

Agilent E2929A/B Opt. 200 PCI-X Performance
Optimizer

User's Guide



Agilent Technologies



Important Notice

All information in this document is valid for both Agilent E2929A and Agilent E2929B testcards.

Copyright

© 2001 Agilent Technologies. All rights reserved.

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies Inc. as governed by United States and international copyright laws.

Author: Anja Schauer, t3 medien GmbH

Notice

The material contained in this document is subject to change without notice. Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Contents

Documentation Overview	7
<hr/>	
PCI-X Performance Optimizer Overview	9
<hr/>	
The Process of PCI-X Design	10
The Approach of the Performance Optimizer	11
The PCI-X Performance Optimizer's User Interface	13
Rules for Proper PCI-X Design	15
Running a Sample PCI-X Performance Optimizer Session	17
<hr/>	
Test Example	18
Loading the Setup File	19
Specifying the Completer Identification	20
Specifying the Requester Identification	21
Determining the Requester ID Number	22
Specifying the Requester-Completer Pair	23
Specifying the Data Capture	24
Viewing the Test Results	25
Setting Up a Real-Time Performance Measurement Test	27
<hr/>	
Predefined Performance Measures	28
How to Select a Testcard for the Measurement	29
How to Set Up the Testcard for the Measurement	29
How to Select Predefined Performance Measures	32
How to Set up Run Options	33
How to Run a Performance Measurement	34

Setting Up a PCI-X Performance Optimizer Test	37
Setting Up the Test Hardware	38
Requester Identification	39
Completer Identification	39
Preparing the Software for the Test	40
Setting up the Completer Identification	41
Setting up the Requester Identification	42
Selecting a Requester-Completer Pair	43
Setting Up the Data Capture	44
Measuring System or Device Performance	47
Running the Performance Measurements	48
Interpreting the Result Charts	49
PCI-X Usage	50
Burst Usage	51
Command	52
Latency	53
Interpreting the Report Output	54
Using the Report Output for Performance Evaluation	55
General Information on the System under Test	56
Analyzing the PCI-X Throughput	57
Analyzing the PCI-X Utilization	58
Analyzing the PCI-X Efficiency	59
Analyzing the Split Behavior	60
The Bus Users Overview	61
Summary for the Performance Evaluation Example	62
Using the Report Output for Performance Optimization	62
Analyzing the Bus Utilization	63
Analyzing the Traffic Overhead	64
Analyzing the Byte Enable and Burst Behavior	65
Analyzing the Bursts	66
Analyzing Command Termination	69
Summary of the Performance Optimization Example	71

Reusing the Test Setups and Results	73
Printing and Exporting Test Results	74
Printing the Test Results	74
Exporting the Report	75
Exporting the Trace Memory	75
Saving the Test Settings	76
Reusing Previously Saved Data	76
Going Further Into Details	79
Verifying Good Performance	79
Performance in the Presented Charts	80
Identifying Design Rule Violations	81
Replacing Cards or Improving Behavior	82
Improving Completer Behavior	83
Minimize First Word Latencies	83
Command Termination	84
Report Reference	87
Typical PCI-X Transactions and Sequences	88
Example Transfers	90
Bus Statistics (Report Sections 1 to 7)	92
General Information (Report Section 1)	93
Test Base (Report Subsection 1.1)	93
Statistical Base (Report Subsection 1.2)	94
Invalid Measures (Report Subsection 1.3)	96
Basic Bus Statistics (Report Section 2)	97
Bus Throughput Statistics (Report Section 3)	98
Efficiency Statistics (Report Section 4)	99
Bus Utilization Statistics (Report Section 5)	101
Bus Users Overview (Report Section 6)	102
Split Transaction Statistics (Report Section 7)	103

Requester-Completer Pair Measurements (Report Section 8)	104
The Header of Report Section 8	105
Statistical Basis (Report Subsection 8.1)	105
Bus Utilization (Report Subsection 8.2)	106
Data Phase (Report Subsection 8.2.1)	107
Overhead (Report Subsection 8.2.2)	109
Command Usage, Observed per Sequence (Report Subsection 8.2.3)	111
Command Termination, Observed per Transactions (Report Subsection 8.2.4)	112
Byte Count over Command, Observed per Sequence (Report Subsection 8.2.5)	114
Byte Count over Command, Observed per Transaction (Report Subsection 8.2.6)	116
Efficiency Statistics (Report Subsection 8.3)	117
Efficiency over Byte Count, observed per Sequence (Report Subsection 8.3.1)	118
Termination Statistics (Report Subsection 8.4)	119
Split Statistics (Report Subsection 8.4.1)	119
Termination Burst Histogram, Observed per Transaction (Report Subsection 8.4.2)	120
Latency Histogram, Observed per Sequence (Report Subsection 8.5)	121
Top Ten List of First Word Latencies (Report Subsection 8.5.1)	122
Definitions of Used Measures	125

Documentation Overview

This section shows you the different types of documents offered by Agilent Technologies and gives you an overview of which documents are available when you work with the Agilent E2929A/B PCI-X Exerciser and Analyzer.

All documents are valid for both Agilent E2929A and Agilent E2929B testcards. The following documents are available:

Getting Started Guide

- **Getting Started Guide**

Introduces standard analysis features and provides an example of how to set up the protocol observer.

This guide also gives detailed information about the hardware and interfaces.

User's Guides

- **Agilent E2929A/B Opt. 300 PCI-X Exerciser User's Guide**

Provides information on programming the testcard as an initiator and/or target device. It shows you how to actively stimulate the PCI-X bus.

This guide shows how to:

- Initiate data transfers on the PCI-X bus (act as requester-initiator).
- Act as completer-target.
- Handle split completion transactions (act as completer-initiator).
- Handle open requests (act as requester-target).

- **Agilent E2929A/B Opt. 100 PCI-X Analyzer User's Guide**

Provides information on how to examine the behavior of a PCI-X device on the bus and shows how to perform functional tests such as data compares.

- **Agilent E2929A/B Opt. 200 PCI-X Performance Optimizer User's Guide**

Provides all features that are needed to evaluate and optimize any device under test in terms of the performance.

GUI and C-API/PPR References

- **Agilent E2929A/B Windows and Dialog Boxes Reference**

Provides reference information on all windows and dialog boxes of the Agilent E2920 graphical user interface (GUI).

- **Agilent E2929A/B Opt. 320 C-API/PPR Reference**

Describes all C functions, types and definitions of the application programming interface of the Agilent E2929A/B PCI-X testcard.

This reference also provides the commands and abbreviations that are used in the command line interface (CLI) of the graphical user interface.



PCI-X Performance Optimizer Overview

The Agilent E2929A/B Opt. 200 PCI-X Performance Optimizer is an additional option available for the Agilent E2920 verification tools series. Its complete integration into the framework lets it be used directly with a PCI-X Exerciser and Analyzer testcard and its graphical user interface (GUI).

The Performance Optimizer provides all the features that are needed to evaluate and optimize any device under test in terms of the performance. This includes the devices on the PCI-X bus as well as the complete PCI-X system. For this purpose, the software defines and calculates performance measures such as efficiency, data throughput, bus utilization or split behavior, which allow you to compare and communicate the test results.

With these features, you can easily reveal which devices cause lower performance in the system under test. Furthermore, you can examine the data traffic of every device in detail to track down the exact cause for the unsatisfactory performance.

With the information provided by the software, you can improve your system either by selecting components with better performance or by debugging or redesigning the affected devices.

The Process of PCI-X Design

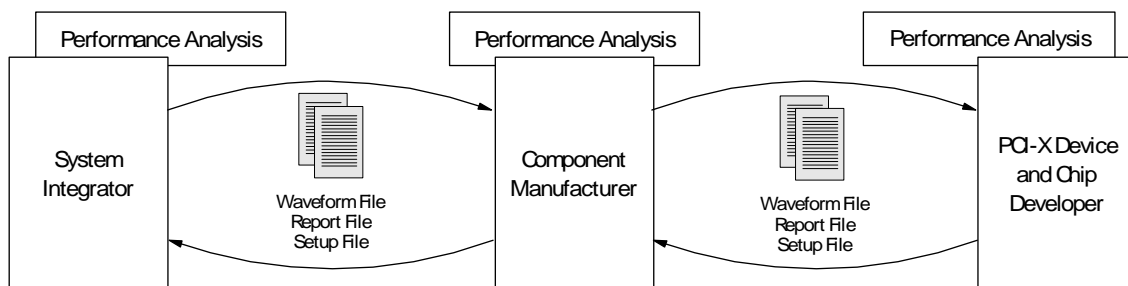
Due to the complex PCI-X protocol, there are many ways to improve PCI-X performance, both on a component level as well as on a system level.

Even when all the rules included in the PCI-X specifications are followed, the performance of your device still depends on many factors.

The PCI-X performance optimizer is a powerful tool in the following cases:

- you are designing a chip or an add-in card,
- you are integrating a system and you need to select between different components on the market,
- you are adjusting system and device parameters in order to optimize the overall performance.

For all imaginable tasks, it is essential to evaluate the quality of the different devices in your system under test. Furthermore, it is important to document your results and to communicate them to colleagues or other companies.



These facts raise the question for a powerful, yet easy-to-use solution to find the bottlenecks in your system. If you manage to identify the part that decreases the system performance due to its poor design, you know what to improve or replace. Eventually, this process will lead to significantly more powerful PCI-X systems.

The Approach of the Performance Optimizer

Methods for Performance Analysis

Basically, there are two different approaches to analyze a system under test. Which approach should be used depends on the particular case. With the Agilent E2929A/B Performance Optimizer, you can employ both approaches for performance analysis:

- Real-Time Measurements

Real-time performance analysis based on programmable counters allows long-term observation of the system's properties like latencies, etc. This method provides valuable information over long time periods about what the general performance of your system is. It is limited, however, in its ability to provide meaningful information to track down the root cause of performance issues.

- Post-Processed Analysis

The post-processed analysis is based on one or more recorded traffic traces in the trace memory. This allows a detailed analysis of all performance aspects like bus utilization, command usage, efficiency, or split statistics.

The observation time, however, is limited due to the large amount of data that is stored.

Recording Traffic Samples

The traffic is recorded in the trace memory of the PCI-X Exerciser and Analyzer card. A set of values in the trace memory is referred to as a traffic sample. One traffic sample can be recorded for every clock cycle. However, if a sequence of clock cycles occurs on the bus without any changes in the relevant bits, for example, idle states, they are stored as one sample. This behavior extends the possible observation time and, thus, yields better measurement results based on a larger statistical base.

Trace Memory Size

The trace memory of the Agilent E2929A/B testcard can store up to 2M traffic samples. One sample here is referred to as a sequence of clock cycles without any changes in the bits that are significant for the performance.

The Steps of Performance Optimization

The Agilent E2929A/B PCI-X Performance Optimizer performs the data analysis by taking the following steps:

- Recording the bus traffic

The measurement is either started by the user or can be set up to be triggered when a certain event occurs on the bus. The trace memory then records the data until the trace memory is full or until another specified bus event occurs on the bus.

Usually, the observation time is only a split-second. Thus, the memory content is a “snapshot” of the bus activities.

- Analyzing the recorded data

For the analysis, the trace memory content is loaded from the testcard into the PCI-X Performance Optimizer. The software then derives a large number of result values from the sampled bus activities.

- Presenting the test results

The Performance Optimizer outputs the results of the analysis in graphical charts and in a textual report. This report is arranged according to the requirements of system and device evaluation and optimization.

These test results give a detailed look into your system’s and devices’ traffic behavior. You can identify the devices with poor performance and get detailed hints for improvement.

- Generating result files (setup file, report file, and waveform file)

Optionally, you can store your test settings and test results in files. With these files, you can reuse the settings or the data from previous settings. They also provide a good method for analyzing the devices in workgroups. The report file holds the results of the post-processed analysis in textual form, the waveform file holds the trace memory contents. The setup file with the test settings can be used to reproduce the test environment.

The PCI-X Performance Optimizer's User Interface

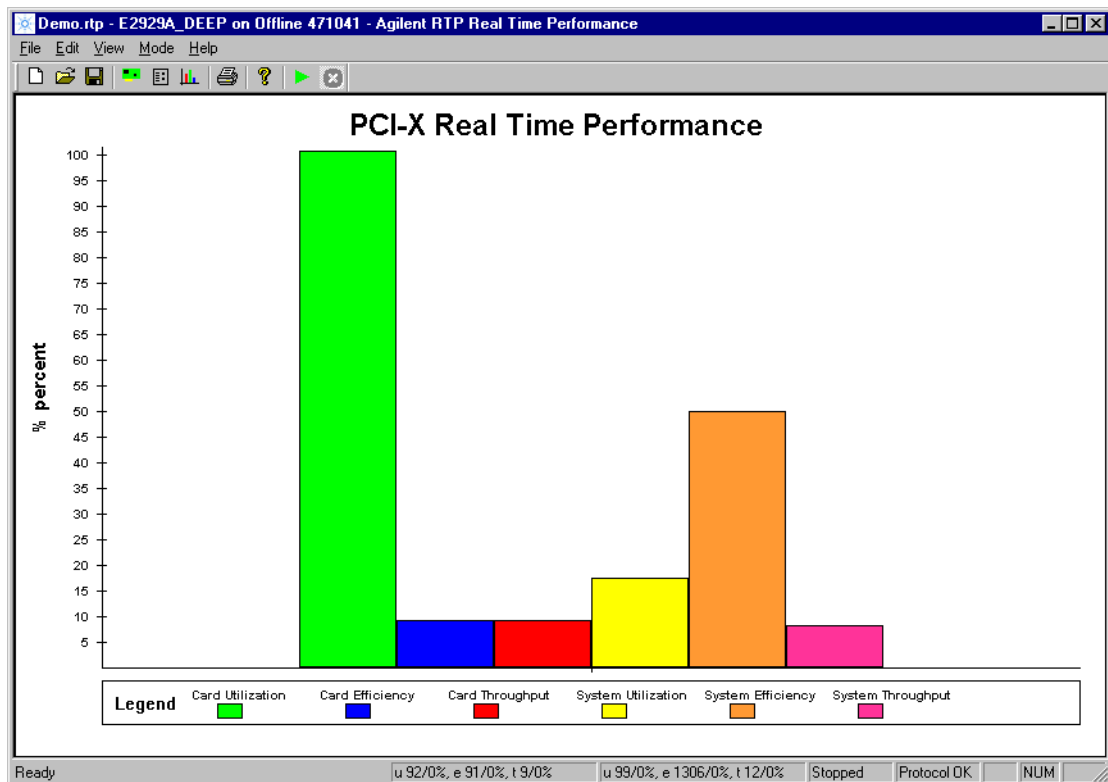
The Agilent E2920 Series software provides two different User Interfaces for real-time performance and post-processed measurements.

Real-Time Performance Analysis

The User Interface for performing real-time performance analysis is available via a separate GUI, which can be accessed by selecting:

Start > Programs > Agilent E2920 PCI-X > PCI-X Real Time Performance GUI

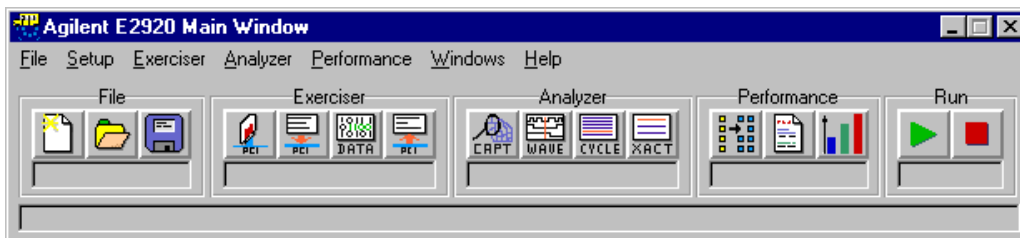
The user interface for the real-time performance measurements looks as follows:



Navigation The most important features of the real-time performance measurement are available via the buttons in the tool bar. All these features and more are available via the menus, as well.

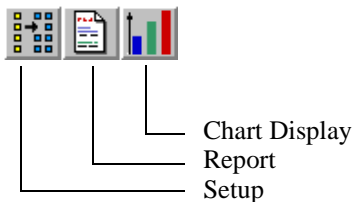
Post-Processed Performance Analysis

The User Interface for performing post-processed performance analysis is fully integrated into the framework of the Agilent E2929A/B Exerciser and Analyzer. All its windows and controls appear as additional features of the GUI that is assumed to be known from the PCI-X Exerciser and Analyzer software.



Navigation The most important features of each part of the framework are available via the buttons in the respective button group. All these features and more are available via the menus of the main window, as well.

Performance Button Group You can quickly open the windows providing the features of the PCI-X Performance Optimizer with the Performance buttons.



Performance Windows Within the PCI-X Performance Optimizer, the following windows are provided for performance analysis and optimization:

- **Performance Setup**
In the Performance Setup window, you can set up the contents of the report to be generated, identify the requester and completer devices to be considered in the test, and control the data capture for the performance analysis.
- **Performance Report**
The performance report lists all results in a hierarchical way, so that you can focus on different levels of detail.
- **Performance Charts**
The performance charts show results of the analysis of the captured PCI-X traffic.

Rules for Proper PCI-X Design

When designing PCI-X devices, a lot of design rules need to be obeyed to avoid device conflicts. Furthermore, there are rules that need to be followed to improve the communication abilities of the devices.

Designing the devices so that they behave according to these rules ensures that the device will not only result in good performance in a test environment, but on all platforms and under most circumstances.

Even when all these rules have been followed, the design with the best performance still depends on the type of the device and its dedicated tasks. Hence, a fair amount of design experience is likely to yield better results.

This document can only name a few rules that are essential for good performance:

- Use long bursts.

On most platforms, read bursts should have a minimum length of 64 dwords for reasonable performance. Write bursts should have a minimum length of one cacheline.

- Use memory commands, avoid I/O commands.

I/O commands can stress the processor. Although the PCI-X bus performance might not be directly affected, the use of I/O commands reduces the performance of the whole system by using CPU time.

- Use split transactions to relieve the bus.

If the target signals a split response, the bus can be used by other initiators to transfer data.

These are only a few basic rules that should always be respected. More hints for identifying poor designs and for improving the performance are given in *“Going Further Into Details” on page 79*.

Running a Sample PCI-X Performance Optimizer Session

The following sections show how to set up a performance measurement with the Agilent E2929A/B Performance Optimizer Graphical User Interface (GUI), by means of an example.

The recommended approach is:

1. Specifying the identification of the devices under test

In detail that means:

- Identify the completer devices by its address ranges.
- Identify the requester devices by its bus, device and function numbers.
- Select the device pair, the test is to be focussed on.

2. Setting up the data capture

Define the start and the duration of the test.

3. Viewing the test results

Analyze the data traffic that is recorded in the trace memory after the test has finished.

Test Example

In a PCI-X bus system with 64-bit bus width and 33-MHz bus speed, both the performance of the whole system and of a particular requester-completer pair is to be evaluated.

Requester Identification The requester device named *MyRequester* has the following identification:

- bus number: ab\h
- device number: 1\h
- function number: 0\h

Completer Identification The completer device named *MyCompleter* has the following identification:

- base address: 40000\h
- size: 5000\h

The performance measurement should not be restricted on any commands.

Requester-Completer Pair The performance measurement has to focus on the traffic between *MyRequester* and *MyCompleter*. All types of commands are of interest for the measurement.

Data Capture The data capture has to start immediately and stop after 200000 samples have been recorded.

Viewing the Results When you view the results in the performance charts, you get quick access to the performance of your system and your device under test.

Loading the Setup File

When preparing the Agilent E2929A/B Performance Optimizer for a test session, you do not necessarily need to load a setup file. The setup file for the example measurements, however, is included in the software package.

To set up your application for the example test, follow the instructions below:

- 1 Select *Load* from the *File* menu in the main window.
- 2 Select the file *SampleSession.bst*.
- 3 Click the *Load* button.

The setup file assumes an Agilent E2929A/B_Deep testcard is to be used for the test—at least in offline/demo mode. If this is not the case, you will be informed in a message box. Change the configuration to the testcard required for the example.

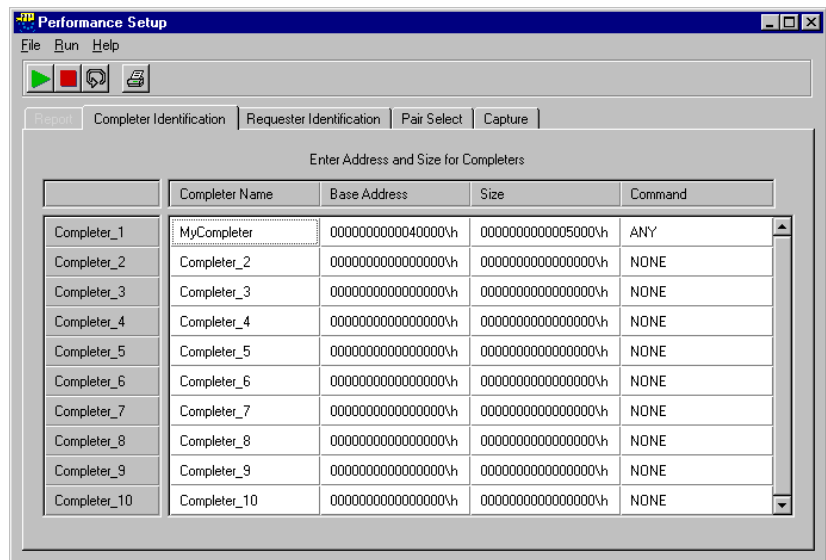
After you have loaded the setup file, the test measurements can be started. However, we need to verify the correctness of the settings. Therefore, we go through the setup step-by-step and make changes where applicable.

Specifying the Completer Identification

In order to enable the Agilent E2929A/B PCI-X Performance Optimizer to identify particular devices communicating on the PCI-X bus, you need to specify them in the setup. Requesters are identified by their bus, device and function numbers, completers by their address range.

To specify the completer:

- 1 Select the *Performance Setup* item from the *Performance* menu.
The Performance Setup window opens, displaying the *Completer Identification* tab.
- 2 Enter a name, the base address, the size and the commands considered in the measurements for a particular completer.
You can specify up to ten different completer devices.
In this example, *MyCompleter* is specified as shown below.

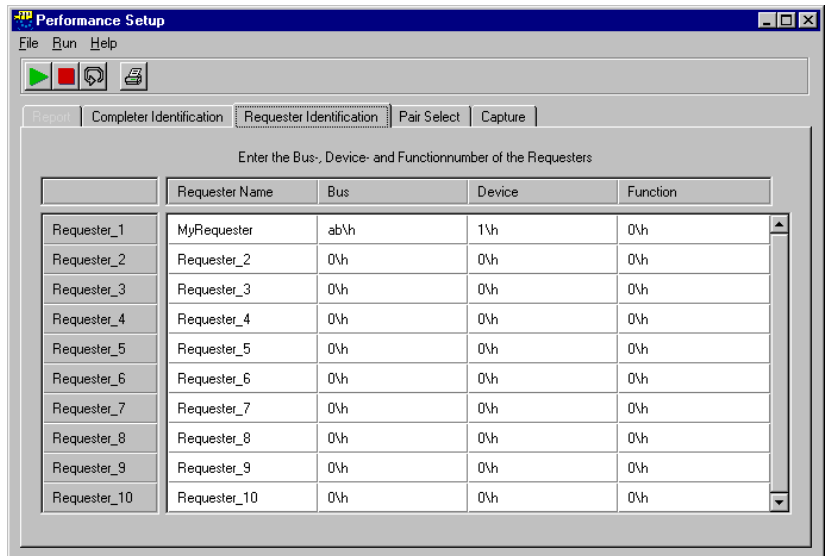


Specifying the Requester Identification

To view the requester identification:

- 1 Select the *Requester Identification* tab in the Performance Setup window.
- 2 Enter the bus, device and function number for a particular requester. You can specify up to ten different requester devices.

In this example, *MyRequester* is specified as shown below.



Determining the Requester ID Number

If you need to know the bus, device and function number of a particular requester, use the Performance Report.

