



Agilent E2926A/B Opt. 200 PCI Performance Optimizer

## **User's Guide**



**Agilent Technologies**



## Important Notice

This document contains propriety information that is protected by copyright. All rights are reserved. Neither the documentation nor software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Agilent Technologies.

© Copyright 1998, 2000 by:  
Agilent Technologies  
Herrenberger Straße 130  
D-71034 Böblingen  
Germany

The information in this manual is subject to change without notice. Agilent Technologies makes no warranty of any kind with regard to this manual, 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 direct, indirect, special, incidental, or consequential damages in connection with the furnishing, performance, or use of this manual.

Brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Authors: Stephan Greisinger and Stefan Kunzi, t3 medien GmbH

# Contents

PCI Performance Optimizer Overview	7
The Process of PCI Design	8
The Approach of the Performance Optimizer	9
Measuring the System Performance in Real Time	10
Using Post-Processed Analysis	10
Selecting the Appropriate Way of Analysis	12
The PCI Performance Optimizer's User Interface	13
Rules for Proper PCI Design	15
Running a Sample PCI Performance Optimizer Session	17
Determining the Overall System Performance	17
Analyzing the Performance of the PCI Host Bridge	20
Loading the Setup File	20
Setting up the Measurements	21
Specifying the Master and Target Identification	22
Specifying the Data Capture	24
Viewing the Test Results	25
Setting Up a PCI Performance Optimizer Test	27
Setting Up the Test Hardware	28
Master Identification	29
Target Identification	30
Preparing the Software for the Test	30
Enabling the PCI Performance Optimizer	32
Setting Up the Report	33
Setting up the Target Identification	34
Setting up the Master Identification	35
Selecting a Master/Target Pair	36
Setting Up the Data Capture	37

Measuring System or Device Performance	41
Running the Performance Measurements	42
Interpreting the Result Charts	43
PCI Usage	44
Burst Usage	45
Command	46
Latency	47
Full Transaction Clocks	48
Interpreting the Report Output	48
Using the Report Output for Performance Evaluation	50
General Information on the System under Test	50
Analyzing the PCI Throughput	51
Analyzing the PCI Utilization	52
Analyzing the PCI Efficiency	53
The Bus Users Overview	54
Summary for the Performance Evaluation Example	55
Using the Report Output for Performance Optimization	55
Analyzing the Bus Utilization	56
Analyzing the Traffic Overhead	56
Analyzing the Wait States	57
Analyzing the Byte Enable and Burst Behavior	58
Analyzing the Bursts	59
Analyzing Command Termination	61
Summary of the Performance Optimization Example	63
Re-Using the Test Setups and Results	65
Printing and Exporting Test Results	66
Printing the Test Results	66
Exporting the Report	67
Exporting the Trace Memory	67
Exporting the Bus Activity Listing	68
Exporting the Test Settings	68
Re-Using Previously Saved Data	69

Going Further Into Details	71
Verifying Good Performance	71
Performance in the Presented Charts	72
Identifying Design Rule Violations	73
Replacing Cards or Improving Behavior	74
Improving Target Behavior	75
Minimize Latencies	75
Command Termination	77
Using the Bus Activity Lister	79
Report Reference	81
Example Transfers	82
Bus Statistics (Report Sections 1 to 7)	83
General Information (Report Section 1)	84
Basic Bus Statistics (Report Section 2)	86
Bus Throughput Statistics (Report Section 3)	87
Efficiency Statistics (Report Section 4)	88
Bus Utilization Statistics (Report Section 5)	90
Bus Users Overview (Report Section 6)	91
Interrupt Latency (Report Section 7)	92
Master-Target Pair Measurements (Report Section 8)	93
The Header of Report Section 8	94
Statistical Basis (Report Subsection 8.1)	94
Bus Usage (Report Subsection 8.2)	95
Bus Utilization (Report Subsection 8.3)	96
Data Phase (Report Subsection 8.3.1)	97
Time Overhead (Report Subsection 8.3.2)	99
Command Usage (Report Subsection 8.3.3)	101
Command Termination (Report Subsection 8.3.4)	102
Wait Histogram (Report Subsection 8.3.5)	104
Burst Length over Command (Report Subsection 8.3.6)	105
Efficiency Statistics (Report Subsection 8.4)	106
Efficiency over Burst Length (Report Subsection 8.4.1)	107
Termination Statistics (Report Subsection 8.5)	109
Termination Burst Histogram (Report Subsection 8.5.1)	110
Latency Histogram (Report Subsection 8.6)	111
Top Ten List of First Word Latencies with Retries (Report	

Subsection 8.6.1)	113
Definitions of Used Measures	115

---



# PCI Performance Optimizer Overview

The Agilent E2926A/B Opt. 200 PCI Performance Optimizer is an additional option available for the Agilent E2920 verification tools series. Being fully integrated into the framework, it is employed together with a PCI Exerciser and Analyzer testcard and its graphical user interface (GUI).

