PRELIMINARY

XILINX®

LogiCore PCI Interface Protocol Checklist (PCI-SIG, Rev. 2.1)

July 7,1996 (Version 1.0)

Table of Contents

Introduction

This document describes the supported and unsupported features available in the Xilinx LC-PCIM (Target/Initiator) and LC-PCIS-C (Target-Only) LogiCore PCI Interfaces for the Xilinx XC4000E-2 FPGA device. The results of various protocol testing are reported using methods described by the PCI Compliance Checklist. The PCI Compliance Checklist is published by the PCI Special Interest Group (PCI-SIG) to help qualify a PCI product by creating a paper trail of testing for PCI compliance.

The results presented herein are based on revision 2.0b of the released PCI Compliance Checklist and the proposed 2.1 draft revision (pending final ratification).

Supported Features

Initiator Functions (beyond Target functionality)

- Initiate Memory Read, Memory Write, Memory Read Multiple, Memory Read Line commands
- Initiate I/O Read, I/O Write commands
- Initiate Configuration Read and Write commands
- Bus Parking
- 32-bit data transfers, burst transfers with linear address ordering

Target Functions

- Type 0 Configuration Space Header
- Up to 2 Base Address Registers (memory or I/O with adjustable block sizes from 16 bytes to 256 Mbytes)
- Parity Error Detection (PERR# and SERR#)
- Memory Read, Memory Write, Memory Read Multiple, Memory Real Line, Memory Write and Invalidate command support
- I/O Read, I/O Write command support
- 32-bit data transfers, burst transfers with linear address ordering
- Target Abort support
- Target Retry and Target Disconnect support
- Full Command/Status Register support

Features Implemented but Not Tested

- Interrupt support

Unsupported Features

- Type 1 Configuration Space Header
- Memory Write and Invalidate command
- Dual Address command
- Fast Back-to-Back command
- Target Lock support
- Cache line support
- Burst transfer to/from Target configuration space

Component Product Information

Date	1-JUL-96
Vendor Name	Xilinx, Inc.
Vendor Street Address	2100 Logic Dr.
Vendor City, State, Zip	San Jose, CA 95124 U.S.A.
Vendor Phone Number	1-408-559-7778
Vendor E-mail	pci@xilinx.com
Vendor Contact, Title	Per Holmberg LogiCore Marketing Manager
Product Name	LogiCore PCI Interface
Product Model Number	LC-DI-PCIM-C LC-DI-PCIS-C (Target Only) XC4000E-2 FPGA
Product Revision Level	Version 1.1

Component Configuration Checklist

Organization

CO1. Does each PCI resource have a configuration space based on the 256 byte template defined in section 6.1., with a pre-defined 64 byte header and a 192 byte device specific region? yes _√_ no___

CO2. Do all functions in the device support the Vendor ID, Device ID, Command, Status, Header Type and Class Code fields in the header? yes _√_ no___

CO3. Is the configuration space available for access at all times? yes _√_ no___
The configuration space is available for access at all times through the internal ADIO bus when other operations are not in progress. The contents of the Command/Status Register is available on the CSR[39:0] bus output from the macro. Supported status bits can be set at any time.

CO4. Are writes to reserved registers or read only bits completed normally and the data discarded? yes _√_ no___

CO5. Are reads to reserved or unimplemented registers, or bits, completed normally and a data value of 0 returned? yes _√_ no___

CO6. Is the vendor ID a number allocated by the PCI SIG? yes _√_ no___
The Xilinx vendor ID is the default value. However, the user should use his or her company's own vendor ID.

CO7. Does the Header Type field have a valid encoding? yes _√_ no___

CO8. Do multi-byte transactions access the appropriate registers and are the registers in "little endian" order? yes _√_ no___

CO9. Are all READ ONLY register values within legal ranges? For example, the Interrupt Pin register must only contain values 0-4. yes _√_ no___

CO10. Is the class code in compliance with the definition in Appendix D? yes _√_ no___

CO11. Is the pre-defined header portion of configuration space accessible as bytes, words, and double-words? yes _√_ no___

CO12. Is the device a multi-function device? yes ___ no_√_

CO13. If the device is multi-function, are configuration space accesses to unimplemented functions ignored? yes ___ no___ N/A_√_

Table 1: Implementation of Configuration Space Header

Location	Name	Required/Optional	N/A	Implemented
00h-01h	Vendor ID	Required		√
02h-03h	Device ID	Required		√
04h-05h	Command	Required		√
06h-07h	Status	Required		√
08h	Revision ID	Required		√
09h-0Bh	Class Code	Required		√
0Ch	Cache Line Size	Required by master devices/functions that can generate Memory Write and Invalidate	√	
0Dh	Latency Timer	Required by master devices/functions that can burst more than two data phases		√
0Eh	Header Type	If the device is multi-functional, then bit 7 must be set to a 1. The remaining bits are required to have a defined value.		√
0Fh	BIST	Optional	√	
10h-27h	Base Address Registers	1 or more required for any address allocation.		√
28h-2Bh	Cardbus CIS Pointer	Optional		√ returns 0
2Ch-2Dh	Subsystem Vendor ID	Optional		√ returns 0
2Eh-2Fh	Subsystem ID	Optional		√ returns 0
30h-33h	Expansion ROM Base Address	Required for devices/functions that have expansion ROM		√ returns 0
34h-3Bh	Reserved			√
3Ch	Interrupt Line	Required by devices/functions that use an interrupt pin		√
3Dh	Interrupt Pin	Required by devices/functions that use an interrupt pin		√
3Eh	Min_Gnt	Optional		√ returns 0
3Fh	Max_Lat	Optional		√ returns 0

Device Control

DC1. When the command register is loaded with a 0000h is the device/function logically disconnected from the PCI, with the exception of configuration accesses? (Devices in BOOT code path are exempt). yes _√_ no___

DC2. Is the device/function disabled after the assertion of PCI RST#? (Devices in BOOT code path are exempt). yes _√_ no___

In the following tables for Command and Status Registers, an "√" in the "Target" or "Master" columns, indicates that applying the bit is appropriate. "N/A" indicates that applying the bit is not applicable, but must return a 0 when read.

Table 2: Implementation of Command Register Bits

Bit	Name	Required/Optional	N/A	Target	Master
0	I/O Space	Required if device/function has registers mapped into I/O space		√	N/A
1	Memory Space	Required if device/function responds to memory space accesses		√	N/A
2	Bus Master	Required		N/A	√
3	Special Cycles	Required for devices/functions that can respond to Special Cycles	√		N/A
4	Memory Write and Invalidate Enable	Required for devices/functions that generate Memory Write and Invalidate cycles	√	N/A	
5	VGA Palette snoop	Required for VGA or graphical devices/functions that snoop VGA color palette	√		N/A
6	Parity Error Response	Required unless exempted per section 3.7.2		√	√
7	Wait cycle control, address stepping	Optional	√		
8	SERR# enable	Required if device/function has SERR# pin	√		
9	Fast Back-to-Back Enable	Required if Master device/function can support fast back-to-back cycles among different targets	√	N/A	
10-15	Reserved			√	√

Device Status

DS1. Do all implemented read/write bits in the Status reset to 0? yes _√_ no___

DS2. Are read/write bits set to a 1 exclusively by the device/function? yes _√_ no___

