PCI IP Core Specification

Authors: Miha Dolenc & Tadej Markovic

mihad@opencores.org

tadej@opencores.org

Rev. 0.6

January 28, 2002

Revision History

Rev.	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

List of Contents

INTRODUCTION 1

1.1 WHAT IS A PCI BRIDGE? 1
1.2 PCI IP CORE INTRODUCTION 1
1.3 PCI IP CORE FEATURES 1

ARCHITECTURE 3

2.1 OVERVIEW 3
2.2 WISHBONE SLAVE UNIT 4
2.2.1 WISHBONE Slave Unit Architecture 5
2.3 PCI TARGET UNIT 6
2.3.1 PCI Target Unit Architecture 6
2.4 CLOCKS 8
2.5 FIFO 8
2.6 ADDRESS TRANSLATION LOGIC 10
2.6.1 Description of Address Translation Logic 10

OPERATION 12

3.1 CONFIGURATION SPACE 12
3.1.1 Configuration Space Access for Host Bus Bridges 13
3.1.2 Configuration Space Access for Guest Bridges 14
3.1.3 Configuration Cycles 15
3.1.4 Generating Configuration Cycles 16
3.1.5 Generating Interrupt Acknowledge Cycles 18
3.2 WISHBONE SLAVE UNIT 18
3.2.1 WISHBONE Slave Unit Functionality 19
3.2.2 Addressing and Images of the WISHBONE Slave Unit 20
3.2.3 WISHBONE to PCI Write Cycles 21
3.2.4 WISHBONE to PCI Read Cycles 23
3.3 PCI TARGET UNIT 25
3.3.1 PCI Target Unit Functionality 25
3.3.2 Addressing and Images of the PCI Target Unit 26
3.3.3 PCI to WISHBONE Write Cycles 28
3.3.4 PCI to WISHBONE Read Cycles 30
3.4 TRANSACTION ORDERING 32
3.5 PARITY 33
3.6 INTERRUPTS 33

REGISTERS 35

4.1 REGISTER LIST AND DESCRIPTION 35
4.1.1 WISHBONE Slave Unit Control & Status 38
4.1.2 PCI Target Unit Control & Status 42
4.1.3 Reporting Registers 51
4.1.4 Interrupt Control & Status Registers 57
4.2 SOFTWARE OBLIGATIONS 60

IO PORTS 61

5.1 PCI INTERFACE 61
5.1.1 Required PCI Interface Pins *61*
5.1.2 Implemented Optional PCI Interface Pins *62*
5.2 WISHBONE INTERFACE 63

WAVEFORMS 65

6.1 WISHBONE SLAVE UNIT 65
6.1.1 WISHBONE Configuration Accesses *65*
6.1.2 WISHBONE to PCI Accesses *67*
6.1.3 PCI Cycles *67*
6.1.4 PCI Terminations *70*
6.2 PCI TARGET UNIT 74
6.1.2 PCI Configuration Accesses *74*
6.2.2 PCI to WISHBONE Accesses With WISHBONE Cycles *75*
6.2.3 WISHBONE Terminations *77*

CORE HW CONFIGURATION 78

A.1 HW CONFIGURATION PARAMETERS 78

INDEX 82

List of Tables

TABLE 3-1: VALUE ON AD[31:11] PCI BUS LINES DURING ADDRESS PHASE OF CONFIGURATION CYCLE TYPE 0 17

TABLE 3-2: VALID ADDR_O(1:0) AND SEL_O(3:0) COMBINATIONS FOR I/O MAPPED ADDRESS SPACE ACCESS 22

TABLE 3-3: BUS COMMAND ENCODING FOR READ CYCLES THROUGH PCI MASTER MODULE 24

TABLE 3-4: VALID AD(1:0) AND BE# (3:0) COMBINATIONS FOR I/O MAPPED ADDRESS SPACE ACCESSES 28

TABLE 3-5: BURST ORDERING COMBINATIONS FOR MEMORY MAPPED ADDRESS SPACE ACCESSES 29

TABLE 3-6: BUS COMMAND ENCODING FOR READ CYCLES THROUGH PCI TARGET MODULE 31

TABLE 4-1: LIST OF REGISTERS 38

TABLE 4-2: WISHBONE CONFIGURATION SPACE BASE ADDRESS REGISTER 38

TABLE 4-3: WISHBONE IMAGE CONTROL REGISTER 39

TABLE 4-4: WISHBONE IMAGE CONTROL REGISTER BIT DESCRIPTIONS 40

TABLE 4-5: WISHBONE BASE ADDRESS REGISTER 40

TABLE 4-6: WISHBONE BASE ADDRESS REGISTER BIT DESCRIPTIONS 40

TABLE 4-7: WISHBONE ADDRESS MASK REGISTER 41

TABLE 4-8: WISHBONE ADDRESS MASK REGISTER BIT DESCRIPTIONS 41

TABLE 4-9: WISHBONE TRANSLATION ADDRESS REGISTER 42

TABLE 4-10: WISHBONE TRANSLATION ADDRESS REGISTER BIT DESCRIPTIONS 42

TABLE 4-11: COMMAND REGISTER OF PCI CONFIGURATION HEADER 45

TABLE 4-12: STATUS REGISTER OF PCI CONFIGURATION HEADER 46

TABLE 4-13: BASE ADDRESS REGISTER OF PCI CONFIGURATION HEADER FOR MEMORY MAPPED SPACE 47

TABLE 4-14: BASE ADDRESS REGISTER OF PCI CONFIGURATION HEADER FOR I/O MAPPED SPACE 47

TABLE 4-15: PCI IMAGE0 BASE ADDRESS REGISTER 47

TABLE 4-16: PCI IMAGE CONTROL REGISTER 48

TABLE 4-17: PCI IMAGE CONTROL REGISTER BIT DESCRIPTIONS 48

TABLE 4-18: PCI BASE ADDRESS REGISTER 49

TABLE 4-19: PCI BASE ADDRESS REGISTER BIT DESCRIPTIONS 49

TABLE 4-20: PCI ADDRESS MASK REGISTER 50

TABLE 4-21: PCI ADDRESS MASK REGISTER BIT DESCRIPTIONS 50

TABLE 4-22: PCI TRANSLATION ADDRESS REGISTER 51

TABLE 4-23: PCI TRANSLATION ADDRESS REGISTER BIT DESCRIPTIONS 51

TABLE 4-24: WISHBONE ERROR CONTROL AND STATUS REGISTER 52

TABLE 4-25: WISHBONE ERROR CONTROL AND STATUS REGISTER BIT DESCRIPTIONS 53

TABLE 4-26: WISHBONE ERRONEOUS ADDRESS REGISTER 53

TABLE 4-27: WISHBONE ERRONEOUS DATA REGISTER 53

TABLE 4-28: PCI ERROR CONTROL AND STATUS REGISTER 53

TABLE 4-29: PCI ERROR CONTROL AND STATUS REGISTER BIT DESCRIPTIONS 55

TABLE 4-30: PCI ERRONEOUS ADDRESS REGISTER 55

TABLE 4-31: PCI ERRONEOUS DATA REGISTER 55

TABLE 4-32: CONFIGURATION ADDRESS REGISTER 55

TABLE 4-33: CONFIGURATION ADDRESS REGISTER BIT DESCRIPTIONS 56

TABLE 4-34: CONFIGURATION DATA REGISTER 57

TABLE 4-35: INTERRUPT ACKNOWLEDGE REGISTER 57

TABLE 4-36: INTERRUPT CONTROL REGISTER 57
TABLE 4-37: INTERRUPT CONTROL REGISTER BIT DESCRIPTIONS 58
TABLE 4-38: INTERRUPT STATUS REGISTER 59
TABLE 4-39: INTERRUPT STATUS REGISTER BIT DESCRIPTIONS 60
TABLE 5-1: PCI ADDRESS AND DATA PINS 61
TABLE 5-2: PCI INTERFACE CONTROL PINS 62
TABLE 5-3: PCI ERROR REPORTING PINS 62
TABLE 5-4: PCI ARBITRATION PINS (INITIATOR ONLY) 62
TABLE 5-5: PCI SYSTEM PINS 62
TABLE 5-6: PCI INTERRUPT PIN 62
TABLE 5-7: PCI INTERFACE CONTROL PINS 63
TABLE 5-8: PCI TARGET UNIT'S WISHBONE INTERFACE (MASTER) 63
TABLE 5-9: WISHBONE SLAVE UNIT'S WISHBONE INTERFACE (SLAVE) 64
TABLE 5-10: WISHBONE COMMON CONTROL AND SYSTEM I/OS 64

List of Figures & Examples

FIGURE 2-1: PCI BRIDGE CORE ARCHITECTURE 4
FIGURE 2-2: WISHBONE SLAVE UNIT ARCHITECTURE 5
FIGURE 2-3: PCI TARGET UNIT ARCHITECTURE OVERVIEW 7
FIGURE 2-4: DETAILED DESCRIPTION OF FIFO REGISTER LINES 8
FIGURE 2-5: FIFO ARCHITECTURE 9
FIGURE 2-6: ADDRESS TRANSLATION LOGIC 11
FIGURE 3-1: PCI BRIDGE CONFIGURATION SPACE 13
FIGURE 3-2: CONFIGURATION SPACE ACCESS FOR HOST BUS BRIDGES 14
FIGURE 3-3: CONFIGURATION SPACE ACCESS FOR GUEST BRIDGES 15
FIGURE 3-4: WISHBONE SLAVE UNIT ARCHITECTURE OVERVIEW 19
EXAMPLE 3-1: ADDRESS RANGE OF WISHBONE SLAVE IMAGE 20
EXAMPLE 3-2: ADDRESS TRANSLATION 21
FIGURE 3-5: PCI TARGET UNIT ARCHITECTURE OVERVIEW 25
EXAMPLE 3-3: ADDRESS RANGE OF WISHBONE SLAVE IMAGE 27
EXAMPLE 3-4: ADDRESS TRANSLATION 28
FIGURE 4-1: WISHBONE CONFIGURATION SPACE BASE ADDRESS REGISTER LAYOUT 39
FIGURE 4-2: WISHBONE IMAGE CONTROL REGISTER LAYOUT 39
FIGURE 4-3: WISHBONE BASE ADDRESS REGISTER LAYOUT 40
FIGURE 4-4: WISHBONE ADDRESS MASK REGISTER LAYOUT 41
FIGURE 4-5: WISHBONE TRANSLATION ADDRESS REGISTER LAYOUT 42
FIGURE 4-6: PCI CONFIGURATION SPACE HEADER (HEADER TYPE 00H) 43

FIGURE 4-7: PCI IMAGE0 BASE ADDRESS REGISTER LAYOUT – IMAGE0 USED FOR ACCESSING THE PCI CONFIGURATION SPACE HEADER (TYPE 00H) 48

FIGURE 4-8: PCI IMAGE CONTROL REGISTER LAYOUT 48

FIGURE 4-9: PCI BASE ADDRESS REGISTER LAYOUT 49

FIGURE 4-10: PCI ADDRESS MASK REGISTER LAYOUT 50

FIGURE 4-11: PCI TRANSLATION ADDRESS REGISTER LAYOUT 51

FIGURE 4-12: WISHBONE ERROR CONTROL AND STATUS REGISTER LAYOUT 52

FIGURE 4-13: PCI ERROR CONTROL AND STATUS REGISTER LAYOUT 54

FIGURE 4-14: CONFIGURATION ADDRESS REGISTER LAYOUT 56

FIGURE 4-15: INTERRUPT CONTROL REGISTER LAYOUT 57

FIGURE 4-16: INTERRUPT STATUS REGISTER LAYOUT 59

FIGURE 6-1: WISHBONE CONFIGURATION READ CYCLE 65

FIGURE 6-2: WISHBONE CONFIGURATION WRITE CYCLE 66

FIGURE 6-3: WISHBONE CONFIGURATION RMW CYCLE 66

FIGURE 6-4: WISHBONE ACCESS TO PCI ADDRESS SPACE 67

FIGURE 6-5: PCI SINGLE READ CYCLE 68

FIGURE 6-6: PCI SINGLE WRITE 68

FIGURE 6-7: PCI BURST READ CYCLE 69

FIGURE 6-8: PCI BURST WRITE CYCLE 69

FIGURE 6-9: MASTER ABORT TERMINATION 70

FIGURE 6-10: TIMEOUT TERMINATION 71

FIGURE 6-11: TARGET ABORT 71

FIGURE 6-12: TARGET RETRY 72

FIGURE 6-13: TARGET DISCONNECT WITHOUT DATA 73

FIGURE 6-14: TARGET DISCONNECT WITH DATA 73

FIGURE 6-15: PCI CONFIGURATION READ CYCLE 74

FIGURE 6-16: PCI CONFIGURATION WRITE CYCLE 74

FIGURE 6-17: PCI TARGET READ CYCLE 75

FIGURE 6-18: PCI TO WISHBONE READ CYCLE 75

FIGURE 6-19: PCI INITIATOR TO TARGET BURST READ CYCLE 76

FIGURE 6-20: PCI INITIATOR TO TARGET BURST WRITE CYCLE 76

FIGURE 6-21: WISHBONE WRITE TRANSFER CAUSED BY PCI TO WISHBONE WRITE CYCLE 76

FIGURE 6-22: RETRY ON WISHBONE BUS CAUSED BY PCI TO WISHBONE TRANSFER ... 77

FIGURE 6-23: ERROR ON WISHBONE BUS CAUSED BY PCI TO WISHBONE TRANSFER... 77

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 *SoC Interconnection 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.

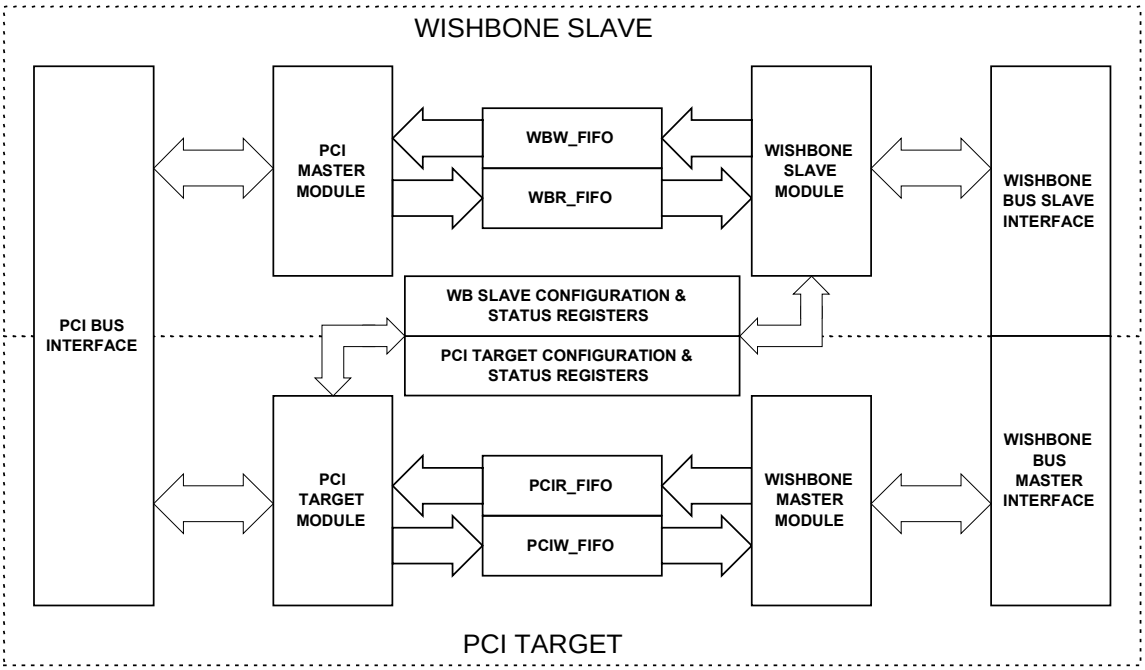


Figure 2-1: PCI bridge core architecture

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 WISHBONE bus address with the masked base address to identify valid WISHBONE cycles. If needed, each valid 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.

