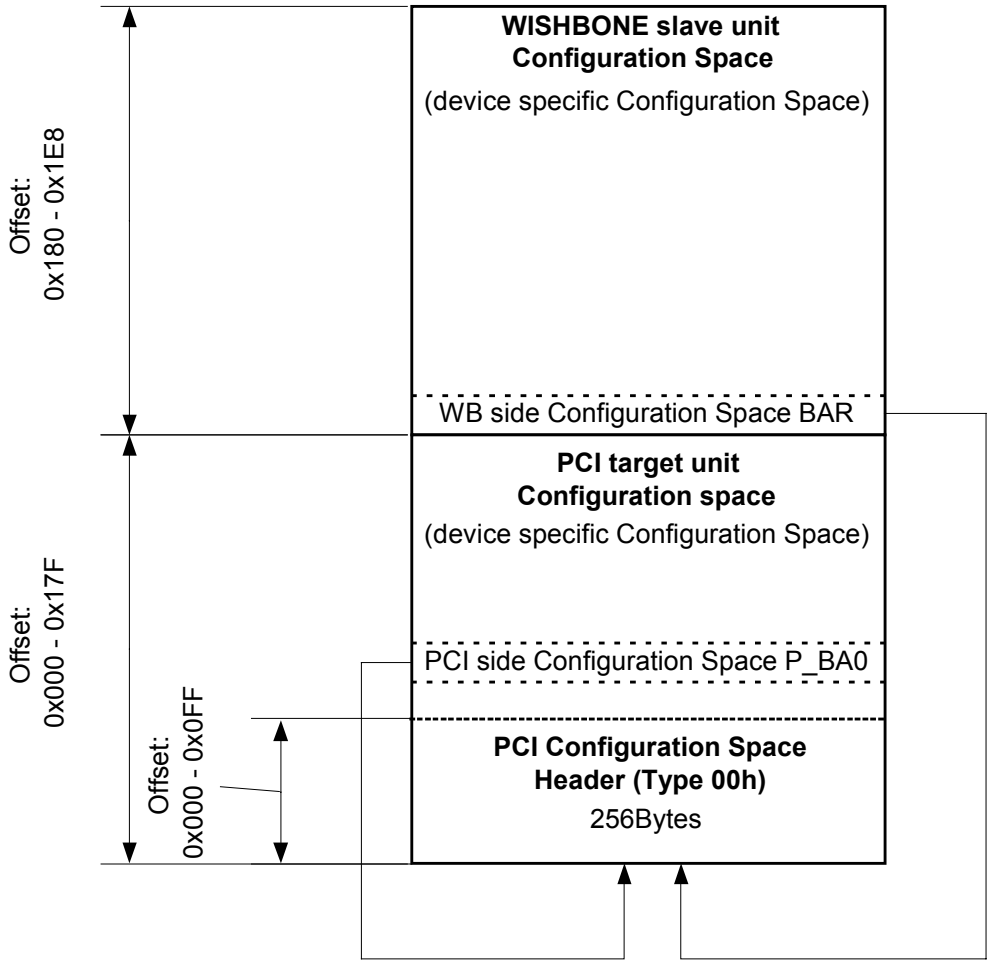


Figure 3.1: PCI Bridge Configuration Space

3.2.1. Configuration Space Access for Host Bus Bridges

The core's host bus bridge implementation provides two types of access to configuration space: Read/Write access for the WISHBONE slave unit and read-only access for the PCI target unit (unless PCI Target image 0 is canceled or used to access the WISHBONE bus—in which case external PCI devices can not read configuration space. See also Addressing and Images of the PCI Target Unit and Register List and Description). Thus, the WISHBONE master takes full responsibility for configuring core registers and any other PCI devices residing on the PCI bus. The WISHBONE side configuration space base address is predefined and cannot be changed once the core has been implemented (the *Design Document* describes in detail how and where the base address is defined.).

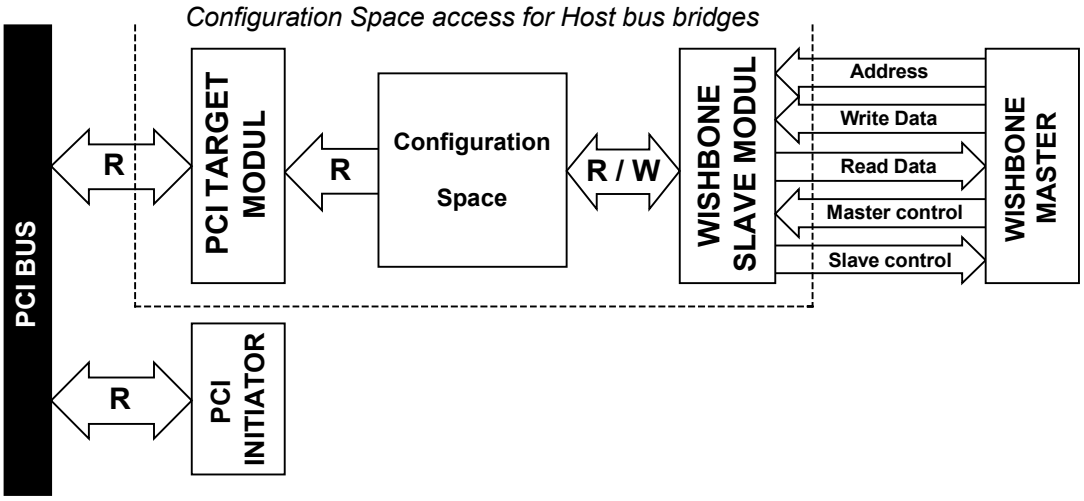


Figure 3.2: Configuration space access for Host Bus Bridges

The WISHBONE master can access configuration space either by Single Read, Single Write, or Read Modify Write (RMW) cycles. If the WISHBONE master attempts a Write cycle to non-implemented space, the cycle is acknowledged by the WISHBONE slave module while Read cycles to non-implemented space return all 0s.
The PCI side configuration space base address must be set by the WISHBONE master. The WISHBONE master must perform a Write cycle to the PCI Base Address 0 register to enable read-only access to PCI agents. The PCI target module provides read-only access to configuration space from the PCI bus using Base Address 0 range, supporting all Memory Access commands. The Memory Write commands have no effect on Configuration registers. After first data transfer completes, the PCI target module signals **Target Disconnect** to the initiator. Read cycles to non-implemented regions of configuration space return all 0s, whereas Write cycles have no effect.

3.2.2. Configuration Space Access for Guest Bridges

The implementation of the core as a guest bridge (more commonly referred to as expansion bus bridge) provides two types of configuration space access: Read/Write access for the PCI target unit and read-only access for the WISHBONE slave unit (unless WB slave image 0 is canceled). Other PCI agents take full responsibility for configuring core registers and any other PCI devices residing on the PCI bus. An agent on the PCI bus (most commonly the host bus bridge) sets the PCI Base Address 0 by performing a Type 0 configuration cycle, as stated in the *PCI Local Bus Specification Rev. 2.2*. This enables device-independent software to map the bridge's configuration space anywhere into the memory address space.

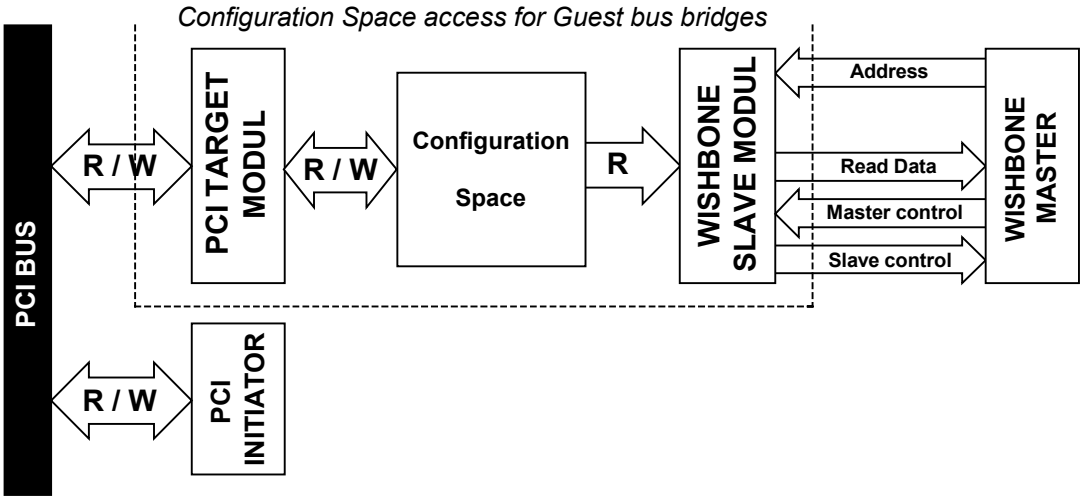


Figure 3.3: Configuration space access for Guest Bridges

The external PCI Initiator can access all registers in the configuration space using the Memory Access commands in the address range of the Bridge's PCI BAR0. It can access the first 256 bytes of the configuration space using the Configuration Access commands. The PCI Bridge terminates the first data phase of all configuration space accesses normally, and terminates subsequent data phases with **Target Disconnect without Data**. Write cycles have no effect on non-implemented configuration space, Reads from it return all 0s.

The WISHBONE slave module provides read-only access to configuration space from the WISHBONE bus. The WISHBONE side configuration space base address is predefined and cannot be changed (the *Design Document* describes in detail how and where a base address is defined). The WISHBONE slave module accepts Read or Write transfers to configuration space. Write cycles to configuration space have no effect on Configuration Space registers. When the WISHBONE master attempts to access a non-implemented region, Write cycles are acknowledged with no effect on configuration space, but Read cycles return all 0s.

3.2.3. Configuration Cycles

Configuration cycles are another way of accessing the configuration space of the core. Only the lower 256 bytes of configuration space are available for Read/Write access with Type 0 configuration cycles for guest (expansion bus) implementation of the core. The host bus bridge implementation provides the Configuration Read operation only.[1] Configuration Write cycles are accepted and acknowledged but have no effect on Configuration registers.

Addressing during configuration cycles differs from normal Read and Write cycles on a PCI bus (For more information, see *PCI Local Bus Specification Rev 2.2*, chapter 3.1.1, "Command Definition").

[1] Note: Because the host bus bridge normally generates configuration commands, and the PCI local bus specification does not require a host bus bridge to respond to configuration cycles, it is most likely that this feature will never be used.

Only Type 00h predefined header portion has been implemented in the lower 256 bytes of the configuration space (in this document also called PCI configuration space). For its organization, see *PCI Local Bus Specification Rev 2.2*, chapter 6.1.

3.2.4. Generating Configuration Cycles

The host bus bridge implementation of the core provides a mechanism for generating configuration cycles on a PCI bus by accessing the CNF_ADDR and CNF_DATA register.

Step 1: The WISHBONE master must write the appropriate data to the CNF_ADDR register, which holds information about register offset, function, device, and bus number. The TYPE bit in this register defines a type of configuration cycle that is generated on the PCI bus (0 = Type 0, 1 = Type 1). The **Offset** field in the CNF_ADDR register identifies a register offset to or from which the WISHBONE master wishes to write or read. The **Function** field is set to the function number of multifunctional device being a target of configuration cycle. The **Device** field defines the address line driven high during the address phase of the configuration cycle, which can be used as an IDSEL signal for a Type 0 configuration cycle. The **Bus** field is set to the bus number the targeted device resides on.

Step 2: To actually begin a configuration cycle on the PCI bus, the WISHBONE master must access the CNF_DATA register. Accesses to CNF_DATA are treated as Single Delayed transactions. The WISHBONE master's access to this register is retried. If it is a Read cycle, the PCI master module arbitrates for the PCI bus, performs the Configuration Read command with byte enables provided by the WISHBONE master (signals **SEL_I(3..0)**), and provides data on the WISHBONE interface when the WISHBONE master retries the transaction. In case of a Write access, the PCI master module arbitrates for the PCI bus, performs a Write cycle with provided byte enables (signals **SEL_I(3..0**)), and acknowledges the transaction when retried by the WISHBONE master.

Driving of PCI bus AD lines during the configuration cycle address phase depends on the TYPE of the configuration cycle. If the WISHBONE master sets the TYPE bit of CNF_ADDR to 1 (indicating Type 1 configuration cycle), the value of lines on the PCI bus is driven with contents of the CNF_ADDR register (AD[31..0] <= CNF_ADDR[31..0]) during address phase. If the TYPE bit indicates TYPE 0 configuration cycle, then AD[31..11] lines on the PCI bus are driven according to the following table (driving depends on the **Device** field in the CNF_ADDR register):

DEVICE field value	Value on AD[31..11] lines during address phase of configuration cycle
0000 0	0000 0000 0000 0000 0000 1
0000 1	0000 0000 0000 0000 0001 0
0001 0	0000 0000 0000 0000 0010 0
0001 1	0000 0000 0000 0000 0100 0
0010 0	0000 0000 0000 0000 1000 0
0010 1	0000 0000 0000 0001 0000 0
0011 0	0000 0000 0000 0010 0000 0

DEVICE field value	Value on AD[31..11] lines during address phase of configuration cycle
0011 1	0000 0000 0000 0100 0000 0
0100 0	0000 0000 0000 1000 0000 0
0100 1	0000 0000 0001 0000 0000 0
0101 0	0000 0000 0010 0000 0000 0
0101 1	0000 0000 0100 0000 0000 0
0110 0	0000 0000 1000 0000 0000 0
0110 1	0000 0001 0000 0000 0000 0
0111 0	0000 0010 0000 0000 0000 0
0111 1	0000 0100 0000 0000 0000 0
1000 0	0000 1000 0000 0000 0000 0
1000 1	0001 0000 0000 0000 0000 0
1001 0	0010 0000 0000 0000 0000 0
1001 1	0100 0000 0000 0000 0000 0
1010 0	1000 0000 0000 0000 0000 0
1010 1	0000 0000 0000 0000 0000 0
1011 0	0000 0000 0000 0000 0000 0
1011 1	0000 0000 0000 0000 0000 0
1100 0	0000 0000 0000 0000 0000 0
1100 1	0000 0000 0000 0000 0000 0
1101 0	0000 0000 0000 0000 0000 0
1101 1	0000 0000 0000 0000 0000 0
1110 0	0000 0000 0000 0000 0000 0
1110 1	0000 0000 0000 0000 0000 0
1111 0	0000 0000 0000 0000 0000 0
1111 1	0000 0000 0000 0000 0000 0

Table 3.1: Value on AD[31:11] PCI bus lines during address phase of configuration cycle Type 0

Specified driving of PCI bus lines AD[31..11] provides a mechanism for tying IDSEL signals of target devices directly to AD lines. This way, device 0 is connected with its IDSEL signal to AD[11], device number 1 to AD[12], until device 20 connects to AD[31]. A total of 21 targets can be accessed with configuration cycles through the PCI bridge. Combinations of **Device** field values of CNF_ADDR register 10101 through 11111 are valid and terminate with **Master Abort** on the PCI bus since none of the targets can respond to the cycle without its IDSEL signal being asserted. Configuration Write data is discarded while Read cycles return all 1s on the WISHBONE bus. The transaction is acknowledged as specified in *PCI Specification Rev. 2.2*.

Other AD lines on the PCI bus are driven during the address phase of the Type 0 configuration cycle with data stored in the CNF_ADDR register, as described in *PCI Specification Rev. 2.2*.

3.2.5. Generating Interrupt Acknowledge Cycles

A special mechanism provides the generation of Interrupt Acknowledge cycles on the PCI bus. The WISHBONE master must perform a Read cycle to the INT_ACK register. This Read cycle is treated as Single Delayed transaction retried until the PCI master module arbitrates for the PCI bus and fetches the data requested. Address and byte enables on the PCI bus are exact copies of ADR_I(31..0) and SEL_I(3..0). The address has no meaning during an interrupt acknowledge cycle while byte enables indicate the size of the interrupt vector returned.

Read cycles of this register from the PCI bus have no effect and return all 0s. Write cycles from the WISHBONE or PCI side are accepted but have no effect.

3.3. WISHBONE Slave Unit

The WISHBONE slave unit connects to WISHBONE masters acting as a slave. This section describes its basic functionality. It is divided into subsections, each of them describing what the WISHBONE master needs to do to initiate WISHBONE to PCI transactions.

3.3.1. WISHBONE Slave Unit Functionality

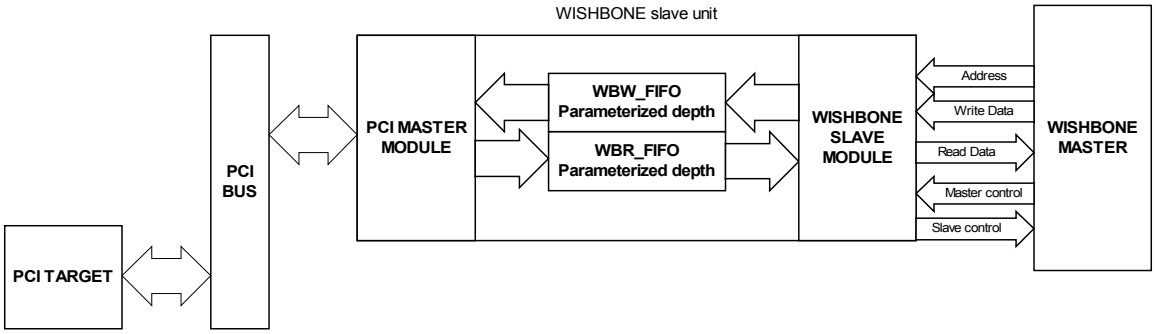


Figure 3.4: WISHBONE Slave Unit Architecture Overview

The WISHBONE slave unit consists of a few functional parts allowing the WISHBONE master to perform Read/Write accesses to the PCI bus.

3.3.1.1 WISHBONE Slave Module

The WISHBONE slave module is a 32-bit WISHBONE slave interface as defined in *WISHBONE Specification Rev. 1B*. It handles Read/Write cycles to images of PCI address space and configuration space accesses.

3.3.1.2 WBW_FIFO

The WISHBONE slave module uses WBW_FIFO (WISHBONE Write FIFO) for posting Memory and I/O Write cycles performed by the WISHBONE master. WBW_FIFO also performs a different bus clock adaptation.

3.3.1.3 WBR_FIFO

The WISHBONE slave module uses WBR_FIFO (WISHBONE Read FIFO) for storing data read from PCI targets. WBR_FIFO also performs a different bus clock adaptation.

3.3.1.4 PCI Master Module

The PCI master module is a 32-bit/66MHz, *PCI Local Bus Specification Rev. 2.2* compliant initiator interface. The core requests the PCI bus through its PCI master module and performs bus operations as described in the following subsections. Chapter PCI Interface provides a detailed overview of the PCI interface of the core.

3.3.2. Addressing and Images of the WISHBONE Slave Unit

As mentioned before, the WISHBONE slave unit incorporates 1 to 5 configurable WISHBONE address space images (the *Design Document and Implementation Notes* discuss in detail how the number of images is defined) and one image used for configuration space accesses from the WISHBONE bus with a fixed base address. This fixed base address points to the starting address of the configuration space. The base address for WISHBONE configuration space points to the offset address of the whole configuration space and is independent of the PCI Base Address0.

The behavior of each image is controlled by its WISHBONE Base Address (W_BA1 – W_BA5), WISHBONE Translation Address (W_TA1 – W_TA5), WISHBONE Image Control (W_IMG_CTRL1 – W_IMG_CTRL5) and WISHBONE Address Mask (W_AM1 – W_AM5) registers. Statuses, errors, and interrupts for each image are recorded in the Status registers of an image described later in this document. The WISHBONE slave module claims the cycle initiated by the master on the WISHBONE bus if one of the WISHBONE images is selected and enabled. An image is enabled if the IMG_EN bit of its W_AM register is set to 1. An image is selected when the address provided during the initial cycle on the WISHBONE bus falls into the range of that image. The range is determined by values of W_BA and W_AM registers. Each image can represent 4KB to 2GB of PCI address space. Whether an image is mapped to memory or I/O space is determined by the address space-mapping bit (ASM) of the image's W_BAn register. If this bit is 0, the image maps to memory space, otherwise to I/0 space.

How to specify a 1MB image of PCI address space with an address range of 0x10100000 - 0x101FFFFF?

The software must write a value of 0x10100XX0 to the image's Base Address register (the LSB of this register is set to 0 to indicate a memory space mapping). This way, the base address is set at 0x10100000. Twelve LS bits are marked as Don't Cares. The minimum block size is 4KB. Then, the software writes a value of 0xFFF00XXX into the W_AM register of the corresponding image. The IMG_EN bit is the MS bit and set to a value of 1 (it is also used for address masking – i.e. how we limit a maximum image size to 2GB). Each bit in the W_AM register corresponds to one address line – if a bit is 1, this address line is used for address comparison, and otherwise it is not. A value of 0xFFF00000 in the W_AM register means that ADR_I(31..20) signals are compared to W_BA[31..20] values. If values match, the image is selected. In this case, ADR_I(19..0) lines define an offset in an address range of 1MB.

Example 3-1: Address range of WISHBONE slave image

If enabled for a selected image (AT_EN bit of W_IMG_CTRLx is 1), address translation is performed between WISHBONE and PCI address by replacing the masked part of a WISHBONE address with the corresponding bits from the W_TA register. This provides very flexible address mapping.

Let's assume that base address and address mask are set as described in the previous example. We want a WISHBONE address range of 0x10100000 – 0x101FFFFF to be mapped elsewhere on the

PCI bus, e.g. 0x01000000 – 0x010FFFFF. To achieve this, we need a translation of addresses coming from the WISHBONE master and set the AT_EN bit of the corresponding W_IMG_CTRL register to a value of 1 and of the corresponding W_AT register to a value of 0x01000XXX. The W_AM register is already set, so address translation replaces ADR_I(31..20) provided by the WISHBONE master with a value of 0x010 set in the W_TA register for accesses on the PCI bus. This way, a PCI address range of 0x01000000 – 0x010FFFFF is accessible on the WISHBONE bus within a range of 0x10100000 – 0x101FFFFF.

Example 3-2: Address translation

3.3.3. WISHBONE to PCI Write Cycles

This section gives a detailed description of Write accesses, assuming that the WISHBONE slave unit has decoded an address to fall within a range of one of its enabled images.

The WISHBONE slave module is capable of handling Single and Block Write transfers through one of its WISHBONE slave images. Read Modify Write (RMW) cycles are not supported.

Note:
WISHBONE Slave Interface identifies the serial block transfers (bursts) using the CAB_I input signal that is outside the scope of the *WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores Specification*, prior to revision B3. This enables the PCI Initiator module to utilize burst transfers on the PCI Bus.

All Write cycles from the WISHBONE master to the PCI bus are handled as Posted Writes and are acknowledged on the WISHBONE bus immediately after receiving a request (before they are finished on the PCI bus) and stored in WBW_FIFO. Each image can be mapped to I/O or memory space, which is determined by a value of the address space-mapping bit (ASM) the W_BAx register of the corresponding image. If an image maps to I/O space, serial block transfers are not possible and the WISHBONE master receives an error signal. Normal block transfers are possible to I/O and memory space since every data beat in a block is treated as Single Posted Write cycle.

The WISHBONE Slave Unit ignores the value presented on the ADR_I(1:0) lines. If accessed image is mapped into the memory space, the value of AD[1:0] on the PCI bus during the address phase will be 0b00. If accessed image is mapped into the I/O space, the AD[1:0] on the PCI bus during the Address Phase will have the value generated according to the following table (value of x means don't care).