DS3. Are read/write bits reset to a 0 when PCIRST# is asserted? yes _√_ no___

DS4. Are read/write bits reset to a 0 by writing a 1 to the bit? yes _√_ no___

Table 3: Implementation Requirements for Status Register Bits

Bit	Name	Required/Optional	N/A	Target	Master
0-4	Reserved	Required		√	√
5	66 MHz Capable	Required for 66 MHz capable devices	√		
6	UDF Supported	Optional	√		
7	Fast Back-to-Back Capable	Optional. Target only.	√		N/A
8	Data Parity Detected	Required. Master only.		N/A	√
9-10	DEVSEL# Timing	Required. Target only.		√	N/A
11	Signaled Target Abort	Required for devices/functions that are capable of signaling target abort		√	N/A
12	Received Target Abort	Required. Master only.		N/A	√
13	Received Master Abort	Required. Master only.		N/A	√
14	Signaled System Error	Required for devices/functions that are capable of asserting SERR#		√	√
15	Detected Parity Error	Required unless exempted per section 3.7.2		√	√

Base Addresses

BA1. If the device/function uses expansion ROM, does it implement the Expansion ROM Base Address Register? ***The expansion ROM base address is not supported.*** yes ___ no_√_

BA2. Do all Base Address registers asking for I/O space request 256 bytes or less? yes _√_ no___

BA3. If the device/function has an Expansion ROM Base Address register, does the memory enable bit in the Command register have precedence over the enable bit in the Expansion ROM base Address register? ***The expansion ROM base address is not supported.*** yes ___ no_√_

BA4. Does the device/function use any address space (memory or I/O) other than that assigned using Base Address registers? (i.e.; Does the device/function hard-decode any addresses?) Note: If the answer is yes, you must list decoded addressses as explanations at the end of this section. yes ___ no_√_

BA5. Does the device/function decode all 32-bits of I/O space? ***The upper 24 bits of address are decoded by the base register. The lower 8 bits would be decoded by the user application.*** yes _√_ no___

BA6. If the device/function has an Expansion ROM Base Address register, is the size of the memory space requested 16MB or smaller?

General Component Protocol Checklist (Master)

The following checklist applies to all master operations.

MP1.	All Sustained Tri-State signals are driven high for one clock before being Tri-Stated. (2.1)	yes _√_ no___
MP2.	IUT always asserts all byte enables during each data phase of a Memory Write Invalidate cycle. (3.1.1). ***Memory Write and Invalidate command not supported.***	yes ___ no_√_
MP3.	IUT always uses Linear Burst Ordering for Memory Write Invalidate cycles. (3.1.1). ***Memory Write and Invalidate command not supported.***	yes ___ no_√_
MP4.	IUT always drives IRDY# when data is valid during a write transaction. (3.2.1)	yes _√_ no___
MP5.	IUT only transfers data when both IRDY# and TRDY# are asserted on the same rising clock edge. (3.2.1)	yes _√_ no___
MP6.	Once the IUT asserts IRDY# it never changes FRAME# until the current data phase completes. (3.2.1)	yes _√_ no___
MP7.	Once the IUT asserts IRDY# it never changes IRDY# until the current data phase completes. (3.2.1)	yes _√_ no___
MP8.	IUT never uses reserved burst ordering (AD[1::0] = “01”. (3.2.2). ***Value driven onto AD bus is controlled by the user application.***	yes ___ no_√_
MP9.	IUT never uses reserved burst ordering (AD[1::0] = “11”. (3.2.2). ***Value driven onto AD bus is controlled by the user application.***	yes ___ no_√_
MP10.	IUT always ignores configuration command unless IDSEL is asserted and AD[1::0] are “00”. (3.2.2)	yes _√_ no___
MP11.	The IUT’s AD lines are driven to stable values during every address and data phase. (3.2.4)	yes _√_ no___
MP12.	The IUT’s C/BE# output buffers remain enabled from the first clock of the data phase through the end of the transaction. (3.3.1). ***The values on the C/BE# pins are driven by the user application.***	yes _√_ no___
MP13.	The IUT’s C/BE# lines contain valid Byte Enable information during the entire data phase. (3.3.1). ***The values on the C/BE# pins are driven by the user application.***	yes _√_ no___
MP14.	IUT never de-asserts FRAME# unless IRDY# is asserted or will be asserted (3.3.3.1)	yes _√_ no___
MP15.	IUT never de-asserts IRDY# until at least one clock after FRAME# is de-asserted. (3.3.3.1)	yes _√_ no___
MP16.	Once the IUT de-asserts FRAME# it never reasserts FRAME# during the same transaction. (3.3.3.1)	yes _√_ no___
MP17.	IUT never terminates with master abort once target has asserted DEVSEL#. (3.3.3.1). True if no target has responded within 7 clocks. ***All targets should respond before this time.***	yes _√_ no___
MP18.	IUT never signals master abort earlier than 5 clocks after FRAME# was first sampled asserted. (3.3.3.1)	yes _√_ no___
MP19.	IUT always repeats an access exactly as the original when terminated by retry. (3.3.3.2.2). ***The retry process is controlled by logic in the user’s application.***	yes _√_ no___

MP20.	IUT never starts cycle unless GNT# is asserted. (3.4.1). ***The IUT never starts a transaction cycle unless the following conditions are true:*** • ***GNT- is asserted.*** • ***The bus is idle.*** • ***The Master Enable bit is set in the Command Register.*** ***The IUT can transition from Initiator IDLE state to DR_BUS when GNT# is asserted and there is not active REQUEST pending.***	yes _√_ no___
MP21.	IUT always Tri-States C/BE# and AD within one clock after GNT# negation when bus is idle and FRAME# is negated. (3.4.3)	yes _√_ no___
MP22.	IUT always drives C/BE# and AD within eight clocks of GNT# assertion when bus is idle. (3.4.3)	yes _√_ no___
MP23.	IUT always asserts IRDY# within eight clocks on all data phases. (3.5.2). ***The IRDY# signal is controlled by the READY input from the user application.***	yes _√_ no___
MP24.	IUT always begins lock operation with a read transaction. (3.6). ***LOCK# function not supported.***	yes ___ no_√_
MP25.	IUT always releases LOCK# when access is terminated by target-abort or master-abort. (3.6). **LOCK# function not supported.**	yes ___ no_√_
MP26.	IUT always de-asserts LOCK# for minimum of one idle cycle between consecutive lock operations. (3.6). ***LOCK# function not supported.***	yes ___ no_√_
MP27.	IUT always uses Linear Burst Ordering for configuration cycles. (3.7.4).	yes _√_ no___
MP28.	IUT always drives PAR within one clock of C/BE# and AD being driven. (3.8.1)	yes _√_ no___
MP29.	IUT always drives PAR such that the number of “1”s on AD[31::0],C/BE[3:0], and PAR equals an even number. (3.8.1)	yes _√_ no___
MP30.	IUT always drives PERR# (when enabled) active two clocks after data when data parity error is detected and . (3.8.2.1)	yes _√_ no___
MP31.	IUT always drives PERR (when enabled) for a minimum of 1 clock for each data phase that a parity error is detected. (3.8.2.1)	yes _√_ no___
MP32.	IUT always holds FRAME# asserted for cycle following DUAL command. (3.10.1). ***Dual Address command not supported.***	yes ___ no_√_
MP33.	IUT never generates DUAL cycle when upper 32-bits of address are zero. (3.10.1). ***Dual Address command not supported.***	yes ___ no_√_

General Component Protocol Checklist (Target)

The following checklist applies to all target operations.