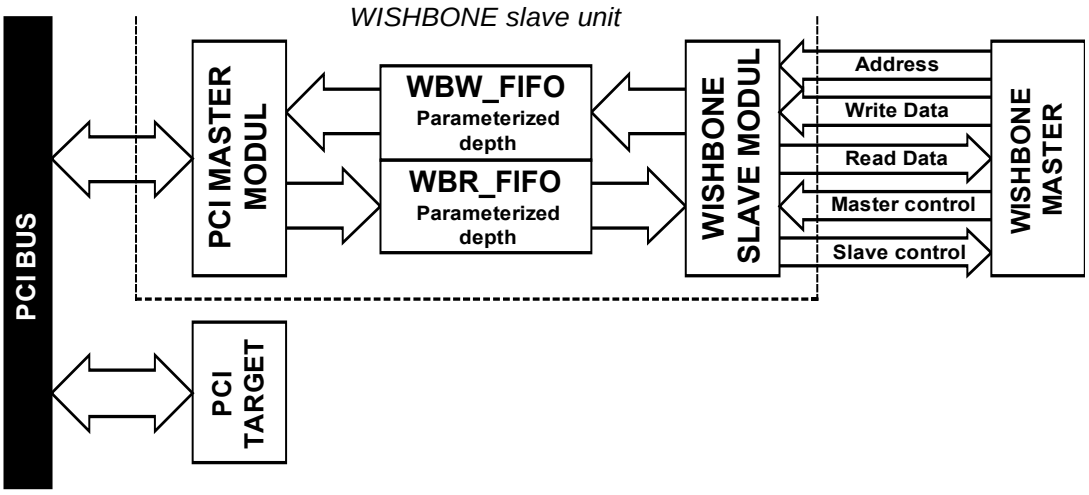


Figure 2-2: WISHBONE slave unit architecture

2.2.1.1 WISHBONE Slave Module

The WISHBONE slave module, which includes one to six image units for address translation from the WISHBONE bus, 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.

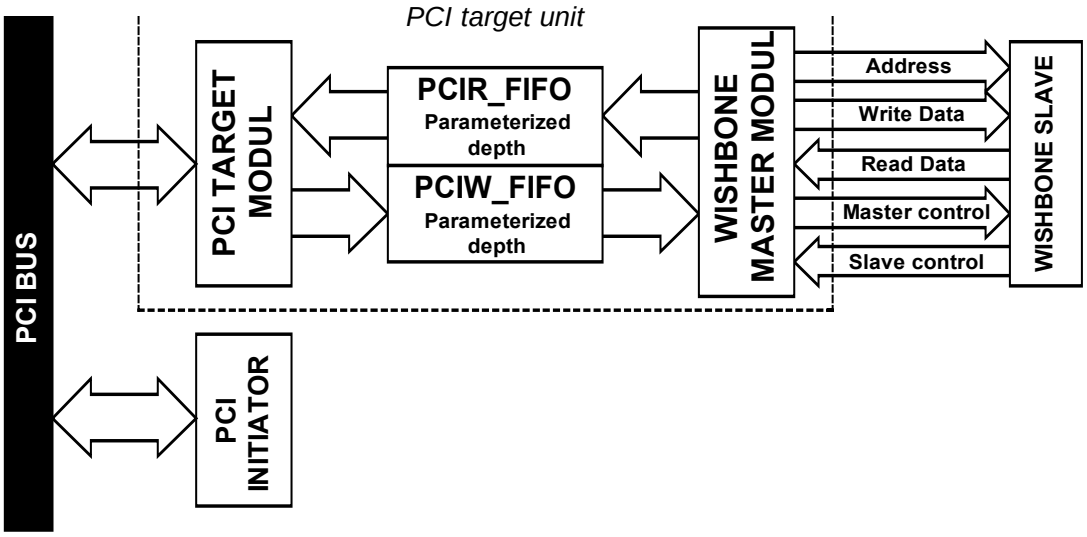


Figure 2-3: PCI target unit architecture overview

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 5.2 WISHBONE 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

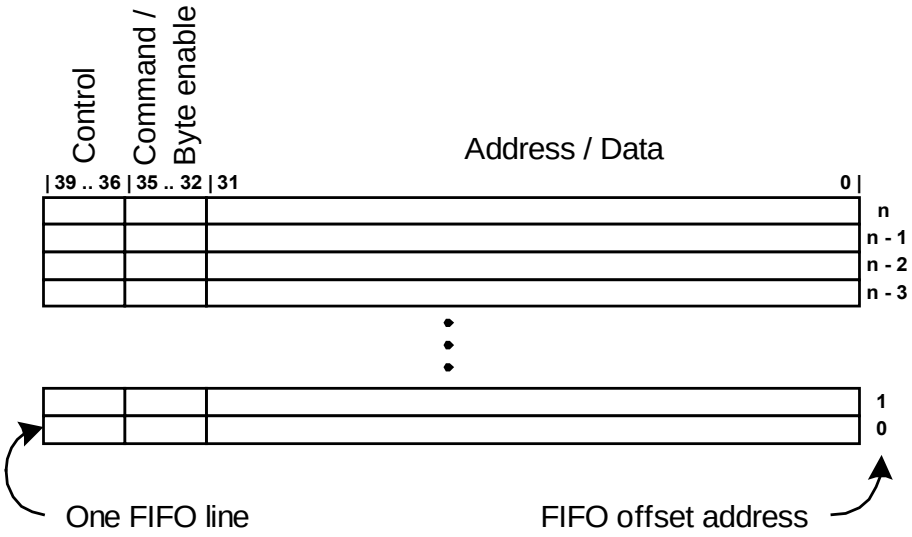


Figure 2-4: Detailed Description of FIFO Register Lines

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.

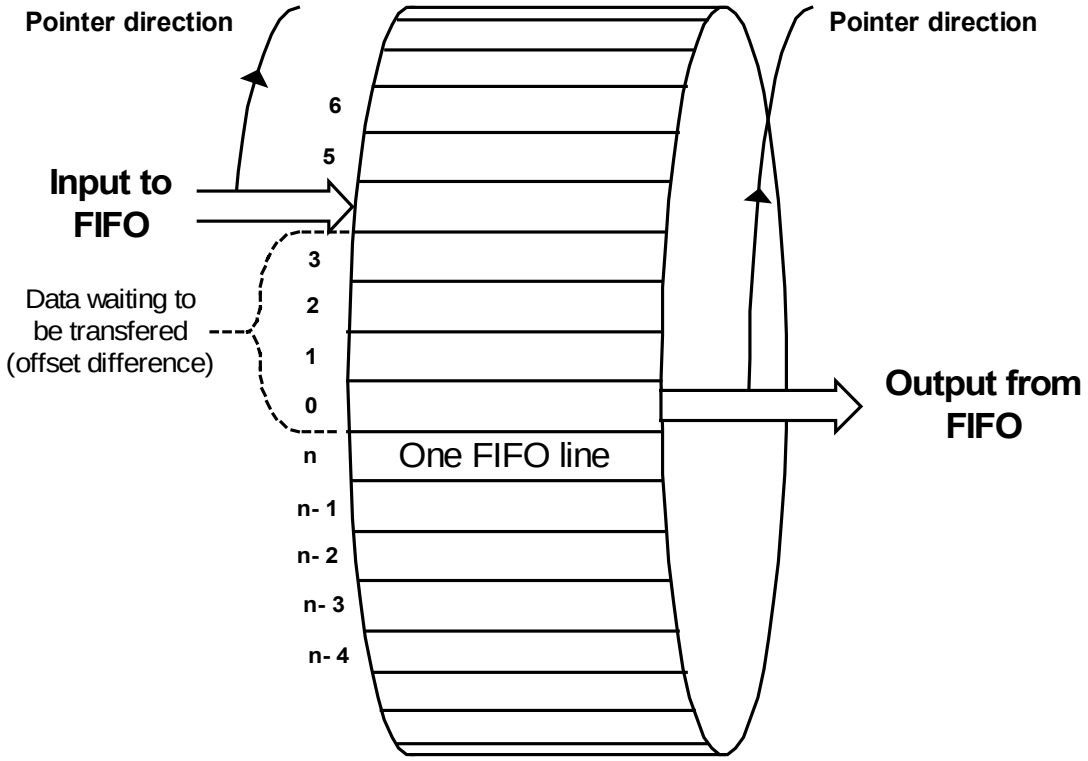


Figure 2-5: FIFO Architecture

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. Each image must have address translation logic (Figure 2-6) including its own set of 32-bit registers:

- Base Address register [31:0]
- Address Mask register [31:0]
- Translation Address register [31:0]
- Image Control register [31:0]

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).

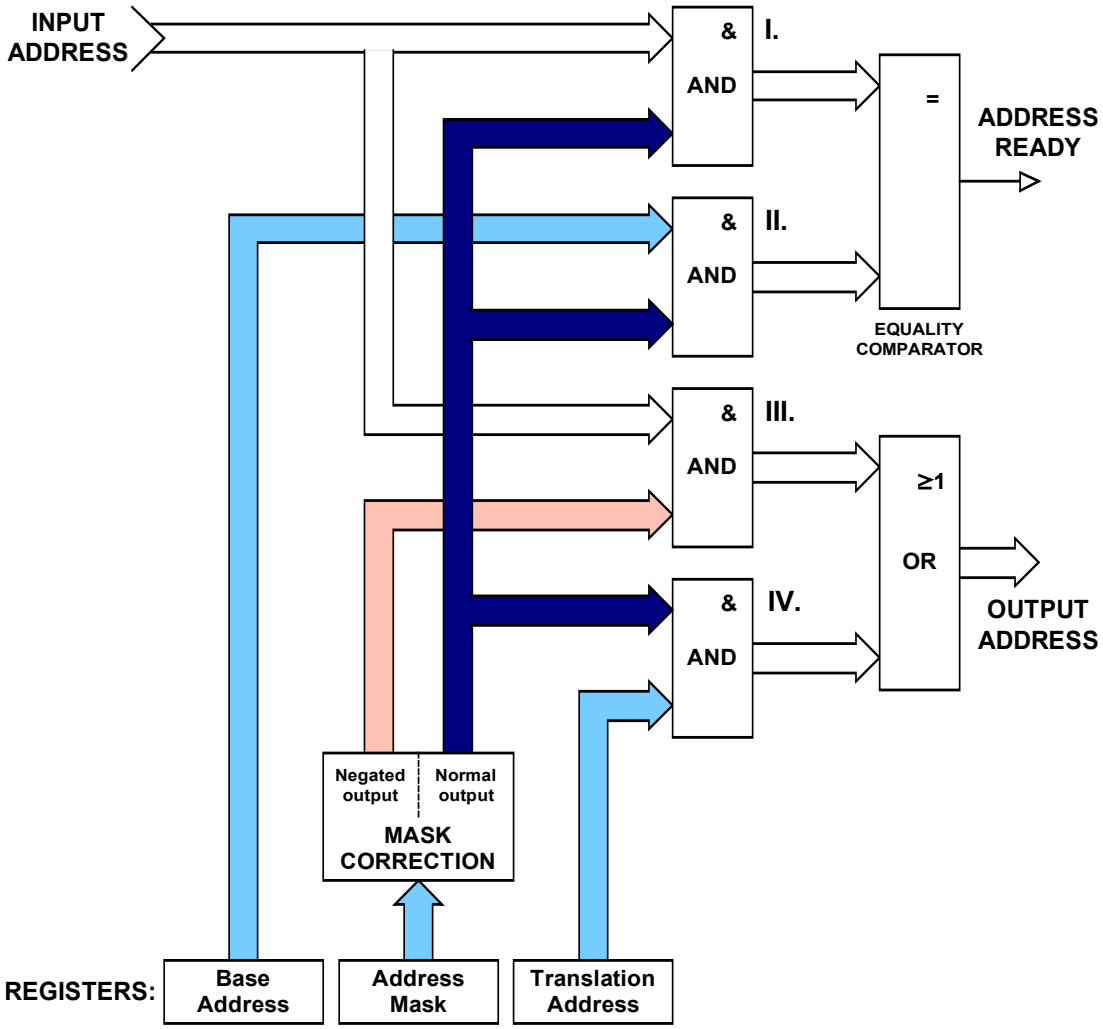


Figure 2-6: Address Translation Logic

3

Operation

3.1 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 from the PCI side).

All registers in the configuration space of a core are 32-bits wide with 8-bit granularity. All accesses must be 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. If any of them are non-zero, the WISHBONE slave module signals a bus error, while PCI target module stops burst after one data is transferred. To access individual bytes, the BE# signals for PCI bus access and the SEL_O signals for WISHBONE bus access must carry an appropriate value.

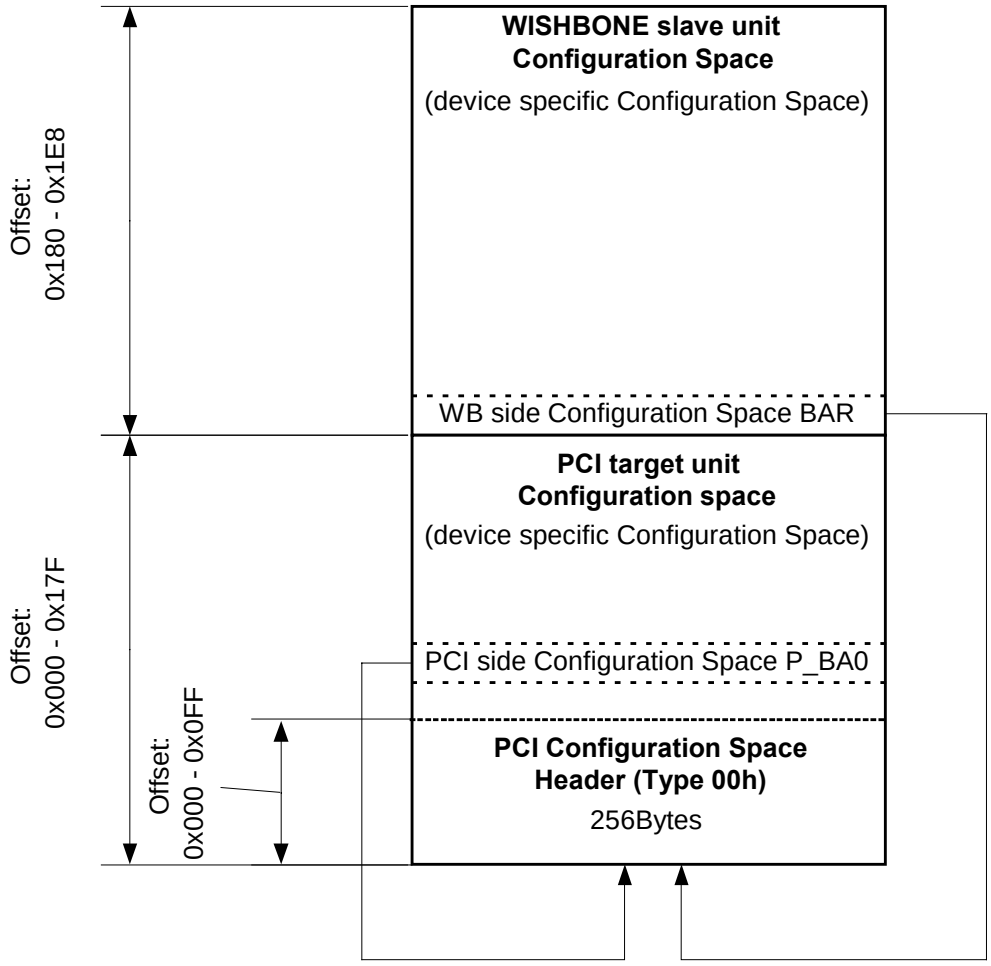


Figure 3-1: PCI Bridge Configuration Space

3.1.1 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 other PCI device can not read configuration space. See also 3.3.2 Addressing and Images of the PCI Target Unit and 4.1 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.).