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

With these features, you can easily reveal which devices cause poor 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 bad performance.

With this 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 Design

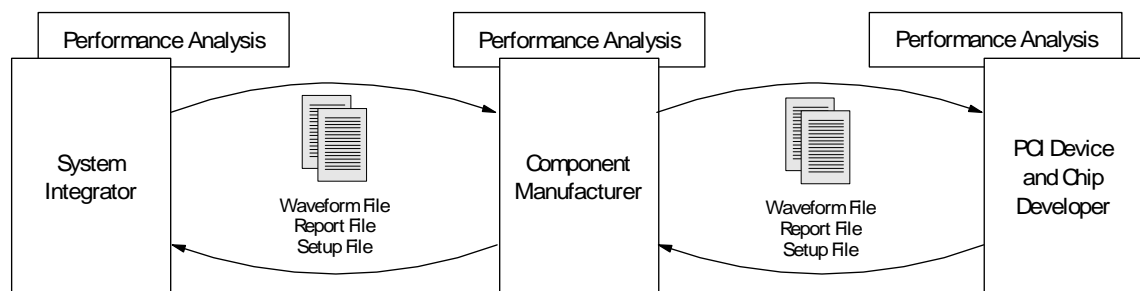
When the PCI technology was introduced in the PC environment, it was a lot faster than previously used buses (ISA, EISA). Initially, there was no need to optimize the performance of the PCI bus. This has changed by the time the requirement for fast data traffic has increased. Now the PCI bus often is the performance bottleneck, especially in high-performance servers that deal with lots of disk and network traffic. Due to the complex protocol there are many ways to improve PCI performance both on a component level as well as on a system level.

A huge number of companies and people develop chipsets, devices and systems using the PCI technology for data exchange. Even when obeying all the rules included in the PCI specification, the performance of your device still depends on many factors.

Possible scenarios within the process of PCI design are:

- 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 systems.



# The Approach of the Performance Optimizer

The Agilent E2926A/B Performance Optimizer provides all features needed to evaluate and optimize any PCI device or system. The test results presented by this software can be used to compare different devices or different combinations of devices. With this tool you gain all information required to improve the performance, either by selecting the best components or by optimizing the communication abilities of particular PCI devices.

## The Methods for Performance Analysis

Basically, there are two different approaches to analyze a system under test. It depends on the particular case which of the options is to be preferred. With the Agilent E2926A/B Performance Optimizer you can employ both methods for performance analysis:

- Real-Time Measurements (part of the PCI Analyzer)

Real-time performance analysis based on programmable counters allows long-termed 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 insight to track down the root cause of performance issues.

- Post-Processed Analysis (with the PCI Performance Optimizer only)

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, burst efficiency, or wait histograms. This applies for the whole bus as well as for specific devices or a master/target pair. This information helps you to quickly determine why your performance problems may exist by allowing you to identify poorly performing PCI devices, drivers, or mismatched master/target pairs.

The trace memory of the Agilent E2926A/B testcard can store up to 64 k 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. In order to base the calculations on a larger amount of traffic data for more reliable results, the Agilent E2995A 155 x 4 M Trace Memory Board can be employed, which can store up to 4 M traffic samples.

## Measuring the System Performance in Real Time

When using the real-time measurement options with the Performance Optimizer, your system under test can be observed over a long time. However, only a few different measures can be taken simultaneously. The software provides some pre-defined measures, but the internal performance counters can be freely programmed to match your personal needs.

### The Display Options

The results of the real-time measurements can be monitored online. They are displayed on the screen while the measurements are running. The user can switch between two display modes:

- loadmeter display

The specified measures are displayed in a bar diagram.

- line diagram

The last 100 values are displayed in a line diagram over time.

The performance counters accumulate the specified bus events during an arbitrary measurement interval. The user can choose to either display the accumulated values or to reset the counters after each interval.

The real-time measurement options do not require the Performance Optimizer to be licensed for your software package. However, for completeness they are included in the Performance menu, too, because they provide important features for performance evaluation.

For a complete description of all features of the real-time measurements, please refer to *Analyzing PCI Performance* in the “Agilent E2926A/B PCI Analyzer User’s Guide”.