TP1.	All Sustained Tri-State signals are driven high for one clock before being Tri-Stated. (2.1)	yes _√_ no___
TP2.	IUT never reports PERR# until it has claimed the cycle and completed a data phase. (2.2.5)	yes _√_ no___
TP3.	IUT never aliases reserved commands with other commands. (3.1.1)	yes _√_ no___
TP4.	32-bit addressable IUT treats DUAL command as reserved. (3.1.1)	yes _√_ no___
TP5.	Once IUT has asserted TRDY# it never changes TRDY# until the data phase completes. (3.2.1)	yes _√_ no___
TP6.	Once IUT has asserted TRDY# it never changes DEVSEL# until the data phase completes. (3.2.1)	yes _√_ no___
TP7.	Once IUT has asserted TRDY# it never changes STOP# until the data phase completes. (3.2.1)	yes _√_ no___
TP8.	Once IUT has asserted STOP# it never changes STOP# until the data phase completes. (3.2.1)	yes _√_ no___
TP9.	Once IUT has asserted STOP# it never changes TRDY# until the data phase completes. (3.2.1)	yes _√_ no___
TP10.	Once IUT has asserted STOP# it never changes DEVSEL# until the data phase completes. (3.2.1)	yes _√_ no___
TP11.	IUT only transfers data when both IRDY# and TRDY# are asserted on the same rising clock edge. (3.2.1)	yes _√_ no___
TP12.	IUT always asserts TRDY# when data is valid on a read cycle. (3.2.1)	yes _√_ no___
TP13.	IUT always signals target-abort when unable to complete the entire I/O access as defined by the byte enables. (3.2.2). ***This function is implemented in the user application. Only the user application could determine if the byte enables were valid for the selected I/O device.***	yes ___ no_√_
TP14.	IUT never responds to reserved encodings. (3.2.2)	yes _√_ no___
TP15.	IUT always ignores configuration command unless IDSEL is asserted and AD[1::0] are “00”. (3.2.2)	yes _√_ no___
TP16.	IUT always disconnects after the first data phase when reserved burst mode is detected. (3.2.2). ***This function would be implemented in the user application.***	yes ___ no_√_
TP17.	The IUT’s AD lines are driven to stable values during every address and data phase. (3.2.4)	yes _√_ no___
TP18.	The IUT’s C/BE# output buffers remain enabled from the first clock of the data phase through the end of the transaction. (3.3.1)	yes _√_ no___
TP19.	IUT never asserts TRDY# during turnaround cycle on a read. (3.3.1)	yes _√_ no___
TP20.	IUT always de-asserts TRDY#, STOP#, and DEVSEL# the clock following the completion of the last data phase. (3.3.3.2)	yes _√_ no___

TP21.	IUT always signals disconnect when burst crosses resource boundary. (3.3.3.2). ***This function would be implemented in the user application.***	yes ___ no_√_
TP22.	IUT always de-asserts STOP# the cycle immediately following FRAME# being de-asserted. (3.3.3.2.1)	yes _√_ no___
TP23.	Once the IUT has asserted STOP# it never de-asserts STOP# until FRAME# is negated. (3.3.3.2.1)	yes _√_ no___
TP24.	IUT always de-asserts TRDY# before signaling target-abort. (3.3.3.2.1)	yes _√_ no___
TP25.	IUT never de-asserts STOP# and continues the transaction. (3.3.3.2.1)	yes _√_ no___
TP26.	IUT always completes initial data phase within 16 clocks. (3.5.1.1)	yes _√_ no___
TP27.	IUT always locks minimum of 16 bytes (3.6). ***LOCK# function not supported.***	yes ___ no_√_
TP28.	IUT always issues DEVSEL# before any other response. (3.7.1)	yes _√_ no___
TP29.	Once IUT has asserted DEVSEL# it never de-asserts DEVSEL# until the last data phase has competed except to signal target-abort. (3.7.1)	yes _√_ no___
TP30.	IUT never responds to special cycles (3.7.2)	yes _√_ no___
TP31.	IUT always drives PAR within one clock of C/BE# and AD being driven. (3.8.1)	yes _√_ no___
TP32.	IUT always drives PAR such that the number of “1”s on AD[31::0],C/BE[3:0], and PAR equals an even number. (3.8.1)	yes _√_ no___

Component Protocol Checklist for a Master Device

Definition: IUT is an acronym for "Implementation Under Test".

Test Scenario: 1.1 PCI Device Speed (as indicated by DEVSEL#) Tests

Note: The Initiator detects and reports a Master Abort condition. However, the LogiCore Initiator de-asserts FRAME- and IRDY- one cycle later than the diagram shown as Figure 3-4 on page 40 in the PCI Local Bus Specification, revision 2.1. Otherwise, the LogiCore Initiator treats Master Abort normally.

Memory Transactions

Test	Description	N/A	Pass
1	Data transfer after write to fast memory slave.		√
2	Data transfer after read from fast memory slave.		√
3	Data transfer after write to medium memory slave.		√
4	Data transfer after read from medium memory slave.		√
5	Data transfer after write to slow memory slave.		√
6	Data transfer after read from slow memory slave.		√
7	Data transfer after write to subtractive memory slave.		√
8	Data transfer after read from subtractive memory slave.		√
9	Master abort bit set after write to slower than subtractive memory slave.		√
10	Master abort bit set after read from slower than subtractive memory slave.		√

I/O Transactions

Test	Description	N/A	Pass
11	Data transfer after write to fast I/O slave.		√
12	Data transfer after read from fast I/O slave.		√
13	Data transfer after write to medium I/O slave.		√
14	Data transfer after read from medium I/O slave.		√
15	Data transfer after write to slow I/O slave.		√
16	Data transfer after read from slow I/O slave.		√
17	Data transfer after write to subtractive I/O slave.		√
18	Data transfer after read from subtractive I/O slave.		√
19	Master abort bit set after write to slower than subtractive I/O slave.		√
20	Master abort bit set after read from slower than subtractive I/O slave.		√

Configuration Transactions

Note:Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
21	Data transfer after write to fast configuration slave.		√
22	Data transfer after read from fast configuration slave.		√
23	Data transfer after write to medium configuration slave.		√
24	Data transfer after read from medium configuration slave.		√
25	Data transfer after write to slow configuration slave.		√
26	Data transfer after read from slow configuration slave.		√
27	Data transfer after write to subtractive configuration slave.		√
28	Data transfer after read from subtractive configuration slave.		√
29	Master abort bit set after write to slower than subtractive configuration slave.		√
30	Master abort bit set after read from slower than subtractive configuration slave.		√

Interrupt Transactions

Test	Description	N/A	Pass
31	Data transfer after interrupt from fast memory slave.	√	
32	Data transfer after interrupt from medium memory slave.	√	
33	Data transfer after interrupt from slow memory slave.	√	
34	Data transfer after interrupt from subtractive memory slave.	√	
35	Master abort bit set for interrupt from slower than subtractive memory slave.	√	

Special Transactions

Test	Description	N/A	Pass
36	Data transfer after Special transaction to slave.	√	
37	Master abort bit is not set after Special transaction.	√	

Test Scenario: 1.2 PCI Bus Target Abort Cycles

Definition: IUT is an acronym for "Implementation Under Test".

Note: The Initiator does not repeat a transaction because it is controlled by the user application. The user application must monitor the Recevied Target Abort bit (CSR28) in the Command Register to prevent a retry after a Target Abort.

