PCI-X System Architecture

MindShare, Inc.
Tom Shanley
Are your company’s technical training needs being addressed in the most effective manner?

MindShare has over 25 years experience in conducting technical training on cutting-edge technologies. We understand the challenges companies have when searching for quality, effective training which reduces the students’ time away from work and provides cost-effective alternatives. MindShare offers many flexible solutions to meet those needs. Our courses are taught by highly-skilled, enthusiastic, knowledgeable and experienced instructors. We bring life to knowledge through a wide variety of learning methods and delivery options.

training that fits your needs

MindShare recognizes and addresses your company’s technical training issues with:

- Scalable cost training
- Customizable training options
- Just-in-time training
- Overview and advanced topic courses
- Reducing time away from work
- Training delivered effectively globally
- Concurrently delivered multiple-site training
- Training in a classroom, at your cubicle or home office

MindShare training courses expand your technical skillset

- **PCI Express 2.0®**
- **Intel Core 2 Processor Architecture**
- **AMD Opteron Processor Architecture**
- **Intel 64 and IA-32 Software Architecture**
- **Intel PC and Chipset Architecture**
- **PC Virtualization**
- **USB 2.0**
- **Wireless USB**
- **Serial ATA (SATA)**
- **Serial Attached SCSI (SAS)**
- **DDR2/DDR3 DRAM Technology**
- **PC BIOS Firmware**
- **High-Speed Design**
- **Windows Internals and Drivers**
- **Linux Fundamentals**
- ... and many more.

All courses can be customized to meet your group’s needs. Detailed course outlines can be found at [www.mindshare.com](http://www.mindshare.com)

*PCI Express ® is a registered trademark of the PCISIG*
Engage MindShare
Have knowledge that you want to bring to life? MindShare will work with you to “Bring Your Knowledge to Life.” Engage us to transform your knowledge and design courses that can be delivered in classroom or virtual classroom settings, create online eLearning modules, or publish a book that you author.

We are proud to be the preferred training provider at an extensive list of clients that include:

- ADAPTEC
- AMD
- AGILENT TECHNOLOGIES
- APPLE
- BROADCOM
- CADENCE
- CRAY
- CISCO
- DELL
- FREESCALE
- GENERAL DYNAMICS
- HP
- IBM
- KODAK
- LSI LOGIC
- MOTOROLA
- MICROSOFT
- NASA
- NATIONAL SEMICONDUCTOR
- NETAPP
- NOKIA
- NVIDIA
- PLX TECHNOLOGY
- QLOGIC
- SIEMENS
- SUN MICROSYSTEMS
- SYNOPSYS
- TI
- UNISYS

www.mindshare.com
PCI-X System
Architecture

First Edition

MINDSHARE, INC.

TOM SHANLEY

ADDISON-WESLEY DEVELOPER'S PRESS
Reading, Massachusetts • Harlow, England • Menlo Park, California
New York • Don Mills, Ontario • Sydney
Bonn • Tokyo • Amsterdam • Mexico City • Seoul
San Juan • Madrid • Singapore • Paris • Taipei • Milan
Part 1:
Basic Concepts

Chapter 1: PCI Needed Improvement
Wait States Yield Poor Performance ................................................................. 13
Relatively Slow Clock Speed ............................................................................. 14
Transfer Size Unknown ...................................................................................... 14
PCI Delayed Transactions Are Inefficient ........................................................ 15
  Initiator Retries Use Up Valuable Bus Time .................................................... 15
  Initiator Doesn’t Supply Transfer Count ........................................................ 15
  Delayed Completion ......................................................................................... 16
  Initiator’s Transaction ID Is Sketchy at Best .................................................. 16
Snoops Hurt Performance .................................................................................. 16
  Host/PCIX Bridge Knows AGP’s Area of Memory Is Non-Cacheable ............. 16
  Memory Used By PCI Masters Mayor May Not Be Cached ......................... 17
  Snoops Slow Down PCI Accesses To Main Memory ....................................... 17
  Snoop Traffic On Processor Bus Can Hurt Processor(s) ................................. 17
  Main Memory Less Available To Processors .................................................. 17
Latency Timer Use Not Optimized in PCI ......................................................... 19
  Min_Gnt Register: Timeslice Request ............................................................ 19
Dynamic Traffic Analysis and Loading Tuning................................................................. 40
Data Phase Parity Error Recovery ....................................................................................... 41
  General ......................................................................................................................... 41
  Recovery Usually Not Possible In PCI ........................................................................ 41
  PCI-X Chipset Does Not Monitor PERR# ................................................................ 42
  What the PCI-X Spec Says ....................................................................................... 42
  Requester Hardware Doesn’t Handle Error Itself....................................................... 42
  With Appropriate Master and Device Driver Design, Recovery Possible .............. 42
  Device Driver With No Recovery Capability = SERR# ........................................... 43
  If Bus In PCI Mode, Bridge Monitors PERR# .......................................................... 43
64-Bit Connection Indication ............................................................................................ 44
MSI Feature Mandatory ................................................................................................. 44
Power Management Mandatory for Add-In Devices ....................................................... 44
Snoops Can Be Eliminated ......................................................................................... 44
Memory BARS Must Be 64-bit Width ........................................................................ 45
Bus Masters That Access Memory Must Support DAC Command .......................... 45
Stepping Eliminated ..................................................................................................... 45
Fast Back-to-Back Transactions Eliminated .................................................................. 46

Chapter 3: Lowest Common Denominator Defines Mode
Bus Protocol/Speed = Lowest Common Denominator .................................................... 48
Discovering a PCI-X Bus ............................................................................................... 48
Discovering PCI Devices On a PCI-X Bus .................................................................... 49
Some Example Systems ............................................................................................... 49
  System With No Connectors On PCI-X Bus ............................................................... 49
  Single Bus System With Connectors ........................................................................ 49
Systems Supporting Both PCI-X and PCI Environments .......................................... 52
  Introduction .............................................................................................................. 52
  System With Single Host/PCIX Bridge ................................................................. 52
  Dual Host/PCIX Bridge System ............................................................................... 54

Chapter 4: Device Types and Bus Initialization
All Devices Support 33MHz PCI .................................................................................... 56
  General ......................................................................................................................... 56
  133MHz PCI-X Device Must Support 33MHz and 66MHz PCI .............................. 56
  When M66EN Is Grounded on a Card .................................................................... 56
    Effect of Grounded M66EN On a PCI Source Bridge ......................................... 56
    Effect of Grounded M66EN On a PCI-X Source Bridge ....................................... 56
  M66EN Usage On a 66MHz PCI Card or a PCI-X Card ........................................ 57
All PCI-X Devices Support 66MHz PCI-X Mode .............................................................. 57
PCIXCAP Indicates Protocol/Frequency Required ..................................................... 58
## Contents

General........................................................................................................................ 58

PCIXCAP Indicates Capabilities of Card’s 1st Device................................................. 61

Device Not Permitted to Use PCIXCAP As Input or Output ......................................... 61

Bridge’s Interpretation of M66EN and PCIXCAP.......................................................... 61

Maximum Reliable Speed Verified By Design and Testing.......................................... 63

Supplying PCI-X Devices With Protocol and Speed....................................................... 64

General........................................................................................................................ 64

Host/PCIX Bridge Pattern Delivery Sequence.............................................................. 64

PCIX-to-PCIX Bridge Pattern Delivery Sequence.......................................................... 65

The Sequence ............................................................................................................. 65

But First......................................................................................................................... 65

Why Bus Must Be Idle When Init Pattern Driven.......................................................... 67

Upon Receipt of Pattern, Device Initializes Itself.......................................................... 67

Init Pattern Setup and Hold Time .................................................................................. 67

General........................................................................................................................ 67

Relationship With Trhff............................................................................................... 68

Reassertion of RST# Necessitates Redelivery of Pattern.............................................. 68

General........................................................................................................................ 68

Secondary Bus RST# Follows RST# On Primary Side.................................................... 68

Secondary Bus RST# Under Software Control............................................................... 69

Bridge Must Support Interfaces in Different Modes..................................................... 70

Hot-Plug PCI-X Bus Initialization.................................................................................. 71

Some Systems Permit Hot-Plug of PCI-X Cards............................................................ 71

For Background On PCI Hot-Plug................................................................................. 71

Problems Associated With PCI-X Hot-Plug................................................................. 71

Determination of Card Capabilities.............................................................................. 72

General........................................................................................................................ 72

Determining the Presence of a PCI Card........................................................................ 73

Unacceptable Method.................................................................................................... 73

Acceptable Method....................................................................................................... 73

Determining the Presence of a 66MHz PCI Card.......................................................... 74

Determining the Presence of a 66MHz-Capable PCI-X Card......................................... 74

Determining the Presence of a 133MHz-Capable PCI-X card....................................... 75

Hot-Install of Card With PCIX-to-PCIX Bridge On It.................................................. 75

Description.................................................................................................................. 75

How To Avoid Bus Renumbering During Run-Time...................................................... 76

Hot-Install of Card With PCI-to-PCI Bridge On It....................................................... 77

Checking PCIXCAP Without Power Applied To Card.................................................. 78

Changing Secondary Bus Number Configuration Register......................................... 78

Conservation of Address Space..................................................................................... 78

General........................................................................................................................ 78

Requesting Memory Space........................................................................................... 79
Contents

Requesting IO Space ................................................................. 80
Early Configuration Access To Newly-Installed Device .............. 80
Possible Adjustment of Max Memory Read Byte Count Values .... 80
Placing Device in Low-Power Mode Requires Quiesce ............... 81

Chapter 5: PCI-X Is a Registered Bus
PCI-X Is a Low-Voltage Swing (LVS) Bus ..................................... 83
Introduction To the Registered Nature of the Bus ....................... 84
Address/Command Decode Example ........................................... 85
Data Read Example .................................................................. 96

Chapter 6: Intro to Commands
Commands Fall Into Two Categories .......................................... 89
Command Encoding ................................................................... 90
Dword Commands ...................................................................... 92
  General .................................................................................. 92
  IO Read and Write Commands .............................................. 92
  Memory Read Dword Command ............................................ 92
  Configuration Read and Write Commands ............................. 93
  Interrupt Acknowledge Command ........................................ 93
  Special Cycle Command ...................................................... 95
Burst Commands ....................................................................... 95
  All Burst-Oriented Commands Are Memory Transfers ........... 95
  Start Address Is Byte-Aligned ............................................ 95
  Transfer Length .................................................................... 95
  Linear Addressing Is Implied .............................................. 96
  Memory Read Block Command .......................................... 96
  Alias To Memory Read Block Command ............................... 96
  Memory Write Block Command .......................................... 97
  Alias To Memory Write Block Command .............................. 97
  Memory Write Command .................................................... 97
  Split Completion Command .............................................. 97
Dual-Address Cycle (DAC) Command ........................................ 98

Chapter 7: Intro to Transaction Phases
The PCI Transaction Phases ...................................................... 100
The PCI-X Transaction Phases .................................................. 100
  Address Phase ................................................................. 100
  Attribute Phase ............................................................... 100
  Response Phase .............................................................. 101
  Data Phase(s) ..................................................................... 101
## Contents

Split Transactions .................................................................................................................. 118
  What Problem Do Split Transactions Solve?................................................................. 118
  Split Completion Uses Sequence ID As the Address....................................................... 118
  Requester and Completer May Reside On the Same Bus ............................................. 119
    Example...................................................................................................................... 119
  Split Completion May Contain a Message.................................................................... 119
  Memory Read May Result In Multiple Split Completion Transactions ..................... 120
    Completer Has All Data Buffered Up: One Split Completion................................. 120
    Some Data Buffered Up: Multiple Split Completions................................................. 120
  Requester and Completer May Reside On Different Buses.......................................... 121
    Example...................................................................................................................... 121
      Case One: Completer Issues a Split Response....................................................... 122
      Case Two: Completer Transfers Data Immediately ................................................. 122
  Requester Buffer Space Management ........................................................................... 124
    Allocate Buffer To Hold All of the Data, Then Arbitrate......................................... 124
    On Split Response, Requester Must Commit Buffer Space...................................... 125
    On Immediate Response, Requester Can Release Buffer Space.............................. 125
  Bridge Can Retry or Disconnect Split Completion......................................................... 125
  Requesters Must Implement a Split Request Timeout..................................................... 125

Part 2:
Transaction Protocol

Chapter 10: Bus Arbitration

Stepping Not Permitted ........................................................................................................ 129
Request and Grant Signals Are Registered ...................................................................... 129
Example Arbitration ......................................................................................................... 130
Device Design Rules ......................................................................................................... 133
  No Fast Back-to-Back.................................................................................................... 133
  REQ# Can Be Asserted or Deasserted At Any Time.................................................... 134
  Issuing REQ# and Then Changing Your Mind............................................................. 134
  After REQ# Asserted, Assert FRAME# or Deassert REQ#......................................... 134
  Initiator Can Start Transaction Two Clocks After GNT# Asserted............................ 134
  GNT# Removal In Clock Prior To FRAME# Too Late To Stop Initiation..................... 135
  Single Idle Clock Between Transactions.................................................................... 135
  Multiple Idle Clocks Between Transactions............................................................... 137
  Back-to-Back Transactions By Same Initiator............................................................ 138
  Device Containing Multiple Initiators......................................................................... 138

Arbiter Design Rules.......................................................................................................... 138
  No GNT# Asserted and Arbiter Must Issue GNT#..................................................... 138
  Must Allow Opportunity To Start Configuration Transaction................................. 139
Contents

When Bus Not Idle, GNT# Can Be Deasserted Sooner .............................................. 140
If Transaction Not Started, Arbiter May Ignore Master .............................................. 140
Removing GNT# From One Initiator and Giving It To Another .............................. 141
Parking Recommendation ............................................................................................. 141

Bus Parking ............................................................................................................................. 141
Floating Bus Causes Power Drain ................................................................................. 141
Bus Parking Prevents Bus Float ..................................................................................... 141
Rules Associated with Bus Parking ............................................................................... 142

How the Initiator Deals With Preemption ................................................................. 143
The Basics .......................................................................................................................... 143
After Timeslice Exhaustion and Preemption, How Soon Must It Yield? ............ 144
What Is the Recommended LT Value?.......................................................................... 144
How Much Data Can Be Transferred During Default Timeslice? ............................ 144

Chapter 11: Detailed Command Description

Dword Commands ................................................................................................................. 147
General ................................................................. 147
Command Encoding ....................................................... 148
Illegal To Assert REQ64# In Dword Transactions ...................................................... 148
IO Read and Write Commands ............................................................. 149
Basic Description .............................................................................................................. 149
Start Address and Byte Enable Format ....................................................................... 149
Start Address Is Byte-Aligned ...................................................................................... 149
Byte Enable Usage ........................................................................................................... 149
IO Transaction With No Byte Enables Asserted ............................................................. 149
Target Response To an IO Access ................................................................................. 151
General .................................................................................................................................. 151
Completer Handling of an IO Access ............................................................................. 151
When Bridge Acts as Target of IO Access ..................................................................... 151
Memory Read Dword Command ..................................................................................... 152
Start Address and Byte Enable Format ....................................................................... 152
Start Address Is Byte-Aligned ...................................................................................... 152
Byte Enable Usage ........................................................................................................... 152
Transaction With No Byte Enables Asserted ................................................................. 152
Target Response To an Memory Read Dword Access ................................................. 152
General .................................................................................................................................. 152
Completer Handling of a Memory Read Dword Access .............................................. 152
Bridge Acts As Target of Memory Read Dword Access ............................................... 153
Configuration Read and Write Commands ..................................................................... 153
Interrupt Acknowledge Command .................................................................................. 154
General .................................................................................................................................. 154
Background ....................................................................................................................... 154
Contents

Memory Write Command .................................................................................................................. 174
  Basic Description....................................................................................................................... 174
  Can Be Purely Speculative........................................................................................................ 175
  Target Never Permitted To Split a Memory Write ................................................................. 175
  Target Response To Memory Write Transaction ..................................................................... 175
  Doesn’t Need All of the Write Data Ready Before Initiating ................................................... 175

Split Completion Command ........................................................................................................ 176
  Basic Description....................................................................................................................... 176
  At Minimum, Completer Must Handle One Split Transaction ................................................. 176
  Can Be Initiated Even If Bus Master Bit Off............................................................................. 176
  Split Transactions Never Associated With Memory Writes ................................................... 177
  One or More Split Completions To Fulfill One Request .......................................................... 177
  Separate Requests May Be Fulfilled Out of Order ................................................................. 177
  The Address Phase ................................................................................................................... 178
  The Attribute Phase................................................................................................................ 178
  Claiming a Split Completion Transaction ................................................................................ 180
    Requester and Completer Are On the Same Bus ................................................................. 180
    Requester and Completer Are On Different Buses ............................................................... 181
    Master Abort or Target Abort on a Split Completion ............................................................ 182
  Treatment of Byte Enables ..................................................................................................... 182
  Requester Must Accept the Split Completion Data ................................................................. 182
  When Bridge Is Target of Completion, Can Retry or Disconnect .......................................... 182
  Initiator of Split Completion Can Disconnect ......................................................................... 183
  After Disconnect, Pick Up Where You Left Off ....................................................................... 183
  Split Completion Messages ...................................................................................................... 183
  To Disconnect First Split Completion At Imminent ADB ......................................................... 184
  Disconnecting First Split Completion Is Problematic ............................................................... 184
    The Problem ........................................................................................................................ 184
    The Solution ........................................................................................................................ 185
  Bridge Buffer Space Problem Due To Corrupted Split Completion ....................................... 186

Dual-Address Cycle (DAC) Command ...................................................................................... 187

Chapter 12: Latency Rules

Initiator Latency Rules .................................................................................................................. 189
  Don’t Start Transfer If You’re Not Ready .................................................................................. 189
  No Initiator Wait States Permitted...Ever ................................................................................. 190
  Behavior When Preempted ....................................................................................................... 191

Target Latency Rules .................................................................................................................... 191
  Target Response Time During Initialization Period ............................................................... 191
  What’s Going On During Initialization Time? .......................................................................... 191
  Definition of Initialization Period In PCI .................................................................................. 191
  Definition of Initialization Period In PCI-X ............................................................................ 192
Contents

Initialization Period and Hot-Plug ................................................................. 192
Target Can Ignore 16-Clock Rule During ROM Shadowing ........................ 192
Target Response Time Limit During Run-Time ............................................ 192
Response Time Limit When No Data Transferred ....................................... 192
Response Time Limit When Data Transferred ............................................ 193
Host/PCIX Bridge Must Obey 16 Clock Rule .............................................. 193
  In PCI ......................................................................................................... 193
  In PCI-X ................................................................................................. 193
Subsequent Data Phase Target Latency Rule ............................................... 193

Maximum Completion Time ....................................................................... 193
  In PCI ........................................................................................................ 193
How PCI-X Is Different ............................................................................. 194
How Can You Prevent Violation of This Time Limit? .............................. 195

Chapter 13: The Address, Attribute and Response Phases
All Transactions Begin With Address and Attribute Phases .................. 198
Memory Transaction May Have Two Address Phases .............................. 199
Attributes Always Delivered On Lower-Half of Bus ................................ 200
Address/Attribute Format Depends On Command Type ....................... 200
Memory Burst Format ............................................................................... 200
  Description ............................................................................................ 200
  No Snoop Attribute Bit ......................................................................... 201
  Background ............................................................................................ 201
    Bridge Knows AGP’s Area of Memory Is Non-Cacheable ...................... 201
    Memory Used By PCI Masters May or May Not Be Cached .................... 202
    Snoops Slow Down PCI Accesses To Main Memory .............................. 202
    Snoop Traffic On Processor Bus Can Hurt Processor(s) ....................... 202
    Main Memory Less Available To Processors ........................................ 202
    PCI-X Driver Can Make Requester’s Buffer Uncacheable .................... 202
    Only Host/PCIX Bridge Pays Attention To NS Bit ............................... 203
    When NS Is Set and CPU Has a Lock in Force .................................... 203
    NS Must Be 0 In Some Transactions .................................................. 203
Dword Command (other than Config) Format ........................................ 205
  Memory Read Dword and IO Format ...................................................... 205
  Special Cycle and Interrupt Acknowledge Format ................................. 206
Configuration Command Format .............................................................. 206
  Two Types of Configuration Transactions .......................................... 206
    Type Zero Configuration Access ......................................................... 207
    Type One Configuration Access ......................................................... 207
  Type 0 Configuration Access Format ................................................... 207
  Type 1 Configuration Access Format ................................................... 208
Split Completion Command Format ......................................................... 209
Contents

The Response Phase: Connecting With the Target .......................................................... 210

Chapter 14: Dword Transactions
General ..................................................................................................................................... 216
General Format of Timing Diagram Descriptions .......................................................... 216
IO Read and Memory Read Dword .................................................................................... 216
   Example One ................................................................................................................. 216
   Example Two ................................................................................................................. 220
IO Write ................................................................................................................................ 224
   Example One ................................................................................................................. 224
   Example Two ................................................................................................................. 227
   Example Three .............................................................................................................. 230
Configuration Read and Write Transactions ..................................................................... 234
Interrupt Acknowledge Command ..................................................................................... 234
Special Cycle Command ................................................................................................... 235

Chapter 15: Burst Transactions
Introduction ......................................................................................................................... 240
   Short Transfer Within a Block .................................................................................... 240
   Long Transfer, But Disconnect On First Block Boundary .......................................... 240
General Format of Timing Diagram Descriptions .......................................................... 240
Memory Read Block Transaction ....................................................................................... 241
   Memory Read Block: Detailed Example .................................................................. 241
   Memory Read Block: Variation One ............................................................................ 245
   Memory Read Block: Variation Two ........................................................................... 246
   Memory Read Block: Variation Three ........................................................................ 247
   Memory Read Block: Variation Four ......................................................................... 248
   Memory Read Block: Variation Five .......................................................................... 249
   Memory Read Block: Variation Six ............................................................................ 250
   Memory Read Block: Variation Seven ....................................................................... 251
   Memory Read Block: Variation Eight ....................................................................... 252
Memory Write Block Transaction ....................................................................................... 253
   Memory Write Block: Detailed Example .................................................................. 253
   On Burst Writes, Insert Wait States In Pairs ............................................................. 257
Memory Write Transaction ............................................................................................... 260
   Memory Write: Detailed Example ............................................................................. 261
   Memory Write: Variation One .................................................................................... 265
   Memory Write: Variation Two .................................................................................... 266
   Memory Write: Variation Three ................................................................................ 267
   Memory Write: Variation Four ................................................................................... 268
   Memory Write: Variation Five .................................................................................... 269
## Contents

Memory Write: Variation Six ................................................................. 270  
Memory Write: Variation Seven .......................................................... 271  
Memory Write: Variation Eight ............................................................ 272  
Memory Write: Variation Nine ............................................................. 273  

Split Completion Transaction ............................................................... 274  
  Split Completion Returning Block of Read Data .............................. 275  
  Returning a Split Completion Error Message ................................. 276  

Alias To Memory Block Commands ..................................................... 277  

Chapter 16: Transaction Terminations  
General Format of Timing Diagram Descriptions .................................. 279  

Termination By the Initiator ................................................................. 280  
  General ........................................................................................... 280  
  Byte Count Satisfaction ................................................................. 280  
    Introduction ............................................................................... 280  
    Ending Transaction of Four or More Data Phases ...................... 281  
    Ending Transaction of Less Than Four Data Phases ................ 281  
      Introduction ........................................................................... 282  
      Three Data Phase Transaction .............................................. 282  
      Two Data Phase Transaction ................................................. 284  
      One Data Phase Transaction ............................................... 286  
  Initiator Issues Disconnect At Next ADB ....................................... 289  
  Initiator Termination Due To Connection Timeout ....................... 290  

Termination By the Target ................................................................. 290  
  Target Abort ................................................................................ 290  
    Target Abort Is Always Fatal .................................................... 290  
  Some Reasons Target Issues Target Abort .................................... 290  
    Broken Target ......................................................................... 290  
    IO Addressing Error ............................................................... 290  
    Address Phase Parity Error ...................................................... 290  
  Initiator’s and Target’s Response to Target Abort ....................... 290  
  Example Target Abort Issued in First Data Phase ....................... 291  
  Example Target Abort Issued In Subsequent Data Phase ............ 293  
  Target Issues a Retry .................................................................. 294  
  Target Issues Single Data Phase Disconnect ............................... 297  
    General .................................................................................... 297  
    Special Case Scenario ............................................................. 299  
  Target Issues Disconnect At Next ADB ....................................... 299  
    Disconnect Issued Four or More Data Phases From ADB .......... 300  
    Target Issues Disconnect Too Close To Block Boundary ............ 301  

---

xvii
## Contents

To Disconnect When Start Address Very Close To ADB ........................................ 302
Start Address Three Data Phases From Block Boundary ................................. 303
Start Address Two Data Phases From Block Boundary .................................... 304
Start Address One Data Phase From Block Boundary ..................................... 306
Target Issues a Split Response ............................................................................ 308
Split Response For a Read .................................................................................. 308
Split Response For an IO or Configuration Write ............................................. 310

### Chapter 17: Split Completion Messages

Purpose of Split Completion Messages ......................................................... 313
SCM Always Terminates a Sequence ............................................................. 315
Upon Receipt of Error Message, Set Status Bit ............................................... 315
Message Format .............................................................................................. 316
  Address Phase Format .................................................................................... 316
  Attribute Phase Format ................................................................................. 316
  Data Phase Message Format ......................................................................... 317
Write Completion Indication ............................................................................. 322
  Good Completion of Split IO or Configuration Write .................................. 322
  Bad Completion of a Split IO or Configuration Write .................................... 323
Read Completion Indication .............................................................................. 323
  Good Completion of a Split Dword Read ....................................................... 323
  Good Completion of a Split Burst Memory Read .......................................... 323
  Bad Completion of a Split Dword Read ......................................................... 324
  Bad Completion of a Split Burst Memory Read ............................................. 324
  General ........................................................................................................ 324
  Remaining Byte Count and Remaining Lower Address Fields ..................... 324
Device-Specific Error Handling ....................................................................... 325

### Chapter 18: 64-Bit Transactions

General Format of Timing Diagram Descriptions .......................................... 327
64-bit Data Transfers and 64-bit Addressing: Separate Capabilities ................ 328
64-Bit Extension Signals .................................................................................... 328
REQ64# and ACK64# Have Same Timing As FRAME# and DEVSEL# .......... 329
In Attribute Phase, Upper Bus Reserved and Driven High ............................... 330
Block Length Remains the Same ................................................................. 330
Bursts Cannot Cross 264 Boundary ................................................................. 330
REQ64# Not Permitted in Dword Transactions ................................................. 330
  General ....................................................................................................... 330
  ...Unless Its a Split Completion ................................................................. 331
MSI Write Always Writes a Single 32-bit Data Value ........................................ 331
Bridge Must Support DAC on Both Interfaces ................................................. 331
## Contents

Width of Function’s Connection to Bus ................................................................. 332
  General .................................................................................................................. 332
  Add-In Card With a Bridge ................................................................................. 334
  Determining the Width of a Bridge’s Interfaces ................................................. 334

64-bit Cards in 32-bit Add-in Connectors ............................................................... 335

Pullups Prevent 64-bit Extension from Floating When Not in Use ..................... 336
  Problem: a 64-bit Card in a 32-bit PCI Connector ........................................... 336
  How 64-bit Card Determines Type of Slot Installed In ..................................... 338

64-bit Data Transfer Capability ........................................................................... 339
  64-bit Transfers: Only Burst Memory Operations ............................................. 340
    Start Address Byte-Aligned ............................................................................ 340
    64-bit Target’s Interpretation of Address ......................................................... 341
    32-bit Target’s Interpretation of Address ......................................................... 341
  64-bit Memory Read Block with 64-bit Target ................................................... 341
  64-bit Write Block or Split Completion With 64-bit Target ............................... 347
  64-bit Memory Write with 64-bit Target ............................................................ 352
    Start Address Alignment Defines Data Path Usage ........................................ 354
    Introduction .................................................................................................... 354
    64-bit to 64-bit Connection ............................................................................. 354
    64-bit to 32-bit Connection ............................................................................. 354
  64-bit Memory Reads from 32-bit Targets ......................................................... 359
    Starting On Even Dword ............................................................................... 359
    Starting On Odd Dword ................................................................................. 365
  64-bit Memory Writes To 32-bit Targets ............................................................ 366
    Starting On Even Dword ............................................................................... 366
    Starting On Odd Dword ................................................................................. 368
      Example One ............................................................................................... 368
      Example Two ............................................................................................... 369
      Example Three ........................................................................................... 370
      Example Four ............................................................................................. 371
      Example Five ............................................................................................. 372

Addressing Memory Above 4GB Boundary ......................................................... 372
  Introduction ....................................................................................................... 372
  Introduction ....................................................................................................... 373
  DAC Support Mandatory for All Initiators ......................................................... 373
  Memory Targets Must Support Wide BARs and DAC ...................................... 374
  Use of DAC Command Changes DEVSEL# and Master Abort Timing .............. 375
  Example 32-bit Transaction Changes DEVSEL# and Master Abort Timing ....... 375
  Example 64-bit Transaction Using DAC Command ........................................... 377
  DEVSEL# Must Not Be Asserted Too Soon ....................................................... 378

Add-In Card Trace Length .................................................................................. 379

Parity Generation and Checking ........................................................................ 382
Chapter 19: Parity Generation and Checking

General Discussion of Parity Generation ................................................................. 383
Parity Generation Is Mandatory ........................................................................... 383
In Data Phases of Writes and Split Completions .................................................. 384
Initiator Must Generate Correct Data Phase Parity ............................................. 384
Toggle Data and Parity When Target Inserts Wait State Pairs .......................... 384
In Reads, the Target Sources Data and Parity..................................................... 384
First Data Phase Data and Parity Can Be Delayed ............................................ 384
Subsequent Data Phase Data and Parity Never Delayed ................................... 384

General Discussion of Parity Checking................................................................. 384
Checking Required In Address and Attribute Phases ........................................... 384
Parity Checking Is Generally Required In Data Phases ..................................... 385
Target Data Parity Checking During Write or Split Completion ....................... 385
Initiator Parity Checking During Reads ............................................................... 385
Parity Not Checked During First Data Phase Wait States ............................... 385
Parity Checked One Clock After Each Subsequent Data Phase .................... 385

In Any Phase, Agent Driving AD Bus Supplies Parity ........................................ 385
As In PCI, Even Parity Is Used ............................................................................. 386
No Parity On Response Phase ............................................................................. 386

Address Phase Parity ......................................................................................... 387
Address Phase Parity Checking Required .......................................................... 387
When DAC Is Used, Check Both Packets ......................................................... 387
On Error, SERR# Required .............................................................................. 387
Error Detected Before Transaction Claimed ...................................................... 387
Error Detected After Transaction Claimed ....................................................... 388
Parity Error In Split Completion Address Phase .............................................. 388

Attribute Phase Parity ....................................................................................... 389
Means Sequence ID and/or Byte Count Corrupted ............................................ 389
When Attributes Corrupted and Split Response Issued .................................... 389

Data Phase Parity ............................................................................................... 389
Parity Always Covers Full Width of Data Bus .................................................... 389
Parity Driven One Clock After Information Presented ....................................... 390
Similar To PCI, But Different (due to registered bus) ......................................... 390
Parity-Related Initiator Responsibilities ............................................................. 390
Initiator May Be Requester, a Bridge, or a Completer ....................................... 390
Requester’s Parity-Related Responsibilities During a Write ............................. 390
Drive Data, Byte Enables and the Parity ............................................................ 391
Requester Checks PERR# .............................................................................. 391
Requester Actions When PERR# Asserted by Target ........................................ 391
Requester’s Actions On PERR# and Split Response ......................................... 391
Contents