## Using Post-Processed Analysis

With the use of the post-processed analysis many different measures can be considered simultaneously. This allows a very detailed analysis of the data traffic on the bus. 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 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, like 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	<p>The trace memory can store up to 64 k samples. For a larger trace memory, the Agilent E2995A 155 x 4 M Trace Memory Board is available. With this option you can record up to 4 M samples.</p>
The Steps of Performance Optimization	<p>The Agilent E2926A/B PCI Performance Optimizer performs the data analysis by taking the following steps:</p> <ul style="list-style-type: none"> <li>• Recording the bus traffic <p>The measurement is either started by the user or can be set up to be triggered on a certain event occurring on the bus. Then the trace memory records the data until the trace memory is full or another specified bus event occurred on the bus.</p> <p>Usually, the observation time is only a split of a second. Thus, the memory content is a “snapshot” of the bus activities.</p> </li> <li>• Analyzing the recorded data <p>For the analysis, the trace memory content is loaded from the testcard into the PCI Performance Optimizer. The software then derives a large number of result values from the sampled bus activities.</p> </li> <li>• Presenting the test results <p>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. Several results are also presented in diagrams, histograms and tables.</p> <p>These test results give a deep insight into your system’s and devices’ traffic behavior. You can identify the devices with poor performance and get detailed hints for improvement.</p> </li> <li>• Providing result files (setup file, report file, and waveform file) <p>Optionally, you can store your test settings and test results in files. With these files you can re-use the settings or the data from previous settings. They are also a convenient basis for communication between several user groups. 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.</p> </li> </ul>

## Selecting the Appropriate Way of Analysis

The two available test methods—post-processed analysis and real-time measurements—are used for different tasks.

The following table gives an overview of the advantages and drawbacks of both methods:

	Real-Time Measurement	Post-Processed Analysis
<b>Advantages</b>	long observation time good base for statistical considerations trend control	many measures simultaneously very detailed analysis re-usable results
<b>Drawbacks</b>	only few measurements at the same time	short observation time (snapshot) risk to measure non-representative data

Both methods complement one another when used simultaneously. Starting the capture for post-processed analysis in an unfavorable moment during the benchmark may result in capturing data useless for the analysis. Monitoring the bus traffic with the real-time measurements while taking the snapshot for the post-processed analysis ensures that the snapshot contains representative data.

For example, when running a video game as benchmark to stress the PCI interface of a graphics card, this game could stop transferring video data just the moment you start the capture for post-processed analysis (perhaps because it is interrupted by something). Then you will probably observe that the PCI bus is idle through most of the observation time. To avoid this risk, use the real-time measuring of PCI utilization or throughput to immediately show when the bus gets idle.

**Combining Both Methods** To utilize the advantages of both methods, proceed as follows:

- Run a benchmark and observe the PCI performance with the real-time measurements.
- Set up a post-processed analysis and reproduce the situation in which the performance dropped.
- Take the snapshot.
- Analyze the recorded data using the post-processed analysis.

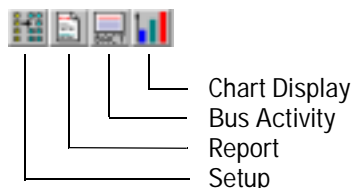
# The PCI Performance Optimizer's User Interface

The User Interface of the Agilent E2926A/B Opt. 200 Performance Optimizer is fully integrated into the framework of the Agilent E2926A/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 Exerciser and Analyzer software.

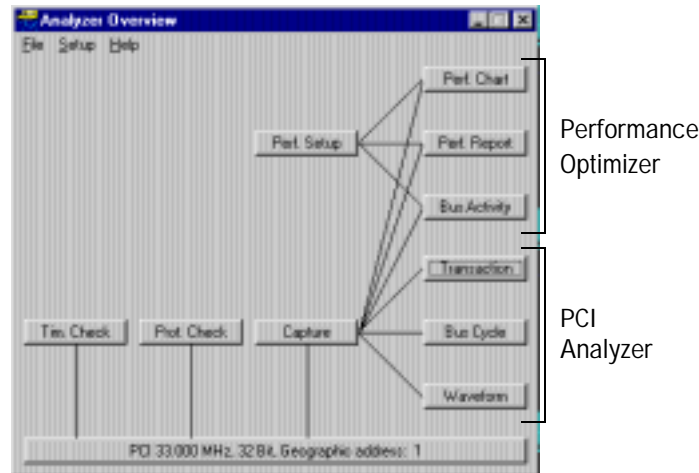


**Navigation** The most important features of each part of the framework are available via the buttons in the respective button group. All these features and more are available via the menus of the main window, as well. As a third way, there are the overview windows provided by the user interface. These windows offer process-oriented access to the individual features.

**Performance Button Group** The direct way to open the windows providing the features of the PCI Performance Optimizer is to click on the respective buttons in the *Performance* button group.



**Overview Window** Both the PCI Analyzer and the PCI Performance Optimizer employ an internal trace memory, which is used to record the data traffic on the PCI bus. Therefore, they are both presented in one overview window. One way to open this window is to select *Show Analyzer Overview* from the Performance menu.



This Analyzer Overview window shows the individual components of the PCI Analyzer and the PCI Performance Optimizer and how they interact. Clicking the buttons brings up the respective windows.