Memory Transactions

Test	Description	N/A	Pass
1	Target Abort bit set after write to fast memory slave.		√
2	IUT does not repeat the write transaction.		√
3	IUT’s Target Abort bit set after read from fast memory slave.		√
4	IUT does not repeat the read transaction.		√
5	Target Abort bit set after write to medium memory slave.		√
6	IUT does not repeat the write transaction.		√
7	IUT’s Target Abort bit set after read from medium memory slave.		√
8	IUT does not repeat the read transaction.		√
9	Target Abort bit set after write to slow memory slave.		√
10	IUT does not repeat the write transaction.		√
11	IUT’s Target Abort bit set after read from slow memory slave.		√
12	IUT does not repeat the read transaction.		√
13	Target Abort bit set after write to subtractive memory slave.		√
14	IUT does not repeat the write transaction.		√
15	IUT’s Target Abort bit set after read from subtractive memory slave.		√
16	IUT does not repeat the read transaction.		√

I/O Transactions

Test	Description	N/A	Pass
17	Target Abort bit set after write to fast I/O slave.		√
18	IUT does not repeat the write transaction.		√
19	IUT's Target Abort bit set after read from fast I/O slave.		√
20	IUT does not repeat the read transaction.		√
21	Target Abort bit set after write to medium I/O slave.		√
22	IUT does not repeat the write transaction.		√
23	IUT's Target Abort bit set after read from medium I/O slave.		√
24	IUT does not repeat the read transaction.		√
25	Target Abort bit set after write to slow I/O slave.		√
26	IUT does not repeat the write transaction.		√
27	IUT's Target Abort bit set after read from slow I/O slave.		√
28	IUT does not repeat the read transaction.		√
29	Target Abort bit set after write to subtractive I/O slave.		√
30	IUT does not repeat the write transaction.		√
31	IUT's Target Abort bit set after read from subtractive I/O slave.		√
32	IUT does not repeat the read transaction.		√

Configuration Transactions

Note:Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
33	Target Abort bit set after write to fast configuration slave.		√
34	IUT does not repeat the write transaction.		√
35	IUT's Target Abort bit set after read from fast configuration slave.		√
36	IUT does not repeat the read transaction.		√
37	Target Abort bit set after write to medium configuration slave.		√
38	IUT does not repeat the write transaction.		√
39	IUT's Target Abort bit set after read from medium configuration slave.		√
40	IUT does not repeat the read transaction.		√
41	Target Abort bit set after write to slow configuration slave.		√
42	IUT does not repeat the write transaction.		√
43	IUT's Target Abort bit set after read from slow configuration slave.		√
44	IUT does not repeat the read transaction.		√
45	Target Abort bit set after write to subtractive configuration slave.		√
46	IUT does not repeat the write transaction.		√
47	IUT's Target Abort bit set after read from subtractive configuration slave.		√
48	IUT does not repeat the read transaction.		√

Interrupt Acknowledge Transactions

Test	Description	N/A	Pass
49	IUT’s Target Abort bit set after interrupt acknowledge from fast slave.	√	
50	IUT does not repeat the interrupt acknowledge transaction.	√	
51	IUT’s Target Abort bit set after interrupt acknowledge from medium slave.	√	
52	IUT does not repeat the interrupt acknowledge transaction.	√	
53	IUT’s Target Abort bit set after interrupt acknowledge from slow slave.	√	
54	IUT does not repeat the interrupt acknowledge transaction.	√	
55	IUT’s Target Abort bit set after interrupt acknowledge from subtractive slave.	√	
56	IUT does not repeat the interrupt acknowledge transaction.	√	

Test Scenario: 1.3 PCI Bus Target Retry Cycles

Memory Transactions

Test	Description	N/A	Pass
1	Data transfer after write to fast memory slave.		√
2	Data transfer after read from fast memory slave.		√
3	Data transfer after write to medium memory slave.		√
4	Data transfer after read from medium memory slave.		√
5	Data transfer after write to slow memory slave.		√
6	Data transfer after read from slow memory slave.		√
7	Data transfer after write to subtractive memory slave.		√
8	Data transfer after read from subtractive memory slave.		√

I/O Transactions

Test	Description	N/A	Pass
9	Data transfer after write to fast I/O slave.		√
10	Data transfer after read from fast I/O slave.		√
11	Data transfer after write to medium I/O slave.		√
12	Data transfer after read from medium I/O slave.		√
13	Data transfer after write to slow I/O slave.		√
14	Data transfer after read from slow I/O slave.		√
15	Data transfer after write to subtractive I/O slave.		√
16	Data transfer after read from subtractive I/O slave.		√

Configuration Transactions

Note: Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
17	Data transfer after write to fast configuration slave.		√
18	Data transfer after read from fast configuration slave.		√
19	Data transfer after write to medium configuration slave.		√
20	Data transfer after read from medium configuration slave.		√
21	Data transfer after write to slow configuration slave.		√
22	Data transfer after read from slow configuration slave.		√
23	Data transfer after write to subtractive configuration slave.		√
24	Data transfer after read from subtractive configuration slave.		√

Interrupt Acknowledge Transactions

Test	Description	N/A	Pass
25	Data transfer after interrupt acknowledge from fast slave.	√	
26	Data transfer after interrupt acknowledge from medium slave.	√	
27	Data transfer after interrupt acknowledge from slow slave.	√	
28	Data transfer after interrupt acknowledge from subtractive slave.	√	

Test Scenario: 1.4 PCI Bus Single Data Phase Disconnect Cycles

Memory Transactions

Test	Description	N/A	Pass
1	Data transfer after write to fast memory slave.		√
2	Data transfer after read from fast memory slave.		√
3	Data transfer after write to medium memory slave.		√
4	Data transfer after read from medium memory slave.		√
5	Data transfer after write to slow memory slave.		√
6	Data transfer after read from slow memory slave.		√
7	Data transfer after write to subtractive memory slave.		√
8	Data transfer after read from subtractive memory slave.		√

I/O Transactions

Test	Description	N/A	Pass
9	Data transfer after write to fast I/O slave.		√
10	Data transfer after read from fast I/O slave.		√
11	Data transfer after write to medium I/O slave.		√
12	Data transfer after read from medium I/O slave.		√
13	Data transfer after write to slow I/O slave.		√
14	Data transfer after read from slow I/O slave.		√
15	Data transfer after write to subtractive I/O slave.		√
16	Data transfer after read from subtractive I/O slave.		√

Configuration Transactions

Note: Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
17	Data transfer after write to fast configuration slave.		√
18	Data transfer after read from fast configuration slave.		√
19	Data transfer after write to medium configuration slave.		√
20	Data transfer after read from medium configuration slave.		√
21	Data transfer after write to slow configuration slave.		√
22	Data transfer after read from slow configuration slave.		√
23	Data transfer after write to subtractive configuration slave.		√
24	Data transfer after read from subtractive configuration slave.		√

Interrupt Acknowledge Transactions

Test	Description	N/A	Pass
25	Data transfer after interrupt acknowledge from fast slave.	√	
26	Data transfer after interrupt acknowledge from medium slave.	√	
27	Data transfer after interrupt acknowledge from slow slave.	√	
28	Data transfer after interrupt acknowledge from subtractive slave.	√	

Test Scenario: 1.5. PCI Bus Multi-Data Phase Target Abort Cycles

Note: The Initiator does not repeat a transaction because it is controlled by the user application. The user application must monitor the Recevied Target Abort bit (CSR28) in the Command Register to prevent a retry after a Target Abort.