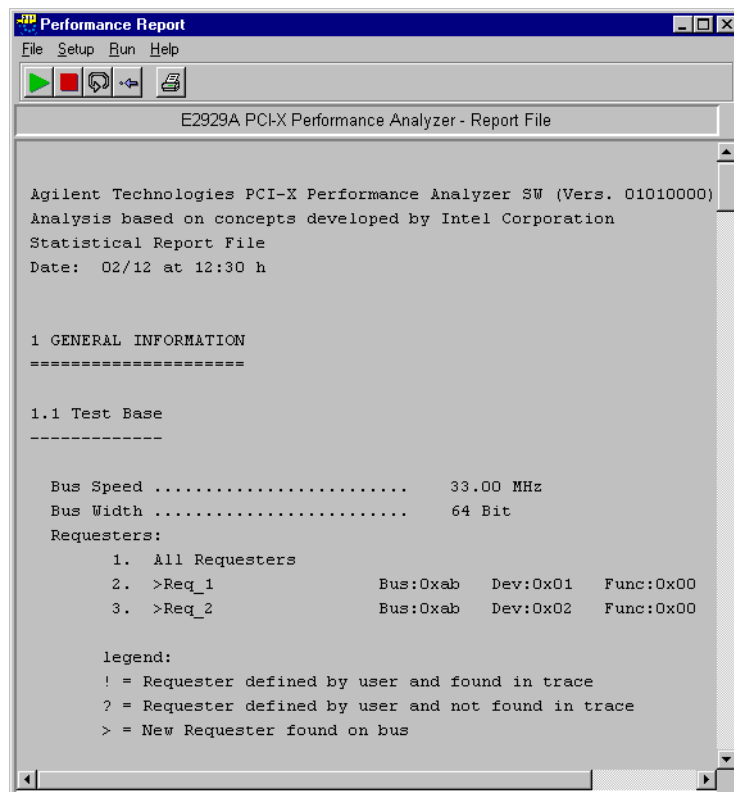
NOTE The following procedure can only be performed in online mode.

To determine the requester ID number:

- 1 Select *Performance Report* from the *Performance* menu in the application's main window to open the Performance Report window.
- 2 Run the performance measurement.

The performance software does a configuration scan on the whole PCI-X bus system and presents the results in the report file.

Subsection 1.1 of the report lists all requesters found on the bus and displays their corresponding bus, device and function numbers.



```

Performance Report
File Setup Run Help
E2929A PCI-X Performance Analyzer - Report File

Agilent Technologies PCI-X Performance Analyzer SW (Vers. 01010000)
Analysis based on concepts developed by Intel Corporation
Statistical Report File
Date: 02/12 at 12:30 h

1 GENERAL INFORMATION
=====

1.1 Test Base
-----

Bus Speed ..... 33.00 MHz
Bus Width ..... 64 Bit
Requesters:
  1. All Requesters
  2. >Req_1          Bus:0xab  Dev:0x01  Func:0x00
  3. >Req_2          Bus:0xab  Dev:0x02  Func:0x00

legend:
! = Requester defined by user and found in trace
? = Requester defined by user and not found in trace
> = New Requester found on bus

```

The desired bus, device and function numbers can now be entered in the *Requester Identification* tab of the *Performance Setup* window.

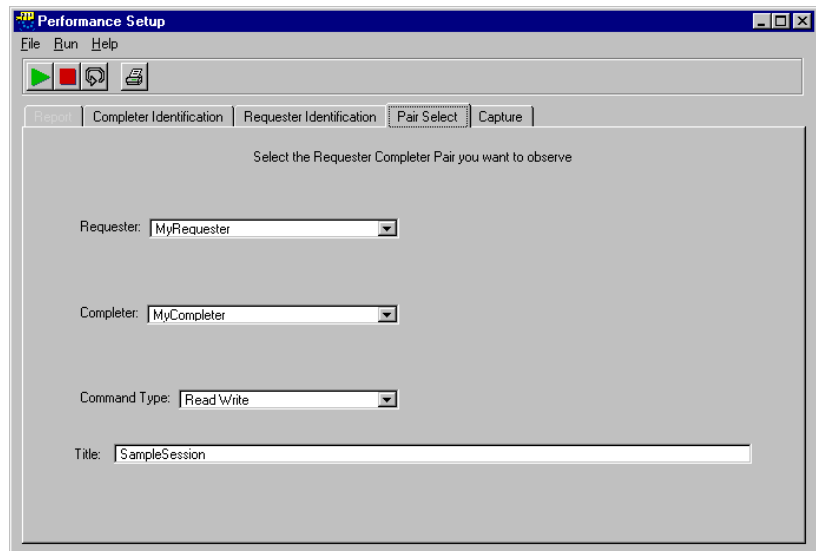
Specifying the Requester-Completer Pair

The setup of the performance test allows you to focus the performance measurement on the traffic of particular devices. Furthermore, the performance optimizer software allows to determine which transfer types (read, write or both) are of interest in your observation.

To specify the requester-completer pair:

- 1 Select the *Pair Select* tab in the Performance Setup window.
- 2 Select the requester and the completer from the lists and determine to which command types the measurement is restricted (Read, Write or both).

In this example, the measurement is focussed on *MyRequester* and *MyCompleter* and the type of command is not restricted.



Specifying the Data Capture

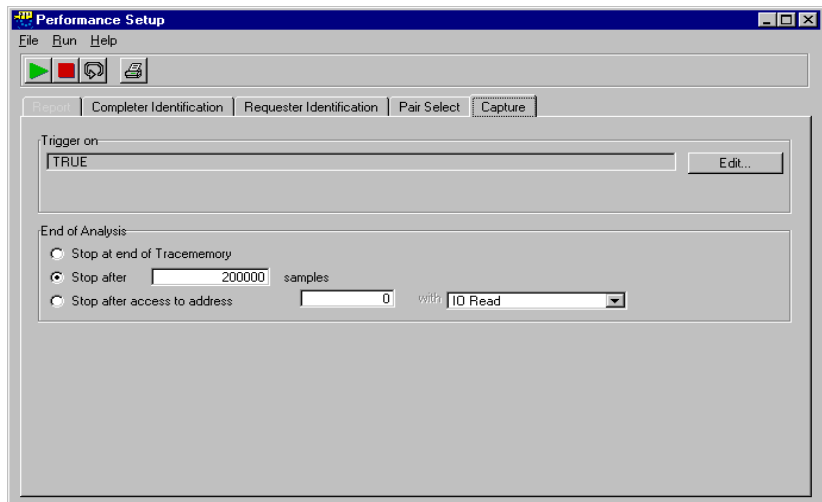
You can set up the performance test not only to focus on the traffic of particular devices, you can also control the start and the duration of the test.

For controlling the start, you can define a particular event that triggers the data capture. For controlling the duration of the test, you can determine the amount of samples to be recorded or specify a particular event that stops data recording.

To specify the data capture:

- 1 Select the *Capture* tab in the Performance Setup window.
- 2 Specify a trigger for the start of the performance measurements and define the duration of the test.

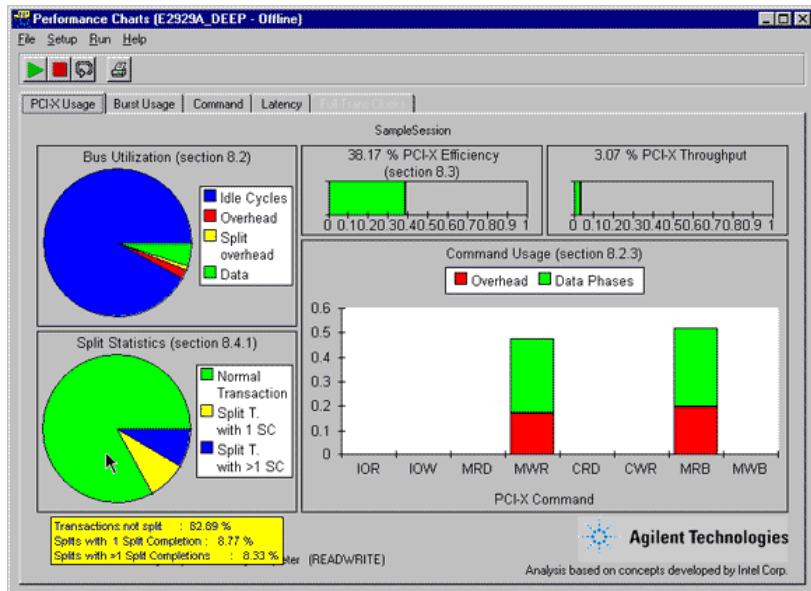
In this test example, the data recording starts immediately. The test stops as soon as the trace memory is filled with 200000 samples.



Viewing the Test Results

The results of the performance tests can be viewed in different ways. For instance, there is the Performance Charts window, that displays the test results in several charts. Another way is the performance report.

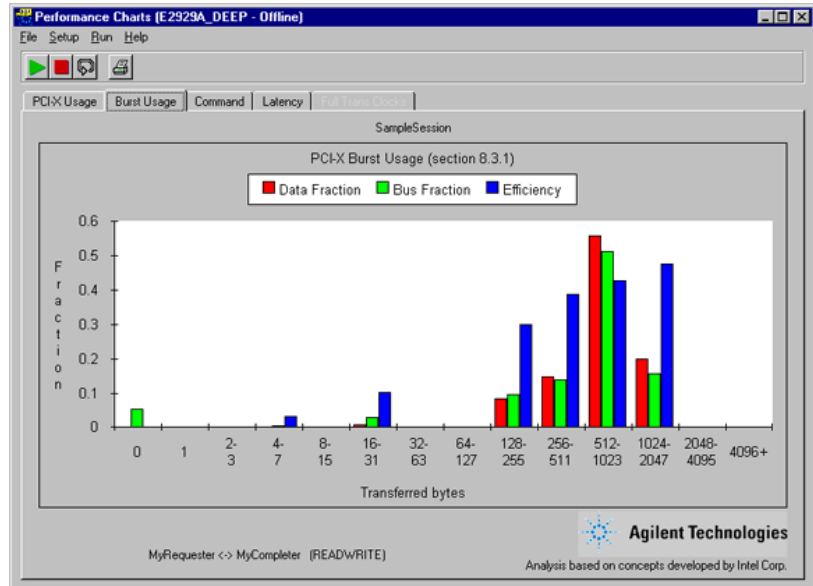
- 1 In the application's main window, select *Performance Charts* from the *Performance* menu to open the Performance Charts window.



This window presents the performance of your system under test on five different tabs. The *PCI-X Usage* tab (as shown above) presents general traffic statistics such as bus utilization, the PCI-X command usage and the split behavior.

- 2 Move the mouse over either pie chart to view the numerical values of the various pie slices. The values are shown as screen tips.

3 Switch to the *Burst Usage* tab.



This tab displays detailed information on the distribution of the data phases and the overhead over the different burst lengths in the traffic.

4 Switch through the different tabs to get an overview of the provided information.

Loading the Example Test Results

For convenience, the results of the example test also are included in the software package. To view these results, do the following steps:

1 In the Performance Charts window, select *Load* from the *File* menu. A message box appears, informing you that the selected action might take a few minutes to complete.

2 Click the *Continue* button to open the file dialog box.

3 In the file dialog box, select the file *SampleSession.wfm* in the path *samples/demo/* and click the *Load* button.

After the loaded data is analyzed, the results are displayed in the Performance Charts window and the other windows of the Performance Optimizer.

NOTE This example waveform file was produced on a 64-bit system using an Agilent E2929A/B_Deep PCI-X testcard. Thus, to successfully load it into the GUI, the bus width and the hardware need to be set up the same way. If your software is set up differently, a message box informs you. Change the hardware settings and repeat the procedure above.

Setting Up a Real-Time Performance Measurement Test

In the bring-up and debug phase of a PCI-X device or a system (containing PCI-X bus and PCI-X devices), you need to evaluate the performance of the device or system under test.

The Agilent PCI-X E2920 software supports real-time performance measurement by providing predefined, standardized performance measurements, such as PCI-X efficiency and PCI-X utilization.

These measures can be set up easily. The results are shown in the main window of the RTP GUI.

The performance measurement is based on counting certain events on the PCI-X bus. For the predefined measurements, the counters are set up automatically.

The PCI-X Performance Optimizer (option #200) expands the possibilities of real-time performance measurements, by providing means for detailed post-processed analysis.

Generating PCI-X Traffic

The PCI-X Analyzer can measure any kind of PCI-X traffic, regardless of how it was generated. However, it is useful to generate traffic in a controlled way for reproducibility in case of troubleshooting or root cause analysis.

Typically, you will use benchmark tests to generate traffic for this purpose.

Predefined Performance Measures

For real-time performance measurements, the PCI-X Analyzer counts occurrences of predefined events or sequences of events on the PCI-X bus. The results are derived and displayed in real time.

Available Measurements The following predefined measures are provided:

- **Throughput**

Throughput is the amount of transferred data per time. It is measured in Mbyte per second. When running a real-time measurement, this value is displayed in percent of the maximum value (for 133-MHz/64-bit systems: 1014 Mbyte per second).

The maximum value can vary between 132 MByte/s in a 33 MHz/32-bit system, 528 MByte/s in a 66 MHz/64-bit system and up to 1014 MByte/s in a 133 MHz/64-bit system.

- **Utilization**

Utilization measures the relation between busy bus time and total bus time during a transfer.

- **Efficiency**

Efficiency is a measure of how well the bus is used. It is the most important value when considering PCI-X performance.

The efficiency of a transfer is the relation between the amount of data that was *really* transferred and the amount of data that *could* have been transferred by the used cycles of that transfer (busy clocks).

Efficiency is derived from throughput and utilization. An efficiency near 100 % means that a device made best use of the time it occupied the bus (utilization) by transferring as much data as possible during that time frame (high throughput).

- **Retry Rate**

This is the ratio between transactions terminated by retry and all terminations.

- **Split Rate**

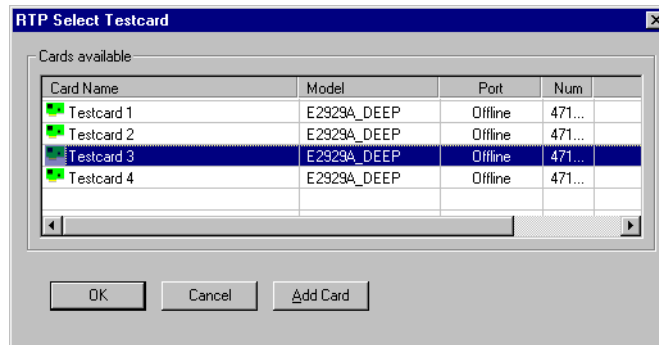
This is the ratio between transactions terminated by split and all terminations.

How to Select a Testcard for the Measurement

To select a testcard:

- 1 Select *Select Testcard...* from the *Edit* menu.

This opens the RTP Select Testcard window.



- 2 Click onto the testcard you need for the performance measurement.

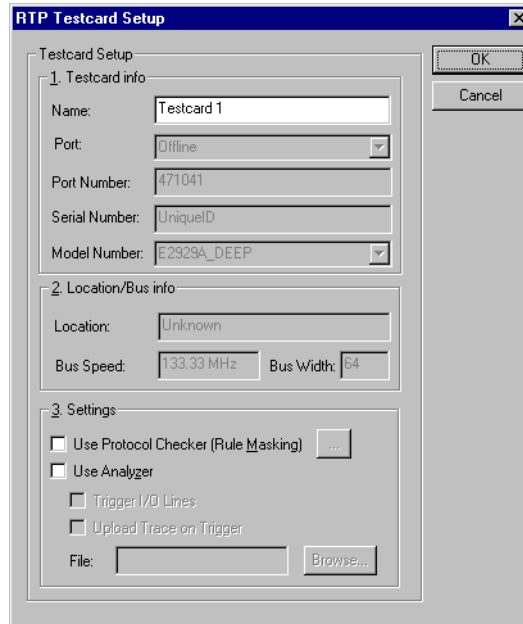
The selected testcard can be specified in the RTP Testcard Setup window.

How to Set Up the Testcard for the Measurement

To view the settings of the available testcards:

- ◆ Select *Setup Testcard...* from the *Edit* menu.

This opens the RTP Testcard Setup window.



You can modify all current testcard settings under *Settings*. Here you can enable and disable card features (protocol checker and analyzer features).

Features The testcard's Analyzer part includes the protocol observer and trigger in/out capabilities.

- Protocol Checker

The testcard's protocol checker continuously monitors the bus and checks for violations of predefined protocol rules, which are partly defined by the PCI-X specification and partly by Agilent. Each individual rule can be masked out. In this case, it neither triggers the trace memory, nor appears in the report. To mask rules, click the details button next to the *Use Protocol Checker (Rule Masking)* check box.

- External/Cross Triggering

To facilitate triggering of external measurement devices, and to trigger other testcards in the system for a *snapshot* whenever an error occurs, the testcards are set up to use the external trigger lines that must be connected to reflect their internal triggering state. Whenever the testcard's trace memory triggers, a trigger-out signal is generated. All trigger-in lines are monitored and used to trigger the card's trace memory.

Which trigger-out line is used for triggering is determined by the testcard's bus number. Therefore, only one testcard per bus needs to be used for cross-triggering.

To find out which trigger-out line is used, use the following formula:

```
triggerline := bus number MOD 12
```

Example:

```
bus number is 16 -> trigger line is 4;
```

```
bus number is 5 -> trigger line is 5; ...
```

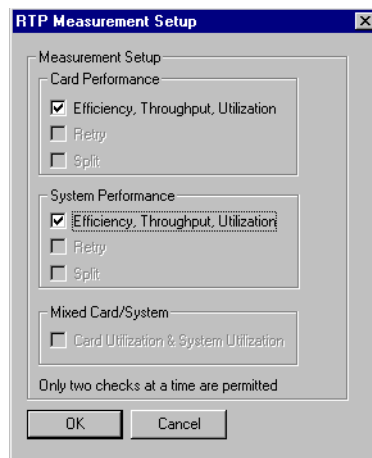
For further information on testcard settings, refer to *RTP Setup Testcard Window* in the *Agilent E2929A/B Windows and Dialog Boxes User Interface Reference*.

How to Select Predefined Performance Measures

The Agilent E2920 Performance software can calculate two real-time measures simultaneously. The test results are displayed side by side on screen.

To select the predefined performance measures to be used:

- 1 From the *Edit* menu, select *Setup Measurement...*



- 2 Check the predefined measurements to be calculated and displayed for card performance, system performance or both.

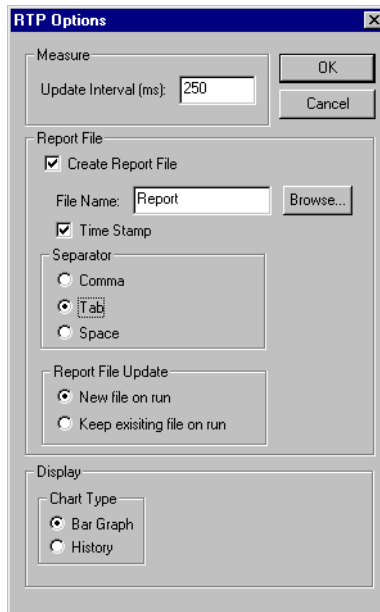
NOTE You can only run two measurements at one time.

The settings will be used the next time you start a performance measurement.

How to Set up Run Options

To set up run options:

- 1 Open the RTP Options dialog box by selecting *Options...* from the *View* menu.




- 2 Set the time interval in which the display is to be updated. You can enter an *Update Interval* between 100 ... 7FFFFFFF\h milliseconds.
- 3 To get a file (*.rtl) that contains all measurements results, check *Create Report File*. This allows you to specify further report options. For more information on report options, please refer to *RTP Options Dialog Box* in the *Agilent E2929A/B Windows and Dialog Boxes User Interface Reference*.
- 4 Specify how the results are displayed:
Select the *Chart Type*:
 - *Bar Graph* shows the results of the last measurement time interval.
 - *History* shows the result history. This is useful to detect peaks.

How to Run a Performance Measurement

After completing the setup, the real-time performance measurement can be started.

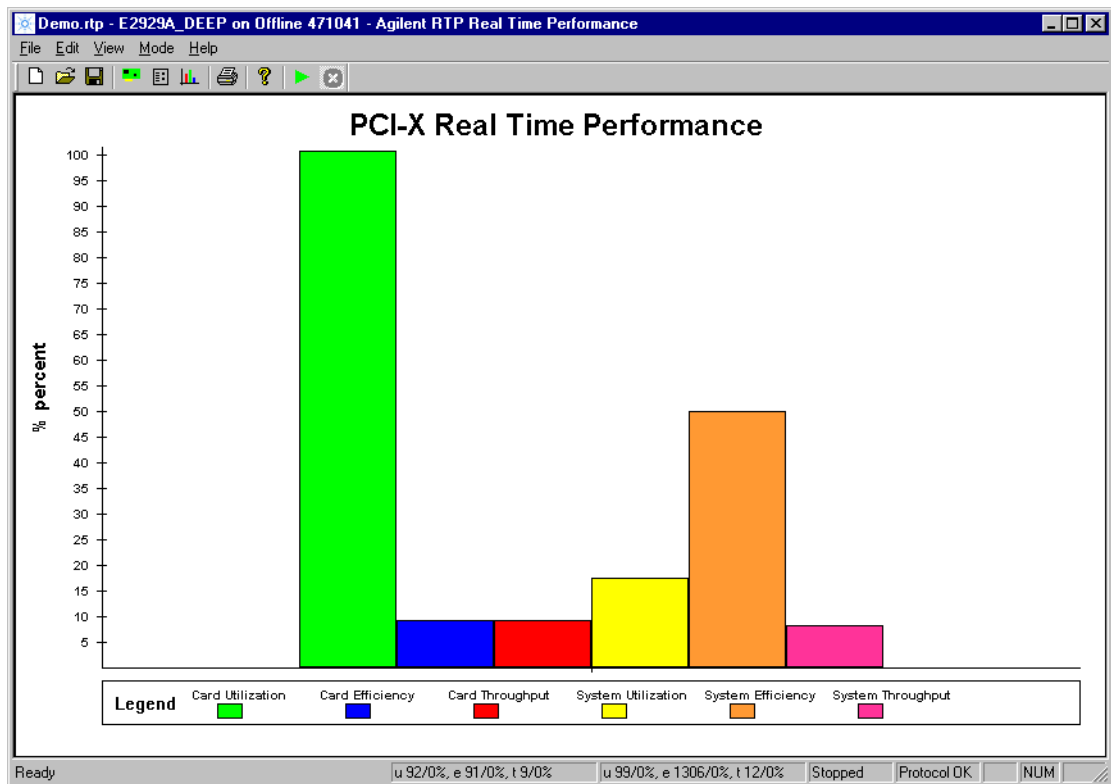
Performance measurements can only be run in online mode. Click *Go Online* in the *Mode* menu, if necessary.

Running the Test To run the test configuration, click either

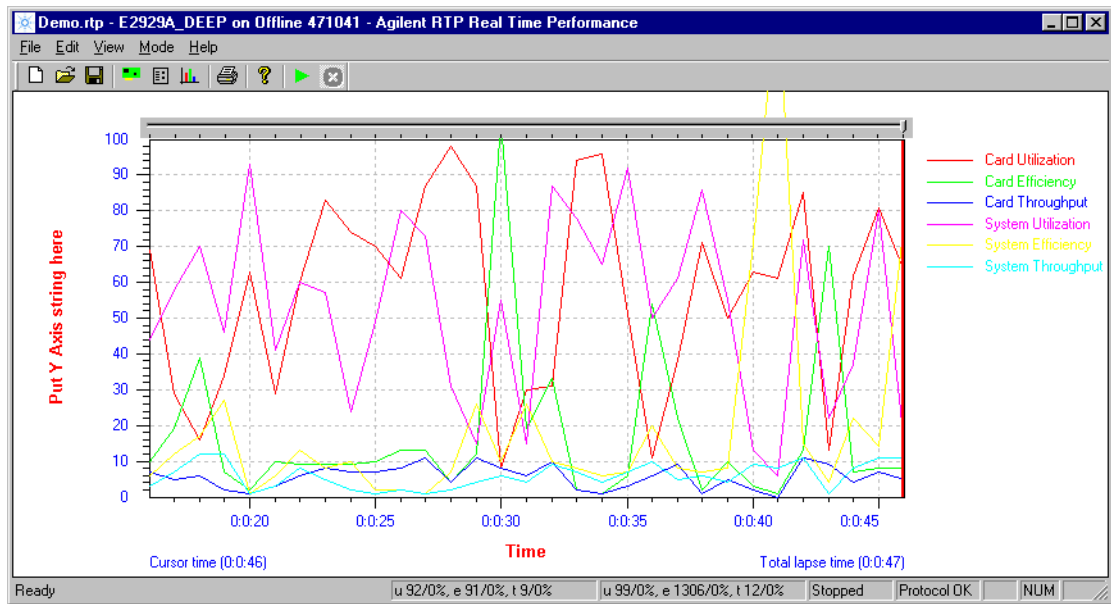
- the *Run* icon  in the tool bar, or
- *Run* in the File menu.