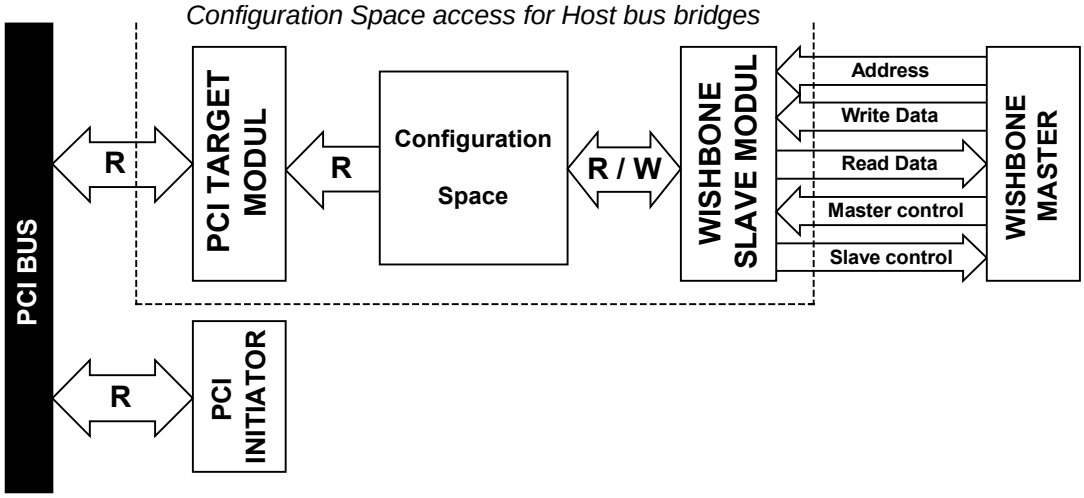


Figure 3-2: Configuration space access for Host Bus Bridges

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 side configuration space Base Address register to enable read-only access to PCI agents. The PCI target module provides read-only access to configuration space from the PCI bus, supporting Memory Read and Memory Write commands, but ignoring all other commands. The Memory Write command has no effect on Configuration registers. During the first data phase, the PCI target module signals **Target Disconnect with Data** to the initiator. Read cycles to non-implemented regions of configuration space return all 0s, whereas Write cycles have no effect.

3.1.2 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 side configuration space base address by performing a Type 0 configuration cycle and writing the base address to the PCI configuration space, as stated in the *PCI Local Bus Specification Rev. 2.2*. The PCI side configuration space Base Address register 0 holds the same value as the first Base Address register in the PCI configuration space Header at offset 0x10. This enables device-independent software to map the bridge configuration space anywhere into the memory address space. After the base address has been set by a Type 0 configuration cycle and the bridge is in a normal mode of

operation, the PCI agent can re-map configuration space anywhere within the memory space by writing to the PCI side configuration space Base Address register 0.

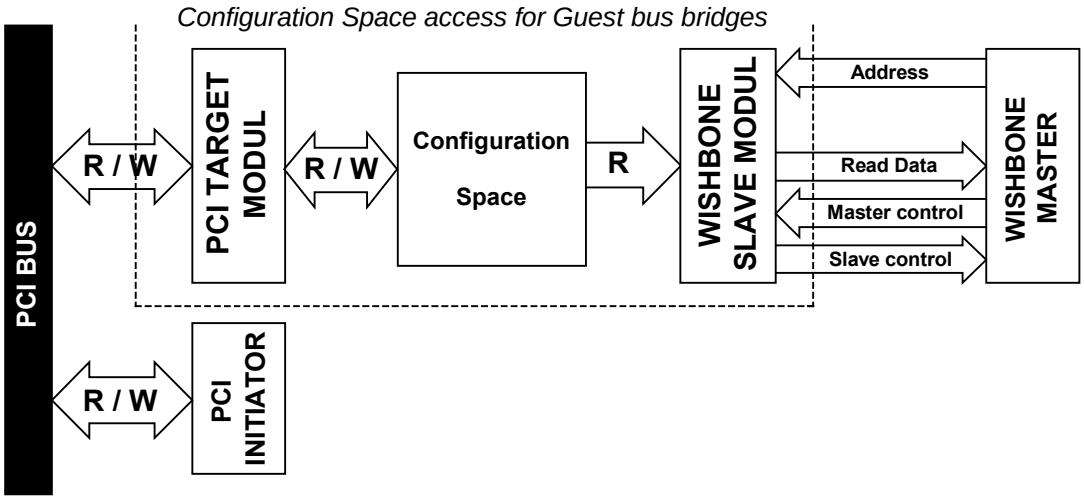


Figure 3-3: Configuration space access for Guest Bridges

Configuration space access can be provided to the PCI initiator as Memory Read or Memory Write. In case the PCI initiator requests configuration space access by using a different bus command, the PCI target module does not respond. If the PCI initiator attempts an access (Read or Write) to non-implemented space, the initial data phase is terminated by signaling **Target Disconnect with Data**. Write cycles have no effect on non-implemented configuration space, but Read cycles 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.

3.1.3 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 in configuration cycles is different 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").

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.

3.1.4 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 devices being a target of configuration cycles. The **Device** field, which identifies the address line that drives high for generating the IDSEL signal for a Type 0 configuration cycle, is set to the device number on the PCI bus. 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_O(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_O(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):

[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.

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
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 **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*.

3.1.5 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_O(31..0) and SEL(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.

3.2 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.

3.2.1 WISHBONE Slave Unit Functionality

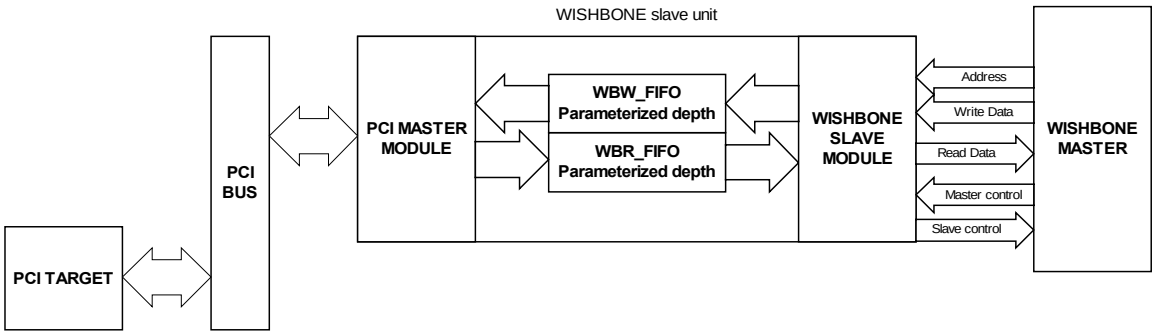


Figure 3-4: WISHBONE Slave Unit Architecture Overview

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

3.2.1.1 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.

3.2.1.2 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. WBW_FIFO also performs a different bus clock adaptation.

3.2.1.3 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.

3.2.1.4 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 5.1 PCI Interface provides a detailed overview of the PCI interface of the core.

3.2.2 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 different from the first Base Address register in the PCI header that is also used for the same configuration space, but different bus (PCI).

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 memory 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 P_Bax 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 ADDR_O(31..20) signals are compared to W_BA[31..20] values. If values match, the image is selected. In this case, ADDR_O(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_AT 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 ADDR_O(31..20) provided by the WISHBONE master with a value of 0x010 set in the W_AT 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

3.2.3 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 (RMF) cycles are not supported.

Note:
Serial block transfers (bursts) are still under discussion because the *WISHBONE Bus Specification* does not provide a mechanism to identify them. Until serial block transfers are specified, Block Write cycles will be handled as Single Write cycles. Proposal: It might be good to use an internal signal hardwired to a value indicating non-burst transfers. The definition of bursts in the *WISHBONE Specification* can be used to indicate whether a block transfer is serial or not. 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.

A Write cycle to an address range occupied by an image that is mapped into memory space must be DWORD-aligned (e.g. ADDR_O(1:0) must be 00), otherwise an error terminates the cycle on the WISHBONE bus.

A Write cycle to an address range occupied by an image that is mapped into I/O space must be byte-aligned. The following table describes valid SEL_O(3:0) encoding for different values on ADDR_O(1:0).

Value on ADDR_O(1:0) lines	Valid SEL_O(3:0) encoding
00	SEL_O(0) must be active

Value on ADDR_O(1:0) lines	Valid SEL_O(3:0) encoding
01	SEL_O(1) must be asserted, SEL_O(0) cannot be asserted
10	SEL_O(2) must be asserted, SEL_O(1:0) can't be asserted
11	SEL_O(3) must be asserted, SEL_O(2:0) cannot be asserted

Table 3-2: Valid ADDR_O(1:0) and SEL_O(3:0) combinations for I/O mapped address space access

All other combinations are invalid. An invalid access is terminated with **Error** on the WISHBONE bus.

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.
- An uncompleted Delayed Read request is still pending in a WISHBONE slave unit (Write cycles cannot be posted until a Read cycle finishes on the PCI bus).

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 commands, depending on the value of the address space-mapping bit (0 = memory, 1 = I/O) of the image 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 cycles 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.

Because all Write cycles are posted to and are therefore immediately acknowledged by the WISHBONE master, there is an alternate way of communicating errors signaled on the PCI bus when Posted Write cycles have already been written to their final destination: 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 proceed normally.

3.2.4 WISHBONE to PCI Read Cycles

Read cycles initiated by the WISHBONE master are handled as Single Delayed Read cycles. Multiple Delayed Read cycles are not supported. 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:

- The PREF_EN bit indicates that the address range of an image is prefetchable, which means that the bridge core can pre-fetch data from the target and store it in WBR_FIFO. This method increases the system performance since a Delayed Read transaction only knows the starting address of the transfer.
- The MRL_EN bit indicates that the PCI master module is free to use the Memory Read Line bus command for Burst Read cycles.
- When both PREF_EN and MRL_EN bits are set, the bridge will use the Memory Read multiple bus command on the PCI bus.
- Images mapped to I/O space handle any Read transaction as single Delayed Read cycle (not as Burst Read cycle). If the WISHBONE master attempts to perform a serial Block Read cycle from an I/O space mapped image, the cycle is terminated with an error by the WISHBONE slave module.
- The bridge core performs pre-fetched Read cycles only through images mapped to memory space. Prefetchable address space is assumed only when the PREF_EN bit or MRL_EN bit of the corresponding W_IMG_CTRLx register is set, and the WISHBONE master signals a serial Block Read cycle.

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 Block Read cycle, or the PREF_EN bit is cleared.

When the WISHBONE slave unit latches address and SEL(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 Read	0	0	Memory 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 prefechable are performed in one data phase on the PCI bus. Only those byte enables are active on the PCI bus, as SEL(3:0) data were active during the Read request. After the first data phase, the PCI master module releases the PCI bus.

All Delayed Reads from address space marked as prefetchable are performed in Burst Read cycles. Here are all byte enables active on the PCI bus, since PCI bridge can not determine, which bytes are significant for a WB device, that initiated the transaction (SEL(3:0) data were active during the Read request only for first data phase). 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 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 PCI target is capable to operate the following terminations:

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

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

Disconnect is a valid termination for Single Read cycles. The PCI master module does not retry these transactions but stores data for Single Read cycles and waits for the WISHBONE master to fetch it.

Target Abort is an error signaled to the WISHBONE master. Retrying the transaction, it receives a bus error termination (the WISHBONE slave module asserts ERR_I).

Master Abort is an error termination. The WISHBONE master receives an error when a transaction ends with **Master Abort** on the PCI bus. The only exception to this rule is a Configuration Read cycle, which returns all 0s.

Block Read length can be of Cache Line size or WBR_FIFO depth. It is the WBR_FIFO depth if Memory Read Multiple bus command is performed on the PCI bus (see Table 3-3, when this command is performed). But there is NO Block Read if Cache Line size is set to 1 or it is set to unsupported value (valid Cache Line sizes are multiples of 4 - see chapter 4.1.2).

3.3 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.

3.3.1 PCI Target Unit Functionality

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

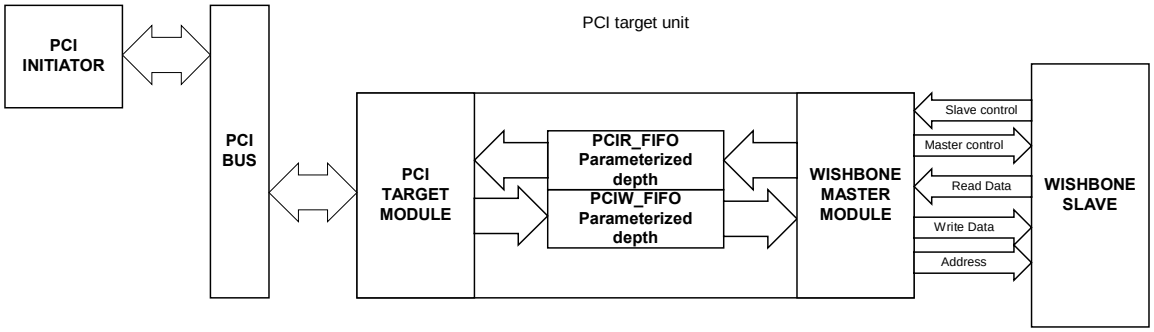


Figure 3-5: PCI target unit architecture overview

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

3.3.1.1 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.

3.3.1.2 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.

3.3.1.3 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.

3.3.1.4 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 5.2 WISHBONE Interface, describes in detail the WISHBONE interface of the core.

3.3.2 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 3.1.1 Configuration Space Access for Host Bus Bridges and 4.1 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

cycle 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 initial cycle on the PCI bus is placed within the memory 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 ADDR_O(31..20) signals are compared with a P_BA[31..20] value. If values match, the image is selected. In this case, ADDR_O(19..0) lines define an offset in an address range of 1MB.

Example 3-3: Address range of WISHBONE slave 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_AT 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 IMG_CTRL register is set to a value of 1 and of the corresponding P_AT 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_AT 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

3.3.3 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:
Serial block transfers (bursts on the WISHBONE bus) are still under discussion because the *WISHBONE Bus Specification* does not provide a mechanism to identify them. Until serial block transfers will be specified, all bursts from the PCI bus are handled as Block Write cycles.

All Write cycles from the PCI master to the WISHBONE bus are handled as Posted Writes. Due to this, the Read Modify Write command is not supported on the WISHBONE bus. 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 that 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, all 32 AD lines are used for full byte address decoding, and 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. Their data phase is terminated with **Target Disconnect with Data**. The PCI initiator then attempts to perform the next access with the data following the first transfer. This is repeated until the "burst" transfer has finished.

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 3-4: 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) bus) provide a DWORD-aligned address. The AD(1:0) lines are not part of the address decoded, but they indicate the order in which the PCI initiator requests data to be transferred, as seen in Table 3-5. The Linear Incrementing Burst mode is fully supported, while the Cache-line Wrap mode is broken into single transfers whose data phase is terminated with **Target Disconnect with Data**. The PCI initiator then attempts to perform

the next access with the data following the first transfer. This is repeated until the Cache-line Wrap mode burst transfer has finished.

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

Table 3-5: Burst Ordering combinations for memory mapped address space accesses

All other combinations are reserved (because of an earlier version of the *PCI Specification*). Therefore, accesses must be terminated with **Disconnect** after the first data phase, but requested memory address space is not affected. There are additional reasons for the PCI target to terminate a current bus-cycle.

When the PCI target unit is unable to respond within its subsequent latency requirement, it terminates a transfer with **Disconnect with/without Data** while data is being transferred, or immediately afterwards, on the initial data phase. This applies to the following conditions:

- The target is not capable of doing a burst (as mentioned above).
- The target is temporarily unable to continue bursting when PCIW_FIFO is already fulfilled with 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:

- An internal resource conflict emerges when PCIW_FIFO is full or cannot provide enough space to accommodate another burst transfer.
- The target is locked by another master when an uncompleted Delayed Read request is still pending in a PCI target unit (Write cycles cannot be posted until a Read cycle finishes on the WISHBONE bus).
- The target is locked by WBU, when there is still a WB to PCI delayed read pending or processing.

All PCI bus transfer terminations described above are PCI target terminations, but masters may also terminate transactions.

Regardless whether image mapping occurs to MEMORY or I/O space, the PCI initiator or target can insert wait cycles into the current Write transfer.

The PCI target module must perform address decoding every time the PCI initiator induces a Write transfer in order to determine if this transfer is related to it. The WISHBONE master module

initiates a transaction on the WISHBONE bus after a complete transaction has been stored in the PCIW_FIFO unit of the PCI target. The module uses Single Write or Block Write transfers, depending on the value of the control bit in the PCIW_FIFO line that indicates a burst from the PCI bus. Block Write cycles on the WISHBONE bus have the same length as bursts from the PCI. If a burst on the PCI bus was cut because of smaller PCIW_FIFO depth, the block size is as large as the size of the burst written into the PCIW_FIFO. When a PCI initiator completes a Burst Write cycle with the next access, it is treated as a new burst transfer written to PCIW_FIFO.