Memory Transactions

Test	Description	N/A	Pass
1	Target Abort bit set after write to fast memory slave.		√
2	IUT does not repeat the write transaction.		√
3	IUT's Target Abort bit set after read from fast memory slave.		√
4	IUT does not repeat the read transaction.		√
5	Target Abort bit set after write to medium memory slave.		√
6	IUT does not repeat the write transaction.		√
7	IUT's Target Abort bit set after read from medium memory slave.		√
8	IUT does not repeat the read transaction.		√
9	Target Abort bit set after write to slow memory slave.		√
10	IUT does not repeat the write transaction.		√
11	IUT's Target Abort bit set after read from slow memory slave.		√
12	IUT does not repeat the read transaction.		√
13	Target Abort bit set after write to subtractive memory slave.		√
14	IUT does not repeat the write transaction.		√
15	IUT's Target Abort bit set after read from subtractive memory slave.		√
16	IUT does not repeat the read transaction.		√

I/O Transactions

Test	Description	N/A	Pass
17	Target Abort bit set after write to fast I/O slave.		√
18	IUT does not repeat the write transaction.		√
19	IUT's Target Abort bit set after read from fast I/O slave.		√
20	IUT does not repeat the read transaction.		√
21	Target Abort bit set after write to medium I/O slave.		√
22	IUT does not repeat the write transaction.		√
23	IUT's Target Abort bit set after read from medium I/O slave.		√
24	IUT does not repeat the read transaction.		√
25	Target Abort bit set after write to slow I/O slave.		√
26	IUT does not repeat the write transaction.		√
27	IUT's Target Abort bit set after read from slow I/O slave.		√
28	IUT does not repeat the read transaction.		√
29	Target Abort bit set after write to subtractive I/O slave.		√
30	IUT does not repeat the write transaction.		√
31	IUT's Target Abort bit set after read from subtractive I/O slave.		√
32	IUT does not repeat the read transaction.		√

Configuration Transactions

Note: Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
33	Target Abort bit set after write to fast configuration slave.		√
34	IUT does not repeat the write transaction.		√
35	IUT’s Target Abort bit set after read from fast configuration slave.		√
36	IUT does not repeat the read transaction.		√
37	Target Abort bit set after write to medium configuration slave.		√
38	IUT does not repeat the write transaction.		√
39	IUT’s Target Abort bit set after read from medium configuration slave.		√
40	IUT does not repeat the read transaction.		√
41	Target Abort bit set after write to slow configuration slave.		√
42	IUT does not repeat the write transaction.		√
43	IUT’s Target Abort bit set after read from slow configuration slave.		√
44	IUT does not repeat the read transaction.		√
45	Target Abort bit set after write to subtractive configuration slave.		√
46	IUT does not repeat the write transaction.		√
47	IUT’s Target Abort bit set after read from subtractive configuration slave.		√
48	IUT does not repeat the read transaction.		√

Memory Read Multiple Transactions

Test	Description	N/A	Pass
49	IUT’s Target Abort bit set after read from fast memory slave.		√
50	IUT does not repeat the read transaction.		√
51	IUT’s Target Abort bit set after read from medium memory slave.		√
52	IUT does not repeat the read transaction.		√
53	IUT’s Target Abort bit set after read from slow memory slave.		√
54	IUT does not repeat the read transaction.		√
55	IUT’s Target Abort bit set after read from subtractive memory slave.		√
56	IUT does not repeat the read transaction.		√

Memory Read Line Transactions

Test	Description	N/A	Pass
57	IUT’s Target Abort bit set after read from fast memory slave.		√
58	IUT does not repeat the read transaction.		√
59	IUT’s Target Abort bit set after read from medium memory slave.		√
60	IUT does not repeat the read transaction.		√
61	IUT’s Target Abort bit set after read from slow memory slave.		√
62	IUT does not repeat the read transaction.		√
63	IUT’s Target Abort bit set after read from subtractive memory slave.		√
64	IUT does not repeat the read transaction.		√

Memory Write and Invalidate Transactions

Test	Description	N/A	Pass
65	Target Abort bit set after write to fast memory slave.	√	
66	IUT does not repeat the write transaction.	√	
67	Target Abort bit set after write to medium memory slave.	√	
68	IUT does not repeat the write transaction.	√	
69	Target Abort bit set after write to slow memory slave.	√	
70	IUT does not repeat the write transaction.	√	
71	IUT’s Target Abort bit set after read from slow memory slave.	√	
72	IUT does not repeat the write transaction.	√	

Test Scenario: 1.6. PCI Bus Multi-Data Phase Retry Cycles

Memory Transactions

Test	Description	N/A	Pass
1	Data transfer after write to fast memory slave.		√
2	Data transfer after read from fast memory slave.		√
3	Data transfer after write to medium memory slave.		√
4	Data transfer after read from medium memory slave.		√
5	Data transfer after write to slow memory slave.		√
6	Data transfer after read from slow memory slave.		√
7	Data transfer after write to subtractive memory slave.		√
8	Data transfer after read from subtractive memory slave.		√

I/O Transactions

Test	Description	N/A	Pass
9	Data transfer after write to fast I/O slave.		√
10	Data transfer after read from fast I/O slave.		√
11	Data transfer after write to medium I/O slave.		√
12	Data transfer after read from medium I/O slave.		√
13	Data transfer after write to slow I/O slave.		√
14	Data transfer after read from slow I/O slave.		√
15	Data transfer after write to subtractive I/O slave.		√
16	Data transfer after read from subtractive I/O slave.		√

Configuration Transactions

Note: Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
17	Data transfer after write to fast configuration slave.		√
18	Data transfer after read from fast configuration slave.		√
19	Data transfer after write to medium configuration slave.		√
20	Data transfer after read from medium configuration slave.		√
21	Data transfer after write to slow configuration slave.		√
22	Data transfer after read from slow configuration slave.		√
23	Data transfer after write to subtractive configuration slave.		√
24	Data transfer after read from subtractive configuration slave.		√

Memory Read Multiple Transactions

Test	Description	N/A	Pass
25	Data transfer after memory read multiple from fast slave.		√
26	Data transfer after memory read multiple from medium slave.		√
27	Data transfer after memory read multiple from slow slave.		√
28	Data transfer after memory read multiple from subtractive slave.		√

Memory Read Line Transactions

Test	Description	N/A	Pass
29	Data transfer after memory read line from fast slave.		√
30	Data transfer after memory read line from medium slave.		√
31	Data transfer after memory read line from slow slave.		√
32	Data transfer after memory read line from subtractive slave.		√

Memory Write and Invalidate Transactions

Test	Description	N/A	Pass
33	Data transfer after memory write and invalidate to fast slave.	√	
34	Data transfer after memory write and invalidate to medium slave.	√	
35	Data transfer after memory write and invalidate to slow slave.	√	
36	Data transfer after memory write and invalidate to subtractive slave.	√	

Test Scenario: 1.7. PCI Bus Multi-Data Phase Disconnect Cycles

Memory Transactions

Test	Description	N/A	Pass
1	Data transfer after write to fast memory slave.		√
2	Data transfer after read from fast memory slave.		√
3	Data transfer after write to medium memory slave.		√
4	Data transfer after read from medium memory slave.		√
5	Data transfer after write to slow memory slave.		√
6	Data transfer after read from slow memory slave.		√
7	Data transfer after write to subtractive memory slave.		√
8	Data transfer after read from subtractive memory slave.		√

I/O Transactions

Test	Description	N/A	Pass
9	Data transfer after write to fast I/O slave.		√
10	Data transfer after read from fast I/O slave.		√
11	Data transfer after write to medium I/O slave.		√
12	Data transfer after read from medium I/O slave.		√
13	Data transfer after write to slow I/O slave.		√
14	Data transfer after read from slow I/O slave.		√
15	Data transfer after write to subtractive I/O slave.		√
16	Data transfer after read from subtractive I/O slave.		√

Configuration Transactions

Note: Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
17	Data transfer after write to fast configuration slave.		√
18	Data transfer after read from fast configuration slave.		√
19	Data transfer after write to medium configuration slave.		√
20	Data transfer after read from medium configuration slave.		√
21	Data transfer after write to slow configuration slave.		√
22	Data transfer after read from slow configuration slave.		√
23	Data transfer after write to subtractive configuration slave.		√
24	Data transfer after read from subtractive configuration slave.		√

Memory Read Multiple Transactions

Test	Description	N/A	Pass
25	Data transfer after memory read multiple from fast slave.		√
26	Data transfer after memory read multiple from medium slave.		√
27	Data transfer after memory read multiple from slow slave.		√
28	Data transfer after memory read multiple from subtractive slave.		√

Memory Read Line Transactions

Test	Description	N/A	Pass
29	Data transfer after memory read line from fast slave.		√
30	Data transfer after memory read line from medium slave.		√
31	Data transfer after memory read line from slow slave.		√
32	Data transfer after memory read line from subtractive slave.		√

Memory Write and Invalidate Transactions

Test	Description	N/A	Pass
33	Data transfer after memory write and invalidate to fast slave.	√	
34	Data transfer after memory write and invalidate to medium slave.	√	
35	Data transfer after memory write and invalidate to slow slave.	√	
36	Data transfer after memory write and invalidate to subtractive slave.	√	

Verify Proper Disconnect with Various Transfer Sizes (Xilinx-only test)

Note: Tests 1 through 36 are performed with a 4 double-word transfer size. Test 37 and 38 test for any effect with smaller transfer sizes. Single double-word transfers are tested in Scenario 1.4.