Display Modes This shows the results of the selected measurements per time interval.

- If you have selected *Bar Graph* in the RTP Options dialog box, the results are displayed as follows.



- If you have selected *History* in the RTP Options dialog box, the results are displayed as follows.



Stop the Measurement To stop the measurement, click the *Stop* button and close the window.

Setting Up a PCI-X Performance Optimizer Test

All steps that are required to set up a PCI-X performance test with the Agilent E2929A/B Opt. 200 Performance Optimizer are introduced here. How to set up a test properly is explained here, but not the use of the different features as such. See *“Measuring System or Device Performance” on page 47* for information on the features.

NOTE The features of the Performance Optimizer are only available if the Performance Optimizer option has been installed with the correct license key.

Basically, you can divide the setup process into two parts:

- The **hardware setup** usually needs to be done only once for every system under test. It is assumed that you already are familiar with the Agilent E2929A/B testcard. If you need more information about how to install the testcard in the system under test or how to connect it to the control PC, refer to the *“Agilent E2929A/B PCI-X Getting Started Guide”*.

The additional steps required for the Performance Optimizer are described in *“Setting Up the Test Hardware” on page 38*.

- The **software setup** includes starting the Performance Optimizer—once per test session—and the individually setting up for every test. Information on these steps is found in *“Preparing the Software for the Test” on page 40*.

Setting Up the Test Hardware

To evaluate a system's performance with the Agilent E2929A/B Opt. 200 Performance Optimizer, the Agilent E2929A/B testcard must be plugged into the system under test. Refer to *Possible PCI-X Configurations* in the *Getting Started Guide* for the different possible configurations.

Overall System Performance

If you wish to test a system's PCI-X performance but not the performance of a particular PCI-X device on the bus, your hardware setup is already finished with the installation of the testcard and the establishment of the connection to the control software. Proceed with *"Preparing the Software for the Test"* on page 40.

Performance of Particular Devices

If, on the other hand, you want to evaluate the traffic of particular PCI-X devices, you need to allow the Performance Optimizer to identify requester and completer devices:

- Requester devices

Requester devices initiate all read/write traffic on the PCI-X bus. Requester devices are identified by their bus number, device number and function number.

How to specify the requester devices in the Performance Optimizer is described in *"Requester Identification"* on page 39.

- Completer devices

Completer devices are devices addressed by a transaction. Thus, completer devices are identified by their address spaces.

How to specify the completer devices and their address spaces is described in *"Completer Identification"* on page 39

Requester Identification

If you want to measure and evaluate the PCI-X traffic initiated by particular requester devices, you need to set up the Performance Optimizer to identify these requesters.

The requesters are specifically identified by their bus number, function number and device number.

The easiest way to find out which numbers the system under test assigns to the various requester devices is by running the performance measurement and viewing the report output. Section 1.1 of the report lists all requesters found on the bus and displays their corresponding bus, device and function numbers.


Completer Identification

Every completer in a PCI-X system is assigned a unique address range during system startup. A completer may have several decoders and may, therefore, use several different address spaces at once. The PCI-X Performance Optimizer considers the different address spaces of such PCI-X devices as independent devices.

Basically, memory and I/O transactions can use the same addresses. Depending on the type of traffic to be examined, you can specify whether only memory commands, I/O commands, or both are to be evaluated by the PCI-X Performance Optimizer. See *“Preparing the Software for the Test” on page 40* for more information.

Preparing the Software for the Test

This section covers all information needed to set up the Agilent E2929A/B Opt. 200 PCI-X Performance Optimizer. It is assumed that the required hardware has already been set up. If this is not the case, refer to “*Setting Up the Test Hardware*” on page 38.

The different steps of the software setup—except for the first one—can be done on the different tabs in the Performance Setup window of the GUI. This window can be opened either by selecting *Performance Setup* from the *Performance* menu or by clicking the Performance Setup button  in the main window.

Steps of Software Setup The different steps of the software setup are:

- *Requester Identification* tab and *Completer Identification* tab

In case you want to run performance measurements on particular PCI-X devices, you need to specify the respective requester and completer devices. See “*Setting up the Completer Identification*” on page 41 and “*Setting up the Requester Identification*” on page 42 for details.

- *Pair Select* tab

To restrict the performance test on a single device or a single requester-completer pair, see “*Selecting a Requester-Completer Pair*” on page 43 for instructions.

- *Capture* tab


Finally, to customize the start and end of the measurement period, you can specify trigger variables. See “*Setting Up the Data Capture*” on page 44 for details.

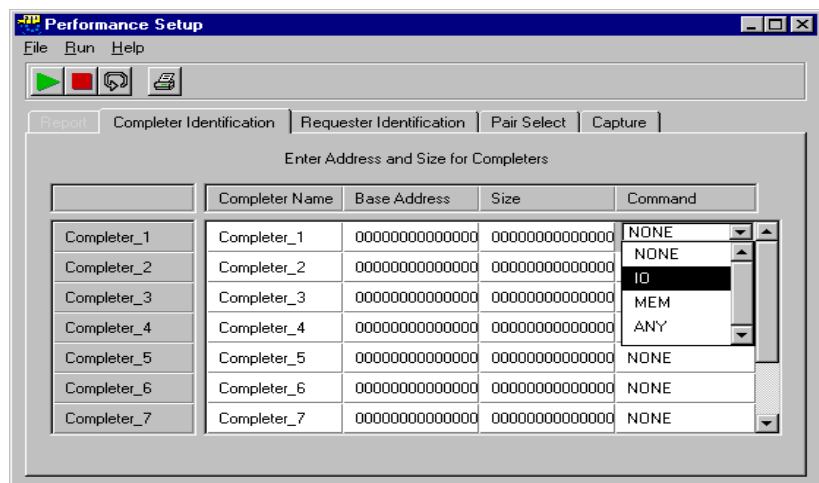
Setting up the Completer Identification

The PCI-X Performance Optimizer allows you to evaluate the performance of single PCI-X devices or communicating pairs of devices. The completer identification is used to specify the completer devices of interest. Every completer is identified by the address range that it decodes. You can monitor up to ten different completers at a time.

One completer device in a PCI-X system may have up to six completer decoders, each using its own address space. Within the Performance Optimizer they are regarded as independent completers.

To define the completer identification proceed as follows:

- 1 If the Performance window is not yet opened, click the Performance Setup button  in the main window.
- 2 Select the *Completer Identification* tab.




- 3 If you wish to have a particular name for a completer in the report and displays, enter the name in *Completer Name*. The description of entering a name is an option.
- 4 Configure the completer by entering the following:
 - The *Base Address* of the completer's memory.
The ending *h* marks hex values.
 - The *Size* of the completer's memory.
The ending *h* marks hex values.
 - Which commands are to be used.
With this selection, you can restrict your measurements to traffic using either I/O commands or memory commands, or ignore it at all.

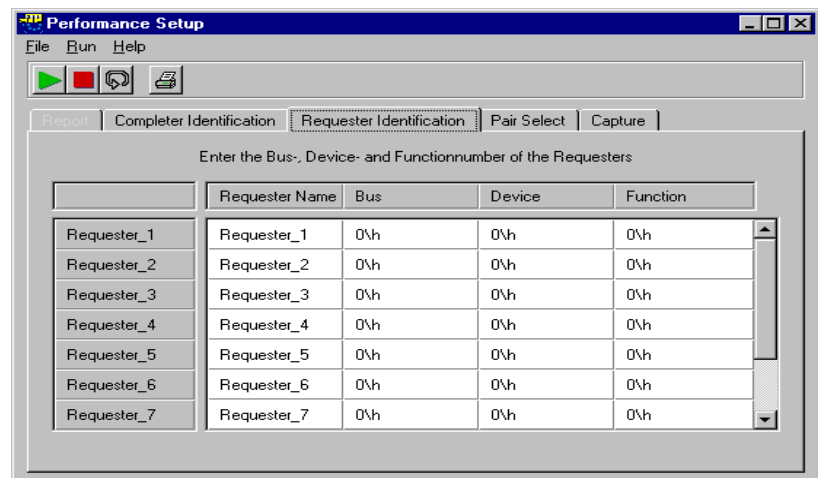
- 5 Repeat the steps for all completers you want to be considered in the performance measurements.

Setting up the Requester Identification

The requester identification is used for performance measurements on one or more particular requester devices. Every requester is identified by its bus, device and function number. You can monitor up to ten different requesters at a time.

To define the requester identification:

- 1 If the Performance window is not yet opened, click the Performance Setup button  in the main window.
- 2 Select the *Requester Identification* tab.




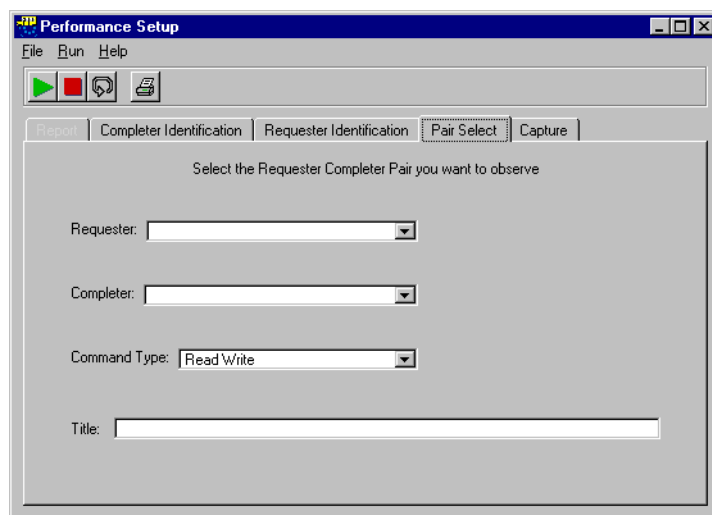
- 3 If you wish to have a particular name for a requester in the report and in the graphical displays, enter a *Requester Name*. The description of entering a name is an option.
- 4 Configure the requester by entering the following:
 - The bus number in which the requester is located.
 - The device number of this requester.
 - The function number of this requester.
- 5 Repeat the steps for all requesters you want to be considered in the performance measurements.

Selecting a Requester-Completer Pair

Besides the more general performance measurements for the specified PCI-X devices, the Performance Optimizer can also report detailed transaction statistics for a particular device or a communicating pair of devices.

To select a device or a pair of devices for the test:

- 1 If the Performance window is not yet opened, click the Performance Setup button  in the main window.
- 2 Select the *Pair Select* tab.




- 3 From the *Requester* list and the *Completer* list select the combination of devices that you want to observe.
- 4 If you want to restrict your measurements to data transfers in one direction only, make the respective selection in the *Command Type* list.
The traffic direction is defined as seen from the requester's view. Thus, *Read* represents the traffic from completer to requester, and *Write* from requester to completer.
- 5 If you want to specify a title that represents the device selection, enter a title in the *Title* text field. This title will then appear in the report and the output charts.

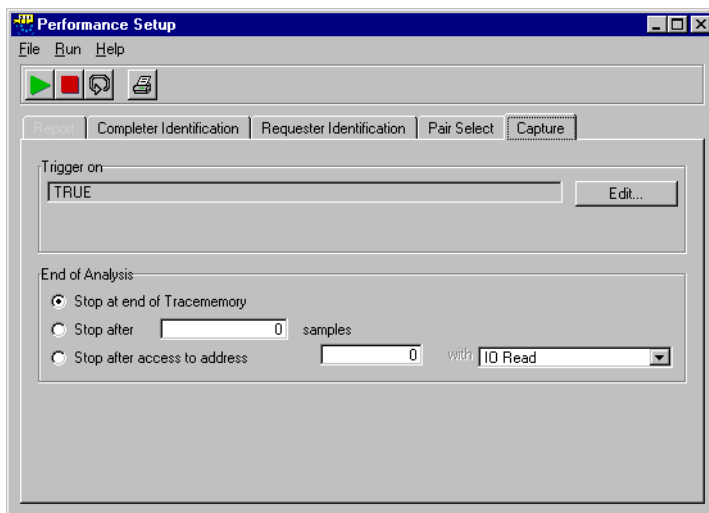
Setting Up the Data Capture

For certain tests, it can be very useful to measure the performance during a particular period of time. Especially if you want to optimize or debug a PCI-X device, it is required to record the data traffic for post-processed analysis only when certain events have occurred.

For purposes like this, it is possible to either delay the start of the test for a fixed time or to wait for a trigger event on the bus. Also, the end of the test can be set to a fixed number of recorded bus clocks or after an access to a certain address with a certain command.

To specify a trigger for a conditional start:

- 1 If the Performance window is not yet opened, click the Performance Setup button  in the main window.
- 2 Select the *Capture* tab.



- 3 Define the trigger in the *Trigger on* group by clicking on the *Edit* button.

The Capture window opens, where you can define the test to either start immediately or after the specified pattern was found to be true.

This window is part of the Agilent PCI-X Analyzer GUI and is not explained in detail here. If you need more information, please refer to *How to Set Up the Trigger* in the “Agilent E2929A/B PCI-X Analyzer User’s Guide” Agilent.

4 Define the length of the test run in the *End of Analysis* group. The three possible options are:

- *Stop at end of Tracememory* (default)

With this option, the measurements continue until the complete trace memory is filled.

- *Stop after ... samples*

This option defines the test to stop after a certain number of recorded traffic samples. Specify the number of samples to record in the text field. This number must not be larger than the size of the trace memory.

A sequence of bus cycles without any signal changes concerning the performance (for example idle states) is stored as one sample.

Thus, the number of observed bus cycles is larger than the number of samples in the trace memory.

- *Stop after access to address ... with ...*

The analysis stops after an access to the specified address with the specified PCI-X bus command. Enter the address in hex format in the text field and select a bus command from the list.

If the address or command does not occur on the bus, the test will run until the trace memory is filled completely.

If the measurements should run too long, you can always manually terminate the tests. The results presented then are calculated based on the completed test repetitions only.

NOTE If you store the waveform file after a repetitive test run, only the data of the most recent test will be stored. Later analysis of this file will possibly yield results other than the statistics on the complete set of tests.

Measuring System or Device Performance

The Agilent E2929A/B Opt. 200 PCI-X Performance Optimizer can be used to measure the performance of a complete PCI-X system as well as to analyze particular PCI-X devices. The presented results and statistics allow you to debug the test devices and to optimize their performance in terms of data throughput, efficiency, bus utilization and split behavior.

After you have set up the test correctly (as described in *“Setting Up a PCI-X Performance Optimizer Test”* on page 37), you can run it and, when finished, examine the results in several different views.

- All information needed to run the performance measurements is found in *“Running the Performance Measurements”* on page 48.
- After the test is finished, the Performance Charts window presents the results. These charts are explained in *“Interpreting the Result Charts”* on page 49.
- An introduction to the contents of the performance report can be found in *“Interpreting the Report Output”* on page 54.
- An example of a performance report of a system evaluation test is explained in more detail in *“Using the Report Output for Performance Evaluation”* on page 55.
- Another example containing statistics on the data traffic of a particular device is found in *“Using the Report Output for Performance Optimization”* on page 62.

Running the Performance Measurements

The Agilent E2929A/B PCI-X Performance Optimizer allows you to record detailed information on the bus traffic in the trace memory and evaluate it after the test is finished.

Assuming that the test was setup properly, the test will record bus traffic information in the trace memory for a certain amount of time. The length depends on the system speed and the size of the trace memory. The test can be controlled with several buttons found in the different windows of the Performance Optimizer—except the Performance Report window.



Start the Test To start the test, click the Run button in any of the windows of the Performance Optimizer. You can also select *Run* from the *Performance* menu in the main window.

Stop the Test If for any reason your test does not seem to finish within a reasonable time, for example, if the specified trigger event does not occur, you can terminate it manually. Click the Stop button in either of the windows of the Performance Optimizer or select *Stop* from the Performance menu in the main window.

After the test is finished, the contents of the trace memory will be evaluated and the results will be presented in the Performance Charts window and the Performance Report window.

Rescan the Trace Memory The Rescan button is meant to be used when doing different performance calculations on the same data.

For example, you can use it if you want to compare the performance of different requester-completer pairs during the same test period. In a case like this, you may not want to run a new test but would rather evaluate the data in the trace memory again with different settings.

After running the test for the first requester-completer pair, you only have to change the selected pair in the setup window and click on the *Rescan* button to calculate the new results.

Interpreting the Result Charts

After successfully running the performance measurements on the system under test, you can view the results in the different windows of the Performance Optimizer. Here a description of the charts of the Performance Charts window is found.

These charts display several results of the analysis. These results always refer to the device pair specified in the test setup. If you want to evaluate the performance of the complete system under test, the device pair *All Requesters/All Completers* needs to be selected. For more information, see *“Selecting a Requester-Completer Pair” on page 43*.

Performance Chart Tabs The result charts in the Performance Charts window are located on the following different window tabs:

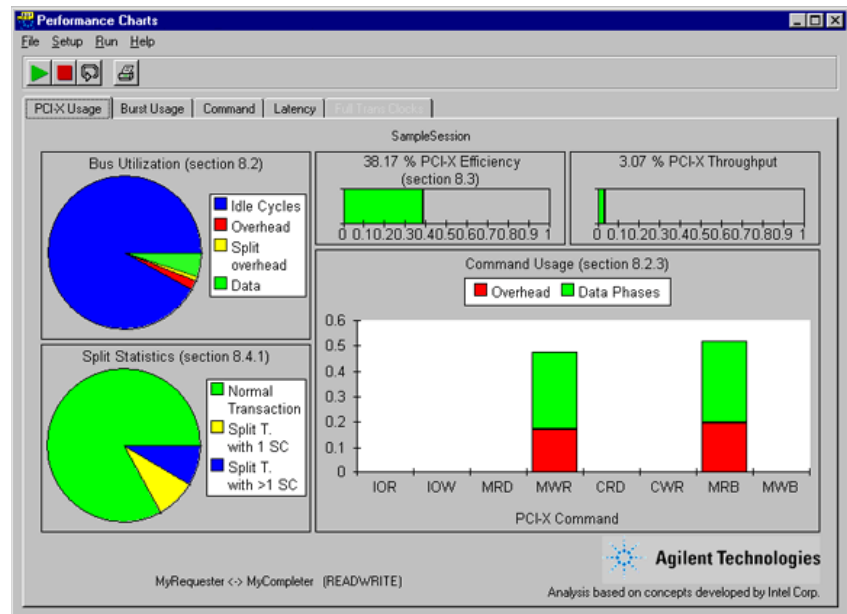
- *“PCI-X Usage” on page 50*
- *“Burst Usage” on page 51*
- *“Command” on page 52*
- *“Latency” on page 53*

Data Source for Performance Charts All performance charts are derived from the data recorded in the trace memory. The complete results of the measurements on that data are presented in the performance report. To view all details concerning a certain performance chart, refer to the respective section in the performance report.

PCI-X Usage

This tab displays general statistics on the usage of the PCI-X bus. It contains up to five charts—depending on the test setup—that provide information on the recorded bus traffic: PCI-X efficiency, bus utilization, PCI-X throughput and command usage.

This window also displays the usage of split and normal transactions. The selection of the requester-completer pair observed in the test is displayed at the bottom of the window.



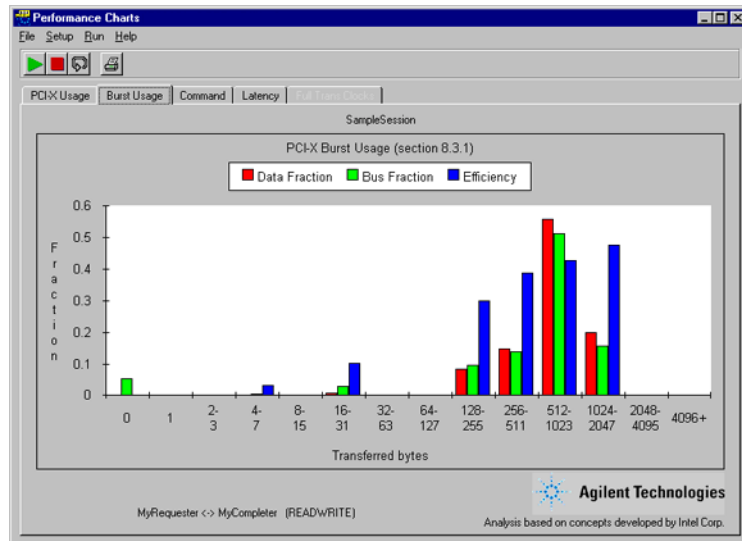
For a detailed description of the charts, see the topic named *PCI-X Usage Tab* in the online help. The contents of the different charts are explained in the respective sections of the performance report.

Depending on the test setup, up to four diagrams are shown in this overview.

Burst Usage

The diagram in this tab corresponds to the section 8.3.1 of the performance report. It shows the distribution of the burst lengths that contributed to the bus traffic.

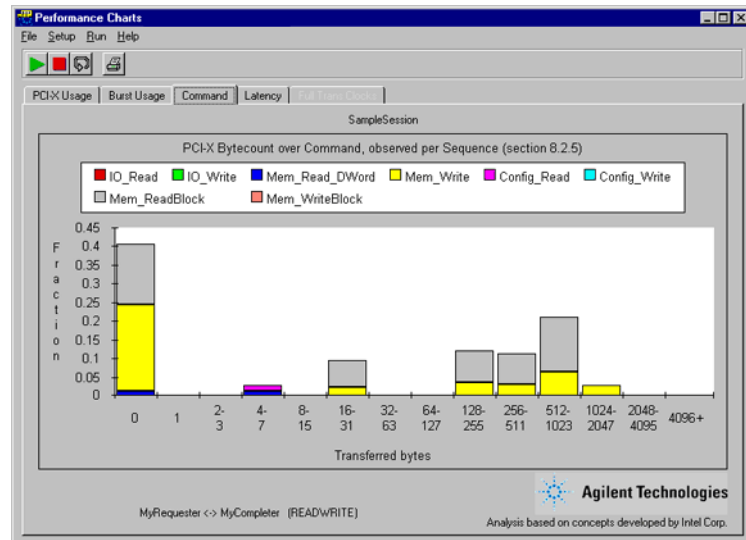
Burst lengths are displayed starting with 0, 1 and then increasing in ranges $\{2^n, 2^{(n+1)} - 1\}$, where n ranges from 1 ... 11.



For a detailed description of the chart, see the topic named *Burst Usage Tab* in the online help.

Command

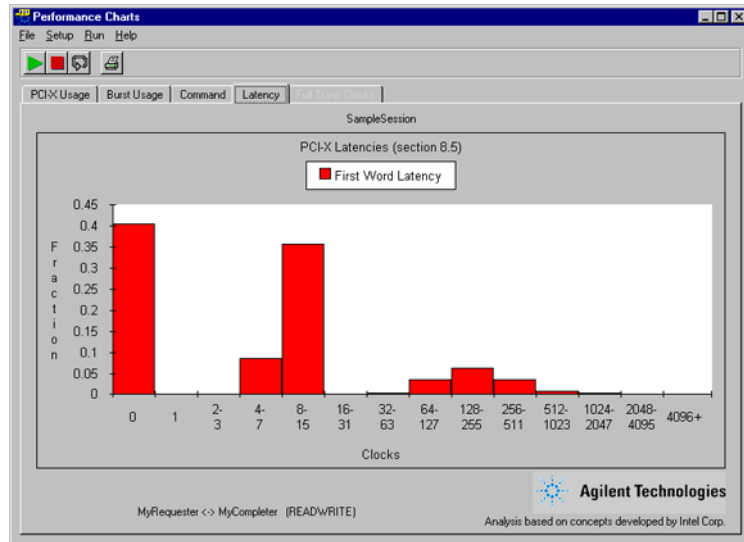
The diagram in this tab corresponds to the section 8.2.5 of the performance report. It displays the distribution of the different bus commands during the traffic (compare the “PCI-X Usage” on page 50). More specifically, it is split up with respect to the different burst lengths. Burst lengths are displayed starting with 0, 1 and then increasing in ranges $\{2^n, 2^{(n+1)} - 1\}$, where n ranges from 1 ... 11.