Because all Write cycles are posted and therefore immediately claimed by the PCI Target module and stored to the PCIW_FIFO, there is an alternate way of communicating errors signaled on a WISHBONE bus when Posted Write cycles actually have been written to their final destination—the mechanism of Error Reporting, which is provided through Error Reporting registers. The error enable bit (ERR_EN) of the PCI Error Control and Status register (P_ERR_CS) must enable this mechanism. If Error Reporting is enabled, errors can generate interrupts when the error interrupt enable bit (EINT_EN) of the P_ERR_CS register is 1. Each Error Reporting register stores part of the 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.

Only the Write transaction that generated an error is discarded, any subsequent transactions are processed normally.

3.3.4 PCI to WISHBONE Read Cycles

Read cycles induced by the PCI initiator are handled as Single Delayed Read cycles. This explains why the Read Modify Write command on the WISHBONE bus as well as Multiple Delayed Read cycles are not supported. Delayed transactions must be completed on the WISHBONE bus before they can complete on the PCI bus.

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 a current cycle. Handling Read transactions is encoded in the PCI Image Control register (P_IMG_CTRLx). To define the PCI target unit's behavior towards images mapped to memory space during Read transactions, several options exist:

-
- The PREF_EN bit indicates that the address range of the PCI memory image is prefetchable, which means that the bridge core can pre-fetch data from the slave and store it in WBR_FIFO. This method increases system performance since the Delayed Read transaction has information on starting address of the transfer only. Read below Table 3-6 for valid byte enables information.

- Images mapped to I/O space handle any Read transaction as Single Delayed Read cycle (no bursts). If the PCI initiator attempts a Burst Read cycle from an image mapped to I/O space, the cycle is terminated with **Disconnect with Data**; thus the initiator can continue reading the rest of the data (by disconnecting the bursts).
- The bridge core performs pre-fetched Reads only through images mapped to memory space. Prefetchable address space is assumed for the following conditions:
- The PREF_EN bit of the corresponding P_IMG_CTRLx register is set (see Table 3-6).

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

- Accesses to I/O mapped address space are always non-prefetched.
- The PCI initiator performs a Single Read cycle, and the PREF_EN bit is cleared.

The following table shows PCI bus Read commands that are considered single or block transfers regarding the PREF_EN bit.

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
	Memory Read Line	X	Block Read
	Memory Read Multiple	X	Block Read

Table 3-6: Bus command encoding for Read cycles through PCI target module

Single Read cycles are performed in one data phase on the WISHBONE bus. Only those byte enables (SEL(3:0)) are active on the WB bus, as PCI byte enables were active during the Read request. After the first data phase, the WISHBONE master module releases the WISHBONE bus.

All Delayed Read cycles from memory address space marked as prefetchable (and MRL and MRM commands) are performed as Block Read cycles. Here are all byte enables active on the WB bus, since PCI bridge can not determine, which bytes are significant for a PCI device, that initiated the transaction (PCI byte enables were active during the Read request only for first data phase). The WISHBONE master module reads data from the 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:

- PCIR_FIFO is full.
- The WISHBONE slave issues **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. When PCIR_FIFO is empty or the PCI initiator issues the Read cycle of an address that is not one DWORD higher than the previous address within the same block transfer, the PCI target module latches information about a new Read request and terminates the cycle with **Retry**.

Any data left in PCIR_FIFO is flushed immediately.

Until now, PCI TO WISHBONE reads have been described as though all of them are completed successfully, but it is common for WISHBONE bus slaves or masters to generate error terminations. Terminations from the WISHBONE bus must be propagated to the PCI bus in to let the PCI initiator know what happened with the initiated transaction.

Following terminations are possible through WISHBONE slaves:

- **Retry**, which is not propagated back to the PCI bus. The bridge core simply retries the transaction.
- **Error** is a termination signaled to the PCI initiator. Retrying the transaction, the PCI initiator receives **Target Abort**.

There are additional reasons for the PCI target to terminate a current bus cycle.

The PCI target unit terminates a transfer with **Disconnect with/without Data** while data is being transferred, or immediately afterwards, on the initial data phase, when it is unable to respond within its subsequent latency requirement:

- The target is not capable of doing a burst (reading from I/O mapped space, as mentioned above).
- The target is temporarily unable to continue bursting when PCIR_FIFO is cleared of the current Burst Read cycle.

The PCI target unit abnormally terminates a transfer with **Target Abort** (otherwise it will never be able to complete the requested transfer) if the master initiates a non-valid combination of AD(1:0) and BE#(3:0) when accessing the I/O mapped image space.

The PCI target unit terminates a transfer with **Retry** before any data is transferred when it is busy and temporarily unable to process the transaction. An internal resource conflict emerges when PCIR_FIFO is empty.

All PCI bus transfer terminations described above are PCI target terminations, but masters may also terminate transactions.

Regardless whether image mapping occurs to MEMORY or I/O space, the PCI initiator or target can insert Wait cycles into the current Write transfer.

On the other side of the PCI target module, the side of the WISHBONE master unit, the WISHBONE slave can also insert Wait cycles.

Block Read length can be of Cache Line size or PCIR_FIFO depth. It is the PCIR_FIFO depth if Memory Read Multiple bus command was performed on the PCI bus (see Table 3-6, for all read command). But there is NO Block Read if Cache Line size is set to 1 or it is set to unsupported value (valid Cache Line sizes are multiples of 4 - see chapter 4.1.2).

3.4 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 is pending or waiting to be retried by the WISHBONE master, 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. Posted Write cycles from WBW_FIFO are processed until WBW_FIFO is empty.
4. The PCI master module completes a Read cycle on the PCI bus.
5. When a Read cycle is complete (e.g. when it becomes a Delayed Read completion), Posted Write cycles are accepted again in WBW_FIFO.
6. The PCI target module retries all non-configuration space accesses from the PCI bus.
7. All Posted Write cycles from PCIW_FIFO are completed on the WISHBONE bus until PCIW_FIFO is empty.
8. The WISHBONE slave unit allows a Read cycle to be completed on the WISHBONE bus.
9. If the Read cycle not complete, WISHBONE slave and PCI target unit allow the posting of Write cycles.

3.5 Parity

Parity monitoring and generation is required by all PCI agents according to the *PCI Local Bus Specification*. The PCI master module monitors a PAR signal during Read cycles and drives it during Write cycles. The PAR signal provides even parity through C/BE# [3:0] and AD [31:0] lines during address and data phase. If the PCI master performs a Write cycle, the target is responsible for monitoring PAR and asserting PERR# if an error is detected. During Read cycles, the PCI master module monitors PAR and asserts PERR# if an error is detected. If a master detects a parity error during a Read transaction or samples the PERR# signal asserted during a Write transaction, it must set the parity error detected bit in its configuration space Status register.

If the parity error response bit is set, the PCI master module must signal a parity error by asserting the PERR# signal during Read transactions.

When the PERR_INT_EN bit is set, the core signals an interrupt request in an additional response to parity errors, as recommended by the *PCI Bus Specification*. Parity error detection has no influence on the PCI master module—it continues the transaction until finished or until terminated by the target.

3.6 Interrupts

The PCI IP core is capable of generating interrupts in response to different events. Interrupt Control and Interrupt Status registers control these interrupts. If the core is implemented as a guest bridge,

interrupts are reported 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. See chapter 4.1.4 what must be enabled to cause appropriate interrupts and which are implemented.

The software must locate and clear the source of an interrupt request before clearing status bits in a bridge core. When a reported error caused an interrupt, error must be cleared before interrupt.

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

* – All 4 PCI Image0 Control and Address registers are implemented when the PCI bridge is implemented as HOST and PCI Image0 is used to access WB bus or none of all 4 PCI Image0 Control and Address registers are implemented if the PCI bridge is implemented as HOST and PCI Image0 is canceled. Otherwise only the PCI Image0 Base Address register (P_BA0) is implemented on the same offset address and is used for access to the entire Configuration Space (see also 3.1.1 Configuration Space Access for Host Bus Bridges and 3.3.2 Addressing and Images of the PCI Target Unit).

Table 4-1: List of registers

4.1.1 WISHBONE Slave Unit Control & Status

The registers of the WISHBONE slave unit start at offset 0x180 from the base address. The base address is pre-defined during the design phase for WISHBONE bus accesses; the base address for the PCI bus is defined with a configuration cycle for Guest Implementation or with writing to this register by the WISHBONE master for Host Implementation (see also chapter 3.1, "Configuration Space").

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 4-2: WISHBONE configuration space Base Address register

Register layout:

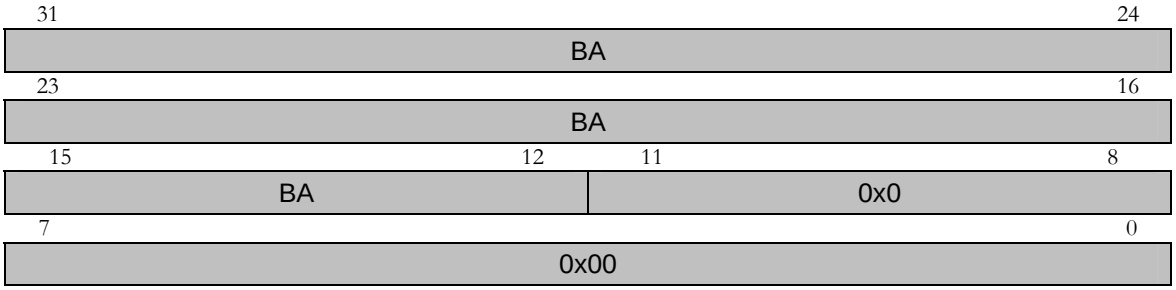


Figure 4-1: WISHBONE configuration space Base Address register layout

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 behavior when an image is selected and enabled.

Table 4-3: WISHBONE Image Control register

Register layout:

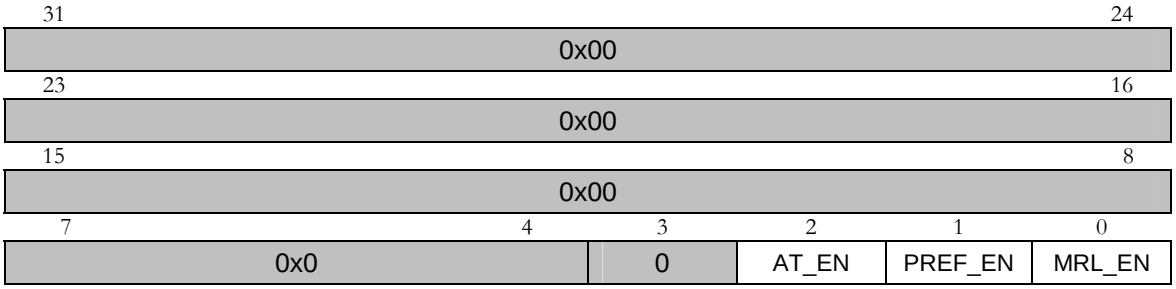


Figure 4-2: WISHBONE Image Control register layout

Bit descriptions:

Bit #	Access	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.
0	Memory Read Line Enable	When the WISHBONE master performs block read cycles, this bit enables the usage of memory access optimizing commands. If the prefetch-enable bit is also set, read will be performed using Memory Read Multiple command,

Bit #	Access	Description
		otherwise the Memory Read Line command will be used.

Table 4-4: WISHBONE Image Control register bit descriptions

Base Address registers: W_BA1- W_BA5

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

Table 4-5: WISHBONE Base Address register

Register layout:

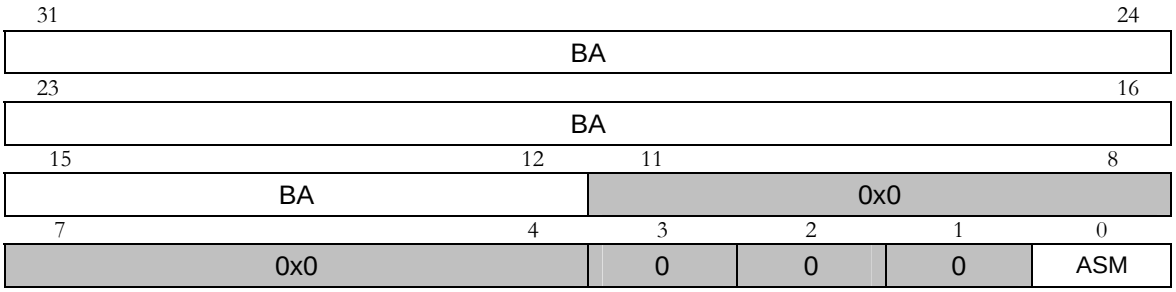


Figure 4-3: WISHBONE Base Address register layout

Bit descriptions:

Bit #	Access	Description
31 – 12	Base Address	Image base address. How many bits from this field are compared with ADDR_I(31:0) is defined in the Address Mask register.
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. 0 – Memory space mapping 1 – I/O space mapping

Table 4-6: WISHBONE Base Address register bit descriptions

Address Mask registers: W_AM1 – W_AM5

Width	Access	Reset	Description

Width	Access	Reset	Description
32	RW	0x00000000	This register value represents an address mask. If the corresponding bit is 1, the address line in the same position is compared with the value in the Base Address register. If the bit is 0, the corresponding address line is not compared with the value in the BA register.

Table 4-7: WISHBONE Address Mask register

Register layout:

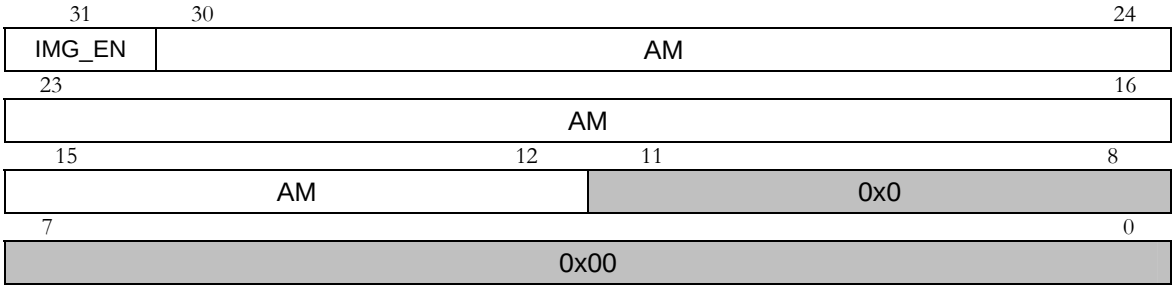


Figure 4-4: WISHBONE Address Mask register layout

Bit descriptions:

Bit #	Access	Description
31	Image Enable & Address Mask (31)	This bit must be set to enable an image. If 0, the corresponding image is not enabled. This bit is also used in Address Masking, i.e. how a limit of 2GB per image is implemented (at least ADDR_I(31)) must be compared with BA for each image.
30 – 12	Address Mask	The remainder of the Address Mask. If bit(x) of the address mask is 1, ADDR_I(x) is compared with the BA(x) bit in the Base Address register; otherwise it is not.
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 4-8: 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, compared address lines from the WISHBONE bus (specified with AM value) are replaced by corresponding values in this register for PCI bus accesses.