Bridge’s Parity-Related Responsibilities During a Write ........................................... 391
Drive Data, Byte Enables and the Parity ................................................................. 392
Bridge Checks PERR# .......................................................................................... 392
Bridge Actions When PERR# Asserted by Target ............................................... 392
Initiator’s Parity-Related Responsibilities During a Split Completion ................. 394
  Who Initiates Split Completion Transactions? ...................................................... 394
  Initiator Drives Data, Byte Enables and Parity .................................................... 394
  Initiator Does Not Monitor PERR# .................................................................... 394
Initiator’s Parity-Related Responsibilities During a Read ..................................... 395
  General ................................................................................................................ 395
  First Read Data Item and Parity Can Be Delayed By Target ................................. 395
  Subsequent Data Phases of a Read ....................................................................... 395
  Parity Checking By Initiator During a Read ....................................................... 395
  Split Response Dummy Data Corrupted ............................................................. 396
Requester’s Parity-Related Responsibilities During a Read .................................. 396
  Read Error and Data Parity Error Recovery Enable Bit = 1 ............................... 397
  Read Error and Data Parity Error Recovery Enable Bit = 0 ............................... 397
Bridge’s Parity-Related Responsibilities During a Read ....................................... 397
  General ................................................................................................................ 397
  Bridge Handling of Parity Error in Immediate Read ............................................ 397
Parity-Related Target Responsibilities ................................................................... 398
  Target May Be Completer, Requester, or a Bridge ............................................. 398
Target’s Responsibilities During a Write .............................................................. 398
  When Completer Acts as the Target of the Write .............................................. 399
  When a Bridge Acts as the Target of the Write .................................................. 399
Target’s Responsibilities During a Split Completion .......................................... 399
  When Requester Acts as the Target of Split Completion ................................... 399
  When a Bridge Acts as the Target of Split Completion ..................................... 400
Target’s Responsibilities During a Read ............................................................... 400
  Data Parity Generation/Checking In PCI-X ......................................................... 401
  General Format of Timing Diagram Descriptions ............................................... 401
  Parity in Memory Write Block (or Alias) or Split Completion ............................ 401
  Parity in a Memory Write Transaction ............................................................... 406
  Parity in Memory Read Block (or Alias) Transaction ......................................... 406
  Parity in Memory Read Dword or IO Read Transaction ...................................... 411
  Parity in Configuration Read ............................................................................... 415
  Parity in IO Write Transaction ......................................................................... 416
  Parity in Configuration Write .......................................................................... 420
Part 3:
Device Configuration

Chapter 20: Configuration Transactions
Configuration Software Mechanism Same As PCI ................................................. 425
Configuration Transactions Can Only Flow Downstream ....................................... 426
Special Cycle Request Can Flow Upstream or Downstream .................................... 426
Type 0: Access Registers in Function on This Bus .................................................... 426
  Device Selection...................................................................................................... 426
  Bus/Device Auto-Updated by Each Type 0 Config Write ....................................... 430
    General ............................................................................................................. 430
    Host/PCIX Bridge Device Number Assignment ............................................... 430
Type 1: Access Registers in Function On Bus Further Out in Hierarchy .................... 430
  General Description.............................................................................................. 430
  Some Example Scenarios ...................................................................................... 433
    Example One .................................................................................................... 433
    Example Two .................................................................................................... 433
    Example Three .................................................................................................. 433
    Example Four .................................................................................................. 434
Type 0 Configuration Transactions ........................................................................... 435
  Action Taken On Master Abort .............................................................................. 435
  IDSELs Routed Over Upper AD Lines .................................................................... 436
  IDSEL Output Pins/Traces Allowed on Host/PCIX Bridge ...................................... 437
    Description........................................................................................................ 437
      Type 0 Access by a Tool .................................................................................. 438
      Resistive-Coupling Requires Address Pre-Drive ............................................. 438
      Bridge With IDSEL Pins Also Requires Address Pre-Drive ......................... 439
      Type 0 Configuration Read .............................................................................. 439
      Type 0 Configuration Write .............................................................................. 443
Type 1 Configuration Transactions ........................................................................... 447
  Although Not Necessary, Address Pre-Drive Used ............................................... 447
  Type 1 Configuration Access Address Format ...................................................... 447
  Target Bus’s Source Bridge Converts Type 1 To Type 0 Access ............................ 448
  Type 1 Transaction Examples ............................................................................... 448
Arbiter’s Treatment of Configuration Access ......................................................... 449
Generation of Special Cycle Under Software Control ............................................... 450
  General.............................................................................................................. 450
  PCIX-to-PCIX Bridges Pass Special Cycle Request To Target Bus ....................... 451
  Special Cycle By Device Other Than Host/PCIX Bridge ....................................... 452
# Chapter 21: Non-Bridge Configuration Registers

Detecting a PCI-X Capable Bridge.......................................................................................... 453
- Detecting Presence of PCI-X Capable Bridge/Bus................................................................. 453
- Detecting Width of Bridge’s Interfaces.................................................................................. 457
- Detecting Frequency Support of Bridge’s Interfaces.............................................................. 458
  - Frequency Support of Bridge’s Primary Interface.............................................................. 458
  - Frequency Support on Bridge’s Secondary Interface......................................................... 458
- Checking Current Frequency/Protocol of Secondary Bus....................................................... 459

Detecting Capabilities of Functions On Bus ............................................................................ 459
- Format of Non-Bridge Function’s PCI-X Capability Registers............................................... 459
- Detecting PCI Functions ........................................................................................................ 461
- Detecting Width of PCI-X Functions....................................................................................... 461
- Detecting Frequency Support of PCI-X Functions.................................................................... 462

Most PCI Configuration Registers Remain Unchanged ............................................................ 462

Some PCI Config Registers Affected by Protocol Mode............................................................. 463
- Some Register Default Values Affected.................................................................................... 463
- Implementation of Some Registers Affected............................................................................. 463
- Usage of Some Registers Affected............................................................................................ 464
- Base Address Registers (BARs).............................................................................................. 464
  - Memory Base Address Registers........................................................................................ 464
  - PCI and Memory BARs......................................................................................................... 464
  - Must Be Implemented As 64-bit Decoders......................................................................... 464
  - Definition of Prefetchable Memory..................................................................................... 464
  - Minimum Memory Range..................................................................................................... 465
  - Diminishes Overall Number of BARs.................................................................................. 465
- IO Base Address Registers....................................................................................................... 467
- Command Register Bits Affected by Protocol Mode............................................................... 467
  - Introduction......................................................................................................................... 467
  - Fast Back-to-Back Enable Bit............................................................................................. 468
  - Stepping Control Bit.............................................................................................................. 468
  - Memory Write and Invalidate Bit....................................................................................... 468
  - Bus Master Bit....................................................................................................................... 468
- Status Register Bits Affected by Protocol Mode.................................................................... 469
  - Latency Timer Default Affected by Protocol Mode.............................................................. 469
  - Cache Line Size Configuration Register.......................................................................... 470
  - Capabilities Pointer Register............................................................................................. 470

Function’s PCI-X Capability Register Set .................................................................................. 470
- Register Set Format.................................................................................................................. 470
## Contents

Background On Load Tuning ................................................................. 471
   Adjusting Requester’s Split Transaction Queue Size ......................... 471
      Problem ......................................................................................... 471
      Solution ......................................................................................... 471
   Adjusting Requester’s Memory Read Transaction Size ....................... 471
      Problem ......................................................................................... 471
      Solution ......................................................................................... 472
PCI-X Command Register ...................................................................... 472
   Max Outstanding Split Transaction Field .......................................... 473
   Max Memory Read Byte Count Field .................................................. 474
Enable Relaxed Ordering......................................................................... 474
   RO Command Bit Enables/Disables Use of RO Attribute Bit ............... 474
      General Description ........................................................................ 475
      Usage In Memory Read ..................................................................... 475
      Usage In Memory Write ..................................................................... 475
      After Disconnect, Be Consistent ......................................................... 476
      Restrictions On Use ........................................................................ 476
      For Additional Information ................................................................. 476
Data Parity Error Recover Enable .......................................................... 476
PCI-X Status Register ........................................................................... 476
   Bus, Device and Function Number Fields .......................................... 477
      Why They Are Necessary ................................................................. 477
      Function Number Is Hardwired .......................................................... 477
      Bus/Device Auto-Updated by Type 0 Configuration Write .................. 477
      Note For PCI-X Tool Designer ............................................................. 478
64-Bit Device Status Bit ......................................................................... 479
133MHz-Capable Status Bit ................................................................. 479
Split Completion Discarded (this is not a good thing!) ......................... 480
Unexpected Split Completion (this is also not a good thing!) ................. 480
Device Complexity ................................................................................ 481
   General .............................................................................................. 481
   What Use Is This Bit? ......................................................................... 481
   Definition of a Simple Device ............................................................... 481
   Rules Governing Behavior of a Simple Device ...................................... 481
   Designed Max Memory Read Byte Count .......................................... 484
   Designed Max Outstanding Split Transactions ..................................... 484
   Designed Max Cumulative Read Size .................................................. 484

---

**Chapter 22: Bridge Configuration Registers**

Discovering a PCIX-to-PCIX Bridge ......................................................... 487
Many Bridge PCI Configuration Registers Unchanged ........................... 487
Some Bridge PCI Configuration Registers Affected By Mode ................. 488
Contents

Command Register Affected ................................................................. 490
Status Register Affected ................................................................. 490
Capability Pointer Register Affected ......................................................... 490
Secondary Status Register Affected ......................................................... 490
  General ................................................................................................................. 490
  Fast Back-to-Back Capable Bit Affected ......................................................... 491
  Master Data Parity Error/Detected Parity Error Bits Affected ......................... 491
  DEVSEL# Timing Bit Field ............................................................................. 491
Cache Line Size Register Affected ........................................................ 491
Both Latency Timer Registers Affected ....................................................... 492
Base Address Registers (BARs) Affected .................................................... 492
Secondary Bus Number Register Affected .................................................. 493
  In PCI Mode, No Change ............................................................................. 493
  In PCI-X Mode, Secondary Requesters May Have Pending Split Accesses... 493
Hot-Plug Event May Cause Bus Renumbering ............................................. 494
RST# Causes Functions To Forget Bus/Device Numbers ............................... 494
Prefetchable Base/Limit Registers Affected ................................................ 495
Prefetchable Base/Limit Extension Registers Affected ....................................... 495
Bridge Control Register Affected ............................................................. 496

Bridge’s PCI-X Capability Register Set ..................................................... 497
  PCI-X Capability Register Set Format .......................................................... 497
PCI-X Secondary Status Register ............................................................... 498
  64-bit Device Status Bit .................................................................................. 499
  133MHz Capable Status Bit ........................................................................... 499
  Split Completion Discarded Bit ....................................................................... 499
  Unexpected Split Completion Bit..................................................................... 500
    PCI Master Addresses PCI-X Target On Secondary ...................................... 500
    Problem: Bridge’s Requester ID, But Bad Tag ............................................ 500
    Error Handling ............................................................................................. 500
  Bridge Secondary Interface Efficiency Status Bits ........................................ 501
    Background On Efficiency Status Bits ........................................................ 501
    Split Completion Overrun Bit ....................................................................... 501
    Split Request Delayed Bit ............................................................................ 502
Secondary Clock Frequency Bit Field ......................................................... 502
PCI-X Bridge (Primary Interface) Status Register ........................................ 503
  64-bit Device Status Bit .................................................................................. 503
  133MHz Capable Status Bit ........................................................................... 503
  Split Completion Discarded Bit ....................................................................... 504
  Unexpected Split Completion Bit..................................................................... 504
    PCI Master Addresses PCI-X Target On Primary .......................................... 504
    Problem: Bridge’s Requester ID, But Bad Tag ............................................ 505
    Error Handling ............................................................................................. 505
Contents

Bridge Primary Interface Efficiency Status Bits.................................................... 505
  Background On Efficiency Bits........................................................................ 505
  Split Completion Overrun Bit.......................................................................... 505
  Split Request Delayed Bit................................................................................. 506
Bus/Device/Function Number Fields ................................................................. 506
Split Transaction Control Registers................................................................. 506
  Basic Register Format............................................................................................... 506
  The Bridge’s Split Completion Data Buffers...................................................... 507
  Controlling Bridge’s Use of Its Split Completion Data Buffers...................... 507
    Capacity Value Indicates Size of Respective Data Buffer............................ 507
    Limit Value Controls Bridge’s Perception of Buffer Size.............................. 508
    Allocation Mode ................................................................................................ 508
    Virtual Buffer Space Mode ............................................................................... 509
    Flood Mode......................................................................................................... 509
    Regaining Control After Operating In Flood Mode........................................... 509
For More Information On Load Tuning... ............................................................ 511
Optional Bridge Registers............................................................................................ 511

Part 4:
Load Tuning

Chapter 23: Load Tuning Mechanisms

Introduction To Load Tuning....................................................................................... 515
Non-Bridge Function Tuning............................................................................................ 516
  Information Fields........................................................................................................ 516
  Adjustable Fields/Registers........................................................................................ 516
Bridge Tuning.................................................................................................................. 519
  Adjusting Usage of Split Completion Buffers.......................................................... 519
    Introduction................................................................................................................ 519
    Interpreting the Efficiency Bits .............................................................................. 519
    Additional Spec Comments ....................................................................................... 519
  Adjusting Bridge’s Timeslice Values........................................................................ 524
    Bridge Is Surrogate for Initiators On Both Sides of Bridge............................... 524
    Software Must Assign Bridge Two Timeslices..................................................... 525
    Load-Tuning Software Can Choose Timeslice Other Than 64d ......................... 525
Part 5:
PCI-X Bridges

Chapter 24: PCIX-to-PCIX Bridges
Performs Same Function As a PCI-to-PCI Bridge............................................................ 532
Support For DAC Command ............................................................................................... 532
    Downstream Movement of DAC Optional For PCI Bridge ........................................ 532
    PCI-X Bridge Must Support Downstream DAC Movement .................................... 533
Bus Width ................................................................................................................................ 534
Memory Writes Crossing Bridge Are Always Posted ..................................................... 534
Other Transactions Crossing Bridge Are Always Split .................................................. 534
How the Bridge Claims Split Completions ...................................................................... 534
When Bridge Can Use Retry or Disconnect At Next ADB............................................. 535
    When Bridge Can Issue a Retry ............................................................................. 535
    When Bridge Can Issue a Disconnect At Next ADB .............................................. 536
Interfaces Can Be In Different Modes/Speeds ................................................................. 536
Translating PCI To PCI-X..................................................................................................... 536
    General ..................................................................................................................... 536
    Writes Are Posted ................................................................................................... 537
    All Others Treated As PCI Delayed Transaction ................................................... 537
    On PCI Side, Follows PCI Ordering Rules ............................................................... 538
    Translating the Transaction Type ........................................................................... 538
    Creating the Attributes .......................................................................................... 541
        Attribute Bits ........................................................................................................ 541
        Requester ID ...................................................................................................... 542
        Transaction Tag ................................................................................................. 542
        Byte Transfer Count ......................................................................................... 542
        Target May Treat As Immediate or Split Transaction ...................................... 543
Translating PCI-X to PCI...................................................................................................... 543
    General ..................................................................................................................... 543
    Writes Are Posted ................................................................................................... 544
    All Others Treated As Split Transactions .................................................................. 544
    Translating the Transaction Type ........................................................................... 544
    PCI Target May Treat As Immediate or Delayed Transaction .............................. 547
    Bridge Creation of a Split Completion ..................................................................... 548
    Effect of Relaxed Ordering Attribute Bit .................................................................. 549
Error Handling.................................................................................................................... 549
    Bridge Error Class SCMs ....................................................................................... 549
    Error Handling Defined by Mode of Originating Interface .................................. 549
    Error Handling When Originating Bus In PCI-X Mode ......................................... 550
Contents

Scenario 1: Data Parity Error In Immediate Read On Destination Bus ............ 550
Scenario 2: Data Parity Error On Split Write ............................................... 551
  General ........................................................................................................ 551
  Scenario 2a ..................................................................................................... 551
  Scenario 2b ..................................................................................................... 552
  Scenario 2c ..................................................................................................... 553
  Scenario 2d ..................................................................................................... 554
Scenario 3: Data Parity Error On Split Completion ........................................... 555
  Originating Bus Parity Error On Split Completion Read Data ...................... 555
  Originating Bus Parity Error On SCM ............................................................ 555
  Destination Bus Parity Error On Data or SCM ............................................ 556
  Bridge Forwards SCMs Quietly (without getting involved) ......................... 556
Scenario 4: Data Parity Error On Posted Memory Write .................................. 556
Scenario 5: Master Abort On Destination Bus .................................................. 557
  General ........................................................................................................ 557
  Master Abort When Forwarding Split Request ............................................. 557
  Master Abort On Posted Memory Write ....................................................... 558
  Master Abort On Split Completion ............................................................... 559
Scenario 6: Target Abort On Destination Bus .................................................... 559
  Bridge Signals Target Abort ........................................................................... 559
  Bridge Receives a Target Abort on Split Transaction .................................. 559
  Bridge Receives Target Abort on Memory Write ........................................ 560
  Bridge Receives Target Abort on Split Completion ..................................... 560
Error Handling When Originating Bus In PCI Mode ....................................... 560
  Background .................................................................................................. 560
  Bridge Handles Errors Same as PCI Bridge Unless SCM Error .................... 561
  SCM Error On a Split Read ........................................................................... 561
  SCM On Split Write ...................................................................................... 561
  Corrupted Split Completion ........................................................................ 562
Buffer Size ....................................................................................................... 562
  Posted Memory Write Buffer Size .............................................................. 562
  Split Completion Data Buffer Size ............................................................. 563
  Buffer Space For Memory Read Data .......................................................... 564
    General ....................................................................................................... 564
    The Problem .............................................................................................. 564
    When Limit = Capacity, This Is Not a Problem ........................................ 564
When a Request Is Bigger Than Bridge Buffer ................................................ 564
Application Bridge .......................................................................................... 564
Bridge Acceptance Rules ................................................................................ 567
Ordering and Passing Rules

Same Rules As PCI Except

Relaxed Ordering Effect On Transaction Ordering

Relaxed Ordering Effects On Memory Reads

Relaxed Ordering Effects On Memory Writes

Split Transaction Effects On Transaction Ordering

General

Same Sequence Split Completions Must Remain In Order

Relaxed Ordering Can Affect Split Completion Delivery

The Rules

Decomposing Split Transactions (sounds morbid!)

Chapter 25: Locked Transaction Series

Definition of Downstream and Upstream

Basics

Only Host/PCIX Bridge Originates Downstream Locked Series

PCI-X Bridges Only Pass Locked Series Downstream

Only EISA Bridge Originates Upstream Locked Traffic

Application Bridge May or May Not Support Locking

EISA Bridge Supports LOCK# As Input, Not As Output

Non-Bridge Devices Ignore LOCK#

Sequence of Events

Split Completion Error Message Terminates Lock

Upstream Bridge (Initiating Bridge) Rules

Downstream Bridge (Target Bridge) Rules

Arbitration

Starting Locked Transaction Series

Retry/Target Abort/Master Abort Cancels Lock

First Access of Series Receives Retry

General

If Target Is a PCI-to-PCI Bridge

If the Target Is a PCIX-to-PCIX Bridge

First Access of Series Receives Target or Master Abort

First Transaction Has Immediate Completion

First Transaction Receives Split Response

Continuing Locked Transaction Series

Attempted Access To Bridge By Device Other Than Owner

Last Transaction In Locked Series

If Last Transaction Receives Immediate Completion

If Last Transaction Receives Split Response

The Unlocking of a Target Bridge

When Bridge Starts 2nd Series Immediately After 1st
## Contents

Retry Issued To Owner Not a Problem ................................................................. 599
DAC Command Lock Timing .................................................................................. 600
  In the First Access .............................................................................................. 600
  In Subsequent Accesses of Locked Series .......................................................... 600

---

### Part 6:
Error Detection and Handling

#### Chapter 26: Error Detection and Handling

**Handling of a Target Abort** .................................................................................. 603
  Introduction ........................................................................................................... 603
  Requester Receives Target Abort ................................................................. 604
  Completer Receives Target Abort on Split Completion ................................. 606
    When Split Completion Target Abort Is Permissible ...................................... 606
    Discard of Completion for a Write or Non-Prefetchable Read ...................... 607
    Discard of Completion for IO, Config, or Prefetchable Memory Read ........ 608

**Handling of a Master Abort** ............................................................................... 609
  Introduction ........................................................................................................... 609
  Requester Handling of Master Abort ................................................................. 609
    If Host/PCI Bridge Type 0 Config Transaction .............................................. 609
    If Not Host/PCIIX Bridge Type 0 Config Transaction ................................. 610
    If Requester Capable of Generating Interrupts ............................................. 610
    If Requester Can’t Generate Interrupts ......................................................... 610
  Completer Handling of Master Abort on a Split Completion ........................... 610
    General ............................................................................................................. 610
    Discard of Completion for a Write or Non-Prefetchable Read ..................... 611
    Discard of Completion of IO, Config, or Prefetchable Memory Read .......... 611

**Target Handling of Address or Attribute Phase Parity Error** ........................... 612

**Data Phase Parity** ............................................................................................. 612
  PCI Chipset Typically Murders System On Data Parity Error ........................ 612
  In PCI-X, Chipset Doesn’t Monitor PERR# .................................................... 613
  General .............................................................................................................. 613
  Requester’s Data Parity Error Recovery Enable Bit ........................................ 613
    Basic Description .......................................................................................... 613
    Set By OS or Driver’s Initialization Code At Startup Time ........................... 614
    If Not Set, PCI Compatibility Dictates Murdering System ......................... 614
    Recovery Must Be Performed Under Software Guidance ......................... 615
    Driver Reports Error To OS ......................................................................... 615
    OS Instructs Driver On Action(s) To Take .................................................. 615
    OS Vendor Specifies Who Clears Parity Error Status Bits .......................... 615
Requester Handling of PERR# with Split Response ................................................... 616
Data Error Received With Split Response On Read ............................................. 616
Data Error Received With Split Response on Dword Write ............................... 618
Requester/Completer Handling of Data Error During Split Completion .............. 620

Split Read Errors .................................................................................................................... 620
General........................................................................................................................ ....... 620
Read Data Returned in One or More Split Completions............................................ 621
Internal Completer Error Can Occur At Any Time .................................................... 622
Receipt of SCM Terminates Read Sequence................................................................. 622
Effect of SCM On a Bridge.............................................................................................. 623
Effect of SCM On the Requester .................................................................................... 623

Requester Handling of Unexpected Split Completion................................................... 626

Requester Handling of Split Completion Error Messages ............................................ 627
Handling Completer Byte Count Out-of-Range Error ............................................... 627
Memory Writes Never Split, So SCMs Don’t Apply .................................................... 627
Read Immediate Treated Same As Write At Device Boundary ............................... 627
Split Read Straddling Device Boundary Causes SCM ............................................ 627
Read Burst Straddling Device Boundary Considered Rare Case ......................... 628
Handling Completer Split Write Data Parity Error .................................................... 629
Handling Completer Device-Specific Error .................................................................. 629
Handling Master Abort On Other Side of a Bridge .................................................... 629
Error Description ...................................................................................................... 629
Effect of Master Abort on Bridge.................................................................................. 630
Bridge Interrupt Acknowledge Master Aborts ......................................................... 630
Bridge Special Cycle Master Aborts ............................................................................ 630
Bridge IO Read Master Aborts ..................................................................................... 630
Bridge IO Write Master Aborts ................................................................................... 632
Bridge Configuration Read Master Aborts ................................................................. 632
Bridge Configuration Write Master Aborts ................................................................. 632
Bridge Memory Read Dword Master Aborts .............................................................. 632
Bridge Memory Write Master Aborts ......................................................................... 632
Bridge Memory Read Ends In Master Abort ............................................................... 633
Bridge Split Completion Ends In Master Abort ........................................................ 633
Effect of Master Abort SCM on Requester ................................................................. 635
Requester Interrupt Acknowledge Gets Master Abort SCM .................................... 635
Requester Special Cycle Gets Master Abort SCM ....................................................... 635
Requester IO Read or IO Write Gets Master Abort SCM ............................................ 636
Requester Configuration Read Gets Master Abort SCM ............................................. 636
Requester Configuration Write Gets Master Abort SCM ............................................ 636
Requester Memory Read Dword Gets Master Abort SCM ......................................... 637
Requester Memory Write Gets Master Abort SCM ..................................................... 637
Requester Memory Read Gets Master Abort SCM ...................................................... 637
Contents

Requester Split Completion Gets Master Abort SCM ........................................... 638
Handling Target Abort on Other Side of Bridge ................................................ 638
Error Description .................................................................................................. 638
Effect of Target Abort on Bridge........................................................................ 638
Bridge Interrupt Acknowledge Target Aborts .................................................. 639
Bridge Special Cycle Target Aborts ................................................................ 639
Bridge IO Read Target Aborts ........................................................................... 639
Bridge IO Write Target Aborts .......................................................................... 640
Bridge Configuration Read Target Aborts......................................................... 640
Bridge Configuration Write Target Aborts....................................................... 640
Bridge Memory Read Dword Target Aborts ....................................................... 640
Bridge Memory Write Target Aborts ................................................................. 640
Bridge Memory Read Block Ends In Target Abort .......................................... 641
Bridge Split Completion Ends In Target Abort ............................................... 641
Effect of Target Abort SCM on Requester ....................................................... 642
Requester Interrupt Acknowledge Gets Target Abort SCM ............................. 642
Requester Special Cycle Gets Target Abort SCM ............................................. 642
Requester IO Read or IO Write Gets Target Abort SCM ................................. 642
Requester Configuration Read Gets Target Abort SCM .................................. 643
Requester Configuration Write Gets Target Abort SCM .................................. 643
Requester Memory Read Dword Gets Target Abort SCM ................................ 643
Requester Memory Write Gets Target Abort SCM .......................................... 644
Requester Memory Read Gets Target Abort SCM ........................................... 644
Requester Split Completion Gets Target Abort SCM ....................................... 644
Handling Split Write Data Parity Error On Other Side of Bridge ...................... 644
Error Description .................................................................................................. 644
Bridge May Deliberately Forward a Bad Split Write ....................................... 644
Or Split Write May Become Corrupted When Bridge Forwards It ................... 645
Effect of Split Write Data Parity Error On Bridge ............................................. 645
Effect of Split Write Data Parity Error On Requester ....................................... 645
Usage of SERR# ........................................................................................................ 646

Part 7:
Electrical Issues

Chapter 27: Electrical Issues
Introduction ........................................................................................................... 651
LVS Bus ................................................................................................................. 651
Attention To Detail ............................................................................................... 651
Most Parameters Tighter Than PCI ................................................................. 652
Maximum Number of Loads and Connectors On Bus ................................... 652
This Chapter

This chapter describes the areas of the PCI bus protocol that the architects of the PCI-X bus protocol identified as needing improvement.

The Next Chapter

The next chapter introduces the primary improvements in PCI-X that targeted deficiencies in the PCI bus protocol. This includes: higher bus speeds, eliminated of Wait States, block-oriented transfers, Latency Timer optimization, transfer size specification, Split Transactions, load tuning, Data Phase parity error recovery, bus width indication, MSI capability, Power Management capability, elimination of snoops, widening of memory decoders, and the elimination of stepping and Fast Back-to-Back transactions.

Wait States Yield Poor Performance

A PCI transaction consists of the Address Phase and one or more Data Phases. In each Data Phase, either the Initiator or the target can delay the transfer of the current dword (or, during 64-bit data transfers, qword). According to the rules first established in the PCI 2.1 specification, the initiator can keep its IRDY# signal deasserted for up to seven clocks in each Data Phase. The target can keep TRDY# deasserted for up to 15 clocks from the assertion of FRAME# in the first Data Phase, and for up to seven clocks in each subsequent Data Phase of a burst transaction.

In other words, the PCI spec permits the initiator and target to make poor use of the bus. Obviously, this type of behavior would yield poor performance for the initiator performing the transaction as well as for other bus masters awaiting bus ownership to initiate transfers.
Relatively Slow Clock Speed

PCI buses may be designed to operate in either the 0-33.33MHz range or the 33.33-66.66MHz range. Data items (i.e., dwords or qwords) are transferred on each rising-edge of the clock on which both IRDY# and TRDY# are sampled asserted. This yields a maximum transfer rate of 33.33M transfers per second on a low-speed bus and 66.66M transfers per second on a high speed bus. The best-case scenario is when the bus is running at 66.66MHz, neither the initiator nor the target inserts any Wait States in the transaction’s Data Phases, and both parties are 64-bit devices. This would permit a maximum transfer rate of 66.66M 8-byte transfers per second, or 533.28MBytes per second.

Transfer Size Unknown

When a PCI master initiates a burst transaction (i.e., a memory transaction), it does not indicate how much data it intends to transfer. The PCI memory access commands are:

- **Memory Read** (MR). Indicates that the master intends to read an unspecified amount of data from memory starting at the indicated dword (or qword if it is performing 64-bit transfers). By using this command, the master is indicating to any bridge that the read may have to traverse either that:
  - it has specific knowledge that the memory region is non-Prefetchable memory, or
  - it doesn’t know whether it’s prefetchable or not.
- **Memory Read Line** (MRL). Indicates the master intends to read an unspecified amount of data from memory starting at the initial dword (or qword) and that it knows that the area of memory from the start dword (or qword) address up to the end of the current line is Prefetchable memory.
- **Memory Read Multiple Line** (MRM). Indicates the master intends to read an unspecified amount of data from memory starting at the initial dword (or qword) and that it knows that the area of memory from the start dword (or qword) address up to the end of the line after the current line is Prefetchable memory.
- **Memory Write** (MW). Indicates the master intends to write an unspecified amount of data to memory starting at the start dword (or qword) address specified in the Address Phase.
- **Memory Write and Invalidate** (MWI). Indicates that the master has issued a start address aligned on a line boundary and that it intends to write an unspecified number of full lines to memory.
Chapter 1: PCI Needed Improvement

In each of these cases the master issues a dword- or qword-aligned start address and does not specify how much data it intends to read or write. This presents buffer management problems to:

- any bridges that the transaction must traverse.
- On a read from memory, a bridge does not know how much data will ultimately be read by the originating master. It therefore doesn’t know how much data to read from the target when it passes the memory read transaction to the target bus and doesn’t know how much buffer space to allocate to hold the returning read data.
- On a write to memory, the bridge doesn’t know how much buffer space to allocate in its posted memory write buffer to absorb the write data into.
- the target that is being read from or written to.
- On a memory read, the target doesn’t know how much data will ultimately be read. It therefore doesn’t know how far ahead it can safely read from its internal memory.
- On a memory write, the target doesn’t know how much buffer space to allocate in its inbound posted memory write buffer to absorb the write data into.

PCI Delayed Transactions Are Inefficient

If the target of a PCI transaction cannot transfer the first data item within 16 clocks from the assertion of FRAME#, it memorizes the transaction and issues a Retry to the Initiator. In other words, it treats it as PCI Delayed Transaction.

Initiator Retries Use Up Valuable Bus Time

After receiving a Retry, the initiator is obliged to repeatedly re-arbitrate for the bus. The target will continue to respond to each repeat of the transaction request with a Retry until it is ready to start transferring data within 16 clocks from the initiation of the transaction.