For a detailed description of the chart, see the topic named *Command Tab* in the online help.


Latency

The diagram in this tab corresponds to the section 8.5 in the performance report. It displays the distribution of the different latencies that occurred during bursts of the different lengths.



For a detailed description of the chart, see the topic named *PCI-X Usage Tab* in the online help.

Interpreting the Report Output

After data traffic from the PCI-X bus was recorded in the trace memory and the performance measurements were done on this data, the Performance Report window displays the results in a text report. To open this window, click the Performance Report button  in the *Performance* group of the main window.


The Report as Chart Reference

The report contains detailed performance information and traffic statistics. A selection of these results is also presented in several charts in the Performance Charts window. The corresponding section of the report is displayed as a reference with the diagrams. See “*Interpreting the Result Charts*” on page 49 for more information.

Navigation in the Report

For convenience, the report is structured into several sections. The first sections present an overview on the performance of the complete system under test. Then there is general information on the performance of the specified requester and completer devices. The last sections contain detailed statistics on the communication between the selected device pair.

Hyperlinks—marked with blue text—are provided in several places in the report. They connect report sections with related subjects. The links lead you to the section describing the performance value in more detail.

To go back to the previously viewed section, click the Back button  in the tool bar of this window.

Report Contents

To learn about the contents provided in the various sections of the report, you can

- work through the “*Using the Report Output for Performance Evaluation*” on page 55. This example describes the sections 1 to 7.
- work through the “*Using the Report Output for Performance Optimization*” on page 62. This example describes the section 8.
- see the “*Report Reference*” on page 87 for a complete and detailed reference of the contents.

Using the Report Output for Performance Evaluation

This example shows how you can use the report sections 1 to 7 to evaluate the performance of a PCI-X system or device using the PCI-X Performance Optimizer.

For this example, a PCI-X card was used as device under test in a 64-bit system. For the system, the PCI-X performance was examined using the PCI-X Performance Optimizer. The report sections that lead to the card's shortcomings, are shown and explained in the following. Some of the report sections can also be viewed in charts.

General Information on the System under Test

The first section in the test report is titled **General Information** and lists the bus speed, the bus width and all specified devices. The example device in this test is the completer device named **MyCompleter** as displayed in the figure below.

```

1 GENERAL INFORMATION
=====

1.1 Test Base
-----

Bus Speed ..... 66.67 MHz
Bus Width ..... 64 Bit

Requesters:

  1. All Requesters
  2. >Req_1          Bus:0xab  Dev:0x02  Func:0x00
  3. >Req_2          Bus:0xab  Dev:0x01  Func:0x00

legend:
! = Requester defined by user and found in trace
? = Requester defined by user and not found in trace
> = New Requester found on bus

Completers:

  1. All Completers      Addr: 0000000000000000\h
                        Size: ffffffff\h
                        I/O and Mem
  2. MyCompleter        Addr: 000000000004003c\h
                        Size: 000000000000100\h
                        I/O and Mem

Selected Pair:          All Requesters -> MyCompleter

```

The data traffic to My Completer is part of the bus traffic and is analyzed in the following report sections.

Analyzing the PCI-X Throughput

Report section 2 displays the basic bus statistics. Here values such as PCI-X throughput, bus utilization and efficiency are presented.

```

2 BASIC BUS STATISTICS
=====
    PCI-X Throughput ..... 118.2687 Megabytes/sec
    ..... 946.1424 Megabits/sec
    PCI-X Utilization ..... 62.15 %
    PCI-X Efficiency ..... 37.41 %
    Average First Word Latency..... 15 clocks
    Average Split Rate..... 9.52 %
  
```

The overall PCI-X throughput in the basic bus statistics has to be compared to the throughput in the bus throughput statistics (report section 3).

```

3 BUS THROUGHPUT STATISTICS
=====
Requester Completer Throughput (MBytes/sec)
-----
          | MyComple | Other Co | All Comp
          | ter       | mpleter  | leters
          |-----|-----|-----|
Req_1     | 0.00     | 79.40    | 79.40
Req_2     | 3.71     | 35.17    | 38.87
All Requesters | 3.71     | 114.56   | 118.27
  
```

From this report section, it can be seen that the throughput of the selected requester-completer pair is a small share of the total throughput of 118.27 MBytes/sec.

For comparison, the PCI-X bus with 66.7 MHz bus speed theoretically supports a maximum transfer of 509 MBytes/sec.

Analyzing the PCI-X Utilization

In the basic bus statistics (report section 2), you can find the PCI-X utilization and other similar statistics.

```

2 BASIC BUS STATISTICS
=====
PCI-X Throughput ..... 118.2687 Megabytes/sec
                    ..... 946.1493 Megabits/sec
PCI-X Utilization ..... 62.15 %
PCI-X Efficiency ..... 37.41 %
Average First Word Latency..... 15 clocks
Average Split Rate..... 9.52 %
    
```

The **PCI-X Utilization** value is 62.15 %. This means that the bus was used only in 62.15 % of the available clock cycles. During the remaining time it was idle.

A more detailed analysis of the bus utilization can be found in the bus utilization statistics (report section 5):

```

5 BUS UTILIZATION STATISTICS
=====
Overhead Utilization ..... 27.99 %
Data Phase Utilization ..... 34.16 %
                    -----
PCI-X Utilization ..... 62.15 %

5.1 Requester Completer Utilization
-----
                | MyComple | Other Co | All Comp
                | ter      | mpleter  | leters
                -----
Req_1           | 0.00 % | 35.26 % | 35.26 %
Req_2           | 2.60 % | 24.28 % | 26.89 %
All Requesters  | 2.60 % | 59.54 % | 62.15 %
    
```

From this report section, it can be seen that half of the bus is occupied by overhead and that the selected requester-completer pair makes a contribution of only 2.60 % to the total utilization.

Analyzing the PCI-X Efficiency

Within the basic bus statistics (report section 2) you can see the values for the PCI-X Efficiency of the system under test during the measurements.

```

2 BASIC BUS STATISTICS
=====
PCI-X Throughput ..... 118.2687 Megabytes/sec
                    ..... 946.1424 Megabits/sec
PCI-X Utilization ..... 62.15 %
PCI-X Efficiency ..... 37.41 %
Average First Word Latency..... 15 clocks
Average Split Rate..... 9.52 %
  
```

An efficiency of 37.41 % is rather low. Usually, a value of about 50 % is considered to be a reasonable efficiency value.

To go further into details, see the efficiency statistics (report section 4).

```

4 EFFICIENCY STATISTICS
=====
Byte Enable Efficiency ..... 68.07 %
Time Efficiency ..... 54.96 %

4.1 Requester Completer Efficiency
-----

          | MyComple | Other Co | All Comp
          | ter      | mpleter  | leters
          |-----|-----|-----|
Req_1     | 0.00 %  | 44.27 % | 44.27 %
Req_2     | 27.97 % | 28.47 % | 28.42 %
All Requesters | 27.97 % | 37.83 % | 37.41 %
  
```

These statistics show a Byte Enable Efficiency value of less than 70 %. This means that the devices do not always transfer the maximum of eight bytes per clock cycle.

The table above also shows the efficiency of the selected requester-completer pair. The value of 27.97 % is very poor.

Therefore, the behavior of the tested completer device must be improved with respect to this property.

Analyzing the Split Behavior

Within the basic bus statistics (report section 2) you can see the values for the average split word latency and the average split rate of the system under test during the measurements.

```

2 BASIC BUS STATISTICS
=====
PCI-X Throughput ..... 118.2687 Megabytes/sec
                    ..... 946.1424 Megabits/sec
PCI-X Utilization ..... 62.15 %
PCI-X Efficiency ..... 37.41 %
Average First Word Latency..... 15 clocks
Average Split Rate..... 9.52 %
    
```

An average split rate of 9.52 % is rather low. Only a tenth of all transactions within one sequence is split. It is necessary to terminate transactions by using split response instead of signaling retry. Using split response allows other requesters to use the bus for data transfer. This way, idle cycles can be avoided.

For further details, see the split transaction statistics (report section 7).

```

7 Split Transaction Statistics
=====
No of split sequences ..... 4435
Average first word latency of split transactions ..... 45 clocks
Average overall length of split transactions ..... 192 clocks
Average time overhead of split transactions ..... 81.68 %

7.1 Pending Requests over time
-----
Req. | 0 | 1 | 2 | 3 | 4 | 5 | 6+ |
-----
%time| 53.62 %| 35.18 %| 11.20 %| 0.00 %| 0.00 %| 0.00 %| 0.00 %|
    
```

The value of 4435 split sequences represents only a fraction of nearly 10 % (see the average split rate value under report section 2). This must be improved.

Compared with the average first word latency of all transactions (15 clocks), the respective value for split transactions (45 clocks) is very high. This can be caused by the following: If a completer signals a split response, the bus can be used by other requesters to transfer data until the split transaction is completed. This can also be a reason for the high overhead value of 81.68 %.

Taking this into consideration, the high values are the results of good performance.

Furthermore, the table Pending Requests over Time shows that during half of the test run, only one or two requests were pending.

The Bus Users Overview

In report section 6 the Bus Users Overview is displayed.

The Bus Users Overview table contains indices that rate the performance of all device pairs, so that you can compare the values of the several devices.

```

6 BUS USERS OVERVIEW
=====
Calculated as: Utilization divided by Efficiency
-----

```

	MyComple ter	Other Co mpleter	All Comp leters
Req_1	---	0.80	0.80
Req_2	0.09	0.85	0.95
All Requesters	0.09	1.57	1.66

The Bus Users Overview index is calculated as utilization divided by efficiency. High utilization and low efficiency result in a high index value. This means that small values indicate good performance. A value of around 1 is good.

The report shows that the traffic initiated by Req_2 has a very good performance.

Summary for the Performance Evaluation Example

Analyzing report sections 1 to 7 makes the following shortcomings of the device under test clear:

- The amount of data transferred per time is low (only 3.71 MByte/s compared with a maximum of 509 MByte/s).
- The amount of transferred bytes in reference to the maximum possible number of bytes that theoretically could be transferred within the used clock cycle is low.
- Half of the bus is occupied with overhead.

Report Section 8 allows a detailed analysis of these shortcomings.

Using the Report Output for Performance Optimization

This example explains how the flaws of a particular PCI-X device can be examined and how to find ways to improve its design. It is assumed that a general evaluation of the system's performance and behavior already was done and the device with the poor performance was identified.

From the report sections 1 to 7, it was derived that throughput and efficiency of the tested completer device must be improved to increase PCI-X performance and that the generation of overhead must be examined in more detail.

In order to improve the PCI-X design of the card, the card's traffic behavior must be analyzed in more detail.

For this purpose, the report section 8 is analyzed here. Section 8 lists detailed statistics about the communication between the specified requester-completer pair, in this case the traffic between *MyCompleter* and *All Requesters*.

Analyzing the Bus Utilization

The overhead of the tested device was identified as a reason for poor performance. Therefore, the analysis of the bus utilization is useful. Report section 8.2 provides the results for the selected requester-completer-pair:

8.2 Bus Utilization	

Overhead	1.17 %
Split Overhead	0.41 %
Data Phases	1.02 %

Sum of Utilization	2.60 %

The **Sum of Utilization** indicates how intensively the selected device pair participated in the whole traffic on the PCI-X bus. Here, only 2.60 % of all sampled clocks were used by this device pair.

Nevertheless, the overhead (split overhead included) amounts nearly 60 % of the whole traffic caused by the selected requester-completer pair.

Analyzing the Traffic Overhead

The next step is to determine whether the requester or the completer device was responsible for the overhead. For this purpose, see the Overhead summary (report section 8.2.2). The overhead analysis breaks down the values of the section 8.2. It points to the causes of the overhead, and it shows what type of overhead occurred.

8.2.2 Overhead			

Phase	Overhead caused by		
	Requester	Completer	Sum

Addr	15.71 %	3.48 %	19.19 %
Waits	19.14 %	53.68 %	72.82 %
Terms	0.00 %	5.39 %	5.39 %
Recover	0.92 %	1.68 %	2.60 %

Total	35.77 %	64.23 %	100 %
Addrphase includes: Addresscycles, Attributecycles			
Waitphase includes: Decodecycles, Target-Responsecycles			

The sum of all different overhead types is always 100 % and corresponds to the time overhead value found in report section 8.2. Out of this total traffic overhead, the requester caused 35.77 %. The completer, on the other hand, caused 64.23 %. It is obvious that the completer lowers the performance more than the requester. The reason for the overhead is obvious as well: 53.68 % of the total overhead is caused by waits inserted by the completer.

To improve the performance of the completer, waits must be eliminated. Reasons for waits could be, for example, that the completer's components are too slow, or that it has problems with its resources or memory.

Furthermore, it is of interest that the completer causes overhead by inserting address phases (3.48 %), terminations (5.39 %) and recover cycles (1.68 %). However, this is insignificant compared with the huge overhead caused by waits.

This indicates that an examination of the inserted waits could help to find further ways of improving the completer device.

Analyzing the Byte Enable and Burst Behavior

This step examines the byte enable and burst behavior of the tested completer device to find further possibilities for improvements.

The detailed analysis of the Byte Enable Efficiency shows that the maximum of 8 bytes per data phase was not always transferred. This was shown earlier in the bus statistics analysis in report section 4 (see *“Analyzing the PCI-X Efficiency” on page 59* for details).

Now, you have to look at the average burst length in the data communication of the selected device pair as shown in report section 8.2.1:

```

8.2.1 Data Phase
-----
0 Bytes ..... 0.00 %
1 Bytes ..... 2.09 %
2 Bytes ..... 2.09 %
3 Bytes ..... 2.25 %
4 Bytes ..... 46.02 %
5 Bytes ..... 0.00 %
6 Bytes ..... 0.00 %
7 Bytes ..... 0.00 %
8 Bytes ..... 46.59 %

-----
Average Byte-Enable Efficiency ..... 71.23 %

Average Burst Length ..... 8.91 data phase(s)

Pure 32-Bit data transfers ..... 42.40 %
64-Bit data transfer tried but refused . 8.90 %
-----
32-Bit data transfers (sum) ..... 51.31 %

64-Bit data transfers ..... 48.69 %

```

The average burst length of 8.91 data phases shows that the tested device is able to handle bursts properly. To improve the efficiency, bursts with higher byte counts must be used. The efficiency will also be improved if more transfers would use the whole bus width.

Analyzing the Bursts

A detailed list of the different PCI-X bus commands used with bursts of a particular length is found in report section 8.2.5. DWord commands are considered in a separate table.

This section shows the measurement results observed per sequence:

8.2.5 Byte count over Command, observed per Sequence						
Transferred Bytes	MEM_WR ITE	ALIAS_MEM_RB	ALIAS_MEM_WB	SCOMPL ETION	MEM_RB LOCK	MEM_WB LOCK
0	22.50%	0.00%	0.00%	--	7.68%	0.00%
1	0.00%	0.00%	0.00%	--	0.00%	0.00%
2- 3	13.92%	0.00%	0.00%	--	13.92%	0.00%
4- 7	0.00%	0.00%	0.00%	--	0.00%	0.00%
8- 15	0.00%	0.00%	0.00%	--	0.00%	0.00%
16- 31	14.22%	0.00%	0.00%	--	4.64%	0.00%
32- 63	0.00%	0.00%	0.00%	--	0.00%	0.00%
64- 127	0.00%	0.00%	0.00%	--	0.00%	0.00%
128- 255	4.64%	0.00%	0.00%	--	4.64%	0.00%
256- 511	4.59%	0.00%	0.00%	--	9.23%	0.00%
512-1023	0.00%	0.00%	0.00%	--	0.00%	0.00%
1024-2047	0.00%	0.00%	0.00%	--	0.00%	0.00%
2048-4095	0.00%	0.00%	0.00%	--	0.00%	0.00%
4096+	0.00%	0.00%	0.00%	--	0.00%	0.00%
Sum	59.88%	0.00%	0.00%	--	40.12%	0.00%
Transferred Bytes	IO_READ	IO_WRITE	MEM_RD WORD	CONFIG_READ	CONFIG_WRITE	
0	0.00%	0.00%	0.00%	0.00%	0.00%	
1	0.00%	0.00%	0.00%	0.00%	0.00%	
2- 3	0.00%	0.00%	0.00%	0.00%	0.00%	
4	0.00%	0.00%	0.00%	0.00%	0.00%	
Sum	0.00%	0.00%	0.00%	0.00%	0.00%	

The table shows that only memory write and memory read block are used and I/O commands are avoided. There are no transfers using DWord commands.

The burst lengths are varying between 0 and 256 ... 511 bytes. A burst length of 0 indicates that incomplete sequence(s) occurred.

This section shows the measurement results observed per transactions.

8.2.6 Byte count over Command, observed per Transaction						
Transferred Bytes	MEM_WR ITE	ALIAS_MEM_RB	ALIAS_MEM_WB	SCOMPL ETION	MEM_RB LOCK	MEM_WB LOCK
0	14.17%	0.00%	0.00%	0.00%	20.39%	0.00%
1	2.04%	0.00%	0.00%	0.00%	0.00%	0.00%
2- 3	9.74%	0.00%	0.00%	6.47%	2.29%	0.00%
4- 7	5.81%	0.00%	0.00%	0.00%	0.00%	0.00%
8- 15	4.43%	0.00%	0.00%	0.00%	0.00%	0.00%
16- 31	7.51%	0.00%	0.00%	2.32%	0.60%	0.00%
32- 63	0.00%	0.00%	0.00%	1.16%	0.57%	0.00%
64- 127	3.33%	0.00%	0.00%	0.85%	2.14%	0.00%
128- 255	4.37%	0.00%	0.00%	5.03%	2.83%	0.00%
256- 511	1.67%	0.00%	0.00%	2.29%	0.00%	0.00%
512-1023	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1024-2047	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
2048-4095	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
4096+	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Sum	53.06%	0.00%	0.00%	18.13%	28.81%	0.00%
Transferred Bytes	IO_REA D	IO_WRI TE	MEM_RD WORD	CONFIG_READ	CONFIG_WRITE	
0	0.00%	0.00%	0.00%	0.00%	0.00%	
1	0.00%	0.00%	0.00%	0.00%	0.00%	
2- 3	0.00%	0.00%	0.00%	0.00%	0.00%	
4	0.00%	0.00%	0.00%	0.00%	0.00%	
Sum	0.00%	0.00%	0.00%	0.00%	0.00%	

The Byte count over Command table is particularly useful when different commands are used in the data traffic.

In this example, the Memory Write and Split Completion were used with the highest burst lengths.

Memory Write and Memory Read Block are used with a high fraction of bursts with a length of 0. That means, the completer stopped the transaction with a split response or retry.

To examine the quality of the transactions with different burst length, see the report section 8.3.1:

8.3.1 Efficiency over Byte count, observed per Sequence

Transferred Bytes	Efficiency	Data Fraction	Bus Fraction
0	0.00 %	0.00 %	10.25 %
1	--	0.00 %	0.00 %
2- 3	2.11 %	4.68 %	18.15 %
4- 7	--	0.00 %	0.00 %
8- 15	--	0.00 %	0.00 %
16- 31	10.66 %	9.13 %	15.53 %
32- 63	--	0.00 %	0.00 %
64- 127	--	0.00 %	0.00 %
128- 255	45.46 %	30.64 %	21.35 %
256- 511	46.75 %	55.55 %	34.72 %
512-1023	--	0.00 %	0.00 %
1024-2047	--	0.00 %	0.00 %
2048-4095	--	0.00 %	0.00 %
4096+	--	0.00 %	0.00 %
Sum		100.00 %	100.00 %

What this table shows is that the tested device actually is capable of long bursts (up to 511 bytes) and that the efficiency would remarkably increase if the card was able to handle these bursts more often.

From section 8.2.6 it can be seen that a high fraction of the transactions was terminated after the first data phase. To investigate the reasons for this behavior, the distribution of the different command terminations is of interest.

Analyzing Command Termination

The wrong usage of PCI-X commands in the device communication can also be the reason for poor performance.

Thus, the behavior of the device under test in response to different PCI-X commands during the data transfers affects PCI-X performance. The following table displays how the different transactions were handled.

8.2.4 Command Termination, observed per Transaction							
Command	Fraction of Terminations						
	M_ COMPL	M_ ABORT	T_ ABORT	DISC_ SDP	DISC_ ADB	RETRY	SPLIT RESP
INT_ACK	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
SPECIAL	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
IO_READ	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
IO_WRITE	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_1	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_2	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
MEM_RDWORD	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
MEM_WRITE	13.95 %	0.00 %	0.00 %	14.70 %	10.24 %	14.17 %	0.00 %
ALIAS_MEM_RB	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
ALIAS_MEM_WB	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
CONFIG_READ	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
CONFIG_WRITE	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
SCOMPLETION	18.13 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
MEM_RBLOCK	2.29 %	0.00 %	0.00 %	0.00 %	6.13 %	4.84 %	15.55 %
MEM_WBLOCK	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Total:	34.37 %	0.00 %	0.00 %	14.70 %	16.37 %	19.01 %	15.55 %

This table shows that 34.37 % of all transfers finished normally. Nearly 20 % of all transactions were terminated with Retry. This is a very high value for PCI-X transactions and a reason for poor performance.

More than 30 % of all transactions were terminated with disconnects (Single Data Phase Disconnect and Disconnect at Next ADB).

Disconnects can only occur after at least one data cycle is completed, otherwise it would be graded as a retry. This again is evidence for the requester not being responsible for the poor performance.

As the completer is able to handle burst transactions, there are too many single data phase disconnects (14.70 %).

This table also shows that more than 15 % of Memory Read Block transactions were terminated with split response. The efficiency would increase if the device would use this command more often.

To identify the device that terminated the transactions, refer to the termination statistics (report section 8.4):

```

8.4 Termination Statistics
-----
8.4.1 Split Statistics
-----
No of split sequences ..... 495
Average first word latency of split transactions ..... 82 clocks
Average overall length of split transactions ..... 180 clocks
Average time overhead of split transactions ..... 86.91 %

8.4.2 Termination Burst Histogram, observed per Transaction
-----
Transferred      | Transfer was stopped by ...
Bytes            | Initiator      | Target
-----
    0 - 127      |      23.56 %   |      60.26 %
   128 - 255    |      6.85 %   |      5.37 %
   256 - 383    |      3.96 %   |      0.00 %
   384 - 511    |      0.00 %   |      0.00 %
   512 - 639    |      0.00 %   |      0.00 %
  2560 - 2687   |      0.00 %   |      0.00 %

... continued ...

 3712 - 3839    |      0.00 %   |      0.00 %
 3840 - 3967    |      0.00 %   |      0.00 %
 3968 - 4095    |      0.00 %   |      0.00 %
 4096+         |      0.00 %   |      0.00 %

-----
      Sum       |      34.37 %   |      65.63 %
    
```

In the termination burst histogram (report section 8.4.2) is listed which bursts were terminated by which device. The last row shows the sum for all bursts.

In this example, the termination burst histogram indicates that transactions were more often terminated by the completer than by the requester.

From the last row, it can be seen that all “normal” terminations (representing the successful transactions) were caused by the requester (34.37 %). This value is the same as in the Command Termination table (report section 8.2.4).

All other terminations were caused by the completer.

As a result, these two tables show that the completer stops too many transfers by retries and disconnects at single data phase.

Summary of the Performance Optimization Example

The selected requester-completer pair in this example was the PCI-X card as “MyCompleter” and “All Requesters”. The completer was found to be the device with the worst performance in the system under test.

After analyzing the data traffic to the device under test in detail with the Agilent E2929A/B PCI-X Performance Optimizer, the following conclusions can be drawn:

- The completer answers too many transactions with a retry.
- Many transactions are terminated by the completer with a single data phase disconnect, even though the completer is able to handle bursts.
- Furthermore, the completer inserts wait cycles (initial latencies).