Table 4-9: WISHBONE Translation Address register

Register layout:

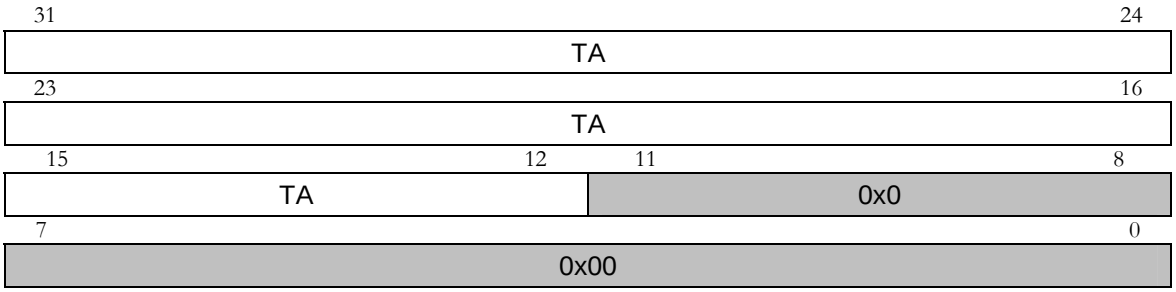


Figure 4-5: WISHBONE Translation Address register layout

Bit descriptions:

Bit #	Access	Description
31 – 12	Translation Address	This register value is used when address translation is enabled. Each value on ADDR_I lines not masked by AM register setting is replaced by the corresponding bit value of the Translation Address register for PCI bus accesses.
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 4-10: 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	16	15	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. 0FFFFh 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 4-11: 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	√	Signaled **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 4-12: Status register of PCI configuration header

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

- The **Cache Line Size register** specifies the size of burst reads, except for a Memory Read Multiple command, which size is Read_FIFO depth. Valid values for this register are multiples of 4 (including 1). In invalid value is written (including 0), then the value of 1 is assumed by both WBU and PCIU and no burst reads are 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 are the same and also accessed in the PCI part of the Configuration space. Each one of them consists of a 28-bit base address for MEMORY mapping or a 30-bit base address for I/O mapping. Here are only up to 20 MSBits implemented. Other bits are control bits and described in the following table.

Bit descriptions:

Bit #	Description
31 – 4	Base address (only the upper 20 bits are valid)
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 4-13: Base Address register of PCI configuration header for memory mapped space

Bit descriptions:

Bit #	Description
31 – 2	Base address (only the upper 20 bits are valid)
1	Reserved
0	I/O space indicator = '1' (always for I/O mapped space)

Table 4-14: Base Address register of PCI configuration header for I/O mapped space

4.1.2.2 PCI Image Control and Address Registers

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

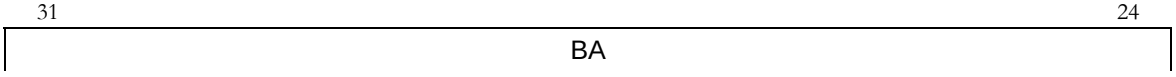
The only exception is the set of 4 PCI Image0 Control and Address registers, which is implemented only when the PCI bridge is implemented as HOST and all Image0 is used to access WB bus (see Table 4-16, Table 4-18, Table 4-20 and Table 4-22). Otherwise, there are five possible configurable PCI target images (PCI image1 – PCI image5), and only the PCI Image0 Base Address register (P_BA0) is implemented for the PCI image0 on the same offset address and is used for access to the entire Configuration Space (see Table 4-15 and Figure 4-7). The other 3 registers are not implemented and therefore cannot be written to (see also 3.1.1 Configuration Space Access for Host Bus Bridges and 3.3.2 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 4-15: PCI Image0 Base Address register

Register layout:



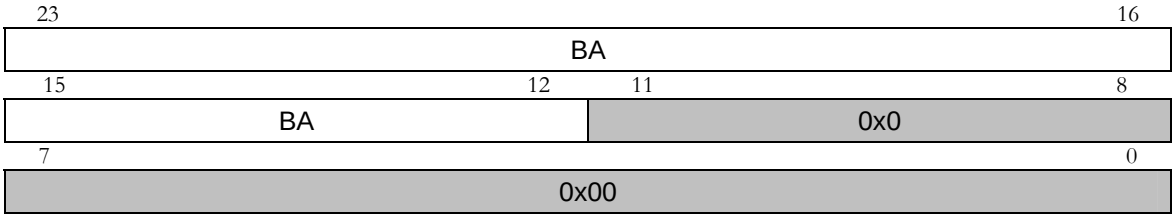


Figure 4-7: PCI Image0 Base Address register layout – Image0 used for accessing the PCI Configuration Space Header (type 00h)

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 4-16: PCI Image Control Register

Register layout:

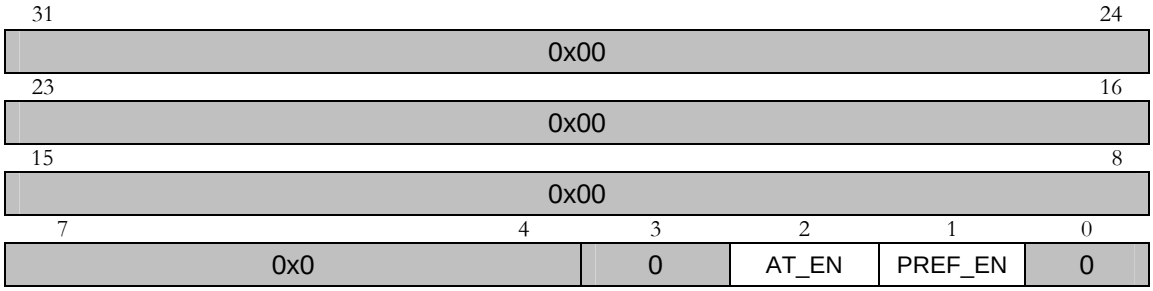


Figure 4-8: PCI Image Control register layout

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 4-17: 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 4-18: PCI Base Address register

Register layout:

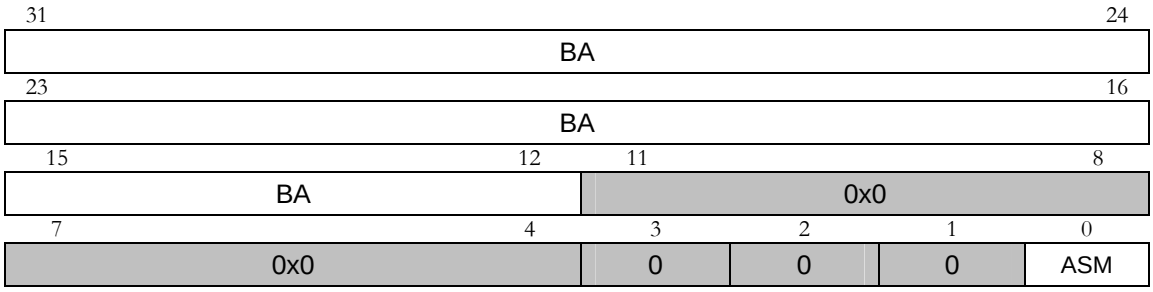


Figure 4-9: PCI Base Address Register Layout

Bit descriptions:

Bit #	Name	Description
31 – 12	Base Address	Image base address. The Address Mask register defines how many bits from this field are compared with ADDR_I(31:0).
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 chapter 3.3.2 Addressing and Images of the PCI Target Unit). 0 – Memory space mapping 1 – I/O space mapping

Table 4-19: PCI Base Address register bit descriptions

Address Mask registers: P_AM0 (P_AM1) – P_AM5

Width	Access	Reset	Description
32	RW	0x00000000	The register value represents the address mask. If the corresponding bit is 1, the address line in the same position is compared with a value in the Base Address register. If the bit is 0, the corresponding address line is not compared with a value in the BA register.

Table 4-20: PCI Address Mask register

Register layout:

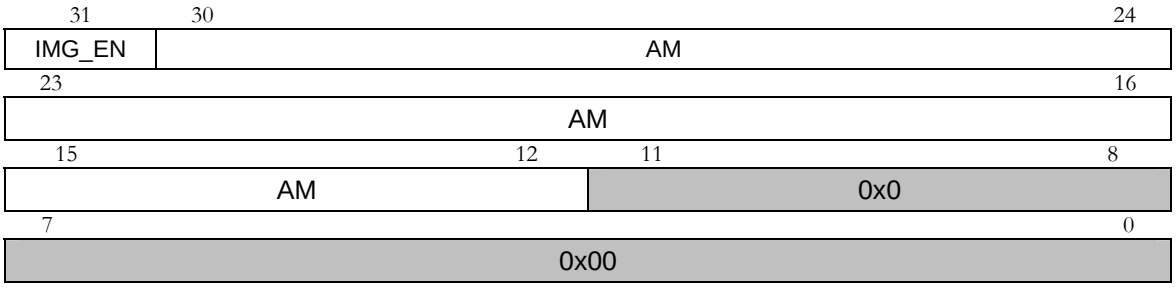


Figure 4-10: PCI Address Mask register layout

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. This bit is also used in Address Masking, i.e. how a limit of 2GB per image is implemented (at least ADDR_I(31) must be compared with BA for each image).
30 – 12	Address Mask	This is the remainder of the Address Mask. If bit(x) of the address mask is 1, then ADDR_I(x) is compared with the BA(x) bit in the Base Address Register; otherwise it is not.
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 4-21: 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 4-22: PCI Translation Address register

Register layout:

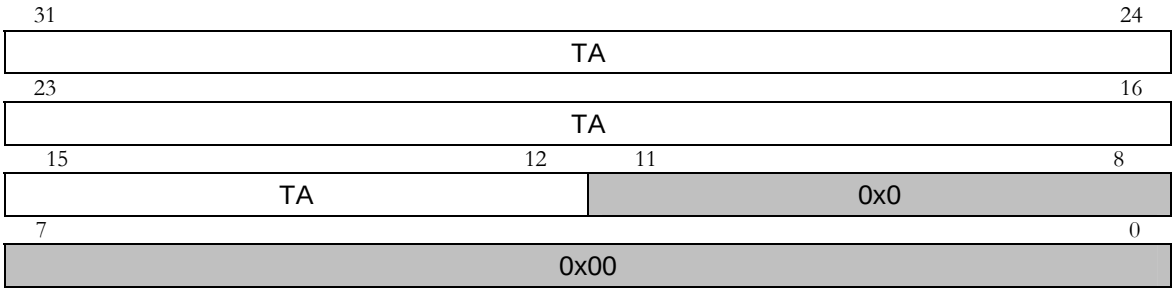


Figure 4-11: PCI Translation Address register layout

Bits descriptions:

Bit #	Name	Description
31 – 12	Translation Address	This register value is used when address translation is enabled. Each value on ADDR_I lines that is not masked by AM register setting is replaced. by the corresponding bit value of the Translation Address register for WISHBONE bus accesses.
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 4-23: 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

Width	Access	Reset	Description
			during the completion of a Posted Write cycle on the PCI bus.

Table 4-24: WISHBONE Error Control and Status register

Register layout:

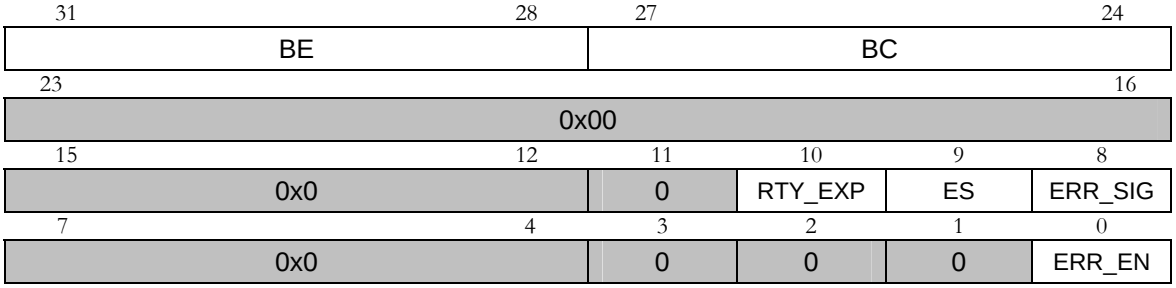


Figure 4-12: WISHBONE Error Control and Status register layout

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.
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 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. See description of bit 10! 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. While this bit is set, all WISHBONE slave unit operation is frozen. Software must clear this bit to enable transactions to resume their path through the WISHBONE slave unit. A bit is cleared

Bit #	Name	Description
		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—the transaction that caused an error is discarded, other transactions continue normally.

Table 4-25: 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 4-26: 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 4-27: 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 4-28: PCI Error Control and Status register

Register layout:

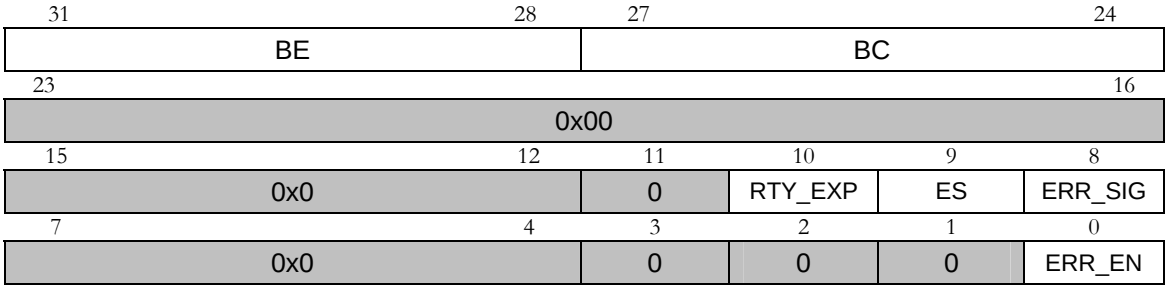


Figure 4-13: PCI Error Control and Status register layout

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

Bit #	Name	Description
		performed – the transaction that caused an error is discarded, other transactions continue normally.

Table 4-29: 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 4-30: 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 4-31: 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 cycles from or writes cycles 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 (e.g. it is used within a host PCI device).

Table 4-32: Configuration Address register

Register layout:

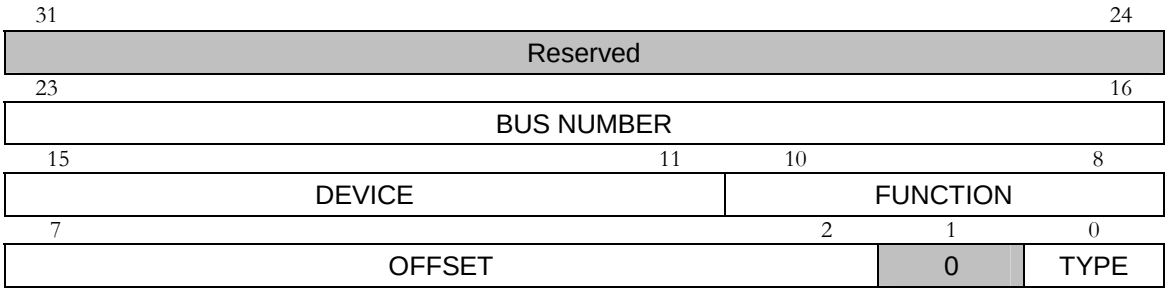


Figure 4-14: Configuration Address register layout

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 XY 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 4-33: Configuration Address register bit descriptions

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.

Configuration data: CNF_DATA

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

Table 4-34: 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 4-35: 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 4-36: Interrupt Control register

Register layout:

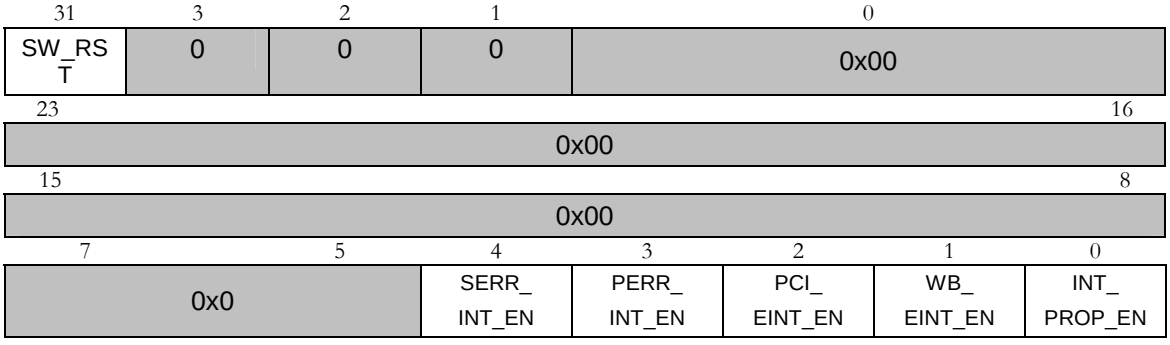


Figure 4-15: Interrupt Control register layout

Bit descriptions:

Bit #	Name	Description

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 Interrupt Enable	If set, this bit enables interrupt request generation when an 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 4-37: Interrupt Control Register bit descriptions

Interrupt Status Register: ISR

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

Table 4-38: Interrupt Status register

Register layout:

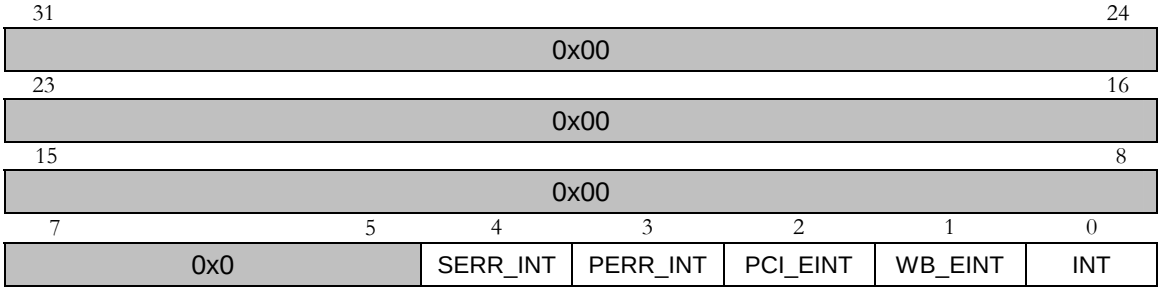


Figure 4-16: Interrupt Status register layout

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

Bit #	Name	Description
		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.*

* 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).