Initiator Doesn’t Supply Transfer Count

The PCI target memorizes all of the transaction information that the initiator supplies it with:
PCI-X System Architecture

- The start dword (or qword) address.
- The command.
- The Byte Enable settings issued in the first Data Phase
- If it’s a an IO or a Config write transaction, the write data supplied in the first Data Phase.

This is referred to as the Delayed Read or Write Request (DRR or DWR). If the request is a memory read (MR, MRL, or MRM), the initiator has not specified the amount of data to be read. As mentioned earlier, this creates buffer management and read-ahead problems for the target or any bridge in the path to the target.

Delayed Completion

Once the target has obtained the requested read data or has delivered the write data, the request has been transformed from a Delayed Request into a Delayed Completion. The target must then await the originating master’s next repeat of the transaction so it can give the completion to it.

Initiator’s Transaction ID Is Sketchy at Best

The only information that the target can use to identify the master that is repeating a previously-memorized request is the start address, command, Byte Enables issued in the first Data Phase, and the write data (if it’s a write). Since it is possible for another master to mimic the request issued earlier by another master, the target may give the completion to the wrong master.

Snoops Hurt Performance

Refer to Figure 1-1 on page 18.

Host/PCIX Bridge Knows AGP’s Area of Memory Is Non-Cacheable

The Host/PCIX bridge typically contains the main memory controller. It is also the communication path between PCI masters and main memory. The AGP 2.0 spec dictates that the region of main memory allocated to the AGP graphics adapter must be designated as non-cacheable memory. This means that the processors do not cache copies of memory lines from this region. There is therefore
Chapter 1: PCI Needed Improvement

no reason for the Host/PCIX bridge to generate snoop transactions on the processor bus whenever the AGP adapter accesses its designated area of memory.

Memory Used By PCI Masters Mayor May Not Be Cached

Other than the region of main memory assigned to the AGP adapter, the Host/PCIX bridge does not know what areas of main memory are cached by the processor(s). This means that it must generate a snoop transaction on the processor bus whenever a PCI master attempts to access main memory.

Snoops Slow Down PCI Accesses To Main Memory

When a PCI adapter starts an access to main memory, the Host/PCIX bridge acts as the target of the PCI transaction. Because the master may be addressing a stale line in system memory, the Host/PCIX bridge cannot allow the PCI master to start transferring data with memory until the results of the snoop have been presented to the Host/PCIX bridge by the processor(s). While awaiting the snoop result, the Host/PCIX bridge may insert Wait States into the first Data Phase or may issue a Retry to the PCI master. This delay in the transfer of the first data item certainly impacts the performance of the PCI master.

Snoop Traffic On Processor Bus Can Hurt Processor(s)

Each time that the Host/PCIX bridge generates a snoop transaction on the processor bus, the processor bus is a little less available to the processor(s). If PCI bus masters access main memory frequently, this can have an adverse effect on the performance of the processor.

Main Memory Less Available To Processors

In addition, during periods of time when PCI masters are generating a lot of traffic to main memory, the memory bus is less available for the processors to use.
Figure 1-1: Block Diagram of Example System
PCI-X Improves On PCI

The Previous Chapter

The previous chapter described the areas of the PCI bus protocol that the architects of the PCI-X bus protocol identified as needing improvement.

This Chapter

This chapter introduces the primary improvements in PCI-X that targeted deficiencies in the PCI bus protocol. This includes: higher bus speeds, eliminated of Wait States, block-oriented transfers, Latency Timer optimization, transfer size specification, Split Transactions, load-tuning, Data Phase parity error recovery, bus width indication, MSI capability, Power Management capability, elimination of snoops, widening of memory decoders, and the elimination of stepping and Fast Back-to-Back transactions.

The Next Chapter

The PCI-X bus is backward-compatible with PCI cards. If the Source Bridge discovers that one or more of the add-in cards on a PCI-X bus are PCI rather than PCI-X cards, it instructs all of the PCI-X capable devices on that bus to use the PCI protocol and also tells them what speed the bus will run at. The next chapter describes how the programmer can determine that a bus is a PCI-X bus and can then determine if any of the devices on the bus are PCI rather than PCI-X devices. Upon detection of a PCI device installed in a PCI-X connector, the programmer almost certainly would inform the end-user of the performance degradation that will result if the card remains installed in the system.

PCI-X Is Backward Compatible With PCI

One of the primary goals was that PCI-X be backward-compatible with PCI. To achieve that goal, the PCI-X spec dictates the following:
PCI-X System Architecture

- **PCI-X uses the traditional PCI connector with one pin definition changed.** Pin B38 is a ground plane pin on a conventional PCI card, but is defined as the PCIXCAP pin on a PCI-X capable card. The PCIXCAP signal line is pulled up to a logic one on the system board. A complete description of this pin can be found in “PCIXCAP Indicates Protocol/Frequency Required” on page 58.

- **PCI-X capable devices must be capable of utilizing either the PCI-X or PCI bus protocols.** If any of the add-in cards installed ground the PCIXCAP signal, all devices on that bus must use the PCI protocol and the bus must operate at a frequency that ensures the proper operation of the slowest PCI card installed. As described in “Supplying PCI-X Devices With Protocol and Speed” on page 64, the Source Bridge for a PCI-X bus provides an initialization pattern to the PCI-X capable devices on that bus upon the removal of the reset signal. Based on the pattern provided, a PCI-X device automatically selects the appropriate protocol to utilize, as well as the frequency range its PLLs (phase-locked loops) must lock to.

**PCI-X Is More System-Centric PCI**

The PCI-X spec has a more system-centric focus than the PCI spec. The PCI spec tends to focus more on the design of the PCI function, its configuration and performance. PCI-X on the other hand has as its primary focus the performance of the system as a whole rather than the performance of each function. As examples:

- The spec limits the maximum transfer size of individual transaction requests that may be issued by a master.
- The spec defines a default timeslice that is strongly recommended for PCI-X masters.
- Special status bits are implemented that permit software to detect how well a bridge is handling acceptance of transactions that cross the bridge in either direction.
- After testing the status bits referenced in the previous bullet item, software can adjust how a bridge uses its internal buffer space. This permits software to smooth the flow of transactions across the bridge in either direction.
- Software can clamp the maximum size of memory read burst transactions that may be issued by a master.
- Software can clamp the maximum number of Split Transactions that a master may have outstanding at a given moment in time.
Chapter 2: PCI-X Improves On PCI

Higher Clock Speeds Possible

While the highest speed a PCI bus may operate at is 66.66MHz, a PCI-X bus may run at anywhere from 50MHz-133.33MHz. As in PCI, a data item can be transferred on each rising-edge of the clock and significantly higher throughput may be obtained over a bus operating in PCI-X mode at a higher clock rate.

Wait States Eliminated

While in PCI both the master and the target can legally insert Wait States during the transfer of individual data items, this is not permitted in PCI-X.

- The master is never permitted to insert Wait States to delay the transfer of any data item.
- The target can insert a limited number of Wait States to delay the transfer of the first data item (because you basically surprised it with the transfer request), but must then keep TRDY# asserted in each subsequent Data Phase.

For a more detailed discussion of the latency rules, refer to Chapter 12, entitled "Latency Rules,” on page 189.

Data Transferred in Blocks

In PCI, Data Transferred as a Series of Data Items

In a PCI transaction, data is transferred as a series of individual dwords (or qwords) and both the master and the target can delay the transfer of each data item by deasserting their respective ready signals (IRDY# and TRDY#). In addition, both the master and the target can disconnect (i.e., terminate) the transaction on any dword (or qword) boundary. The master can disconnect the transaction by simply indicating to the target that the current data item is the final one of the transfer (by asserting IRDY# and deasserting FRAME#). The target can issue a disconnect to the master by asserting the STOP# signal in any Data Phase.
In PCI-X, Data Transferred as a Series of Blocks

In PCI-X, data is transferred as a series of blocks, each of which consists of 128 bytes (it should be noted, however, that a PCI-X master can transfer less than a block if it so chooses). A block boundary is one that is divisible by 128 and is referred to as an Allowable Disconnect Boundary (ADB).

Master Cannot Delay Transfer of First Data Block

When starting a transaction, the master must be immediately ready to transfer at least the first data block. If it’s not ready to do so, it must not arbitrate for bus ownership.

- On a write, the master must therefore have the first data block buffered up and ready to output at full-speed.
- On a read, the master must have at least one block of buffer space reserved to receive the first data block at full speed.

Target Can Only Delay Transfer of First Block

The target is permitted to delay the transfer of the first data block (by keeping TRDY# deasserted for a limited number of Wait States), but must either:

- Transfer the remaining data blocks at full-speed (i.e., it is not permitted to deasserted TRDY# to delay a data transfer), or
- Assert STOP# to force the master to disconnect on a block boundary.

Master and Target Can Only Disconnect On Block Boundaries

If a transfer consists of more than one data block, the master and the target are only permitted to disconnect the transaction on block boundaries (due to current buffer availability conditions).

But...There are Two Exceptions

1. The target can issue a Single Data Phase Disconnect (see “Single Data Phase Disconnect” on page 108 for more detail) to the master when transferring the first data item. A target that doesn’t support burst transactions uses this method to force the initiator to disconnect the transaction upon completion of the first Data Phase.
2. If the transfer size specified by the master indicates a transfer of less than a block, then the transaction will terminate when the transfer count is exhausted.
Disconnecting on Block Boundaries

Master Disconnection of a Transfer

Assume that a master ultimately intends to write 1024 bytes of data (eight blocks of data) to memory starting on an address divisible by 128 (but note that in PCI-X the start address can be any byte-specific address). It currently has several of the initial data blocks ready to write, but is still accumulating the remainder of the data. Since it has several blocks ready to write, it arbitrates for the bus and initiates a burst memory write transaction stating its intention to write a total of 1024 bytes of data. It writes the first three blocks of data and then begins writing the fourth block. As it approaches the boundary (i.e., ADB) between the fourth and fifth blocks, it signals to the target that it wishes to disconnect the transaction on the upcoming block boundary before the transfer count is exhausted. The target will honor the disconnection and the current burst memory write transaction will terminate on the ADB between the fourth and fifth blocks.

When the master has one or more additional blocks of the remaining data buffered up and ready to write at full-speed, it rearbitrates for the bus and resumes the burst memory write at the start of the next block.

Target Disconnection of a Transfer

Using the previous example, the target may have sufficient buffer space to accept some but not all of the data that the master intends to write. In this case, it accepts the data at full-speed into its buffer until it only has one remaining block (128 bytes) of buffer space remaining. It then asserts STOP# during the transfer of this block, thereby commanding the master to disconnect the transaction on the upcoming block boundary.

The master is then forced to rearbitrate for the bus and will then resume the burst memory write at the point of disconnection.

Latency Timer Usage

As in PCI, each bus master is assigned a timeslice. If it is preempted (i.e., its GNT# signal is deasserted by the arbiter) before it has transferred all of the data it intended to transfer, it may continue the transaction until its timeslice has expired and must then disconnect the transaction on the next block boundary.
The Previous Chapter

The previous chapter introduced the primary improvements in PCI-X that targeted deficiencies in the PCI bus protocol. This includes: higher bus speeds, eliminated of Wait States, block-oriented transfers, Latency Timer optimization, transfer size specification, Split Transactions, load tuning, Data Phase parity error recovery, bus width indication, MSI capability, Power Management capability, elimination of snoops, widening of memory decoders, and the elimination of stepping and Fast Back-to-Back transactions.

This Chapter

The PCI-X bus is backward-compatible with PCI cards. If the Source Bridge discovers that one or more of the add-in cards on a PCI-X bus are PCI rather than PCI-X cards, it instructs all of the PCI-X capable devices on that bus to use the PCI protocol and also tells them what speed the bus will run at. This chapter describes how the programmer can determine that a bus is a PCI-X bus and can then determine if any of the devices on the bus are PCI rather than PCI-X devices. Upon detection of a PCI device installed in a PCI-X connector, the programmer almost certainly would inform the end-user of the performance degradation that will result if the card remains installed in the system.

The Next Chapter

As introduced in this chapter (see “Bus Protocol/Speed = Lowest Common Denominator” on page 48), the Source Bridge for a PCI-X bus interrogates two signals at startup time to determine the protocol and bus speed to be used on its secondary bus during this power-up session. The protocol and speed is selected to ensure that the least-capable device functions correctly. The next chapter provides a detailed description of this process.
PCI-X System Architecture

Bus Protocol/Speed = Lowest Common Denominator

A Source Bridge is the bridge which originates a PCI or a PCI-X bus. In Figure 3-1 on page 51 and Figure 3-2 on page 53, the Host/PCIX bridge is the Source Bridge for the PCI-X bus directly on the other side of it. In Figure 3-3 on page 54, the left-hand Host/PCIX bridge is the Source Bridge for the PCI-X bus, while the right-hand Host/PCIX bridge is the Source Bridge for the PCI bus.

At startup time (during the assertion of reset), the Source Bridge samples two signals to determine the protocol and bus speed supported by the least-capable device installed on the bus. The bridge then tells the PCI-X capable devices on its secondary bus what protocol and bus speed will be used during this power-up session. In other words, the bus protocol and bus clock chosen will ensure that the least-capable device functions correctly. A detailed discussion of this process can be found in Chapter 4, entitled "Device Types and Bus Initialization," on page 55.

It should be obvious that the benefits yielded by the more efficient PCI-X protocol and its higher bus speeds are lost if the end-user is permitted to install a conventional PCI card on a PCI-X bus.

Discovering a PCI-X Bus

The programmer discovers a PCI-X bus in the following manner:

1. Using the Class Code configuration register, discover a North Bridge (i.e., a Host/PCI bridge) or a PCI-to-PCI bridge.
2. Determine if the Capabilities List status bit in the bridge’s PCI configuration Status register is hardwired to one. If it is, go to step 3; if it isn’t then it is not a PCI-X bridge.
3. Traverse the bridge’s New Capabilities register sets to determine if it implements the PCI-X capability configuration register set. If it does, then the bus on the bridge’s secondary side is a PCI-X bus.

A detailed description of the PCI configuration registers can be found in the MindShare book entitled *PCI System Architecture, Fourth Edition* (published by Addison-Wesley). A detailed description of the PCI-X configuration registers can be found in “Function’s PCI-X Capability Register Set” on page 470 and in “Bridge’s PCI-X Capability Register Set” on page 497.
Discovering PCI Devices On a PCI-X Bus

As mentioned earlier in this chapter, for performance reasons it’s advisable to detect PCI cards on a PCI-X bus and then prompt the end-user to either remove them from the machine or to move the card to a bus intended only for PCI devices.

A PCI function can be detected by checking for the absence of the PCI-X configuration registers:

1. Determine if the Capabilities List status bit in the function’s PCI configuration Status register is hardwired to one. If it is, go to step 2; if it isn’t then it is not a PCI-X function.
2. Traverse the function’s New Capabilities register sets to determine if it implements the PCI-X capability configuration register set. If it doesn’t, then the function is not a PCI-X function.

Some Example Systems

System With No Connectors On PCI-X Bus

If the system pictured in Figure 3-1 on page 51 had no connectors on the PCI-X bus, the PCI-X bus would operate in PCI-X protocol mode and at a speed that ensures the proper operation of the least-capable PCI-X device on the bus.

Single Bus System With Connectors

Figure 3-1 on page 51 illustrates a system with a single PCI-X bus with two connectors. There are a number of possible scenarios:

- Both cards may be PCI-X capable. In this case, the PCI-X bus will operate in PCI-X protocol mode and at a speed that ensures the proper operation of the least-capable PCI-X device.
- One card may be PCI-X capable while the other is a PCI device, or both may be PCI devices. In this case, the PCI-X bus will operate as a PCI bus at a lower speed to ensure that the PCI card functions correctly.
Either card may incorporate a PCIX-to-PCIX bridge. In this case, the bridge’s primary side will operate in PCI-X mode, but there are two possible operational scenarios for its secondary side:

- All devices on the bridge’s secondary bus may be embedded PCI-X capable devices. In this case, the secondary bus operates in PCI-X mode at a speed that ensures proper operation of the least-capable PCI-X device on the secondary bus.
- The bus on the bridge’s secondary side may include one or more connectors. The secondary bus will then operate in the protocol/speed mode that ensures proper operation of the least-capable device on the bus.

Either card may incorporate a PCI-to-PCI bridge. In this case, both buses will operate in PCI mode, resulting in severe performance degradation on the PCI-X capable bus (because it’s operating in PCI mode).
Chapter 3: Lowest Common Denominator Defines Mode

Figure 3-1: Simple System Block Diagram
Device Types and Bus Initialization

The Previous Chapter
The PCI-X bus is backward-compatible with PCI cards. If the Source Bridge discovers that one or more of the add-in cards on a PCI-X bus are PCI rather than PCI-X cards, it instructs all of the PCI-X capable devices on that bus to use the PCI protocol and also tells them what speed the bus will run at. The previous chapter described how the programmer can determine that a bus is a PCI-X bus and can then determine if any of the devices on the bus are PCI rather than PCI-X devices. Upon detection of a PCI device installed in a PCI-X connector, the programmer almost certainly would inform the end-user of the performance degradation that will result if the card remains installed in the system.

This Chapter
As introduced in the previous chapter (see “Bus Protocol/Speed = Lowest Common Denominator” on page 48), the Source Bridge for a PCI-X bus interrogates two signals at startup time to determine the protocol and bus speed to be used on its secondary bus during this power-up session. The protocol and speed is selected to ensure that the least-capable device functions correctly. This chapter provides a detailed description of this process.

The Next Chapter
The next chapter describes the registered nature of the PCI-X bus.
All Devices Support 33MHz PCI

General

All PCI and PCI-X devices support operation on a 33MHz PCI bus, while it’s optional whether a PCI or a 66MHz PCI-X device supports operation on a 66MHz PCI bus. Refer to Figure 4-1 on page 57. A device indicates that it will operate correctly on a high-speed PCI bus by not grounding the M66EN pin (pin B49).

133MHz PCI-X Device Must Support 33MHz and 66MHz PCI

A 133MHz-capable PCI-X device must support both 33MHz and 66MHz PCI operation.

When M66EN Is Grounded on a Card

As illustrated in Figure 4-1 on page 57, M66EN on the system board has a required pull-up resistor on it that maintains it in the asserted state unless a card installed in a connector grounds it.

- When a 33MHz PCI card is installed in a connector on a 66MHz PCI bus or a PCI-X bus, it grounds the M66EN signal to the Source Bridge.
- When a 66MHz PCI-X card capable of 33MHz (but not 66MHz) PCI operation is installed in a connector, it grounds M66EN.

Effect of Grounded M66EN On a PCI Source Bridge

This informs the clock generator within the PCI bridge that the clock must be divided by two to ensure proper operation of the 33MHz-capable PCI device.

Effect of Grounded M66EN On a PCI-X Source Bridge

A low on M66EN informs a PCI-X bridge that if the PCI protocol is used, it must use a clock speed of 33MHz or less in order to ensure proper operation of the 33MHz-capable PCI device.
M66EN Usage On a 66MHz PCI Card or a PCI-X Card

On a 66MHz-capable PCI device or on a PCI-X card, M66EN is either:

- not-connected to anything on the card. In this case, by not grounding M66EN, the card (either PCI or PCI-X) is indicating that it supports 66MHz PCI operation.
- Alternatively, M66EN may be connected as an input on the card. This is the case if a PCI card can only operate properly on a high-speed PCI bus. It can determine if it’s located on a high- or low-speed bus by sampling the state of its M66EN input. If it determines that it is located on a low-speed PCI bus, it could then set a bit in a device-specific status register to inform its device driver that it needs to be relocated to a high-speed PCI bus in order to achieve adequate performance.

All PCI-X Devices Support 66MHz PCI-X Mode

All PCI-X devices support operation in PCI-X protocol mode from 50MHz through 66MHz. Optionally, a PCI-X device may support operation from 50MHz through 133MHz.
PCIXCAP Indicates Protocol/Frequency Required

General

PCIXCAP (pin B38) is the only new signal defined on the standard PCI connector to support PCI-X. Refer to Figure 4-2 on page 60 and Table 4-1 on page 58. The PCIXCAP signal is tested (along with M66EN) by the Source Bridge at power-up time to determine the least-capable device present on the bus.

Table 4-1: Treatment of PCIXCAP On Add-In Card

<table>
<thead>
<tr>
<th>Treatment On Card</th>
<th>Effect on PCIXCAP Input To Source Bridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grounded on a PCI card.</td>
<td>Grounded, providing a logic low on the Source Bridge's PCIXCAP input. This informs the Source Bridge that at least one card installed on the PCI-X bus is a PCI device. The least-capable device on the bus is therefore a PCI device. The bridge must therefore inform all PCI-X capable devices on the bus that the PCI protocol will be used during this power-up session.</td>
</tr>
<tr>
<td>Tied to a decoupling capacitor.</td>
<td>Pulled to a logic one by the pull-up resistor on the system board unless grounded by a PCI card or partially pulled down by a 66MHz capable PCI-X card. If the bridge detects a logic one on PCIXCAP, the least-capable device on the bus is therefore a 133MHz-capable PCI-X device. The bridge must inform all PCI-X capable devices on the bus that the PCI-X protocol will be used during this power-up session and the clock speed will operate in the range between 66MHz and 133MHz.</td>
</tr>
</tbody>
</table>
Chapter 4: Device Types and Bus Initialization

Table 4-1: Treatment of PCIXCAP On Add-In Card (Continued)

<table>
<thead>
<tr>
<th>Treatment On Card</th>
<th>Effect on PCIXCAP Input To Source Bridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulled partially down by a 10K Ohm resistor</td>
<td>The on-card resistor forms a voltage divider with the pull-up resistor on the system board, yielding a signal level somewhere between a logic high and a logic low (unless PCIXCAP is grounded by a PCI card). If the bridge detects a logic level somewhere between a logic high and a logic low on PCIXCAP, the least-capable device on the bus is therefore a 66MHz-capable PCI-X device. The bridge must inform all PCI-X capable devices on the bus that the PCI-X protocol will be used during this power-up session and the clock speed will operate in the range between 50MHz and 66MHz.</td>
</tr>
</tbody>
</table>
5

PCI-X Is a Registered Bus

The Previous Chapter
As introduced in an earlier chapter (see “Bus Protocol/Speed = Lowest Common Denominator” on page 48), the Source Bridge for a PCI-X bus interrogates two signals at startup time to determine the protocol and bus speed to be used on its secondary bus during this power-up session. The protocol and speed is selected to ensure that the least-capable device functions correctly. The previous chapter provided a detailed description of this process.

This Chapter
This chapter describes the registered nature of the PCI-X bus.

The Next Chapter
The next chapter introduces the PCI-X commands (i.e, the transaction types). They are divided into three categories:

- The Dword commands.
- The Burst commands.
- The Dual-Address Cycle (DAC) command.

PCI-X Is a Low-Voltage Swing (LVS) Bus
When a bus is designed to run at high clock speeds, the speed at which an output driver can change the state of a signal from one logic state to another becomes very important. While a 5 Volt signaling environment is fine for a PCI bus running at speeds up to 33.33MHz, at 66MHz it would take an output driver too long to change the state of a signal line from one logic state to another. That’s why 66MHz PCI must be implemented using 3.3 Volt drivers and receivers. It takes less time to change the state of a signal on a bus that utilizes a low-Voltage swing (LVS). PCI-X devices may also optionally support the 5V signaling environment when operating in 33MHz PCI mode.
Likewise, to ensure proper operation at speeds ranging from 66MHz-to-133Mhz, PCI-X is implemented as an LVS bus and requires all devices to implement 3.3 Volt drivers and receivers.

Introduction To the Registered Nature of the Bus

Due to the high-speed nature of the PCI-X clock (50-133MHz), the bus protocol dictates that all devices must employ registered inputs and outputs. Refer to the general example illustrated in Figure 5-1 on page 85:

**Clock 1.**

*On* the rising-edge of clock one:

- a device’s output register clocks out a new state onto a signal line.

**Clock 2.**

*On* the rising-edge of clock two:

- Another device (or devices) latches the state of the signal line into its input register.

*During* clock cycle two:

- The registered signal level is examined by the receiving device’s logic and the device develops its response to the signal’s registered state.

**Clock 3.**

*On* the rising-edge of clock three:

- The receiving device clocks its response out of its output register onto the appropriate signal line.
Address/Command Decode Example

Refer to the example pictured in Figure 5-2 on page 86:

**Clock 3.**
- **On** the rising-edge of clock three:
  - All targets clock the address and command presented on AD[31:0] and C/BE[#3:0] into their input registers.
- **During** clock cycle three:
  - The targets begin the decode. One of the targets has been configured to recognize this address, so its internal decode logic develops an asserted level (low) on its internal version of DEVSEL# (s1_DEVSEL#).

**Clock 4.**
- **On** the rising-edge of clock four:
  - The target clocks an asserted level onto the external DEVSEL# signal.

**Clock 5.**
- **On** the rising-edge of clock five:
  - The initiator clocks DEVSEL# into its input register for the first time.
**PCI-X System Architecture**

**During** clock cycle five:
- The initiator’s internal logic samples its internal, registered version of DEVSEL# (s1_DEVSEL#) and detects it asserted. This informs the initiator that it has connected with a target.

*Figure 5-2: Registered Bus Example Using Command/Address Decode*

---

**Data Read Example**

This example continues that discussed in the previous example. Refer to the example pictured in Figure 5-3 on page 88:

**Clock 5.**

**During** clock cycle five:
- The initiator’s internal logic detected that the target had asserted DEVSEL# during clock four. This defines the initiator’s first data sampling point as the rising-edge of clock six.
Chapter 5: PCI-X Is a Registered Bus

Clock 6.  
On the rising-edge of clock six:
- The initiator registers data from AD[31:0] and the state of TRDY#.
During clock cycle six:
- The initiator detects the deasserted state of its registered version of TRDY# (s1_TRDY#). As a result, it discards the data that was registered from AD[31:0] on the rising-edge of clock six.

Clock 7.  
On the rising-edge of clock seven:
- The initiator once again registers data from AD[31:0] and the state of TRDY#.
During clock cycle seven:
- The initiator detects the deasserted state of its registered version of TRDY# (s1_TRDY#). As a result, it discards the data that was registered from AD[31:0] on the rising-edge of clock seven.

Clock 8.  
On the rising-edge of clock eight:
- The initiator once again registers data from AD[31:0] and the state of TRDY#.
During clock cycle eight:
- The initiator detects the asserted state of its registered version of TRDY# (s1_TRDY#). As a result, it places the data that was registered from AD[31:0] on the rising-edge of clock eight into its internal read buffer.
The Previous Chapter

The previous chapter described the registered nature of the PCI-X bus.

This Chapter

This chapter introduces the PCI-X commands (i.e., the transaction types). They are divided into three categories:

- The Dword commands.
- The Burst commands.
- The Dual-Address Cycle (DAC) command.

The Next Chapter

The next chapter introduces the four phases of a PCI-X transaction:

- Address Phase.
- Attribute Phase.
- Response Phase.
- Data Phase(s).

Commands Fall Into Two Categories

PCI-X commands (i.e., transaction types) can be divided into three categories:

- those that only transfer a single dword or a subset thereof;
- those that can be used to transfer blocks of between one byte and 4KB of memory data.
- the Dual-Address Cycle (DAC) command which is only used to address memory above the 4GB address boundary.
The following three sections provide an introduction to each of the commands. A detailed discussion of the commands can be found in Chapter 11, entitled “Detailed Command Description,” on page 147. A detailed discussion of the DAC command can be found in “Addressing Memory Above 4GB Boundary” on page 372.

**Command Encoding**

The command and address are issued on C/BE[3:0] by the initiator in the transaction’s Address Phase (see clock two of Figure 6-1 on page 91). Table 6-1 on page 90 lists the Dword command encoding, while Table 6-2 on page 90 lists the Burst transaction command encoding.

**Table 6-1: PCI-X Dword Commands**

<table>
<thead>
<tr>
<th>C/BE[3:0]# or C/BE[7:4]# (binary)</th>
<th>PCI-X Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Interrupt Acknowledge</td>
</tr>
<tr>
<td>0001</td>
<td>Special Cycle</td>
</tr>
<tr>
<td>0010</td>
<td>I/O Read</td>
</tr>
<tr>
<td>0011</td>
<td>I/O Write</td>
</tr>
<tr>
<td>0110</td>
<td>Memory Read Dword</td>
</tr>
<tr>
<td>1010</td>
<td>Configuration Read</td>
</tr>
<tr>
<td>1011</td>
<td>Configuration Write</td>
</tr>
</tbody>
</table>

**Table 6-2: PCI-X Burst Commands**

<table>
<thead>
<tr>
<th>C/BE[3:0]# or C/BE[7:4]# (binary)</th>
<th>PCI-X Command</th>
<th>Byte Enable Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111</td>
<td>Memory Write</td>
<td>Byte Enables define bytes to be transferred within target dword. Any combination of Byte Enables is valid.</td>
</tr>
</tbody>
</table>
Table 6-2: PCI-X Burst Commands (Continued)

<table>
<thead>
<tr>
<th>C/BE[3:0]# or C/BE[7:4]# (binary)</th>
<th>PCI-X Command</th>
<th>Byte Enable Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Alias to Memory Read Block</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>Alias to Memory Write Block</td>
<td>Byte Enables are reserved during Data Phases and must be driven deasserted (high) by initiator.</td>
</tr>
<tr>
<td>1100</td>
<td>Split Completion</td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>Memory Read Block</td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>Memory Write Block</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-1: Example PCI-X Transaction
Dword Commands

General

As mentioned earlier, the dword commands are used to transfer a single dword or a subset thereof. The dword commands are:

- IO Read and Write commands
- Memory Read Dword command
- Configuration Read and Write commands
- Interrupt Acknowledge command
- Special Cycle command

The sections that follow introduce each of these commands.

IO Read and Write Commands

The IO Read and IO Write commands are used to access a device’s control/status/data registers when they are mapped into IO space rather than memory space. Hypothetically, in PCI the IO Read and Write commands can be used to perform burst IO reads or writes. In reality, however, they have only been implemented to transfer single IO dwords or a subset thereof. The PCI-X spec is therefore just recognizing reality in only permitting single Data Phase IO transactions.

For a detailed description of the IO Read and Write commands, refer to “IO Read and Memory Read Dword” on page 216 and “IO Write” on page 224.

Memory Read Dword Command

The Memory Read Dword command is used to read a single dword or a subset thereof from memory-mapped IO ports (i.e., a device’s control/status/data registers when they are mapped into memory space rather than IO space).

For a detailed description of the Memory Read Dword Command, refer to “IO Read and Memory Read Dword” on page 216.
Chapter 6: Intro to Commands

Configuration Read and Write Commands

The Configuration Read and Write commands are used to access a function’s configuration registers. Hypothetically, in PCI the Configuration Read and Write commands can be used to perform burst Configuration reads or writes. In reality, however, they have only been implemented to transfer single Configuration dwords or a subset thereof. The PCI-X spec is therefore just recognizing reality in only permitting single Data Phase configuration transactions.

For a detailed description of the Configuration Read and Write commands, refer to:

- “Configuration Read and Write Transactions” on page 234.
- Chapter 20, entitled “Configuration Transactions,” on page 425.

Interrupt Acknowledge Command

Refer to Figure 6-2 on page 94. The Interrupt Acknowledge command is issued by the Host/PCIX Bridge (see Figure 6-2 on page 94) to read the interrupt vector from the interrupt controller and deliver it to the processor. The interrupt vector can be between one byte and four bytes in length and can therefore be read in a single Data Phase transaction.