Test	Description	N/A	Pass
37	Data transfer after memory write and memory read operations with a 3 double-word burst transfer size.		√
38	Data transfer after memory write and memory read operations with a 2 double-word burst transfer size.		√

Test Scenario: 1.8 Multi-Data Phase & TRDY# Cycles

Memory Transactions

Test	Description	N/A	Pass
1	Verify that data is written to primary target when TRDY# is released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#		√
2	Verify that data is read from primary target when TRDY# is released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#		√
3	Verify that data is written to primary target when TRDY# is released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		√
4	Verify that data is read from primary target when TRDY# is released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		√
5	Verify that data is written to primary target when TRDY# is released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#		√
6	Verify that data is read from primary target when TRDY# is released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#		√
7	Verify that data is written to primary target when TRDY# is released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		√
8	Verify that data is read from primary target when TRDY# is released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		√
9	Verify that data is written to primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#		√
10	Verify that data is read from primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#		√
11	Verify that data is written to primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#		√
12	Verify that data is read from primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#		√

Dual Address Transactions

Test	Description	N/A	Pass
13	Verify that data is written to primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#	√	
14	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#	√	
15	Verify that data is written to primary target when TRDY# released after 4th rising clock edge and asserted on 5th rising clock edge after FRAME#	√	
16	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 5th rising clock edge after FRAME#	√	
17	Verify that data is written to primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#	√	
18	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#	√	
19	Verify that data is written to primary target when TRDY# released after 5th rising clock edge and asserted on 7th rising clock edge after FRAME#	√	
20	Verify that data is read from primary target when TRDY# released after 5th rising clock edge and asserted on 7th rising clock edge after FRAME#	√	
21	Verify that data is written to primary target when TRDY# alternately re-leased for one clock cycle and asserted for one clock cycle after FRAME#	√	
22	Verify that data is read from primary target when TRDY# alternately re-leased for one clock cycle and asserted for one clock cycle after FRAME#	√	
23	Verify that data is written to primary target when TRDY# alternately re-leased for two clock cycles and asserted for two clock cycles after FRAME#	√	
24	Verify that data is read from primary target when TRDY# alternately re-leased for two clock cycles and asserted for two clock cycles after FRAME#	√	

Memory Read Multiple Transactions

Test	Description	N/A	Pass
25	Verify that data is read from primary target when TRDY# released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#		√
26	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		√
27	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#		√
28	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		√
29	Verify that data is read from primary target when TRDY# alternately re-leased for one clock cycle and asserted for one clock cycle after FRAME#		√
30	Verify that data is read from primary target when TRDY# alternately re-leased for two clock cycles and asserted for two clock cycles after FRAME#		√

Memory Read Line Transactions

Test	Description	N/A	Pass
31	Verify that data is read from primary target when TRDY# released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#		√
32	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		√
33	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#		√
34	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		√
35	Verify that data is read from primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#		√
36	Verify that data is read from primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#		√

Memory Write and Invalidate Transactions

Test	Description	N/A	Pass
37	Verify that data is written to primary target when TRDY# released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#		√
38	Verify that data is written to primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		√
39	Verify that data is written to primary target when TRDY# released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#		√
40	Verify that data is written to primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		√
41	Verify that data is written to primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#		√
42	Verify that data is written to primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#		√

Test Scenario: 1.9 Bus Data Parity Error Single Cycles

Memory Transactions

Test	Description	N/A	Pass
1	Verify the IUT sets Data Parity Error Detected bit when Primary Target asserts PERR# on IUT Memory Write		√
2	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT Memory Read		√
3	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT Memory read		√

I/O Transactions

Test	Description	N/A	Pass
4	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT I/O Write		√
5	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT I/O Read		√
6	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT I/O read		√

Configuration Transactions

Note: Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
7	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT Configuration Write		√
8	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT Configuration Read		√
9	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT Configuration read		√

Test Scenario: 1.10 Bus Data Parity Error Multi-Data Phase Cycles

Memory Transactions

Test	Description	N/A	Pass
1	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT multi data phase Memory Write		√
2	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT multi data phase Memory Read		√
3	Verify the IUT sets Parity Error Detected bit when odd		√

Dual Address Transactions

Test	Description	N/A	Pass
4	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT dual address multi data phase Write	√	
5	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT dual address multi data phase Read	√	
6	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT dual address multi data phase read	√	

Configuration Transactions

Note: Configuration transactions tested using VirtualChips environment. Not available in VIEWsim testbench.

Test	Description	N/A	Pass
7	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT Configuration multi data phase Write		√
8	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT Configuration multi data phase Read		√
9	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT Configuration multi data phase read		√

Memory Read Multiple Transactions

Test	Description	N/A	Pass
10	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT memory read multiple data phase.		√
11	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT memory read multiple data phase.		√

Memory Read Line Transactions

Test	Description	N/A	Pass
12	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT memory read line data phase.		√
13	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT memory read line data phase.		√

Memory Write and Invalidate Transactions

Test	Description	N/A	Pass
14	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT Memory Write and Invalidate data phase.	√	

Test Scenario: 1.11 Bus Master Timeout

Note: Configuration transactions not tested.

Test	Description	N/A	Pass
1	Memory write transaction terminates before 4 data phases completed		√
2	Memory read transaction terminates before 4 data phases completed		√
3	Configuration write transaction terminates before 4 data phases completed	√	
4	Configuration read transaction terminates before 4 data phases completed	√	
5	Memory read multiple transaction terminates before 4 data phases		√
6	Memory read line transaction terminates before 4 data phases		√
7	Dual Address write transaction terminates before 4 data phases completed		√
8	Dual Address read transaction terminates before 4 data phases completed		√

Memory Write and Invalidate Transactions

Test	Description	N/A	Pass
9	Memory write invalidate terminates on line boundary	√	

Latency timer after 2 cycles, disconnect 2 clocks after DEVSEL# asserted (Xilinx-only test)

Test	Description	N/A	Pass
10	Memory write transaction terminates correctly.		√
11	Memory read transaction terminates correctly.		√

Test Scenario: 1.12 Target Lock

Test	Description	N/A	Pass
1	IUT does not perform bus transaction (read lock) on locked resource	√	
2	IUT does establish lock after lock is released	√	
3	IUT does release lock after write to primary target	√	
4	IUT does not establish lock when it detects retry	√	
5	IUT does not establish lock when it detects target abort	√	

Test Scenario: 1.13. PCI Bus Master Parking

Drive PCI bus to stable conditions if it is idle and GNT# is asserted.

Test	Description	N/A	Pass
1	IUT drives AD[31::00] to stable values within eight PCI Clocks of GNT#.		√
2	IUT drives C/BE[3::0]# to stable values within eight PCI Clocks of GNT#.		√
3	IUT drives PAR one clock cycle after IUT drives AD[31::0]		√

Tri-state the bus when GNT# is not asserted.

Test	Description	N/A	Pass
4	IUT Tri-states AD[31::00] and C/BE[3::0] and PAR when GNT# is re-leased.		√

Perform operations when starting in the DR_BUS state (Xilinx-only test). GNT# asserted before REQUEST asserted by user application.