Table 4-39: Interrupt Status register bit descriptions

4.2 Software obligations

If bridge is implemented as GUEST, software (running on a PCI host device) should during initialization phase first set the registers in the PCI Configuration Space Header with configuration cycles (PCI Base Addresses must be mapped without interleaving and not used images must be disabled and finally PCI Master and Target Memory and/or IO spaces must be enabled). Other control and status registers and WBU registers can now be set through Image0 with memory cycles.

If bridge is implemented as HOST, software (running on a WB host device) should during initialization phase first set the PCI and WB Image registers and other status and control registers (PCI and WB Base Addresses must be mapped without interleaving and not used images must be disabled and finally PCI Master and Target Memory and/or IO spaces must be enabled).

When system is running, software MUST NOT just change any Image related registers, but must first disable appropriate Unit (WBU or PCIU or both), otherwise no one can determine what transactions were processed and where (e.g. if Translation Address is changed when the opposite side is writing through that Image).

If bridge is implemented as HOST, WB device can always access Configuration space and change WB Image related registers, since WB device is accessing Configuration space. Before changing the PCI Image relate registers, WB device must first disable PCI Target Memory and IO spaces.

If bridge is implemented as GUEST, PCI host device can always access Configuration space and change PCI Image related registers, since PCI device is accessing Configuration space. Care must be taken, if PCI host device wont to change PCI Base Address 0 (through which it is accessing Configuration space). This must be done with PCI configuration cycles. Before changing the WB Image relate registers, PCI host device must first disable PCI Master operation.

5

IO Ports

5.1 PCI Interface

The PCI interface contains both required and optional pins. All of them are organized in functional groups. Required pins must be implemented but there is also a description of implemented optional pins (needed for requested features).

5.1.1 Required PCI Interface Pins

Port	Width	Direction	Description
AD	32	I/O	Multiplexed address and data bus (little endian)
C/BE#	4	I/O	Multiplexed command and byte enable bus (This bus indicates a PCI command during address phases and Byte Enables during data phases.)
PAR	1	I/O	Parity bit

Table 5-1: PCI address and data pins

Port	Width	Direction	Description
FRAME#	1	I/O	Start and end of a transaction
IRDY#	1	I/O	Initiator ready (The assertion of this signal indicates that the initiator is ready to send or receive data.)
DEVSEL#	1	I/O	Device selected (When a target recognizes its address on the bus it asserts this signal to claim the transaction.)
TRDY#	1	I/O	Target ready (The assertion of this signal indicates that the target is ready to send or receive data.)
STOP#	1	I/O	Stop (This pin is used by a target to signal various

Port	Width	Direction	Description
			terminating conditions.)
IDSEL	1	I	Individual device select (This signal is used for configuration and requests a unique IDSEL line per agent.)

Table 5-2: PCI interface control pins

Port	Width	Direction	Description
PERR#	1	I/O	Parity error
SERR#	1	I/O	System error

Table 5-3: PCI error reporting pins

Port	Width	Direction	Description
REQ#	1	O	Asserted by initiator to request bus ownership
GNT#	1	I	Asserted by Arbiter to grant bus ownership

Table 5-4: PCI arbitration pins (INITIATOR only)

Port	Width	Direction	Description
CLK	1	I	PCI input clock (Signals are sampled on the rising edge of the clock.)
RST#	1	I/O	Asynchronous reset (The PCI device must tri-state all signals during reset.)

Table 5-5: PCI system pins

5.1.2 Implemented Optional PCI Interface Pins

Port	Width	Direction	Description
INTA#	1	O	Asserted by initiator to request an interrupt.

Table 5-6: PCI interrupt pin

Port	Width	Direction	Description
M66EN	1	I	Mode 66 MHz Enable (This signal indicates to a device whether the bus segment is operating at 66 or 33 MHz.)
CLKRUN#	1	I/O/Z	Clock running (This is the central resource request permission to stop or slow down **CLK**. The central resource must provide the pull-up for **CLKRUN**#.).
PME#	1	O	Power Management Event (This signal can be used by a device to request a change in the device or system power state. The assertion and deassertion of **PME#** is asynchronous to **CL**K.)

Table 5-7: PCI interface control pins

5.2 WISHBONE Interface

The SoC interface is a WISHBONE Rev. B compliant interface. The WISHBONE slave unit of the PCI IP core is connected to the WISHBONE bus as a slave while the PCI target unit connects to the WISHBONE bus as a master.

Port	Width	Direction	Description
ADDR_O	32	O	Address output
MDATA_I	32	I	Data input
MDATA_O	32	O	Data output
SEL_O	4	O	WE_O asserted indicates valid bytes on the MDATA_O bus WE_O deasserted indicates which bytes must be supplied by slave on MDATA_I bus
WE_O	1	O	Write enable indicates a Write cycle when asserted high and a Read cycle when low
CYC_O	1	O	Encapsulates a valid transfer cycle
STB_O	1	O	Indicates a valid transfer to the slave
ACK_I	1	I	Acknowledgment input slave signals a normal cycle termination
ERR_I	1	I	Slave signals abnormal cycle termination
RTY_I	1	I	Slave signals that the interface is not ready and that the master should retry the operation
CAB_O	1	O	Indicates to the slave that consecutive address block transfer is in progress

Table 5-8: PCI target unit's WISHBONE interface (master)

Port	Width	Direction	Description
ADDR_I	32	I	Address input
SDATA_I	32	I	Data input
SDATA_O	32	O	Data output
SEL_I	4	I	WE_O asserted indicates valid bytes on MDATA_I bus WE_O deasserted indicates which bytes must be supplied on MDATA_O bus.
WE_I	1	I	Write enable – indicates a Write cycle when asserted high and a Read cycle when asserted low
CYC_I	1	I	Encapsulates a valid transfer cycle
STB_I	1	I	Indicates a valid transfer to the slave
ACK_O	1	O	Acknowledgment output – slave signals a normal cycle termination
ERR_O	1	O	Slave signals abnormal cycle termination
RTY_O	1	O	Slave signals that the interface is not ready and that the master should retry the operation
CAB_I	1	I	Master signals consecutive address block transfer, which is in progress when 1

Table 5-9: WISHBONE slave unit's WISHBONE interface (slave)

Port	Width	Direction	Description
CLK_I	1	I	Clock input (application side clock)
RST_I	1	I	Reset input (application side reset)
RST_O	1	O	Used for propagating RST# from PCI bus to application side of the bridge; also used for initiating software reset
INTA_O(*)	1	O	Interrupt output
INTA_I(*)	1	I	Interrupt input

(*) These two signals will never be used at the same time. Guest Implementation of the core will signal interrupts to the PCI bus, so only INTA_I is used. Host Bridge Implementation will signal interrupts to the WISHBONE bus, so INTA_O is used.

Table 5-10: WISHBONE common control and system I/Os

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

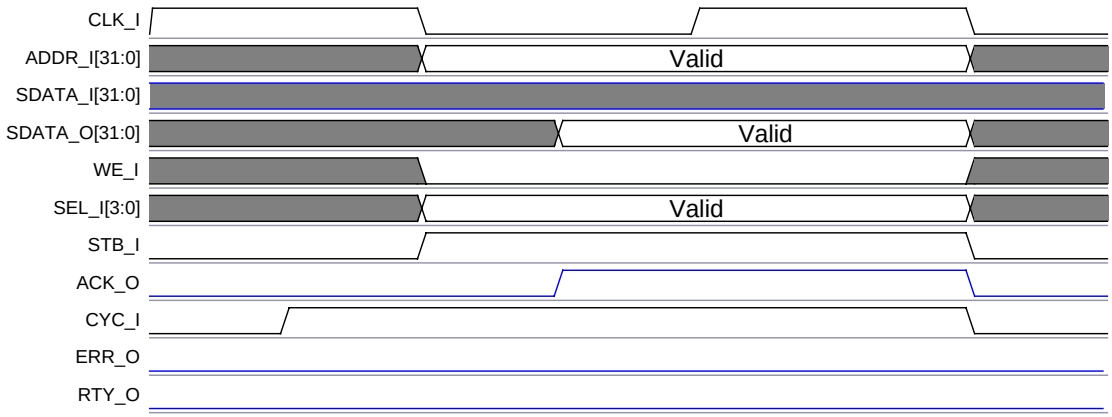


Figure 6-1: WISHBONE configuration Read cycle

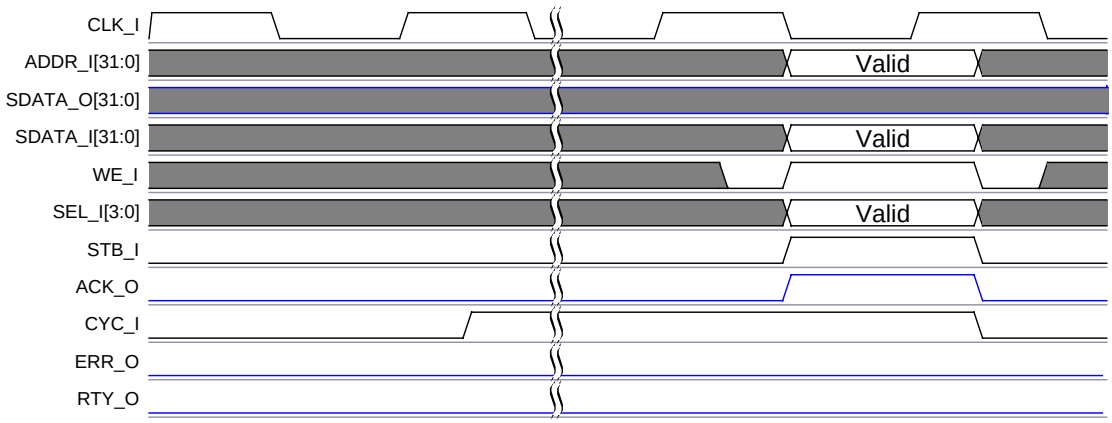


Figure 6-2: WISHBONE Configuration Write cycle

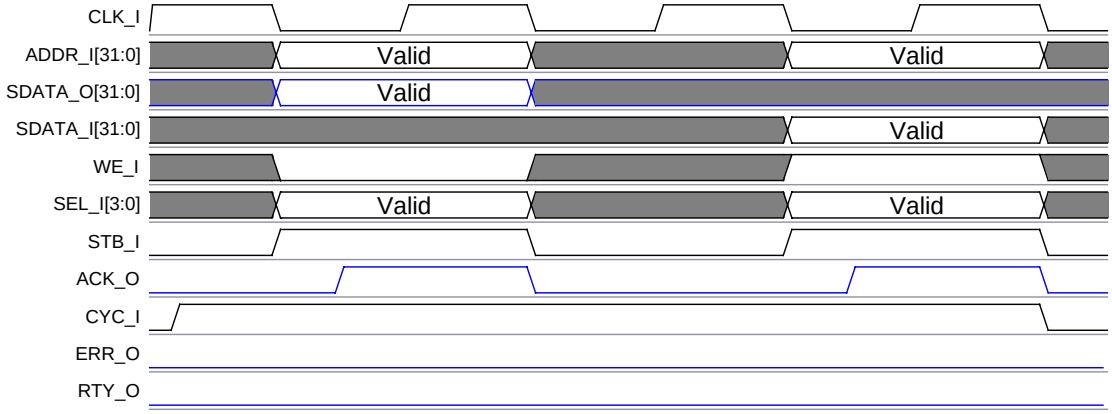


Figure 6-3: WISHBONE configuration RMW cycle

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

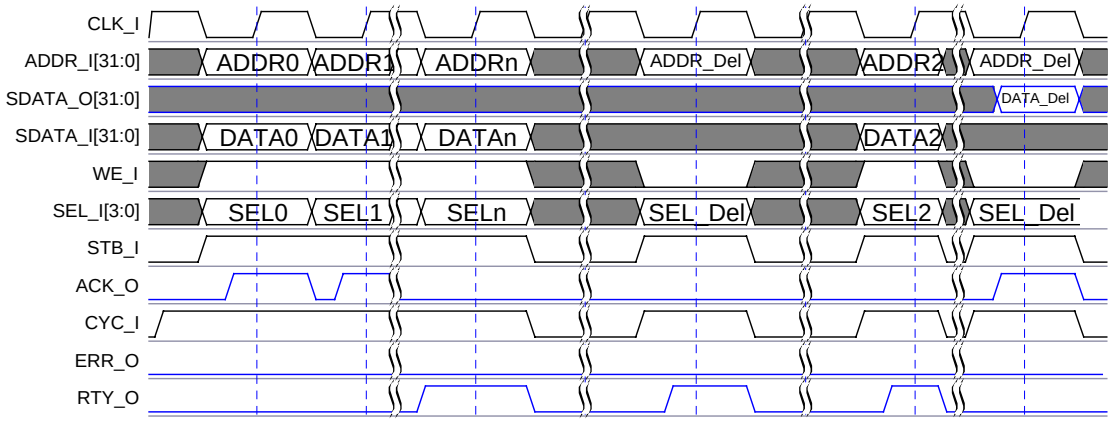


Figure 6-4: WISHBONE access to PCI address space

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 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.

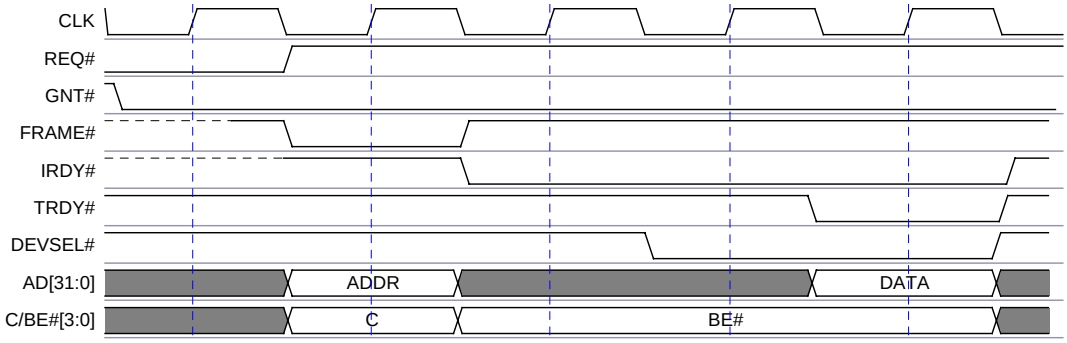


Figure 6-5: PCI Single Read cycle

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 claims the bus cycle by asserting FRAME# on the next rising edge of the clock. The 2nd clock cycle is also an address phase, so address and bus command information is provided on ADDR 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 access by asserting DEVSEL#. This clock cycle is used as Turnaround cycle (target starting to drive AD lines) inserted 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.