For a detailed description of the Interrupt Acknowledge command, refer to “Interrupt Acknowledge Command” on page 234.
The Previous Chapter

The previous chapter introduces the PCI-X commands (i.e., the transaction types). They are divided into three categories:

- The Dword commands.
- The Burst commands.
- The Dual-Address Cycle (DAC) command.

This Chapter

This chapter introduces the four phases of a PCI-X transaction:

- Address Phase.
- Attribute Phase.
- Response Phase.
- Data Phase(s).

The Next Chapter

The next chapter introduces the reasons why the initiator or target would terminate a transaction. It also introduces the manner in which a bridge handles transactions that target internal locations within the bridge as well as transactions that must traverse the bridge.
The PCI Transaction Phases

The PCI transaction consists of two phases:

- In the **Address Phase**, the initiator drives out the address and the command and asserts FRAME# to indicate that a transaction has been initiated.
- In each **Data Phase** (the transaction may include one or more Data Phases), the initiator transfers a data item between itself and the target. A data item consists of a dword or a subset thereof (if it’s a 32-bit transfer), or a qword or a subset thereof (if it’s a 64-bit transfer).

The PCI-X Transaction Phases

Refer to Figure 7-1 on page 102. Each PCI-X transaction consists of four or more phases: the Address Phase, the Attribute Phase, the Response Phase, and the Data Phase(s). The following sections provide a brief description of each phase.

**Address Phase**

In the **Address Phase** (see clock two of Figure 7-1), the initiator drives out the **start byte address** onto the AD bus, the **command** onto the C/BE# bus, and asserts **FRAME#** to indicate that a transaction has been initiated. The Address Phase is one clock in duration (note, however, that if the initiator uses the DAC command, there will be two Address Phases, each of which is one clock in duration). The targets on the bus clock the address, the command, and the state of the FRAME# signal into their respective input registers on the next rising-edge of the clock. A detailed description of the Address Phase can be found in Chapter 13, entitled "The Address, Attribute and Response Phases,” on page 197.

**Attribute Phase**

The **Attribute Phase** (see clock three of Figure 7-1) is one clock in duration and immediately follows the Address Phase. In the Attribute Phase, the initiator outputs the following information onto AD[31:0] and C/BE#[3:0]:

- The byte transfer count (if it’s a memory burst transaction), or the Byte Enables (if it’s a dword transaction).
Chapter 7: Intro to Transaction Phases

- The identity of the entity that originated the transaction request (the initiator of the current transaction could be the Requester, or a bridge passing the transaction along for a master on the other side of the bridge).
- The transaction ID.
- If it’s a memory transaction, the No Snoop and Relaxed Ordering attribute bits.
- If it’s a configuration transaction, the number of the bus that the transaction is being performed on.
- If it’s a Split Completion transaction, a set of attributes different than those mentioned earlier in the list.

A detailed description of the Attribute Phase can be found in Chapter 13, entitled "The Address, Attribute and Response Phases," on page 197.

Response Phase

In the Response Phase (see clock four of Figure 7-1), the initiator awaits the target’s assertion of DEVSEL#. The assertion of DEVSEL# informs the initiator that it has successfully connected with a target. If no target asserts DEVSEL#, the initiator experiences a Master Abort and returns the bus to the Idle state. In all cases other than a Special Cycle transaction, this is an error.

A detailed description of the Response Phase can be found in Chapter 13, entitled "The Address, Attribute and Response Phases," on page 197.

Data Phase(s)

In the Data Phase (or Data Phases; see clocks five-through eight of Figure 7-1), a data item (a dword or a qword, or a subset of a dword or a qword) is transferred between the initiator and target on each rising-edge of the clock when both IRDY# and TRDY# are asserted.

A more detailed description can be found in:

- Chapter 12, entitled "Latency Rules," on page 189.
- Chapter 14, entitled "Dword Transactions," on page 215.
- Chapter 15, entitled "Burst Transactions," on page 239.
- Chapter 16, entitled "Transaction Terminations," on page 279.
Figure 7-1: Example PCI-X Burst Transaction

<table>
<thead>
<tr>
<th>Address Phase</th>
<th>Attribute Phase</th>
<th>Response Phase</th>
<th>Data Phase 1</th>
<th>Data Phase 2</th>
<th>Data Phase 3</th>
<th>Data Phase 4</th>
<th>Bus Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

CLK

FRAME#

AD[31:0]

C/BE#[3:0]

IRDY#

TRDY#

DEVSEL#
The Previous Chapter
The previous chapter introduced the four phases of a PCI-X transaction:

- Address Phase.
- Attribute Phase.
- Response Phase.
- Data Phase(s).

This Chapter
This chapter introduces the reasons why the initiator or target would terminate a transaction. It also introduces the manner in which a bridge handles transactions that target internal locations within the bridge as well as transactions that must traverse the bridge.

The Next Chapter
In PCI-X, Split Transactions take the place of PCI Delayed transactions. The next chapter introduces the concepts and terminology associated with PCI-X Split Transactions. In addition, it defines PCI-X Immediate transactions.

Introduction
The initiator and the target involved in a transaction have various reasons and methods of terminating a transaction. The primary and best reason is that they have achieved byte count satisfaction. In other words, they’re done. The sections that follow introduce the reasons for and the methods used to terminate a transaction.
More detailed descriptions of the various terminations are found in subsequent chapters such as Chapter 16, entitled "Transaction Terminations," on page 279.

Initiator Termination of Transaction

There are four reasons why the initiator would terminate a transaction of its own accord:

- It’s completed its transfer and it’s happy.
- It’s almost full and it wants the target to stop feeding it.
- It’s almost empty and it wants to temporarily stop the transaction until it has more data to write.
- It’s lonely because nobody responds to its request.

That’s a pretty folksy way of stating it, so here’s the more technically correct way of stating it:

- It has achieved byte count satisfaction.
- It’s approaching a buffer dry or a buffer full condition before the byte transfer count has been satisfied.
- No connection has been established with the addressed target.

Byte Count Satisfaction

As stated earlier, the primary and best reason for terminating a transaction is because the byte transfer count has been exhausted. The initiator issues a byte-specific start address and a byte transfer count at the start of the transaction (in the Address and Attribute Phases, respectively). Both parties decrement the initial byte count and terminate the transaction upon completing the final Data Phase.

A detailed description of this termination type can be found in “Byte Count Satisfaction” on page 280.

Initiator Approaching Buffer Full or Dry Condition

As mentioned in “Data Transferred in Blocks” on page 31, data in PCI-X transactions is transferred in blocks of 128 bytes each. Consider the following example scenario:
Chapter 8: Intro to Transaction Termination

1. A PCI-X device wishes to write 2KB of data into memory, but it currently has only 512 bytes of write data buffered up and ready to blast out to memory.
2. While continuing to accumulate the additional data from a background source (e.g., a disk drive or the network), the device arbitrates for bus ownership and initiates a Memory Write Block transaction specifying the full 2KB to be written.
3. The background data source cannot keep up with the device’s transfer rate, so the device will approach a buffer dry condition before it has accumulated the full 2KB of write data to pump out to memory.
4. As the write progresses, the initiator writes several blocks and then starts to transfer the last data block it has available.
5. As it approaches that block boundary, the initiator signals to the target that it needs to disconnect on the upcoming block boundary. This is referred to as a Disconnect At Next ADB issued by the initiator. The target will honor it and both of them will back off the bus in the clock immediately following the transfer of the last data item of the block.

A detailed description of initiator issuance of a Disconnect at Next ADB can be found in “Initiator Issues Disconnect At Next ADB” on page 289.

Connection Timeout

As in PCI, the initiator allows a finite amount of time for the addressed target to indicate the establishment of a connection (by asserting DEVSEL#). If the target doesn’t respond within the allotted time, the initiator experiences a Master Abort and returns the bus to the Idle state to make the bus available to other initiators. As in PCI, a PCI-X initiator samples DEVSEL# a total of four times and, if it is deasserted all four times, the initiator returns the bus to the Idle state.

In all cases other than the Special Cycle transaction this is an error, while it is the natural end to a Special Cycle transaction (because no target is supposed to respond to the quiet broadcast of a message). In transactions other than Special Cycle, the initiator sets the Received Master Abort bit in its PCI configuration Status register (see Figure 8-1 on page 106) and invokes its driver, typically by generating an interrupt, to come check its status.

A complete description of Master Abort can be found in “Initiator Termination Due To Connection Timeout” on page 290.
Early Target Termination of Transaction

General

There are a number of reasons why a target would terminate a transaction before the requested data transfer has completed:

- It’s broken.
- The initiator is addressing locations it doesn’t implement.
- It’s full and has no room to memorize the initial request or, if it’s a memory write, to accept the write data.
- The initiator wants to burst, but it can’t handle bursts.
- It’s going to take a while to deliver the write data or to fetch the read data requested, so the target memorizes the transaction and breaks the connection. It will reconnect with the request originator (i.e., the Requester) later to give you the result.
Target Abort

Reasons For a Target Abort

The following are the three reasons why a target might issue a Target Abort to the initiator:

- The target is broken and issues a Target Abort to the initiator, forcing the initiator to terminate the transaction.
- The target of a Split Completion transaction (the Requester) detects an Address Phase parity error.
- Observing the Byte Enable settings in a Data Phase, the target determines that the initiator is addressing one or more unimplemented IO or memory-mapped IO ports within the currently-addressed IO dword, or the memory-mapped IO dword. Consequently, the target issues a Target Abort to the initiator.

Target Abort Always Ends a Transaction

The reception of a Target Abort in the first or any other Data Phase always causes the initiator to terminate the transaction.

- The target sets the Signaled Target Abort bit in its PCI configuration Status register (see Figure 8-1 on page 106).
- The initiator sets the Received Target Abort bit in its PCI configuration Status register (see Figure 8-1 on page 106). In addition, the initiator invokes its driver, typically by generating an interrupt, to come check its status.

A detailed description of a Target Abort can be found in “Target Abort” on page 290.

Retry

Reason For Issuing a Retry

The following are reasons why a target might issue a Retry to the initiator:

- Upon the initiation of a memory write transaction, the target has a temporarily full posted memory write buffer condition and can not accept any data from the initiator. It issues a Retry to the initiator.
Intro to Split and Immediate Transactions

The Previous Chapter
The previous chapter introduced the reasons why the initiator or target would terminate a transaction. It also introduced the manner in which a bridge handles transactions that target internal locations within the bridge as well as transactions that must traverse the bridge.

This Chapter
In PCI-X, Split Transactions take the place of PCI Delayed transactions. This chapter introduces the concepts and terminology associated with PCI-X Split Transactions. In addition, it defines PCI-X Immediate transactions.

The Next Chapter
The next chapter provides a detailed description of bus arbitration in the PCI-X bus environment. It provides:

- a detailed, step-by-step description of an example arbitration among a number of initiators.
- the initiator design rules
- the arbiter design rules
- a discussion of bus parking
- and a detailed description of preemption and the initiator’s Latency Timer.
Definition of Requester and Completer

The Requester is defined as the originator of a transaction request, while the Completer is the target addressed in the request. They may both be on the same bus or could be on different buses. In that case, the request would have to traverse one or more bridges to arrive at the Completer.

Definition of a Sequence

A Sequence is defined as the series of one or more transactions that satisfy a transaction request issued by a Requester.

Definition of Requester ID, Tag, and Sequence ID

When a Requester issues a transaction request, it issues the following information:

- The start byte address and the command.
- The byte transfer count.
- The Requester ID in the form of its Bus Number, Device Number, and Function Number.
- The Requester assigns a unique transaction ID, referred to as the transaction Tag, to the transaction.

Taken together, the Requester ID plus the Tag forms a unique request identifier, referred to as the Sequence ID.

How Does a Device Know Its Requester ID?

In the previous section, it states that a PCI-X initiator supplies the Completer with its Requester ID (Bus Number, Device Number, and Function Number) whenever it issues a transaction. It should be obvious that a PCI-X function explicitly knows what function it is within a PCI-X device, but how would it know which device (i.e., package) it lives in, and what bus it resides on?

Each PCI-X function implements a PCI-X Status register (see Figure 9-1 on page 115) in its configuration space. This register contains the function’s Requester ID information. The Function field is hardwired, but the Bus and Device fields are
read/writable. On power-up, they do not contain valid information. When the configuration software discovers a function (by reading its Vendor ID register) and causes the Source Bridge that the function resides on to generate one or more Type 0 configuration write transactions to the function’s configuration registers, the bridge automatically supplies the function targeted by the configuration write with the bus number (supplied from the bridge’s Secondary Bus Number register) and device number (supplied in the address of the configuration write transaction). The function updates its Bus and Device fields each time a Type 0 configuration write is performed to any of its configuration registers.

Detailed descriptions of the PCI-X configuration registers and configuration transactions can be found in:

- Chapter 21, entitled "Non-Bridge Configuration Registers," on page 453.
- Chapter 22, entitled "Bridge Configuration Registers," on page 487.

Figure 9-1: PCI-X Status Register
Immediate Transaction

Definition of An Immediate Transaction

An Immediate transaction is defined as:

- A transaction that transfers at least some of the data immediately. This includes:
  - A transaction that receives a Single Data Phase Disconnect.
  - A transaction that transfers some data and then receives a Disconnect At Next ADB.
  - A transaction that immediately transfers all of the data.
  - A transaction that terminates due to a Master or Target Abort.

Transactions that do not complete as Immediate transactions are:

- Transactions terminated with Retry.
- Transactions terminated with Split Response.

Immediate Completion Completes the Sequence...

When a Requester experiences immediate completion of a transaction (see the previous section), the Sequence is considered complete and the requester may or may not re-arbitrate for the bus to resume the transaction. If it chooses to resume the transaction, it may use the same or a different Tag (because an immediate completion retires the Tag assigned to the current transaction).

...Unless It’s a Memory Write or Memory Write Block

The spec contains the following statement (note that the italicized text has been added by the author):

“If the Sequence is a burst write and is disconnected either by the initiator (by issuing a Disconnect At Next ADB) or target (by issuing a Disconnect At Next ADB, or a Single Data Phase Disconnect), the Sequence has more than one transaction. After a disconnection, the initiator must resume the sequence (using the same Sequence ID) by initiating another burst write transaction using the same command and adjusting the starting address and..."
Chapter 9: Intro to Split and Immediate Transactions

byte count for the data already sent. The initiator must deliver the full byte count of the Sequence no matter how many times the Sequence is disconnected and regardless of whether continuations after a disconnection are terminated with Retry.”

... or Unless First Data Phase of Memory Write Receives Retry

The spec says the following regarding Retry and memory writes:

“If the Sequence is a burst write and the target signals Retry on the first Data Phase of the Sequence, the Sequence ends immediately. The requester is not obligated to repeat a Sequence terminated with Retry on the first Data Phase (unless it is required by the application, e.g., a PCI-X bridge forwarding a memory write Sequence). However, if the requester repeats the burst write, it is considered a new Sequence and is permitted to use the same or a different command and attributes (byte count, Tag, etc.). The requester is permitted to reuse a Tag as soon as the memory write Sequence completes on the requester’s bus, independent of whether there are one or more PCI-X bridges between the requester and completer, and if so, when those bridges forward the Sequence to the completer.”

Memory Writes Are Posted

General

Targets handle Memory Write and Memory Write Block transactions by posting the write data in a posted-write buffer and writing the data to memory at a later time when it flushes its buffers.

Can Be Initiated Before All Write Data Is Ready

A Requester can initiate a Memory Write or a Memory Write Block transaction when it has some of the write data (but not all of it) buffered up and ready to write. During the transaction, as it approaches the block boundary where it will have a buffer dry condition, it issues a Disconnect At Next ADB to the target. When it has more of the write accumulated and ready to write, it would then rearbitrate for the bus and pick up where it left off.
The Previous Chapter

In PCI-X, Split Transactions take the place of PCI Delayed transactions. The previous chapter introduced the concepts and terminology associated with PCI-X Split Transactions. In addition, it defined PCI-X Immediate transactions.

This Chapter

This chapter provides a detailed description of bus arbitration in the PCI-X bus environment. It includes:

- a detailed, step-by-step description of an example arbitration among a number of initiators,
- the initiator design rules,
- the arbiter design rules,
- a discussion of bus parking,
- and a detailed description of preemption and the initiator’s Latency Timer.

The Next Chapter

The next chapter provided a detailed description of the Dword commands, the Burst commands, and the Dual-Address Cycle (DAC) command.

Stepping Not Permitted

Address and Data Stepping are not permitted on a bus operating in PCI-X mode, so this behavior does not need to be accounted for by the arbiter.

Request and Grant Signals Are Registered

Refer to Figure 10-1 on page 130. The arbiter clocks all of the REQ# signals into its input register on the rising-edge of the clock and submits the registered ver-
sions of the REQ# signals to its internal arbitration logic during that clock cycle. The GNT# outputs of its arbitration logic are clocked out of its output register on the next rising-edge of the clock.

The spec refers to the arbiter’s registered version of a REQ# signal as $s1_{\text{REQ}#}$, and to its internal version of a GNT# signal as $s1_{\text{GNT}#}$.

**Figure 10-1: The Arbiter**

---

**Example Arbitration**

Refer to Figure 10-2 on page 131 during the following discussion. In this scenario, there are three initiators competing for bus ownership. The example scenario has the following characteristics:

- Initiator A has the lowest priority, B has mid-level priority, and C has the highest priority.
- At some point earlier in time, A had requested bus ownership, had received its grant and had initiated a bus transaction.
- When A started its transaction, it kept its REQ#-A line asserted because it has a second transaction that it wishes to perform after it completes the first one.
At some earlier point in time, initiator A had requested bus ownership (REQ#-A asserted), the arbiter had granted it ownership (GNT#-A asserted), and initiator A had started a transaction (FRAME# and IRDY# asserted).

Clock 1. At some earlier point in time, initiator A had requested bus ownership (REQ#-A asserted), the arbiter had granted it ownership (GNT#-A asserted), and initiator A had started a transaction (FRAME# and IRDY# asserted).

**During** clock cycle one:
- Initiator B asserts REQ#-B to request bus ownership.
The arbiter registers REQ#-B and submits s1_REQ#-B to its internal arbitration logic during clock two.

**During** clock cycle two:
- The arbiter detects s1_REQ#-B and determines that initiator B is to be the next owner of the bus (because it has a higher priority than initiator A). As a result, the arbiter deasserts s1_GNT#-A during clock two.

**Clock 3.**
**On** the rising-edge of clock three:
- The arbiter clocks the s1_GNT# signals through its output register onto the external bus GNT# signals. This causes GNT#-A to be deasserted in clock three.

**During** clock cycle three:
- The arbiter asserts its s1_GNT#-B output to indicate that initiator B will be the next bus owner (after A has completed its transaction).

**Clock 4.**
**On** the rising-edge of clock four:
- The arbiter clocks the s1_GNT# outputs through to the external bus grant signals, causing the external GNT#-B signal to be asserted.
- Initiator A latches the state of GNT#-A (deasserted) and presents s1_GNT#-A to its internal logic (not shown) during clock four. This informs A that has been preempted and must surrender bus ownership.
- Initiator A deasserted FRAME#, thereby indicating that it will, at the latest, stop driving the bus two clocks later (during clock six). In this example, it is deasserting FRAME# at the exact moment that it registers GNT#-A in the deasserted state. It should be stressed that there is no connection between the loss of its GNT#-A and the fact that its wrapping up its transaction (i.e., it’s a coincidence!). It is finishing up the transaction because it has transferred all of the data or because it is disconnecting at an ADB.
- Initiator C decides that it needs to use the bus and asserts REQ#-C.

**Clock 5.**
**On** the rising-edge of clock five:
- GNT#-B is registered by initiator B and its s1_GNT# signal is submitted to its internal logic (not shown) during clock five.
- Initiator B also registers the state (deasserted) of FRAME# and its s1_FRAME# signal is submitted to its internal logic (not shown).
- The arbiter registers the state of REQ#-C and submits s1_REQ#-C for arbitration during clock five. The arbitration takes place during clock five and the result (s1_GNT#-B deasserted) is available to clock out of the arbiter’s output register on the rising-edge of clock six.
Chapter 10: Bus Arbitration

**Clock 6.**
**On** the rising-edge of clock six:
- Initiator B’s internal logic samples s1_GNT# asserted and s1_FRAME# deasserted.
  - GNT#-B is deasserted.
  - s1_GNT# asserted indicates that B is to be the next bus owner.
  - s1_FRAME# deasserted indicates that, at the latest, the current bus owner (A) is backing its output drivers from the bus during clock six.
- As a result of seeing s1_REQ#-C asserted during clock five, the arbiter deasserted s1_GNT#-B during clock five. This is reflected on the external GNT#-B signal when the inputs of the arbiter’s output register are clocked through the output register on the rising-edge of clock six. This causes GNT#-B to be deasserted.

**During** clock cycle six:
- Initiator B may therefore internally initiate its transaction in clock six. It internally initiates the transaction by:
  - asserting its s1_FRAME# signal,
  - driving the address onto its internal s1_AD bus,
  - and the command onto its internal s1_C/BE bus.
- The arbiter asserts s1_GNT#-C to its output register.

**Clock 7.**
**On** the rising-edge of clock seven:
- Initiator B starts its transaction by clocking out the address, command, and by asserting FRAME#.
- The arbiter’s output register clocks out the s1_GNT# signals and drives them onto the external GNT# signals. This causes GNT#-C to be asserted. Initiator C cannot start its transaction, however, until initiator B finishes up its transaction (not illustrated).

---

**Device Design Rules**

The design rules apply to the design of an initiator are covered in the sections that follow.

---

**No Fast Back-to-Back**

When operating in PCI-X mode, PCI Fast Back-to-Back transactions are not permitted.
The Previous Chapter
The previous chapter provided a detailed description of bus arbitration in the PCI-X bus environment. It included:

- a detailed, step-by-step description of an example arbitration among a number of initiators,
- the initiator design rules,
- the arbiter design rules,
- a discussion of bus parking,
- and a detailed description of preemption and the initiator’s Latency Timer.

This Chapter
This chapter provides a detailed description of the Dword commands, the Burst commands, and the Dual-Address Cycle (DAC) command.

The Next Chapter
The next chapter provides a detailed description of how quickly the initiator and target must be capable of transferring data. This includes the target’s rules of behavior during both startup time and run-time, as well as a description of the Maximum Completion Time limit imposed upon non-bridge devices.

Dword Commands

General
As mentioned in “Dword Commands” on page 92, the dword commands are used to transfer a single dword or a subset thereof. The dword commands are:
PCI-X System Architecture

- IO Read and Write Commands
- Memory Read Dword Command
- Configuration Read and Write Commands
- Interrupt Acknowledge Command
- Special Cycle Command

The sections that follow describe each of these commands.

**Command Encoding**

Table 11-1 on page 148 defines the encoding of the PCI-X dword commands. The command is always issued on C/BE#[3:0] during the transaction’s Address Phase.

<table>
<thead>
<tr>
<th>C/BE[3:0]#</th>
<th>PCI-X Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Interrupt Acknowledge</td>
</tr>
<tr>
<td>0001</td>
<td>Special Cycle</td>
</tr>
<tr>
<td>0010</td>
<td>I/O Read</td>
</tr>
<tr>
<td>0011</td>
<td>I/O Write</td>
</tr>
<tr>
<td>0110</td>
<td>Memory Read Dword</td>
</tr>
<tr>
<td>1010</td>
<td>Configuration Read</td>
</tr>
<tr>
<td>1011</td>
<td>Configuration Write</td>
</tr>
</tbody>
</table>

**Illegal To Assert REQ64# In Dword Transactions**

Because Dword transactions by definition only transfer one dword or a subset thereof, it is illegal for the initiator to assert REQ64# in the transaction’s Address Phase. This would indicate that it wished to transfer a full qword in the Data Phase.
Chapter 11: Detailed Command Description

IO Read and Write Commands

Basic Description

The IO Read and IO Write commands are used to access a device’s control/status/data registers when they are mapped into IO space rather than memory space. Hypothetically, in PCI the IO Read and Write commands can be used to perform burst IO reads or writes. In reality, however, they have only been implemented to transfer a single IO dword or a subset thereof. The PCI-X spec is therefore just recognizing reality in only permitting single Data Phase IO transactions.

The settings on C/BE#[3:0] identify which locations in the IO dword are to be read or written. For a detailed description of the IO Read and Write transactions, refer to “IO Read and Memory Read Dword” on page 216 and “IO Write” on page 224.

Start Address and Byte Enable Format

Start Address Is Byte-Aligned. The initiator must issue a byte-aligned start address on AD[31:0] in the Address Phase of the transaction. If any Byte Enables are asserted in the Attribute Phase, the start byte address issued in the Address Phase must be the address associated with the least-significant Byte Enable asserted. If no Byte Enables are asserted, the start byte address may be that of any of the four locations in the IO dword.

Byte Enable Usage. Any combination of Byte Enables (including none) is permitted in the Attribute Phase. There is one constraint, however. If any Byte Enables are asserted, it is illegal to assert any Byte Enables associated with IO addresses numerically lower than the IO start byte address issued in the Address Phase. Table 11-2 on page 150 lists some example start byte address/Byte Enable combinations.

IO Transaction With No Byte Enables Asserted. It might not seem to make any sense that the spec permits an IO transaction with no Byte Enables asserted. This indicates that the initiator doesn’t really want to read or write any bytes. An example implementation that this would make sense for would be an IO address associated with a hardware device’s trigger port. The initiator addresses the IO port address for a read or a write, but doesn’t care what it reads (if it’s an IO read), or, if it’s a write, the IO port doesn’t expect to receive any write data. The simple act of addressing that IO port for a read or a write triggers the IO device to take some action (e.g., it might be a BIOS flash update trigger port).
Table 11-2: Some Example Start Address/Byte Enable Combinations in IO Read, IO Write, and Memory Read Dword Transactions

<table>
<thead>
<tr>
<th>Start Byte Address Issued in Address Phase (hex)</th>
<th>BE#[3:0] in Attribute Phase</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000100h</td>
<td>0000b</td>
<td>Valid combination. Initiator is addressing all four locations in the IO dword consisting of locations 00000100h-through-00000103h</td>
</tr>
<tr>
<td>00000101h</td>
<td>0001b</td>
<td>Valid combination. Initiator is addressing the last three locations (00000101h-through-00000103) in the IO dword consisting of locations 00000100h-through-00000103h</td>
</tr>
<tr>
<td>00000101h</td>
<td>0000b</td>
<td>Invalid combination. Initiator is addressing IO address 00000101h in the Address Phase, but has asserted BE#[0] in the Attribute Phase (which corresponds to IO address 00000100h). It means the initiator lied in the Address Phase when it said it was starting at IO address 00000101h.</td>
</tr>
<tr>
<td>00000102h</td>
<td>0011b</td>
<td>Valid combination. Initiator is addressing the last two locations in the IO dword consisting of locations 00000100h-through-00000103h.</td>
</tr>
<tr>
<td>00000103h</td>
<td>0111b</td>
<td>Valid combination. Initiator is addressing only the last location in the IO dword consisting of locations 00000100h-through-00000103h.</td>
</tr>
<tr>
<td>00000100h</td>
<td>0110b</td>
<td>Valid combination. Initiator is addressing the first and last locations in the IO dword consisting of locations 00000100h-through-00000103h.</td>
</tr>
<tr>
<td>00000102h</td>
<td>1111b</td>
<td>Valid combination. Initiator is addressing location 0000102h in the Address Phase, but has no Byte Enables asserted in the Attribute Phase.</td>
</tr>
</tbody>
</table>
Chapter 11: Detailed Command Description

Target Response To an IO Access

**General.** The device that acts as the target (i.e., it asserts DEVSEL# to claim the transaction) of the IO Read or Write transaction could be a simple, non-bridge IO target (in other words, it is the Completer and resides on the same bus as the Requester). Alternately, the device acting as the target could be a PCI-X bridge claiming the transaction. The next two sections discuss both of these possible cases.

**Completer Handling of an IO Access.** The Completer can handle the transaction in one of two ways:

- If it can perform the read or write within 16 clocks from the assertion of FRAME#, the Completer can complete the transaction immediately by asserting TRDY# and providing the read data (if it’s an IO Read) or accepting the write data (if it’s an IO Write).
- If the Completer cannot complete the access within 16 clocks from the assertion of FRAME#, it will memorize the transaction (including the write data if it’s an IO write) and issue a Split Response to the initiator within 8 clocks from the assertion of FRAME#. When it has successfully performed the internal read or write (or has incurred an internal error), the Completer arbitrates for the bus and performs a Split Completion transaction to deliver the read data, the write completion message (see “Write Completion Indication” on page 322), or an error message to the Requester.

**When Bridge Acts as Target of IO Access.** There are two possible cases wherein a bridge acts as the target of an IO access:

- the IO access targets an internal location within the bridge. In other words, the bridge is the addressed target (i.e., the Completer).
- the IO Access targets a Completer on the other side of the bridge. In this case, the bridge must split the transaction. It memorizes the transaction (including the write data if it’s an IO write) and issues a Split Response to the initiator. It then performs the IO Read or Write on the destination side of the bridge. When the bridge has successfully performed the read or write on the destination side of the bridge (or has incurred an error in the attempted transaction), the bridge arbitrates for the originating bus and performs a Split Completion transaction to deliver the read data, the write completion message, or an error message to the originator of the IO transaction (see Chapter 17, entitled “Split Completion Messages,” on page 313).
The Previous Chapter
The previous chapter provided a detailed description of the Dword commands, the Burst commands, and the Dual-Address Cycle (DAC) command.

This Chapter
This chapter provides a detailed description of how quickly the initiator and target must be capable of transferring data. This includes the target’s rules of behavior during both startup time and run-time, as well as a description of the Maximum Completion Time limit imposed upon non-bridge target devices.

The Next Chapter
The previous chapter provided a detailed description of the Address, Attribute, and Response Phases of memory burst transactions, Dword transactions, Configuration transactions, and Split Completion transactions.

Initiator Latency Rules

Don’t Start Transfer If You’re Not Ready

When starting a transaction, the initiator of a transaction must (at a minimum) immediately have the data to be transferred within the first block ready to transfer. This is necessary because the initiator of a PCI-X transaction is not permitted to delay (by keeping IRDY# deasserted) the transfer of the data in the first block.

Refer to Figure 12-1 on page 190. It is a rule that the initiator must assert IRDY# in the fourth clock cycle of the transaction (the fifth clock cycle if it uses the Dual-Address Cycle command, as it adds a second Address Phase).
If it’s a Memory Read Block transaction, the initiator’s assertion of IRDY# indicates that it’s immediately ready to start receiving the data within the first block at full speed. In other words, at a minimum, it has reserved sufficient buffer space to hold the data between the start byte address and the current block boundary.

If it’s a Memory Write Block or Memory Write transaction, the initiator’s assertion of IRDY# indicates that it’s immediately ready to start transferring the data within the first block at full speed. In other words, at a minimum, it has buffered up the data between the start byte address and the current block boundary.

No Initiator Wait States Permitted...Ever

Once the transfer of a data block begins (IRDY# and TRDY# both asserted), neither party is permitted to temporarily deassert its respective ready signal to delay the transfer of a data item within a block. In other words, once you assert IRDY# to start the transmission of a block, you must keep it asserted during the entire period while the remainder of that block is transferred.

Figure 12-1: Example Memory Read Block Transaction
Chapter 12: Latency Rules

Behavior When Preempted

For a detailed discussion of preemption, refer to “How the Initiator Deals With Preemption” on page 143.

Target Latency Rules

Target Response Time During Initialization Period

What’s Going On During Initialization Time?