Value on SEL_I(3:0) lines	Generated PCI Bus Address (AD[1:0])
0bxxx1	0b00
0bxx10	0b01
0bx100	0b10
0b1000	0b11

Table 3.2: Generation of byte address for PCI I/O accesses

In some cases, Write cycles initiated by the WISHBONE master cannot be accepted and are terminated with **Retry**:

- WBW_FIFO is full or does not have enough space left to accommodate another transfer.
- A Delayed Read request is pending in the WISHBONE Slave unit (Write cycles cannot be posted until a Read cycle finishes on the PCI bus).
- A Delayed Read completion is present in the PCI Target unit.

The PCI master module requests a PCI bus after a complete transaction has been stored in the WISHBONE slave unit WBW_FIFO. After the PCI bus has been granted to the PCI master module, it initiates a transaction on the PCI bus. The module uses Memory Write or I/O Write PCI bus command, depending on the value of the address space-mapping bit (0 = memory, 1 = I/O) of the image's W_BAx register. In case the WISHBONE master posted a serial Block Write cycle, the PCI master module performs a burst of the same length to the PCI target. Single Posted Write cycles or non-serial Block Write cycles are completed as Single Write transactions on the PCI bus. If the PCI bus arbiter revokes mastership from the PCI master module (#GNT is deasserted), it finishes the current cycle and releases the PCI bus for which it afterwards has to re-arbitrate in order to continue any Posted Write cycles left in a WBW_FIFO. The core handles **Retry** and **Target Disconnect** terminations by retrying the transaction until it completes or some other termination is signaled.

All Write cycles are handled as posted writes and are therefore immediately acknowledged by the WISHBONE Slave. The PCI Bridge implements an error reporting mechanism for write cycles, that were already acknowledged on the WISHBONE bus but later terminated with an error on the PCI bus. Error Reporting registers provide an Error Reporting mechanism. Error Reporting must be enabled by the errors enable (ERR_EN) bit of the WISHBONE Error Control and Status (W_ERR_CS) register. When enabled, errors can generate interrupts if the error interrupt enable (EINT_EN) bit of the W_ERR_CS register is 1. Each of the Error Reporting registers stores a part of information about the Posted Write transaction on the PCI that was terminated with an error.

- A value of 1 in the error signaled bit (ERR_SIG) of the W_ERR_CS register indicates that an error has been recorded. The Field Bus Command (BC) of this register stores a bus command used for an access that has been terminated with Error, while field Byte Enables (BE) stores the value of byte enables during the transfer. The error source bit (ES) indicates the source of an error (1 = Master (Master Abort), 0 = Target (**Target Abort**)).
- W_ERR_ADDR stores a 32-bit address that the PCI master module tried to access when the error occurred.
- W_ERR_DATA stores 32 bits of data used in a transfer that was terminated with an error.

Error terminated write transactions are discarded while other posted transactions proceede normally.

3.3.4. WISHBONE to PCI Read Cycles

Read cycles initiated by the WISHBONE master are handled as Delayed Read cycles. The Bridge does not support Multiple Delayed Read requests. Delayed transactions must be completed on the PCI bus before they can be completed on the WISHBONE bus. The section on addressing and images has described how the WISHBONE slave unit decodes addresses to know if it is a slave for a current cycle. Handling of Read transactions is encoded in the Image Control register (W_IMG_CTRLx). There are a few options how to define the behavior of the WISHBONE slave unit during Read transactions for images mapped to memory space:

- If PREF_EN and MRL_EN bits are both cleared, that indicates that the Image's address range is not prefetchable. The PCI Bridge will read a single word of data from the PCI bus using the Memory Read command, regardless of the type of the read cycle requested.
- The PREF_EN bit indicates that the address range of an image is prefetchable When it is set and MRL_EN bit is cleared and serial block read is requested, the PCI Bridge will prefetch one cache line of data (set in the Cache Line Size register) using the Memory Read command on the PCI bus.
- The MRL_EN bit indicates that the address range of an image is prefetchable When it is set and PREF_EN bit is cleared and serial block read is requested, the PCI Bridge will prefetch one cache line of data (set in the Cache Line Size register) using the Memory Read Line command on the PCI bus.
- When both PREF_EN and MRL_EN bits are set and serial block read is requested, the PCI Bridge will prefetch as much data as it can fit into the WBR_FIFO and use the Memory Read Multiple command on the PCI bus.

The PCI Bridge reads all words within a prefetched burst memory read with all byte enables asserted on the PCI bus. It uses the negated byte enables sent by the external WISHBONE Master for single memory or I/O reads.

The PCI Bridge handles only single read or non-serial block read requests through the images mapped to I/O space. It responds with an error on the WISHBONE bus when serial block read through an I/O mapped image is requested. It never prefetches the I/O read requests – it always reads a single 32 bit word using the I/O Read command and byte enables sent from the WISHBONE Master on the PCI bus.

Non-prefetchable address space is assumed for the following conditions:

1. Accesses to I/O mapped address space are always non-prefetched.
2. The WISHBONE master performs a Single or non-serial Block Read cycle, or the PREF_EN and MRL_EN bits are both cleared.

When the WISHBONE slave unit latches address and SEL_I(3:0) data of a Read request, the PCI master module requests mastership for the PCI bus. When mastership is granted, the PCI master module initiates a PCI Read transaction. The bus command used for the transaction depends on various parameters described in the following table:

Address space mapping of image	Cycle initiated by WISHBONE master	PREF_EN bit value	MRL_EN bit value	Bus command used
I/O	Single or Block Read	X	X	I/O Read
Memory	Single or Block Read	X	X	Memory Read
	Serial Block	0	0	Memory Read

Address space mapping of image	Cycle initiated by WISHBONE master	PREF_EN bit value	MRL_EN bit value	Bus command used
	Read	0	1	Memory Read Line
		1	0	Memory Read
		1	1	Memory Read Multiple

Table 3.3: Bus command encoding for Read cycles through PCI master module

Read cycles to address space that is not prefetchable are performed in one data phase on the PCI bus. After the first data phase, the PCI master module releases the PCI bus.

All serial block Delayed Read requests from address space marked as prefetchable are performed in Burst Read cycles. The PCI master module reads data from the target and puts it into WBR_FIFO. The PCI master module finishes a Burst Read cycle and releases the PCI bus if any of the following conditions is met:

1. WBR_FIFO is full.
2. The target issues **Target Disconnect**.
3. The mastership of the PCI bus is revoked by the PCI arbiter (#GNT is de-asserted).

When the WISHBONE master retries this Read transaction, data is ready and the WISHBONE slave module pulls data out of the WBR_FIFO and provides it on the WISHBONE bus.

Any data left in WBR_FIFO after the WISHBONE master ends a Read cycle is flushed immediately.

So far, WISHBONE to PCI Read cycles have been described as if they always completed successfully, but it is common for PCI bus targets or masters to generate error terminations. Terminations from the PCI bus must be propagated to the WISHBONE bus to let the WISHBONE master know what happened to the transaction it initiated.

The WISHBONE Slave Unit's PCI Master handles the following terminations during completion of Delayed Read requests on the PCI bus:

- **Retry**
- **Disconnect** with data
- **Disconnect** without data
- **Target Abort**

The **Retry** termination is not propagated back to the WISHBONE bus. The PCI Initiator simply retries the transaction.

Disconnect is treated as normal termination. The PCI Master does not retry the reads terminated with disconnect, even if the full sized burst was not complete yet. It provides the data fetched before (or/and during) the Disconnect termination to the WISHBONE Master.

Target Abort is an error termination propagated back to the WISHBONE bus. The WISHBONE Slave module will provide any data fetched before Target Abort termination to the WISHBONE Master normally. When the WISHBONE Master initiates a read from the location that was

terminated with Target Abort on the PCI bus, WISHBONE Slave module terminates the transfer with an error. It is possible that WISHBONE Master will not access the Target Aborted location. In this case, the error will not be signaled.

Master Abort is an error termination. When the WISHBONE master retries the read request terminated with Master Abort on the PCI bus, the WISHBONE Slave module will terminate the cycle with an error.

3.3.5. WISHBONE SoC Interconnection Rev. B3 support

The WISHBONE Slave unit's WISHBONE Slave interface can be configured before synthesis to support the Registered Feedback cycles defined in WISHBONE bus specification Rev. B3.. See Appendix A for more information regarding hardware configuration. When you configure the hardware parameters properly, a special logic internal to the PCI Bridge translates Registered Feedback cycles to Single, Block or serial Block transfer cycles.

The following table describes the WISHBONE Rev. B3 cycle type translation for WISHBONE Slave Interface.

WISHBONE Rev. B3 Cycle Type	Starting address	WISHBONE Slave Internal Cycle type	PCI Master initiated transaction(s)
"Classic"	X	Single or Block	One or multiple single transfers.
"Constant Address Burst Cycle"	X	Block	Multiple single transfers.
"Incrementing Linear Burst Cycle"	X	serial Block	Burst transfer.
"Incrementing Wrap 4 Burst Cycle"	*wbs_adr_i[3:2]* == 0x0	serial Block	Burst transfer.
	wbs_adr_i[3:2] != 0x0	Block	Multiple single transfers.
"Incrementing Wrap 8 Burst Cycle"	*wbs_adr_i[4:2]* == 0x0	serial Block	Burst Transfer.
	wbs_adr_i[4:2] != 0x0	Block	Multiple single transfers.
"Incrementing Wrap 16 Burst Cycle"	*wbs_adr_i[5:2]* == 0x0	serial Block	Burst Transfer.
	wbs_adr_i[5:2] != 0x0	Block	Multiple single Transfers.
"Reserved"	X	Single or Block	One or multiple single transfers.
"End-Of-Burst"	X	Single	Single Transfer.

Table 3.4 - WISHBONE Slave Registered feedback cycle translation

Note that during burst cycles that decode to serial Block cycles each write transfer takes 3 WISHBONE clock cycles to complete (if WBW_FIFO is not full), while each read transfer takes 1 WISHBONE clock cycle (if read data is present in the WBR_FIFO) to complete.

3.4. PCI Target Unit

The PCI target unit connects to PCI initiators acting as a target. This section describes the basic functionality of the PCI target unit and is divided into subsections, each of them defining what a PCI initiator needs to do to initiate PCI to WISHBONE transactions.

3.4.1. PCI Target Unit Functionality

This part gives a functional overview of the PCI target unit. Detailed description is provided in the following sections.

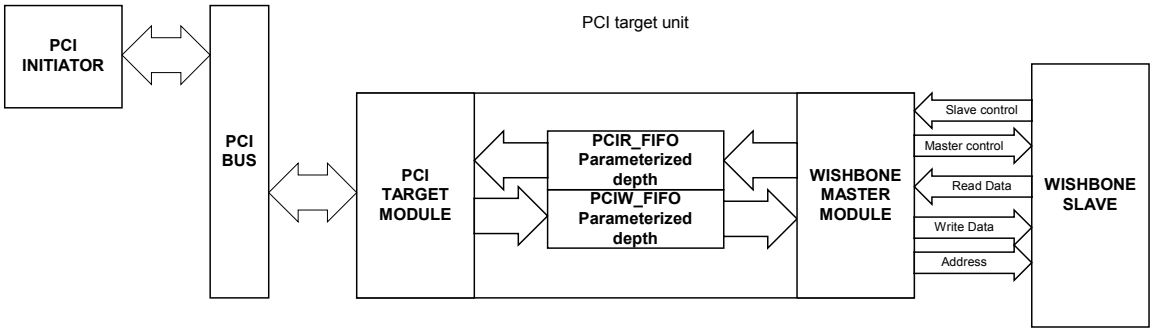


Figure 3.5: PCI target unit architecture overview

The PCI target unit consists of several functional parts allowing PCI initiators to perform Read/Write access to the WISHBONE bus.

3.4.1.1 PCI Target Module

The PCI target module is a 32-bit/66MHz (33MHz for FPGA implementation) *PCI Local Bus Specification Rev. 2.2* compliant target interface. It handles Read/Write cycles to images of WISHBONE address space and configuration space accesses.

3.4.1.2 PCIW_FIFO

The PCI target module uses PCIW_FIFO (PCI Write FIFO) for posting Memory and I/O Write cycles performed by the PCI initiator. PCIW_FIFO also performs a different bus clock adaptation.

3.4.1.3 PCIR_FIFO

The WISHBONE master module uses PCIR_FIFO (PCI Read FIFO) for storing data read from WISHBONE slaves. PCIR_FIFO performs a different bus clock adaptation.

3.4.1.4 WISHBONE Master Module

The WISHBONE master module is a 32-bit WISHBONE master interface as defined in *WISHBONE Specification Rev. 1B*. The core requests the WISHBONE bus through its WISHBONE master module. Chapter WISHBONE Slave Interface, describes in detail the WISHBONE interface of the core.

3.4.2. Addressing and Images of the PCI Target Unit

As mentioned above, the PCI target unit incorporates 1 to 5 configurable PCI address space images (The *Design Document and Implementation Notes* discuss in detail how to define the number of images) and one special image used for configuration space accesses from the PCI bus with a configurable base address. In host bridge implementations, this special image can be configured to provide access to normal address space or can be canceled – therefore configuration space would not be accessible (see also Configuration Space Access for Host Bus Bridges and Register List and Description).

The behavior of each image is controlled by its PCI Base Address (P_BA0 – P_BA5), PCI Translation Address (P_TA0 – P_TA5), PCI Image Control (P_IMG_CTRL0 – P_IMG_CTRL5), and PCI Address Mask (P_AM0 – P_AM5) registers. Status, errors, and interrupts for each image are recorded in the Status registers described later in this document. The PCI target module claims the transaction started by the initiator on the PCI bus if one of the PCI images is selected and enabled. An image is enabled if the IMG_EN bit of its P_AM register is set to 1. An image is selected when the address provided during the address phase on the PCI bus is within the memory or I/O range of that image. The range is determined with values of P_BA and P_AM registers. Each image can represent 4KB to 2GB of the WISHBONE address space.

Each image can be mapped to memory or I/O space, determined by the address space-mapping bit (ASM) of the image's P_BAx register (bit 0). If the ASM bit is 0, the image maps to memory space, and otherwise to I/O space. For host bridge implementations, the predefined values can later be changed by writing an appropriate value, but for guest bridge implementations, the predefined values are fixed (hardwired), because device independent software must know in advance where to map each PCI Base Address.

How to specify a 1MB image of WISHBONE address space with an address range of 0x10100000 – 0x101FFFFF?

Software must write a value of 0x10100XX0 to the Base Address register of an image (the LSB of this register is set to 0 to indicate memory space mapping). This way, the base address is set at 0x10100000. Twelve LS bits are marked as Don't Cares. The minimum block size is 4KB. Software writes a value of 0xFFF00XXX into the P_AM register of the corresponding image. The MS bit is the IMG_EN bit, which is set to a value of 1. It is also used for address masking, i.e. how we limit a maximum image size to 2GB. Each bit in the P_AM register corresponds to one address line. If the bit is 1, then this address line is used in address comparison, and otherwise it is not. A value of 0xFFF00000 in the P_AM register means that AD[31:20] signals received during the address phase are compared with a P_BA[31..20] value. If values match, the image is selected. In this case, AD[19..0] lines define an offset in an address range of 1MB.

Example 3-3: Address range of PCI Target Image

If address translation is enabled for a selected image (AT_EN bit of P_IMG_CTRLx is 1), it is performed between PCI and WISHBONE address. Address translation is done by replacing the masked part of the PCI address with the corresponding bits from the P_TA register. This provides very flexible address mapping (off course address translation must be implemented).

Let's assume that base address and address mask are set as described in previous example. We want a PCI address range of 0x10100000 – 0x101FFFFF to be mapped elsewhere on the WISHBONE bus, e.g. at 0x01000000 – 0x010FFFFF. To achieve this, we need a translation of addresses coming from the PCI initiator. The AT_EN bit of the corresponding P_IMG_CTRL register is set to a value of 1 and of the corresponding P_TA register to a value of 0x01000XXX, respectively. The P_AM register is already set, so address translation replaces AD(31..20) provided by the PCI initiator with a 0x010 value set in the P_TA register for accesses on the WISHBONE bus. This way, we have a WISHBONE address range of 0x01000000 – 0x010FFFFF, accessible on the PCI bus in a range of 0x10100000 – 0x101FFFFF.

Example 3-4: Address translation

3.4.3. PCI to WISHBONE Write Cycles

The previous section described how a PCI target unit knows if it is the target of a current cycle initiated by a PCI initiator. In this section, Write accesses are described in detail, assuming that a PCI target unit decodes an address to fall within a range of one of its enabled images.

The PCI target module is capable of handling Single and Burst Write transfers through one of its PCI target images.

Note:

The WISHBONE Master module performs burst transfers as Block Cycles, described in the *WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores Specification*. It implements a special CAB_O signal, outside the scope of WISHBONE bus specification, to identify them.

All Write cycles, originating on the PCI bus and ending on the WISHBONE bus are handled as Posted Writes. Due to this, the WISHBONE Master Module does not support Read Modify Write cycle. Write cycles are claimed on the PCI bus immediately after receiving a request and are stored in PCIW_FIFO. Each image can be mapped to I/O or memory space, which is determined by a value of the address space-mapping bit (ASM) in the P_BAx register of the corresponding image (for guest bridges ASM bit is fixed, for host bridges ASM bit can be changed, see chapter 3.3.2).

If an image maps to I/O space, AD(1:0) lines indicate the least significant valid byte for the transaction. The byte enable lines BE#(3:0) indicate the size of the transfer within the DWORD. They must be consistent with AD(1:0) as seen in Table 3.5. All other combinations are invalid. Invalid access is terminated with **Target Abort** on the PCI bus.

All PCI bursts to I/O space are treated as Single Posted Writes; therefore, Burst transfers are broken into single transfers. The PCI Target module posts only one 32 bit word and responds with disconnect on subsequent data phases.

Value on AD(1:0) lines	Starting Byte	Valid BE#(3:0) encoding
00	Byte 0	xxx0
01	Byte 1	xx01
10	Byte 2	x011
11	Byte 3	0111

Table 3.5: Valid AD(1:0) and BE# (3:0) combinations for I/O mapped address space accesses

If an image maps to memory space, 30 AD lines (the AD(31:2)) provide a DWORD-aligned address. The AD(1:0) lines indicate the burst mode, as seen in Table 3.6. The Linear Incrementing Burst mode is fully supported, while the Cache-line Wrap and reserved mode writes are broken into single transfers The PCI Target module posts a single 32 bit word and responds with disconnect on subsequent data phases, if unsupported burst mode is requested.

Value on AD(1:0) lines	Burst Ordering encoding
00	Linear Incrementing
01	Reserved (disconnect after first data phase)
10	Cache-line Wrap mode (disconnect after first data phase)
11	Reserved (disconnect after first data phase)

Table 3.6: Burst Ordering combinations for memory mapped address space accesses

The PCI target unit terminates a write transaction with **Disconnect,** when the following conditions apply:

- The external PCI Master initiates a burst write transaction to I/O mapped PCI Target Image.
- The external PCI Master initiates a burst write transaction to memory mapped PCI Target Image with unsupported burst mode.
- The target is temporarily unable to continue bursting when PCIW_FIFO is filled during the current Burst Write.

The PCI target unit abnormally terminates a transfer with **Target Abort** when it detects a fatal error of the following kind (otherwise it would not be able to complete the requested transfer):

- The master initiates a non-valid combination of AD(1:0) and BE#(3:0) when accessing I/O mapped image space (as mentioned above).

When it is busy and temporarily unable to process the transaction, the PCI target unit terminates a transfer with **Retry** before any data is transferred. This applies to the following situations:

- The PCIW_FIFO is full or cannot provide enough space to accommodate another burst transfer.
- A Delayed Read request is pending in the PCI target unit (Write cycles cannot be posted until a Read cycle finishes on the WISHBONE bus).
- A Delayed Read Completion is present in the WISHBONE Slave unit.

The WISHBONE master module initiates a write cycle on the WISHBONE bus after a complete PCI write transaction has been stored in the PCIW_FIFO of the PCI Target unit. The module uses Single Write or Block Write WISHBONE cycles. The Block Write cycles on the WISHBONE bus are of the same length as write bursts accepted on the PCI bus.

All Write cycles are handled as posted writes and are therefore immediately accepted by the PCI Target module. The PCI Bridge implements an error reporting mechanism for write cycles, that were already accepted on the PCI bus but later terminated with an error on the WISHBONE bus. Error Reporting registers provide an Error Reporting mechanism. Error Reporting must be enabled by the errors enable (ERR_EN) bit of the PCI Error Control and Status (P_ERR_CS) register. When enabled, errors can generate interrupts if the error interrupt enable (EINT_EN) bit of the P_ERR_CS register is 1. Each of the Error Reporting registers stores a part of information about the Posted Write transaction on the WISHBONE bus that was terminated with an error.

- A value of 1 in the error signaled bit (ERR_SIG) of the P_ERR_CS register indicates that an error has been recorded. The Field Bus Command (BC) of this register stores the bus command used on the PCI bus for the access that terminated with an error on the WISHBONE bus while the field Byte Enables (BE) stores the value of byte enables (SEL_O(3:0) lines) during the transfer.
- P_ERR_ADDR stores the 32-bit address the WISHBONE master module tried to access when the error occurred.
- P_ERR_DATA stores the 32 bits of data used in the transfer on the WISHBONE bus that terminated with an error.

The remainder of the write transaction that generated an error is discarded, any subsequent transactions are processed normally.

3.4.4. PCI to WISHBONE Read Cycles

Read transactions initiated by the external PCI master are handled as Delayed Read transactions. Delayed Read transactions must be completed on the WISHBONE bus before they can complete on the PCI bus. The PCI Bridge supports only one Delayed Read request at a time. The Read transaction starts when the external PCI master initiates a read transaction within an address range of one of the Target images. The PCI Target module terminates it with **Retry** and stores address and byte enable information. The WISHBONE master module performs a Single or Block Read cycle on the WISHBONE bus and stores the data in the PCIR_FIFO. When external PCI master repeats the read request, the PCI Target unit provides the data stored in the PCIR_FIFO.

Above, the section on addressing and images described how the PCI target unit decodes an address to find out if it is the target for current read transaction. Handling Read transactions is encoded in the PCI Image Control register (P_IMG_CTRLx). Several options exist to define the PCI target unit's behavior when images mapped to memory space are accessed:

- The PREF_EN bit indicates that the address range of the image is prefetchable regardless of the Memory Read command used to access it. The PCI Bridge will prefetch data from the WISHBONE bus when an image with PREF_EN bit set is accessed.
- If the PCI Target is accessed using the Memory Read Line or Memory Read Multiple commands, the memory space is assumed to be prefetchable. The PCI Bridge will prefetch the read data from the WISHBONE bus regardless of the value of PREF_EN bit.

Non-prefetchable address space is assumed for the following conditions:

- Read accesses to I/O mapped address space are always non-prefetched.
- The PCI initiator requests a Single Read transaction (Memory Read command), and the PREF_EN bit is cleared.

The following table shows the PCI Target unit's response to different PCI Read commands and PREF_EN bit settings.

Address space mapping of image	Bus command initiated by PCI initiator	PREF_EN bit value	Used cycle by WISHBONE master
I/O	I/O Read	X	Single Read
Memory	Memory Read	0	Single Read
	Memory Read	1	Block Read (1 cache line)
	Memory Read Line	X	Block Read (1 cache line)
	Memory Read Multiple	X	Block Read (full FIFO)

Table 3.7: Bus command encoding for Read cycles through PCI target module

Single Read cycles are performed in one data phase on the WISHBONE bus. The WISHBONE Master module reads only the bytes requested by the external PCI Master – it activates only those SEL_O(3:0) lines that correspond to active (BE#) lines during the read request on the PCI bus.