If the performance of the device under test is improved, the PCI-X performance of the whole system will increase.

Reusing the Test Setups and Results

In the large field of system development, a testing environment is not very useful if you cannot document your test results or communicate them to others. Therefore, the Agilent E2929A/B Opt. 200 Performance Optimizer provides features for exporting the used test settings as well as the achieved measurement results in several formats.

Some of the exported data is saved in ASCII code and can be viewed with any text editor. Other files—like waveform files—need the software of the Agilent E2920 series to be displayed or reused.

NOTE You can view test results that were done with any testcard of the Agilent E2920 series. For this purpose, go to the configuration dialog box, switch to demo mode and select the correct testcard before opening the data files.

Exporting Data Files How to export the files produced by the Agilent E2929A/B Opt. 200 Performance Optimizer is described in *“Printing and Exporting Test Results”* on page 74.

Reusing Data Files How to view or reuse previously exported files is explained in *“Reusing Previously Saved Data”* on page 76.


Printing and Exporting Test Results

After successfully running the test measurements, you usually need to document the results for later use. A few possible situations for the need of previous test data are:

- You want to modify the configuration of the system under test, run the test again, and compare the results.
- You need to run tests on different systems and compare the results.
- You need to communicate the test results to colleagues or a device vendor or manufacturer.
- You want to rerun the same test after the device vendor or manufacturer has redesigned the product.

Printing the Test Results

If you need to present the results of the test measurements without opening files on a PC, you can print them out on paper. This applies for all tabs of the Performance Setup window and the Performance Charts window as well as for the report in the Performance Report window.

If you want to print the contents of a certain window, activate the desired view and click the Print button  to print the current view. Alternatively, you can also select the *Print* item from the *File* menu in this window.

For the different windows, note the following:

- Performance Setup window
Only the current view of the window is printed. If not all contents appear on the window tab, enlarge the window to display—and print—the complete information.
- Performance Charts window
Only the current view of the window is printed. If not all contents appear on the window tab, enlarge the window to display—and print—the complete information.
The charts only give an overview of the data found in the report. For more detailed performance information, printing the report is the preferable option.

- Performance Report window

When you print the report, the entire report is the complete report as listed in the report window, not only the currently displayed range.

Exporting the Report

The report of the Agilent E2929A/B Performance Optimizer can be saved as a text file for later use. This file is saved in ASCII format and can then be read with any text editor.

To export the report file to a text file, use the *Export to File* item from the *File* menu in the Performance Report window.

Exporting the Trace Memory

The performance of your system under test is calculated based on bus traffic recorded in the trace memory. In order to repeat the measurements or to make new calculations on the same data, you need to save the trace memory contents to a file. The trace memory contents are saved with the trigger information and device information.

To save the contents of the trace memory to a file, select *Save to file* from the *File* menu in any of the following windows:

- Performance Setup window
- Performance Report window
- Performance Charts window
- Waveform Viewer window, Bus Cycle Lister, and Transaction Lister window (all part of the Analyzer software)

The file will be saved as a waveform file (with the suffix *.wfm*). This file is in ASCII format. Nevertheless, it is recommended to view it as a waveform diagram with the Agilent E2929A/B software.

Saving the Test Settings

For presenting your test results, it is important to document the test settings that were used. This applies for all types of tests including performance measurements. Therefore, for all possible test scenarios, the way to export the test settings is identical. In the Agilent E2920 Main Window, choose one of the following options:

- Click the Save button (with the disk icon) or select *Save* from the *File* menu.

This action will save the current settings to the setup file (.*bst* file).

- If no setup file is yet in use or if you select *Save As* from the File menu, a file dialog box opens. Here you can select a name for the setup file and save all settings to this file.

The setup file (.*bst* file) contains all information that was entered in the Graphical User Interface (GUI). This includes the used hardware and software, the properties of the PCI-X bus, and the connected requester and completer devices. Furthermore, this file contains all settings done regarding the device behavior, including their names, address spaces, and their bus, device and function numbers. All the settings needed for the performance measurements are stored in this file as well.

Reusing Previously Saved Data

The instructions found here explain how to reuse the different types of data files originating from previous test sessions.

For the analysis of previously saved data files, no Agilent Exerciser and Analyzer card is needed. In offline/demo mode, the software can simulate any testcard of the Agilent E2920 series.

It is assumed that the Agilent E2929A/B software is running in offline/demo mode and the PCI-X Performance Optimizer option is enabled.

Loading Setup Files The setup files (*.bst* files) contain the complete setup of the Agilent E2929A/B PCI-X Performance Optimizer. This includes the used test hardware and software, the properties of the PCI-X bus and the connected requester and completer devices. Furthermore, these files contain all settings done regarding the device behavior, their names, address spaces, and their bus, device and function numbers.

To reuse a setup file, click the *Load* item in the file menu of the main window and select the desired setup file (with suffix *.bst*) from the appearing dialog box.

Loading Waveform Files In some cases, it is of interest to run new calculations on previously saved bus traffic data. The bus traffic is recorded to the trace memory and can be saved as a waveform file (with suffix *.wfm*).

To import a waveform file, proceed as follows:

- 1 From the main window, activate any of the windows of the Performance Optimizer (Performance Setup window, Performance Report window, or Performance Charts window) by clicking the respective button in the *Performance* group.

- 2 From the *File* menu, select *Load*.

A message box opens, telling you that loading a waveform file can take several minutes to complete.

- 3 Click the *Continue* button.

A file dialog box opens where you can locate the waveform file.

- 4 Select the correct waveform file (suffix *.wfm*) and click the *Open* button.

NOTE Loading a waveform file always causes the performance measurements to be recalculated with the current test settings. When finished, the results are presented in the two result windows of the Performance Optimizer. Thus, you basically do not need to separately save the text files containing the report.

Loading Report Files The report files contain the results of previous tests in as text. They are calculated from the data found in the respective setup files and waveform files. Report files cannot be loaded into the GUI directly. You can use any text editor to view them.

Going Further Into Details

The Agilent E2929A/B Opt. 200 Performance Optimizer is a software tool that is very flexible and easy to handle. It can be used within basically any PCI-X system to examine any device that communicates by using the PCI-X bus.

The number of possible test scenarios, PCI-X configurations, and possible difficulties with the data traffic is far too large to list here completely. Thus, only a few general hints can be given, for example, where to search for traffic problems and their possible source.

Verifying Good Performance

When examining your system under test, your first step will always be to get a general overview of the performance of the complete system as well as of the selected requester-completer pair.

Assuming you properly set up the test, the basic information is found in the Performance Chart window. The different charts present the performance of the selected devices:

- If the selected requester-completer pair is *All Requesters -> All Completers*, the window displays the different performance properties of the whole system under test.
- If the requester-completer pair is selected that you want to examine in particular, the charts in the window only contain the information for this pair.

You can compare the two results to see whether existing performance problems affect only the selected devices or the whole system.

Performance in the Presented Charts

Within the different charts of the Performance Charts window, you can view all the properties that concern the performance. The following list shortly summarizes how to identify good performance of the observed devices.

- **Bus Utilization**
A good performance is indicated by a high data fraction (green) and little overhead (red and yellow). Also, the bus utilization expresses how much of the total bus time was actually used by the selected devices, and how much was used by others or was idle (blue).
- **Split Statistics (on the *PCI-X Usage* tab)**
Good performance is indicated by a completer device that signals split response instead of retry. Splitting transactions allow other devices to use the bus until the split transaction is completed.
- **PCI-X Efficiency (on the *PCI-X Usage* tab)**
The efficiency is a measure that directly represents the quality of the data communication between the regarded PCI-X devices. It is affected by every change of the behavior of the participating devices. Thus, optimizing the device behavior will result in the increased efficiency.
- **PCI-X Throughput (on the *PCI-X Usage* tab)**
The maximum value can vary between 132 MByte/s in a 33 MHz/32-bit system, 528 MByte/s in a 66 MHz/64-bit system up to 1014 MByte/s in a 133 MHz/64-bit system.
- **Command Usage (on the *PCI-X Usage* tab)**
The usage of I/O commands should be avoided where possible, because they consume CPU time.
Also very important for good performance is a high fraction of data phases for each command and as little overhead as possible.
- **Burst Length (on the *Burst Usage* tab)**
For good performance, the preferred burst length should be adjusted to the requirements of the participating PCI-X devices. For instance, if the size of a cache line is eight dwords, the appropriate burst length to write into this memory would be eight as well.

Identifying Design Rule Violations

A very important factor for good performance of a PCI-X system is whether all devices always stick to the rules for efficient data traffic (shortly introduced in “*Rules for Proper PCI-X Design*” on page 15). Rule violations in this context do not cause errors. It merely means that devices that follow these rules achieve the best possible performance in any PCI-X system.

Long Bursts The most important rule for high performance is the use of long bursts. The longer a burst is, the better is the ratio between data phases and address phases. To check the distribution of the used burst lengths, see the *Burst Usage* tab in the Performance Charts window.

Plain Memory Commands Apart from that, memory commands should be used instead of I/O commands whenever possible. The usage of I/O commands consumes CPU time. Furthermore, they cannot be used in 64-bit transfers, but require 32-bit transfers.

- Use *Memory Read* to read less than one cache line from a memory.
- Use *Memory Write* to write less than one cache line into a memory.

For improvements regarding the completer behavior, like reducing latencies or avoiding wait states, see “*Improving Completer Behavior*” on page 83.

Replacing Cards or Improving Behavior

Depending on the motivation for your system or device tests, your options for improvement might be limited.

In case you are a device developer and testing your own PCI-X chipset or card design, you can verify the efficient behavior in any environment by checking the rules. And when you find room for improvement, you can redesign your product.

However, if the aim is to find the optimum combination of a range of existing devices, you only can check whether they do keep to these rules. Then you can choose the set of devices that produce the best results instead of redesigning a device.

Identifying Priorities for Improvements

Even if there are several options at hand, you still need to know how valuable the possible changes in your system are. Or in other words, the importance of a device must be taken into account before deciding on the way of improvement.

For example, the amount of bytes transferred between a particular device pair may be very small compared to the total amount of bytes transferred via the bus during the test.

In this case you need to find out whether this has been caused by the selected test setup or benchmark. If you conclude that the device hardly participates in PCI-X traffic under realistic circumstances, an expensive improvement of this device might not be worth the effort.

When testing your system with fairly realistic data traffic, read the “*Bus Users Overview (Report Section 6)*” on page 102 to see which devices have to handle the majority of the traffic.

Improving Completer Behavior

In many cases, traffic problems occur in PCI-X systems because a requester attempts to initiate a transfer to or from a completer that cannot handle the PCI-X command, the burst length, and so forth. In such cases, it makes sense to improve the completer behavior so that it can communicate at the highest speed.

Minimize First Word Latencies

One of the basic rules for speeding up the traffic is by minimizing the *First Word Latency*. The *First Word Latency* is the time from the beginning of the address phase until the beginning of the first data phase.

Large first word latencies indicate slow address decoding of the completer and/or many initial waits, for example due to locked completer resources.

Proper PCI-X design provides rapid replies between the devices and avoids the insertion of waits during transactions.

You can use the latency diagram (report section 8.5) to see for which burst lengths additional bus clocks were inserted.

Command Termination

To achieve good performance with your PCI-X devices, it is very important to make the completer capable of all possible types of transactions. Especially when transferring long bursts using extended bus commands, the requirements to the completer's abilities are high. If a transaction cannot be processed by the completer for some reason, it is terminated immediately.

To view the different types of transaction terminations, see the “*Command Termination, Observed per Transactions (Report Subsection 8.2.4)*” on page 112.

Types of Transaction Terminations

This analysis shows how many of the transactions in the traffic were terminated normally, or prematurely with a retry or with a disconnect.

- Accept

This is the preferred command termination. It means that the transaction was finished successfully as initiated by the requester, or that the completer-target signals a split response.

- SDP Disconnect

The completer-target signals a disconnect at single data phase.

- Retry

The completer-target signals a retry.

- TAbort

The completer-target signals a target abort.

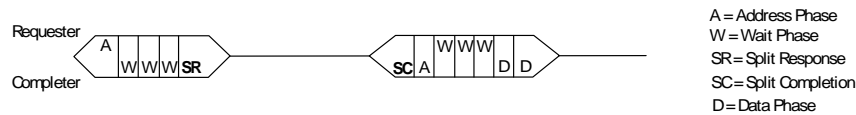
- Disconnect at Next ADB

The completion is disconnected at every n-th allowable disconnect boundary (ADB).

Split Response

With signaling split response, the completer-target indicates that the transaction will be completed as split transaction. The completer-target is permitted to terminate any transaction with split response except a memory write transaction.

The following figure shows how a split response is typically performed on the bus:



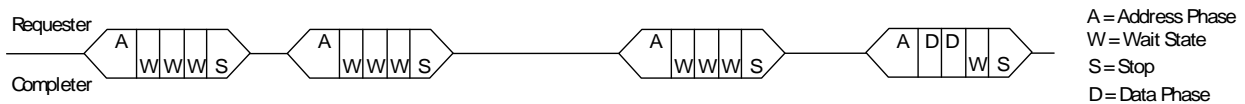
The requester tries to start a transfer and drives the address onto the bus. The addressed completer-target answers with split response by asserting TRDY#, deasserting DEVSEL# and keeping STOP# deasserted on the first data phase of the transactions.

Completer Retries

A completer terminates a transaction with a retry if it cannot supply or accept data, for example because it is too slow.

NOTE In general, to ensure a good PCI-X performance, completer devices should avoid signaling retries. Signaling split response instead (if possible) free the bus for other requesters to transfer data.

The following figure shows how retries are typically performed on the bus:



The requester tries to start a transfer and drives the address onto the bus. The addressed completer answers with three waits (as an example) and finally stops the transaction with a retry. From this behavior the requester notices that the completer cannot transfer any data at the moment. Unlike in PCI systems, the completer can not assume that the the requester tries again later.

Reasons for Completer Retries

In PCI-X systems, retries are only allowed under certain conditions, for example, if the transaction is a memory write and all of the buffers for accepting memory write transactions are currently full with previous memory write transactions.

Instead of forcing retries under these conditions, the completer could have inserted wait states until it is ready to transfer the data. This, however, would have locked the bus for other users. If the completer returns a retry as described, other users can access the bus in between, which increases the performance.

If a completer generates considerably many retries, and, therefore, throughput is low, the completer device should either be replaced or should be examined for possible improvement, like adjusting its retry frequency or implementing a larger buffer.

Completer Disconnects

If a completer encounters internal problems during the data phases of a transaction, it can terminate the current transaction with a completer disconnect.

A disconnect may occur either at a single data phase (SDP Disconnect) and at next allowable disconnect boundary (Disconnect at Next ADB).

- **SDP Disconnect**

The completer-target signals the intention to complete a single data phase and then disconnect the transaction by signaling Single Data Phase Disconnect.

Single data phase disconnection is intended for completers with address spaces that are generally not accessed by burst transactions.

- **Disconnect at Next ADB**

Completer-targets are permitted to disconnect burst transactions only on ADBs, where an ADB is 128-byte.

Reasons for Disconnects

Disconnects are issued by the completer during data transfer, for example due to the following reasons:

- **The completer is too slow to complete the data phase.**

The completer must indicate that it is unable to finish the data transfer within a certain number of clocks (latency). This ensures that the completer cannot block the bus for an extended time.

- **The completer memory does not understand the addressing sequence.**

A problem occurred in the address phase of a burst, the reason may be incompatibility due to different supported PCI-X specification revisions.

- **The transfer crosses the completer's address boundary.**

- **The burst memory transfer crosses the cache line boundary.**

Report Reference

The Agilent E2929A/B Opt. 200 Performance Optimizer presents the results of performance measurements in several forms. One of these is the report found in the Performance Report window. This report is divided into sections and subsections. Results in other windows, for example, the Performance Charts window, relate to particular sections in the report and are displayed with the respective section number.

The following report reference gives a short explanation of each report section, sorted by the section number. For detailed information on how to set up the tests for your purposes, refer to “*Setting Up a PCI-X Performance Optimizer Test*” on page 37. If you need to know how to run the tests and how to interpret the results, see “*Measuring System or Device Performance*” on page 47.

Typical Transactions and Sequences

For a better understanding of the terminology used in the performance report, typical PCI-X transactions and sequences are shown in “*Typical PCI-X Transactions and Sequences*” on page 88.

They are used to explain typical data transfers on the PCI-X bus. Please make yourself familiar with them first, because it will help to understand the meaning of the various measures used in the report.

Example Transfers

The example data transactions are introduced in “*Example Transfers*” on page 90.

Report Sections 1 to 7

The report sections 1 through 7 list general information and statistics about the PCI-X bus and the connected devices. Explanations of these sections are found in “*Bus Statistics (Report Sections 1 to 7)*” on page 92.

Report Section 8

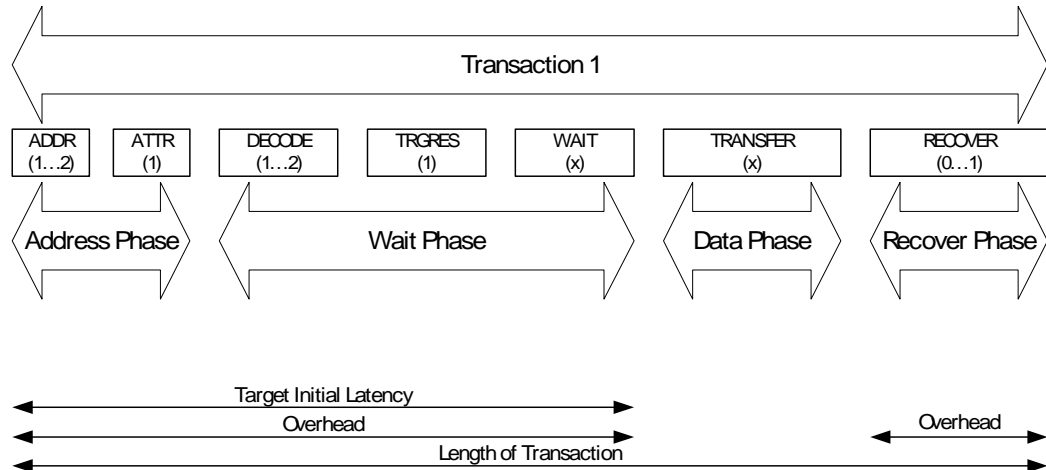
The report section 8 contains detailed statistics on the data communication of an arbitrary requester or completer device or a pair of devices. The explanations of section 8 are found in “*Requester-Completer Pair Measurements (Report Section 8)*” on page 104.

Definitions of Used Measures

Finally, a definition of the used terms and measures that are used throughout the sections of the report is found in “*Definitions of Used Measures*” on page 125.

Typical PCI-X Transactions and Sequences

The following figure shows a typical DWord transaction:



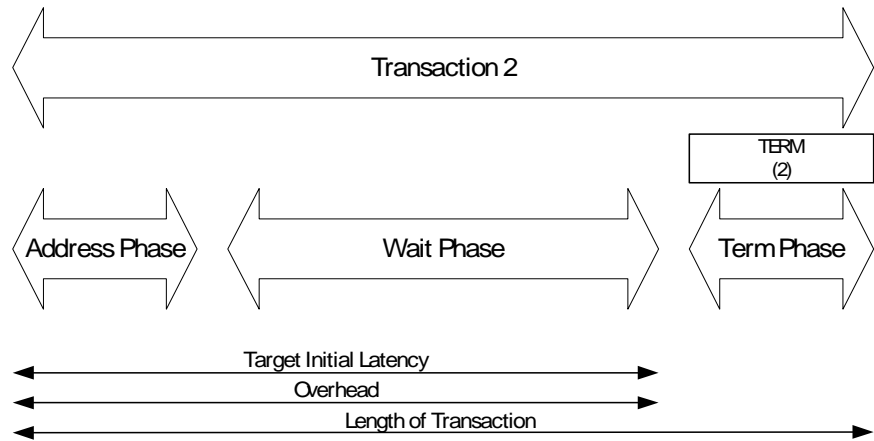
In this figure, the values in brackets indicate the minimum used clock cycles for the different phases.

The address phase covers all address and attribute cycles. The address can either consist of dual address cycles (DAC1 and DAC2) or of a single address cycle (SAC). The attribute phase always uses one clock cycle.

The wait phase covers all decoding, target response and wait cycles, the data phase covers all data cycles.

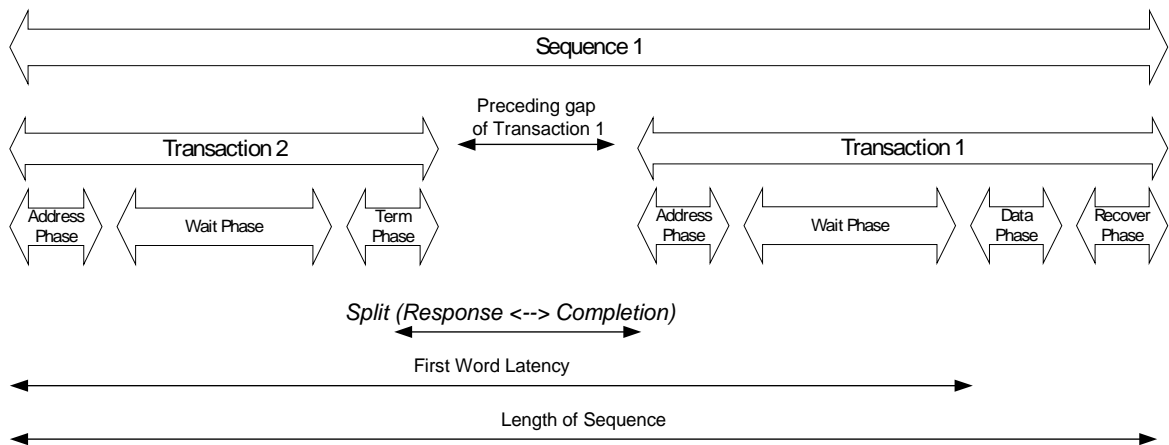
The recover phase covers up to one recover cycle, which is only used in DWord transactions.

Terminated Transaction The following figure shows a terminated transaction:



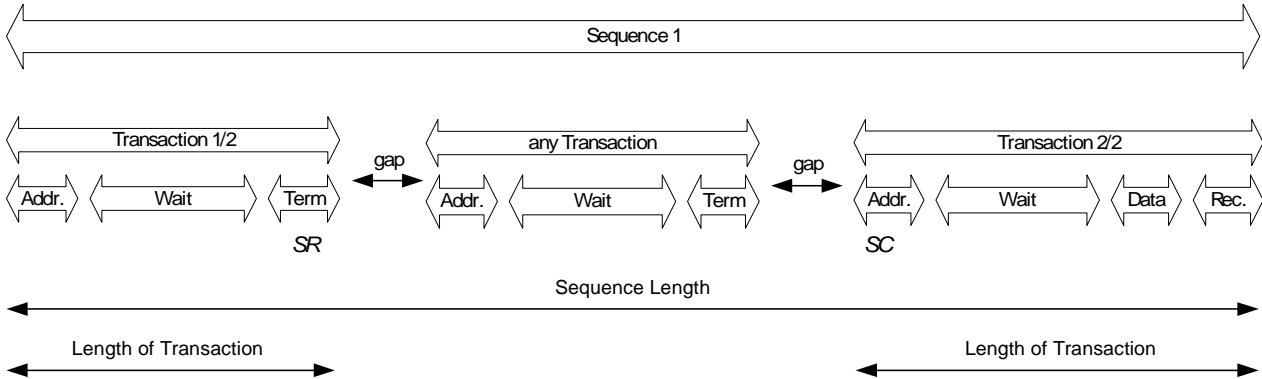
The transfer initiated by the requester is terminated by the completer either with a disconnect (Single Data Phase Disconnect or Disconnect at Next ADB), a Retry or a Split Response. Termination with Split Completion causes a split transaction which is explained in the following figure.

Split Transaction The following figure shows a sequence with a split transaction:



The time between split response and split completion can be used by other requesters to occupy the bus.

Sequence This is shown in the following figure, which displays a sequence with a normal and a split transaction:



The requester initiates a transfer and drives the address onto the bus (transaction 1). This transaction is terminated by the completer with Split Response (SR).