During initialization time, the startup configuration software is accessing the configuration registers within each function to determine the presence of a function as well as its resource requirements. Immediately after RST# is removed from a PCI or a PCI-X function, it may not be prepared to service configuration accesses on a timely basis. As an example, a function’s configuration registers might not contain valid default values immediately after RST# is removed. Perhaps the function must start backloading this information into its configuration registers from a serial EEPROM. In this case, it could be a substantial amount of time after RST# removal before the function can provide read data from or accept write data into its configuration registers. For this reason, functions do not have to obey the 16 clock first Data Phase completion rule during initialization time.

Definition of Initialization Period In PCI

As defined in the PCI 2.2 spec, Initialization Time (Trhfa) begins when RST# is deasserted and completes $2^{25}$ PCI clocks later (32 mega-cycles). This parameter is referred to in the spec as Trhfa (Time from Reset High-to-First-Access). At a bus speed of 33MHz, this equates to 1.0066 seconds, while it equates to 0.5033 seconds at a bus speed of 66MHz. Run-time follows initialization-time. If a target is accessed during initialization-time, it is allowed to do any of the following:

- Ignore the request (except if it is a boot device). A boot device is one that must respond as a target in order to allow the processor to access the boot ROM. In a typical PC design, this would be the PCI-to-ISA bridge. Devices in the processor’s path to the boot ROM should be prepared to be the target of a transaction immediately after Trhff expires (five clock cycles after RST#)
is deasserted).

- Claim the access and hold in Wait States until it can complete the request, not to exceed the end of Initialization Time.
- Claim the access and terminate with Retry.

**Definition of Initialization Period In PCI-X**

In PCI-X, Trhfa is $2^{26}$ clocks (64 mega-cycles) in duration rather than $2^{25}$ as it is in PCI. This is because the PCI-X clock speed can be substantially faster than (up to 133MHz) the PCI clock speed and if this parameter remained the same as the PCI trhfa spec, Initialization Time would be reduced to 0.25 seconds (at a clock speed of 133MHz).

During Initialization Time, a PCI-X target has the same options available as a PCI target does (see previous section).

**Initialization Period and Hot-Plug**

See “Early Configuration Access To Newly-Installed Device” on page 80.

**Target Can Ignore 16-Clock Rule During ROM Shadowing**

The PCI-X spec also states that during system setup a PCI-X target is permitted to ignore the 16-clock rule when software is performing the memory reads to copy (i.e., shadow) the function’s device ROM code into main memory (because the ROM typically has a very slow access time).

**Target Response Time Limit During Run-Time**

**Response Time Limit When No Data Transferred**

A target can terminate a transaction with no data transferred. The following target terminations result in no data being transferred:

- Retry.
- Split Response.
- Target-Abort.

If a target is going to issue one these responses to the initiator, it must do so within eight clocks from the assertion of FRAME#. In other words, issue it quickly to free up the bus for someone else to use.
Chapter 12: Latency Rules

Response Time Limit When Data Transferred

If the target intends to transfer any data at all in the transaction, it must assert TRDY# within 16 clocks from the assertion of FRAME#. This would include the following target terminations:

- Single Data Phase Disconnect.
- Disconnect At Next ADB.

Host/PCIX Bridge Must Obey 16 Clock Rule

In PCI. A Host/PCI bridge that is snooping is optionally permitted to exceed the 16 clock limit, but may never exceed 32 clocks. Assume that a PCI master is accessing main memory. The Host/PCI bridge can start inserting Wait States in the first Data Phase while it sends the memory address back to the processors to be snooped in their caches. In the event of a snoop hit on a modified line, the processor with the modified line will transfer the line to the bridge. If the bridge knows that this process can be accomplished within 32 PCI clocks from the start of the PCI master’s transaction, then it is legal for it to hold the PCI bus in Wait States while the snoop and possibly the memory update take place.

In PCI-X. The Host/PCIX must obey the 16-clock rule just like any other PCI-X target.

Subsequent Data Phase Target Latency Rule

Like the initiator (see “No Initiator Wait States Permitted...Ever” on page 190), once the target asserts TRDY# to indicate its readiness to start the transfer of the first data block, it must keep TRDY# asserted for the remainder of the transmission of the block.

Maximum Completion Time

In PCI

When data is written to a memory target, the target can handle it in one of three ways:

Method 1. The target can immediately accept the data and write it into memory. How fast this can be accomplished is dependent on the write latency of the memory being written to (but it must be able to accomplish the write with 16 clocks).
The Previous Chapter
The previous chapter provided a detailed description of how quickly the initiator and target must be capable of transferring data. This included the target’s rules of behavior during both startup time and run-time, as well as a description of the Maximum Completion Time limit imposed upon non-bridge devices.

This Chapter
This chapter provides a detailed description of the Address, Attribute, and Response Phases of memory burst transactions, Dword transactions, Configuration transactions, and Split Completion transactions.

The Next Chapter
The next chapter provides detailed examples of the Dword transaction types:

- IO Read and Write transactions.
- The Memory Read Dword transaction.
- The Interrupt Acknowledge transaction.
- The Special Cycle transaction.

Although the Configuration Read and Write transactions are Dword transactions, a detailed discussion of them is deferred until Chapter 20, entitled "Configuration Transactions," on page 425.
All Transactions Begin With Address and Attribute Phases

Refer to Figure 13-1 on page 198. Each transaction begins with the Address Phase (clock two), immediately followed by the Attribute Phase (clock three).

During the Address Phase, the initiator outputs:

- The start address on the AD bus (the address type is defined as a memory, IO, or configuration address by the command type).
- The command (i.e., transaction) type on the C/BE# bus. The format of the start address is defined by the command type.

During the Attribute Phase, the initiator delivers additional information about the transaction. This information is delivered on AD[31:0] and C/BE#[3:0]. The format of the Attribute information is defined by the command type.

Subsequent sections in this chapter define the Address/Attribute format for each type of Command.

*Figure 13-1: Example PCI-X Transaction*
Chapter 13: The Address, Attribute and Response Phases

Memory Transaction May Have Two Address Phases

Refer to Figure 13-2 on page 199. When an initiator issues a start memory address that is above the 4GB address boundary, it uses the Dual-Address Cycle command in the first Address Phase (clock two) to inform the targets that this is the first of two Address Phases. The initiator delivers a 64-bit memory start address in two packets over AD[31:0] and it delivers the actual memory command (e.g., Memory Read Block) on C/BE#[3:0] during the second Address Phase (clock three). The Attribute Phase (clock four) immediately follows the second Address Phase.

A detailed description of 64-bit memory addressing can be found in “Addressing Memory Above 4GB Boundary” on page 372.

Figure 13-2: Example Memory Access Above 4GB Address Boundary
Attributes Always Delivered On Lower-Half of Bus

Refer to Figure 13-1 on page 198. The Attributes are always delivered over AD[31:0] and C/BE#[3:0]. During the Attribute Phase of memory transactions initiated by a 64-bit initiator (see clock four in Figure 13-2 on page 199), AD[63:32] and C/BE#[7:4] are always reserved and driven high.

Address/Attribute Format Depends On Command Type

As mentioned earlier, the format of the Address/Attribute information is defined by the command type. The sections that follow define the formats for the various command types.

Memory Burst Format

Description

The memory burst-oriented commands are:

- Memory Read Block
- Memory Write Block
- Memory Write
- Alias To Memory Read Block
- Alias To memory Write Block

Figure 13-3 on page 201 illustrates the information output during the Address Phase of any of these transaction types. The initiator issues the start byte address on AD[31:0] and the memory command type on C/BE#[3:0]. When initiating a transaction with a start memory address above the 4GB address boundary, there are two Address Phases in the transaction. A detailed description of this can be found in “Addressing Memory Above 4GB Boundary” on page 372.

Figure 13-4 on page 201 illustrates the information output during the transaction’s Attribute Phase:
Chapter 13: The Address, Attribute and Response Phases

- **Requester ID.** Consists of the Requester’s Bus Number, Device Number, and Function Number supplied from its PCI-X Status register (see Figure 21-9 on page 462).
- **Transaction Tag.** The Requester’s transaction number.
- **Byte Transfer Count.** This 12-bit field indicates the transfer size in bytes.
  - $000h = 4096$ bytes
  - $001h-FFFh = 1$-through $4095$ bytes.
- **No Snoop (NS) attribute bit.** See “No Snoop Attribute Bit” on page 201.
- **Relaxed Ordering (RO) attribute bit.** See “Relaxed Ordering Effect On Transaction Ordering” on page 569.

Figure 13-3: Memory Transaction Address Phase Format

```
<table>
<thead>
<tr>
<th>C/BE[3:0]#</th>
<th>AD Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>Command</td>
<td></td>
</tr>
<tr>
<td>Start Byte Address</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 13-4: Memory Burst Transaction Attribute Phase Format