All prefetchable Delayed Read transactions from memory address space are performed as Block Read cycles. All byte enable lines are active during Block Read cycles on the WISHBONE bus. The WISHBONE master module reads data from the external WISHBONE slave and puts it into PCIR_FIFO. It finishes a Block Read cycle and releases the WISHBONE bus if any of the following conditions occurs:

- The requested quantity of data was stored in the PCIR_FIFO.
- PCIR_FIFO is full.
- The WISHBONE slave terminates the cycle with **Error** or **Retry**.

When the PCI initiator retries this Read transaction, data is ready and the PCI target module pulls out data from PCIR_FIFO and provides it on the PCI bus.

Any data not read from the PCIR_FIFO is flushed immediately after the read transaction is finished on the PCI bus.

If PCIR_FIFO becomes empty while the read transaction is completing on the PCI bus, the PCI Target module terminates with Target Disconnect. The external PCI Master must repeat the read request if it wants to read more data.

WISHBONE Slave terminations during read transfers are handled in the following manner:

- **Acknowledge** is a normal termination during which the transfer occurs. The read data is stored in the PCIR_FIFO and waits to be fetched by external PCI Master.
- **Retry** is a normal termination during which the transfer does not occur. If the Retry termination is received during the first attempted read transfer, then the WISHBONE Master module retries until some other termination is received.
- **Error** termination is stored in the PCIR_FIFO. The read cycle is not retried on the WISHBONE bus. When the external PCI Master repeats the read request on the PCI Bus, the PCI Target module terminates with Target Abort.

The PCI target unit abnormally terminates a read transfer with **Target Abort** without any access to the WISHBONE bus, if the external PCI master initiates a read transaction with non-valid combination of AD(1:0) and BE#(3:0) when accessing the I/O mapped image.

3.4.5. WISHBONE SoC Interconnection Rev. B3 support

Since the release of WISHBONE SoC bus specification Rel. B3, the out of specification identification of serial Block cycles is no longer necessary. The following table specifies what kind of Registered Feedback cycle the WISHBONE Master Interface initiates on the WISHBONE bus for different kinds of PCI transactions.

PCI Bus Transaction	WISHBONE Master Cycle Type

PCI Bus Transaction	WISHBONE Master Cycle Type
Single Write Transfer	"End-Of-Burst"
Burst Write Transfer	"Incrementing Linear Burst Cycle" – **see Note below**
Single Read Request	"End-Of-Burst"
Prefetched Read Request	"Incrementing Linear Burst Cycle"

Table 3.8 - WISHBONE Master Registered Feedback cycle support

Note: It is not common for PCI devices to change byte enable (BE#) lines during burst write transfers, but it is allowed. The WISHBONE Registered Feedback cycles' specification does not allow the byte enable (SEL_O(3:0)) lines to change between transfers in the same burst cycle. Therefore, the WISHBONE Master interface breaks up the PCI burst write transactions into smaller burst or single cycles on the WISHBONE bus, if the byte enables change between the dataphases of the PCI burst write transfer.

3.5. Transaction Ordering

In order to satisfy PCI transaction ordering rules, the following functionality is implemented:

1. When the WISHBONE slave unit receives a Read request and no other Delayed Read request or completion is pending, it latches address and byte enable information and terminates the cycle with **Retry**.
2. After receiving a Read request, the WISHBONE slave unit locks out any non-configuration space access. (All requests to the WISHBONE slave unit are terminated with **Retry**.)
3. The PCI master module processes Posted Write cycles until WBW_FIFO is empty.
4. The PCI master module performs a requested Read transaction on the PCI bus.
5. When a Read transaction is finished on the PCI bus, it becomes a Delayed Read completion and WISHBONE Slave unit can accept Posted Write cycles.
6. The PCI target module retries all non-configuration space accesses from the PCI bus.
7. The WISHBONE Master module processes all Posted Writes until PCIW_FIFO is empty.
8. The WISHBONE slave unit allows a Delayed Read completion to finish on the WISHBONE bus.
9. When the PCI Target unit receives a Read request and no other Delayed Read request or completion is pending, it latches address and byte enable information and terminates the cycle with **Retry**.
10. After receiving a Read request, the PCI target unit locks out any non-configuration space access. (All requests to the PCI target unit are terminated with **Retry**.)
11. The WISHBONE master module processes Posted Write cycles until PCIW_FIFO is empty.
12. The WISHBONE master module performs a requested Read transaction on the WISHBONE bus.

13. When a Read transaction is finished on the WISHBONE bus, it becomes a Delayed Read completion and PCI target unit can accept Posted Write cycles.
14. The WISHBONE slave module retries all non-configuration space accesses from the WISHBONE bus.
15. The PCI Master module processes all Posted Writes until WBW_FIFO is empty.
16. The PCI target unit allows a Delayed Read completion to finish on the PCI bus.

3.6. PCI Bus Parity generation and checking

Parity monitoring and generation is required by all PCI agents according to the *PCI Local Bus Specification*. The PCI Bridge core generates even parity across AD, C/BE# and PAR signals during:

- the address phase initiated by the internal PCI master module.
- all of the data phases of the internal PCI master module initiated write transactions.
- all of the data phases of the external PCI master initiated read transactions.

The PCI Bridge core monitors for even parity across AD, C/BE# and PAR signals during:

- the first or second address phase initiated by the external PCI master.
- all of the data phases of the internal PCI master module initiated read transactions.
- all of the data phases of the external PCI master initiated write transactions.

The PCI Bridge responds to detected or reported parity errors as specified in the *PCI Local Bus Specification*. It does not implement any additional functionality, such as terminating the transactions etc..

3.7. Interrupts

The PCI IP core is capable of generating interrupts in response to different events. Interrupt Control and Interrupt Status registers control the generation of interrupt requests. If the core is implemented as a guest bridge, interrupts are requested on the PCI bus through assertion of the INTA# pin; if it is implemented as a host, they are reported on the WISHBONE bus through assertion of the INTA_O pin. The Interrupt Control register is used for enabling/disabling interrupts originating from different sources. The Interrupt Status register is used to determine the source of an interrupt and to clear interrupt requests.

The software must locate and clear the source of an interrupt request before clearing status bits in the Interrupt Status register.

3.8. Compact PCI Hot Swap support

The support for Hot Swap is currently supported for GUEST implementations only.

The I/O signal and register interface for the Compact PCI Hot Swap support is designed in accordance to the *CompactPCI Hot Swap Specification R2.0*. The PCI Bridge RTL design includes the following additional I/O signals required by the Hot Swap specification:

- LED# output – controlled by *pci_cpci_hs_led_o* and *pci_cpci_hs_led_oe_o*.
- ENUM# output – controlled by *pci_cpci_hs_enum_o* and *pci_cpci_hs_enum_oe_o*.
- Handle Switch input – connected to *pci_cpci_hs_es_i*.

The PCI Bridge implements Programming Interface 0, as specified in the *CompactPCI Hot Swap Specification*.

The PCI Bridge also supports the register interface required by the *CompactPCI Hot Swap Specification*. The required register interface is implemented in the first 256 bytes of the bridge's configuration space – accessible via configuration transactions or memory transactions using PCI Base Address 0 range. The following changes in the Configuration Space can be observed, when the Hot Swap functionality is implemented:

- Bit 4 in the Status register of PCI configuration header (offset 0x6) is hardwired to value of 1.
- Capabilities Pointer in the PCI configuration header is set to the value of 0x80.
- The Hot Swap Control and Status Register is implemented on offset 0x80.

3.8.1. LED# output functional description

This is active low output signal with output enable. The PCI Bridge never drives this signal to inactive state – e.g. *pci_cpci_hs_led_o* is hardwired to the value of 0, while the PCI Bridge controls the value of *pci_cpci_hs_led_oe_o* to turn the blue led on or off. Open drain or output pad with output enable control can be used to drive the LED# signal. The active value for the *pci_cpci_hs_led_oe_o* is selected via the *pci_user_constants.v* file. See Appendix A for more information.

The PCI Bridge will activate the LED# output signal by asserting *pci_cpci_hs_led_oe_o* on the following conditions:

- *pci_rst_i* input signal is asserted (low).
- *pci_rst_i* input signal is de-asserted (high) and the power on configuration is in progress (when Serial Power On Configuration Interface is implemented).
- *pci_rst_i* input signal is de-asserted (high) and software writes a value of 1 to the LOO bit, located in the Hot Swap Control and Status Register.

3.8.2. ENUM# output functional description

This is active low output signal with output enable. The PCI Bridge never drives this signal to inactive state – e.g. *pci_cpci_hs_enum_o* is hardwired to the value of 0, while the PCI Bridge controls the value of *pci_cpci_hs_enum_oe_o* to assert or tri-state the ENUM# output signal. The active value for the *pci_cpci_hs_enum_oe_o* is selected via the *pci_user_constants.v* file. See Appendix A for more information. The PCI Bridge will activate the ENUM# output signal by asserting *pci_cpci_hs_enum_oe_o* if both of the following conditions apply:

- *pci_rst_i* input signal is de-asserted (high).
- either INS or EXT bit in the Hot Swap Control and Status Register is set (1) and EIM bit is cleared(0).

3.8.3. Handle Switch input functional description

The PCI Bridge constantly monitors the *pci_cpci_hs_es_i* input signal for any changes after *pci_rst_i* is de-asserted (high). The PCI Bridge interprets the low value on this signal as Handle Switch Unlocked indication and a high value as Handle Switch Locked indication. The state of the Handle Switch input does not influence the PCI Bridge's response to PCI bus transactions in any way. The assumed state of the switch after the PCI reset is Unlocked state. The signal must remain in the same state for at least 2^^16 clock cycles for 33MHz implementation and 2^^17 clock cycles for 66MHz implementation in order for PCI Bridge to process the switch state change. This accounts for the required de-bounce period for handle switch signal, specified in the *CompactPCI Hot Swap Specification*.

The PCI Bridge handles the Handle Switch input signal values in the following manner:

- During the insertion process (after the *pci_rst_i* is de-asserted)
    - the PCI Bridge performs a power on configuration, if the Serial Power On Configuration Interface is implemented. It does not respond to any PCI bus transactions during this time. It also does not respond to *pci_cpci_hs_es_i* value change. The de-bounce circuitry for *pci_cpci_hs_es_*i is enabled.
    - the power on configuration completes or is not implemented – the PCI Bridge waits for handle switch to go into Locked position (*pci_cpci_hs_es_i* high) or for the de-bounce period to expire (see text above). The PCI Bridge responds to valid PCI bus transactions during this time.
    - the *pci_cpci_hs_es_i* has been in high state (switch Locked) for at least the de-bounce period – the PCI Bridge sets the INS bit in the Hot Swap Control and Status Register to the value of 1. It ignores any further changes on the *pci_cpci_hs_es_i* until the INS bit is cleared.
- During the extraction process (the *pci_rst_i* is de-asserted, software has already cleared the INS bit)
    - if the *pci_cpci_hs_es_i* signal has been low (Handle Switch Unlocked) for at least the de-bounce period, the PCI Bridge sets the EXT bit in the Hot Swap Control and Status Register.

- if EXT bit in the Hot Swap Control and Status Register is set, the PCI Bridge ignores any changes on the *pci_cpci_hs_es_i* signal. The de-bounce circuitry is active and the PCI Bridge responds to any valid PCI bus transaction.
- if software clears the EXT bit in the Hot Swap Control and Status Register and Handle Switch has been Locked for at least the de-bounce period (*pci_cpci_hs_es_i* high), the PCI Bridge sets the INS bit in the same register.

3.8.4. PCI Device Status Register

Bit 4 in this register, located in the PCI Configuration Header, is hardwired to the value of 1, when Hot Swap implementation is selected. This indicates to the device independent software that Capabilities Pointer and Capabilities List are implemented.

3.8.5. Capabilities Pointer

When the device independent software determines the presence of the Capabilities List, it reads the Capabilities List Pointer register, to obtain the offset of the first item in the linked list of additional capabilities of the device. The PCI Bridge hardwires the value of Capabilities List Pointer to the value of 0x80, which is an offset of the Hot Swap Control and Status Register.

3.8.6. Hot Swap Control and Status Register

The register is implemented in the Configuration Space as part of the Capability List, specified in the *PCI Local Bus Specification, Revision 2.2*. It is located on the offset 0x80 and pointed to by Capabilities Pointer.

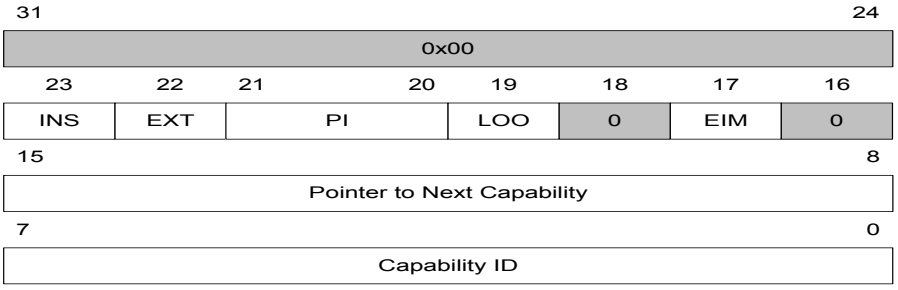


Figure 3.6: Hot Swap Control and Status Register layout

Bit #	Name	Description
31 – 24	HS_CSR Reserved	Reserved field. Write as zeros, reads return zeros.

Bit #	Name	Description
23	ENUM# Status – Insertion	If set, it indicates that the PCI Bridge has been plugged into the system. Cleared with writing a 1 to its position. It is set after the PCI reset is released, PCI Bridge has finished the configuration procedure and the Handle Switch is in the Locked position. It is also set when EXT bit has been cleared by software and the Handle Switch returned to locked position. This bit triggers the assertion of the ENUM# output, if EIM bit is 0.
22	ENUM# Status – Extraction	If set, it indicates that the operator wishes to unplug the PCI Bridge from the system. Cleared with writing a 1 to its position. This bit is set after the software clears the INS bit and the Handle Switch is in open state. This bit triggers the assertion of the ENUM# output, if EIM bit is 0.
21 – 20	Programming Interface	Since the PCI Bridge implements Programming Interface 0, this field is hardwired to 0b00.
19	Led On/Off	Turns the blue led on or off during normal operation. 1 – LED# output active (LED on) 0 – LED# output inactive (LED off)
18	Reserved	Write as 0, reads return 0.
17	ENUM# signal mask	Prevents the assertion of ENUM# output, even if one of INS or EXT bits is set.
16	Reserved	Write as 0, reads return 0.
15 – 8	Next Item	0x00 – Pointer to the next capability. A value of 0 means no other capabilities implemented.
7 – 0	Capability ID	0x06 – This ID number is assigned to CompactPCI Hot Swap capability.

Table 3.9: Hot Swap Control and Status Register field descriptions

3.9. Serial Power On Configuration Interface

The Serial Power On Configuration Interface is currently implemented only for the GUEST implementations of the PCI Bridge. This is a simple 2 wire serial interface used after local PCI bus reset is released. The PCI Bridge Core reads configuration data from the external serial EPROM and loads it into the configuration registers. While configuration loading procedure is in progress, the PCI Bridge does not respond to any PCI transactions, not even the configuration cycles. This qualifies the PCI Bridge as the *Initially not Responding Device* by the *CompactPCI Hot Swap Specification*. All of the PCI output pads are in HighZ state during the configuration loading procedure – e.g. all *pci_*_oe_o* signals are driven to their inactive state.

The Serial Power On Configuration Interface supports read and write operations to I²C compatible serial EPROM devices. The interface cannot be connected to the I²C bus, since it does not support arbitration and synchronization between multiple master devices. You can connect serial EPROMs with sizes ranging from 2 to 16 kbits. Both 66 and 33 MHz implementations have the transfer rate of 100kbps. In case of 66MHz implementation operating at 33MHz clock, the transfer rate will be 50kbps.

3.9.1. Serial EPROM Configuration Data Organization

When using Serial Power On Configuration Interface, the external serial EPROM must be programmed as described in this chapter, otherwise the PCI Bridge will not function properly. The configuration data must start at address 0. The following figure depicts how the PCI Bridge interprets the bytes read from the serial EPROM.

REGISTER NUMBER 0	0x0
DATA BYTE 0	0x1
DATA BYTE 1	0x2
DATA BYTE 2	0x3
DATA BYTE 3	0x4
REGISTER NUMBER 1	0x5
DATA BYTE 0	0x6
DATA BYTE 1	0x7
DATA BYTE 2	0x8
DATA BYTE 3	0x9
REGISTER NUMBER 2	0xA
. . . . .	
REGISTER NUMBER m	m * 5
DATA BYTE 0	m * 5 + 1
DATA BYTE 1	m * 5 + 2
DATA BYTE 2	m * 5 + 3
DATA BYTE 3	m * 5 + 4
REGISTER NUMBER n = 0xFF	n * 5
X	

Figure 3.7: Serial EPROM Data organization

During the Power On Configuration Sequence the PCI Bridge reads bytes from sequential addresses, starting at address 0x0. The first byte read is treated as the register number byte, followed by four data bytes, followed by second register number byte, followed by four data bytes and so on, until the register number byte with the value of 0xFF is read from the EPROM. When the PCI Bridge encounters the *REGISTER NUMBER* byte with the value of 0xFF, it stops the Power On Configuration Sequence. The value of 0xFF on the address 0x0 is valid and means that configuration

sequence will complete successfully without configuring any of the PCI Bridge Configuration registers'.

The bytes of *REGISTER NUMBER* type are used to identify the configuration space DWORD offset, to which the following four bytes of *DATA* type will be written to. For example, if you want to load a value into the P_AM1 register (offset 0x118) during power up, you have to program the *REGISTER NUMBER* byte to the value of 0x46 and the following four bytes to the value you want for P_AM1 register to have after power up. Basically, you have to shift the register offset by two bits to the right, to get the correct register number for it.

The bytes of *DATA* type are used to write the initial value to the configuration register selected by the *REGISTER NUMBER* byte in the following manner:

- *DATA BYTE 0* is used to write bits [7:0] of the selected register.
- *DATA BYTE 1* is used to write bits [15:8] of the selected register.
- *DATA BYTE 2* is used to write bits [23:16] of the selected register.
- *DATA BYTE 3* is used to write bits [31:24] of the selected register.

Make sure to program a value of 0x0 for reserved bits (or bytes) of the selected register. Even though most of the registers don't need all four bytes to be programmed, you must program all four bytes in the EPROM. This way the programming structure of the EPROM is maintained for the complete configuration space.

3.9.2. Power On Configuration Sequence

After PCI Reset signal is de-asserted, the following Sequential Read sequence can be observed on the Serial Power On Configuration Interface, if implemented:

1. The PCI Bridge transmits the control byte with value 0b10100000, MSB first. If acknowledge condition is not received, the sequence stops and NO_ACK bit in the Serial EPROM Control and Status Register is set.
2. The PCI Bridge transmits the data byte with value 0x00. If acknowledge is not received, the sequence stops and NO_ACK bit in the Serial EPROM Control and Status Register is set. This sets EPROM's address counter to the value of 0.
3. After acknowledge in step 2 is received, the PCI Bridge immediately sends another control byte, before generating the stop condition on the interface. The value of the control byte is 0b10100001. The EPROM must acknowledge this byte, otherwise the sequence is stopped and NO_ACK bit in the Serial EPROM Control and Status Register is set. This puts the serial EPROM into the read mode.
4. The EPROM starts transmitting the *REGISTER NUMBER* and *DATA* bytes and the PCI Bridge receives them and writes the data into the configuration registers. The PCI Bridge acknowledges all bytes until the *REGISTER NUMBER* byte with the value of 0xFF is read from the EPROM. The PCI Bridge does not acknowledge this byte and generates the stop condition, which signals to external EPROM that it must stop transmitting the data. The configuration sequence has now successfully completed.

After the completion of the configuration sequence (successful or unsuccessful), the PCI Bridge starts responding to valid PCI transactions.

3.9.3. Serial EPROM Control and Status Register

This register is used to determine the result of the latest serial EPROM operation as well as for initiating read and write operations to/from external EPROM during normal operation.

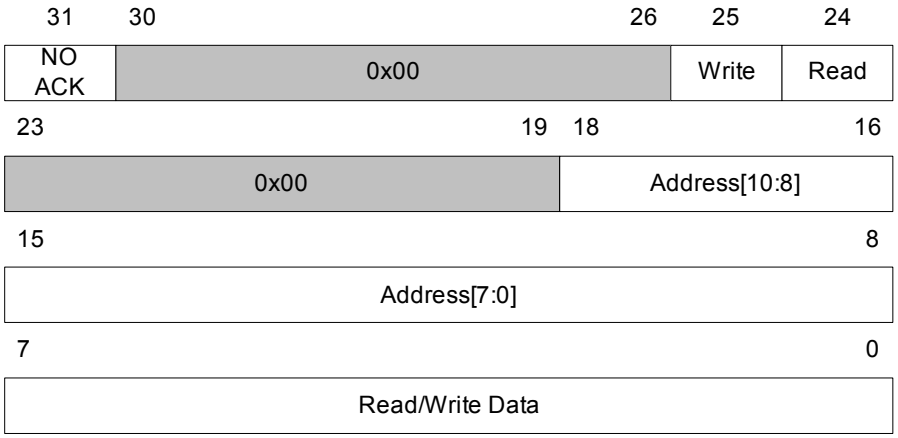


Figure 3.8: 3.9.3.Serial EPROM Control and Status Register layout

The register is located at the offset 0x3FC (register number 0xFF), which prevents the register to be accessed during the power on configuration sequence. Reset value is 0x00000000. When initialization completes and the PCI Bridge starts responding to the PCI bus transactions, the *NO ACK* bit can be set, which means that the Power On Configuration Sequence was not completed successfully. The register cannot be accessed using the PCI Configuration Read/Write commands. The PCI Target Base Address 0 must first be configured to enable accesses to this register.

Bit #	Name	Description
31	NO ACK	Set when serial EPROM does not send the acknowledge bit after control or data byte is transmitted. This could mean that serial EPROM is not present, is busy during write or erase operation or is write protected. The bit is cleared with writing a 1 to its location.
30-26	Reserved	Write as 0, reads return 0.
25	Write	Software sets this bit when it wants to initiate the byte write sequence to the external serial EPROM. The bit is cleared automatically when the write sequence is completed. **Do not**

Bit #	Name	Description
		change the contents of the register while this bit is set.
24	Read	Software sets this bit when it wants to initiate the random read sequence to the external serial EPROM. The bit is cleared automatically when the read sequence is completed. **Do not change the contents of the register while this bit is set.**
23-19	Reserved	Write as 0, reads return 0.
18-8	Byte Address	Software sets this field to the address of the byte in the serial EPROM it wants to read or write.
7-0	Byte Value	Software puts the byte value to this field when initiating the write sequence. The data retrieved during the read sequence can be obtained from this field, after the read sequence completes.

Table 3.10: Serial EPROM Control and Status Register fields

You must not set *Write* and *Read* bits to the value of 1 simultaneously, otherwise undefined behaviour can occur.

3.9.4. Initiating EPROM Byte Write Sequence

The software must follow this steps to program a single byte in the serial EPROM attached to the PCI Bridge:

1. Write appropriate value into the Serial EPROM Control and Status Register (byte value into the Byte Value field, byte address into the Byte Address field and Write bit set to 1).
2. Poll the Write bit in the Serial EPROM Control and Status Register until it is clear (0). When Write and NO ACK bits are both read as 0, the write sequence completed successfully.