Test	Description	N/A	Pass
5	IUT completes a Memory Write operation.		√
6	IUT completes a Memory Read operation.		√

Test Scenario: 1.14. PCI Bus Master Arbitration

Complete bus transaction when GNT# is de-asserted coincident with FRAME# asserted (2-cycle GNT#).

Test	Description	N/A	Pass
1	IUT completes write and read transaction when de-asserting GNT# is co-incident with asserting FRAME#.		√

Complete bus transaction when GNT# is asserted for only one cycle (Xilinx-only test).

Test	Description	N/A	Pass
2	IUT completes write and read transaction when GNT# is asserted for only one cycle.		√

Wait for idle bus (FRAME# and IRDY# de-asserted) after receiving GNT# before starting Initiator transaction (Xilinx-only test).

Test	Description	N/A	Pass
3	IUT waits for idle bus after receiving GNT# before completing write and read transaction.		√

LogiCore PCI Initiator does not attempt bus transaction if Bus Master Enable bit is not set in the Command Register (Xilinx-only test).

Test	Description	N/A	Pass
4	Reset Bus Master Enable bit in the IUT command register. Attempt a write transaction. Verify that the IUT does not attempt a transaction.		√

LogiCore PCI Initiator should attempt bus transaction if Bus Master Enable bit is set in the Command Register and a REQUEST is pending (Xilinx-only test).

Test	Description	N/A	Pass
5	Set Bus Master Enable bit in the IUT command register. Attempt a write transaction. Verify that the IUT successfully completes transaction.		√

Component Protocol Checklist for a Target Device

Test Scenario: 2.1. Target Reception of an Interrupt Cycle

Test	Description	N/A	Pass
1	IUT generates Interrupts when programmed	√	
2	IUT clears Interrupts when serviced (may include driver specific actions)	√	

Test Scenario: 2.2. Target Reception of Special Cycle

Test	Description	N/A	Pass
1	No DEVSEL# Assertion by IUT after Special Cycle		√
2	IUT receives encoded special cycle *Message received but not processed in macro. Would be processed by user application.*		

Test Scenario: 2.3. Target Detection of Address and Data Parity Error for Special Cycle

Test	Description	N/A	Pass
1	IUT reports address parity error via SERR#		√
2	IUT reports data parity error via SERR#	√	
3	IUT keeps SERR# active for at least one clock		√

Test Scenario: 2.4. Target Reception of I/O Cycles with Legal and Illegal Byte Enables

If Target does not support I/O cycles, mark 1 through 4 as “N/A” or if Target claims all 32 bits during an I/O cycle, mark 1 and 2 as “N/A”

Test	Description	N/A	Pass
1	IUT asserts TRDY# following 2nd rising edge from FRAME on all Legal BE"	√	
2	IUT terminates with TARGET Abort for each illegal BE"	√	
3	IUT asserts STOP	√	
4	IUT de-asserts STOP after FRAME de-assertion	√	

The LogiCore Target supports 32-bit I/O transfers. The macro does not automatically generate Target Abort or Disconnect during illegal transfers. However, this function can be added by the user application.

Test Scenario: 2.5. Target Ignores Reserved Commands

Test	Description	N/A	Pass
1	IUT does not respond to RESERVED COMMANDS. Commands issued are ‘0100’b, ‘0101’b, ‘1000’b, and ‘1001’b.		√
2	Initiator detects master abort for each transfer		√
3	IUT does not respond to 64-bit cycle (dual address)		√

Test Scenario: 2.6. Target Receives Configuration Cycles

If Target does not support Type 1 configuration cycles, mark 3 through 5 as "N/A"

Test	Description	N/A	Pass
1	IUT responds to all configuration cycles type 0 read/write cycles appropriately. Read and write to Base Address Registers 0 and 1. Rotate AD[1:0] from '00'b through '11'b. Check for appropriate response when AD[1:0]='00'b. Check for Master Abort condition on all other conditions.		√
2	IUT does not respond to configuration cycles type 0 with IDSEL inactive. Check for Master Abort condition.		√
3	IUT responds to all configuration cycles type 1 read/write cycles appropriately	√	
4	IUT responds to all configuration cycles type 0 read/write cycles appropriately	√	
5	IUT does not respond (master abort) on illegal configuration cycle types	√	

The LogiCore Target does not support burst transfers in or out of its configuration space.

Test Scenario: 2.7. Target Receives I/O Cycles with Address and Data Parity Errors

Test	Description	N/A	Pass
1	IUT reports address parity error via SERR# during I/O read/write cycles		√
2	IUT reports data parity error via PERR# during I/O write cycles		√

Test Scenario: 2.8. Target Receives Configuration Cycles with Address and Data Parity Errors

Test	Description	N/A	Pass
1	IUT reports address parity error via SERR# during configuration read/write cycles		√
2	IUT reports data parity error via PERR# during configuration write cycles		√

Test Scenario: 2.9. Target Receives Memory Cycles

Test	Description	N/A	Pass
1	IUT completes single memory read and write cycles appropriately. Rotate byte enables from '0000' to '0011'.		√
2	IUT completes memory read line cycles appropriately. Two double-word transfer. Rotate byte enables from '0000' to '0011'.		√
3	IUT completes memory read multiple cycles appropriately. Two double-word transfer. Rotate byte enables from '0000' to '0011'.		√
4	IUT completes memory write and invalidate cycles appropriately. Two double-word transfer. Rotate byte enables from '0000' to '0011'.		√
5	IUT completes one cycle and disconnects on RESERVED memory operations	√	
6	IUT disconnects on burst transactions that cross its address boundary	√	

The LogiCore Target supports burst transfers with Linear Burst Ordering. The macro does not automatically generate Disconnect during reserved memory operations nor does it automatically generate Target Abort when a burst transaction crosses an address boundary. However, these functions can be added by the user application.

Test Scenario: 2.10. Target gets Memory Cycles with Address and Data Parity Errors

Test	Description	N/A	Pass
1	IUT reports address parity error via SERR# during all memory read and write cycles		√
2	IUT reports data parity error via PERR# during all memory write cycles		√

Test Scenario: 2.11. Target gets Fast Back-to-Back Cycles

Test	Description	N/A	Pass
1	IUT responds to back to back memory writes appropriately	√	
2	IUT responds to memory write followed by memory read appropriately	√	
3	IUT responds to back to back memory writes with 2nd write selecting IUT	√	
4	IUT responds to memory write followed by memory read with read selecting IUT	√	

Test Scenario: 2.12. Target Performs Exclusive Access Cycles

Test	Description	N/A	Pass
1	IUT responds to exclusive access by initiator and accepts LOCK	√	
2	IUT responds with RETRY when second initiator attempts an access	√	
3	IUT responds to access releasing LOCK by initiator	√	
4	IUT responds to access by second initiator	√	

Test Scenario: 2.13. Target Receives Cycles with IRDY# Used for Data Stepping (IRDY# wait states)

Test	Description	N/A	Pass
1	IUT responds appropriately with a wait state inserted on phase 1 of 3 data phases. Perform Memory Write and Read.		√
2	IUT responds appropriately with a wait state inserted on phase 2 of 3 data phases. Perform Memory Write and Read.		√
3	IUT responds appropriately with a wait state inserted on phase 3 of 3 data phases. Perform Memory Write and Read.		√
4	IUT responds appropriately with a wait state inserted on all of 3 data phases. Perform Memory Write and Read.		√
5	IUT responds appropriately when Initiator has maximum initial latency, eight wait state before IRDY#- asserted. Perform Memory Write and Read. Xilinx-only test.		√

Test Scenario: 2.14. Target Signals and Responds to Various Target Termination Conditions (Xilinx-only tests)

Normal Termination Conditions (READY asserted, TERM de-asserted, T_ABORT de-asserted, KEEPOUT de-asserted).