This allows other requesters to occupy the bus (any transaction). After termination of *any* transaction, the completer signals its intention to complete the split transaction with Split Completion (SC) and transfer the data.

NOTE This figure shows that the length of a sequence is *not* the sum of its transactions, because of preceding gaps.

Example Transfers

The example introduced here was used to run a performance test with the Agilent E2929A/B Opt. 200 Performance Optimizer. All figures in this report reference display sections from the test report calculated on this example, except where explicitly mentioned in the text.

This test example covers 229 clock cycles that contain 9 sequences and 11 data transactions as they are possible to occur on PCI-X busses. A 64-bit system is assumed.

In 64-bit systems, a maximum of eight bytes can be transferred per clock cycle. The actual number of transferred bytes is determined by the byte enable bits.

The following table give all relevant information about the transfers within a time of 229 clock cycles.

Sequence Number	Transaction Number	Command	Pre Gap	Overhead/ Clocks	Data Phase/ Clocks	Transaction Length	Byte Count/ Data Phase	Termination
1	1	Config Read	0	16	1	17	4	DISC_SDP
2	2	I/O Write	19	5	1	6	4	M_COMPL
3	3	I/O Read	2	14	1	15	4	M_COMPL
4	4	Mem Write Block	11	5	4	9	32	M_COMPL
5	5	Mem Read DWord	1	14	1	15	4	M_COMPL
6	6	Mem Read Block (1)	3	6	0	6	0	SPLIT_RESP
7	7	Mem Write	28	4	4	8	32	M_COMPL
8	8	Alias Mem Read Block	1	6	0	6	0	SPLIT_RESP
6	9	Split Completion (1)	34	4	8	12	32	M_COMPL
9	10	Alias Mem Write Block	1	4	4	8	32	M_COMPL
8	11	Split Completion	16	4	8	12	32	M_COMPL
		(Sum)	116	81	32	113	176	

The 81 overhead cycles are further divided into:

Requester Address Overhead: 18 cycles
Completer Address Overhead: 4 cycles
Completer Waits Overhead: 47 cycles
Completer Terms Overhead: 4 cycles

The example does not take into account which devices communicate via the PCI-X bus. However, the Agilent E2929A/B Opt. 200 Performance Optimizer can evaluate the traffic of different devices separately. Requesters are identified by their bus number, device number and function number, completers by their address space.

Bus Statistics (Report Sections 1 to 7)

The report sections 1 through 7 contain general information on the system under test, the test coverage, and basic bus statistics.

Additionally, there are subsections that characterize the performance of the complete system under test using values such as the bus throughput, the bus efficiency, the bus utilization, the bus users overview, and the interrupt protocol.

The top of the report contains the software version together with the date and time of the test.

General Information (Report Section 1)

This report section provides information about the system under test and is divided into the following subsections:

Test Base (Report Subsection 1.1)

Report section 1.1 provides general information on the system under test: the test settings and information about the statistical base. It also contains a list of all identified requesters and completers.

```

1 GENERAL INFORMATION
=====

1.1 Test Base
-----

Bus Speed ..... 66.67 MHz
Bus Width ..... 64 Bit
Requesters:
  1. All Requesters
  2. >Req_1          Bus:0xf1   Dev:0x05   Func:0x00

legend:
! = Requester defined by user and found in trace
? = Requester defined by user and not found in trace
> = New Requester found on bus

Completers:
1. All Completers          Addr: 0000000000000000\h
                           Size: ffffffff\h
                           I/O and Mem

Selected Pair:          All Requesters -> All Completers

```

In detail, there are:

- Bus Speed

According to the PCI-X specification, this value should be between 33 MHz and 133 MHz.

- Bus Width

The bus width is 32 or 64 bit.

- Identified Requesters

The first to be listed here is always *All Requesters*. After that, individual requesters are listed with their bus number, device number and function number. They are specially signed with “!”, “?” and “>” to identify the requesters according to the following:

- defined by user and found in trace
- defined by user and not found in trace
- new requester found on bus

- Identified Completers

The first to be listed here is always *All Completers* with the base address *00000000h* and the size of the total memory space available for addressing completers.

If any completers were set up to be identified in the test, they would be listed here.

- Selected Pair

Here the device pair is listed, for which the data traffic was evaluated in the test. In the example test, the selection *All Requesters -> All Completers* caused the complete traffic to be measured.

Statistical Base (Report Subsection 1.2)

The subsection 1.2 contains the statistical base. It tells the length of the test and the amount of data traffic recorded in the trace memory. This is important for estimating how accurate the calculated results are. Recording a large amount of data traffic gives a higher probability of gaining representative data and, therefore, the results are more reliable.

The values in this section are given in absolute numbers. All other report sections (except the statistical base of the requester-completer pair report and the split statistics in section 8) contain relative numbers to keep the reports of different test sessions comparable.

```

1.2 Statistical Base
-----
Test covered .....                229 clocks
Test covered .....                0.00000343 sec
No of captured address/attr cycles .. 22 clocks
No of captured data cycles .....    32 clocks
Number of bytes transferred .....   176 bytes
Number of sequences.....            9
Number of transactions.....         11

```

In detail, there are:

- Test covered

Absolute length of the test run both in clock cycles and seconds. The number of clocks divided by the test length is the average bus speed, in this case 66.67 MHz.

- No of captured address/attr cycles

Total number of initiated transactions, no matter whether they were completed successfully or not.

- No of captured data cycles

Number of data phases recorded during the test. The ratio of address to data phases is important for the performance evaluation.

- Number of bytes transferred

Tells how many bytes actually were transferred between the requesters and completers.

Only those bytes in the data phases are counted that are indicated by the byte enable bits.

- Number of sequences

Total number of sequences that occurred during the test. A sequence consists of at least one transaction.

- Number of transactions

Total number of transactions that occurred during the test.

Invalid Measures (Report Subsection 1.3)

The subsection 1.3 contains invalid measures. It tells the number of incomplete measures and the number of invalid cycles.

The values in this section are given in absolute numbers. All other report sections (except the statistical base of the requester-completer pair report and the split statistics in section 8) contain relative numbers to keep the reports of different test sessions comparable.

NOTE The values presented in the following report section do not refer to the example data traffic. In the example, no incomplete sequences and no invalid cycles occurred and therefore, this table is empty in the report.

1.3 Invalid Measures	

No of incomplete sequences.....	2
Total no of invalid clocks.....	47

In detail, there are:

- No of incomplete sequences

A sequence is incomplete if at least one transaction is missing to complete the sequence. (Because any cycle that appear in an incomplete sequence will be counted as *IDLE*, this sequence will not be listed in the *Statistical Base* (report subsection 1.2.)).

The cause for an incomplete sequence can be: A response to a request appears on the bus which happened *before* traffic is recording into the trace memory. It is also possible that the trace memory stops recording and there are still outstanding responses to a sequence.

- Total number of invalid clocks

Invalid clocks are all cycles from incomplete sequences and any cycle of transactions that have errors or are incomplete.

Incomplete transactions can appear at the start or end of the trace memory. Errors in transactions are due to a wrong state transition or other wrong signal patterns that appear on the bus.

Basic Bus Statistics (Report Section 2)

This report section contains some information on the performance of the complete system under test.

```

2 BASIC BUS STATISTICS
=====
PCI-X Throughput ..... 48.8639 Megabytes/sec
                    ..... 390.9112 Megabits/sec
PCI-X Utilization ..... 49.34 %
PCI-X Efficiency ..... 19.47 %
Average First Word Latency ..... 25 clocks
Average Split Rate ..... 22.22 %

```

In detail, there are:

- PCI-X Throughput

The throughput is the amount of data transferred per time. In the example, 176 bytes are transferred in 229 bus cycles, 15 ns each. The resulting throughput is 48.86 MByte/s or 390.9 Mbit/s.

- PCI-X Utilization

The utilization is the ratio between used bus cycles and unused (idle) cycles.

NOTE

In this example, the value of used bus cycles can be seen in *“Statistical Basis (Report Subsection 8.1)” on page 105*, because in the selected requester-completer pair represents the whole generated traffic.

In the example, during 113 clocks—of a total of 229 clocks—the bus was busy. Therefore, the utilization is 49.3 %.

- PCI-X Efficiency

The efficiency indicates the quality of the communication on the bus. It is derived from throughput and utilization. In the example, 176 bytes were transferred within the 113 busy clock cycles. Up to eight bytes can be transferred per cycle. This makes a theoretical maximum of 904 bytes during the 113 cycles, out of which 176 bytes are 19.47 %.

- Average First Word Latency

The average first word latency is the ratio between the sum of first word latency cycles of all sequences and the number of all sequences.

- Average Split Rate

The average split rate is the ratio between the number of split transactions and the number of all sequences. In this example: Two split transaction during nine covered sequences, which provides an average split rate of 22.22 %.

Bus Throughput Statistics (Report Section 3)

The report section 3 is called Bus Throughput Statistics and holds a table where the data throughput between all requester-completer pairs is analyzed. The values can be compared to find the devices with highest or lowest throughput.

3 BUS THROUGHPUT STATISTICS	
=====	
Requester	Completer Throughput (MBytes/sec)

	All Comp
	leters

Req_1	48.86
All Requesters	48.86

The requesters (left column) and the completers (in the top row) are the same that are listed in “*Test Base (Report Subsection 1.1)*” on page 93. The entries *All Requesters* and *All Completers* always appear, the others were specified by the user.

Efficiency Statistics (Report Section 4)

This report section presents the efficiency of the data communication on the PCI-X bus. First the overall efficiency of the system under test is displayed. Then in subsection 4.1 a table shows the efficiency of the traffic between all requester-completer-pairs. Finally, in 64-bit systems, subsection 4.2 presents the distribution of 32-bit transfers and 64-bit transfers.

```

4 EFFICIENCY STATISTICS
=====
Byte Enable Efficiency ..... 68.75 %
Time Efficiency ..... 28.32 %
    
```

In detail, there are:

- **Byte Enable Efficiency**

The byte enable efficiency indicates how many bytes in average the devices could transfer via the PCI-X bus per clock cycle. The maximum is eight bytes (64-bit).

In the example, 176 bytes were sent during 32 data cycles. 176 bytes out of a maximum of $8 * 32 = 256$ bytes results in 68.75 %. This reveals that more than the half of all transactions used all eight bytes of the bus.

NOTE From this measure a device can be found that does not use the full bus width, for example, because it uses only 16-bit cache line size.

- **Time Efficiency**

The time efficiency shows the average fraction of clock cycles in a transaction that are data cycles. If an 8-cycle transaction contains 4 data phases, the time efficiency is 50 %.

Requester-Completer Efficiency

The table in subsection 4.1 shows the efficiency of the data traffic between all requester-completer-pairs. The values can be compared in order to find the device pair with the lowest performance.

```

4.1 Requester Completer Efficiency
-----
                                     | All Completers
-----|-----
Requester_1 | 19.47 %
All Requesters | 19.47 %
    
```

The completers (in the top row) and the requesters (left column) are the same that are listed in “*Test Base (Report Subsection 1.1)*” on page 93. The entries *All Requesters* and *All Completers* always appear in this table, the others were specified by the user during test setup.

64-Bit Bus Statistics If you are testing a 64-bit system, the performance measurements also investigate the device behavior regarding 64-bit data transfers.

4.2 64-BIT BUS STATISTICS	

Pure 32-Bit data transfers	62.50 %
64-Bit data transfer tried but refused...	0.00 %

32-Bit data transfers (sum)	62.50 %
64-Bit data transfers	37.50 %

The 64-bit bus statistics list:

- Pure 32-Bit Data Transfers

These are the transactions that were initiated by the requester as 32-bit transfers and completed as such.

- 64-Bit data transfer tried but refused

These are the transactions that were initiated by the requester as 64-bit transfers, but rejected by the completer, and then implemented as 32-bit transfers.

- 64-Bit Data Transfers

These are the transactions that were successfully completed as 64-bit transfers.

Bus Utilization Statistics (Report Section 5)

The bus utilization statistics informs about the fraction of time the bus was active or idle. First the statistics show the total activity of the PCI-X bus. Then, in subsection 5.1, a table is displayed that analyzes the bus usage of the specified requester-completer-pairs. The values are all given as the fraction of the total number of clock cycles in the test. Thus, they are directly comparable to find the requester-completer-pairs that mainly occupy the bus.

```

5 BUS UTILIZATION STATISTICS
=====
Overhead Utilization ..... 35.37 %
Data Phase Utilization ..... 13.97 %
-----
PCI-X Utilization ..... 49.34 %

5.1 Requester Completer Utilization
-----
                | All Comp
                | leters
-----
Req_1           | 49.34 %
All Requesters | 49.34 %
    
```

In detail, there are:

- Overhead Utilization

The overhead utilization indicates how much of the time the bus was occupied for transferring overhead data.

In the example, the transfer use 32 clocks for data transfer during 113 *busy* clocks. This means an overhead of 81 clocks. This results in 35.37 % overhead utilization: 81 of 229 *covered* clocks were used for overhead transfer.

- Data Phase Utilization

The data phase utilization shows how much the bus is used for transferring data phases.

In the example, the data utilization is 13.97 % (32 out of 229 bus cycles were data phases).

- PCI-X Utilization

The PCI-X utilization is the sum of overhead utilization and data phase utilization and is covered in “*Basic Bus Statistics (Report Section 2)*” on page 97.

Bus Users Overview (Report Section 6)

This report section contains a table showing the Bus Users Overview. The values are derived from the bus utilization (report section 5.1) and the efficiency (report section 4.1). They are an index for the device performance.

6 BUS USERS OVERVIEW	
=====	
Calculated as: Utilization divided by Efficiency	

	All Comp letters

Req_1	2.53
All Requesters	2.53

This index is calculated as utilization divided by efficiency. It expresses how efficient a device uses the bus while keeping the utilization low. The higher this index is, the higher the utilization and the lower the efficiency of the communication of the respective requester-completer pair.

A high index indicates a device with a poor PCI-X design. You can compare the indices in the table to find the device with the highest and lowest performance. The performance index of the example transfer is 2.53 (49.34 % divided by 19.47 %).

The requesters (left column) and the completers (in the top row) are the same as listed in “*Test Base (Report Subsection 1.1)*” on page 93.

Split Transaction Statistics (Report Section 7)

The split transaction statistics give an overview about all values needed to evaluate the split behavior of the PCI-X system under test.

Split Transactions A split transaction is a transaction terminated by the completer with the intend to complete this transaction as soon as possible. This report shows all relevant values used to characterize the split behavior of the system under test.

```

7 SPLIT TRANSACTION STATISTICS
=====
No of split sequences..... 2
Average first word latency of split transactions.. 84 clocks
Average overall length of split transactions..... 92 clocks
Average time overhead of split transactions..... 91.30 %

7.1 Pending Requests over time
-----
Req. | 0 | 1 | 2 | 3 | 4 | 5 | 6+ |
-----
%time| 25.24 % | 63.68 % | 11.07 % | 0.00 % | 0.00 % | 0.00 % | 0.00 % |
    
```

In detail, there are:

- No of split sequences
Displays the number of all sequences that contain split transactions.
- Average first word latency of split transactions
This value shows the average number of clocks passed by until the first data is transferred during a sequence.
- Average overall length of split transactions
This value shows the average length of all covered split sequences.
- Average time overhead of split transaction
The time overhead of a split transaction is the time of the overall length of this split transaction that is not used to transfer data.

Furthermore, this subsection 7.1 lists all pending requests over time. For example, the time in which one request is pending is 63.68 % of the overall bus time.

Requester-Completer Pair Measurements (Report Section 8)

In report section 8 and its subsections, the detailed analysis of the data traffic between a particular requester-completer pair is presented. Additionally, the analysis can be restricted on the traffic in a certain direction, considering either only data reads or data writes, or both.

The Selected Device Pair The device pair must have been selected in the test setup as described in “*Selecting a Requester-Completer Pair*” on page 43. Otherwise, the default pair and traffic direction will be used: “All Requesters” and “All Completers”, “Read and Write”, resulting in the complete traffic being analyzed.

The Source of the Figures Through this report reference, the short example is used that is introduced in “*Example Transfers*” on page 90. All figures display sections from the test report calculated on this example. The data transfers in the example are assumed to be read/write transactions of the selected device pair and are, thus, covered within this report section 8.

The Header of Report Section 8

The header of report section 8 shows the selected device pair and the traffic direction from the requester's view. The figure below shows an example how the header may look like.

```
8 REQUESTER - COMPLETER PAIR:
All Requesters <-> All Completers      (Read Write)
=====
```

Statistical Basis (Report Subsection 8.1)

This report section presents the statistical base of the traffic of the device pair. To gain representative results, the calculations should be based on a large amount of data traffic.

```
8.1 Statistical Base
-----
This pair was busy for .....          113 clocks
This pair was busy for .....          0.00000169 sec
No of captured address/attr cycles .   22 clocks
No of captured data cycles .....       32 clocks
No of bytes transferred .....          176 bytes
No of sequences .....                  9
No of transactions .....               11
```

In detail, there are:

- The time, this device pair was busy, both in clock cycles and in seconds.
- The number of address and attribute cycles that the requester sent to address this particular completer.
- The number of data phases that occurred in the traffic between the device pair.
- The number of bytes that actually were transmitted between the requester and the completer during the test period.
- The number of sequences and the number of transactions.

Comparing these values to the statistical basis of the total bus traffic listed in “*Statistical Base (Report Subsection 1.2)*” on page 94 allows conclusions on how intensively this device or device pair takes part in the communication on the PCI-X bus.

Bus Utilization (Report Subsection 8.2)

This report subsection gives a simple overview of the types of bus cycles the bus has been used for by the selected requester-completer pair.

8.2 Bus Utilization	

Overhead	30.13 %
Split Overhead	5.24 %
Data Phases	13.97 %

Sum of Utilization	49.34 %

In detail, this subsection contains:

- Overhead

The time that the device pair occupied the bus without transferring data. This value does not cover transactions that were terminated by the completer with a retry.

- Split Overhead

All phases that are not data phases of sequences that were split.

NOTE

The address and wait phases of the first transaction are added to Overhead.

- Data Phases

The data phases, finally, represent the amount of time during which data was transferred between the requester and the completer.

- Sum of Utilization

The sum of the three types of clock cycles mentioned above.

For good performance, the overhead values should be as small as possible. The bus occupation for data phases, however, may be large to transfer large amounts of data.

For more detailed statistics on the bus utilization of the selected requester and completer, refer to the report subsections 8.2.1 to 8.2.5.

Data Phase (Report Subsection 8.2.1)

The data phase analysis provides information on the number of bytes sent per data phase, the average burst length and the distribution of 64-bit and 32-bit transfers.

8.2.1 Data Phase	

0 Bytes	0.00 %
1 Bytes	0.00 %
2 Bytes	0.00 %
3 Bytes	62.50 %
4 Bytes	0.00 %
5 Bytes	0.00 %
6 Bytes	0.00 %
7 Bytes	0.00 %
8 Bytes	37.50 %

Average Byte Enable Efficiency	68.75 %
Average Burst Length	3.56 data phase(s)

Bytes per Data Phase Within the 64-bit system, the data phase analysis shows the byte enable usage for 8-byte enable bits.

The byte enable bits on the PCI-X bus define how many and which of the data bytes contain useful data. The list above resolves the distribution of the data phases with a different number of data bytes. The sum of the five values equals 100 %, because every data phase is covered here.

In this example, the average byte enable efficiency turns out as 68.75 % when sending either four or eight bytes per data phase. The average burst length of 3.5 data phases per transaction is very low and indicates devices that probably cannot handle longer bursts.

Average Byte Enable Efficiency The average byte enable efficiency is calculated as the number of transferred bytes divided by the maximum number transferable (all data phases holding four bytes in a 32-bit system). In the example used for the report, the byte enable efficiency is 81.81 %. This results from 180 transferred bytes divided by 4 times 55 data cycles.

Average Burst Length Also included in this subsection is the average length of the transactions in the traffic. Longer bursts usually yield better performance results, because many data phases can be sent with only one address phase.

32-Bit and 64-Bit Transactions The last rows of this report subsection contain the distribution of 32-bit and 64-bit transfers.

Pure 32-Bit Data Transfers	62.50 %
64-Bit Data Transfer tried 64 but refused ...	0.00 %

32-Bit Data Transfers (sum)	62.50 %
64-Bit Data Transfers	22.64 %

The different cases distinguished here are:

- **Pure 32-Bit Data Transfers**
 These are all transactions that were initiated by the requester as 32-bit transactions.
- **64-Bit Data Transfer tried 64 but refused**
 This figure covers all transactions that were initiated by the requester as 64-bit transactions, but were performed as 32-bit transfers due to the completer's requirements.
- **32-Bit Data Transfers (sum)**
 This is the total fraction of 32-bit transactions, calculated as the sum of the two figures above.
- **64-Bit Data Transfers**
 This fraction covers all transactions that were initiated and performed as 64-bit transactions.

Overhead (Report Subsection 8.2.2)

This report section analyzes the overhead in the recorded traffic. Every clock cycle in a transaction that is not a data phase is considered as overhead: address phases, waits, termination and recover cycles. The table reveals which device caused what type of overhead.

8.2.2 Overhead			

Phase	Overhead caused by		
	Requester	Completer	Sum

Addr	22.22 %	4.94 %	27.16 %
Waits	4.94 %	58.02 %	62.96 %
Terms	0.00 %	4.94 %	4.94 %
Recover	3.70 %	1.23 %	4.94 %

Total	30.86 %	69.14 %	100 %

Addrphase includes: Addresscycles, Attributecycles			
Waitphase includes: Decodecycles, Target-Responsecycles			

Time Overhead Table The different figures found in the table are:

- Address Phases

This row covers the fraction of overhead cycles that are address phases. Address phases can only be inserted by requester devices.

- Waits Phases

This row holds the amount of initial wait states caused by the different devices.

- Termination Phases

Termination caused by the requester can be master completion, master abort, single data phase disconnect and disconnect at next ADB.

Termination caused by the completer can be target abort, split response and retry.

- Recover Phases

The recover cycle only appears in DWORD transactions. During this cycle, there is no action on the bus, but the bus is not idle as #FRAME and #IRDY are still active. This cycle is necessary, as the initiator needs one cycle to respond to #TRDY of the target.

- Sum

This row lists the total overhead caused by requester and/or completer. The total far to the right always equals 100 %, because every overhead cycle is covered in this table.

For a more detailed analysis of the different latencies in the data traffic, refer to the “*Latency Histogram, Observed per Sequence (Report Subsection 8.5)*” on page 121.

NOTE The figures in the table above were not taken from the report on the example transactions.

Command Usage, Observed per Sequence (Report Subsection 8.2.3)

The Agilent E2929A/B Opt. 200 Performance Optimizer records the data traffic on the system under test during a specified number of clock cycles. This includes the PCI-X bus commands that have been used for the communication between the different devices.

This report subsection lists the distribution of the different PCI-X bus commands as well as the amount of data transferred with these commands. The analysis allows conclusions on which and how much bus commands are used by the considered device pair.

8.2.3 Command Usage, observed per Sequence						
Command	Fraction of:				64bit data	Effi- ciency
	Used Clocks: Overh.	Data	Bytes moved	Split		
SPECIAL	0.00 %	0.00 %	0.00 %	0.00 %	--	--
IO_READ	12.39 %	0.88 %	2.27 %	0.00 %	0.00 %	3.33 %
IO_WRITE	4.42 %	0.88 %	2.27 %	0.00 %	0.00 %	8.33 %
RESERVED_1	0.00 %	0.00 %	0.00 %	0.00 %	--	--
RESERVED_2	0.00 %	0.00 %	0.00 %	0.00 %	--	--
MEM_RDWORD	12.39 %	0.88 %	2.27 %	0.00 %	0.00 %	3.33 %
MEM_WRITE	3.54 %	3.54 %	18.18 %	0.00 %	100.00 %	50.00 %
ALIAS_MEM_RB	8.85 %	7.08 %	18.18 %	50.00 %	0.00 %	22.22 %
ALIAS_MEM_WB	3.54 %	3.54 %	18.18 %	0.00 %	100.00 %	50.00 %
CONFIG_READ	14.16 %	0.88 %	2.27 %	0.00 %	0.00 %	2.94 %
CONFIG_WRITE	0.00 %	0.00 %	0.00 %	0.00 %	--	--
MEM_RBLOCK	8.85 %	7.08 %	18.18 %	50.00 %	0.00 %	22.22 %
MEM_WBLOCK	3.54 %	3.54 %	18.18 %	0.00 %	100.00 %	50.00 %
Total:	100 %	100 %	100 %	100 %		

For each of the available PCI-X bus commands, the following values are reported:

- The fraction of clocks used to transfer overhead with this command.
- The fraction of clocks used to transfer data with this command.
- The fraction of bytes moved with this command.
- The fraction of transactions that were split.
- The fraction of 64-bit data shows the percentage of the commands performed as 64-bit transactions.
- The overall efficiency of the transactions that used this command.