After Write bit in the Serial EPROM Control and Status Register is set, the PCI Bridge performs the following sequence:

1. Transmits a START condition on the serial interface.
2. Transmits 0b1010 sequence, followed by the 3 most significant bits of the Byte Address, followed by 0 (write). If the EPROM responds with the ACK condition, the PCI Bridge continues with step 3, otherwise it stops transmitting, sets the NO ACK bit and clears the Write bit, both located in Serial EPROM Control and Status Register.
3. Transmits 8 least significant bits of the Byte Address. If the EPROM responds with the ACK condition, the PCI Bridge continues with step 4, otherwise it stops transmitting, sets the NO ACK bit and clears the Write bit, both located in Serial EPROM Control and Status Register.

4. Transmits 8 bits of the Byte Value. If the EPROM responds with the ACK condition, the PCI Bridge continues with step 5, otherwise it stops transmitting, sets the NO ACK bit and clears the Write bit, both located in Serial EPROM Control and Status Register.
5. Transmits the STOP condition, clears Write bit in the Serial EPROM Control and Status Register. The write sequence completed successfully.

Usually, it takes some time for EPROMs to program a new byte value into their array. During this time, they do not respond to accesses. Therefore, you will probably have to initiate a write sequence more that once, when programming multiple bytes in the EPROM. You have to repeat a write until it finishes successfully (Write bit clear, NO ACK bit clear) or wait appropriate amount of time, before initiating subsequent byte write sequence.

3.9.5. Initiating EPROM Byte Read Sequence

The software must follow this steps to read a single byte from the serial EPROM attached to the PCI Bridge:

1. Write appropriate value into the Serial EPROM Control and Status Register (byte address into the Byte Address field and Read bit set to 1).
2. Poll the Read bit in the Serial EPROM Control and Status Register until it is clear (0). When Read and NO ACK bits are both read as 0, the Byte Value field contains the data read from the external serial EPROM.

After Read bit in the Serial EPROM Control and Status Register is set, the PCI Bridge performs the following sequence:

1. Transmits a START condition on the serial interface.
2. Transmits 0b1010 sequence, followed by the 3 most significant bits of the Byte Address, followed by 0 (write). If the EPROM responds with the ACK condition, the PCI Bridge continues with step 3, otherwise it stops transmitting, sets the NO ACK bit and clears the Read bit, both located in Serial EPROM Control and Status Register.
3. Transmits 8 least significant bits of the Byte Address. If the EPROM responds with the ACK condition, the PCI Bridge continues with step 4, otherwise it stops transmitting, sets the NO ACK bit and clears the Read bit, both located in Serial EPROM Control and Status Register.
4. Transmits a START condition on the serial interface.
5. Transmits 0b1010 sequence, followed by the 3 most significant bits of the Byte Address, followed by 1 (read). If the EPROM responds with the ACK condition, the PCI Bridge continues with step 6, otherwise it does not receive any data, sets the NO ACK bit and clears the Read bit, both located in Serial EPROM Control and Status Register.

6. Shifts in 8 bits transmitted by the serial EPROM and stores them into the Byte Value field in the Serial EPROM Control and Status Register.
7. Transmits a NO ACK condition.
8. Transmits a STOP condition and clears the Read bit in the Serial EPROM Control and Status Register. The read sequence completed successfully.

4. Registers

This section describes all Control and Status registers inside the PCI core, also called configuration space. It consists of the PCI Configuration Space Header (Type 00h) and device specific Configuration Space registers. The **Width** field specifies the number of bits in the register, *Access* specifies the valid access types, R/W stands for Read and Write access, and R for Read Only access.

4.1. Register List and Description

Name	Address	Width	Access	Description
PCI Configuration Space	0x000 – 0x0FF			PCI Specification Rev. 2.2 configuration space
P_IMG_CTRL0*	0x100	32	R/W	PCI Image0 Control register
P_BA0*	0x010 and 0x104	32	R/W	PCI Image0 Base Address register
P_AM0*	0x108	32	R/W	PCI Image0 Address Mask register
P_TA0*	0x10C	32	R/W	PCI Image0 Translation Address register
P_IMG_CTRL1	0x110	32	R/W	PCI Image1 Control register
P_BA1	0x014 and 0x114	32	R/W	PCI Image1 Base Address register
P_AM1	0x118	32	R/W	PCI Image1 Address Mask register
P_TA1	0x11C	32	R/W	PCI Image1 Translation Address register
P_IMG_CTRL2	0x120	32	R/W	PCI Image2 Control register
P_BA2	0x018 and 0x124	32	R/W	PCI Image2 Base Address register
P_AM2	0x128	32	R/W	PCI Image2 Address Mask register

Name	Address	Width	Access	Description
P_TA2	0x12C	32	R/W	PCI Image2 Translation Address register
P_IMG_CTRL3	0x130	32	R/W	PCI Image3 Control register
P_BA3	0x01C and 0x134	32	R/W	PCI Image3 Base Address register
P_AM3	0x138	32	R/W	PCI Image3 Address Mask register
P_TA3	0x13C	32	R/W	PCI Image3 Translation Address register
P_IMG_CTRL4	0x140	32	R/W	PCI Image4 Control register
P_BA4	0x020 and 0x144	32	R/W	PCI Image4 Base Address register
P_AM4	0x148	32	R/W	PCI Image4 Address Mask register
P_TA4	0x14C	32	R/W	PCI Image4 Translation Address register
P_IMG_CTRL5	0x150	32	R/W	PCI Image5 Control register
P_BA5	0x024 and 0x154	32	R/W	PCI Image5 Base Address register
P_AM5	0x158	32	R/W	PCI Image5 Address Mask register
P_TA5	0x15C	32	R/W	PCI Image5 Translation Address register
P_ERR_CS	0x160	32	R/W	PCI Error Control and Status register
P_ERR_ADDR	0x164	32	R	PCI Erroneous Address register
P_ERR_DATA	0x168	32	R	PCI Erroneous Data register
WB_CONF_SPC_BAR (Base for WISHBONE bus)	0x180	32	R	WISHBONE Configuration Space Base Address
W_IMG_CTRL1	0x184	32	R/W	WISHBONE Image1 Control register
W_BA1	0x188	32	R/W	WISHBONE Image1 Base Address register
W_AM1	0x18C	32	R/W	WISHBONE Image1 Address Mask register
W_TA1	0x190	32	R/W	WISHBONE Image1 Translation Address register

Name	Address	Width	Access	Description
W_IMG_CTRL2	0x194	32	R/W	WISHBONE Image2 Control register
W_BA2	0x198	32	R/W	WISHBONE Image2 Base Address register
W_AM2	0x19C	32	R/W	WISHBONE Image2 Address Mask register
W_TA2	0x1A0	32	R/W	WISHBONE Image2 Translation Address register
W_IMG_CTRL3	0x1A4	32	R/W	WISHBONE Image3 Control register
W_BA3	0x1A8	32	R/W	WISHBONE Image3 Base Address register
W_AM3	0x1AC	32	R/W	WISHBONE Image3 Address Mask register
W_TA3	0x1B0	32	R/W	WISHBONE Image3 Translation Address register
W_IMG_CTRL4	0x1B4	32	R/W	WISHBONE Image4 Control register
W_BA4	0x1B8	32	R/W	WISHBONE Image4 Base Address register
W_AM4	0x1BC	32	R/W	WISHBONE Image4 Address Mask register
W_TA4	0x1C0	32	R/W	WISHBONE Image4 Translation Address register
W_IMG_CTRL5	0x1C4	32	R/W	WISHBONE Image5 Control register
W_BA5	0x1C8	32	R/W	WISHBONE Image5 Base Address register
W_AM5	0x1CC	32	R/W	WISHBONE Image5 Address Mask register
W_TA5	0x1D0	32	R/W	WISHBONE Image5 Translation Address register
W_ERR_CS	0x1D4	32	R/W	WISHBONE Error Control and Status register
W_ERR_ADDR	0x1D8	32	R	WISHBONE Erroneous Address register
W_ERR_DATA	0x1DC	32	R	WISHBONE Erroneous Data register
CNF_ADDR	0x1E0	32	R/W	Configuration Cycle Generation Address register
CNF_DATA	0x1E4	32	R/W	Configuration Cycle Generation Data register

Name	Address	Width	Access	Description
INT_ACK	0x1E8	32	R	Interrupt Acknowledge register
ICR	0x1EC	32	R/W	Interrupt Control register
ISR	0x1F0	32	R/W	Interrupt Status register

* – For GUEST implementation of the bridge, only P_BA0 is implemented, since image 0 is used to access the configuration registers. In HOST implementation, image 0 can be used to access WISHBONE address space. If so, all PCI Target image 0 registers are implemented.

4.1.1. WISHBONE Slave Unit Control & Status

The registers of the WISHBONE slave unit start at offset 0x180.

4.1.1.1 WISHBONE Configuration Space BAR

Bit #	Access	Reset	Description
32	R	*	This register stores the base address to access core registers from the WISHBONE bus. It is read only.

* – Value at reset is defined before implementation in parameter file

Table 4.1: WISHBONE configuration space Base Address register

Register layout:

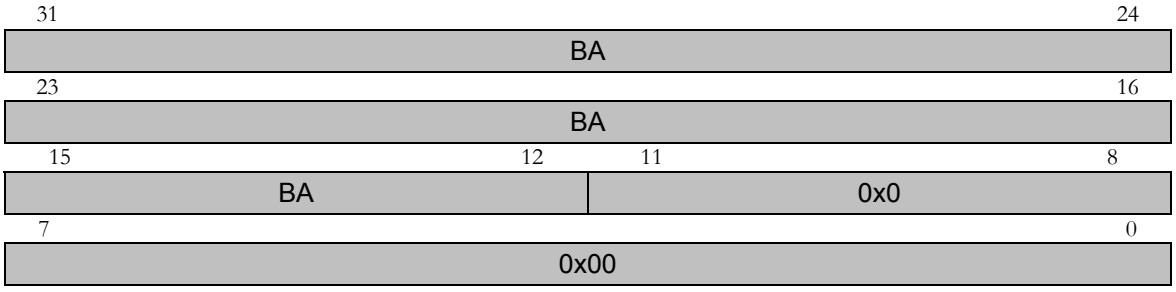


Figure 4.1: WISHBONE configuration space Base Address register layout

The register is read only. Bits 31 – 12 define the WISHBONE configuration space base address. Bits 11 – 0 are always 0 because the minimum image size is 4KB.

4.1.1.2 WISHBONE Image Control and Address Registers

Five configurable WISHBONE slave images can be implemented. Each of these images implements its own set of registers. Image Control and Address registers are the same for all five images.

Image Control registers: W_IMG_CTRL1 - W_IMG_CTRL5

Bit #	Access	Reset	Description
32	RW	0x00000000	The register value controls the WISHBONE slave unit behaviour when an image is selected and enabled.

Table 4.2: WISHBONE Image Control register

Register layout:

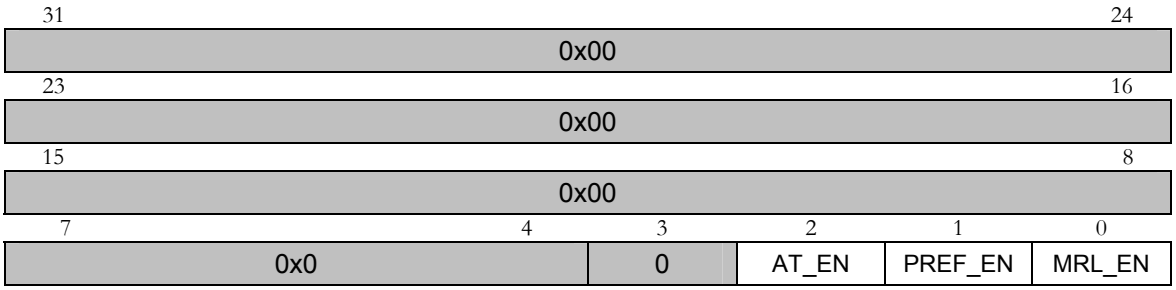


Figure 4.2: WISHBONE Image Control register layout

Bit descriptions:

Bit #	Name	Description
31 – 3	N/A	Not used
2	Address Translation Enable	If this bit is set, address translation for the corresponding image is enabled.
1	Prefetch enable	This bit marks address space occupied by an image as prefetchable. Don't care for I/O mapped images.
0	Memory Read Line Enable	When set, this bit enables the usage of memory access optimizing commands for WISHBONE Slave unit serial block read requests. If the prefetch enable bit is also set, read will be performed using Memory Read Multiple command, otherwise the Memory Read Line command will be used. If this bit is cleared, the PCI Master always uses the Memory Read command for serial block read requests.

Table 4.3: WISHBONE Image Control register bit descriptions

Base Address registers: W_BA1- W_BA5

Width	Access	Reset	Description
32	RW	0x00000000	Each of these registers stores the base address of corresponding WISHBONE image and the information on the type of address space it is mapped to.

Table 4.4: WISHBONE Base Address register

Register layout:

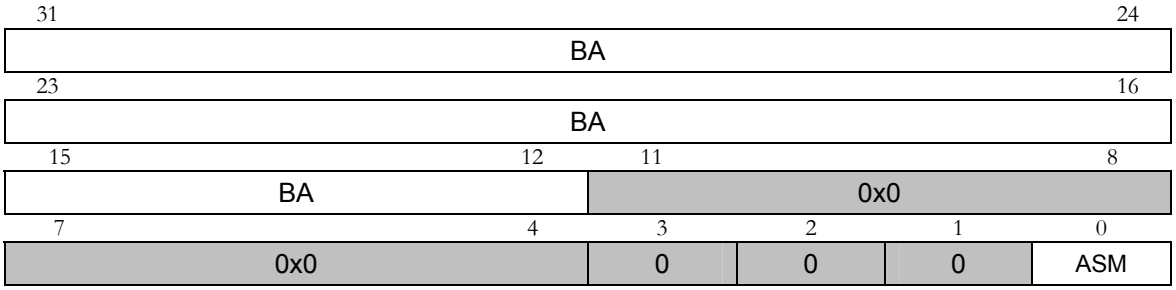


Figure 4.3: WISHBONE Base Address register layout

Bit descriptions:

Bit #	Name	Description
31 – 12	Base Address	Image's base address.
11-1	N/A	Because the minimum block size is 4KB, this field is reserved.
0	Address Space Mapping	This bit defines to which address space an image is mapped: 0 – Memory space mapping 1 – I/O space mapping

Table 4.5: WISHBONE Base Address register bit descriptions

Address Mask registers: W_AM1 – W_AM5

Width	Access	Reset	Description
32	RW	0x00000000	The Address Mask selects the bits of the incoming WISHBONE address that are compared to the value in the corresponding W_BA register when decoding an access to the WISHBONE Slave unit. It also selects the bits of the incoming WISHBONE address that are replaced with the value in the corresponding W_TA register before accessing the PCI bus, if address translation is enabled.

Table 4.6: WISHBONE Address Mask register

Register layout:

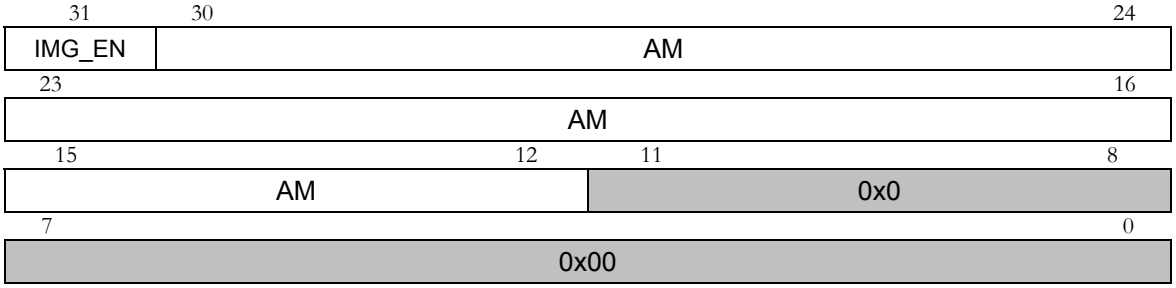


Figure 4.4: WISHBONE Address Mask register layout

Bit descriptions:

Bit #	Name	Description
31	Image Enable & Address Mask (31)	This bit must be set to enable an image. If 0, the corresponding WISHBONE image is disabled.
30 – 12	Address Mask	The remainder of the Address Mask.
11-0	N/A	Because the minimum block size is 4KB, this field is always 0x000 (the twelve lower address lines are never compared with the W_BAx register value).

Table 4.7: WISHBONE Address Mask register bit descriptions

Translation Address registers: W_TA1 – W_TA5

Width	Access	Reset	Description
32	RW	0x00000000	If address translation is enabled, the incoming WISHBONE address bits selected with the value in the corresponding W_AM register are replaced with the value in this register.

Table 4.8: WISHBONE Translation Address register

Register layout:

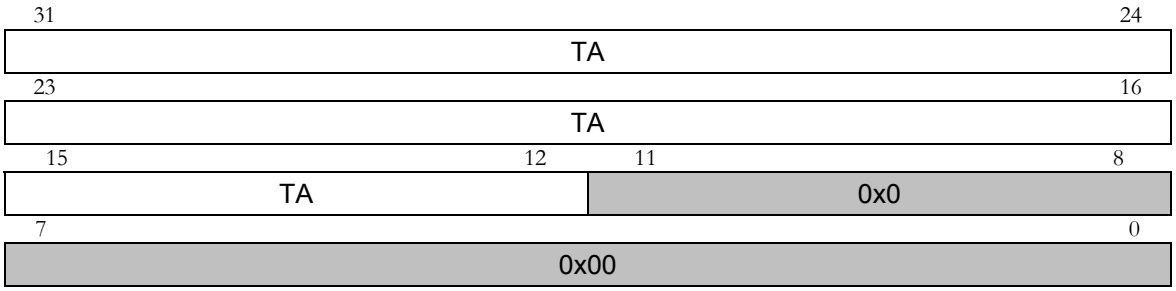


Figure 4.5: WISHBONE Translation Address register layout

Bit descriptions:

Bit #	Name	Description
31 – 12	Translation Address	This register value is used when address translation is enabled.
11-0	N/A	Because the minimum block size is 4KB, this field is always 0x000 (the twelve lower address lines are never replaced).

Table 4.9: WISHBONE Translation Address register bit descriptions

4.1.2. PCI Target Unit Control & Status

Guest bridge implementation always provides R/W access to Configuration space by configuring the Base Address 0 register. Other PCI agents are responsible for this by performing a Type 0 configuration cycle. Host bridge implementation can provide read-only access to Configuration Space or can be set not to do that at all. This way, all six PCI Base Addresses can be used for accessing the WISHBONE address space (see PCI IP Core Design document and chapter A.1, which images are implemented in current design).

31	24	23	16	15	8	7	0	
Device ID				Vendor ID				00h
Status				Command				04h
Class Code							Revision ID	08h
BIST		Header Type		Latency Timer			Cache Line Size	0Ch
Base Address Register #0								10h
Base Address Register #1								14h
Base Address Register #2								18h
Base Address Register #3								1Ch
Base Address Register #4								20h
Base Address Register #5								24h
CardBus CIS Pointer								28h
Subsystem ID				Subsystem Vendor ID				2Ch
Expansion ROM Base Address								30h
Reserved							Cap List Pointer	34h
Reserved								38h
Max_Lat		Min_Gnt		Interrupt Pin			Interrupt Line	3Ch

Figure 4.6: PCI Configuration Space Header (Header type 00h)

All PCI-compliant devices must support Vendor ID, Device ID, Command, Status, Revision ID, Class Code, and Header Type. The Header Type is type 00h, which defines the header structure of Figure 4.6.

The configuration space header used for device identification includes the following:

- **Vendor ID:** This field identifies the manufacturer of the device. To ensure uniqueness, the PCI SIG allocates valid vendor identifiers. 0xFFFFh is an invalid value for the Vendor ID.
- **Device ID:** This field identifies the particular device. It is allocated by the vendor.
- **Revision ID:** This register specifies a device specific revision identifier whose value is chosen by the vendor. An acceptable value is zero. This field should be viewed as a vendor-defined extension to the Device ID.
- **Header Type:** This byte identifies the layout of the second part of the predefined header (beginning at byte 10h in configuration space) and also whether or not the device contains multiple functions. Bit 7 in this register is used to identify a multi-functional device. If the bit is 0, the device is single-functional. If the bit is 1, it has multiple functions. Bits 6 through 0 identify the layout of the second part of the predefined header.

- **Class Code:** The Class Code register is read only. It is used to identify the generic function of the device and, in some cases, a specific register-level programming interface (see the *PCI 2.2 Specification* for detailed description).

The Command register serves device control functions. When 0, the device is logically disconnected from the bus (except for configuration accesses). The following table shows bit descriptions.

Bit #	Implemented	Description
15 – 10	–	Reserved
9	NO	Fast Back-to-Back Enable. This optional Read/Write bit controls whether or not a master can do fast back-to-back transactions to different devices. A value of 1 indicates that the master is allowed to generate fast back-to-back transactions to different agents. A value of 0 means that fast back-to-back transactions are allowed only to the same agent. The state after RST**#** is 0.
8	√	SERR# enable. A value of 0 disables the SERR**#** driver, a value of 1 enables it. The state of this bit after RST**#** is 0. Address parity errors are reported only if this bit and bit 6 are 1.
7	NO	Stepping control. This bit is used to control whether or not a device does address/data stepping. Devices that never do stepping must hardwire this bit to 0.
6	√	Parity Error Response. This bit controls the device's response to parity errors. If set, the device must take its normal action when a parity error is detected. If the bit is 0, the device sets its detected parity error status bit (bit 15 in the Status register) when an error is detected but does not assert PERR**#** and continues normal operation. The state after RST**#** is 0.
5	NO	VGA Palette Snoop. This bit controls how VGA compatible devices and graphics devices handle access to the VGA Palette registers. When this bit is 1, palette snooping is enabled (i.e. the device does not respond to Palette Register Write cycles and snoops the data).
4	NO	Memory Write and Invalidate. This is an enable bit for using the Memory Write and Invalidate command. When this bit is 1, masters may generate the command. When it is 0, Memory Write must be used instead. The state after RST**#** is 0.
3	NO	Special cycles. Controls a device's action on Special Cycle operations. A value of 0 causes the device to ignore all Special Cycle operations. A value of 1 allows the device to monitor Special Cycle operations. The state after RST# is 0.
2	√	Bus master. This bit controls the device's ability to act as a master on the PCI bus. A value of 0 disables the device from generating PCI accesses. A value of 1 allows the device to

Bit #	Implemented	Description
		behave as a bus master. The state after RST# is 0.
1	√	Memory space. This bit controls the response to memory space access. A value of 0 disables the device response. A value of 1 allows responding to memory space access. The state after RST# is 0.
0	√	I/O space. This bit controls the response to I/O space access. A value of 0 disables the device response. A value of 1 allows the device to respond to I/O space access. The state after RST# is 0.

Table 4.10: Command register of PCI configuration header

The Status register notes the device status. Reserved bits are read only and return 0 after reading. A 1 bit is reset whenever a 1 is written to a corresponding bit location. The following table provides a description of the corresponding bits.