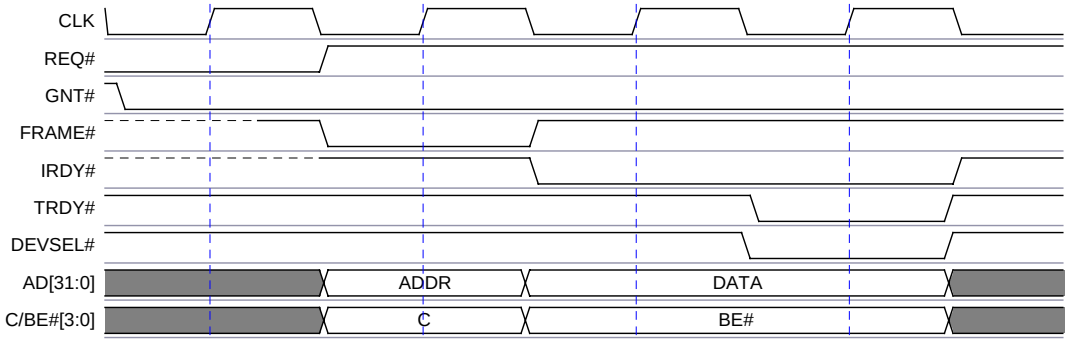


Figure 6-6: PCI Single Write

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 claims 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 ADDR 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.

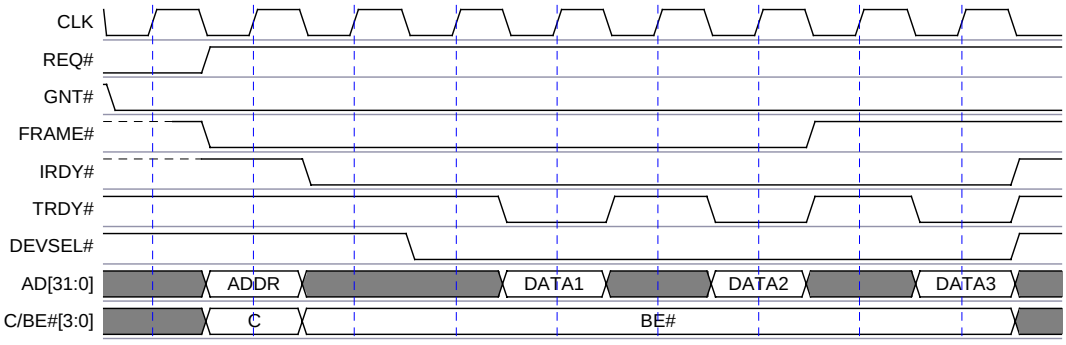


Figure 6-7: PCI Burst Read Cycle

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 Turnaround 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.

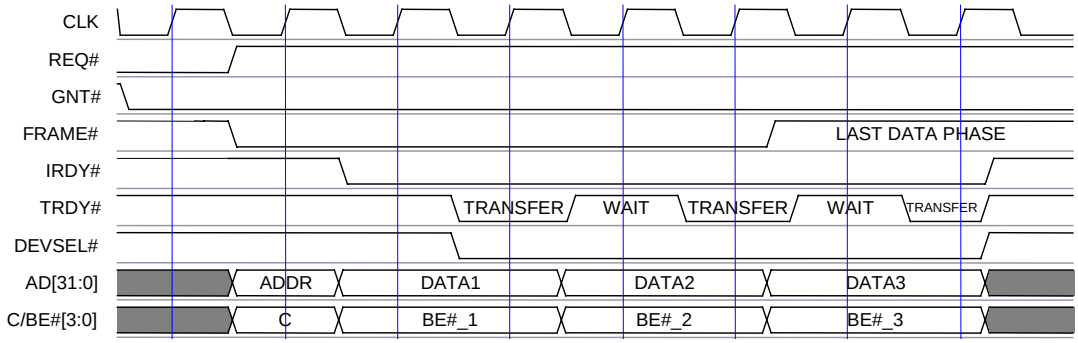


Figure 6-8: PCI Burst Write cycle

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 beat 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

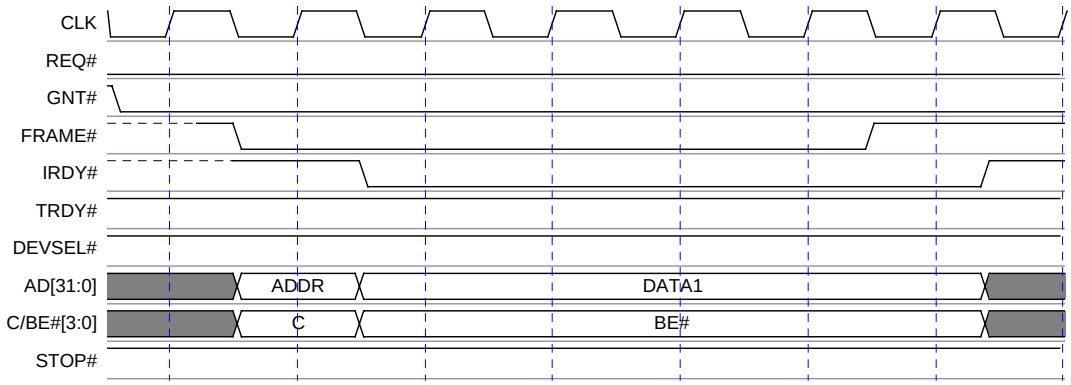


Figure 6-9: Master Abort termination

The PCI master module terminates the transaction with Master Abort, as shown in Figure 6-9. What happens? The master initiates a transaction 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 0) and the WISHBONE slave unit locks out all, but the configuration space accesses until the proper error status bit will be cleared. The current transaction is discarded (pulled out of WBW_FIFO) while any other Posted Write cycles 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.

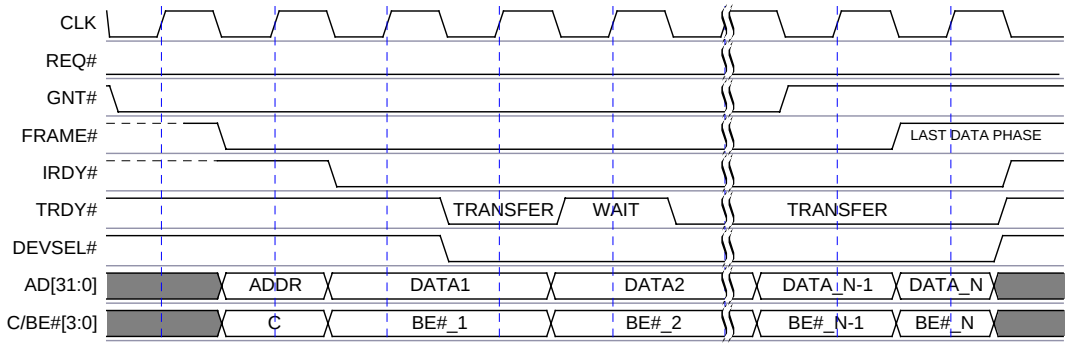


Figure 6-10: Timeout termination

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 mode, 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

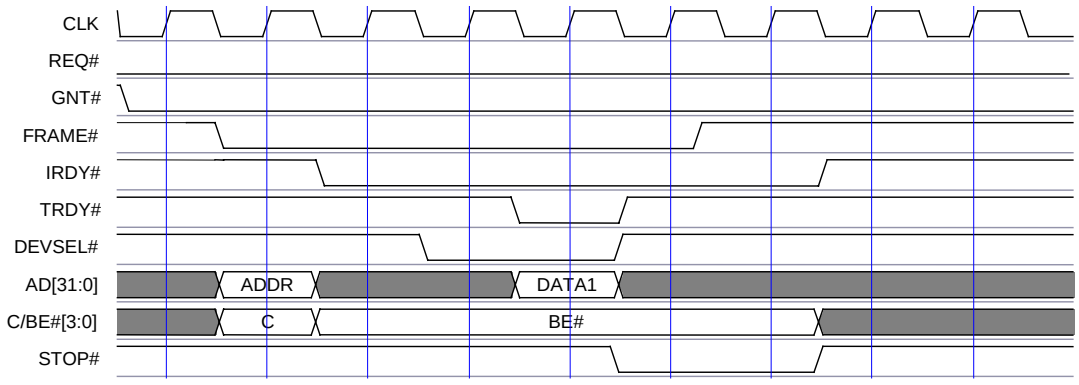


Figure 6-11: Target Abort

A target signals **Target Abort** to the master when it is and will be unable to complete the access initiated by the master. In this case, 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 and locks out any non-configuration space accesses until the corresponding error status bit is cleared.

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 cycle in order that the WISHBONE master will never perform a Read cycle to that address).

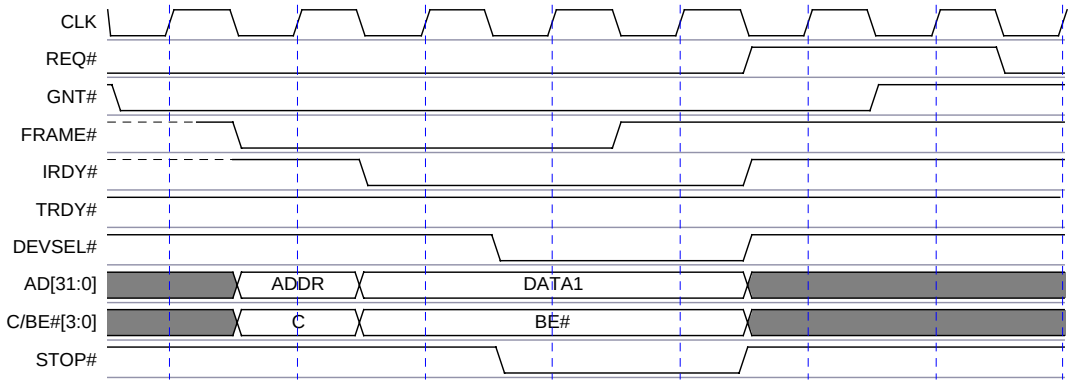


Figure 6-12: Target Retry

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.

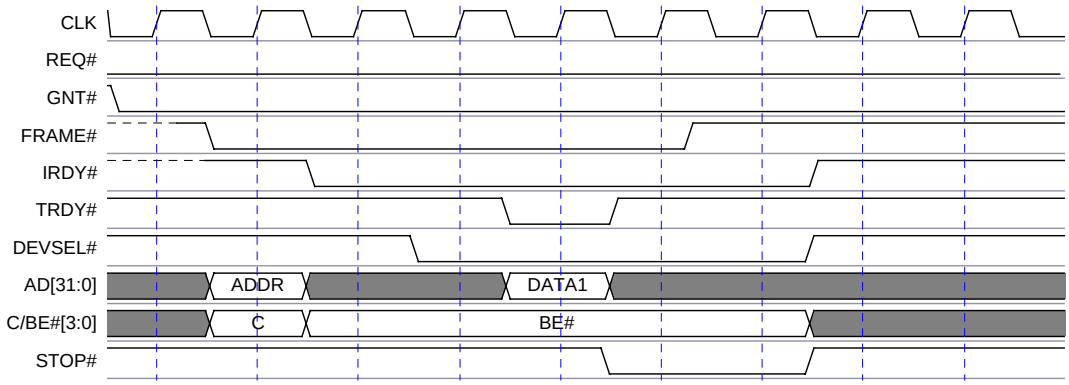


Figure 6-13: Target Disconnect without data

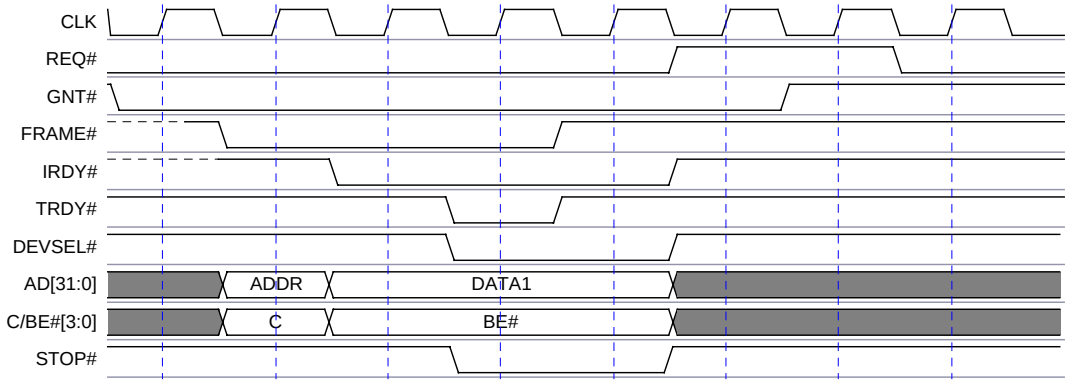


Figure 6-14: Target Disconnect with data

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.

6.1.2 PCI Configuration Accesses

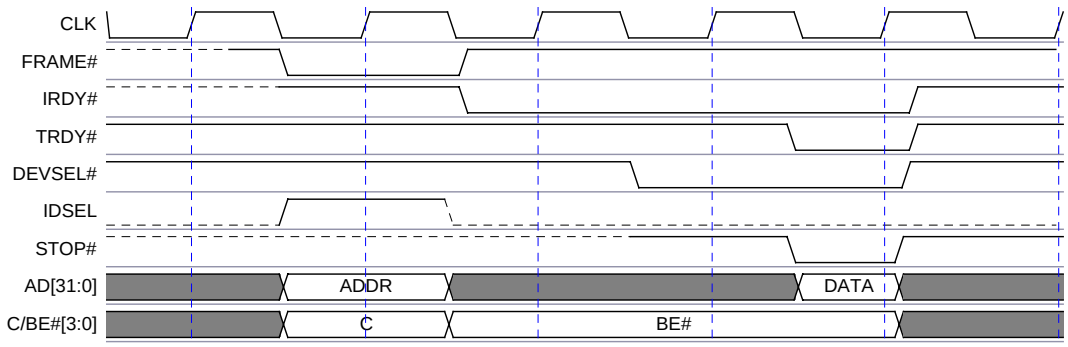


Figure 6-15: PCI Configuration Read cycle

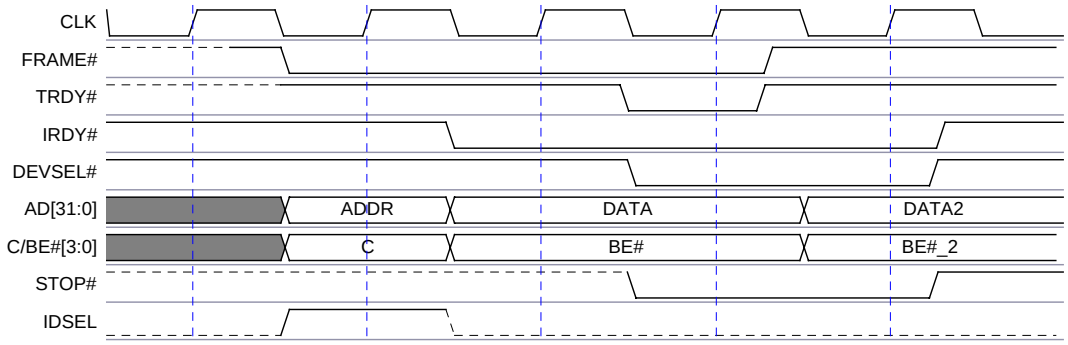


Figure 6-16: PCI Configuration Write cycle

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 command. The PCI target module accepts the Read command. 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.