When developing a PCI-X device, it is advisable to use memory commands instead of I/O commands whenever possible. For existing devices, of course, it can only be examined whether they do follow these rules.

In the Performance Charts window on the *PCI-X Usage* tab, the *Command Usage* chart presents the contents of this table in a graphical way. See also “*PCI-X Usage*” on page 50.

Command Termination, Observed per Transactions (Report Subsection 8.2.4)

The command termination analysis shows what types of transaction terminations occurred for the different PCI-X bus commands.

8.2.4 Command Termination, observed per Transaction							
Command	Fraction of Terminations						
	M_ COMPL	M_ ABORT	T_ ABORT	DISC_ SDP	DISC_ ADB	RETRY	SPLIT RESP
INT_ACK	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
SPECIAL	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
IO_READ	9.09 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
IO_WRITE	9.09 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_1	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_2	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
MEM_RDWORD	9.09 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
MEM_WRITE	9.09 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
ALIAS_MEM_RB	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	9.09 %
ALIAS_MEM_WB	9.09 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
CONFIG_READ	0.00 %	0.00 %	0.00 %	9.09 %	0.00 %	0.00 %	0.00 %
CONFIG_WRITE	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
SCOMPLETION	18.18 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
MEM_RBLOCK	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	9.09 %
MEM_WBLOCK	9.09 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Total:	72.73 %	0.00 %	0.00 %	9.09 %	0.00 %	0.00 %	18.18 %

The displayed types of transaction terminations for the PCI-X bus commands are:

- Single Data Phase (SDP) Disconnect

This type of termination only occurs if the initiator does a burst read although the target does not support bursts. In this case, the target can send a single data phase and terminate it with a *DISC_SDP*.

Thereupon, the initiator performs the next burst read.

- Disconnect at next ADB (Allowable Disconnect Boundary)
Initiators and targets disconnect burst transactions on ADBs. The ADB refers to the next memory boundary on bus, which is 128 bytes.
- Retry
This column holds the fraction of data transfers that were terminated by the target with a retry. This means that the target was not ready to transfer the data.
- Split Response
If the target of the completer indicates that it will complete the transaction as a split transaction, the initiator of the completer will later send split completions. Terminating the split completions with *DISC_ADB* allows the initiator of the completer to send the data in more than one split completion.

NOTE Master and target abort are not covered in this subsection.

The last row of the table lists the sum of the respective columns. This is the total fractions of transfers terminated by a certain type.

A detailed analysis on the transaction terminations distributed over the burst length is found in the *“Termination Statistics (Report Subsection 8.4)”* on page 119.

Byte Count over Command, Observed per Sequence (Report Subsection 8.2.5)

This report subsection explicitly lists every data transaction between the observed device pair according to its byte count and the used PCI-X bus command (DWord commands are considered in a separate table).

8.2.5 Byte Count over Command, observed per Sequence						
Transferred Bytes	MEM_WR ITE	ALIAS_ MEM_RB	ALIAS_ MEM_WB	SCOMPL ETION	MEM_RB LOCK	MEM_WB LOCK
0	0.00%	0.00%	0.00%	--	0.00%	0.00%
1	0.00%	0.00%	0.00%	--	0.00%	0.00%
2- 3	0.00%	0.00%	0.00%	--	0.00%	0.00%
4- 7	0.00%	0.00%	0.00%	--	0.00%	0.00%
8- 15	0.00%	0.00%	0.00%	--	0.00%	0.00%
16- 31	0.00%	0.00%	0.00%	--	0.00%	0.00%
32- 63	11.11%	11.11%	11.11%	--	11.11%	11.11%
64- 127	0.00%	0.00%	0.00%	--	0.00%	0.00%
128- 255	0.00%	0.00%	0.00%	--	0.00%	0.00%
256- 511	0.00%	0.00%	0.00%	--	0.00%	0.00%
512-1023	0.00%	0.00%	0.00%	--	0.00%	0.00%
1024-2047	0.00%	0.00%	0.00%	--	0.00%	0.00%
2048-4095	0.00%	0.00%	0.00%	--	0.00%	0.00%
4096+	0.00%	0.00%	0.00%	--	0.00%	0.00%
Sum	11.11%	11.11%	11.11%	--	11.11%	11.11%
Transferred Bytes	IO_REA D	IO_WRI TE	MEM_RD WORD	CONFIG _READ	CONFIG _WRITE	
0	0.00%	0.00%	0.00%	0.00%	0.00%	
1	0.00%	0.00%	0.00%	0.00%	0.00%	
2- 3	0.00%	0.00%	0.00%	0.00%	0.00%	
4	11.11%	11.11%	11.11%	11.11%	0.00%	
Sum	11.11%	11.11%	11.11%	11.11%	0.00%	

The table indicates, for example, whether I/O commands are implemented, although plain read/write commands are preferable, and whether burst lengths are restricted to small values in reference to sequences. Restricted burst lengths can, for example, result from small internal buffers of the completer's PCI-X chipset.

The entries in this table refer to the example transactions introduced in “*Example Transfers*” on page 90. From the last rows of these tables you can see that nearly all commands (except Split Completion and Configuration Write) are used. All DWord transactions had a burst length of four, all burst transactions use always burst lengths between 32 and 63 transferred bytes.

These results are also shown as a graphic on the *Command tab* in the Performance Charts window. Refer to “*Command*” on page 52 for details.

Byte Count over Command, Observed per Transaction (Report Subsection 8.2.6)

This report subsection explicitly lists every data transaction between the observed device pair according to its burst length and the used PCI-X bus commands (DWord commands are considered in a separate table).

8.2.6 Byte Count over Command, observed per Transaction						
Transferred Bytes	MEM_WR ITE	ALIAS_MEM_RB	ALIAS_MEM_WB	SCOMPL ETION	MEM_RB LOCK	MEM_WB LOCK
0	0.00%	9.09%	0.00%	0.00%	9.09%	0.00%
1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
2- 3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
4- 7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
8- 15	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
16- 31	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
32- 63	9.09%	0.00%	9.09%	18.18%	0.00%	9.09%
64- 127	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
128- 255	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
256- 511	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
512-1023	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1024-2047	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
2048-4095	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
4096+	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Sum	9.09%	9.09%	9.09%	18.18%	9.09%	9.09%
Transferred Bytes	IO_REA D	IO_WRI TE	MEM_RD WORD	CONFIG_READ	CONFIG_WRITE	
0	0.00%	0.00%	0.00%	0.00%	0.00%	
1	0.00%	0.00%	0.00%	0.00%	0.00%	
2- 3	0.00%	0.00%	0.00%	0.00%	0.00%	
4	9.09%	9.09%	9.09%	9.09%	0.00%	
Sum	9.09%	9.09%	9.09%	9.09%	0.00%	

The table indicates, for example, whether only plain read/write commands are implemented, although extended commands are preferable, and whether burst lengths are restricted to small values. Restricted burst lengths can, for example, result from small internal buffers of the completer's PCI-X chipset.

The table indicates, for example, the fraction of split completions, and whether burst lengths are restricted to small values. Restricted burst lengths can result from small internal buffers of the completer's PCI-X chipset.

The entries in this table refer to the example transactions introduced in "Example Transfers" on page 90. From the last rows of these tables you can see that nearly a fifth of the burst transactions use split completion and use a burst length between 32 and 63 bytes.

In this table, you can further see that the memory read block transactions have a burst length of 0 bytes. This indicates that these transactions have been split.

Efficiency Statistics (Report Subsection 8.3)

The efficiency statistics and its subsections can be used to determine the efficiency of the data transfers between the selected requester-completer pair.

<pre> 8.3 Efficiency Statistics ----- PCI-X Efficiency of this pair 19.47 % </pre>
--

PCI-X efficiency of this pair is the overall efficiency of the complete data traffic of the selected device pair considered in the test.

The value shown in this subsection of the report is also displayed on the *PCI-X Usage* tab of the Performance Charts window. A description to this tab is found in "PCI-X Usage" on page 50.

Efficiency over Byte Count, observed per Sequence (Report Subsection 8.3.1)

This report subsection splits up the efficiency statistics for the different byte counts. The table shows which byte counts occurred in the observed traffic and how efficient they were. It also reveals the fraction of data phases and the fraction of bus clocks that were used for the bursts of the different lengths.

8.3.1 Efficiency over Byte count (observed per Sequence)			
Transferred Bytes	Efficiency	Data Fraction	Bus Fraction
0	--	0.00 %	0.00 %
1	--	0.00 %	0.00 %
2- 3	--	0.00 %	0.00 %
4- 7	3.77 %	12.50 %	46.90 %
8- 15	--	0.00 %	0.00 %
16- 31	--	0.00 %	0.00 %
32- 63	33.33 %	87.50 %	53.10 %
64- 127	--	0.00 %	0.00 %
128- 255	--	0.00 %	0.00 %
256- 511	--	0.00 %	0.00 %
512-1023	--	0.00 %	0.00 %
1024-2047	--	0.00 %	0.00 %
2048-4095	--	0.00 %	0.00 %
4096+	--	0.00 %	0.00 %
Sum		100.00 %	100.00 %

Efficiency The efficiency indicates how well the bus was used during the transfer. Its value is coherent with data and bus fraction, and usually increases with increasing byte count. If it does not, the bursts contain large overheads. In practice, 50 % is a good value.

Data Fraction The data fraction shows the distribution of the data cycles over the different burst lengths. In other words, the table shows the fraction of the total number of data cycles in the test that were used in a burst of a particular length.

Bus Fraction The bus fraction shows the time used for the different bursts in percent of the total number of clocks the selected device pair was busy.

The contents of this table can also be viewed on the *Burst Usage* tab of the Performance Charts window. A description to this view is found in "*Burst Usage*" on page 51.

Termination Statistics (Report Subsection 8.4)

The termination statistics display all information relevant to assess the split behavior of the selected requester-completer pair and lists the data transfers by their byte counts and the devices that terminated them.

Split Statistics (Report Subsection 8.4.1)

The split statistics give an overview about all values needed to evaluate the split behavior of the selected requester-completer pair.

8.4.1 Split Statistics	

No of split sequences.....	2
Average first word latency of split transactions...	84 clocks
Average overall length of split transactions.....	92 clocks
Average time overhead of split transactions.....	91.30 %

In detail, these values are explained in “*Split Transaction Statistics (Report Section 7)*” on page 103.

Termination Burst Histogram, Observed per Transaction (Report Subsection 8.4.2)

The termination burst histogram lists the data transfers by their byte counts and the devices that terminated them. This analysis does not take into account whether a transaction was finished successfully as initiated or whether it was terminated too early.

8.4.2 Termination Burst Histogram, observed per Transaction		
Transferred Bytes	Transfer was stopped by ...	
	Initiator	Target
0- 127	72.73 %	27.27 %
128- 255	0.00 %	0.00 %
256- 383	0.00 %	0.00 %
384- 511	0.00 %	0.00 %
512- 639	0.00 %	0.00 %
640- 767	0.00 %	0.00 %
768- 895	0.00 %	0.00 %
896-1023	0.00 %	0.00 %
1024-1151	0.00 %	0.00 %
1152-1279	0.00 %	0.00 %
... continued ...		
4096+	0.00 %	0.00 %
Sum	72.73 %	27.27 %

The values in this table show how many of all transfers terminated by a device were bursts of a particular length. Therefore, the sum of both columns for requester and completer always is 100 %.

For good performance, preferably long bursts should be used that are then terminated by the requester when they reach the initiated burst length. In the example, none of the transaction exceed a byte count of 127 clocks and most of the transactions were terminated by the requester.

Terminating Devices The devices that can terminate a transaction are the requester and the completer. The requester is assumed to be the terminating device, unless it is a target abort, a disconnect, a retry or a split response.

Another representation of the transfer terminations is found in *“Command Termination, Observed per Transactions (Report Subsection 8.2.4)”* on page 112. There the different termination reasons are displayed for the different PCI-X bus commands that were used in the transactions.

Latency Histogram, Observed per Sequence (Report Subsection 8.5)

The latency histogram in this section lists the fraction of sequence lengths and the fraction of first word latencies.

8.5 Latency Histogram, observed per Sequence		
Clocks	Sequence Length	First Word Latency
0	0.00 %	0.00 %
1	0.00 %	0.00 %
2- 3	0.00 %	0.00 %
4- 7	11.11 %	44.44 %
8- 15	55.56 %	33.33 %
16- 31	11.11 %	0.00 %
32- 63	0.00 %	0.00 %
64- 127	22.22 %	22.22 %
128- 255	0.00 %	0.00 %
256- 511	0.00 %	0.00 %
512-1023	0.00 %	0.00 %
1024-2047	0.00 %	0.00 %
2048-4095	0.00 %	0.00 %
4096+	0.00 %	0.00 %
Sum	100.00 %	100.00 %

The different columns in the table are:

- Sequence Length

This column shows the fraction of sequences with specific lengths. For example, 55.56 % of all sequences are 8 - 15 clocks long.

- First Word Latency

The first word latency is the time from the begin of the address phase until the begin of the first data phase.

Large first word latencies indicate slow address decoding of the completer and/or many initial waits, for example, due to locked completer resources.

Top Ten List of First Word Latencies (Report Subsection 8.5.1)

This report subsection lists the ten data transfers with the largest first word latency that occurred in the traffic of the selected device pair. This table allows you to see the ten commands to which the completer has been reacting extraordinary slowly.

8.5.1 Top 10 list of first word latencies				
Nr	Command	Address	First Word Latency (clocks)	First Word Latency (sec)
10.	MEM_RBLOCK	f0000000	87 clk	0.00000130 s
9.	ALIAS_MEM_RB	f0000000	81 clk	0.00000121 s
8.	CONFIG_READ	0002101c	15 clk	0.00000022 s
7.	IO_READ	0000001c	13 clk	0.00000019 s
6.	MEM_RDWORD	f0000200	13 clk	0.00000019 s
5.	IO_WRITE	0000001c	4 clk	0.00000006 s
4.	MEM_WBLOCK	f0000200	4 clk	0.00000006 s
3.	MEM_WRITE	f0000000	4 clk	0.00000006 s
2.	ALIAS_MEM_WB	f0000000	4 clk	0.00000006 s
1.	--	--	--	--

The table lists the following parameters of the ten data transfers:

- Command

This is the PCI-X bus command that was used by the requester.

- Address

This is the completer address used for the transaction.

- **First Word Latency (clocks)**
This is the total number of clock cycles the device pair occupied the bus from the first address phase until the first successful data transfer, including all failed attempts (retries).
- **First Word Latency (sec)**
This is the total time the device pair occupied the bus from the first address phase until the first successful data transfer, including all failed attempts.

Definitions of Used Measures

This section provides a list of names, terms, and phrases that are used in the field of PCI-X data communication in general and in the Agilent E2929A/B Opt. 200 Performance Optimizer in particular. Many terms are common terms, others may have different interpretations. Therefore, when communicating any test scenarios, measures, or results to other people, it is always recommended to refer to the used definitions to avoid misinterpretations.

The definitions are listed in alphabetical order.

32-Bit Data Transfer (Sum)

The summary of 32-bit data transfer is the data transfer consisting of pure 32-bit data transfer and data transfer caused by refused 64-bit data transfer. See “64-Bit Data Transfer” on page 126.

$$32\text{-Bit Data Transfers (Sum)}[\%] = \frac{\text{datacount} - \text{data64count}}{\text{datacount}} \times 100$$

datacount: All clocks used for data phases.

data64count: All 64-bit transfer cycles.

This value is presented in the performance report. See “Data Phase (Report Subsection 8.2.1)” on page 107 for details.

64-Bit Data Transfer

64-bit data performance analysis is available only within 64-bit systems. In a 64-bit system, the bus statistics section shows the percentage of successful 64-bit transfers and 32-bit transfers, with the latter not making full use of the bandwidth.

$$64\text{-Bit Data Transfers [\%]} = \frac{\text{data64count}}{\text{datacount}} \times 100$$

datacount: All clocks used for data phases.

data64count: All 64-bit transfer cycles.

The 32-bit transfers are distinguished as:

- Pure 32-bit transfers that were initiated by the requester instead of 64-bit transfers.

$$\text{Pure 32-Bit Data Transfers [\%]} = \frac{\text{datacount} - \text{data64count} - \text{data64tried}}{\text{datacount}} \times 100$$

- 64-bit transfers initiated by the requester that were rejected by the completer and then implemented as 32-bit transfers.

$$64\text{-Bit Data Transfer Tried but refused [\%]} = \frac{\text{data64tried}}{\text{datacount}} \times 100$$

datacount: All clocks used for data phases.

data64count: All 64-bit transfer cycles.

data64tried: All transfer that was intended as 64-bit transfer, but refused by the target.

These values are presented in the performance report. See “Data Phase (Report Subsection 8.2.1)” on page 107 for details.

A Average Burst Length

This is the average burst length of the recorded sequences measured in clock cycles. This value include all successful data transfers.

$$\text{Average Burst Length} = \frac{\text{datacount}_{RC}}{\text{nof_sequences}}$$

datacount_{RC}: All data cycles of all transactions of the selected requester-completer pair.

nof_sequences: Number of all sequences covered by the performance measurement.

This value is presented in the performance report. See “Data Phase (Report Subsection 8.2.1)” on page 107 for details.

Average Byte Enable Efficiency

The average byte enable efficiency shows how far the bandwidth of the bus is used to full capacity when transferring data by the selected requester-completer pair.

$$\text{Average Byte Enable Efficiency}[\%] = \frac{\text{bytecount}_{RC}}{\text{divisor}} \times \frac{100}{\text{datacount}_{RC}}$$

datacount_{RC}: All data cycles of all transactions of the selected requester-completer pair.

bytecount: All bytes that occurred in transfer cycles.

divisor: The values for the divisor are:

- 8, if the bus width is 64 bit
- 4, if the bus width is 32 bit

This value is presented in the performance report. See “Data Phase (Report Subsection 8.2.1)” on page 107 for details.

Average Cycles from “Split Response” to “Split Completion”

The average cycles from split response to split completion is defined by the following equation:

$$\text{Average cycles from 'SplitResp' to 'SplitComp'} = \frac{\textit{splitrescom}}{\textit{nof_split_sequences}}$$

splitrescom: Cycles from the split response phase starting at first non-wait cycle of split transaction to first occurring split completion cycle for this sequence.

nof_split_sequences: All transactions that were terminated with the split response command and were fully completed with the split completion command.

Average First Word Latency

The average first word latency is the average number of cycles between the start of the first sequence and the first transferred data of all sequences. This is defined in the following equation:

$$\text{Average First Word Latency} = \frac{\textit{fwordlat}}{\textit{nof_sequences}}$$

fwordlat: Summary of all first word latency cycles of all sequences.

nof_sequences: Number of all sequences.

This value is presented in the performance report. See “*Basic Bus Statistics (Report Section 2)*” on page 97 for details.

Average First Word Latency of Split Transactions

The average first word latency for split transactions is the average number of all first word latency cycles of all split transaction. This is defined in the following equation:

$$\text{Average First Word Latency of Split Transactions} = \frac{\text{split_1st_wd_lat_cycles}}{\text{nof_split_sequences}}$$

split_1st_wd_lat_cycles: Summary of all first word latency cycles of all split transactions.

nof_split_sequences: All transactions that were terminated with 'Split Response' and were successfully completed with 'Split Completions'.

This value is presented in the performance report for:

- the whole test system
See “Basic Bus Statistics (Report Section 2)” on page 97.
- for a selected requester-completer pair
See “Split Statistics (Report Subsection 8.4.1)” on page 119.

Average Overall Length of Split Transactions

The average overall length of split transactions is the average of the lengths of all sequences that are split transactions.

$$\text{Average Overall Length of Split Transaction} = \frac{\textit{split_seq_len_cycles}}{\textit{nof_split_sequences}}$$

split_seq_len_cycles: Summary of the lengths of all sequences that are split transactions.

nof_split_sequences: All transactions that were terminated with 'Split Response' and were successfully completed with 'Split Completions'.

This value is presented in the performance report for:

- the whole test system
See “*Split Transaction Statistics (Report Section 7)*” on page 103.
- for a selected requester-completer pair
See “*Split Statistics (Report Subsection 8.4.1)*” on page 119.

Average Split Rate

The average split rate is the number of all transactions that were terminated with 'Split Response' in all sequences. This is defined in the following equation:

$$\text{Average Split Rate} = \frac{\textit{nof_splittransactions}}{\textit{nof_sequences}}$$

nof_splittransactions: Number of split transactions.

nof_sequences: Number of all sequences.

This value is presented in the performance report. See “*Basic Bus Statistics (Report Section 2)*” on page 97 for details.

Average Time Overhead Split Transactions

The average time overhead of split transactions is the average number of overhead cycles caused by split transactions in all sequences that are split transactions. This is defined in the following equation:

$$\text{Average Time Overhead of Split Transactions} = \frac{\text{split_seq_len_cycles} - \text{split_seq_datacount}}{\text{split_seq_len_cycles}}$$

split_seq_len_cycles: Sum of the length of all sequences that are split transactions.

split_seq_datacount: All datacycles of split transactions.

This value is presented in the performance report for:

- the whole test system
See “*Split Transaction Statistics (Report Section 7)*” on page 103.
- for a selected requester-completer pair
See “*Split Statistics (Report Subsection 8.4.1)*” on page 119.

B Burst Length

The burst length is the number of data phases within a single transaction. Long bursts contain a lot of data phases while only needing one completer address to start at. Thus, transactions with long bursts make more efficient use of the bus than those with short bursts.

This value is presented in the performance report. See the “*Latency Histogram, Observed per Sequence (Report Subsection 8.5)*” on page 121 for details.

Furthermore, this value is displayed in a diagram on the *Latency* tab of the Performance Charts window. For a description of this view, refer to “*Latency*” on page 53.

Bus Fraction

This term is used in tables where the transactions of the recorded data traffic are listed in terms of the burst length, for instance.

$$BusFraction_{Bytecnt} [\%] = \frac{busycount_{Bytecnt}}{busycount_{RC}}$$

- busycount*_{Bytecnt}: Number of clock cycles, that were used for transactions with a particular burst length.
- busycount*_{RC}: Total number of clocks during the requester-completer pair occupied the bus. (Address phase, wait phase, data phase termination phase and recover phase.)

This value is presented in the performance report. See “*Efficiency over Byte Count, observed per Sequence (Report Subsection 8.3.1)*” on page 118 for details.

Furthermore, this value is displayed in a diagram on the *Burst Usage* tab of the Performance Charts window. For a description of this view, refer to “*Burst Usage*” on page 51.

See also “*Data Fraction*” on page 135.

Bus Time

The bus time is the ratio of any cycle on the bus (gap_cycles included) and the bus speed.

$$\text{Bus Time} = \frac{\text{cyclecount}}{\text{bus_speed}}$$

Bus Users Overview

The bus users overview is an index that evaluates the performance of an analyzed PCI-X device. It is derived from utilization and efficiency. The higher this value is, the poorer is the device's PCI-X design and the more it adversely affects the whole system performance.

$$\text{Bus Users Overview} = \frac{\text{Utilization}_{RC}}{\text{Efficiency}_{RC}}$$

This value is presented in the performance report. See the “*Bus Users Overview (Report Section 6)*” on page 102 for details.

Byte Enable Efficiency

The byte enable efficiency is a measure that indicates how well the maximum bandwidth of the PCI-X bus was employed by the communicating devices. It is given as the percentage of the maximum number of bytes that can be transferred within the data phases.