Bit descriptions:

Bit #	Implemented	Description
15	√	Detected Parity Error. The device must set this bit whenever it detects a parity error, even if parity error handling is disabled (as controlled by bit 6 in the Command register).
14	√	Signaled **System Error**. This bit must be set whenever the device asserts SERR#.
13	√	Received **Master Abort**. A master device must set this bit whenever its transaction (except for special cycles) is terminated with **Master Abort**. All master devices must implement this bit.
12	√	Received **Target Abort**. A master device must set this bit whenever its transaction is terminated with **Target Abort**.
11	√	Signalled **Target Abort**. A target device must set this bit whenever it terminates a transaction with **Target Abort**.
10 – 9	√	DEVSEL timing: 00 – fast; 01 – medium; 10 – slow. These bits are read-only and must indicate the slowest time that a device needs to assert DEVSEL# for any bus command, except Configuration Read and Configuration Write.
8	√	Master Data Parity Error. This bit is implemented by bus masters only. It is set when three conditions are met: 1) The bus agent asserted PERR# itself (on a Read cycle) or observed PERR# asserted (on a Write cycle). 2) The agent setting the bit acted as the bus master for the operation during which the error occurred. 3) The parity error response bit (Command register) is set.

Bit #	Implemented	Description
7	√	Fast Back-to-Back Capable. This optional read only bit indicates whether or not the target is capable of accepting fast back-to-back transactions when the transactions do not refer to the same agent.
6	–	Reserved
5	√	66 MHz capable. This optional read only bit indicates whether or not this device is capable of running at 66 MHz. A value of 1 indicates that the device is 66 MHz capable.
4	NO	List of compatibilities. A value of zero indicates that no new capabilities' linked list is available. A value of one indicates that the value read at offset 34h is a pointer in configuration space to a linked list of new capabilities.
3 – 0	-	**Reserved**

Table 4.11: Status register of PCI configuration header

The following descriptions include only miscellaneous (device independent), already implemented registers:

- The **Cache Line Size register** is used by WISHBONE Slave and PCI Target units for prefetched read transactions. Valid values for this register are multiples of 4 (including 1). If invalid value is written (including 0), then the value of 1 is assumed by both units and burst reads are not performed.
- The **Latency Timer register** specifies the timer value in units of PCI bus clocks. After RST#, the register value is 0.
- The **Interrupt Line register** tells to which input of the system interrupt controller(s) the device's interrupt pin is connected (the *Design Document* describes in detail how it is implemented).
- The **Interrupt Pin register** tells which interrupt pin the device uses. A value of 1 corresponds to INTA# and so on. The values from 05h to FFh are reserved.
- There are **6 Base Address registers** in Configuration space Header. This registers can be accessed in the PCI part of the Configuration space also. Each one of them consists of a 28-bit base address for MEMORY mapping or a 30-bit base address for I/O mapping. Other bits are control bits and described in the following table.

Bit descriptions:

Bit #	Description
31 – 4	Base address (only the 20 MS bits can be implemented as R/W in the PCI Bridge core)
3	Prefetchable
2 –1	Type: 00 – 32-bit address space; 01 – reserved; 10 – 64-bit address space; 11 –

Bit #	Description
	reserved
0	Memory space indicator = '0' (always for MEMORY mapped space)!!!

Table 4.12: Base Address register of PCI configuration header for memory mapped space

Bit descriptions:

Bit #	Description
31 – 2	Base address (only the 20 MS bits can be implemented as R/W in the PCI Bridge core)
1	Reserved
0	I/O space indicator = '1' (always for I/O mapped space)

Table 4.13: Base Address register of PCI configuration header for I/O mapped space

4.1.2.1 PCI Image Control and Address Registers

There are six possible configurable PCI target images. Each of these images implements its own set of registers.

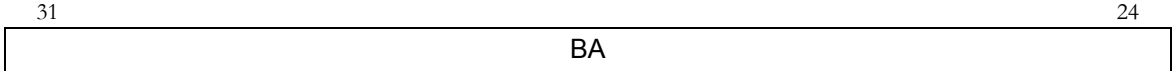
The only exception is the set of 4 PCI Image0 Control and Address registers, which is implemented only when the PCI bridge is implemented as HOST and Image0 is used to access WB bus (see Table 4.15, Table 4.17, Table 4.19 and Table 4.21). Otherwise, there are five possible configurable PCI target images (PCI Image1 – PCI Image5), while PCI Image0 Base Address register (P_BA0) is implemented and used for access to the entire Configuration Space (see Table 4.14 and Figure 4.7). The other 3 registers are not implemented and therefore cannot be written to (see also Configuration Space Access for Host Bus Bridges and Addressing and Images of the PCI Target Unit).

Base Address Registers: P_BA0

Width	Access	Reset	Description
32	RW	0x00000000	This register stores the base address for accessing core registers from the PCI bus.

Table 4.14: PCI Image0 Base Address register

Register layout:



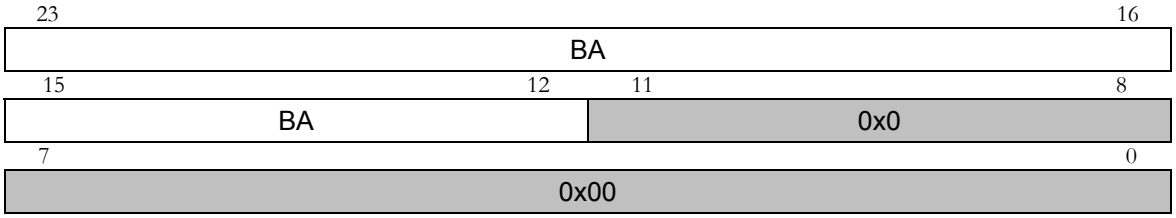


Figure 4.7: PCI Image0 Base Address register layout – Image0 used for accessing the PCI Configuration Space

Image Control registers: P_IMG_CTRL0 (P_IMG_CTRL1) – P_IMG_CTRL5

Width	Access	Reset	Description
32	RW	0x00000000	The register value controls the PCI target unit behavior when an image is selected and enabled.

Table 4.15: PCI Image Control Register

Register layout:

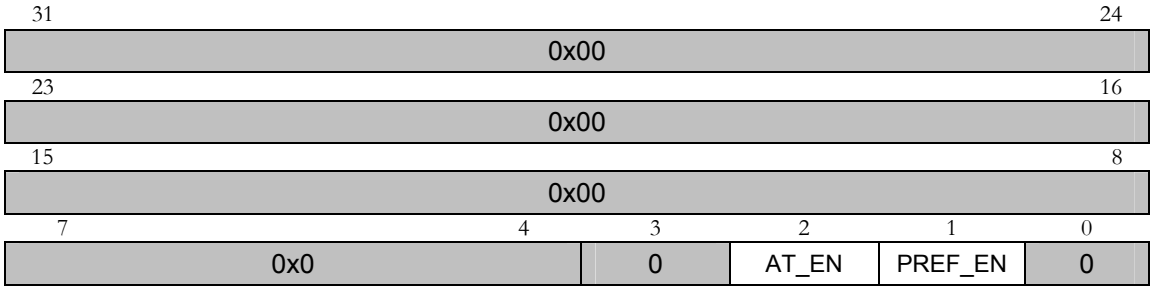


Figure 4.8: PCI Image Control register layout

Bit descriptions:

Bit #	Name	Description
31 – 3	N/A	Not used
2	Address Translation Enable	If this bit is set, address translation for the corresponding image is enabled.
1	Pre-fetch enable	This bit marks address space occupied by an image as prefetchable.
0	N/A	Not used

Table 4.16: PCI Image Control Register bit descriptions

Base Address Registers: P_BA0 (P_BA1) - P_BA5

Width	Access	Reset	Description
32	RW	0x00000000	The register value holds the PCI bus base address of an image.

Table 4.17: PCI Base Address register

Register layout:

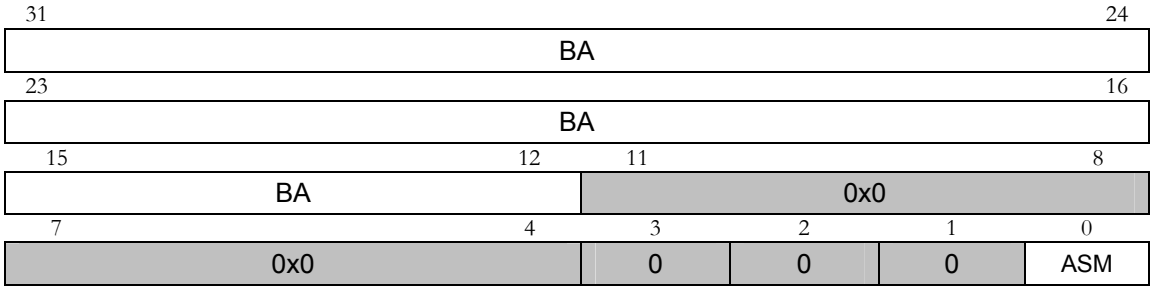


Figure 4.9: PCI Base Address Register Layout

Bit descriptions:

Bit #	Name	Description
31 – 12	Base Address	Image base address. The corresponding Address Mask register selects bits from this field that are compared to the incoming PCI address.
11-1	N/A	Because the minimum block size is 4KB, this field is reserved.
0	Address Space Mapping	This bit defines to which address space an image maps on the PCI bus. Predefined value can be changed later for HOST bridges. Predefined value can NOT be changed for GUEST bridges (see Addressing and Images of the PCI Target Unit). 0 – Memory space mapping 1 – I/O space mapping

Table 4.18: PCI Base Address register bit descriptions

Address Mask registers: P_AM0 (P_AM1) – P_AM5

Width	Access	Reset	Description
32	RW	0x00000000	The Address Mask selects the bits of the incoming PCI address that are compared to the value in the corresponding P_BA register when decoding an access to the PCI Target unit. It also selects the bits of the incoming PCI address that are replaced with the value in the corresponding P_TA register before accessing the WISHBONE bus, if address translation is enabled.

Table 4.19: PCI Address Mask register

Register layout:

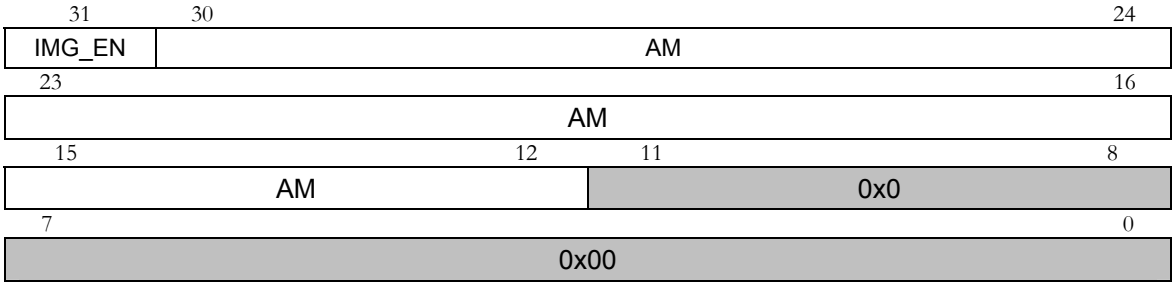


Figure 4.10: PCI Address Mask register layout

Bit descriptions:

Bit #	Name	Description
31	Image Enable & Address Mask(31)	This bit must be set for an image to be enabled. If the bit is 0, the corresponding image is not enabled.
30 – 12	Address Mask	Remainder of the address mask.
11-0	N/A	Because the minimum block size is 4KB, this field is always 0x000 (the twelve lower address lines are never compared with the BA register value).

Table 4.20: PCI Address Mask register bit descriptions

Translation Address registers: P_TA0 (P_TA1) – P_TA5

Width	Access	Reset	Description
32	RW	0x00000000	If address translation is enabled, compared address lines from the PCI bus (specified with AM value) are replaced by corresponding values in this register for WISHBONE bus accesses.

Table 4.21: PCI Translation Address register

Register layout:

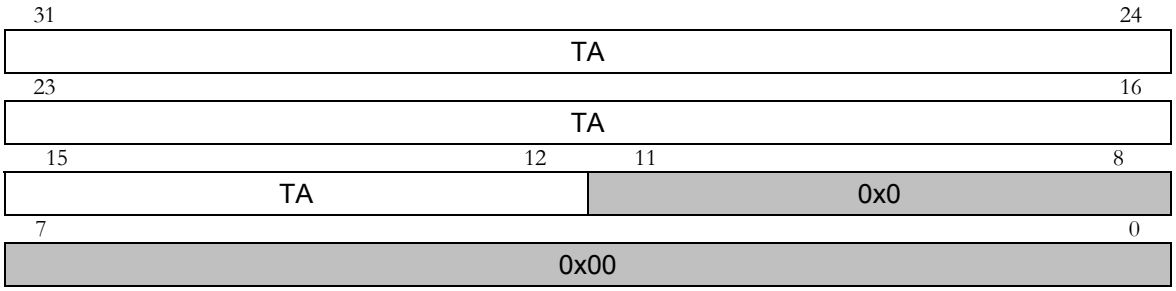


Figure 4.11: PCI Translation Address register layout

Bits descriptions:

Bit #	Name	Description
31 – 12	Translation Address	This register value is used when address translation is enabled.
11-0	N/A	Because the minimum block size is 4KB, this field is always 0x000 (the twelve lower address lines are never replaced).

Table 4.22: PCI Translation Address register bit descriptions

4.1.3. Reporting Registers

Error Reporting registers are provided because of Posted Write cycles, which are always acknowledged on the WISHBONE bus before they actually complete on the PCI bus, and vice-versa, so errors detected on PCI or WISHBONE buses cannot be reported back to WISHBONE master or PCI initiator using the standard bus protocol.

4.1.3.1 WISHBONE Slave Unit Error Reporting Registers

WISHBONE Error Control and Status register: W_ERR_CS

Width	Access	Reset	Description
32	RW	0x00000000	Part of this register is used for controlling the Error Reporting mechanism, another part for reporting statuses and additional information about an error that occurred during the completion of a Posted Write cycle on the PCI bus.

Table 4.23: WISHBONE Error Control and Status register

Register layout:

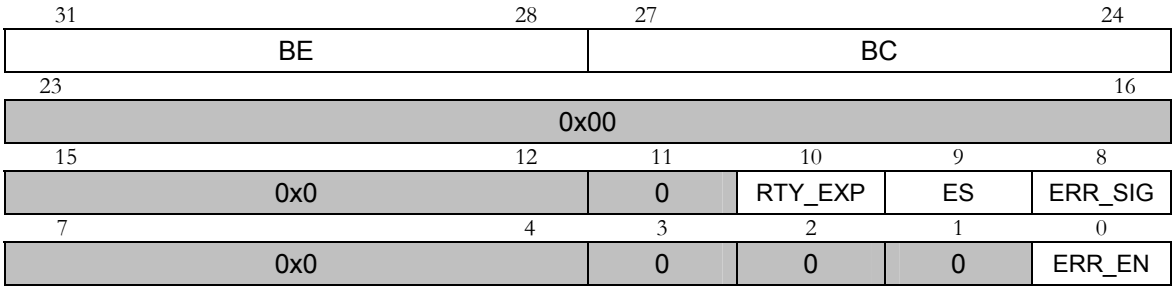


Figure 4.12: WISHBONE Error Control and Status register layout

Bit descriptions:

Bit #	Name	Description
31 – 28	Byte Enables	The field value reports the state of BE# signals used in the Posted Write cycle that terminated with an error.
27-24	Bus Command	This field value reports a bus command used for the Posted Write cycle that terminated with an error.
16 – 11	N/A	Not used
10	Retry Counter Expired	THIS BIT IS RESERVED FOR FUTURE USE! It's function is to report that a Posted Write cycle has been retried *MAX_RETRY* times. The *PCI Local Bus Specification* requires any transaction terminated with **Retry** to be repeated unconditionally, so this bit is not implemented at this time.
9	Error Source	The ES bit indicates that the master terminated the transaction with **Master Abort**. Software can distinguish between two kinds of **Master Abort** terminations the PCI master module performs: If the RTY_EXP bit is cleared, **Master Abort** was performed because no target claimed the transaction; if the RTY_EXP is set, the target signaled too many **Retry** terminations. A cleared ES bit indicates that the target of the transaction signaled **Target Abort**.
8	Error Signaled	If set, this bit indicates that an error has been reported. This bit is cleared by writing 1 to its location.
7-1	N/A	Not used
0	Error Enable	Setting this bit enables the Error Reporting mechanism. Clearing this bit means that Error Reporting is not performed.

Table 4.24: WISHBONE Error Control and Status register bit descriptions

WISHBONE Erroneous Address Register: W_ERR_ADDR

Width	Access	Reset	Description
32	R	0x00000000	When Error Reporting is enabled and an error is signaled, this register stores the address of the transaction on the PCI bus that caused an error.

Table 4.25: WISHBONE Erroneous Address register

WISHBONE Erroneous Data: W_ERR_DATA

Width	Access	Reset	Description
32	R	0x00000000	When Error Reporting is enabled and an error is signaled, this register stores data of the transaction on the PCI bus that caused an error.

Table 4.26: WISHBONE Erroneous Data register

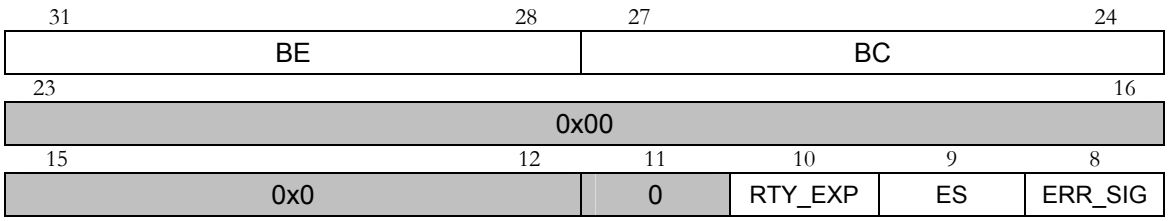
4.1.3.2 PCI Target Unit Error Reporting Registers

PCI Error Control and Status register: P_ERR_CS

Width	Access	Reset	Description
32	RW	0x00000000	Part of this register is used for controlling the Error Reporting mechanism, another part for reporting statuses and additional information about an error that occurred during the completion of a Posted Write cycle on the WISHBONE bus.

Table 4.27: PCI Error Control and Status register

Register layout:



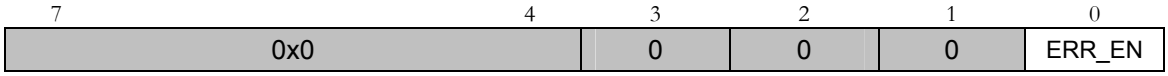


Figure 4.13: PCI Error Control and Status register layout

Bit descriptions:

Bit #	Name	Description
31 – 28	Byte Enables	This field value reports the state of byte enable signals SEL_O(3:0) used in the Posted Write cycle that terminated with an error on the WB bus.
27-24	Bus Command	This field value reports a bus command from the PCI bus used for a Posted Write cycle that terminated with an error on the WB bus.
16 – 11	N/A	Not used
10	Retry Counter Expired	This bit reports that a Posted Write cycle has been retried *MAX_RETRY* times or that there was no response on the WISHBONE bus for *MAX_RETRY* times (internal **Retry** is generated if the WISHBONE slave does not respond for 8 cycles).
9	Error Source	The ES bit indicates that the WISHBONE Master Module of the PCI Target Unit stopped (terminated) the write transaction. The WISHBONE slave signaled too many **Retry** terminations. In this case, the RTY_EXP bit is also set. A cleared ES bit indicates that the WISHBONE Master Module of the PCI Target Unit was not able to continue the write transaction because of the WISHBONE slave: If the RTY_EXP bit is cleared, the WISHBONE slave signaled an **Error** termination; if the RTY_EXP bit is set, the WISHBONE slave did not respond to the initiated transaction.
8	Error Signaled	If set, this bit indicates that an error has been reported. The bit is cleared by writing 1 to its location.
7-1	N/A	Not used
0	Error Enable	Setting this bit enables the Error Reporting mechanism. Clearing this bit means that Error Reporting will not be performed – the transaction that caused an error is discarded, other transactions continue normally.

Table 4.28: PCI Error Control and Status register Bit Descriptions

PCI Erroneous Address Register: P _ERR_ADDR

Width	Access	Reset	Description
32	R	0x00000000	When Error Reporting is enabled and an error is signaled, this register stores the address of the transaction on the WISHBONE bus that caused an error.

Table 4.29: PCI Erroneous Address register

PCI Erroneous Data: P_ERR_DATA

Width	Access	Reset	Description
32	R	0x00000000	When Error Reporting is enabled and an error is signaled, this register stores data of the transaction on the WISHBONE bus that caused an error.

Table 4.30: PCI Erroneous Data Register

4.1.3.3 Configuration Cycle Generation Registers

Two registers are provided for generating configuration cycles on the PCI bus. The WISHBONE master initiates a configuration cycle in two steps:

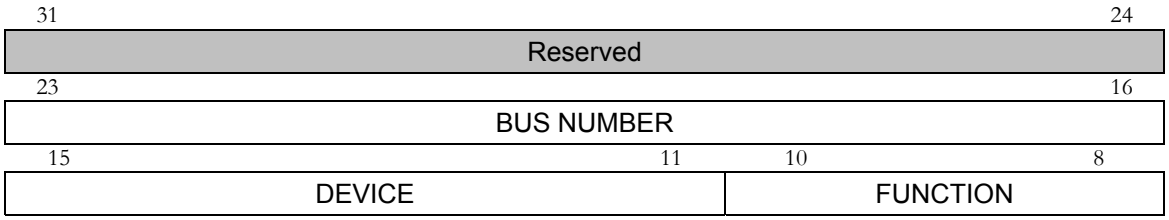
1. It writes the appropriate value in the CNF_ADDR register and
2. Reads from or writes to the CNF_DATA register to generate a Configuration Read or Write cycle respectively.

Configuration address: CNF_ADDR

Width	Access	Reset	Description
32	RW	0x00000000	This register stores all information needed to drive address lines during the Address phase of a configuration cycle.

Table 4.31: Configuration Address register

Register layout:



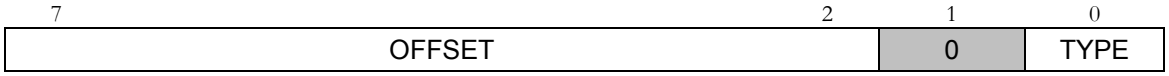


Figure 4.14: Configuration Address register layout

Bit descriptions:

Bit #	Name	Description
31 – 24	N/A	A value in this field is ignored for any kind and type of configuration cycle.
23 – 16	Bus number	This field holds a bus number on which a target of the configuration space access resides. It is only used in Type 1 configuration cycles (TYPE bit = 1).
15 – 11	Device number	The value in this field represents a device number. This field is driven directly to AD(15:11) lines during the Address phase for Type1 (TYPE = 1) configuration cycle and is decoded for Type0 configuration cycles (See Table 3.1 for Device number decoding).
10 – 8	Function number	The value in this field is a function number for multifunctional devices.
7 –2	Register number	This field holds the register offset for a device addressed with configuration cycle.
1	N/A	Not used—always 0
0	Type	Type of configuration cycle (0 – Type 0, 1 – Type 1)

Table 4.32: Configuration Address register bit descriptions

Configuration data: CNF_DATA

A Read cycle from or a Write cycle to this register will perform a configuration cycle on the PCI bus using information written to the CNF_ADDR register.

Width	Access	Reset	Description
32	RW	0x00000000	This register stores Read or Write data for configuration cycles.

Table 4.33: Configuration Data Register

4.1.3.4 Interrupt Acknowledge Cycle Generation Register

A Read cycle from the INT_ACK register generates an Interrupt Acknowledge cycle on the PCI bus.

Width	Access	Reset	Description
32	R	0x00000000	This register stores interrupt vector data returned during an Interrupt Acknowledge cycle.