```
<table>
<thead>
<tr>
<th>C/BE[3:0]#</th>
<th>AD Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28</td>
<td>24 23 16 15 11 10 8 7 0</td>
</tr>
<tr>
<td>Upper Byte Count</td>
<td>Tag</td>
</tr>
</tbody>
</table>
```

**No Snoop Attribute Bit**

**Background**

*Bridge Knows AGP’s Area of Memory Is Non-Cacheable.* Refer to Figure 13-5 on page 204. The Host/PCI Bridge typically contains the main memory controller. It is also the communication path between the AGP graphics adapter and main memory. The AGP 2.0 spec dictates that the region of main memory allocated to the AGP graphics adapter must be designated as non-cacheable memory. This means that the processors do not
The Previous Chapter

The previous chapter provided a detailed description of the Address, Attribute, and Response Phases of memory burst transactions, Dword transactions, Configuration transactions, and Split Completion transactions.

This Chapter

This chapter provides detailed examples of the Dword transaction types:

- IO Read and Write transactions.
- The Memory Read Dword transaction.
- The Interrupt Acknowledge transaction.
- The Special Cycle transaction.

Although the Configuration Read and Write transactions are Dword transactions, a detailed discussion of them is deferred until Chapter 20, entitled “Configuration Transactions,” on page 425.

The Next Chapter

The next chapter provides detailed examples of the Burst transaction types:

- Memory Read Block.
- Memory Write Block.
- Memory Write.
- Split Completion
- Alias To Memory Read and Write Block commands.
General

All timing diagrams in this chapter assume that the initiator already arbitrated for and has now achieved bus ownership (i.e., its GNT# is asserted and it detects bus Idle (FRAME# and IRDY# both high).

General Format of Timing Diagram Descriptions

The sections that follow provide a detailed description of various transaction scenarios. Each transaction is described clock-by-clock, with the events during each clock divided into two categories:

- **ON** the rising-edge of clock n.
- **DURING** clock n.

Please keep in mind that some events could actually be correctly placed in either of these event categories.

IO Read and Memory Read Dword

Background on the IO Read command may be found in “IO Read and Write Commands” on page 149. Background on the Memory Read Dword command may be found in “Memory Read Dword Command” on page 152.

Figure 14-1 on page 220 and Figure 14-2 on page 224 are two examples of either Memory Read Dword or IO Read transactions. The two sections that follow provide a detailed description of each transaction example.

Example One

**Clock 1.** The transaction has not yet begun. The initiator detected its GNT# asserted (not shown) on the rising-edge of clock one, informing it that it will be the next bus owner. The initiator has been tracking bus activity, so it also knows that the bus will be Idle on the rising-edge of clock two (for more information, refer to Chapter 10, entitled “Bus Arbitration,” on page 129).
Chapter 14: Dword Transactions

**Cl ck 2.**

**On** the rising-edge of clock two:
- The initiator starts the IO Read or Memory Read Dword transaction by:
  - driving out the byte-specific IO or memory address onto AD[31:0].
  - driving out the IO Read or Memory Read Dword command onto C/BE#[3:0].
  - asserting FRAME#.

**Cl ck 3.**

**On** the rising-edge of clock three:
- All targets on the bus clock the address, command, and the state of the FRAME# signal into their input registers.
- The initiator drives out the attributes (see Figure 13-7 on page 206) onto AD[31:0] and the Byte Enables onto C/BE#[3:0]. The Byte Enables identify the bytes to be read from the IO or memory dword.

**During** clock three:
- All targets begin the decode of the registered copy of the address and command. In this example, the currently-addressed target asserts its internal version of the DEVSEL# signal prior to the rising-edge of clock four.

**Cl ck 4.**

**On** the rising-edge of clock three:
- The initiator backs its output drivers off of AD[31:0] (in preparation for the target’s delivery of the data over the AD bus).
- The currently-addressed target clocks an asserted level onto DEVSEL#. This is the Decode A time slot.

**During** clock four:
- The initiator places the Bytes Enables in the Reserved and Driven High state for the remainder of the transaction.

**Cl ck 5.**

**On** the rising-edge of clock five:
- The initiator asserts IRDY#, indicating that it has buffer space available to hold the requested data.
- The rising-edge of clock five is the first point at which the initiator clocks the state of DEVSEL# into its input register. In this case, it detects an asserted level on its registered copy of DEVSEL# during clock cycle five. This tells the initiator it has established a connection with the addressed target and it defines the first point at which the initiator will register the read data and the state of TRDY# as the rising-edge of clock six.
The earliest time slot in which the target can assert TRDY# is the clock immediately following its assertion of DEVSEL#. In this case, the target does not assert TRDY# on the rising-edge of clock five, thereby indicating that it is not yet supplying the requested read data on the AD bus.

On any form of a read, it is a rule that the target must take ownership of the AD bus in the clock immediately following its assertion of DEVSEL#. In this case, the target is not yet ready to drive the requested read data onto the AD bus, so it drives a dummy data pattern to keep the bus from floating until it does have the data ready to deliver.

**Clock 6.**

**On** the rising-edge of clock six:
- The initiator clocks the content of the AD bus and the state of the TRDY# signal into its input register on the rising-edge of clock 6. In this case, the deasserted state of its registered copy of TRDY# tells the initiator that the requested data was not yet present on the AD bus, so the initiator discards the registered data during clock cycle six.

**During** clock six:
- The deasserted state of its registered copy of TRDY# causes the initiator to insert a Wait State in clock cycle six.

**Clock 7.**

**On** the rising-edge of clock seven:
- The initiator once again clocks the content of the AD bus and the state of the TRDY# signal into its input register. In this case, the deasserted state of its registered copy of TRDY# tells the initiator that the requested data was not yet present on the AD bus, so the initiator discards the registered data during clock cycle seven.

**During** clock seven:
- The deasserted state of its registered copy of TRDY# causes the initiator to insert a second Wait State in clock cycle 7.

**Clock 8.**

**On** the rising-edge of clock eight:
- The initiator once again clocks the content of the AD bus and the state of the TRDY# signal into its input register on the rising-edge of clock 8. In this case, the asserted state of its registered copy of TRDY# tells the initiator that the requested data was present on the AD bus, so the initiator places the registered read data into its read buffer during clock cycle eight.
Chapter 14: Dword Transactions

**During** clock eight:
- Knowing that the initiator read the data on the rising-edge of clock eight, the target:
  - Deasserts TRDY#. Since it’s a sustained tri-state signal, the deassertion protocol is to drive it high for one clock cycle and then back off the output driver from the signal line.
  - Deasserts DEVSEL# (also a sustained tri-state signal line).
  - Backs off its output drivers from the AD bus.

**Clock 9.**
- On the rising-edge of clock nine:
  - Because it received the requested read data on the rising-edge of clock eight, on the rising-edge of clock nine the initiator ends the transaction. It:
    - Deasserts FRAME#. Since it’s a sustained tri-state signal, the deassertion protocol is to drive it high for one clock cycle and then back off the output driver from the signal line.
    - Deasserts IRDY# (also a sustained tri-state signal line).
    - Regarding the Byte Enables, in clock cycle nine the initiator may continue to drive them high, drive some other pattern, or back off its output drivers.

**Clock 10.**
- On the rising-edge of clock ten:
  - The bus returns to the Idle state (FRAME# and IRDY# both high).
The Previous Chapter

The previous chapter provided detailed examples of the Dword transaction types:

- IO Read and Write transactions.
- The Memory Read Dword transaction.
- The Interrupt Acknowledge transaction.
- The Special Cycle transaction.

Although the Configuration Read and Write transactions are Dword transactions, a detailed discussion of them is deferred until Chapter 20, entitled "Configuration Transactions," on page 425.

This Chapter

This chapter provides detailed examples of the Burst transaction types:

- Memory Read Block.
- Memory Write Block.
- Memory Write.
- Split Completion
- Alias To Memory Read and Write Block commands.

The Next Chapter

The next chapter provides a detailed description of both initiator and target terminations of a transaction. It includes clock-by-clock descriptions of example timing diagrams.
Introduction

With the exception of the final timing diagram in this chapter (Figure 15-23 on page 276), all of the examples consists of four Data Phases and represent one of the two possible scenarios defined in the next two sections. In both of these cases, the end of the transaction is identical and appears as illustrated in each timing diagram in this chapter. For example, refer to Figure 15-5 on page 249:

- In clock eight, the initiator deasserts FRAME to inform the target that the next-to-last data item (dword or qword) is being transferred in this clock.
- In clock nine, the last data item is transferred.
- The initiator and target surrender the bus in clock 10.

Short Transfer Within a Block

An initiator issues a memory start byte address and a byte transfer count wherein all of the data to be transferred falls within one block. As each data item is transferred, the initiator and target decrement the specified byte transfer count and recognize the approaching transaction end.

Long Transfer, But Disconnect On First Block Boundary

Consider the following set of assumptions:

- An initiator issues a memory start byte address that is four (or more) Data Phases from a block boundary.
- The byte transfer count identifies an end byte address somewhere beyond the imminent block boundary.
- However, the initiator issues aDisconnect At Next ADB to the target as it approaches the imminent block boundary to force a disconnect before all of the data has been transferred.

General Format of Timing Diagram Descriptions

The sections that follow provide a detailed description of various transaction scenarios. Each transaction is described clock-by-clock, with the events during each clock divided into two categories:
Chapter 15: Burst Transactions

- **ON** the rising-edge of clock n.
- **DURING** clock n.

Please keep in mind that some events could actually be correctly placed in either of these event categories.

Memory Read Block Transaction

The Memory Read Block transaction is an all-inclusive read starting at the byte-specific memory start address through the end byte address as defined by the byte transfer count. Additional detail about the Memory Read Block transaction can be found in “Memory Burst Format” on page 200 and in “Memory Read Block Command” on page 167.

Memory Read Block: Detailed Example

Refer to Figure 15-5 on page 249.

**CL ck 1.** The transaction has not yet begun. The initiator detected its GNT# asserted (not shown) on the rising-edge of clock one, informing it that it will be the next bus owner. The initiator has been tracking bus activity, so it also knows that the bus will be Idle on the rising-edge of clock two (for more information, refer to Chapter 10, entitled “Bus Arbitration,” on page 129).

**CL ck 2.**

- **ON** the rising-edge of clock two:
  - The initiator starts the Memory Read Block transaction by:
    - driving out the byte-specific memory start address onto AD[31:0].
      In this example, the start byte address could be any address within the first block.
    - driving out the Memory Read Block command onto C/BE#[3:0].
    - asserting FRAME#.

**CL ck 3.**

- **ON** the rising-edge of clock three:
  - All targets on the bus clock the address, command, and the state of the FRAME# signal into their input registers.
  - The initiator drives out the attributes (see Figure 13-4 on page 201) onto AD[31:0] and C/BE#[3:0]. Among other things, the attributes contain the byte transfer count. In this example, there are two possibilities:
    - The start byte address issued in the Address Phase could be any byte within the first block and the byte transfer count encompasses four Data Phases but does not cross the next block boundary.
The start byte address issued in the Address Phase is one of the locations in the fourth dword from the end of block and the byte transfer count identifies an end address that is beyond the imminent block boundary.

**During** clock cycle three:
- All targets begin the decode of the registered copy of the address and command. In this example, the currently-addressed target asserts its internal version of the DEVSEL# signal prior to the rising-edge of clock four.

**Clock 4.**
- **On** the rising-edge of clock four:
  - The currently-addressed target clocks an asserted level onto DEVSEL#.
  - This is the Decode A time slot.
- **During** clock cycle four:
  - The initiator backs its output drivers off of AD[31:0] (in preparation for the target’s delivery of the data over the AD bus).
  - The initiator places the Byte Enables in the Reserved and Driven High state for the remainder of the transaction.

**Clock 5.**
- **On** the rising-edge of clock five:
  - The initiator asserts IRDY#, indicating that it has buffer space available to hold at least the first block (or a subset thereof) of the requested data.
  - The rising-edge of clock five is the first point at which the initiator clocks the state of DEVSEL# into its input register. In this case, it detects an asserted level on its registered copy of DEVSEL# during clock cycle five. This tells the initiator that it has established a connection with the addressed target. This defines the first point at which the initiator will register the state of TRDY# and the content of the AD bus as the rising-edge of clock six.
  - The target drives the first read data item and asserts TRDY#.

**Clock 6.**
- **On** the rising-edge of clock six:
  - The initiator clocks the first read data item and the state of TRDY# into its input register.
  - The target drives out the second data item.
- **During** clock cycle six:
  - The initiator determines that its internal copy of the TRDY# signal is asserted, indicating that good data was registered on the rising-edge of clock six. The initiator places this data item into its read buffer.
Chapter 15: Burst Transactions

- Internally, both the initiator and the target adjust the byte count remaining and increment their current address pointers to point to the start address of the next dword.

Clock 7.

On the rising-edge of clock seven:
- The initiator clocks the second read data item and the state of TRDY# into its input register.
- The target drives out the third data item.
- The initiator deasserts FRAME# for one of two reasons:
  - This transaction of four (or more) Data Phases is a small transfer, wholly-contained within a block (i.e., it does not cross a block boundary). In this clock, both parties realize (by checking the byte count remaining) that the next-to-last data item is being transferred.
  - The start byte address issued in the Address Phase was a location in the fourth dword from the upcoming block boundary. Although the transfer count specified in the Attribute Phase indicates that the transfer would cross at least the upcoming block boundary, in this example the initiator must force the target to Disconnect At Next ADB. As an example, the initiator of the Memory Read Block may be a PCIX-to-PCIX bridge and it may not have enough buffer space reserved to go beyond the upcoming block boundary. In this case, the initiator deasserts FRAME# to inform the target that it’s disconnecting the transaction on the upcoming block boundary.

During clock cycle seven:
- The initiator determines that its internal copy of the TRDY# signal is asserted, indicating that good data was registered on the rising-edge of clock seven. The initiator places the second data item into its read buffer.
- Internally, both the initiator and the target adjust the byte count remaining.

Clock 8.

On the rising-edge of clock eight:
- The initiator clocks the third read data item and the state of TRDY# into its input register.
- The target clocks the state of the FRAME# signal into its input register and determines that it has been deasserted by the initiator. This informs the target that the data item to be transferred on the rising-edge of clock nine is the final one to be transferred (even if it doesn’t exhaust the transfer count).
- The target drives out the fourth data item. This is either the final data item of the current block, or it is the last data item to be transferred within the block (as defined by the byte transfer count).
The Previous Chapter
The previous chapter provided detailed examples of the Burst transaction types:

- Memory Read Block.
- Memory Write Block.
- Memory Write.
- Split Completion
- Alias To Memory Block commands.

This Chapter
This chapter provides a detailed description of both initiator and target terminations of a transaction. It includes clock-by-clock descriptions of example timing diagrams.

The Next Chapter
The next chapter describes the purpose of the Split Completion Message, its format, the messages associated with writes and reads, and device-specific error handling.

General Format of Timing Diagram Descriptions
The sections that follow provide a detailed description of various transaction scenarios. Each transaction is described clock-by-clock, with the events during each clock divided into two categories:
Termination By the Initiator

General

The initiator of a transaction may terminate a transaction for any of the following reasons:

- Byte count satisfaction.
- The initiator issues a Disconnect At Next ADB.
- The initiator fails to connect with a target.

These terminations are covered in the next three sections. It should also be noted that the initiator may terminate a transaction because the target has instructed it to do so. These target-initiated terminations are covered in "Termination By the Target" on page 290.

Byte Count Satisfaction

Introduction

The protocol used by the initiator when ending a transaction due to byte count satisfaction is defined by the number of Data Phases in the transaction. There are two possibilities:

- Ending a transaction consisting of four or more Data Phases.
- Ending a transaction consisting of less than four Data Phases.

The next two sections describe the methodology used by the initiator in each of these two cases.
Chapter 16: Transaction Terminations

Ending Transaction of Four or More Data Phases

With the exception of Figure 15-23 on page 276, every example transaction in Chapter 15, entitled “Burst Transactions” consisted of four Data Phases and therefore illustrated how the initiator ends a transaction of four (or more) Data Phases due to byte count satisfaction.

Refer to Figure 16-1 on page 281. In short, both the initiator and the target are decrementing the byte transfer count in each Data Phase. In the Data Phase where the next-to-last data item (as defined by the remaining byte transfer count) is transferred, the initiator deasserts FRAME#. The final data item is then transferred in the next Data Phase. Both parties then release the bus. For a more detailed description of this example, refer to “Memory Read Block: Detailed Example” on page 241.

Figure 16-1: Ending Transaction of Four (or more) Data Phases
Ending Transaction of Less Than Four Data Phases

**Introduction.** The protocol used when ending a transaction of less than four Data Phases is defined in the following three sections.

**Three Data Phase Transaction.** Refer to Figure 16-2 on page 284.

**Clock 1.** The transaction has not yet begun. The initiator detected its GNT# asserted (not shown) on the rising-edge of clock one, informing it that it will be the next bus owner. The initiator has been tracking bus activity, so it also knows that the bus will be Idle on the rising-edge of clock two (for more information, refer to Chapter 10, entitled "Bus Arbitration," on page 129).

**Clock 2.**

On the rising-edge of clock two:
- The initiator starts the burst memory transaction by:
  - driving out the byte-specific memory start address onto AD[31:0]. In this example, the start byte address could be any address within the block.
  - driving out the burst memory command onto C/BE#[3:0].
  - asserting FRAME#.

**Clock 3.**

On the rising-edge of clock three:
- All targets on the bus clock the address, command, and the state of the FRAME# signal into their input registers.
- The initiator drives out the attributes (see Figure 13-4 on page 201) onto AD[31:0] and C/BE#[3:0]. Among other things, the attributes contains the byte transfer count. In this example, the start byte address issued in the Address Phase could be any byte within the block and the byte transfer count encompasses three Data Phases within the same block.
- **During** clock cycle three:
  - All targets begin the decode of the registered copy of the address and command during clock cycle three. In this example, the currently-addressed target asserts its internal version of the DEVSEL# signal prior to the rising-edge of clock four.

**Clock 4.** The currently-addressed target clocks an asserted level onto DEVSEL# on the rising-edge of clock 4. This is the Decode A time slot.

**Clock 5.**

On the rising-edge of clock five:
- The initiator asserts IRDY#, indicating that it is ready to start transferring data. If it’s a burst write transaction, the initiator drives out the first data item onto the AD bus.
Chapter 16: Transaction Terminations

- The rising-edge of clock five is the first point at which the initiator clocks the state of DEVSEL# into its input register. In this case, it detects an asserted level on its registered copy of DEVSEL# during clock cycle five. This tells the initiator that it has established a connection with the addressed target. This defines the first point at which the initiator will register the state of TRDY# as the rising-edge of clock six.
- The target asserts TRDY# to indicate that it’s ready to transfer the first data item. If it’s a burst read, the target drives out the first data item.

Clock 6.
- On the rising-edge of clock six:
  - The initiator registers the state of TRDY#. The asserted state of TRDY# indicates that the target has started the transfer (i.e., it is not inserting Wait States to delay the transfer of the data).
  - The device sourcing the data (the initiator if it’s a write or a Split Completion; the target if it’s a read) drives out the second data item. This is the next-to-last data item.
- During clock cycle six:
  - Both parties decrement the remaining byte transfer count. Realizing that the next-to-last data item will be transferred on the rising-edge of clock seven, the initiator prepares to deassert FRAME# on the rising-edge of clock seven. Note that the initiator could not turn off FRAME# if TRDY# had been registered in the deasserted state on the rising-edge of clock six.

Clock 7.
- On the rising-edge of clock seven:
  - The device sourcing the data (the initiator if it’s a write or a Split Completion; the target if it’s a read) drives out the last data item.
  - The initiator deasserts FRAME#.
- During clock cycle seven:
  - Both parties decrement the remaining byte transfer count. Realizing that the last data item will be transferred on the rising-edge of clock eight, the initiator prepares to deassert IRDY#, and the target prepares to deassert TRDY# and DEVSEL# on the rising-edge of clock eight.

Clock 8.
- On the rising-edge of clock eight:
  - The final data item is transferred.
  - The initiator deasserts IRDY# to return the bus to the Idle state.
  - The target deasserts TRDY# and DEVSEL#.
The Previous Chapter

The previous chapter provided a detailed description of both initiator and target terminations of a transaction. It included clock-by-clock descriptions of example timing diagrams.

This Chapter

This chapter describes the purpose of the Split Completion Message, its format, the messages associated with writes and reads, and device-specific error handling.

The Next Chapter

The next chapter provides a detailed discussion of 64-bit data transfers, 64-bit memory addressing, and additional issues associated with 64-bit devices and 64-bit memory addressing.

Purpose of Split Completion Messages

A device sends a Split Completion Message (SCM) back to a Requester if a problem is encountered while attempting to perform the Split Transaction request. The following types of transactions may be split:

- Memory Read Block
- Alias to Memory Read Block
- Memory Read dword
- Interrupt Acknowledge
- I/O Read
- I/O Write
**PCI-X System Architecture**

- Configuration Read
- Configuration Write

Remember that Memory Write and Memory Write Block transactions are never split; they are posted in the target’s posted-memory write buffer.

After issuing a Split Response to a request, the target is required to deliver one or more Split Completion transactions at a later time to fulfill the request or report an error. Table 17-1 on page 314 defines the possible results that may be delivered back to the Requester for a transaction that has been split.

*Table 17-1: After Splitting Transaction, Target Must Deliver Some Form of Completion*

<table>
<thead>
<tr>
<th>Split Request Type</th>
<th>Target is required to deliver:</th>
</tr>
</thead>
</table>
| Memory Read Block (or Alias To Memory Read Block) | • Requested read data, or  
• A message indicating an error occurred when the Completer attempted to read the requested data from its internal locations. 
Note that while the Completer may read and supply the requested read data to the Requester without a problem, the data may become corrupted somewhere in its flight path back to the Requester. In this case, the Requester will check the parity of the returning data and determine that it is bad. |
| IO Read                                 | Note that while the Completer may read and supply the requested read data to the Requester without a problem, the data may become corrupted somewhere in its flight path back to the Requester. In this case, the Requester will check the parity of the returning data and determine that it is bad. |
| Configuration Read                      | • The interrupt vector, or  
• A message indicating an error occurred when the Completer attempted to read the vector from the Interrupt Controller. 
Note that while the Completer may read and supply the interrupt vector to the Requester without a problem, the vector may become corrupted somewhere in its flight path back to the Requester. In this case, the Requester will check the parity of the returning vector and determine that it is bad. |
| Memory Read Dword                        | • A message indicating successful delivery of the write data, or  
• A message indicating that the write data was corrupted in flight. |
| Interrupt Acknowledge                   |                                                                                                 |
| IO Write                                |                                                                                                 |
Chapter 17: Split Completion Messages

Table 17-1: After Splitting Transaction, Target Must Deliver
Some Form of Completion (Continued)

<table>
<thead>
<tr>
<th>Split Request Type</th>
<th>Target is required to deliver:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Write</td>
<td>• A message indicating successful delivery of the write data, or</td>
</tr>
<tr>
<td></td>
<td>• A message indicating that the write data was corrupted in flight.</td>
</tr>
</tbody>
</table>

SCM Always Terminates a Sequence

Receipt of an SCM associated with a request completes the request. The SCM may have been preceded by one or more Split Completion transactions returning previously-requested memory read data.

Upon Receipt of Error Message, Set Status Bit

Upon receipt of a Split Completion Message with the Split Completion Error (SCE) bit set to one in the Attribute Phase (see Figure 17-3 on page 317), the Requester must set the Received Split Completion Error Message bit in its PCI-X status register (see Figure 17-1 on page 315).

Figure 17-1: PCI-X status Register
A Split Completion transaction that contains a Split Completion Message (SCM) always consists of a single Data Phase during which the initiator of the Split Completion supplies the SCM to the Requester.

**Address Phase Format**

The information delivered in the Address Phase is pictured in Figure 17-2 on page 316 and has the following characteristics:

- **Requester ID and Transaction Tag.** Decoded by the Requester to identify which of its previously-Split Transactions the Split Completion Message is associated with.
- **Lower Address field.** Always cleared to zero.
- **Relaxed Ordering bit.** Always cleared to zero.

**Figure 17-2: Split Completion Address Format**

<table>
<thead>
<tr>
<th>3/BE[3:0]</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>11</th>
<th>10</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC Command</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>Tag</td>
<td>Requester Bus Number</td>
<td>Requester Device Number</td>
<td>Requester Function Number</td>
<td>R</td>
<td>Lower Address [6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Attribute Phase Format**

The information delivered in the Attribute Phase is pictured in Figure 17-3 on page 317 and has the following characteristics:

- **Completer ID.** Identifies the originator of the Split Completion transaction for the benefit of a diagnostic tool that may be monitoring transaction flow.
- **Upper and Lower Byte Count fields.** Always set to a byte count of four (because a message always consists of a single dword value).
- **Byte Count Modified (BCM) bit.** Always cleared to zero.
- **Split Completion Message (SCM) and Split Completion Error (SCE) bits.** The SCM bit is always set to one in a Split Completion transaction that contains a message. Refer to Table 17-2 on page 317.
Chapter 17: Split Completion Messages

Figure 17-3: Split Completion Attribute Format

<table>
<thead>
<tr>
<th>Upper Byte Count</th>
<th>Lower Byte Count</th>
<th>AD Bus</th>
<th>Completer Bus Number</th>
<th>Completer Device Number</th>
<th>Completer Function Number</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 24 23</td>
<td>16 15 11 10 8 7 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 17-2: Encoding of SCM and SCE Bits In Attribute Phase

<table>
<thead>
<tr>
<th>SCM</th>
<th>SCE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The transaction does not contain a message. This would be the bit setting for a Split Completion transaction returning previously-requested read data without error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Reserved.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Error-free completion of a split IO or Configuration Write transaction. A message is included in the Data Phase (but the only message currently-defined indicates good completion of the write). Refer to “Good Completion of Split IO or Configuration Write” on page 322.</td>
</tr>
</tbody>
</table>
| 1   | 1   | Error completion of a read or write. A message is included in the Data Phase. Refer to:  
  • “Bad Completion of a Split IO or Configuration Write” on page 323.  
  • “Bad Completion of a Split Burst Memory Read” on page 324.  
  • See “Byte Count Out-of-Range” in Table 17-6 on page 321.  
  • See “Target Abort” in Table 17-5 on page 319.  
  • See “Master Abort” in Table 17-5 on page 319. |

Data Phase Message Format

The format of a message is pictured in Figure 17-4 on page 318 and consists of the following elements:
The Previous Chapter
The previous chapter described the purpose of the Split Completion Message, its format, the messages associated with writes and reads, and device-specific error handling.

This Chapter
This chapter provides a detailed discussion of 64-bit data transfers, 64-bit memory addressing, and additional issues associated with 64-bit devices and 64-bit memory addressing.

The Next Chapter
The next chapter provides a detailed discussion of parity generation, parity checking, Address and Attribute Phase parity, and Data Phase parity.

General Format of Timing Diagram Descriptions
The sections that follow provide a detailed description of various transaction scenarios. Each transaction is described clock-by-clock, with the events during each clock divided into two categories:

- **ON** the rising-edge of clock \( n \).
- **DURING** clock \( n \).

Please keep in mind that some events could actually be correctly placed in either of these event categories.
PCI-X System Architecture

64-bit Data Transfers and 64-bit Addressing: Separate Capabilities

The PCI and PCI-X specifications provide a mechanism that permits a 64-bit initiator to perform 64-bit data transfers with a 64-bit target. At the beginning of a transaction, the 64-bit PCI-X initiator automatically senses if the responding target is a 64-bit or a 32-bit device. If it’s a 64-bit device, up to eight bytes (a qword) may be transferred during each Data Phase. Throughput of 1.06Gbytes/second can be achieved at a bus speed of 133MHz (8 bytes/transfer x 133.33 million transfers/second). If the responding target is a 32-bit device, the initiator automatically senses this and steers all data to or from the target over the lower four data paths (AD[31:0]).

The specifications also define 64-bit memory addressing capability. This capability is only used to address memory targets that reside above the 4GB address boundary. While this capability is optional in the PCI environment, PCI-X requires that all initiators (both 32- and 64-bit initiators) that address memory must be capable of using 64-bit memory addressing. Furthermore, in PCI-X all memory BARs (Base Address Registers; i.e. memory decoders) must be implemented as 64-bit BARs rather than 32-bit BARs. In PCI, they may be either 32- or 64-bits wide.

It is important to note that 64-bit addressing and 64-bit data transfer capability are two features, separate and distinct from each other.

- An initiator that performs any memory transactions is required to support 64-bit memory addressing, while one that never addresses memory (or memory-mapped IO locations) will not support it.
- A target that contains memory (or memory-mapped IO locations) must support 64-bit memory addressing, while one that contains no memory (or memory-mapped IO locations) will not support it.
- An initiator may or may not support the ability to perform 64-bit data transfers.
- A target may or may not support the ability to perform 64-bit data transfers.

64-Bit Extension Signals

In order to support the 64-bit data transfer capability, the PCI and PCI-X buses implement an additional thirty-nine pins:
Chapter 18: 64-Bit Transactions

- **REQ64#** is asserted by a 64-bit initiator to indicate that it would like to perform 64-bit data transfers. REQ64# has the same timing and duration as the FRAME# signal. The REQ64# signal line must be supplied with a pullup resistor on the system board. REQ64# cannot be permitted to float when a 32-bit initiator is performing a transaction.

- **ACK64#** is asserted by a target in response to REQ64# assertion by the initiator (if the target supports 64-bit data transfers). ACK64# has the same timing and duration as DEVSEL# (but ACK64# must not be asserted unless REQ64# is asserted by the initiator). Like REQ64#, the ACK64# signal line must also be supplied with a pullup resistor on the system board. ACK64# cannot be permitted to float when a 32-bit device is the target of a transaction.

- **AD[63:32]** comprise the upper four address/data paths. The system board designer must provide pullup resistors on these signal lines so they will not float during bus Idle time or during transactions that are only using the lower-half of the bus.

- **C/BE#[7:4]** comprise the upper four command/byte enable signals. The system board designer must provide pullup resistors on these signal lines so they will not float during bus Idle time or during transactions that are only using the lower-half of the bus.

- **PAR64** is the parity bit that provides even parity for the upper four AD paths and the upper four C/BE signal lines. The system board designer must provide a pullup resistor on this signal line so it will not float during bus Idle time or during transactions that are only using the lower-half of the bus.

The following sections provide a detailed discussion of 64-bit data transfer and addressing capability.

**REQ64# and ACK64# Have Same Timing As FRAME# and DEVSEL#**

Just as in 64-bit PCI, the initiator always asserts REQ64# and FRAME# together when it starts a 64-bit transaction and deasserts them simultaneously.

Likewise, the target of a 64-bit transaction always asserts and deasserts ACK64# along with DEVSEL# (assuming that it's a 64-bit target and that the initiator asserted RE64#). Refer to Figure 18-7 on page 346. This is also true in the case where a 64-bit initiator connects with a 64-bit target and the target issues a Single Data Phase Disconnect in the first Data Phase (see Figure 16-8 on page 299). The target must deassert ACK64# in the same clock that it deasserts DEVSEL#.
In Attribute Phase, Upper Bus Reserved and Driven High

During the Attribute Phase, the upper half of the bus (AD[63:32] and C/BE#[7:4]) are treated as follows:

- If the initiator is a 64-bit initiator, the upper half of the bus is reserved and must be driven high.
- If the initiator is a 32-bit initiator, the required pullup resistors on the system board maintain these signals in the logic high state.

Block Length Remains the Same

Irrespective of whether the current transaction is a 64- or 32-bit transfer, the block length (i.e., the ADQ length) is always 128 bytes. Obviously then, a block can be transferred in half the number of Data Phases when performing 64-bit transfers.

Bursts Cannot Cross $2^{64}$ Boundary

Memory burst transactions are forbidden to cross the $2^{64}$ address boundary.

REQ64# Not Permitted in Dword Transactions

General

Since the nature of a dword transaction is that it transfers a dword or a subset thereof, it is illegal for an initiator to assert REQ64# when it initiates any of the dword transactions. This includes the Memory Read Dword transaction.

When a 64-bit capable initiator starts a dword transaction, it does not use the upper half of the bus. The required pullups on the system board maintain the upper half of the bus in the logic high state during periods when it is not in use.
Chapter 18: 64-Bit Transactions

…Unless It’s a Split Completion

The Completer can initiate a Split Completion as a 64-bit transfer if the Split Request was a dword transaction. In the Data Phase, the Completer drives the dword of read data or the Split Completion Message onto AD[31:0]. The Lower Address field in the Split Completion address is cleared to zero (see Figure 13-11 on page 210).

MSI Write Always Writes a Single 32-bit Data Value

The PCI 2.2 spec stipulates that the memory write that is performed to write the MSI (Message Signaled Interrupt) data value always consists of a single, 32-bit data value:

- The 16-bit message value is driven onto the lower two data paths, AD[15:0].
- The upper two data paths, AD[31:16], must be driven to zero.
- C/BE#[3:0] are asserted.

For additional information regarding MSI generation in the PCI-X environment, refer to “MSI Feature Optional in PCI Environment” on page 22. For a complete description of the MSI capability, refer to the chapter on interrupts in MindShare’s PCI System Architecture book (Fourth Edition or later; published by Addison-Wesley).

Bridge Must Support DAC on Both Interfaces

Refer to Figure 18-1 on page 332. The 1.1 PCI-to-PCI Bridge Architecture spec states that a PCI-to-PCI bridge must support upstream movement of a memory transaction that uses the DAC command (for a detailed description of the DAC command and 64-bit memory addressing, refer to “Addressing Memory Above 4GB Boundary” on page 372). In the figure, this would be a memory transaction initiated on bus one that targets a memory address above the 4GB address boundary in main memory. In other words, the bridge must recognize the DAC command when it latches the transaction from bus one, and must be capable of re-issuing the transaction on bus zero. However, a PCI-to-PCI bridge does not have to support downstream movement of a memory transaction (from bus zero to bus one).
The Previous Chapter
The previous chapter provided a detailed discussion of 64-bit data transfers, 64-bit memory addressing, and additional issues associated with 64-bit devices and 64-bit memory addressing.

This Chapter
This chapter provides a detailed discussion of parity generation, parity checking, Address and Attribute Phase parity, and Data Phase parity.

The Next Chapter
The next chapter provides a detailed description of Type 0 and Type 1 Configuration transactions, Special Cycle Requests, and the arbiter’s treatment of configuration transactions.

General Discussion of Parity Generation

Parity Generation Is Mandatory
The agent driving the AD bus in any phase (other than the Response Phase) is required to provide correct parity for the information driven on the AD and the C/BE buses in that phase.
In Data Phases of Writes and Split Completions

Initiator Must Generate Correct Data Phase Parity

During a write or a Split Completion transaction, the initiator is required to generate the correct parity for the data and Byte Enables it provides in each Data Phase (including all clocks of the first Data Phase when the target is inserting Wait States in the first data transfer; see next section).

Toggle Data and Parity When Target Inserts Wait State Pairs

When a target inserts Wait States in the initial Data Phase of a burst write, it must always insert Wait States in pairs (see “On Burst Writes, Insert Wait States In Pairs” on page 257). This forces the initiator to toggle between the first and second data item until the target begins to accept the write or Split Completion data. In this case, the initiator must toggle between the parity for the first and second data items as well.

In Reads, the Target Sources Data and Parity

First Data Phase Data and Parity Can Be Delayed

In the first Data Phase of a read transaction, the target may insert Wait States to delay the delivery of the first data item as well as its parity bit. When the target finally drives the first data item and asserts TRDY# to indicate its presence, it must then drive the correct parity one clock later.

Subsequent Data Phase Data and Parity Never Delayed

Each subsequent Data Phase of a read burst transaction is a single clock in duration. The target provides the next data item, followed immediately by its associated parity bit in the next clock.

General Discussion of Parity Checking

Checking Required In Address and Attribute Phases

All devices on the bus is required to check the parity of the Address and Attribute Phases of each transaction.
Chapter 19: Parity Generation and Checking

Parity Checking Is Generally Required In Data Phases

The device receiving the data in each Data Phase is generally required to check the Data Phase parity. As in PCI, some exceptions are acceptable. As an example, the designer of a video frame buffer might decide not to check the integrity of data being written into the buffer. If any data corruption should occur, the only thing affected would be the visual image presented to the end-user.

Target Data Parity Checking During Write or Split Completion

The target of a write or Split Completion transaction must not check data parity while it is inserting Wait States in the first Data Phase. Once it asserts TRDY# to indicate its readiness to start accepting the write or Split Completion data, it then begins checking parity.

Initiator Parity Checking During Reads

Parity Not Checked During First Data Phase Wait States

Since the target has not yet presented the first data item, the initiator may not check its parity.

Parity Checked One Clock After Each Subsequent Data Phase