In a 32-bit system, the maximum number of bytes that can be transferred in one data phase is four, in a 64-bit system eight. The byte enable efficiency is calculated as shown below.

$$\text{Byte Enable Efficiency} [\%] = \frac{\text{bytecount}}{\text{divisor}} \times \frac{100}{\text{datacount}}$$

bytecount: All bytes occurred in transfer cycles.

datacount: All clocks used for data phases.

divisor: The values for the divisor are:

- **8**, if the bus width is 64 bit
- **4**, if the bus width is 32 bit

The byte enable efficiency of the complete data traffic on the PCI-X bus is presented in the performance report. See the “*Efficiency Statistics (Report Section 4)*” on page 99 for details.

The byte enable efficiency of the selected requester-completer pair is found in “*Data Phase (Report Subsection 8.2.1)*” on page 107.

See also “*Average Byte Enable Efficiency*” on page 127, “*PCI-X Efficiency*” on page 140, and “*PCI-X Time Efficiency*” on page 142.

D Data Fraction

This term is used in tables where the transactions or sequences of the recorded data traffic are listed in terms of the transferred bytes, for instance. In this context the data fraction gives the number of data phase cycles, that were used within bursts of a particular length, out of the total number of data phases in the traffic of this requester-completer pair (RC).

$$DataFraction_{Bytecnt} [\%] = \frac{datacount_{Bytecnt}}{datacount_{RC}} \times 100$$

$datacount_{Bytecnt}$: Number of data phase cycles, that were used within bursts of a particular lengths.

$datacount_{RC}$: Total number of data phases in the traffic of this requester-completer pair (RC).

This value is presented in the performance report. See “*Efficiency over Byte Count, observed per Sequence (Report Subsection 8.3.1)*” on page 118 for details.

Furthermore, this value is displayed in a diagram on the *Burst Usage* tab of the Performance Charts window. For a description of this view, refer to “*Burst Usage*” on page 51.

See also “*Bus Fraction*” on page 132.

Data Phase Utilization

The data phase utilization gives the percentage of all clock cycles that were used for data phases. Thus, it is the fraction of bus time that was actually used for data transfer.

$$\text{Data Phase Utilization [\%]} = \frac{\text{datacount}}{\text{cyclecount}} \times 100$$

datacount: All clocks used for data phases.

cyclecount: The total number of cycles on the bus (gap-cycles included).

This value is used in the performance report. For the data utilization of the complete bus traffic, refer to the “*Bus Utilization Statistics (Report Section 5)*” on page 101. The data utilization of the selected requester-completer pair is found in “*Bus Utilization (Report Subsection 8.2)*” on page 106.

The latter is also displayed in the *Bus Utilization* diagram on the *PCI-X Usage* tab of the Performance Charts window. For a description of this diagram, refer to “*PCI-X Usage*” on page 50.

Disconnect at Next ADB

Disconnect at next allowable disconnect boundary (ADB) means that the target reads/writes up to the next ADB. The ADB refers to the next memory boundary on bus and is 128 Bytes (the seven least significant bits are zero).

E Efficiency over Byte Count

The efficiency in reference to the occurred burst lengths is defined by the following equation:

$$Efficiency_{Bytecnt} [\%] = \frac{bytecount_{Bytecnt}}{divisor} \times \frac{100}{busycount_{Bytecnt}}$$

*bytecount*_{Bytecnt}: Number of transferred bytes.
*busycount*_{Bytecnt}: Number of clock cycles, that were used for transactions with a particular burst length.
 (Address phase, wait phase, data phase, termination phase and recover phase.)

This value is presented in the performance report. See “*Efficiency over Byte Count, observed per Sequence (Report Subsection 8.3.1)*” on page 118.

The subscript *Bytecnt* in the formula above indicates the affiliation to the byte count range in the left column shown in this report subsection.

F First Word Latency

The first word latency is the time between the begin of the address phase and the transfer of the first data word. It is measured in clock cycles.

This value is presented in the performance report. See “*Latency Histogram, Observed per Sequence (Report Subsection 8.5)*” on page 121 for details.

Furthermore, this value is displayed in a diagram on the *Latency* tab of the Performance Charts window. For a description of this view, refer to “*Latency*” on page 53.

I Invalid Measures

Invalid measures are caused by incomplete sequences or invalid cycles.

- Incomplete Sequence

A sequence is incomplete if at least one transaction is missing.

- Invalid Cycles

Invalid cycles are all cycles from incomplete sequences and any cycle of transactions that have errors or are incomplete.

These values are presented in the performance report. See “*Invalid Measures (Report Subsection 1.3)*” on page 96 for details.

0 Overhead

Overhead covers all address, wait, recover and termination cycles of all sequences of a requester-completer pair that are no parts of split transactions.

The overhead can be caused by initiator and completer.

- Overhead caused by Initiator are:

- addressphases (address cycles and attribute cycles included)
- termination cycles, if the transaction was not terminated with Single Data Phase Disconnect

- Overhead caused by Target are:

- wait cycles (decode cycles, target response cycles and wait cycles included),
- recover cycles
- termination cycles, if the transaction was terminated with Single Date Phase Disconnect

The overhead is presented in the performance report. See “*Overhead (Report Subsection 8.2.2)*” on page 109 for details.

Overhead Utilization

The overhead utilization is the time the bus was occupied for transferring overhead data, that is, all clock cycles except data phases.

$$\text{Overhead Utilization [\%]} = \frac{\text{addrcount} + \text{waitcount} + \text{termcount} + \text{recovercount}}{\text{cyclecount}} \times 100$$

<i>addrcount:</i>	All address cycles (dual address cycles, single address cycles) and attribute cycles.
<i>waitcount:</i>	All wait, decoding and target response cycles.
<i>recovercount:</i>	All recover cycles. The recover cycle is only used in DWORD transactions, where the initiator needs one cycle to respond to #TRDY of the target. For more information, refer to “ <i>Recover Cycle</i> ” on page 144.
<i>termcount:</i>	All termination cycles, caused by retry, target abort or split response.
<i>cyclecount:</i>	The total number of cycles on the bus (gap-cycles included).

The overhead utilization of the complete bus traffic is presented in the performance report. See the “*Bus Utilization Statistics (Report Section 5)*” on page 101 for details.

P PCI-X Efficiency

The PCI-X efficiency is a measure that indicates how well the bus was used by the communicating devices. It is one of the most important values when classifying a system’s performance.

The efficiency is calculated as the number of *actually* transferred bytes divided by the number of bytes that *theoretically* could be transferred within the used clock cycles.

$$PCI\text{-}X\ Efficiency\ [\%] = \frac{PCI\text{-}X\ Throughput}{PCI\text{-}X\ Utilization} = PCI\text{-}X\ Time\ Efficiency \times Byte\ Enable\ Efficiency\ [\%]$$

$$PCI\text{-}X\ Efficiency\ [\%] = \frac{bytecount}{divisor} \times \frac{100}{addrcount + waitcount + datacount + recovercount + termcount}$$

bytecount: All bytes that occurred in transfer cycles.

divisor: The values for the divisor are:

- 8, if the bus width is 64 bit
- 4, if the bus width is 32 bit

addrcount: All address cycles (dual address cycles, single address cycles) and attribute cycles.

waitcount: All wait, decoding and target response cycles.

datacount: All clocks used for data phases.

recovercount: All recover cycles. The recover cycle is only used in DWORD transactions, where the initiator needs one cycle to respond to #TRDY of the target. For more information, refer to “Recover Cycle” on page 144.

termcount: All termination cycles, caused by retry, target abort or split response.

The different performance measures can be derived from each other. Therefore, an alternative way to calculate the efficiency is to divide the PCI-X throughput by the PCI-X utilization or to multiply the time efficiency and the byte enable efficiency.

An efficiency near 100 % means that much data has been transferred while the bus was occupied to a minimum.

The efficiency is displayed as follows:

- On the *PCI-X Usage* tab of the Performance Charts window, the efficiency is displayed for the data traffic of the selected device pair. For a description of this view, refer to “*PCI-X Usage*” on page 50. The respective section of the performance report is described in “*Efficiency Statistics (Report Subsection 8.3)*” on page 117.
- On the *Burst Usage* tab of the Performance Charts window, the efficiency is displayed for transactions with different burst length. For a description, refer to “*Burst Usage*” on page 51.

The respective section of the performance report is described in “*Efficiency over Byte Count, observed per Sequence (Report Subsection 8.3.1)*” on page 118.

Also refer to “*Byte Enable Efficiency*” on page 134 and “*PCI-X Time Efficiency*” on page 142.

PCI-X Throughput

The throughput is the amount of data transferred per time.

$$PCI\text{-}X\text{ Throughput [MB/s]} = \frac{\textit{Transferred Data}}{\textit{Time}} \text{ [MB/s]} = \frac{\textit{bytecount}}{\textit{bustime}} \times \frac{1}{2^{20}}$$

This value is presented in the performance report. See “*Bus Throughput Statistics (Report Section 3)*” on page 98 for details.

If this value is given as a percentage, 100 percent refers to actual possible maximum value of the current system under test. This maximum depends on the detected bus width and bus speed. Therefore, the maximum value can vary between 132 MByte/s in a 33 MHz/32-bit system, 528 MByte/s in a 66 MHz/64-bit system and up to 1014 MByte/s in a 133 MHz/64-bit system.

PCI-X Time Efficiency

The time efficiency indicates the relation between the bus time used for transferring overhead and the bus time used for the actual data transfer.

$$PCI-X \text{ Time Efficiency } [\%] = \frac{\text{Efficiency}}{\text{Byte Enable Efficiency}} = \frac{\text{Time used for Overhead Phases}}{\text{Time used for Overhead and Data Phases}} [\%]$$

$$PCI-X \text{ Time Efficiency } [\%] = \frac{\text{datacount}}{\text{addrcount} + \text{cyclecount} + \text{datacount} + \text{termcount} + \text{recovercount}} \times 100$$

- datacount:* All clocks used for data phases.
- addrcount:* All address cycles (dual address cycles, single address cycles) and attribute cycles.
- cyclecount:* The total number of cycles on the bus (gap-cycles included).
- recovercount:* All recover cycles. The recover cycle is only used in DWORD transactions, where the initiator needs one cycle to respond to #TRDY of the target. For more information, refer to “*Recover Cycle*” on page 144.
- termcount:* All termination cycles, caused by retry, target abort or split response.

This value only takes into account the occurrence of data phases. It is not considered whether the full bandwidth of the PCI-X bus is used to transfer data (defined by the byte enable signal #BE). Thus, the time efficiency can also be determined by dividing the total efficiency by the byte enable efficiency.

The time efficiency is presented in the performance report. See “*Efficiency Statistics (Report Section 4)*” on page 99 for details.

Also refer to “*Byte Enable Efficiency*” on page 134 and “*PCI-X Efficiency*” on page 140.

PCI-X Utilization

The utilization is a measure for the relation between used (busy) and unused (idle) bus time in the data traffic.

$$PCI-X \text{ Utilization } [\%] = \frac{\text{Busy Clocks}}{\text{Total Clocks}} \quad [\%] = \frac{\text{addrcount} + \text{waitcount} + \text{datacount} + \text{recovercount} + \text{termcount}}{\text{cyclecount}} \times 100$$

bytecount: All bytes that occurred in transfer cycles.

divisor: The values for the divisor are:

- **8**, if the bus width is 64 bit
- **4**, if the bus width is 32 bit

addrcount: All address cycles (dual address cycles, single address cycles) and attribute cycles.

waitcount: All wait, decoding and target response cycles.

datacount: All clocks used for data phases.

recovercount: All recover cycles. The recover cycle is only used in DWORD transactions, where the initiator needs one cycle to respond to #TRDY of the target. For more information, refer to “*Recover Cycle*” on page 144.

termcount: All termination cycles, caused by retry, target abort or split response.

cyclecount: The total number of cycles on the bus (gap-cycles included).

The PCI-X bus is considered as being busy if any of the signals FRAME#, IRDY#, TRDY#, DEVSEL#, or STOP# is low.

The different utilization values are presented in the performance report. Refer to “*Bus Utilization Statistics (Report Section 5)*” on page 101 and its subsection for more information.

R Recover Cycle

The recover cycle is only used in DWORD transactions. During this cycle, there is no action on the bus, but the bus is not idle as #FRAME and #IRDY are still active. This cycle is necessary, as the initiator needs one cycle to respond to #TRDY of the target. #TRDY becomes inactive after transferring the data and indicates the end of the transaction.

NOTE In burst operations, no recover cycles are necessary, because the initiator relies on the target to inactivate #TRDY on the last data phase.

Requester-Completer Pair Efficiency

The requester-completer pair efficiency is a measure that indicates how well the bus was used by the communicating device pair.

$$Efficiency_{RC}[\%] = \frac{bytecount_{RC}}{divisor} \times \frac{100}{addrcount + waitcount + datacount + recovercount + termcount}$$

Requester-Completer Pair Throughput

The throughput of the requester-completer pair is the amount of data transferred per time between this pair.

$$Throughput_{RC}[\text{MB/s}] = \frac{Transferred\ Data_{RC}}{Time} [\text{MB/s}] = \frac{bytecount_{RC}}{bustime} \times \frac{1}{2^{20}}$$

This value is presented in the performance report. See “*Bus Throughput Statistics (Report Section 3)*” on page 98 for details.

S Single Data Phase Disconnect

If the requester performs a burst read, the target can send a single data phase and disconnect the transaction by signaling *Single Data Phase Disconnect*. This signals the requester to perform the next burst read. This possibility of terminating a transaction is necessary for targets that do not support burst.

Split Overhead

The split overhead covers all address, wait, recover and termination cycles of a requester-completer pair that are part of split transactions.

This value is presented in the performance report. See “*Bus Utilization (Report Subsection 8.2)*” on page 106 for details.

Split Response

The target of the completer signals that it will complete the transaction as a split transaction. The initiator of the completer later sends split completions. Terminating the split completion with *Disconnect at Next ADB* allows the initiator of the completer to send the data in more than one split completion.

Split Sequences

A split sequence is a sequence that consists of split transactions.

Index

#

- 32-Bit Data Transfer (Sum)
 - Definition 125
- 64-bit Bus Statistics (report subsection 4.2)
 - Description 100
- 64-bit Data
 - Command Usage Table 111
- 64-bit Data Transfer
 - Definition 126

A

- Average
 - Burst Length 127
- Average Burst Length
 - Definition 127
 - Report subsection 8.2.1 107
- Average Byte Enable Efficiency
 - Definition 127
 - Report subsection 8.2.1 107
- Average Cycles from “Split Response” to “Split Completion”
 - Definition 128
- Average First Word Latency
 - Definition 128
- Average First Word Latency of Split Transactions
 - Definition 129
- Average Overall Length of Split Transactions
 - Definition 130
- Average Split Rate
 - Definition 130
- Average Time Overhead Split Transactions
 - Definition 131

B

- Basic Bus Statistics (report section 2)
 - Analysis 57
 - Description 97
- Burst
 - Analysis 66
- Burst Behavior 65
- Burst Length
 - Average Length 107
 - Definition 131
- Burst Usage (tab) 51
 - Example 26

- Bus Fraction
 - Definition 132
 - report subsection 8.3.1 118
- Bus Statistics (report sections 1 to 7)
 - Description 92
- Bus Throughput Statistics (report section 3)
 - Analysis 57
 - Description 98
- Bus Users Overview (report section 6)
 - Analysis 61
 - Definition 133
 - Description 102
- Bus Utilization (report subsection 8.2)
 - Analysis 63
 - Description 106
- Bus Utilization Statistics (report section 5)
 - Analysis 58
 - Description 101
- Byte 134
 - Enable 65
- Byte count
 - Efficiency (report) 118
 - over DWord Command (report subsection 8.2.5)
 - Analysis 66
 - over DWord Command (report subsection 8.2.6) 67
 - Analysis 67
- Byte count over Command
 - observed per Sequence (report subsection 8.2.5)
 - Analysis 66
 - observed per Transaction (report subsection 8.2.6)
 - Analysis 67
- Byte count over Command (report subsection 8.2.5)
 - Analysis 66
- Byte Enable
 - Average Efficiency 107

C

- Command
 - Tab 52
- Command Termination
 - Completer Disconnect 86
 - Completer Retry 85
 - Disconnect at Next ADB 86
 - Split Response 85

Types 84

- Command Termination (report subsection 8.2.4)
 - Analysis 69
 - Description 112
- Command Usage (report section 8.2.3)
 - Description 111
- Completer
 - Device Identification 38
 - Disconnect 86
 - Retry 85
- Completer Disconnect
 - Command Termination 86
- Completer Identification 39
 - Example 20
 - Setup 41
 - Tab 41
- Completer Retry
 - Command Termination 85

D

- Data Capture
 - Example 24
 - see also Capture (tab)
 - Setup 44
- Data Fraction
 - Definition 135
 - report subsection 8.3.1 118
- Data Phase (report subsection 8.2.1)
 - Analysis 65
 - Description 107
- Data Phase Utilization
 - Definition 136
- Data Transfer
 - 32-Bit 125
 - 64-bit 126
- Definition
 - 32-Bit Data Transfer 125
 - 64-bit Data Transfer 126
 - Average Burst Length 127
 - Average Byte Enable Efficiency 127
 - Average Cycles from “Split Response” to “Split Completion” 128
 - Average First Word Latency 128
 - Average First Word Latency of Split Transactions 129
 - Average Overall Length of Split Transactions 130
 - Average Split Rate 130
 - Average Time Overhead Split

- Transactions 131
 - Burst Length 131
 - Bus Fraction 132
 - Bus Users Overview 133
 - Data Fraction 135
 - Data Phase Utilization 136
 - Disconnect at Next ADB 136
 - Efficiency over Byte Count 137
 - First Word Latency 137
 - Invalid Measures 138
 - Overhead 138
 - Overhead Utilization 139
 - PCI-X Byte Enable Efficiency 134
 - PCI-X Efficiency 140
 - PCI-X Throughput 141
 - PCI-X Time Efficiency 142
 - PCI-X Utilization
 - Requester-Completer Pair Throughput 144
 - Single Data Phase Disconnect 144
 - Split Overhead 145
 - Split Response 145
 - Split Sequences 145
- Determining Requester ID
- Example 22
- Disconnect at Next ADB
- Definition 136
- Documentation Overview 7
- E**
-
- Efficiency 28
- over Burst Length (report section 8.3.1)
 - Description 118
 - see also PCI-X Time Efficiency
 - Statistics (report section 4)
 - Analysis 59
 - Description 99
 - Statistics (report subsection 8.4)
 - Description 117
- Efficiency over Byte Count
- Bus Fraction 118
 - Data Fraction 118
 - Definition 137
- Example
- Loading Setup Files 19
- Example for
- Burst Usage (tab) 26
 - Completer Identification 20
 - Data Capture 24
 - Determining Requester ID 22
 - Loading Test Results 26
 - PCI-X Usage (tab) 25
 - Performance Setup (window)
 - Report (tab) 20
 - Requester Identification 21
 - Requester-Completer Pair Specification 23
- Existing Data, Reusing 76
- Existing Results, Reusing 76
- Exporting
- Report 75
 - Trace Memory 75
- F**
-
- First Word Latency
- Definition 137
 - Minimization 83
 - Report subsection 8.5.1 122
- G**
-
- General Information
- on the System under Test 56
- General Information (report section 1)
- Analysis 56
 - Description 93
- H**
-
- Hardware Setup 38
- I**
-
- Identification
- see Requester, Completer and Address Range Identification
- Interpreting
- Reports 54
 - Results 49
- Invalid Measures
- Definition 138
- Invalid Measures (report subsection 1.3)
- Description 96
- L**
-
- Latency
- Histogram 121
 - Tab 53
- Latency Histogram (report subsection 8.5)
- Description 121
- Loading
- Example Test Results 26
 - Report Files 77
 - Setup Files 77
 - Setup Files (example) 19
 - Waveform Files 77
- N**
-
- Navigation
- in the Report 54
 - Main Window 13, 14
- O**
-
- Overall System Performance
- Hardware Setup 38
- Overhead
- Definition 138
- Overhead (report subsection 8.2.2)
- Analysis 64
 - Description 109
- Overhead Utilization
- Definition 139
- Overview
- Documentation 7
- P**
-
- Pair Select (tab) 43
- PCI-X Byte Enable Efficiency
- Definition 134
- PCI-X Design
- Process 10
 - Rules 15
- PCI-X Efficiency
- Analysis 59
 - Definition 140
- PCI-X Performance
- Running a Measurement 34
- PCI-X Performance Optimizer
- General Description 9
 - Running Measurements 48
 - Setting Up a Test 37
 - Steps of Performance Analysis 12
 - User Interface 13
- PCI-X Throughput
- Analysis 57
 - Definition 141
- PCI-X Time Efficiency
- Definition 142
- PCI-X Usage (tab) 50
- Example 25
- PCI-X Utilization
- Analysis 58
 - Definition 143
- Performance Button Group 14
- Performance Charts (window)
- Burst Usage (tab) 26, 51
 - Charts and Report Sections 54
 - Command (tab) 52
 - Data Source 49
 - Latency (tab) 53
 - PCI-X Usage (tab) 25, 50
- Performance Measures
- Predefined 28
- Performance of Particular Devices (Hardware Setup) 38
- Performance Setup (window)
- Completer Identification (tab) 41
 - Pair Select (tab) 43
 - Requester Identification (tab) 42
- Performance Windows 14
- Post-Processed Analysis
- Definition 11
 - Traffic Samples 11
- Printing Test Results 74

protocol checker 30

R

Real-Time Measurements

Definition 11

Report

Contents 54

Export 75

Interpreting 54

Loading Report Files 77

Navigation 54

Results Presented as Charts 54

Using the Output for Performance

Evaluation 55

Using the Output for Performance

Optimization 62

Report Tab

Example 20

Requester Identification 39

Example 21

Setup 42

Tab 42

Requester-Completer

Efficiency (report subsection 4.1)

Description 99

Pair 43

Pair Measurements (report section 8)

Description 104

Requester-Completer Pair

Example for Specification 23

Throughput 144

Requester-Completer Pair Throughput

Definition 144

Rescan the Trace Memory 48

Result Charts

Interpreting 49

Results, Reusing 76

Retry Rate 28

Reusing

Previously Saved Data 76

Test Setups and Results 73

RTP Measurement Setup (window) 32

Rules for good Performance

I/O Commands 15

Memory and I/O Commands 15

Memory Commands 15

Split Transactions 15

Rules for Proper PCI-X Design 15

S

Selecting

Predefined Performance Measures 32

Settings

Testcard 29

Setup

Completer Identification 41

Data Capture 44

Example Files 19

Files 77

Requester Identification 42

Reusing Test Setups 73

Software 40

Test Hardware 38

Single Data Phase Disconnect

Definition 144

Single Phase Data (SDP) Disconnect

Command Termination 86

Software Setup 40

Split Behavior

Analysis 60

Split Overhead

Definition 145

Split Rate 28

Split Response

Command Termination 85

Definition 145

Split Sequences

Definition 145

Split Statistics (report subsection 8.4.1)

Description 119

Split Transaction Statistics (report section 7)

Analysis 60

Description 103

Split Transactions

Rules for good Performance 15

Statistics (report) 103

Start

Test 34

Statistical Base (report subsection 1.2)

Description 94

Statistical Basis (report subsection 8.1)

Description 105

Stop after ... Samples (option) 45

Stop after Access to Address ... with (option) 45

Stop at End of Trace Memory (option) 45

System under Test, General Information 56

T

Terminating Devices 121

Termination Burst Histogram (report section 8.4.2)

Description 120

Termination Statistics (report subsection 8.4)

Analysis 70

Description 119

Test

Run 34

Start 34

Test Results

Loading the Example Results 26

Printing 74

Reuse 73

Viewing 25

Testcard

Cross Triggering 31

External Triggering 31

Settings 29

testcard

protocol checker 30

Throughput 28

Time Efficiency

see PCI-X Time Efficiency

Top Ten List of First Word Latencies (report subsection 8.5.1)

Description 122

Trace Memory 11

Export 75

Rescan 48

Traffic

Overhead 64

Samples 11

trigger out line 31

Triggering

Cross 31

External 31

U

Update Interval (performance measure) 33

Utilization 28

see PCI-X Utilization

V

Viewing

Test Results 25

W

Waveform Files 77

Publication Number: 5988-4907EN