Table 4.34: Interrupt Acknowledge register

4.1.4. Interrupt Control & Status Registers

Interrupt Control register: ICR

Width	Access	Reset	Description
32	RW	0x00000000	This register is used to enable/disable the generation of interrupt requests from various sources.

Table 4.35: Interrupt Control register

Register layout:

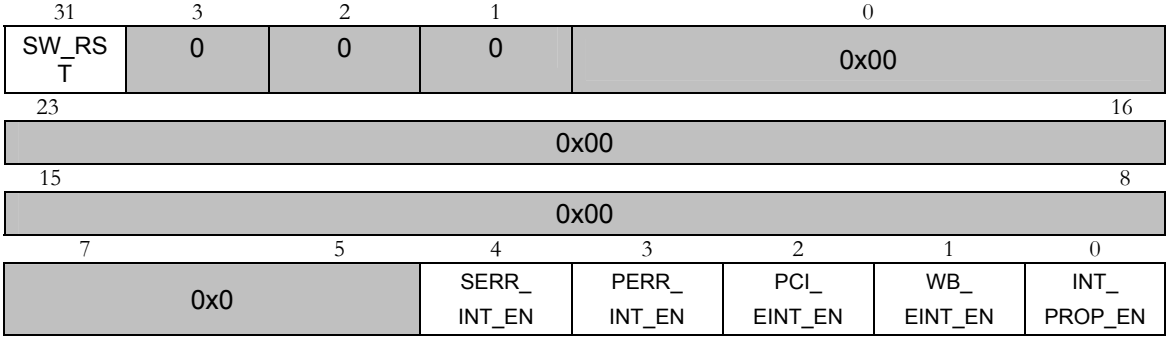


Figure 4.15: Interrupt Control register layout

Bit descriptions:

Bit #	Name	Description
0	Interrupt Propagation Enable	For Guest bridge implementation this bit indicates that INT_I line assertion on the WISHBONE bus will generate an interrupt request on the PCI bus through an assertion of the INTA# pin. For Host bridge implementation this bit indicates that an assertion of the INTA# pin on the PCI bus will generate an interrupt request on the WISHBONE bus through an assertion of the INT_O pin.
1	WISHBONE Error	If set, this bit enables interrupt request generation when an

Bit #	Name	Description
	Interrupt Enable	error is REPORTED during the execution of Posted Write cycles through the WISHBONE slave unit. A cleared bit disables these interrupts but does not disable Error Reporting (see bits 0 and 8 of WB Error Control and Status register – W_ERR_CS).**
2	PCI Error Interrupt Enable	If set, this bit enables interrupt request generation when an error is REPORTED during the execution of Posted Write cycles through the PCI target unit. A cleared bit disables these interrupts but does not disable Error Reporting (see bits 0 and 8 of PCI Error Control and Status register – P_ERR_CS).**
3	Parity Error Interrupt enable	This bit enables/disables the generation of interrupt requests when a parity error is detected by the PCI master module. This interrupt is meaningful on Host Bridge Implementation only.*
4	System Error Interrupt Enable	This bit enables/disables the generation of interrupt requests when a system error (address parity error) is detected by the PCI master module. This interrupt is decisive on Host Bridge Implementation only.*
31	Software Reset	Setting this bit causes software initiated reset. Host bridge implementation uses this bit to reset the PCI bus, Guest implementation uses it to reset the WISHBONE bus.

* Interrupt triggering upon PERR# and SERR# detection for Guest Implementation has no meaning because Guest Implementation triggers interrupts on the PCI bus. An agent that is responsible for routing interrupts to a host processor may trigger an interrupt when one of these errors is detected.

** For reporting Error Interrupt, appropriate Error Reporting Enable bit must be SET (bit 0 of P_ERR_CS and W_ERR_CS registers) besides Error Interrupt Enable bit (see also chapters 4.1.3.1 and 4.1.3.2).

Table 4.36: Interrupt Control Register bit descriptions

Interrupt Status Register: ISR

Width	Access	Reset	Description
32	RW	0x00000000	This register is used to determine and clear the source of the pending interrupt request.

Table 4.37: Interrupt Status register

Register layout:

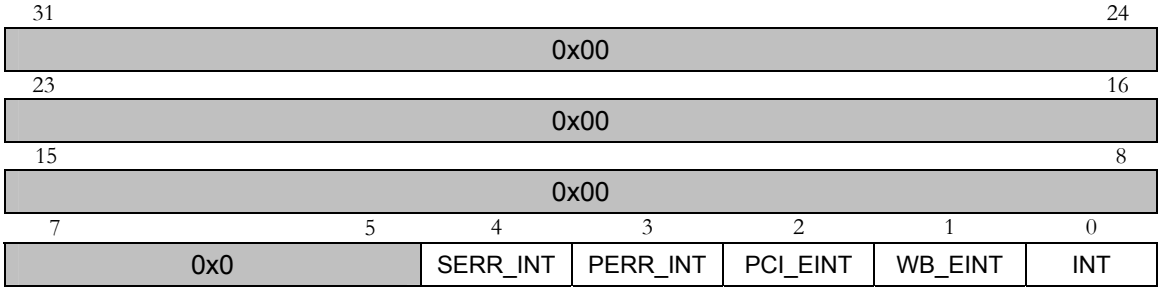


Figure 4.16: Interrupt Status register layout

Bit descriptions:

Bit #	Name	Description
0	Interrupt	For Guest bridge implementation this bit indicates that an INT_I input on the WISHBONE bus has been asserted and propagated to the INTA# pin on the PCI bus. This is to say that some device on the WISHBONE bus generated an interrupt request to the host processor. For Host Bridge Implementation this bit indicates that the INTA# pin on the PCI bus has been asserted and propagated to the INT_O pin on the WISHBONE bus. This means that some device residing on the PCI bus generated an interrupt request to the host processor.
1	WISHBONE Error Interrupt	If set, this bit indicates an interrupt request from the Error Reporting mechanism, which detected an error during the execution of a Posted Write cycle through the WISHBONE slave unit. Only reported error causes this interrupt.**
2	PCI Error Interrupt	If set, this bit indicates an interrupt request from the Error Reporting mechanism, which detected an error during the execution of a Posted Write cycle through the PCI target unit. Only reported error causes this interrupt.**
3	Parity Error Interrupt	This bit indicates that an interrupt request has been generated due to a Parity Error on the PCI bus. This interrupt has meaning only on Host Bridge Implementation.*
4	System Error Interrupt Enable	This bit indicates that an interrupt request has been generated due to System Error (Address Parity) on the PCI bus. This interrupt has meaning only on Host Bridge Implementation.*

Table 4.38: Interrupt Status register bit descriptions

* Interrupt triggering upon PERR# and SERR# detection for Guest Implementation has no meaning because Guest Implementation triggers interrupts on the PCI bus. In Guest Implementation, these two bits will never be set.

** For reporting Error Interrupt, appropriate Error Reporting Enable bit must be SET (bit 0 of P_ERR_CS and W_ERR_CS registers) besides Error Interrupt Enable bit (see also chapters 4.1.3.1 and 4.1.3.2).

5. IO Ports

5.1. PCI Interface

The PCI interface contains all required pins.

Port	Width	Direction	Description
pci_clk_i	1	I	PCI bus CLK signal input.
pci_rst_i	1	I	PCI bus RST# input. Only used in GUEST implementation. An active state on this input is propagated to the *wb_rst_o*.
pci_rst_o	1	O	PCI bus RST# output. Only used in HOST implementation. An active state on *wb_rst_i* is propagated to this output. The value is always 0. The assertion of the RST# is controlled with *pci_rst_oe_o*.
pci_rst_oe_o	1	O	PCI bus RST# output enable. Only used in HOST implementation. The PCI Bridge never drives the RST# pin to inactive state. It enables the output driver when active reset is required.
pci_inta_i	1	I	PCI bus INTA# input. Only used in HOST implementation. Used to propagate interrupt requests from the PCI bus to the WISHBONE bus if enabled.
pci_inta_o	1	O	PCI bus INTA# output. Only used in GUEST implementation. Used to propagate interrupt requests from the WISHBONE bus to the PCI bus if enabled. The value is always 0. The assertion of interrupt request is controlled with *pci_inta_oe_o*.
pci_inta_oe_o	1	O	Only used in GUEST implementation. The PCI Bridge never drives the INTA# signal to inactive state. It enables the output driver if active INTA#

Port	Width	Direction	Description
			signal value is required.
pci_req_o	1	O	PCI REQ# output signal. Used by the PCI Master module to signal to the PCI arbiter that it needs a mastership of the PCI bus.
pci_req_oe_o	1	O	PCI REQ# output enable signal. This signal is inactive during PCI bus reset and enabled when PCI bus reset is released.
pci_gnt_i	1	I	PCI GNT# input signal. The external arbiter grants the bus to the PCI Master module when active.
pci_frame_i	1	I	PCI FRAME# input signal.
pci_frame_o	1	O	PCI FRAME# output signal.
pci_frame_oe_o	1	O	PCI FRAME# output enable signal.
pci_irdy_i	1	I	PCI IRDY# input signal.
pci_irdy_o	1	O	PCI IRDY# output signal.
pci_irdy_oe_o	1	O	PCI IRDY# output enable signal.
pci_devsel_i	1	I	PCI DEVSEL# input signal.
pci_devsel_o	1	O	PCI DEVSEL# output signal.
pci_devsel_oe_o	1	O	PCI DEVSEL# output enable signal.
pci_trdy_i	1	I	PCI TRDY# input signal.
pci_trdy_o	1	O	PCI TRDY# output signal.
pci_trdy_oe_o	1	O	PCI TRDY# output enable signal.
pci_stop_i	1	I	PCI STOP# input signal.
pci_stop_o	1	O	PCI STOP# output signal.
pci_stop_oe_o	1	O	PCI STOP# output enable signal.
pci_ad_i	32	I	PCI AD input bus.
pci_ad_o	32	O	PCI AD output bus.
pci_ad_oe_o	32	O	PCI AD bus output enable.
pci_cbe_i	4	I	PCI C/BE# input bus.
pci_cbe_o	4	O	PCI C/BE# output bus.
pci_cbe_oe_o	4	O	PCI C/BE# bus output enable.
pci_idsel_i	1	I	PCI IDSEL input signal.
pci_par_i	1	I	PCI PAR input signal.
pci_par_o	1	O	PCI PAR output signal.
pci_par_oe_o	1	O	PCI PAR output enable signal.
pci_perr_i	1	I	PCI PERR# input signal

Port	Width	Direction	Description
pci_perr_o	1	O	PCI PERR# output signal.
pci_perr_oe_o	1	O	PCI PERR# output enable signal.
pci_serr_o	1	O	PCI SERR# output signal.
pci_serr_oe_o	1	O	PCI SERR# output enable signal.
pci_cpci_hs_enum_o	1	O	CompactPCI ENUM# output signal. Optional! See Appendix A.
pci_cpci_hs_enum_oe_o	1	O	CompactPCI ENUM# output enable signal. Optional! See Appendix A.
pci_cpci_hs_led_o	1	O	CompactPCI LED# output signal. Optional! See Appendix A.
pci_cpci_hs_led_oe_o	1	O	CompactPCI LED# output enable signal. Optional! See Appendix A.
pci_cpci_hs_es_i	1	I	CompactPCI Handle Switch state input signal. Optional! See Appendix A.

Table 5.1: PCI interface

5.2. WISHBONE Slave Interface

The WISHBONE Slave interface is a WISHBONE Rev. B compliant slave interface.

Port	Width	Direction	Description
wb_clk_i	1	I	WISHBONE bus CLK_I input signal.
wb_rst_i	1	I	WISHBONE bus RST_I input signal. Only used in HOST implementation. The active value on this signal enables the PCI RST# signal via the activation of *pci_rst_oe_o* signal.
wb_rst_o	1	O	WISHBONE bus RST_O output signal. Only used in GUEST implementation. The assertion of RST# on the PCI is propagated to this signal.
wb_int_i	1	I	WISHBONE bus INT_I input signal. Only used in GUEST implementation. The active value on this signal results in INTA# assertion on the PCI bus, if enabled.
wb_int_o	1	O	WISHBONE bus INT_O signal. Only used in HOST implementation. The active value on the *pci_inta_i* propagates to this signal, if

Port	Width	Direction	Description
			enabled.
wbs_adr_i	32	I	WISHBONE ADR_I(31:0) address bus input.
wbs_dat_i	32	I	WISHBONE DAT_I(31:0) data bus input.
wbs_dat_o	32	O	WISHBONE DAT_O(31:0) data bus output.
wbs_sel_i	4	I	WISHBONE SEL_I(3:0) byte select bus input.
wbs_cyc_i	1	I	WISHBONE CYC_I input signal.
wbs_stb_i	1	I	WISHBONE STB_I input signal.
wbs_we_i	1	I	WISHBONE WE_I input signal.
wbs_cab_i	1	I	The serial Block cycle identifier input. Optional (see Appendix A and Chapter 3.3.5).
wbs_cti_i	3	I	WISHBONE Registered Feedback cycle identifier input. Optional (see Appendix A and Chapter 3.3.5).
wbs_bte_i	2	I	WISHBONE Registered Feedback burst type identifier input. Optional (see Appendix A and Chapter 3.3.5).
wbs_ack_o	1	O	WISHBONE ACK_O output signal.
wbs_rty_o	1	O	WISHBONE RTY_O output signal.
wbs_err_o	1	O	WISHBONE ERR_O output signal.

Table 5.2: WISHBONE Slave interface signals

5.3. WISHBONE Master Interface

The WISHBONE Master interface is a WISHBONE Rev. B compliant master interface. Clock, reset and interrupt pins were described in the previous section.

Port	Width	Direction	Description
wbm_adr_o	32	O	WISHBONE ADR_O(31:0) address bus output.
wbm_dat_i	32	I	WISHBONE DAT_I(31:0) data bus input.
wbm_dat_o	32	O	WISHBONE DAT_O(31:0) data bus output.
wbm_sel_o	4	O	WISHBONE SEL_O(3:0) byte select bus output.
wbm_cyc_o	1	O	WISHBONE CYC_O output signal.
wbm_stb_o	1	O	WISHBONE STB_O output signal.

Port	Width	Direction	Description
wbm_we_o	1	O	WISHBONE WE_O output signal.
wbm_cab_o	1	O	The serial Block cycle identifier output. **Obsolete!**
wbm_cti_o	3	O	WISHBONE Registered Feedback cycle type identifier.
wbm_bte_o	2	O	WISHBONE Registered Feedback burst type identifier.
wbm_ack_i	1	I	WISHBONE ACK_I input signal.
wbm_rty_i	1	I	WISHBONE RTY_I input signal.
wbm_err_i	1	I	WISHBONE ERR_I input signal.

5.4. Serial Power On Configuration Interface

The signals described in this chapter are implemented only if *PCI_SPOCI* macro is defined in the *pci_user_constants.v* file. See Appendix A for more information.

Port	Width	Direction	Description
spoci_scl_o	1	O	Serial clock output. (hardwired to 0)
spoci_scl_oe_o	1	O	Serial clock output enable.
spoci_sda_i	1	I	Serial data input.
spoci_sda_o	1	O	Serial data output. (hardwired to 0)
spoci_sda_oe_o	1	O	Serial data output enable.

6. Waveforms

6.1. Wishbone Slave Unit

This section describes basic waveforms of various accesses to the core's configuration space and mapped PCI address space. Waveforms supplied have only informational purpose at this time.

6.1.1. WISHBONE Configuration Accesses

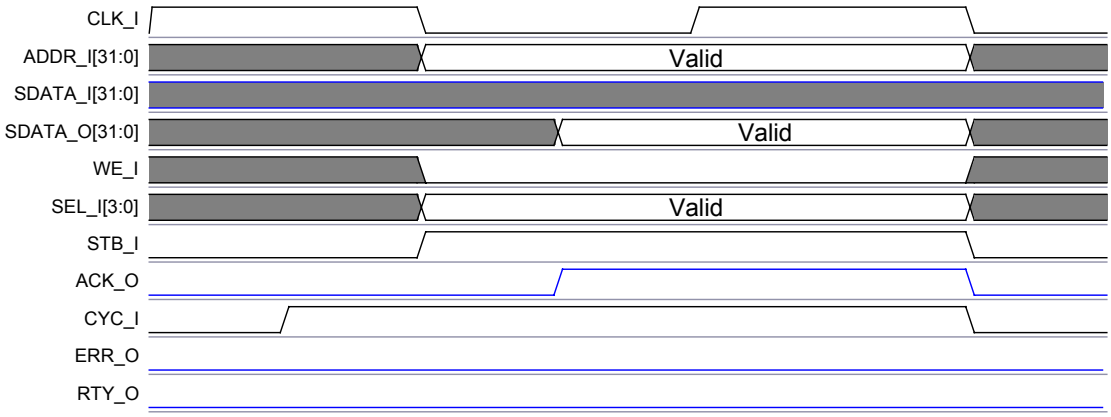


Figure 6.1: WISHBONE configuration Read cycle

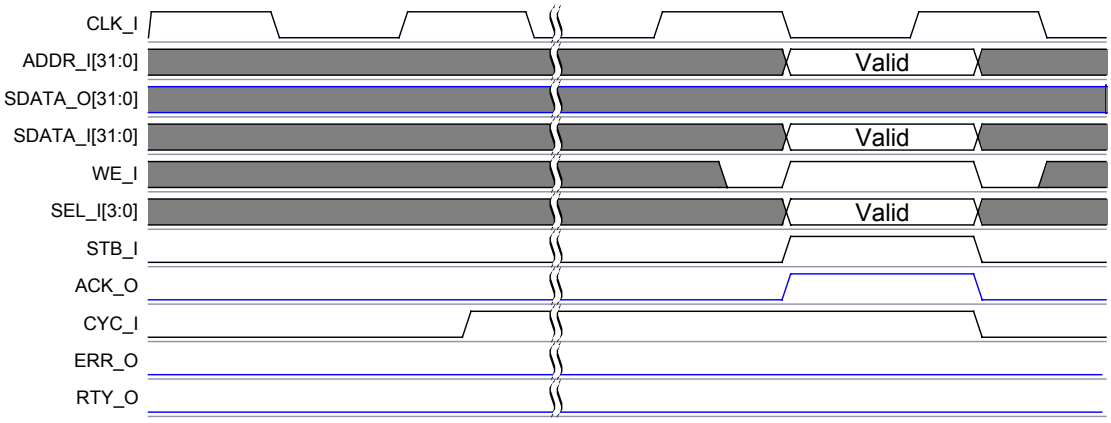


Figure 6.2: WISHBONE Configuration Write cycle

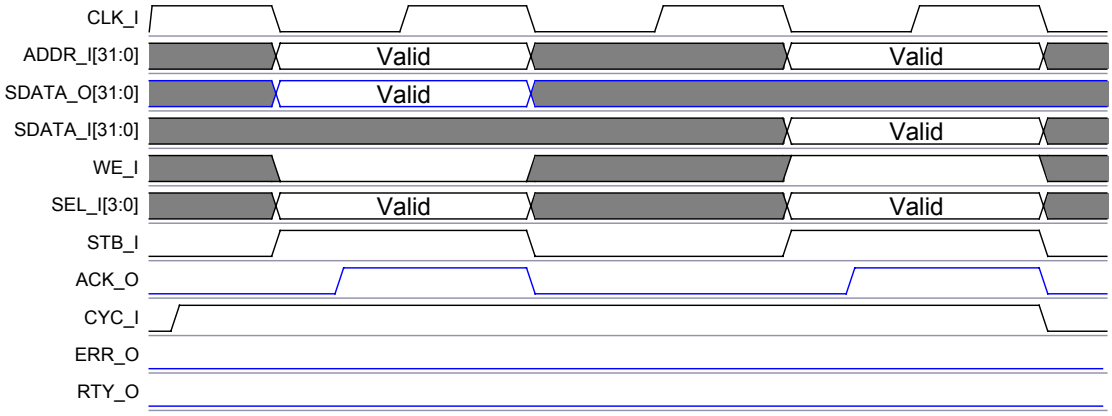


Figure 6.3: WISHBONE configuration RMW cycle

Wishbone masters will most commonly use Single Read cycles for accessing the core's configuration space as shown in Figure 6.1. A Write cycle to the core's register space by the WISHBONE master is shown in Figure 6.2. Writes to unimplemented configuration space have no effect while Read cycles return all 0s. RMW cycles to the core's configuration space are also accepted, as shown in Figure 6.3, and are most commonly used for interrupt handling since a RMW cycle is defined as atomic (indivisible) operation in the *WISHBONE Bus Specification*.

6.1.2. WISHBONE to PCI Accesses

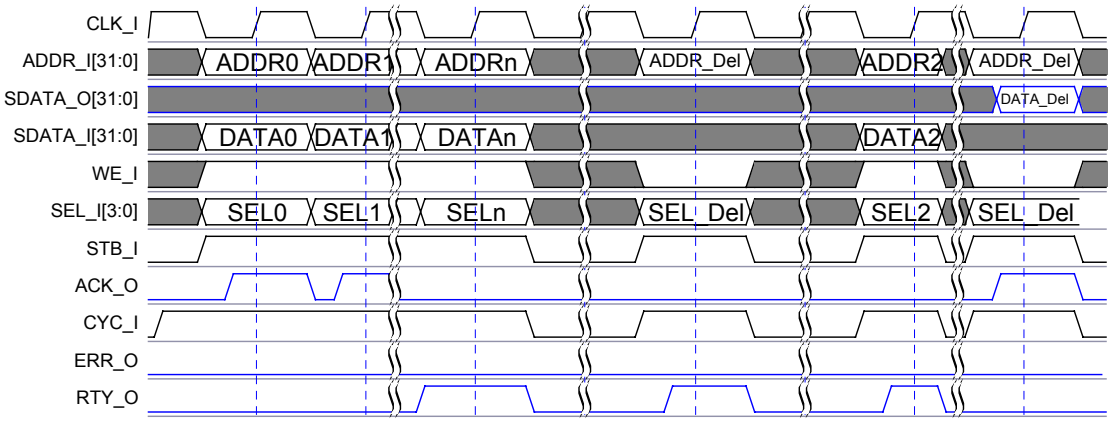


Figure 6.4: WISHBONE access to PCI address space

Figure 6.4 shows how the WISHBONE master perceives cycles intended for PCI address space traveling through the WISHBONE slave unit of the core. The first cycle in the figure initiated by the WISHBONE master is a Block Write cycle. The WISHBONE slave module accepts Write cycles until WBW_FIFO is full. Subsequent Write cycles in this block cycle are terminated with **Retry** (RTY_O asserted on ADDRn, DATAn, SELn transfer). The second cycle in the figure is a Read cycle. Read cycles from PCI address space are retried immediately (RTY_O asserted on first ADDR_Del, SEL_Del transfer). Address, byte enable, and CAB_I information is latched by the WISHBONE slave unit on the first rising edge of CLK_I where STB_I is asserted. The third cycle is a Write cycle to the PCI address space and is retried, too. In this case, the WISHBONE slave unit signals a **Retry** if one of the following possibilities occurs:

- WBW_FIFO is still full from previous transfers.
- A delayed Read cycle latched in a previous transfer has not completed on the PCI bus yet.
- A Delayed Read completion is present in the PCI target unit and has not been completed on the PCI bus yet.

In the 4th cycle, the WISHBONE master retries a Read request initiated and latched by the WISHBONE slave module in the 2nd cycle. Since the PCI master module has already performed a Read cycle on the PCI bus and stored data in WBR_FIFO, the WISHBONE slave module takes data from the FIFO and delivers it on the WISHBONE bus. The WISHBONE slave module can supply data for the master as long as WBR_FIFO contains any data and Read addresses are serial and DWORD aligned.