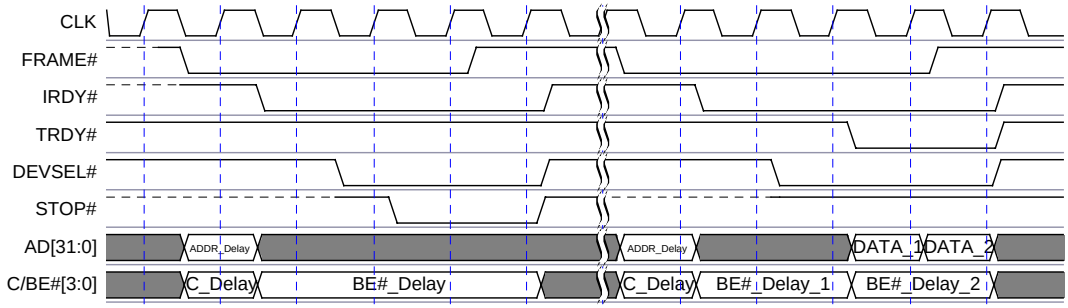


Figure 6-17: PCI Target Read cycle

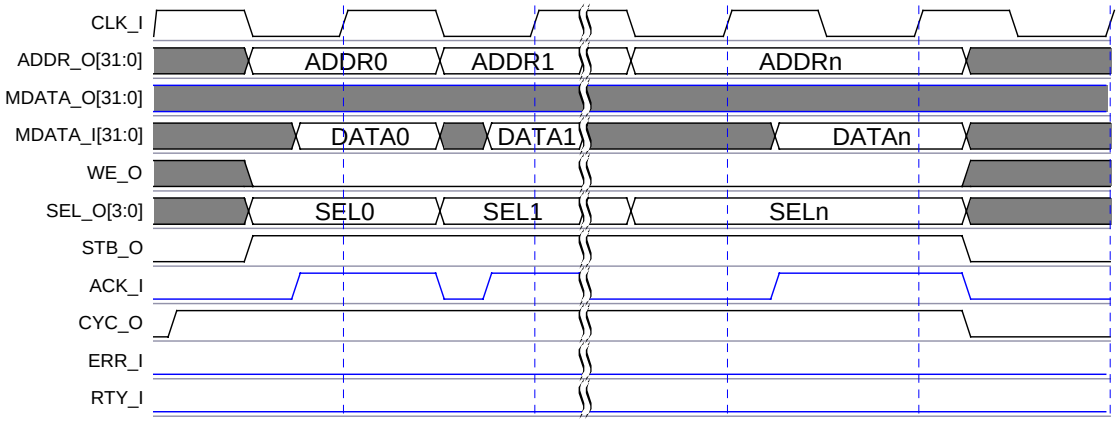


Figure 6-18: PCI to WISHBONE Read cycle

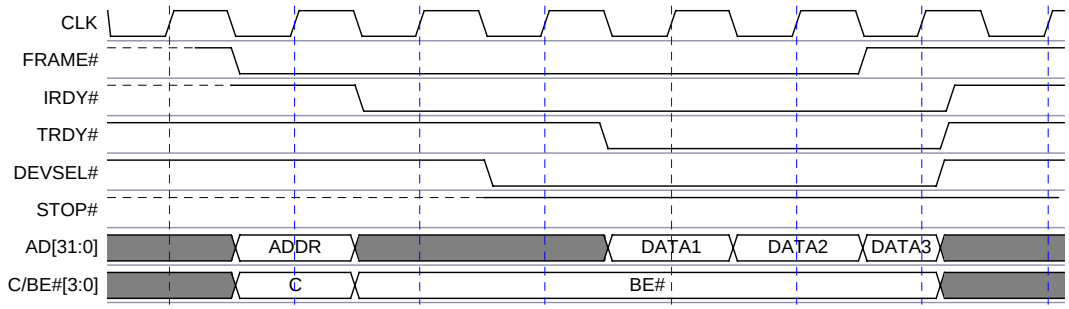


Figure 6-19: PCI Initiator to Target Burst Read cycle

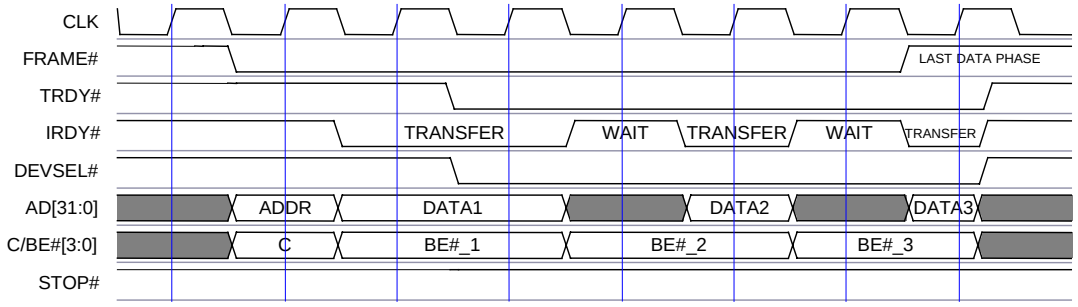


Figure 6-20: PCI Initiator to Target Burst Write cycle

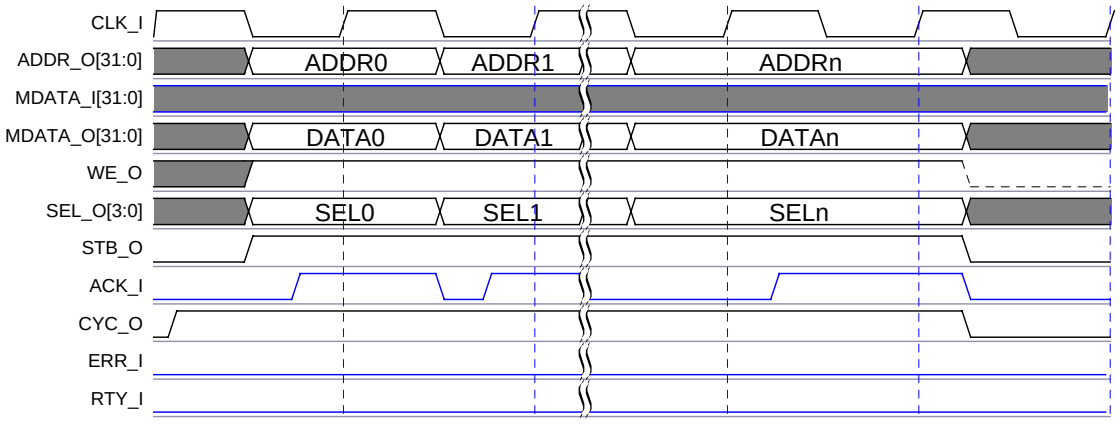


Figure 6-21: WISHBONE Write transfer caused by PCI to WISHBONE Write cycle

6.2.3 WISHBONE Terminations

Terminations on the WISHBONE bus are always performed by WISHBONE slaves. Chapters 3.3.3 PCI to WISHBONE Write Cycles and 3.3.4 PCI to WISHBONE Read Cycles describe the causes of **Retry** or **Error** on the WISHBONE bus.

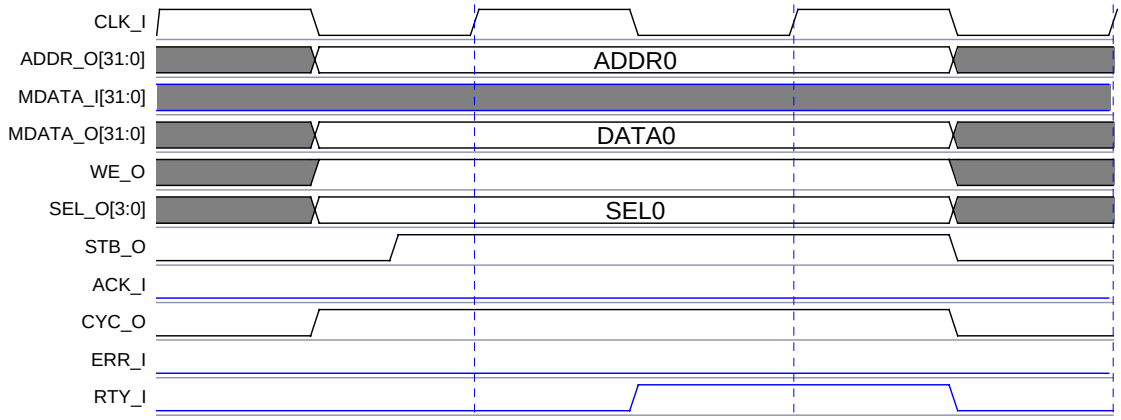


Figure 6-22: Retry on WISHBONE bus caused by PCI to WISHBONE transfer

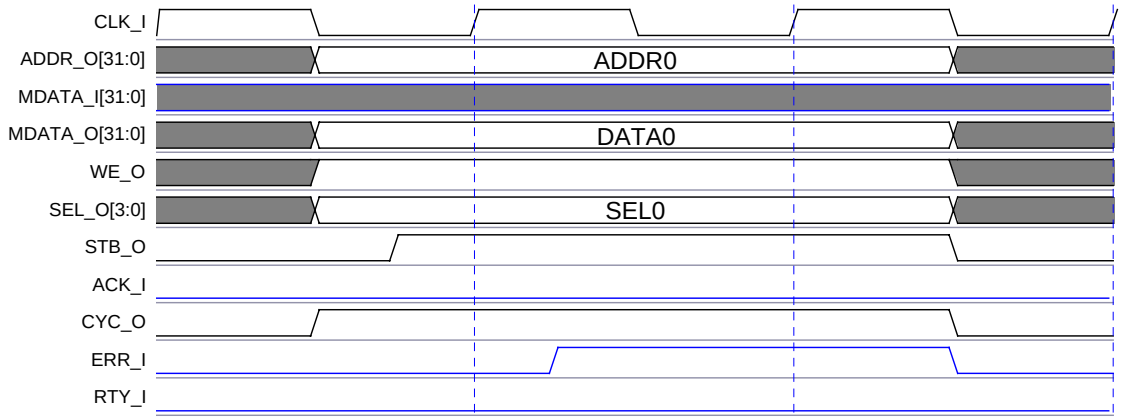


Figure 6-23: Error on WISHBONE bus caused by PCI to WISHBONE transfer

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.

A.1 HW Configuration Parameters

Configuration parameters are grouped into one 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 3.1).	
WBW_ADDR_LENGTH	
WBR_ADDR_LENGTH	
PCIW_ADDR_LENGTH	
PCIR_ADDR_LENGTH	
Numbers defined here define each one of four 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.	
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.	

Parameter	Defined value / Defined (yes, no)
Description	
PCI_NUM_OF_DEC_ADDR_LINES	
Number defined here is used for controlling implementation of PCI images' decoders. It defines how many MSB 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	
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 *	
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 needs to be defined, since it is always used for accessing Configuration space.	
PCI_IMAGE2 *	
PCI_IMAGE3 *	
PCI_IMAGE4 *	
PCI_IMAGE5 *	
If whichever defined, then that PCI Target image is implemented.	
PCI_AM0 ***	
PCI_AM1	
PCI_AM2 **	
PCI_AM3 **	
PCI_AM4 **	
PCI_AM5 **	
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 ***	
PCI_BA1_MEM_IO	
PCI_BA2_MEM_IO **	

Parameter	Defined value / Defined (yes, no)
Description	
PCI_BA3_MEM_IO **	
PCI_BA4_MEM_IO **	
PCI_BA5_MEM_IO **	
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.	
WB_NUM_OF_DEC_ADDR_LINES	
Number defined here is used for controlling implementation of WISHBONE images' decoders. It defines how many MSB 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.	
WB_CONFIGURATION_BASE	
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.	
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.	

Parameter	Defined value / Defined (yes, no)
Description	
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 image 1 is always implemented, without any exceptions

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

*** – This value is significant only if PCI image 0 is implemented to access WB bus

Table 6-1: User Useful HARDWARE Configuration Parameters

Index

address translation logic
  address mask register, setting rule 10
  address range 10
  architecture 11
  registers 10
architecture
  address translation logic 10–11
  clocks 8
  FIFO 8–9
  PCI bridge, general overview 3
  PCI target unit 6–8, 25
  WISHBONE slave unit 4–6, 19
clocks 8, 9
compliances
  PCI interface 3
  WISHBONE 3
configuration cycles 14, 15–18
  access to configuration space 15
  field values 16–17
  generating 16
  PCI, waveforms 74
  registers 55–57
  WISHBONE, waveforms 65–66
configuration parameters 78
configuration space 12–18
  access for guest bus bridges 14
  access for host bus bridges 13
  access to configuration cycles 15
  access, general 12
  definition 12
  header
    class code 44
    device ID 43
    header type 43
    registers 44–47
    revision ID 43
    vendor ID 43
  interrupt acknowledge cycles 18
configuration write cycles 66, 74
decoder 4
device identification ..... *See* configuration space header
encoding 21, 24, 28, 29, 31
expansion bus bridges ..... *See* guest bus bridges
features, PCI IP core 1–2
field values, configuration cycles 16–17
FIFO 8–9
  architechture 9
  architecture 8
  PCI read FIFO 7, 26
  PCI write FIFO 7, 26
  register lines 8
  WISHBONE read FIFO 5, 6, 19
  WISHBONE write FIFO 5, 6, 19
First in First out ..... 8–9. *See* also FIFO
identification ..... *See* configuration space header
interrupt acknowledge cycles
  generating 18
  register 57
interrupts, generating and reporting 33–34
IO ports
  PCI interface
    address and data pins 61
    arbitration pins 62
    error reporting pins 62
    interface control pins, optional 63
    interface control pins, required 62
    interrupt pins, optional 62
    system pins 62
  WISHBONE interface
    common control and system I/Os 64
    PCI target unit 63
    WISHBONE slave unit 64
operation
  configuration space 12–18
  interrupts 33–34
  parity 33
  transaction ordering 32–33
  WISHBONE slave unit 18–25
parity 33
PCI bridge, introduction

architecture....3
function....1
PCI target unit....3
WISHBONE slave unit....3
PCI target unit....3
address range, example....27
address space access
I/O mapped....28
memory mapped....29
address space, non-prefetchable....31
address translation, example....28
architecture....6–8, 25
basic functionality....25
configuration space header....43–47
encoding....28, 29, 31
error reporting mechanism....30
error reporting registers....53–55
function....6
images mapped to I/O space....31
images mapped to memory space....30
images, configurable....26
images, selecting....6
read FIFO....7, 26
target module....7, 26
termination signals....24, 32
waveforms....74–77
WISHBONE master module....8, 26
write cycles to WISHBONE....28–30
write FIFO....26
pins
optional....62–63
required....61–62
read cycles
block reads....23, 31
burst reads....23, 24, 31, 32, 69, 76
delayed reads....5, 22, 23, 24, 29, 30, 31, 33, 67
single reads....12, 14, 25, 31, 66, 68, 74
WISHBONE to PCI....21–23
registers
interrupt, control & status....60
list of....35–38
PCI target unit, configuration space header....43–47
PCI target unit, control & status....51
reporting....57
WISHBONE slave unit, control & status....42
termination cycles
PCI....70–73
WISHBONE....77
termination signals
disconnect....25, 29
disconnect with data....24, 31, 73
disconnect with/without data....29, 32
disconnect without data....24, 73
error....22, 23, 30, 31, 32, 52, 53, 54, 55, 70, 77
master abort....18, 22, 25, 45, 52, 70
retry....22, 24, 25, 29, 31, 32, 33, 52, 54, 67, 72, 75, 77
system error....45
target abort....22, 24, 25, 28, 29, 32, 45, 52, 71, 72
target disconnect....22, 24, 73
target disconnect with data....14, 15, 28, 73
target disconnect without data....73
target retry....72
timeout termination....71
transaction ordering....32–33
waveforms
PCI target unit
burst read cycle, initiator to target....76
burst write cycles, initiator to target....76
configuration read cycle....74
configuration write cycle....74
error on WISHBONE bus....77
read cycle to WISHBONE....75
retry on WISHBONE bus....77
target read cycle....74, 75
write transfer, WISHBONE....76
WISHBONE slave unit
access to PCI address space....67
burst read cycle, PCI....69
burst write cycle, PCI....69
configuration read cycle....65
configuration read modify write cycle....66
configuration write cycle....66
master abort termination, PCI....70
single read cycle, PCI....68
single writes, PCI....68
target abort, PCI....71
target disconnect with data, PCI....73
target disconnect without data, PCI....73
target retry, PCI....72
timeout termination....71
WISHBONE

bus agents 4
slave module
read FIFO 6
slave unit
address range, example 20
address space, non-prefetchable 23
address translation, example 21
architecture 4–6
decoder 4
encoding 21, 24
error reporting mechanism 22–23
error reporting registers 51–53
function 4
images mapped to I/O space 23
images mapped to memory space 23
images, configurable 4, 20
operation 18–25
PCI master module 6, 19
read cycles to PCI 23, 25
read FIFO 5, 19
slave module 5, 19
termination signals 32
waveforms 65–73
write cycles to PCI 21–23
write FIFO 5, 6, 19
write cycles
block writes 21, 22, 28, 30, 67
burst writes 29, 30, 69
PCI to WISHBONE 28–30
posted writes 5, 6, 21, 22, 28, 30, 33, 51, 70, 72
read modify writes (RMW) 14, 21, 66
single writes 12, 14, 21, 22, 28, 30, 68
WISHBONE to PCI 21–23