The device receiving data in each Data Phase checks the associated parity bit on the clock rising-edge that immediately follows the edge on which the data is received.

In Any Phase, Agent Driving AD Bus Supplies Parity

As in PCI, the agent that owns and is driving the AD bus is responsible for providing the parity bit. This means:
PCI-X System Architecture

- In the Address Phase, the initiator drives the address onto the AD bus and the command onto the C/BE bus. The initiator is therefore responsible for providing the parity that covers the address and command.
- In the Attribute Phase, the initiator drives the attributes onto the AD bus and the C/BE bus. The initiator is therefore responsible for providing the parity that covers these attributes.
- In the Response Phase, parity is neither presented nor checked for correctness.
- In the Data Phase, the agent sourcing the data is responsible for providing proper parity for the data it is presenting as well as the Byte Enable setting.
  - In any form of a read transaction (including Interrupt Acknowledge), the agent sourcing the read data is the target. It is therefore also responsible for presenting the parity that covers the read data it is presenting as well as the Byte Enables received from the initiator.
  - In any form of a write transaction (including Special Cycle), the agent sourcing the write data is the initiator. It is therefore also responsible for presenting the parity that covers the write data and the Byte Enables it is driving to the target.

As In PCI, Even Parity Is Used

Just as in PCI, the agent that supplies the parity must set the parity bit either high or low to force an even number of one bits in the 37-bit pattern consisting of AD[31:0], C/BE#[3:0], and PAR.

During 64-bit transfers, the agent driving the upper half of the bus must set the parity bit for the upper half of the bus either high or low to force an even number of one bits in the 37-bit pattern consisting of AD[63:32], C/BE#[7:4], and PAR64.

No Parity On Response Phase

Since no data is transferred during the Response Phase, no device checks for parity errors on the AD and C/BE# buses in the clock following the Attribute Phase.
Chapter 19: Parity Generation and Checking

Address Phase Parity

Address Phase Parity Checking Required

All devices on the bus must check the parity associated with the information driven during the Address Phase (i.e., the address and the command).

When DAC Is Used, Check Both Packets

When an initiator uses the DAC command, it is indicating that it is issuing two address/command packets. All devices must perform a parity check on both address/command packets. For more information on the DAC command, refer to “Addressing Memory Above 4GB Boundary” on page 372.

On Error, SERR# Required

When a function detects a parity error on an Address Phase, the function is required to set the Detected Parity Error bit in its Status register. In addition, if the SERR# Enable bit is set to one in the function’s Command register, the function is also required to assert SERR#. In that case, the function will also set the Signaled System Error bit to one in its Status register (in addition to setting the Detected Parity Error bit).

Error Detected Before Transaction Claimed

A target that uses a decode speed other than A may detect an Address Phase parity error prior to finishing its decode and asserting DEVSEL#. As already stated, the target will set the Detected Parity Error and Signaled System Error bits in its Status register and will assert SERR#. Regarding how the target (assuming that it appears to be addressed by the transaction) terminates the transaction, it has two options:

Option 1. The target may choose not to claim the transaction and just let it end in a Master Abort. The initiator terminates the transaction without establishing a connection and sets the Received Master Abort bit in its Status register. In addition, the initiator generates an interrupt to invoke its driver to check its status.

Option 2. See the next section.
The Previous Chapter

The previous chapter provided a detailed discussion of parity generation, parity checking, Address and Attribute Phase parity, and Data Phase parity.

This Chapter

This chapter provides a detailed description of Type 0 and Type 1 Configuration transactions, Special Cycle Requests, and the arbiter’s treatment of configuration transactions.

The Next Chapter

The next chapter describes the detection of a PCI-X bridge, the determination of a function’s capabilities, how a function’s PCI configuration registers are affected by mode selection, and the implementation and usage of the function’s PCI-X capability register set.

Configuration Software Mechanism Same As PCI

The method software uses to stimulate the Host/PCIX Bridge to generate a PCI-X configuration transaction is identical to that used in the PCI environment. For a detailed description of this mechanism, refer to the chapter entitled Configuration Transactions in the MindShare book entitled PCI System Architecture, Fourth Edition (published by Addison-Wesley).
Configuration Transactions Can Only Flow Downstream

With one exception (refer to the next section), PCIX-to-PCIX bridges will only pass configuration transactions downstream (i.e., away from the processor and main memory). The theory is that configuration registers are only accessed by the software executing on the processor. This means that, in order for a tool (such as the Agilent Technology PCI-X Bus Exerciser) to access configuration registers within functions on buses other than the one it resides on, the tool must be located on a bus upstream from the targeted function.

Special Cycle Request Can Flow Upstream or Downstream

Any Requester is permitted to issue a specially-formatted configuration write transaction that is really a request for a bridge to pass a Special Cycle Request from one bus to another. Bridges are designed to pass this type of requests in either direction. For more information on the Special Cycle Request, refer to “Generation of Special Cycle Under Software Control” on page 450.

Type 0: Access Registers in Function on This Bus

Device Selection

The Type 0 Configuration transaction is used to access the configuration registers within a PCI-X function residing on the bus that the transaction is being performed on. It is being performed either by the Source Bridge for this bus, or by a PCI-X bus exerciser tool (such as the Agilent Technologies PCI-X bus exerciser). Prior to initiating the transaction, the Source Bridge decodes the target device number specified by the programmer (see the Host/PCIX bridge to bus four in Figure 20-2 on page 428). The Source Bridge is always device 0 on the bus. Table 20-1 on page 427 illustrates the decoder’s routing of its IDSEL outputs onto the upper address lines during the Address Phase of the Type 0 Configuration transaction. Figure 20-1 on page 427 illustrates the format of the address generated by the Source Bridge during the transaction’s Address Phase. Figure 20-3 on page 429 illustrates the connection of the upper AD lines to each device. Just as in PCI, the system board designer must ensure that the coupling resistor is located in close proximity to the physical device.
Figure 20-1: Type 0 Configuration Address Format

<table>
<thead>
<tr>
<th>31</th>
<th>16 15</th>
<th>11 10</th>
<th>8 7</th>
<th>2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single 1 bit used for IDSEL</td>
<td>Device Number</td>
<td>Function Number</td>
<td>DW Number</td>
</tr>
</tbody>
</table>

Source bridge decodes Device Number field and places a one on appropriate upper AD line. Used to select (i.e., IDSEL) device.

Not present in Type 0 PCI Config transaction. Target device uses this field to update the Device Number field in its PCI-X Status register. It updates the Bus Number field in its PCI-X Status register from the Secondary Bus Number field provided in the attributes of the Type 0 Config transaction.

Table 20-1: Routing of IDSELS To Upper AD Lines

<table>
<thead>
<tr>
<th>IDSEL for Device Number:</th>
<th>Is always routed to the Device over AD Line:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AD[16]. Note that Device 0 is always the Source Bridge for the bus.</td>
</tr>
<tr>
<td>1</td>
<td>AD[17]</td>
</tr>
<tr>
<td>2</td>
<td>AD[18]</td>
</tr>
<tr>
<td>3</td>
<td>AD[19]</td>
</tr>
<tr>
<td>4</td>
<td>AD[20]</td>
</tr>
<tr>
<td>5</td>
<td>AD[21]</td>
</tr>
<tr>
<td>6</td>
<td>AD[22]</td>
</tr>
<tr>
<td>7</td>
<td>AD[23]</td>
</tr>
<tr>
<td>8</td>
<td>AD[24]</td>
</tr>
<tr>
<td>9</td>
<td>AD[25]</td>
</tr>
<tr>
<td>10</td>
<td>AD[26]</td>
</tr>
</tbody>
</table>
Table 20-1: Routing of IDSELs To Upper AD Lines (Continued)

<table>
<thead>
<tr>
<th>IDSEL for Device Number</th>
<th>Is always routed to the Device over AD Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>AD[27]</td>
</tr>
<tr>
<td>12</td>
<td>AD[28]</td>
</tr>
<tr>
<td>13</td>
<td>AD[29]</td>
</tr>
<tr>
<td>14</td>
<td>AD[30]</td>
</tr>
<tr>
<td>15</td>
<td>AD[31]</td>
</tr>
</tbody>
</table>

Figure 20-2: Each Source Bridge Incorporates a Device-to-IDSEL Decoder
Chapter 20: Configuration Transactions

Figure 20-3: Resistive-Coupling of Upper AD Lines To IDSEL Pins at Each Physical Device
The Previous Chapter
The previous chapter provided a detailed description of Type 0 and Type 1 Configuration transactions, Special Cycle Requests, and the arbiter’s treatment of configuration transactions.

This Chapter
This chapter describes the detection of a PCI-X bridge, the determination of a function’s capabilities, how a function’s PCI configuration registers are affected by mode selection, and the implementation and usage of the function’s PCI-X capability register set.

The Next Chapter
The next chapter describes the bridge’s configuration registers. It identifies the PCI registers that are unaffected by the mode, those that are affected (and how), and describes the bridge’s PCI-X capability register set.

Detecting a PCI-X Capable Bridge

Detecting Presence of PCI-X Capable Bridge/Bus
The presence of a PCI-X capable bridge indicates that the bus on its secondary side is a PCI-X bus. This being the case, the configuration programmer then checks the capabilities of each function present on that bus to ensure the highest possible performance on that bus. Remember that the PCI-X bus is essentially a lowest-common denominator bus: the presence of a less-capable device causes
the bridge to force all devices on the bus to operate using a protocol and frequency that ensures proper operation of the less-capable device.

A PCI-X capable bridge is detected by checking for the presence of the PCI-X Capability register set.

1. Refer to Figure 21-1 on page 454. The programmer checks the state of the Capabilities List bit in the bridge function’s PCI Status register. A one indicates that the function implements one or more New Capability configuration registers sets, while a zero indicates that none are implemented (including the PCI-X Capability register set). A zero would therefore indicate that the function resides within a PCI device rather than a PCI-X capable device, while a one indicates that the function may or may not reside within a PCI-X capable device (see the next step).

2. A one in the Capabilities List bit indicates that the Capabilities Pointer register is implemented (see dword 13, byte 0 in Figure 21-2 on page 455). The dword-aligned value in this register is used by the programmer to select and read the indicated dword within the bridge function’s configuration space.

3. New Capability register sets have the general format indicated in Figure 21-3 on page 456. The first byte indicates which New Capability register set this is, while the second byte either contains a dword-aligned pointer to position of the next New Capability register set, or zero if the function has no additional New Capabilities. As indicated in Table 21-1 on page 456, Capability ID 07h indicates that this is the bridge function’s PCI-X Capability register set. Its format is illustrated in Figure 21-4 on page 457.

Figure 21-1: PCI Status Register

![Figure 21-1: PCI Status Register](image-url)
Chapter 21: Non-Bridge Configuration Registers

Figure 21-2: Configuration Header Type 1 Register Set Template

![Configuration Header Type 1 Register Set Template Diagram]

- **Device ID**
- **Vendor ID**
- **Status Register**
- **Command Register**
- **Class Code**
- **Revision ID**
- **BIST**
- **Header Type**
- **Latency Timer**
- **Cache Line Size**
- **Base Address 0**
- **Base Address 1**
- **Secondary Latency Timer**
- **Subordinate Bus Number**
- **Secondary Bus Number**
- **Primary Bus Number**
- **Secondary Status**
- **I/O Limit**
- **I/O Base**
- **Memory Limit**
- **Memory Base**
- **Prefetchable Memory Limit**
- **Prefetchable Memory Base**
- **Prefetchable Base Upper 32 Bits**
- **Prefetchable Limit Upper 32 Bits**
- **I/O Limit Upper 16 Bits**
- **I/O Base Upper 16 Bits**
- **Reserved**
- **Capability Pointer**
- **Expansion ROM Base Address**
- **Bridge Control**
- **Interrupt Pin**
- **Interrupt Line**

**Doubleword Number (in decimal):**

**Required configuration registers**
PCI-X System Architecture

Figure 21-3: General Format of New Capability Registers Sets

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>Reserved.</td>
</tr>
<tr>
<td>03h</td>
<td><strong>VPD.</strong> Refer to the description in MindShare’s book entitled <em>PCI System Architecture, Fourth Edition</em> (published by Addison-Wesley).</td>
</tr>
<tr>
<td>04h</td>
<td><strong>Slot Identification.</strong> This capability identifies a bridge that provides external expansion capabilities (i.e., an expansion chassis containing add-in card slots). Full documentation of this feature can be found in the revision 1.1 <em>PCI-to-PCI Bridge Architecture Specification</em>. For a detailed description, refer to the topics entitled <em>Introduction To Chassis/Slot Numbering Registers and Chassis and Slot Number Assignment</em> in MindShare’s book entitled <em>PCI System Architecture, Fourth Edition</em> (published by Addison-Wesley).</td>
</tr>
<tr>
<td>05h</td>
<td><strong>Message Signaled Interrupts.</strong> Refer to the topic entitled <em>Message Signaled Interrupts (MSI)</em> in MindShare’s book entitled <em>PCI System Architecture, Fourth Edition</em> (published by Addison-Wesley).</td>
</tr>
<tr>
<td>06h</td>
<td><strong>CompactPCI Hot Swap.</strong> Refer to the topic entitled <em>CompactPCI and PMC</em> in MindShare’s book entitled <em>PCI System Architecture, Fourth Edition</em> (published by Addison-Wesley).</td>
</tr>
<tr>
<td>07h</td>
<td>Reserved in 2.2 PCI spec, but ID 07h was subsequently assigned to PCI-X devices.</td>
</tr>
<tr>
<td>8-255h</td>
<td>Reserved in 2.2 PCI.</td>
</tr>
</tbody>
</table>

7d Reserved in 2.2 PCI spec, but ID 07h was subsequently assigned to PCI-X devices.
Detecting Width of Bridge’s Interfaces

Refer to Figure 21-5 on page 457 and Figure 21-6 on page 458. Software may determine the width of the bridge’s connection to the buses by checking the state of the 64-Bit Device status bit. The programmer may then check the width of each device’s connection to the bridge’s secondary bus to ensure that devices of the appropriate width are installed in the connectors on that bus. For more information, refer to “Detecting Width of PCI-X Functions” on page 461.

Figure 21-5: Bridge’s PCI-X Status Register
The Previous Chapter
The previous chapter described the detection of a PCI-X bridge, the determination of a function’s capabilities, how a function’s PCI configuration registers are affected by mode selection, and the implementation and usage of the function’s PCI-X capability register set.

This Chapter
This chapter describes the bridge’s configuration registers. It identifies the PCI registers that are unaffected by the mode, those that are affected (and how), and describes the bridge’s PCI-X capability register set.

The Next Chapter
The next chapter identifies the load tuning mechanisms available for both non-bridge and bridge functions. These mechanisms permit system software to alter the manner in which devices use the bus, thereby allowing software to smooth the flow of data throughout the system.

Discovering a PCIX-to-PCIX Bridge
This topic is covered in “Detecting a PCI-X Capable Bridge” on page 453.

Many Bridge PCI Configuration Registers Unchanged
Refer to Figure 22-1 on page 489. The implementation and usage of the following bridge PCI Configuration registers does not change no matter what mode (PCI or PCI-X) the bridge’s two interfaces are in:
Some Bridge PCI Configuration Registers Affected By Mode

Refer to Figure 22-1 on page 489. If either interface of the bridge is in PCI-X mode, the following bridge PCI Configuration registers are affected:

- Command register.
- Status register.
- Capability Pointer register.
- Secondary Status register.
- Cache Line Size register.
- Latency Timer and Secondary Latency Timer registers.
- Base Address registers.
- Secondary Bus Number register.
- Prefetchable Memory Base and Prefetchable Memory Limit registers.
- Prefetchable Memory Base and Limit Extension registers.
- Bridge Control register.

The sections that follow provide a description of how each of these registers are affected if either interface of the bridge is in PCI-X mode.
Chapter 22: Bridge Configuration Registers

**Figure 22-1: PCIX-to-PCIX Bridge Configuration Header Registers**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Doubleword Number (in decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Device ID</td>
</tr>
<tr>
<td>01</td>
<td>Vendor ID</td>
</tr>
<tr>
<td>02</td>
<td>Status Register</td>
</tr>
<tr>
<td>03</td>
<td>Command Register</td>
</tr>
<tr>
<td>04</td>
<td>Class Code</td>
</tr>
<tr>
<td>05</td>
<td>Revision ID</td>
</tr>
<tr>
<td>06</td>
<td>BIST</td>
</tr>
<tr>
<td>07</td>
<td>Header Type</td>
</tr>
<tr>
<td>08</td>
<td>Latency Timer</td>
</tr>
<tr>
<td>09</td>
<td>Cache Line Size</td>
</tr>
<tr>
<td>0A</td>
<td>Base Address 0</td>
</tr>
<tr>
<td>0B</td>
<td>Base Address 1</td>
</tr>
<tr>
<td>0C</td>
<td>Secondary Latency Timer</td>
</tr>
<tr>
<td>0D</td>
<td>Subordinate Bus Number</td>
</tr>
<tr>
<td>0E</td>
<td>Secondary Bus Number</td>
</tr>
<tr>
<td>0F</td>
<td>Primary Bus Number</td>
</tr>
<tr>
<td>10</td>
<td>Secondary Status</td>
</tr>
<tr>
<td>11</td>
<td>I/O Limit</td>
</tr>
<tr>
<td>12</td>
<td>I/O Base</td>
</tr>
<tr>
<td>13</td>
<td>Memory Limit</td>
</tr>
<tr>
<td>14</td>
<td>Memory Base</td>
</tr>
<tr>
<td>15</td>
<td>Prefetchable Memory Limit</td>
</tr>
<tr>
<td>16</td>
<td>Prefetchable Memory Base</td>
</tr>
<tr>
<td>17</td>
<td>Prefetchable Base Upper 32 Bits</td>
</tr>
<tr>
<td>18</td>
<td>Prefetchable Limit Upper 32 Bits</td>
</tr>
<tr>
<td>19</td>
<td>I/O Limit Upper 16 Bits</td>
</tr>
<tr>
<td>1A</td>
<td>I/O Base Upper 16 Bits</td>
</tr>
<tr>
<td>1B</td>
<td>Reserved</td>
</tr>
<tr>
<td>1C</td>
<td>Capability Pointer</td>
</tr>
<tr>
<td>1D</td>
<td>Expansion ROM Base Address</td>
</tr>
<tr>
<td>1E</td>
<td>Bridge Control</td>
</tr>
<tr>
<td>1F</td>
<td>Interrupt Pin</td>
</tr>
<tr>
<td>20</td>
<td>Interrupt Line</td>
</tr>
</tbody>
</table>

Required configuration registers
Command Register Affected

The bridge’s Command register controls its operational characteristics on its primary interface. If the bridge’s primary interface has been initialized to PCI-X mode (by the Source Bridge for the primary bus), the functionality of some of the Command register bits are altered. A complete description of the changes can be found in “Command Register Bits Affected by Protocol Mode” on page 467. If the primary interface has been initialized in PCI mode, however, the Command register bits operate exactly as defined by the PCI spec.

Status Register Affected

The bridge’s Status register reflects the status of its operation on its primary interface. If the bridge’s primary interface has been initialized to PCI-X mode (by the Source Bridge for the primary bus), the functionality of some of the Status register bits are altered. A complete description of the changes can be found in “Status Register Bits Affected by Protocol Mode” on page 469. If the primary interface has been initialized in PCI mode, however, the Status register bits operate exactly as defined by the PCI spec.

It should be noted that the Capabilities List bit (bit 4) is set to one in both modes (because the bridge implements at least one new capability, PCI-X). This also means that, in both modes, the Capabilities Pointer register contains the pointer to the first New Capability register set.

Capability Pointer Register Affected

See the previous section.

Secondary Status Register Affected

General

When the bridge’s secondary interface is initialized in PCI-X mode, the following bits are affected:

- Fast Back-to-Back Capable bit.
- Master Data Parity Error bit.
Chapter 22: Bridge Configuration Registers

- Detected Parity Error bit.
- DEVSEL# Timing bit field.

The sections that follow describe how these bits operate differently if the bridge’s secondary interface is in PCI-X mode.

**Fast Back-to-Back Capable Bit Affected**

In PCI-X mode, the Fast Back-to-Back Capable bit is ignored (because there is no such thing as Fast Back-to-Back transactions on a bus operating in PCI-X mode). It can be initialized to any value.

**Master Data Parity Error/Detected Parity Error Bits Affected**

In a read transaction in the PCI environment, the Master Data Parity Error and Detected Parity error bits are normally only set in the Status register of the initiator of the transaction when it receives a corrupted read data item. While this is generally true in PCI-X mode, there is one exception:

- In a Split Completion transaction returning previously-requested read data, the device acting as the target is the Requester that originated the read request or a bridge on the path from the Completer back to the Requester.

When the bridge’s secondary interface is acting as the target of a Split Completion transaction returning previously-requested read data, it checks the parity of each read data item being returned to it by the initiator of the transaction. In the event of a read parity error, the bridge sets the Detected Parity Error and Master Data Parity Error bits in its Secondary Status register.

**DEVSEL# Timing Bit Field**

This bit field always indicates PCI DEVSEL# timing (i.e., Fast, Medium, or Slow assertion of DEVSEL#) no matter which mode the bridge’s secondary interface is operating in. This indicates how quickly the bridge’s secondary interface asserts DEVSEL# when it is acting as the target of a transaction initiated by an initiator on the bridge’s secondary bus.

**Cache Line Size Register Affected**

A bridge interface (primary and/or secondary) operating in PCI-X mode ignores the Cache Line Size configuration register, while an interface operating in PCI mode uses it as defined by the PCI spec:
23 Load Tuning Mechanisms

The Previous Chapter

The previous chapter described the bridge’s configuration registers. It identified the PCI registers that are unaffected by the mode, those that are affected (and how), and described the bridge’s PCI-X capability register set.

This Chapter

This chapter identifies the load tuning mechanisms available for both non-bridge and bridge functions. These mechanisms permit system software to alter the manner in which devices use the bus, thereby allowing software to smooth the flow of data throughout the system.

The Next Chapter

This chapter covers the following bridge-related topics:

- Support for the DAC command.
- Posting of memory writes.
- Split Transaction handling.
- Claiming Split Completions.
- Translating PCI transactions to PCI-X transactions.
- Translating PCI-X transactions to PCI transactions.
- Error handling.
- Buffer size.

Introduction To Load Tuning

In a well-designed, high-end platform, the OS would schedule a utility to run on a periodic basis, the purpose of which is to analyze how well data is flowing
PCI-X System Architecture

throughout the platform. Based on its assessment of the situation, the utility will, if necessary, tune the operational characteristics of the system’s Requesters, Completers, and bridges to make better use of the buses and of the buffers within bridges in order to achieve a smooth flow of data between Requesters and Completers with minimum delay incurred.

This chapter describes the mechanisms available at both the device/function- and bridge-level that may be adjusted to achieve this goal.

Non-Bridge Function Tuning

Information Fields

In a non-bridge function, the programmer has the following read-only register fields available to discover the operational limits of the function:

- The **Designed Max Memory Read Byte Count field** in the function’s PCI-X Status register (see Figure 23-1 on page 517).
- The **Designed Max Cumulative Read Size field** in the function’s PCI-X Status register (see Figure 23-1 on page 517). For more information, refer to “Designed Max Memory Read Byte Count” on page 484.
- The **Designed Max Outstanding Split Transactions field** in the function’s PCI-X Status register (see Figure 23-1 on page 517).

Adjustable Fields/Registers

The programmer has the following mechanisms available to adjust the manner in which the function utilizes the bus:

- The **Max Memory Read Byte Count field** in the PCI-X Command register (see Figure 23-2 on page 517). For more information, refer to “Max Memory Read Byte Count Field” on page 474 and “Adjusting Requester’s Memory Read Transaction Size” on page 471. By adjusting this value, software can adjust the maximum size memory read transaction that the Requester is permitted to initiate. This field is useful in adjusting the amount of a bridge’s internal, Split Completion buffer space that this Requester utilizes.
- The **Max Outstanding Split Transactions field** in the PCI-X Command register (see Figure 23-2 on page 517). For more information, refer to “Adjusting Requester’s Split Transaction Queue Size” on page 471. This field is used to adjust how many outstanding Split Transactions the Requester is...
Chapter 23: Load Tuning Mechanisms

capable of handling. Software may choose to decrease the size of the Requester’s queue because it has a tendency to dominate the Split Transaction queues of one or more Completers or bridges.

- The **Latency Timer** register in the function’s configuration header region (see Figure 23-3 on page 518). By adjusting the timeslice assigned to a Requester or Completer, the software can increase the amount of bus time available to a function (so it can transfer large amounts of data more quickly), or it may decrease the amount of bus time available to a function in order to increase the amount of bus time available to other Requesters or Completers. For more information on the Latency Timer, refer to “How the Initiator Deals With Preemption” on page 143.

![Figure 23-1: PCI-X Status Register In Non-Bridge Function](image)

![Figure 23-2: PCI-X Command Register In Non-Bridge Function](image)
Figure 23-3: Latency Timer Register In Non-Bridge Function
Chapter 23: Load Tuning Mechanisms

Bridge Tuning

Adjusting Usage of Split Completion Buffers

Introduction

For background on the bridge’s efficiency status bits, refer to “Bridge Secondary Interface Efficiency Status Bits” on page 501 and “Bridge Primary Interface Efficiency Status Bits” on page 505. For an introduction to the mechanism used to adjust how the bridge utilizes its internal Split Completion buffers, refer to “Split Transaction Control Registers” on page 506.

Interpreting the Efficiency Bits

On a dynamic, on-going basis the load-tuning utility software performs the following sequence of steps. Table 23-1 on page 521 provides a more detailed description of the interpretation of the efficiency bits and the actions taken if one or both bits should become set.

1. The default setting after reset is Limit = Capacity.
2. Wait an appropriate time interval before checking the state of the efficiency bits associated with each of the bridge’s interfaces. The interval depends upon the rate at which traffic patterns in the system change.
3. Check the Split Request Delayed bit and the Split Completion Overrun bit and adjust the Split Transaction Commitment Limit register as follows:
   • If neither bit is set, the Limit value is good.
   • If the Split Request Delayed bit is set and the Split Completion Overrun bit is not set, the Limit is too low. Increase the Limit.
   • If the Split Request Delayed bit is not set and the Split Completion Overrun bit is set, the Limit is too high. Decrease the Limit.
   • If both bits are set, the Limit is too high. Decrease the Limit.
4. Loop back to step 2.

Additional Spec Comments

The spec also contains the following comments regarding the efficiency bits and the Capacity and Limit fields (in some cases, the author has edited and/or annotated the spec text):
The Previous Chapter
The previous chapter identified the load tuning mechanisms available for both non-bridge and bridge functions. These mechanisms permit system software to alter the manner in which devices use the bus, thereby allowing software to smooth the flow of data throughout the system.

This Chapter
This chapter covers the following bridge-related topics:

- Support for the DAC command.
- Posting of memory writes.
- Split Transaction handling.
- Claiming Split Completions.
- Translating PCI transactions to PCI-X transactions.
- Translating PCI-X transactions to PCI transactions.
- Error handling.
- Buffer size.

The Next Chapter
The next chapter describes the locking mechanism that permits the processor, through the Host/PCIX bridge, to perform a series of transactions to a Completer with the guarantee that no PCI-X Requester upstream of the Completer will be able to access the Completer. In addition, no Requester other than the Host/PCIX bridge (in other words, the processor) will be able to access main memory.
PCI-X System Architecture

Performs Same Function As a PCI-to-PCI Bridge

As is the case in a PCI-to-PCI bridge, a PCIX-to-PCIX bridge is a traffic director between two buses. It acts as the surrogate initiator for the two communities of initiators that reside on its primary and secondary sides. Unless this chapter states differently, its operation is identical to that of a PCI-to-PCI bridge. The bridge contains a set of PCI configuration registers as well as a set of PCI-X Capability registers. Chapter 22, entitled "Bridge Configuration Registers," on page 487, provided a detailed description of the bridge's configuration registers. This included a description of:

- How the bridge’s PCI configuration registers operate when one both interfaces are in PCI-X mode (see “Some Bridge PCI Configuration Registers Affected By Mode” on page 488).
- How the bridge’s PCI-X Capability registers are implemented and how they operate (see “Bridge’s PCI-X Capability Register Set” on page 497.)

This chapter focuses on the differences between PCIX and PCI bridges. If an operational characteristic of the bridge is not covered in this chapter, that means that it is the same as that of a PCI-to-PCI bridge. A complete description of the PCI-to-PCI bridge can be found in the chapter entitled The PCI-to-PCI Bridge in the MindShare book entitled PCI System Architecture, Fourth Edition (published by Addison-Wesley).

Support For DAC Command

Downstream Movement of DAC Optional For PCI Bridge

Refer to Figure 24-1 on page 533. In a PCI system, it would not be uncommon for a Requester on a subordinate bus to initiate a memory access with main memory at an address above the 4GB address boundary. For this reason, PCI-to-PCI bridges are required to support upstream movement of a transaction using the DAC command. Whether or not the bridge supports downstream movement of the DAC command is optional, however (wherein a Requester on a bridge’s primary side is addressing a memory target on the bridge’s secondary side at an address above the 4GB address boundary).
Chapter 24: PCIX-to-PCIX Bridges

PCI-X Bridge Must Support Downstream DAC Movement

Earlier sections of the book (see “Base Address Registers (BARs)” on page 464) highlighted the fact that memory decoders (i.e., Base Address registers) in PCI-X functions must be implemented as 64-bit registers. This gives the configuration software the flexibility of assigning them memory address ranges either below or above the 4GB address boundary.

The configuration software may assign memory addresses above the 4GB address boundary to PCI-X functions that reside on the secondary side of a PCIX-to-PCIX bridge. This being the case, it is mandatory that the bridge must support downstream movement of the DAC command through the bridge. This means that the bridge must implement the extension registers for the Prefetchable Memory Base and Limit registers (see “Prefetchable Base/Limit Extension Registers Affected” on page 495).

Figure 24-1: Example System
Bus Width

The bridge’s primary and secondary bus interfaces may be implemented either as 32- or 64-bit interfaces.

Memory Writes Crossing Bridge Are Always Posted

Just as in a PCI bridge, a PCIX bridge always deals with transactions using the Memory Write, Memory Write Block, and Alias To Memory Write Block that must traverse the bridge by immediately accepting the write data into its posted memory write buffer. From the perspective of the initiator, the data has already been written to the target memory. At a later time, the bridge will arbitrate for ownership of the opposite bus and perform the memory writes and will ensure that it uses the same Sequence IDs that were given to it by the Requesters that originated the memory write transactions.

Other Transactions Crossing Bridge Are Always Split

In a PCI bridge, all transactions other than memory writes that must traverse the bridge are treated as Delayed transactions. When a PCIX bridge interface is operating in PCI-X mode, however, they are treated as Split Transactions.

How the Bridge Claims Split Completions

When one of the bridge’s interfaces latches a Split Completion transaction, it uses its internal bus number registers to decide whether or not to claim the Split Completion and pass it to the opposite bridge interface:

- When the bridge latches a Split Completion transaction on its primary interface, it compares the Bus Number portion of the Requester ID (see Figure 24-2 on page 535) to its Secondary Bus Number and Subordinate Bus Number register values. If the target bus number falls within the range of buses defined by the bridge’s Secondary Bus Number and Subordinate Bus Number registers, the bridge claims the transaction (i.e., it asserts DEVSEL#). It accepts the Split completion data and, at a later time, passes the Split Completion transaction to its opposite interface.
- When the bridge latches a Split Completion transaction on its secondary interface, it compares the Bus Number portion of the Requester ID to its Pri-
Chapter 24: PCIX-to-PCIX Bridges

mary Bus Number register. If it matches, the bridge claims the transaction (i.e., it asserts DEVSEL#). If it doesn’t match the bridge’s Primary Bus Number register and it’s outside the range of buses defined by the bridge’s Secondary Bus Number and Subordinate Bus Number registers, the bridge claims the transaction (because the target bus is not on the downstream side of the bridge). The bridge accepts the Split completion data and, at a later time, passes the Split Completion transaction to its opposite interface.

Figure 24-2: Split Completion Address Phase Information Format

<table>
<thead>
<tr>
<th>C/BE[3:0]#</th>
<th>AD Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 0</td>
<td>31 30 29 28 24 23 16 15 11 10 8 7 6 0</td>
</tr>
<tr>
<td>SC Command</td>
<td>Requester Bus Number</td>
</tr>
</tbody>
</table>

When Bridge Can Use Retry or Disconnect at Next ADB

When Bridge Can Issue a Retry

A bridge interface that is operating in PCI-X mode is permitted to issue a Retry in response to a transaction under the following circumstances:

- The transaction is a memory write and the bridge’s posted memory write buffer is temporarily full.
- The transaction needs to be Split and this bridge interface’s Split Transaction Queue is currently full.
- The bridge has been locked and an attempt is made to cross the bridge by an initiator other than the one that locked the bridge. For more information, refer to Chapter 25, entitled “Locked Transaction Series,” on page 579.
- A corrupted Split Completion (i.e., a Split Completion whose size or address did not match its Split Request, or a corrupt Requester Bus Number field in the Split Completion address caused it to cross the wrong bridge) crossed the bridge some time since the last rising edge of RST#. For more information, refer to “Corrupted Split Completion” on page 562.
The Previous Chapter
The previous chapter covered the following bridge-related topics:

- Support for the DAC command.
- Posting of memory writes.
- Split Transaction handling.
- Claiming Split Completions.
- Translating PCI transactions to PCI-X transactions.
- Translating PCI-X transactions to PCI transactions.
- Error handling.
- Buffer size.

This Chapter
This chapter describes the locking mechanism that permits the processor, through the Host/PCIX bridge, to perform a series of transactions to a Completer with the guarantee that no PCI-X Requester upstream of the Completer will be able to access the Completer. In addition, no Requester other than the Host/PCIX bridge (in other words, the processor) will be able to access main memory.

The Next Chapter
The next chapter provides a detailed description of error handling both for non-bridge and bridge functions.

Definition of Downstream and Upstream
Refer to Figure 25-1 on page 586. As defined earlier in the book, a transaction is moving downstream if it is moving away from the processor(s) and main memory, while a transaction is moving upstream if it is moving towards the processor(s) and main memory.
Basics

Refer to Figure 25-1 on page 586. The locking mechanism was included in the PCI and PCI-X specs to support an operational characteristic of Intel x86 processors. In some circumstances, the Intel processor instructs the Host/PCIX bridge to perform a locked transaction series (the first of which is always a memory read) that targets a Completer on a PCI-X bus. This means that the processor needs to perform a series of two (or more) transactions in the PCI-X realm (i.e., the Completer resides on a bus beyond the Host/PCIX bridge). The Host/PCIX bridge is the only device that is allowed to originate a locked transaction series.

The completion of the first transaction of the locked transaction series (a memory read) locks all of the bridges in the path between the processor and the Completer. The bridges will remain locked until the locked transaction series completes or until a fatal error is detected (Master Abort, Target Abort, or the receipt of a Split Completion Message that contains an error). While the bridges remain locked, they have the following operational characteristics (including the Host/PCIX bridge):

- The bridges’ secondary interfaces have the following operational characteristics:
  - It keeps LOCK# asserted (to keep the target it addressed earlier in the locked state) either until the locked transaction series completes, or a fatal error occurs in a subsequent transaction of the series (Target Abort, Master Abort, or it receives a Split Completion Error Message).
  - It will not accept any transactions moving upstream towards main memory (with the exception of Split Completion transactions).
- The primary interface of any PCIX-to-PCIX bridges in the path to the Completer have the following operational characteristics:
  - It will not accept (i.e., will issue a Retry in response to) any transactions moving downstream from any initiator other than the one that locked it (see the exception stated in the next item).
  - It will accept any Split Completion transactions that are moving downstream and will pass them to its secondary bus.
  - It will remain in the full-lock state until FRAME# and LOCK# are both sampled deasserted on the same rising-edge of the clock on its primary interface.

After the first memory read has been completed and the lock on all bridges along the path to the Completer has been established, the following conditions are in force:
Chapter 25: Locked Transaction Series

- No PCI-X Requesters on any PCI-X bus can access main memory. They will receive a Retry from the Host/PCIX bridge.
- The Host/PCI-X bridge will accept Split Completion transactions moving towards the processor.
- No Requesters (other than the Host/PCIX bridge) residing on intervening buses along the path can access any device that resides on the Completer’s bus.
- The Completer itself is not locked.
- The Completer can be accessed by Requesters that reside on its bus or on any bus downstream of its bus.
- Requesters on each bus can perform unlocked transactions with Completers that reside on the same bus as them.
- PCIX-to-PCIX bridges along the path will accept Split Completion transactions moving in both directions through the bridge.

Only Host/PCIX Bridge Originates Downstream Locked Series

The Host/PCIX bridge is the only device that is permitted to originate a locked transaction series moving in the downstream direction (i.e., away from the processor and main memory). No other device (including PCIX-to-PCIX bridges) ever originates a locked transaction series that addresses a target that resides downstream of the Host/PCIX bridge.

The Host/PCIX bridge only initiates a locked transaction series when instructed to do so by the processor and only passes it to the PCI-X bus when the processor is targeting a memory device that resides beyond the Host/PCIX bridge (in the PCI-X or PCI bus environment).

PCI-X Bridges Only Pass Locked Series Downstream

A PCI-X bridge will pass a locked transaction series from its primary to its secondary side, but never in the opposite direction. This permits the processor, through the auspices of the Host/PCIX bridge, to perform a locked transaction series with any Completer in the system. It does not, however, permit devices on subordinate PCI-X buses to originate locked transaction series targeting main memory (which would require PCI-X bridge support for upstream movement of locked transactions through the bridge):
**PCI-X System Architecture**

- If a PCI-X bridge latches a transaction on its secondary side with LOCK# asserted, it ignores LOCK# (i.e., it does not lock itself) and passes the transaction to its primary side without asserting LOCK# on the primary side.
- If a PCI-X bridge latches a transaction on its primary side with LOCK# asserted, it honors the LOCK# (i.e., it does lock itself) and passes the transaction to its secondary side with LOCK# asserted on the secondary side.

**Only EISA Bridge Originates Upstream Locked Traffic**

Refer to Figure 25-1 on page 586. The only device that is permitted to originate a locked transaction series moving upstream towards main memory is the PCIX-to-EISA bridge. However, the EISA bridge is not permitted to utilize the PCI-X LOCK# signal to lock the PCI-X bus. Rather, it must utilize sideband signaling to establish a lock on the PCI-X bus. In addition, since PCIX-to-PCIX bridges will not pass a locked transaction series from secondary-to-primary side with the lock intact, the EISA bridge must reside on PCI-X bus 0 in order to establish a lock on main memory.

As an example, the EISA bridge could assert a sideband signal to the PCI-X bus arbiter in the Host/PCIX bridge to request exclusive ownership of PCI-X bus 0. The Host/PCIX bridge then asserts a return signal to the EISA bridge indicating that the PCIX/EISA bridge will retain ownership of PCI-X bus 0 until it removes the request. The EISA bridge can then perform its locked transaction series with main memory without fear that another master on bus 0 will gain bus ownership. When it has completed its locked transaction series, the EISA bridge releases the arbiter lock by deasserting the signal it had asserted to the arbiter.

In versions of the PCI spec prior to 2.2, the target memory was recommended to lock a 16 byte block of memory (aligned on an address divisible by 16), but could lock its entire memory address space. This still applies for an upstream locked access to main memory by the EISA bridge. For downstream locked accesses, the locked resources are the PCIX-to-PCIX bridges in the path to the Completer (and the Completer could be the EISA bridge).