6.1.3. PCI Cycles

The WISHBONE slave unit incorporates a PCI master module that is capable of initiating various types of PCI address space accesses.

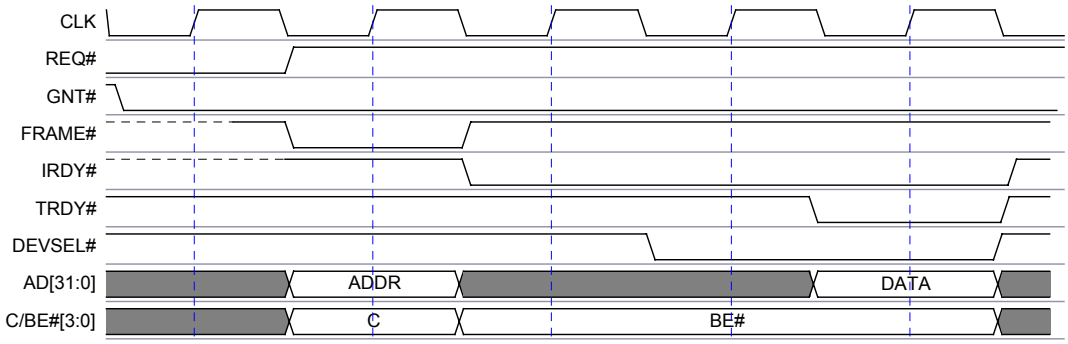


Figure 6.5: PCI Single Read cycle

Figure 6.5 shows a Single Read cycle on the PCI bus performed by the PCI master module. On the first clock edge, the PCI master module samples its GNT# signal asserted and starts the bus cycle by asserting FRAME# on the next rising edge of the clock. The 2nd clock cycle is an address phase, so address and bus command information is provided on AD and C/BE# lines respectively. At the end of an address phase, the master module de-asserts FRAME# and asserts IRDY#, indicating its wish to perform a single data phase only. A device with medium decoding has been assumed for a diagram, so nothing happens on the 3rd rising edge of clock. On the 4th clock, the target device claims the transaction by asserting DEVSEL#. Target inserted a wait cycle by delaying assertion of TRDY#. On the 5th clock, actual data transfer occurs, indicated by TRDY# and IRDY# being asserted at the same time. Immediately afterwards, the master module de-asserts IRDY#, indicating the end of transfer.

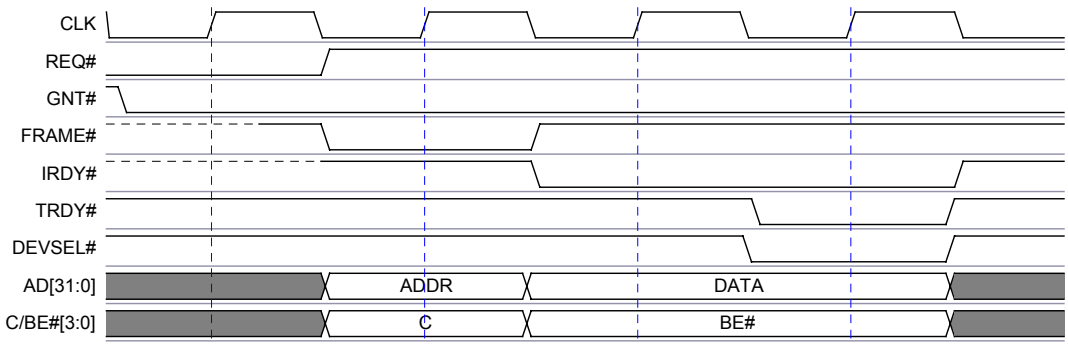


Figure 6.6: PCI Single Write

Figure 6.6 shows a Single Write cycle on the PCI bus performed by the PCI master module. On the first clock edge, the PCI master module samples its GNT# signal asserted and starts the bus cycle by asserting FRAME# on the next rising edge of the clock. The 2nd clock cycle is also an address phase, thus address and bus command information is provided on AD and C/BE# lines respectively. At the end of an address phase, the master module de-asserts FRAME# and asserts IRDY#, indicating its wish to perform a single data phase only. By asserting IRDY#, Write data and byte enables must be driven on AD and C/BE# lines respectively. A device with medium decoding has been assumed for a diagram, so nothing happens on the 3rd rising edge of the clock. On the 4th clock, the target device

claims access by asserting DEVSEL#. On this clock, actual data transfer occurs also, indicated by TRDY# and IRDY# being asserted at the same time. Immediately afterwards, the master module de-asserts IRDY#, indicating the end of transfer.

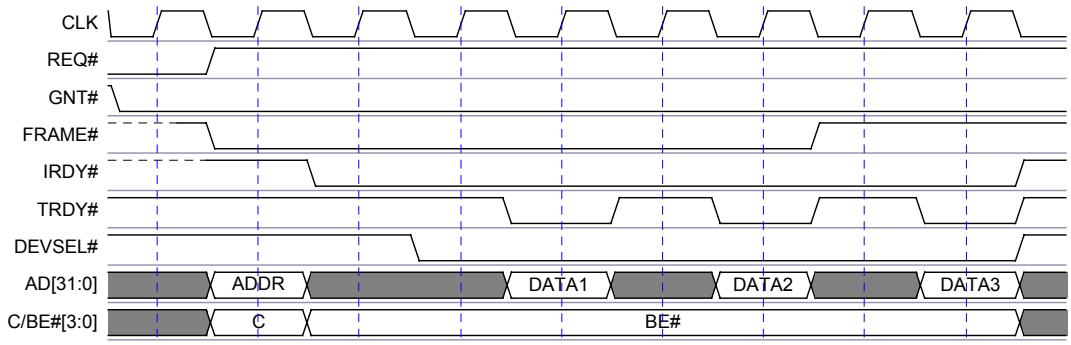


Figure 6.7: PCI Burst Read Cycle

Figure 6.7 shows how the PCI master module performs Burst Read transactions. The mechanism for claiming the bus is the same as in previous diagrams. The main difference lies with the fact that FRAME# stays asserted till the last data transfer. A medium decode target device is assumed for the diagram that inserts a wait cycle on clock 4. The target also inserts one WS after each data phase. Byte enables do not change during bursts. They are always 0000. The last data phase is phase 3, which is indicated by FRAME# de-asserted and IRDY# asserted at the same clock edge. Immediately after the master module latched data from the bus (clock edge when TRDY# is asserted), it de-asserts IRDY# to indicate an end of the transfer.

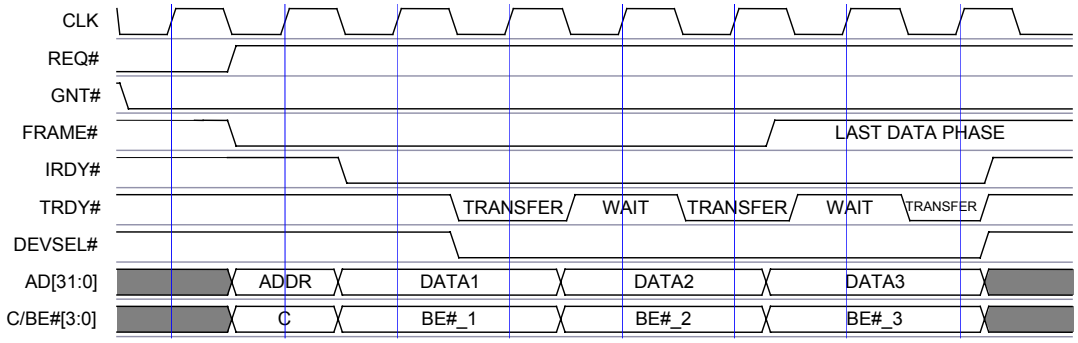


Figure 6.8: PCI Burst Write cycle

Figure 6.8 shows PCI Burst Write cycles performed by the PCI master module. The mechanism for claiming the bus is the same as in the previous diagrams. FRAME# stays asserted till the last data transfer. A medium decode target device is assumed for a diagram that claims access and latches the first data on clock 4. The target also inserts one WS after each data phase. The last data phase is phase 3, which is indicated by FRAME# de-asserted and IRDY# asserted at the same clock edge.

Immediately after the target latched data from the bus (clock edge when TRDY# is asserted), the master module de-asserts IRDY# to indicate an end of the transfer.

6.1.4. PCI Terminations

6.1.4.1 Master Initiated Terminations

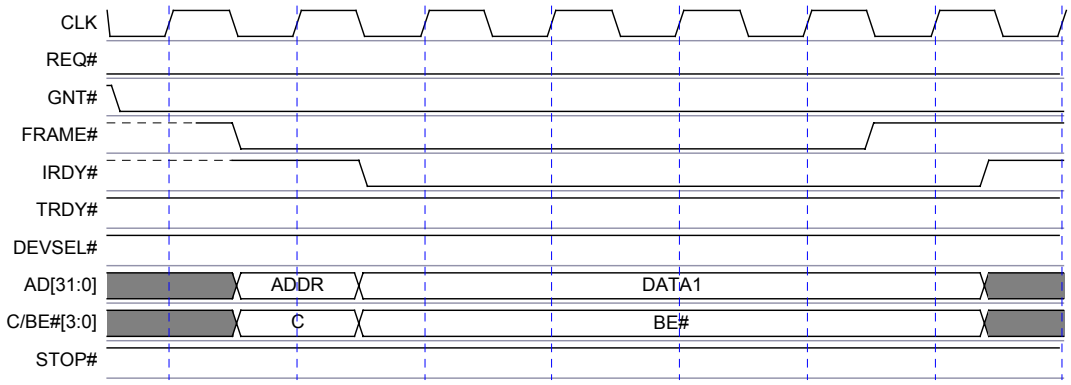


Figure 6.9: Master Abort termination

The PCI master module terminates the transaction with Master Abort, as shown in Figure 6.9. What happens? The master initiates a transaction starting with the address phase and waits for the target to respond by asserting DEVSEL#. The master is only required to wait for the assertion of DEVSEL# for 4 clocks. If DEVSEL# will not have been asserted by the 4th clock (subtractive decode devices), the master de-asserts FRAME# and must hold IRDY# asserted for an additional clock cycle indicating the end of the transaction.

If Error Reporting is enabled and the transaction is a Posted Write cycle, then address, bus command, data, and byte enables are stored in corresponding registers (see chapter 3.3.3). The current transaction is discarded (pulled out of WBW_FIFO) while any other Posted Write transactions are not influenced by Error.

If the transaction is a Read cycle, the termination is signaled to the WISHBONE master with an error on the WISHBONE bus when it retries a Read request.

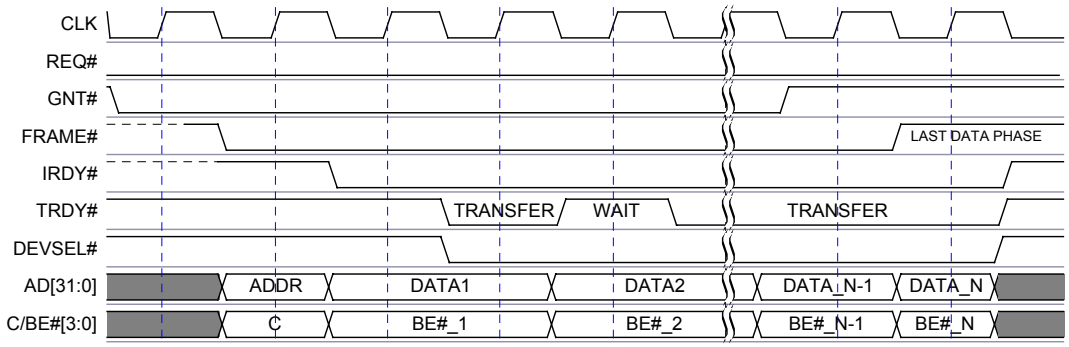


Figure 6.10: Timeout termination

The **Timeout** termination is specified in the *PCI Local Bus Specification*. It must be implemented in the PCI master module. Timeout termination is not an abnormal termination; it is simply a means of assuring other masters access to the PCI bus within a reasonable span of time. The master is supposed to complete the transaction by the time the latency timer expires and its GNT# has been removed by the PCI arbiter. In other words, when the master latency timer expires, the PCI master module must sample its GNT# on every rising edge of clock. If it samples it in de-asserted state, it must complete the transaction as soon as possible. As shown in Figure 6.10, the latency timer of the master is assumed to expire and its grant to be removed by data phase N-1. The master module samples GNT# de-asserted, thus it completes an access on the next clock cycle by de-asserting FRAME#.

Timeout terminations are not signaled to the WISHBONE bus since the PCI master module can resume transaction the next time it gains bus mastership.

Timeout detection is implemented with a counter and the Master Latency Timer register in the PCI configuration space. The counter is enabled when the PCI master module asserts FRAME# and is cleared and suspended as soon as FRAME# is de-asserted.

6.1.4.2 Target Terminations Handled by PCI Master Module

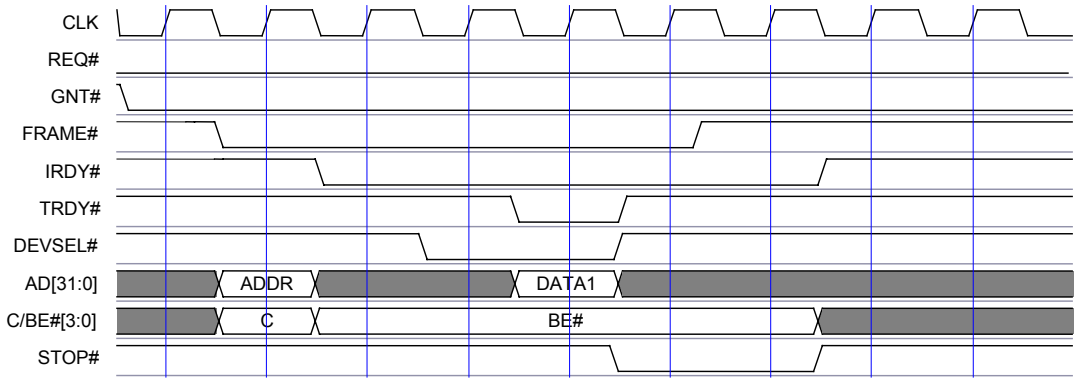


Figure 6.11: Target Abort

A target signals **Target Abort** to the master when it is and will be unable to complete the access initiated by the master. The master should not attempt to retry accesses terminated with **Target Abort**.

Posted Write cycles terminated with **Target Abort** are discarded. If Error Reporting is enabled, the WISHBONE slave unit reports an error (see Chapter 3.3.3).

The **Target Abort** termination during Read cycles is signaled to the WISHBONE master when retrying the request. Access to the address that resulted in **Target Abort** is terminated with an error on the WISHBONE bus. If the WISHBONE master never accesses the address that resulted in **Target Abort**, termination will not be signaled in any way (**Target Abort** can be signaled because the PCI master module reads over address space boundaries of a specific target during a pre-fetched Read transaction).

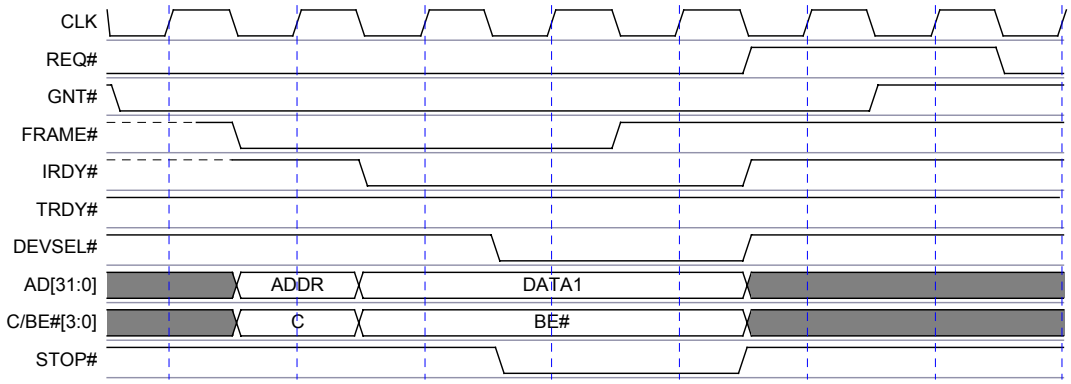


Figure 6.12: Target Retry

A target signals a **Retry** to the master when it is not ready to process the request. No data is transferred during **Retry**. Nevertheless, the PCI master must still terminate normally by de-asserting FRAME# and keeping IRDY# asserted for one PCI clock cycle to indicate the last data phase. The master must relinquish the PCI bus for at least two cycles after it received a **Target Retry** by de-asserting its REQ# line. It must also retry the same request at a later time.

Target Retry is not signaled on the WISHBONE bus. The PCI master module retries the transaction transparently on the PCI bus.

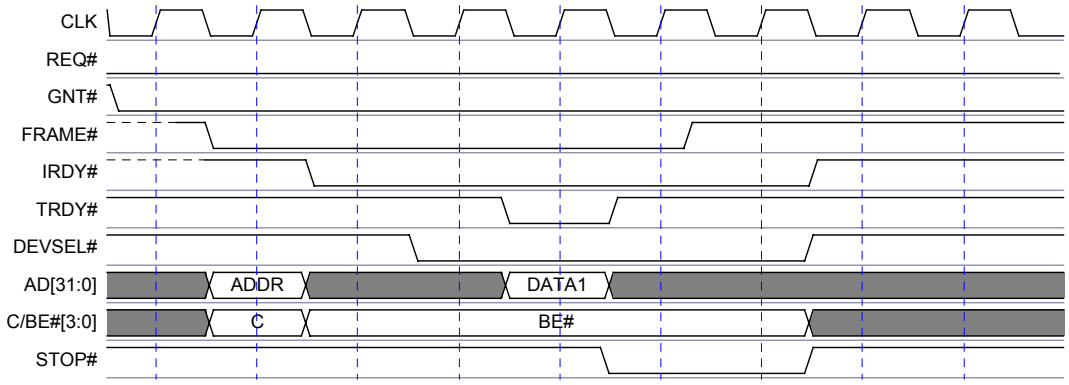


Figure 6.13: Target Disconnect without data

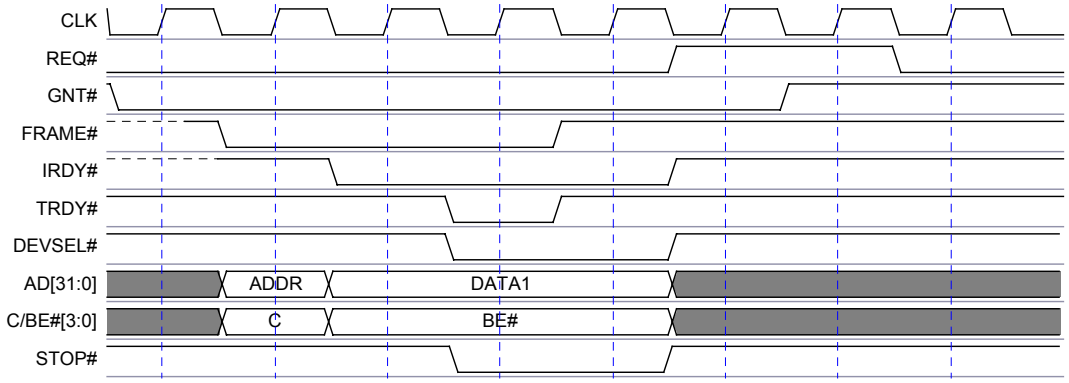


Figure 6.14: Target Disconnect with data

A target signals **Target Disconnect** to the master when it is not capable of receiving or supplying any more data from/to the master. Data must be transferred with (**Disconnect with Data**) or before (**Disconnect without Data**) the target signals **Target Disconnect**. The master must terminate the transaction normally by de-asserting FRAME# and keeping IRDY# asserted for one clock cycle. If the target signals **Target Disconnect with data** on the last data phase (FRAME# de-asserted, IRDY#, TRDY#, and STOP# asserted), the termination is treated as a normal master termination. (e.g. STOP# is a Logical Don't Care for a master when FRAME# is de-asserted and IRDY# and TRDY# are asserted).

Target Disconnect is not an abnormal termination and will not be signaled to the WISHBONE master in any way.

6.2. PCI Target Unit

This section describes basic waveforms of various accesses to core configuration space and mapped WISHBONE address space. Waveforms supplied have only informational value at this time.

6.2.1. PCI Configuration Accesses

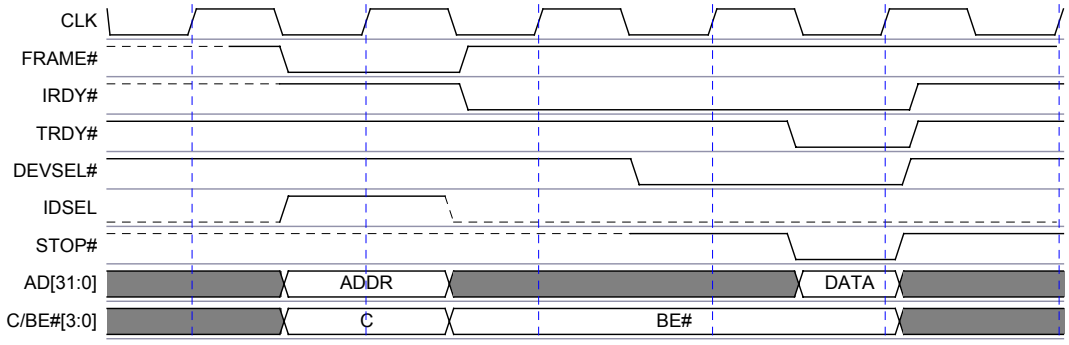


Figure 6.15: PCI Configuration Read cycle

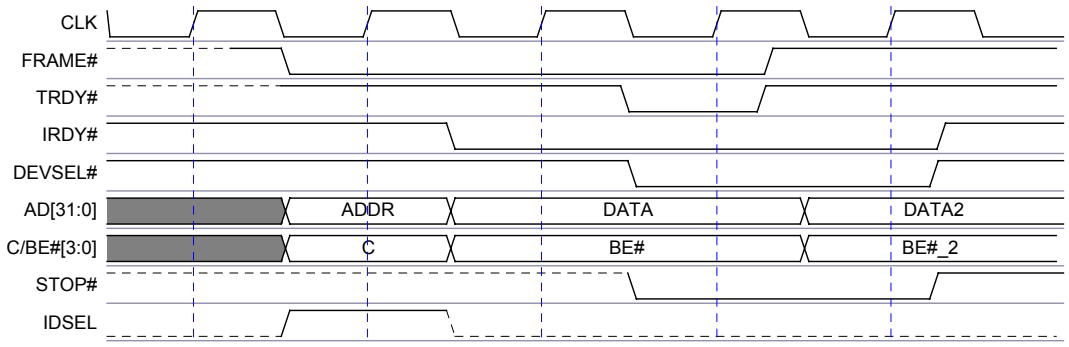


Figure 6.16: PCI Configuration Write cycle

PCI initiators will most commonly use Single Read cycles for accessing the core configuration space as shown in Figure 6.15. A Write cycle to the register space of the core by the PCI initiator is shown in Figure 6.16. Write cycles to unimplemented configuration space have no effect, while Read cycles return all 0s.

6.2.2. PCI to WISHBONE Accesses With WISHBONE Cycles

The following figures show how the PCI initiator sees cycles intended for the WISHBONE address space, traveling through the PCI target unit of the core. The first cycle in Figure 6.17, started by the PCI initiator, is a Delayed Read request. The PCI target module accepts the Read request. Subsequent Reads in this cycle are terminated with **Retry**. The next figure shows the previous transaction transferred to the WISHBONE bus. The second cycle in the first figure is a Read from the PCI master.