Test	Description	N/A	Pass
1	IUT responds appropriately to a Memory Write operation, READY asserted immediately.		√
2	IUT responds appropriately to a Memory Write operation, READY asserted immediately.		√
3	IUT responds appropriately to a Memory Write operation, READY asserted after 7 clocks.		√
4	IUT responds appropriately to a Memory Write operation, READY asserted after 7 clocks.		√

Target Disconnect (READY asserted, TERM asserted, T_ABORT de-asserted, KEEPOUT de-asserted).

Test	Description	N/A	Pass
5	IUT responds appropriately to a Memory Write operation, disconnect with data on first cycle.		√
6	IUT responds appropriately to a Memory Write operation, disconnect with data on first cycle.		√
7	IUT responds appropriately to a Memory Write operation, disconnect with data after 7 cycles, READY and TERM asserted after 7 clocks.		√
8	IUT responds appropriately to a Memory Read operation, disconnect with data after 7 cycles, READY and TERM asserted after 7 clocks.		√
9	IUT responds appropriately to a Memory Write operation with 2 double words, disconnect on second word.		√
10	IUT responds appropriately to a Memory Read operation with 2 double words, disconnect on second word.		√

Target Retry (READY de-asserted, TERM asserted, T_ABORT de-asserted, KEEPOUT de-asserted on first transfer).

Test	Description	N/A	Pass
11	IUT responds appropriately to a Memory Write operation.		√
12	IUT responds appropriately to a Memory Read operation.		√
13	IUT responds appropriately to a Memory Write operation. Initiator asserts FRAME# and de-asserts IRDY# for seven cycles. Verify IUT asserts STOP# until FRAME# de-asserted.		√
14	IUT responds appropriately to a Memory Read operation. Initiator asserts FRAME# and de-asserts IRDY# for seven cycles. Verify IUT asserts STOP# until FRAME# de-asserted.		√

Keepout (READY de-asserted, TERM asserted, T_ABORT de-asserted, KEEPOUT asserted).

Test	Description	N/A	Pass
15	IUT responds to a Memory Write operation with a Target Retry. Verify that ADIO internal bus is Hi-Z.		√
16	IUT responds to a Memory Read operation with a Target Retry. Verify that ADIO internal bus is Hi-Z.		√
17	IUT responds to a Configuration Write operation with a Target Retry. Verify that ADIO internal bus is Hi-Z.		√
18	IUT responds to a Configuration Read operation with a Target Retry. Verify that ADIO internal bus is Hi-Z.		√

Target Abort (T_ABORT asserted, KEEPOUT de-asserted).

Test	Description	N/A	Pass
19	IUT responds appropriately to a two double-word Memory Write operation with a Target Abort.		√
20	Verify that Signaled Target Abort bit was set in the Status Register.		√
21	IUT responds appropriately to a two double-word Memory Read operation with a Target Abort.		√
22	Verify that Signaled Target Abort bit was set in the Status Register.		√

Timing Specification

CE39.	Component is operational at any frequency between DC and 33 MHz? Note: "na" implies component intended for motherboard use only To satisfy this requirement, designs are allowed to require software to place the component in the proper state before stopping the clock and return it to an operational state after restarting the clock.	yes _√_ na___
CE40.	Component is operational with a CLK High Time of 11 nS for 33 MHz PCI, 6 ns for 66 MHz PCI?	na ___ yes _√_ no___
CE41.	Component is operational with a CLK Low Time of 11 nS for 33 MHz PCI, 6 ns for 66 MHz PCI?	na ___ yes _√_ no___
CE42.	All bussed signals are driven valid between 2 and 11 nS after CLK for 33 MHz PCI, between 2 and 6 ns for 66 MHz PCI?	yes _√_ no___
CE43.	REQ# and GNT# signals are driven valid between 2 and 12 nS after CLK for 33 MHz PCI, between 2 and 6 ns for 66 MHz PCI?	yes _√_ no___
CE44.	All Tri-state signals become active no earlier than 2 nS after CLK?	yes _√_ no___
CE45.	All Tri-state signals float no later than 28 nS after CLK for 33 MHz PCI, no later than 14 nS for 66 MHz PCI?	yes _√_ no___
CE46.	All bussed inputs require no more than 7 nS setup to CLK for 33 MHz PCI, no more than 3 nS for 66 MHz PCI?	yes _√_ no___
CE47.	REQ# requires no more than 12 nS setup to CLK for 33 MHz PCI, no more than 5 nS for 66 MHz PCI?	na ___ yes _√_ no___
CE48.	GNT# requires no more than 10 nS setup to CLK for 33 MHz PCI, no more than 5 nS for 66 MHz PCI?	na ___ yes _√_ no___
CE49.	All inputs require no more than 0 nS of hold time after CLK? Design must used the placement and routing guide file provided with the LogiCore product.	yes _√_ no___
CE50.	All outputs are Tri-stated within 40 nS after RST# goes low?	yes _√_ no___

all timings (CE39 through C50) verified by (check all that apply):
__√_ static timing design tools (XDelay Static Timing Analyzer)
____ dynamic timing design tools (Verilog, Qsim, Quicksim, ViewSim, VHDL, ...)
__√_ silicon AC testing
____ other ____
note: maximum and minimum timings assume different output loadings for both 5.0V and 3.3V parts. See PCI Spec Rev 2.1 page 134 note #2.

Potential Discrepancies

Master Abort

The LogiCore Initiator does detect and respond to a Master Abort condition (no DEVSEL# asserted by the addressed Target). However, the LogiCore Initiator de-asserts FRAME# and IRDY# one cycle later than the diagram shown as Figure 3-4 on page 40 in the PCI Local Bus Specification, revision 2.1. Otherwise, the LogiCore Initiator treats Master Abort normally.

Master Abort is intended to keep an Initiator from "hanging" the bus when addressing a non-existent or malfunctioning Target. A one cycle latency should not adversely affect most designs. The extra clock cycle eliminates a critical path in the LogiCore Initiator control logic.

Target Disconnect after Reserved Memory Operation

The LogiCore Target supports burst transfers with Linear Burst Ordering. The macro does not automatically generate Disconnect during reserved memory operations nor does it automatically generate Target Abort when a burst transaction crosses an address boundary. However, these functions can be added by the user application.

XILINX® The Programmable Logic Company℠

Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
U.S.A.

Tel: 1 (800) 255-7778
or 1 (408) 559-7778
Fax: 1 (800) 559-7114

Net: hotline@xilinx.com
Web: http://www.xilinx.com

North America

Irvine, California
(714) 727-0780

Englewood, Colorado
(303)220-7541

Sunnyvale, California
(408) 245-9850

Schaumburg, Illinois
(847) 605-1972

Nashua, New Hampshire
(603) 891-1098

Raleigh, North Carolina
(919) 846-3922

West Chester, Pennsylvania
(610) 430-3300

Dallas, Texas
(214) 960-1043

Europe

Xilinx Sarl
Jouy en Josas, France
Tel: (33) 1-34-63-01-01
Net: frhelp@xilinx.com

Xilinx GmbH
Aschheim, Germany
Tel: (49) 89-99-1549-01
Net: dlhelp@xilinx.com

Xilinx, Ltd.
Byfleet, United Kingdom
Tel: (44) 1-932-349401
Net: ukhelp@xilinx.com

Japan

Xilinx, K.K.
Tokyo, Japan
Tel: (03) 3297-9191

Asia Pacific

Xilinx Asia Pacific
Hong Kong
Tel: (852) 2424-5200
Net: hongkong@xilinx.com