**Application Bridge May or May Not Support Locking**

If the system software and the platform hardware design guarantee that a locked transaction series will never cross an application bridge, the bridge would not have to support downstream movement of a locked transaction series through the bridge. If this guarantee does not exist, the bridge must support locking.
Chapter 25: Locked Transaction Series

EISA Bridge Supports LOCK# As Input, Not As Output
The EISA bridge supports LOCK# as an input because the processor may instruct the Host/PCIX bridge to perform a locked transaction series targeting a memory location on the EISA bus. The EISA bridge does not, however, implement LOCK# as an output pin on its PCI-X interface because (as described in “Only EISA Bridge Originates Upstream Locked Traffic” on page 582) it is not permitted to use the PCI-X locking mechanism.

Non-Bridge Devices Ignore LOCK#
Only the following devices implement the PCI-X LOCK# signal:

- The Host/PCIX bridge implements it as an output so it can originate locked transactions series on the PCI-X bus on behalf of the processor.
- A PCIX-to-PCIX bridge implements LOCK# as an input on its primary interface (because it acts as the target of locked transactions initiated either by the Host/PCIX bridge or by the secondary interface of a PCIX-to-PCIX bridge that resides between the Host/PCIX bridge and this bridge).
- A PCIX-to-PCIX bridge implements LOCK# as an output on its secondary interface (so it can pass locked transactions received on its primary side to its secondary side).

Devices other than bridges do not implement the LOCK# signal either as an output or as an input. When a non-bridge device (such as the Completer in Figure 25-1 on page 586) is targeted and instructed to lock itself, it permits the access but does not lock itself against accesses from initiators other than the initiator that locked it.

Sequence of Events
Refer to Figure 25-1 on page 586. As mentioned earlier in this chapter, the locking mechanism was included in the PCI and PCI-X specs to support an operational characteristic of Intel x86 processors. In some circumstances, the Intel processor instructs the Host/PCIX bridge to perform a locked transaction series (the first of which is always a memory read) that targets a Completer on a PCI-X bus. This means that the processor needs to perform a series of two (or more) transactions in the PCI-X realm (i.e., the Completer resides on a bus beyond the Host/PCIX bridge). This following list details the steps involved in performing a locked transaction series with the Completer illustrated in Figure 25-1 on page 586:
26 Error Detection and Handling

The Previous Chapter
The previous chapter described the locking mechanism that permits the processor, through the Host/PCIX bridge, to perform a series of transactions to a Completer with the guarantee that no PCI-X Requester upstream of the Completer will be able to access the Completer. In addition, no Requester other than the Host/PCIX bridge (in other words, the processor) will be able to access main memory.

This Chapter
This chapter provides a detailed description of error handling for non-bridge and bridge functions.

The Next Chapter
The next chapter highlights some of the more important aspects of the electrical portion of the PCI-X spec.

Handling of a Target Abort

Introduction
When an initiator starts a transaction, the target may respond by issuing a Target Abort. This is a fatal error and terminates the transaction. There are three scenarios:
A Requester may have initiated a transfer request. The target that responds to the transaction (either the Completer, or a bridge residing in the path to the Completer) may issue a Target Abort to the Requester. This subject is covered in “Requester Receives Target Abort” on page 604.

In the case of a transfer request that receives a Split Response, the Completer later initiates a Split Completion transaction to return the previously-requested read data or a write completion notice. The target that responds (either the Requester or a bridge in the path back to the Requester) may issue a Target Abort in response to the Completer’s Split Completion transaction. This subject is covered in “Completer Receives Target Abort on Split Completion” on page 606.

A PCIX-to-PCIX bridge may be re-initiating a transfer request that was originated by a Requester residing on the other side of the bridge, and the target that responds may issue a Target Abort to the bridge. This subject is covered in “Handling Target Abort on Other Side of Bridge” on page 638.

A detailed description of Target Abort signaling may be found in “Target Abort” on page 290.

**Requester Receives Target Abort**

In this scenario, the Requester initiated a transfer request and the target that responds to the transaction (either the Completer, or a bridge residing in the path to the Completer) may issue a Target Abort to the Requester. The Target Abort may be issued in any Data Phase of the transaction. This is handled as follows:

- Due to the fatal nature of the error (a Target Abort), the Requester will not repeat the transaction and the Sequence ID is retired and may be reused in a future transaction request.
- The Requester sets the Received Target Abort bit in its PCI configuration Status register (see Figure 26-1 on page 605).
- The target that responded sets the Signaled Target Abort bit in its PCI configuration Status register.
- The Requester then generates an interrupt to invoke a handler to service the error:
  - Assuming that the Requester is capable of generating an interrupt, it will do so, causing an interrupt to the processor. The processor executes the interrupt handler within the Requester’s driver and the handler checks the function’s Status register to determine the reason for the interrupt. When it determines that the Requester received a Target Abort, it clears the Requester’s Received Target Abort status bit. The
Chapter 26: Error Detection and Handling

Handler then checks the Status registers of each function on the Requester’s bus to determine which function delivered the Target Abort. The handler clears the Signaled Target Abort bit in the offending target’s Status register and then reports the problem back to the OS. The action taken by the OS is OS-specific.

- If the Requester is not capable of generating a function-specific interrupt (i.e., it implements neither an interrupt pin nor MSI capability), it will assert SERR# and set the Signaled System Error bit in its status register (if enabled to do so with the SERR# Enable bit in its Command register (see Figure 26-2 on page 606). Upon detecting the assertion of SERR#, the chipset will typically generate a fatal hardware interrupt (e.g., an NMI in an x86 processor environment), causing the processor to start execution of its fatal hardware error handler. The handler code scans the Status registers in all platform functions to determine which function(s) asserted SERR#. When it detects the Signaled System Error status bit set, it determines that the Received Target Abort status bit is also set. It clears the status bits and scans the bus for the offending target (the one with the Signaled Target Abort set to one in its status register).

Figure 26-1: PCI Configuration Status Register
Completer Receives Target Abort on Split Completion

In the case of a transfer request that receives a Split Response, the Completer later initiates a Split Completion transaction to return the previously-requested read data or a write completion notice. The target that responds to the Split Completion (either the Requester or a bridge in the path back to the Requester) may issue a Target Abort in response to the Completer’s Split Completion transaction.

When Split Completion Target Abort Is Permissible

The Requester is permitted to signal Target-Abort in response to a Split Completion only under error conditions in which the integrity of data in the system cannot be guaranteed. An example of such an error condition is a parity error in the Split Completion address. In that case, since the address consists of the bus, device and function number of the targeted Requester as well as its transaction tag, a Split Completion with a corrupted address is either trying to deliver read data or a Split Write completion message to the wrong Requester, or perhaps to
the correct Requester, but for the wrong transaction number (i.e., tag). In this case, just as in the PCI environment, all targets that detect an Address Phase parity error must assert SERR# (if enabled to do so via the SERR# Enable bit in their respective Command registers). Also, targets that assert SERR# must set the Signaled System Error and Detected Parity Error bits in their respective Status registers. In addition, the target (i.e., Requester) that appears to be the target of the Split Completion might also assert DEVSEL# and generate a Target Abort.

The spec states the following regarding the target that appears to be addressed by the corrupted Split Completion address:

“If the device asserts DEVSEL# prior to detecting a parity error in the address or Attribute Phase, the device has the option either to complete the transaction as if no error occurred or to signal Target-Abort (even if the transaction is a Split Completion).”

Regarding this text, the author offers this caveat: as already noted, a corrupted Split Completion address is quite serious. If, in addition to asserting SERR#, the Requester chooses (as noted above) to complete the transaction as if no error occurred (i.e., it accepts all of the Split Completion data), it must discard the data (since it either wasn’t meant for this Requester or is being supplied to fulfill a different request issued by this Requester).

Assuming that the Requester does signal Target Abort to a Split Completion, it is handled as indicated in the following two sections.

**Discard of Completion for a Write or Non-Prefetchable Read**

If the Split Request was a write (either IO or configuration) or if it addressed a memory location that has no read side effects (i.e., Prefetchable memory), the Completer must discard the Split Completion and take no further action (i.e., in the Completer no status bits are set, no interrupt is generated, and SERR# is not asserted). The Completer does not set the Received Target Abort bit in its Status register.

The Requester that issued the Target Abort may or may not take any additional action. If it has no outstanding Split requests, it almost certainly would take no additional action (aside from setting its Signaled System Error status bit if it asserted SERR#). If, for example, it were to generate an interrupt to invoke the handler within its driver, the driver would almost certainly be confused because it had not previously-issued any transaction requests to its Requester.
27 Electrical Issues

The Previous Chapter

The previous chapter provided a detailed description of error handling for non-bridge and bridge functions.

This Chapter

This chapter highlights some of the more important aspects of the electrical portion of the PCI-X spec.

Introduction

It is not the author’s intention to cut-and-paste all of the tables and notes from the electrical portion of the spec into this chapter. Rather, this chapter is intended to highlight some important issues related to the electrical design. The reader is urged to consult the electrical portion of the spec for a detailed description of the electrical aspects of PCI-X system and device design.

LVS Bus

As described in “PCI-X Is a Low-Voltage Swing (LVS) Bus” on page 83, the PCI-X bus runs at relatively high speeds and must be implemented as a 3.3 Volt signaling environment.

Attention To Detail

In any high-speed bus design, the system board designer must pay close attention to trace layout, impedance, crosstalk, and inductance. The device designer must pay close attention to trace layout on add-in cards as well as driver/receiver characteristics.
Most Parameters Tighter Than PCI

Many of the timing and design parameters specified for 66 and 133 MHz PCI-X operation are tighter than those specified for a PCI bus implementation. Rather than attempt to highlight all of those differences here, the reader is urged to study the comparison tables and figures in the electrical portion of the specification. The spec writers listed the 33 MHz PCI, 66 MHz PCI, 66 MHz PCI-X, and 133 MHz PCI-X characteristics in side-by-side comparison tables and also superimposed the PCI-X output high and output low driver I/V curves over the PCI curves.

Maximum Number of Loads and Connectors On Bus

As discussed in “Maximum Reliable Speed Verified By Design and Testing” on page 63, the system designer engages in strenuous testing to establish the maximum speed at which a bus may be reliably run.

Cards Keyed As 3.3 Volt or Universal Cards

PCI-X add-in cards must be keyed as 3.3V cards or as Universal cards. For detail on add-in card keying, refer to the chapter entitled Add-In Cards and Connectors in the MindShare book PCI System Architecture, Fourth Edition (published by Addison-Wesley).

133MHz PCI-X Device Must Support 66MHz PCI

The spec dictates that 133Mhz-capable PCI-X devices must support both 33 and 66 MHz PCI operation. A 66 MHz-capable PXI-X device must support 33 MHz PCI, and may optionally support 66 MHz PCI.

Add-In Card Trace Lengths

The add-in card trace length specifications in the PCI-X spec differ from those in the PCI 2.2 spec in the following manner:

- In the 2.2 PCI spec, no minimum trace length is specified for the signals associated with the lower half of the bus. The PCI-X spec calls out a mini-
mum trace length of 0.75 inches. This was added to give the system board designer a more accurate idea of the minimum trace lengths on each card. To ensure proper operation of the reflective wave bus, this is important when performing trace length budgeting.

- The 2.2 PCI spec called out no minimum trace length and a maximum trace length of 2.0 inches for the 64-bit extension signals on an add-in card. PCI-X adds a minimum trace length of 1.75 inches (to facilitate system board trace length budgeting). It has also extended the maximum trace length to 2.75 inches. While this technically makes PCI-X non-compliant with the PCI spec, it was necessary because many designers found it difficult-to-impossible to meet the 2.0 inch maximum specified in the PCI spec.
- While the PCI spec does not specify a minimum or a maximum for the length of the RST# trace on the add-in card, the PCI-X spec dictates a minimum length of 0.75 inches and a maximum length of 3.0 inches. This was done to ensure that the initialization pattern and REQ64# remain stable for the appropriate time after the rising-edge of RST#.

### Initialization Pattern Setup/Hold Time

#### General

The Source Bridge must ensure that the initialization pattern is stable for a minimum of 10 clock cycles prior to the removal of RST# and that the pattern is held on the bus for no longer than 50ns (it can be any value between 0 and 50ns).

#### Trhff Must Be Taken Into Account

The PCI 2.2 spec dictates that the Source Bridge must ensure that no initiator starts a transaction for a minimum of five clock cycles after the removal of RST#. This time interval is referred to as Trhff (Time from RST# high to first FRAME# assertion) and is guaranteed by designing the arbiter within the Source Bridge so that no initiator is granted ownership during this time interval.

At a clock speed of 133MHz, each clock cycle is 7.5ns in duration. Five clock cycles is therefore 37.5ns, while six clock cycles would be 40ns in duration. If the arbiter is designed to grant bus ownership to a requesting initiator within five or six clocks after RST# is removed, the Source Bridge must make sure that it limits the initialization pattern hold time to less than this period. The following must also be taken into account:
“Any signals in the initialization pattern asserted after the rising edge of RST# must be deasserted no later than two clocks before the first FRAME# and must be floated no later than one clock before FRAME# is asserted.”

IDSEL Routing

General

As discussed in “Device Selection” on page 426, the Source Bridge routes the outputs of its Device decoder (i.e., the IDSEL signals) over the upper AD lines as shown in Table 27-1 on page 654. As shown, the spec dictates that the Source Bridge itself is always Device 0 on its secondary bus. The spec also recommends that in systems with add-in board connectors that route the IDSELS over AD bus, the first four add-in board connectors are recommended to be connected as shown in Table 27-1 on page 654 to minimize the length of the IDSEL traces. It’s interesting to note that this is consistent with the slot numbering defined in the PCI spec. For more information, refer to the section entitled Chassis/Slot Numbering in the MindShare book PCI System Architecture, Fourth Edition (published by Addison-Wesley).

Table 27-1: Routing of IDSELS To Upper AD Lines

<table>
<thead>
<tr>
<th>IDSEL for Device Number</th>
<th>Is always routed to the Device over AD Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AD[16]. Note that Device 0 is always the Source Bridge for the bus.</td>
</tr>
<tr>
<td>1</td>
<td>AD[17]; recommended that it be connected to add-in card connector one.</td>
</tr>
<tr>
<td>2</td>
<td>AD[18]; recommended that it be connected to add-in card connector two.</td>
</tr>
<tr>
<td>3</td>
<td>AD[19]; recommended that it be connected to add-in card connector three.</td>
</tr>
<tr>
<td>4</td>
<td>AD[20]; recommended that it be connected to add-in card connector four.</td>
</tr>
<tr>
<td>5</td>
<td>AD[21]</td>
</tr>
</tbody>
</table>
Chapter 27: Electrical Issues

Table 27-1: Routing of IDSELS To Upper AD Lines (Continued)

<table>
<thead>
<tr>
<th>IDSEL for Device Number:</th>
<th>Is always routed to the Device over AD Line:</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>AD[22]</td>
</tr>
<tr>
<td>7</td>
<td>AD[23]</td>
</tr>
<tr>
<td>8</td>
<td>AD[24]</td>
</tr>
<tr>
<td>9</td>
<td>AD[25]</td>
</tr>
<tr>
<td>10</td>
<td>AD[26]</td>
</tr>
<tr>
<td>11</td>
<td>AD[27]</td>
</tr>
<tr>
<td>12</td>
<td>AD[28]</td>
</tr>
<tr>
<td>13</td>
<td>AD[29]</td>
</tr>
<tr>
<td>14</td>
<td>AD[30]</td>
</tr>
<tr>
<td>15</td>
<td>AD[31]</td>
</tr>
</tbody>
</table>

**IDSEL Series Resistor Value**

The PCI-X spec defines the value of the series resistors that connect the appropriate upper AD trace to the device’s IDSEL pin as 2K Ohms +/- 5%.
Appendix A

Protocol Rules
Protocol Rules

Introduction

This appendix is provided as a convenience for the reader and contains all of the protocol rules listed in the spec. The reader should note that all section references (e.g., “See Section 2.12.1 for dual address cycles”), are references to sections within the PCI-X spec itself.

General Bus Rules

The following rules generally apply to all transactions:

1. As in conventional PCI, the first clock in which FRAME# is asserted is the Address Phase. In the Address Phase, the AD bus contains the starting address (except Split Completion, Interrupt Acknowledge, or Special Cycle) and the C/BE# bus contains the command. (See Section 2.12.1 for dual address cycles.)

2. Except as listed below, the starting address of all transactions is permitted to be aligned to any byte. As in conventional PCI, the starting address of Configuration Read and Configuration Write transactions is aligned to a DWORD boundary. Split Completion transactions use only a partial starting address as described in Section 2.10.3. As in conventional PCI, Interrupt Acknowledge and Special Cycle transactions have no address.

3. The Attribute Phase follows the Address Phase(s). C/BE[3::0]# and AD[31::00] contain the attributes. C/BE[7::4]# and AD[63::32] are reserved and driven high by 64-bit initiators. The attributes include additional information about the transaction.

4. The C/BE# bus is reserved (driven high) the clock after the Attribute Phase.

5. Burst transactions include the byte count in the attributes. The byte count indicates the number of bytes between the first byte of the transaction and the last byte of the Sequence, inclusive.

6. DWORD transactions do not use a byte count.

7. The target Response Phase is one or more clocks after the Attribute Phase and ends when the target asserts DEVSEL#.
8. As in conventional PCI, there are no Data Phases if the target does not assert DEVSEL#, resulting in a Master-Abort. All other transactions have one or more Data Phases following the target Response Phase.

9. As in conventional PCI, transactions using the I/O Read, I/O Write, Configuration Read, Configuration Write, Interrupt Acknowledge, and Special Cycle commands are initiated only as 32-bit transactions (REQ64# deasserted). Memory Read DWORD commands also have the same restriction in PCI-X mode. In PCI-X, the length of all these transactions is limited to one Data Phase. Transactions using the Memory Write, Memory Read Block, Alias to Memory Read Block, Alias to Memory Write Block, and Split Completion are permitted by both 64- and 32-bit initiators and are permitted to have one or more Data Phases, up to the maximum required to satisfy the byte count.

10. As in conventional PCI, data is transferred on any clock in which both IRDY# and TRDY# are asserted.

11. The following rules apply to the use of byte enables:
   a. Byte enables are included in the Requester Attributes for all DWORD transactions. Byte enables are included on the C/BE# bus during the Data Phases of all Memory Write burst transactions. Byte enables further qualify the bytes affected by the transaction. Only bytes for which the byte enable is asserted are affected by the transaction.
   b. The C/BE# bus is reserved and driven high during the single Data Phase of all DWORD transactions and throughout all Data Phases of all burst transactions except Memory Write.
   c. DWORD transactions are permitted to have any combination of byte enables, including no byte enables asserted. See Section 2.3 for restrictions on starting address and byte enables.
   d. Memory Write transactions are permitted to have any combination of byte enables between the starting and ending addresses, inclusive. Byte enables must be deasserted for bytes before the starting address and after the ending address (if those addresses are not aligned to the width of the bus). See Section 2.12.3 for exceptions and additional requirements when a 64-bit initiator addresses a 32-bit target.
   e. The byte count of Memory Write transactions is not adjusted for bytes whose byte enables are deasserted within the transaction. In other words, the byte count is the same whether all or none of the byte enables were asserted.

12. Device state machines must not be confused by target control signals (DEVSEL#, TRDY#, and STOP#) asserting while the bus is Idle (FRAME# and IRDY# both deasserted). (In some systems, the PCI-X initialization pattern appears on the bus when another device is being hot-inserted onto the bus. See Section 6.2.3.2.)
Appendix A: Protocol Rules

13. Like conventional PCI, no device is permitted to drive and receive a bus signal at the same time. (See Section 3.1.)

Initiator Rules

The following rules control the way a device initiates a transaction:

1. As in conventional PCI, a PCI-X initiator begins a transaction by asserting FRAME#. (See Section 2.7.2.1 for differences for configuration transactions.)

2. In most cases, the initiator asserts FRAME# within two clocks after GNT# is asserted and the bus is Idle. If the transaction uses a configuration command, the initiator must assert FRAME# six clocks after GNT# is asserted and the bus is Idle.

3. The initiator asserts and deasserts control signals as follows:
   - a. The initiator asserts FRAME# to signal the start of the transaction. It deasserts FRAME# on the later of the following two conditions:
     - 1) one clock before the last Data Phase
     - 2) two clocks after the target asserts TRDY# (or terminates the transaction in some other way as described in Section 2.11.2)
   - b. Initiator Wait States are not permitted. The initiator asserts IRDY# two clocks after the Attribute Phase. It deasserts it on the later of the following two conditions:
     - 1) one clock after the last Data Phase
     - 2) two clocks after the target asserts TRDY# (or terminates the transaction in some other way as described in Section 2.11.2)
   
4. If no target asserts DEVSEL# on or before the Subtractive decode time, the initiator ends the transaction as a Master-Abort.

5. For write and Split Completion transactions, the initiator must drive data on the AD bus two clocks after the Attribute Phase. If the transaction is a burst with more than one Data Phase, the initiator advances to the second data value two clocks after the target asserts DEVSEL#, in anticipation of the target asserting TRDY#. If the target also inserts Wait States, the initiator must toggle between its first and second data values until the target asserts TRDY# (or terminates the transaction). See Section 2.12.3 for requirements for a 64-bit initiator writing to 32-bit targets.
6. The initiator is required to terminate the transaction when the byte count is satisfied.
7. The initiator is permitted to disconnect a burst transaction (before the byte count is satisfied) only on an ADB. If the initiator intends to disconnect the transaction on the first ADB, and the first ADB is less than four Data Phases from the starting address, the initiator must adjust the byte count to terminate the transaction on that ADB.
8. If a burst transaction would otherwise cross the next ADB, and the target signals Disconnect at Next ADB four Data Phases before an ADB or on the first Data Phase, the initiator deasserts FRAME# two clocks later and disconnects the transaction on the ADB. The initiator treats Disconnect at Next ADB the same as Data Transfer in all other Data Phases.
9. If the transaction has four or more data phases, the initiator floats the C/BE# bus on the clock it deasserts IRDY#. If the transaction has less than four data phases, the initiator floats the C/BE# bus either on the clock it deasserts IRDY# or one clock after that.
10. If the transaction is a write with four or more data phases, the initiator floats the AD bus on the clock it deasserts IRDY#. If the transaction is a write with less than four data phases, the initiator floats the AD bus either on the clock it deasserts IRDY# or one clock after that.
11. The default Latency Timer value for initiators in PCI-X mode is 64. Initiators must disconnect the current transaction on the next ADB if the Latency Timer expires and GNT# is deasserted.

Target Rules

The following rules apply to the way a target responds to a transaction:

1. Memory address ranges (including those assigned through Base Address registers) for all devices must be no smaller than 128 bytes. System configuration software assigns the memory range of each function of each device (that requests Memory Space) to different ranges aligned to ADBs. No two device-functions respond to addresses between the same two adjacent ADBs.
2. The target claims the transaction by asserting DEVSEL# using decodes A, B, C, or Subtractive, as given in Table 2-6.
3. After a target asserts DEVSEL#, it must complete the transaction with one or more Data Phases by signaling one or more of the following: Split Response, Target-Abort, Single Data Phase Disconnect, Wait State, Data Transfer, Retry, or Disconnect at Next ADB. See Table 2-14.
4. The target is not permitted to signal Wait State after the first Data Phase. If the target signals Split Response, Target-Abort, or Retry, the target must do
# Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Order</td>
<td>The byte address is incremented sequentially beginning with the start address of the sequence. For example, Split Completions in the same sequence (i.e., resulting from a single Split Request) must be returned in address order.</td>
</tr>
<tr>
<td>Allowable Disconnect Boundary (ADB)</td>
<td>An address divisible by 128. The initiator and the target are only permitted to disconnect burst transactions on ADBs.</td>
</tr>
<tr>
<td>ADB Delimited Quanta (ADQ)</td>
<td>An ADQ is the amount of data transferred within each ADB. When a burst transaction starts, the byte-specific start address issued may or may not start on an ADB (i.e., an address divisible by 128). In addition, the burst may or may not continue across the next ADB. As an example, assume that a transaction starts at any address within an ADB, crosses two ADB, and completes without transferring all of the data within the next ADB. This transaction includes three ADQs of data.</td>
</tr>
<tr>
<td>Application Bridge</td>
<td>A device that resides on a PCI or a PCI-X bus on one side and connects to a PCI-X bus on its other side, but it implements a Type 00 Configuration Header and a Class code that indicates its function as something other than a bridge.</td>
</tr>
</tbody>
</table>
Attributes and the Attribute Phase

The Attributes consists of the 36-bits of additional transaction information provided by the initiator during the Attribute phase of the transaction. This information is provided on AD[31:0] and C/BE[3:0]#.

The Attribute phase occurs in the clock immediately following the Address phase. C/BE[7:4]# and AD[63:32] are reserved and must be driven high by 64-bit initiators.

Burst Transaction

A burst transaction can be of any length from 1-to-4096 bytes and can be initiated as either 32- or 64-bit transactions.

In each Data Phase of a Memory Write burst transaction, the C/BE signals indicate the bytes to be transferred within the current dword (or quadword, if a 64-bit transaction).

In all other burst transactions, the C/BE signals are reserved and must be driven high by the initiator in all Data Phases.

The burst transaction types are:

- Memory Read Block
- Memory Write Block
- Memory Write
- Alias to Memory Read Block
- Alias to Memory Write Block
- Split Completion

Byte Count

The number of bytes to be transferred in a Sequence. In all burst transactions other than Memory Write, the bytes that are transferred is governed solely by the Byte Count, while the Byte Enables are not used.

In the Memory Write Transaction, the Byte Count defines the overall span of memory locations within which bytes are to be written, while the Byte Enables in each Data Phase identify the bytes to be written within the current dword (or qword, if a 64-bit transfer).
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completer</td>
<td>The device addressed by a transaction request. This is not necessarily the target that responds to the transaction request by asserting DEVSEL#. As an example, the initiator of the transaction request may reside on one bus while the addressed device resides on the other side of a PCI-X bridge. In this case, the completer is the device addressed by the requests, while the target that responds with DEVSEL# to the initiator is the bridge.</td>
</tr>
<tr>
<td>Completer Attributes</td>
<td>The target of a transaction request may respond with a Split Response to indicate that it will complete the transaction at a later time by initiating a Split Completion transaction. In the Attribute Phase of the subsequent Split Completion transaction, the Completer supplies an array of attributes referred to as Completer Attributes. This includes the Completer’s ID, the transfer count, the request completion status, etc.</td>
</tr>
<tr>
<td>Completer ID</td>
<td>The Completer’s bus, device, and function number. This information is embedded within the attributes delivered in the Attribute Phase of each Split Completion transaction.</td>
</tr>
<tr>
<td>Complex device</td>
<td>See “Device Complexity” on page 481, and “Application Bridge” on page 564.</td>
</tr>
<tr>
<td>Configuration Attributes</td>
<td>The attribute information delivered in the Attribute Phase of a Type 0 configuration transaction. This includes the Requester ID, Secondary Bus Number, and the Sequence Tag. If the configuration transaction originated as a Type 1 configuration transaction on the primary side of a PCI-X bridge and was converted to a Type 0 configuration transaction when it was passed to the destination bus, the bridge inserts the number of its secondary bus in the Secondary Bus Number field.</td>
</tr>
</tbody>
</table>
Data Phase completes when the target signals the transfer of a data item or terminate the transaction. Transactions terminated by the target with a Split Response or a Retry have a single Data Phase. A Data Phase may consists of more than one clock cycle if the target inserts Wait States into the Data Phase.

Disconnection Defined as the termination of a burst transaction after some but not all of the byte count has been satisfied. While it is the termination of the current transaction, it does not complete the Sequence.

- Targets can disconnect any transaction after a single Data Phase by signaling Single Data Phase Disconnect
- Targets can also disconnect on any ADB by signaling Disconnect at Next ADB.
- Initiators can disconnect on any ADB four or more Data Phases from the starting address by deasserting FRAME# two clocks before the ADB.

Device A PCI or PCI-X compliant component or card. As defined in the PCI spec, the device may contain one or more functions.

Downstream See “Definition of Downstream and Upstream” on page 579.

Dword Four bytes of data aligned on address divisible by four.
\textbf{Numerics}

133MHz Capable Bit 462
133MHz Capable Status Bit 458, 479, 499, 503
33MHz PCI support 56
64-bit addressing 372
64-bit card in 32-bit connector 335, 336
64-Bit Connection Indication 44
64-bit Data Transfer Capability 339
64-Bit Device Status Bit 457, 461, 479, 499, 503
64-bit extension signal pullups 336
64-Bit Extension Signals 328
64-bit transfers 339
66MHz PCI support 56

\textbf{A}

ACK64\# 329
AD 329
ADB 32
Add-In Card Trace Length 379, 652
Address or Attribute Phase Parity Error 612
Address Phase 100, 160, 178, 198, 316
Address Phase parity 384, 387
Address Phase parity error 290, 621
address pre-drive 438
addressing memory above the 4GB address boundary 372
ADQ 330
Agilent Technologies PCI-X bus exerciser 438
AGP 16, 201, 456
Alias To Memory Read Block Command 96, 169, 277
Alias To Memory Write Block Command 97, 174, 277
Allocation mode 508
Allowable Disconnect Boundary 32
APIC bus 157
Application Bridge 564, 582
arbiter and configuration accesses 449
arbiter design rules 138
Arbitration and locking 588
arbitration, bus 129
Attribute Bits 541
Attribute Phase 100, 160, 178, 198, 316
Attribute Phase in 64-bit transaction 330
Attribute Phase parity 384, 389
Attribute Phase parity error 621
attributes, bridge creation of 541

\textbf{B}

back-to-back locked transaction series 599
BCM 179
bridge Acceptance Rules 567
Bridge Control register 69, 496
bridge Creation of a Split Completion 548
bridge creates the Attributes 541
Bridge Error Class SCMs 549
Bridge Receives a Target Abort on Split Transaction 559
Bridge Receives Target Abort on Memory Write 560
Bridge Receives Target Abort on Split Completion 560
Bridge Signals Target Abort 559
Bridge Tuning 519
Buffer Size 508, 562
Buffer Space 124, 169, 186
Burst Commands 95, 163
bus exerciser 438, 478
bus Idle 67
Bus Master Bit 468
Bus Number 478, 506
Bus Width 534
Byte Count Modified 179
Byte Count Out-of-Range 321, 625
Byte Count Out-of-Range Error 627
Byte Count Satisfaction 280
Byte Enables 149, 152
Byte Transfer Count 542

\textbf{C}

C/BE\# 329
Cache Line Size Register 470, 491
Capabilities List bit 24, 470
Index

Capabilities Pointer register 24, 454
Capability ID 23, 456
Capability Pointer register 490
card keying 652
card, determining presence of 73
clock 14
Command Register 490
commands 90
commands, burst 95
commands, dword 92
CompactPCI 456
Completer 34, 114
Completer Error Class 318, 624
Completer ID 178
configuration accesses and the arbiter 449
Configuration Command Format 206
configuration mechanism 425
Configuration Read 90, 148
Configuration Read and Write commands 93, 153, 234
Configuration Read parity 415
configuration transactions 139
Configuration Write 90, 148
corrupted Split Completion 562

D
DAC command 45, 98, 187, 331, 373, 387, 532
DAC Command and locking 600
Data Error During Split Completion 620
data parity error 621
Data Parity Error In Immediate Read On Destination Bus 550
Data Parity Error On Posted Memory Write 556
Data Parity Error On Split Completion 555
Data Parity Error On Split Write 551
Data Parity Error Recover Enable bit 476
Data Parity Error Recovery Enable bit 391, 397, 613
Data Phase Parity 385, 389, 612
Data Phase Parity Error Recovery 41
Data Phase parity error recovery, pci 21
Data Phase(s) 101, 160
data toggling 257, 369
Decomposing Split Transactions 577
Delayed Completion 16
Delayed Request 16
Delayed Transaction 15, 537, 547
Designed Max Cumulative Read Size field 484, 516
Designed Max Memory Read Byte Count field 484, 516
Designed Max Outstanding Split Transaction field 473
Designed Max Outstanding Split Transactions field 484, 516
Destination Bus Parity Error On Data or SCM 556
Detected Parity Error bit 469, 491
Device Complexity bit 481
device containing multiple initiators 138
Device Number 478, 506
devices on bus, number of 63
Device-Specific Error 322, 325, 626, 629
DEVSEL# 210, 491
Discard of Completion for a Write or Non-Prefetchable Read 607
Discard Timer SERR# Enable bit 496
Discard Timer Status bit 496
Disconnect 182
Disconnect At Next ADB 109, 535
Disconnect at Next ADB 567
Disconnect At Next ADB by initiator 289
Disconnect At Next ADB by target 299
Disconnect Four or More Data Phases From ADB 300
Disconnect Too Close To Block Boundary 301
Disconnect When Start Address Very Close To ADB 302
downstream 332, 426, 507, 532, 579
Dual Address Cycle 373
Dual-Address Cycle command 98, 187, 199
INDEX

dword commands 147

E
Efficiency Status Bits 501, 505, 519
EISA Bridge 582
Enable Relaxed Ordering bit 474
exerciser 438, 478

F
Fast Back-to-Back Capable bit 490
fast back-to-back transactions 46, 133, 468, 496
Flood mode 508
full-lock 585
Function Number 477, 506

G
General Purpose Register 159
GPR 159

H
Halt 159, 162
Hot Swap 456
Hot-Plug 71, 192, 494

I
I/O Read 90, 148
I/O Write 90, 148
Idle 67, 135
IDSEL 139, 427, 436
IDSEL output pins 437
IDSEL Routing 654
IDSEL Series Resistor Value 655
Immediate Completion and locking 589
Immediate Transaction 116
Initialization Pattern 64
Initialization Pattern setup and hold time 67, 653
Initialization Time 80, 191
Initiator Latency 189
initiator-lock state 584
Interrupt Acknowledge 90, 93, 148, 154, 155, 234
interrupt table 154, 155, 159
INTR 155, 157
IO BAR 79, 374, 467
IO Read and Write commands 92, 149
IO Read command 216
IO Write command 224

L
Latency 189
Latency Timer 19, 34, 143, 463, 469, 492, 517, 525
linear addressing 96, 166
load-tuning 41, 471, 515
LOCK# 580
locked transaction series 203, 580
locked transaction series, last transaction of 598
LT 19, 143, 144
LVS bus 83, 651

M
M66EN 56
Machine Check 22
Master Abort 105, 161, 290, 319, 375, 493, 499, 557, 580, 589
Master Abort On Destination Bus 557
Master Abort On Other Side of a Bridge 629
Master Abort On Posted Memory Write 558
Master Abort On Split Completion 559
Master Data Parity Error bit 469, 490
Max Memory Read Byte Count field 80, 516
Max Outstanding Split Transactions field 473, 516
Maximum Completion Limit 194, 482
Maximum Memory Read Byte Count field 473, 474
Maximum Memory Write Completion Time 194
Maximum Number of Loads and Connectors 652
Memory Addressing 165
memory BAR 26, 45, 374, 492
memory BAR, minimum size 79
Memory BARs 464
Memory Read 14
Index

Memory Read Block command 96, 167, 241, 465
Memory Read Dword and IO Format 205
Memory Read Dword command 92, 152, 216, 465
Memory Read Line 14, 492
Memory Read Multiple 14, 492
Memory Write and Invalidate 14, 172, 468
Memory Write Block command 97, 169, 253, 314
Memory Write command, PCI-X 97, 174
Memory Write, PCI 14
Memory Write, PCI-X 90, 164, 260, 314
Memory Write-and Invalidate 492
Memory writes are posted 117, 534
memory writes, bridge handling of 110
message 95, 159
Message Class 318
Message Index 318
Message Signaled Interrupt 22, 44, 331, 456
messages, Special Cycle 162
Min_Gnt register 19
Minimum Memory Range 465
MSI 22, 44, 331, 456
New Capabilities 23
New Capability configuration registers 454
NMI 22
No Snoop 201
Non-Bridge Function Tuning 516
Phase 100
Parity Checking 384
Parity Error in Immediate Read 397
Parity Error In Split Completion Address Phase 388
Parity Generation 383
Parity in Configuration Write 420
Parity in IO Write 416
Parking 141
PCI Chipset 612
PCI-X Bridge (Primary Interface) Status Register 503
PCI-X Bridge Error Class 318, 624
PCI-X bus exerciser 438, 478
PCI-X Capability register set 454
PCI-X Capability Register Set Format 497
PCI-X Command Register 472
PCI-X Secondary Status Register 498
PCI-X Status Register 476
PCIXCAP 30, 58, 78
PERR# 42
phase-locked loop 30
phases, transaction 100
PLLs 30, 66
Posted Memory Write Buffer Size 562
power management 23, 44, 81, 456
power management of Legacy PCI Devices 23
power management register set 26
PowerPC 159
Preemption 143
preemption 19
Prefetchable Base/Limit Registers 495
Prefetchable Memory 464
PReP 159
Primary Discard Timer bit 496
pullup resistors 336
quiesce 494
Registered bus, PCI-X is 84
registered nature of the PCI-X bus 390

686
Relaxed Ordering 201, 474, 549, 569
Remaining Byte Count 179, 318, 324
Remaining Lower Address field 318, 324
REQ64# 329, 338
Requester 34, 114
Requester ID 35, 114, 542
resistively-coupled IDSELs necessitate stepping 438
Response Phase 101, 210, 386
Retry 15, 17, 107, 182, 202, 294, 535, 567, 588
RO 201, 474
ROM Shadowing 192
RST# 68, 494
RST# and REQ64# 338
S
SAC 373
SCE 180, 315, 622
SCM 119, 180, 183, 313, 315, 622
SCM Error On a Split Read 561
SCM On Split Write 561
Secondary Bus Number Register 493
Secondary Bus Reset 69
Secondary Clock Frequency field 459, 502
Secondary Discard Timer bit 496
Secondary Latency Timer 525
Secondary Status Register 490
Sequence 114
Sequence ID 34
SERR# 290, 646
Shutdown 159, 162
simple device 481, 565
Single Address Command 373
Single Data Phase Disconnect 32, 108, 297
Slot Identification 456
Snoops 16, 17, 44, 202
South Bridge 155
Special Cycle 90, 95, 105, 148, 159, 235
Special Cycle generation under software control 450
Special Cycle or Interrupt Acknowledge Format 206

Special Cycle Request 426, 451
Split Completion 118, 274
Split Completion and Byte Enables 182
Split Completion and Master Abort 182
Split Completion and Target Abort 182
Split Completion Buffers 519
Split Completion command 97, 176
Split Completion command format 209
Split Completion Data Buffer Size 563
Split Completion Data Buffers 507
Split Completion Discarded 480
Split Completion Discarded Bit 499, 504
Split Completion Error 180, 315, 622
Split Completion error message 276, 586, 627
Split Completion Message 119, 180, 184, 313, 580
Split Completion Message Format 316
Split Completion Overrun Bit 501, 505
Split Completion parity 394
Split Completion, bridge claiming of 534
Split Read Completion 323
Split Read Errors 620
Split Read Straddling Device Boundary 627
Split Request Delayed Bit 502, 506
Split Request Timeout 125
Split Response 109, 118, 308, 314, 599
Split Response and Locking 592
Split Transaction 36, 118
Split Transaction Capacity register 506
Split Transaction Commitment Limit register 506
Split Transaction Control registers 506
Split Transaction Effects On Transaction Ordering 571
Split Transaction Queue 109, 176, 471
Split Write Completion 322
Split Write Data Parity Error 321, 625, 629
Split Write Data Parity Error On Other Side of Bridge 644
Start Address 95, 149, 152, 165, 340
Index

Status register 463, 490
Stepping 27, 45, 129
Stepping Control bit 468

TA# 159
Tag 34, 114, 500, 505, 542
Target Abort 107, 290, 320, 499, 580, 589, 603
Target Abort on Other Side of Bridge 638
Target Abort on Split Completion 606
Target Latency 191
target-lock state 584
termination, initiator 104, 280
termination, target 106, 290
timeslice 19
toggling data 257, 369, 384
Trace Length 379, 652
Transaction of Four or More Data Phases 281
Transaction of Less Than Four Data Phases 282
Transfer Acknowledge 159
Transfer Length 95, 166
translating PCI To PCI-X 536
Translating PCI-X to PCI 543
Trhfa 80, 191
Trhff 68, 653
Type 0 Configuration Read 439
Type 0 Configuration transaction 206, 207, 426, 436
Type 0 Configuration Write 430, 443, 477
Type 1 Configuration access 430
Type 1 configuration access 206, 207, 447

Unexpected Split Completion Bit 480, 500, 504, 626
Unlocking of a Target Bridge 599
upstream 331, 426, 507, 579

vector 154
Virtual Buffer Space mode 508
VPD 456

W
Wait State pairs 257, 384
Wait States 13, 31
width of a PCI-X function’s connection 332
Write Completion Class 318, 624
Write Data Parity error 320

V