For reference: There are also burst accesses from the PCI through the PCI target module (Read and Write) on Figure 6.19 and Figure 6.20. Last follows a diagram of a Write transfer on the WISHBONE bus initiated by the PCI initiator.

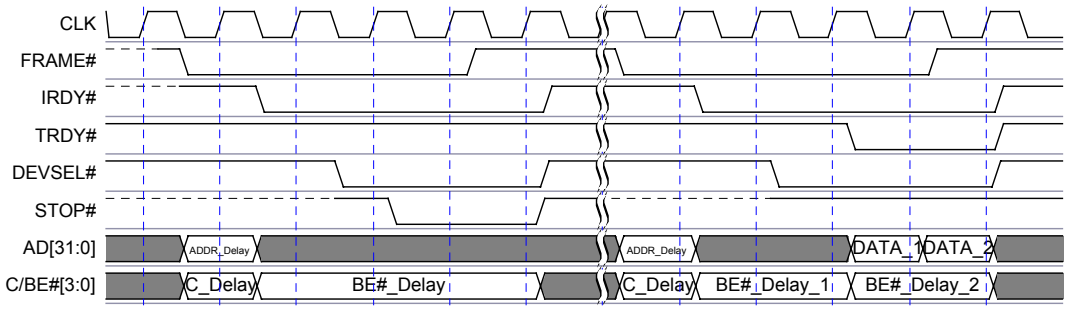


Figure 6.17: PCI Target Read cycle

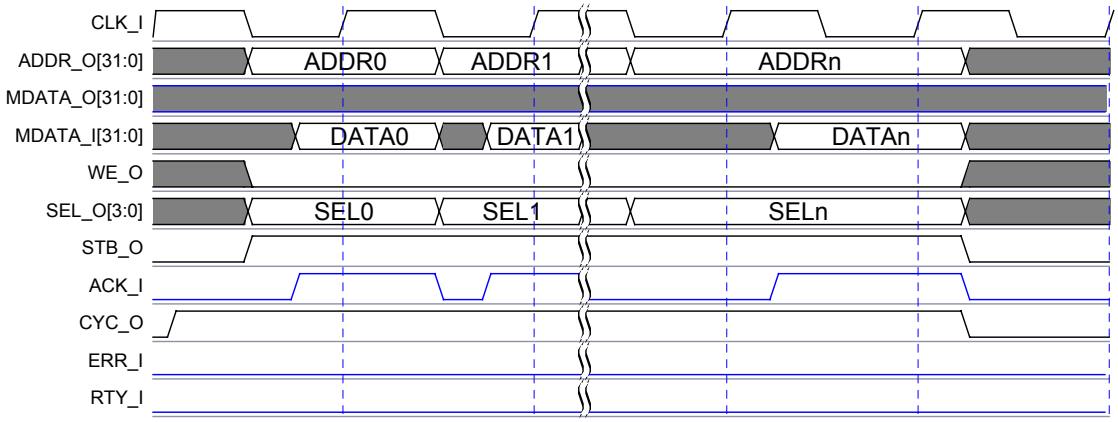


Figure 6.18: PCI to WISHBONE Read cycle

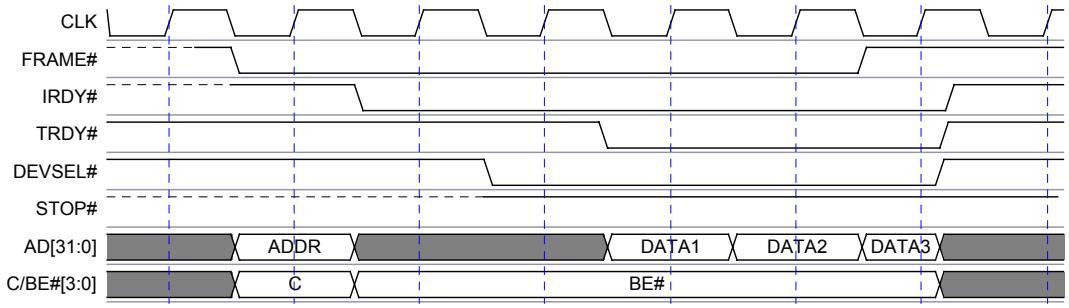


Figure 6.19: PCI Initiator to Target Burst Read cycle

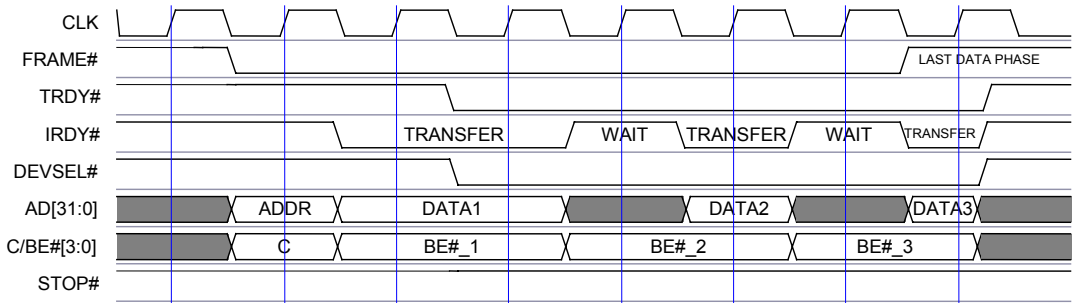


Figure 6.20: PCI Initiator to Target Burst Write cycle

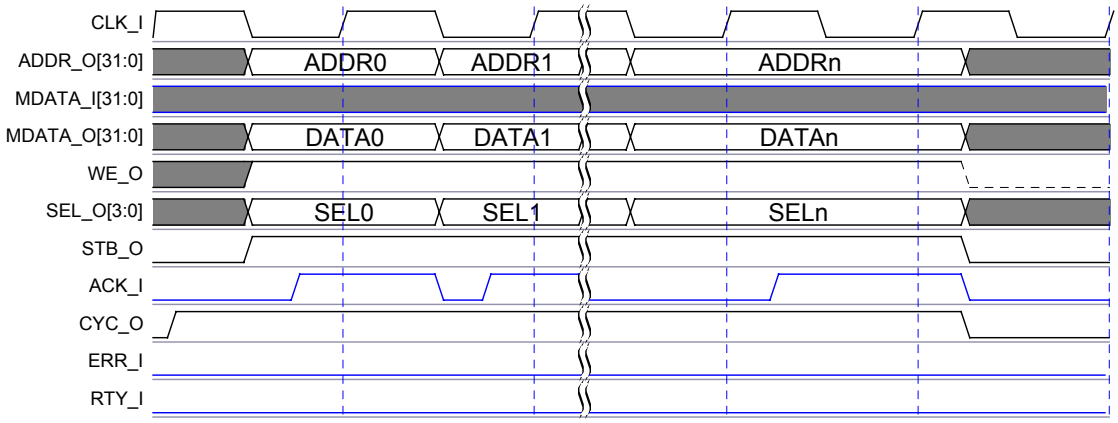


Figure 6.21: WISHBONE Write transfer caused by PCI to WISHBONE Write cycle

6.2.3. WISHBONE Terminations

Terminations on the WISHBONE bus are always performed by WISHBONE slaves. Chapters PCI to WISHBONE Write Cycles and PCI to WISHBONE Read Cycles describe the causes of **Retry** or **Error** on the WISHBONE bus.

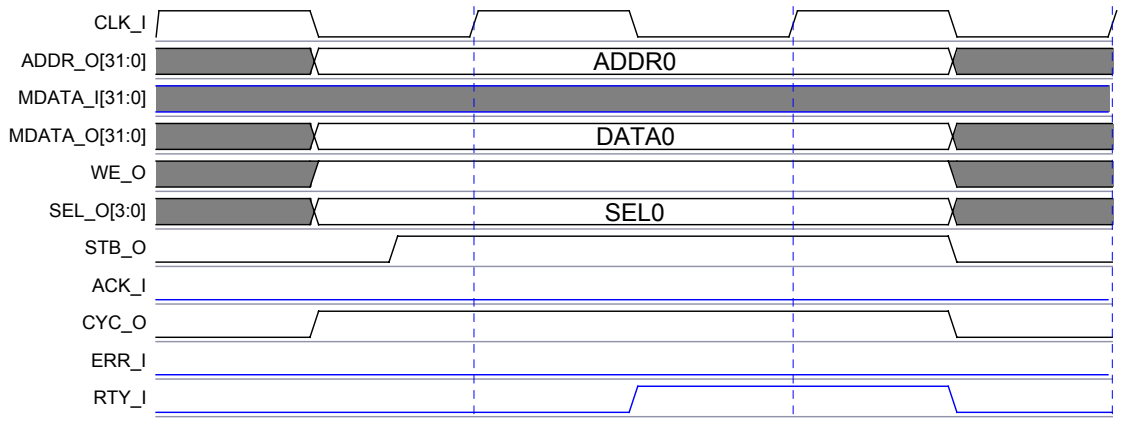


Figure 6.22: Retry on WISHBONE bus caused by PCI to WISHBONE transfer

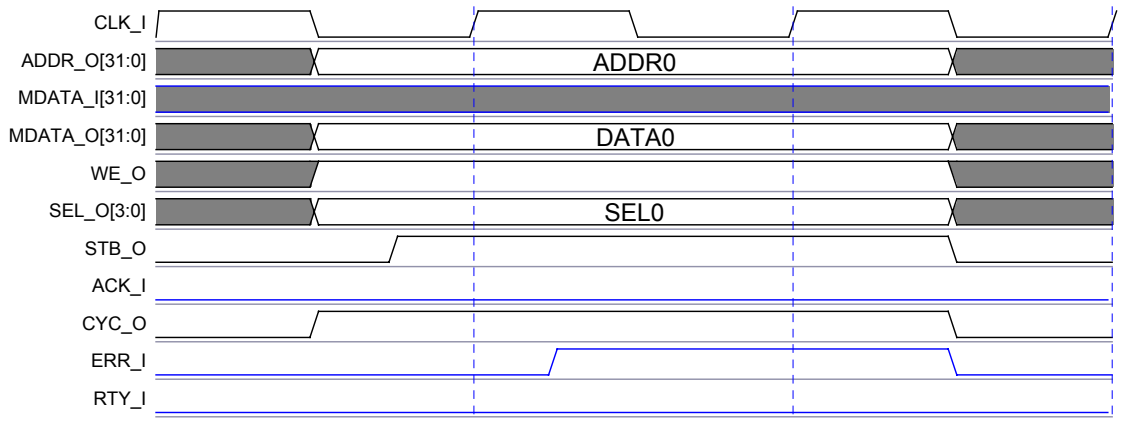


Figure 6.23: Error on WISHBONE bus caused by PCI to WISHBONE transfer

Appendix A

Core HW Configuration

This section summarizes parameters that are set by the system designer of the core and define cores configuration, the user (e.g. programmer) must know. The system designer must set the parameters before actually using the core in simulation or synthesis. For details read PCI IP Core design document.

Configuration parameters are grouped into the *pci_user_constants.v* file, which can be edited by the system designer, depending on the needs of the application (see chapters 2.4 and 2.5 of the PCI IP Core Design document). Only parameters useful for the user of the core are summarized here. The system designer should mark which parameters are defined (or the value of the parameter).

Parameter	Defined value / Defined (yes, no)
Description	
HOST / GUEST	
These two defines are mutually exclusive. Core will be implemented or simulated with HOST or GUEST bridge features enabled (see chapter 3).	
WBW_ADDR_LENGTH	
If WB_RAM_DONT_SHARE is defined, this value must be less than or equal to WB_FIFO_RAM_ADDR_LENGTH value. If WB_RAM_DONT_SHARE is not defined, this value must be less than WB_FIFO_RAM_ADDR_LENGTH value.	
WBR_ADDR_LENGTH	
If WB_RAM_DONT_SHARE is defined, this value must be less than or equal to WB_FIFO_RAM_ADDR_LENGTH value. If WB_RAM_DONT_SHARE is not defined, this value must be less than WB_FIFO_RAM_ADDR_LENGTH value.	
PCIW_ADDR_LENGTH	
If PCI_RAM_DONT_SHARE is defined, this value must be less than or equal to PCI_FIFO_RAM_ADDR_LENGTH value. If PCI_RAM_DONT_SHARE is not defined, this value must be less than PCI_FIFO_RAM_ADDR_LENGTH value.	
PCIR_ADDR_LENGTH	

Parameter	Defined value / Defined (yes, no)
Description	
If PCI_RAM_DONT_SHARE is defined, this value must be less than or equal to PCI_FIFO_RAM_ADDR_LENGTH value. If PCI_RAM_DONT_SHARE is not defined, this value must be less than PCI_FIFO_RAM_ADDR_LENGTH value.	
Four values defined above define each FIFO's size. Size is calculated as 2^^ADDR_LENGTH. Note that FIFO's control logic is such, that one location in RAM is always empty, so usable FIFO size is (2^^ADDR_LENGTH) – 1. Any value equal to or larger than 3 is valid here – the only restriction is the size of RAMs instantiated for FIFO storage.	
PCI_FIFO_RAM_ADDR_LENGTH	
Address length of RAM instance used in *pci_pci_tpram.v* file.	
WB_FIFO_RAM_ADDR_LENGTH	
Address length of RAM instance used in *pci_wb_tpram.v*.	
PCI_RAM_DONT_SHARE	
Selects the type of implementation for PCI Target unit's FIFOs. If defined, each FIFO in the PCI Target unit uses its own RAM instance. If not defined, both FIFOs in the PCI Target unit use the same RAM instance for their storage space.	
WB_RAM_DONT_SHARE	
Selects the type of implementation for WISHBONE Slave unit's FIFOs. If defined, each FIFO in the WISHBONE Slave unit uses its own RAM instance. If not defined, both FIFOs in the WISHBONE Slave unit use the same RAM instance for their storage space.	
ACTIVE_LOW_OE / ACTIVE_HIGH_OE	
These two mutually exclusive defines select the active levels for *pci_*_oe_o* and *pci_cpci_*_oe_o* signals.	
REGISTER_WBS_OUTPUTS	
If defined, WISHBONE Slave module registers all of its outputs on the WISHBONE bus. Useful for applications with a lot of interconnection logic. The speed of the WISHBONE Slave interface decreases if REGISTER_WBS_OUTPUTS is defined. If **PCI_WB_REV_B3** is defined, the outputs are already registered, so you need not to define this macro.	
ADDR_TRAN_IMPL	
If defined, address translation functionality is added to decoders for both, PCI and WISHBONE accesses. Address translation implementation is useful when application uses fixed address map, while PCI address map is configurable.	
PCI_NUM_OF_DEC_ADDR_LINES	
Number defined here is used for controlling implementation of PCI images' decoders. It defines how many MS address lines are used for decoding PCI Target accesses and therefore defines what minimum image size can be. Maximum number allowed is 20 (4KB minimum image size) and minimum is 1 (2GB minimum image size – this value implies that more than two images cannot be enabled at the same time).	
NO_CNF_IMAGE	

Parameter	Defined value / Defined (yes, no)
Description	
If defined, it prevents Read-Only configuration image to be implemented. Read-Only Configuration space access can be provided through PCI image 0 for HOST implementation of the Core, and through WB image 0 for GUEST implementation. If NO_CNF_IMAGE is defined, then this image is not implemented (some additional space is saved).	
PCI_IMAGE0 [1]	
This define only has meaning when HOST and NO_CNF_IMAGE are defined also. This enables usage of additional PCI Target image 0 (PCI_IMAGE0) for accessing WISHBONE bus address space from PCI address space. Otherwise, PCI_IMAGE0 does not need to be defined, since it is always used for accessing Configuration space.	
PCI_IMAGE2 [1]	
PCI_IMAGE3 [1]	
PCI_IMAGE4 [1]	
PCI_IMAGE5 [1]	
If whichever defined, then that PCI Target image is implemented.	
PCI_AM0 [3]	
PCI_AM1	
PCI_AM2 [2]	
PCI_AM3 [2]	
PCI_AM4 [2]	
PCI_AM5 [2]	
Numbers defined here are initial (reset) values of PCI address masks' registers. These are very important if the Core is implemented as GUEST, since configuration is done via PCI Target state machine. If the designer wants an implemented PCI Target image to be detected by device independent software at system power-up, he has to set initial masks to enabled state – MS bit has to be 1. Other bits can have a value of 1 or zero, depending on what size of an image has to be presented to the software. The masks can be set inactive also, but device independent software won't detect implemented PCI Target images and therefore not configure them. Device specific software will then have to jump in to configure images with inactive initial masks defined, which also means that it will probably have to rebuild PCI address space map.	
PCI_BA0_MEM_IO [3]	
PCI_BA1_MEM_IO	
PCI_BA2_MEM_IO [2]	
PCI_BA3_MEM_IO [2]	
PCI_BA4_MEM_IO [2]	
PCI_BA5_MEM_IO [2]	

Parameter	Defined value / Defined (yes, no)
Description	
Numbers defined here are initial (reset) values of PCI Base Address registers' bits 0. If the Core is configured as HOST, this initial values can later be changed by writing appropriate value to appropriate PCI Base Address register. If the core is GUEST, than this values are hardwired, because device independent software must know in advance where to map each PCI Base Address.	
PCI_TA0 [3]	
PCI_TA1	
PCI_TA2 [2]	
PCI_TA3 [2]	
PCI_TA4 [2]	
PCI_TA5 [2]	
The macros above must be defined as 20 bit values and specify the reset value of the corresponding PCI Translation Address n register. The values are relevant only if **ADDR_TRAN_IMPL** is also defined.	
PCI_AT_EN0 [3]	
PCI_AT_EN1	
PCI_AT_EN2 [2]	
PCI_AT_EN3 [2]	
PCI_AT_EN4 [2]	
PCI_AT_EN5 [2]	
The macros above must be defined as 1 bit values and specify the reset value of the Address Translation Enable bit in the corresponding PCI Image Control n register. The values are relevant only if **ADDR_TRAN_IMPL** is also defined.	
WB_NUM_OF_DEC_ADDR_LINES	
Number defined here is used for controlling implementation of WISHBONE images' decoders. It defines how many MS address lines are used for decoding WISHBONE Slave accesses and therefore defines what minimum image size can be. Maximum number allowed is 20 (4KB minimum image size) and minimum is 1 (2GB minimum image size – this value implies that more than two images cannot be enabled at the same time).	
WB_IMAGE2	
WB_IMAGE3	
WB_IMAGE4	
WB_IMAGE5	
If whichever defined, then that WB Slave image is implemented. WISHBONE Image 1 is always implemented.	
WB_BA1	
WB_BA2 [4]	
WB_BA3 [4]	

Parameter	Defined value / Defined (yes, no)
Description	
WB_BA4 [4]	
WB_BA5 [4]	
The macros above must be defined as 20 bit values and specify the reset value of the corresponding WISHBONE Base Address n register.	
WB_BA1_MEM_IO	
WB_BA2_MEM_IO [4]	
WB_BA3_MEM_IO [4]	
WB_BA4_MEM_IO [4]	
WB_BA5_MEM_IO [4]	
The macros above must be defined as 1 bit values and specify the reset value of the address space mapping bit in the corresponding WISHBONE Base Address n register.	
WB_AM1	
WB_AM2 [4]	
WB_AM3 [4]	
WB_AM4 [4]	
WB_AM5 [4]	
The macros above must be defined as 20 bit values and specify the reset value of the corresponding WISHBONE Address Mask n register.	
WB_TA1	
WB_TA2 [4]	
WB_TA3 [4]	
WB_TA4 [4]	
WB_TA5 [4]	
The macros above must be defined as 20 bit values and specify the reset value of the corresponding WISHBONE Translation Address n register. The values are relevant only if **ADDR_TRAN_IMPL** is also defined.	
WB_AT_EN1	
WB_AT_EN2 [4]	
WB_AT_EN3 [4]	
WB_AT_EN4 [4]	
WB_AT_EN5 [4]	
The macros above must be defined as 1 bit values and specify the reset value of the Address Translation Enable bit in the corresponding WISHBONE Image Control n register. The values are relevant only if **ADDR_TRAN_IMPL** is also defined.	
WB_CONFIGURATION_BASE	

Parameter	Defined value / Defined (yes, no)
Description	
Number defined here is a 20 bit value for WISHBONE configuration image address. Those bits are compared to 20 MS bits of WB Slave address to decode Configuration accesses from WB bus. This is constant value and cannot be changed after the Core is implemented, since WB bus does not provide any special mechanism for device configuration.	
WB_RTY_CNT_MAX	
Number defined here is used to prevent deadlock in WB Master state machine for maximum counting value of RTY terminations on WB bus, before ACK or ERR terminations. The last two terminations reset the counter. This counter is also used, when no WB device responds (e.g. if accessing to unused memory locations). In that case internal **set_retry** signal is set every 8 WB clock periods and counter counts to maximum value defined.	
PCI_WBM_NO_RESPONSE_CNT_DISABLE	
Disables the WISHBONE Master's internal no response counter. Useful if the application consists of one or more WISHBONE Slaves that will need a lot of cycles to respond.	
PCI_WB_REV_B3	
Enables the WISHBONE Rev. B3 to WISHBONE Rev. B2 translation logic for WISHBONE Slave interface. You need to include the *pci_wbs_wbb3_2_wbb2.v* into the application, since it contains the necessary logic. Since the outputs for WISHBONE Slave interface are registered in this module, you do not need to define REGISTER_WBS_OUTPUTS.	
PCI_WBS_B3_RTY_DISABLE	
Disables the RTY termination generation for WISHBONE Rev. B3 Slave interface. If your application contains WISHBONE B3 master cores that do not support RTY termination, you have to define this macro to prevent non-linear incrementing bursts to be interpreted in the wrong way.	
PCI33 / PCI66	
These two defines are mutually exclusive. They are used for simulation purposes (PCI clock speed) and to set 66MHz Capable bit in PCI Device Status register, if PCI66 is defined. There are no other features dependent on those defines.	
HEADER_VENDOR_ID	
Each PCI bus compatible hardware vendor gets its 16 bit hexadecimal ID from PCI SIG organization. It should be specified in this define. This value shows up in Vendor ID register of PCI Type0 Configuration Header.	
HEADER_DEVICE_ID	
Device ID is vendor specific, 16 bit hexadecimal value. It shows up in Device ID register of PCI Type0 Configuration Header.	
HEADER_REVISION_ID	
Revision ID is vendor specific, 8 bit hexadecimal value, that shows up in Revision ID register of PCI Type0 Configuration Header.	
PCI_CPCI_HS_IMPLEMENT	

Parameter	Defined value / Defined (yes, no)
Description	
If defined, the RTL implementation of the PCI Bridge will have CompactPCI How Swap functionality enabled. This feature is currently supported for GUEST bridge implementations only. The enabled Hot Swap functionality provides additional pins on the top level of the design (*pci_bridge32.v*) as well as a few changes in the core's configuration space. See Compact PCI Hot Swap support.	
PCI_SPOCI	
Only use this define for GUEST implementations of the PCI Bridge. If this Macro is defined, the Serial Power On Configuration Interface logic will be implemented in the final design, enabling the PCI Bridge to configure its registers from the data in the serial EPROM device, without external intervention.	

1 – PCI image 1 is always implemented, without any exceptions

2 – This value is significant only if appropriate PCI image is implemented

3 – This value is significant only if PCI image 0 is implemented to access WB bus for HOST implementation

4 – This value is significant only if appropriate WISHBONE image is implemented.

Table 6.1: User Useful HARDWARE Configuration Parameters

Index