**Performance Windows** Within the PCI Performance Optimizer, there are the following windows provided for post-processed performance analysis and optimization:

- Performance Setup

With the Performance Setup window you can set up the contents of the report to be generated, identify master and target devices to be considered in the test, and control the data capture for the performance analysis.

- Performance Charts

The performance charts show graphical representations of the performance-relevant aspects found in the captured PCI traffic.

- Performance Report

The textual performance report lists all results in a hierarchical way, so you can focus on different levels of detail.

- Bus Activity Lister

The bus activity lister summarizes all transactions found in the captured data. However, for each bus activity, only properties are displayed that are of interest in a performance analysis.

**Real-Time Performance  
Measurements**

Additionally, the PCI Performance Optimizer employs features for real-time performance measurements. These features are assumed to be known from the PCI Analyzer. For a short introduction, refer to *“Determining the Overall System Performance” on page 17*.

For detailed information refer to *Analyzing PCI Performance* in the *“Agilent E2926A/B PCI Analyzer User’s Guide”*.

## Rules for Proper PCI Design

When designing PCI devices a lot of design rules need to be obeyed to avoid your design conflicting with other devices. Furthermore, there are rules that need to be followed to improve the communication abilities of the devices. Making the devices behave according to these rules ensures that the device will not only work with good performance in a test environment, but on all platforms and under most circumstances.

Even when respecting all of these rules, 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.

Within this text, only a few rules can be named that are essential for good performance:

- Implement extended PCI commands

The following commands are optimized for certain types of data transfer and should be used instead of plain Memory Read/Memory Write (MR and MW) commands whenever possible:

- Memory Read Line – MRL
- Memory Read Multiple – MRM
- Memory Write and Invalidate – MWI

- Use long bursts

On most platforms read bursts should have a minimum length of 64 dwords to gain 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 bus performance might not be directly affected, the usage of I/O commands reduces the performance of the whole system by using CPU time.

- Minimize latency

Latency is referred to as the time between two events on the bus. Minimizing the latency means reducing the time between two events and, thus, speeding up the communication considerably.

PCI devices can insert wait cycles into transfers for different reasons. For example, a device can have no access to its resources because they are occupied or locked by another device.

Such wait cycles increase the latency and should, therefore, be avoided where possible.

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 71*.





# Running a Sample PCI Performance Optimizer Session

The Agilent E2926A/B Opt. 200 Performance Optimizer provides all features needed to gain full insight into the performance of any PCI system and its communicating devices. There are several ways to evaluate the characteristic measures of the system under test. Basically, you can distinguish between real-time measurements—while the system is running— and a detailed post-processed data traffic analysis. An introduction to these approaches, explained on an easy example, is found in the following.

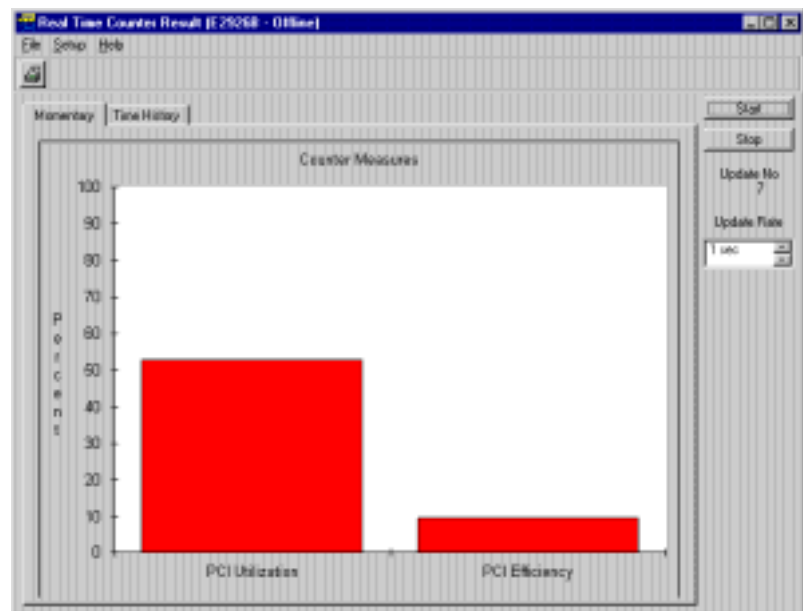
## Determining the Overall System Performance

Using the real-time performance measures of the Agilent E2926A/B PCI testcard provides a very quick overview on the overall system performance. While the system under test is running, you can watch the values of a set of measures in two different views. For the measures to observe you can select four pre-defined variables like PCI efficiency or utilization. Alternatively, you can also define your own variables.

To run real-time measurements on the system under test, take the following steps:

- 1 Select *Real Time Counter Result* from the *Performance* menu of the application's main window.

The Real Time Counter Result window opens.

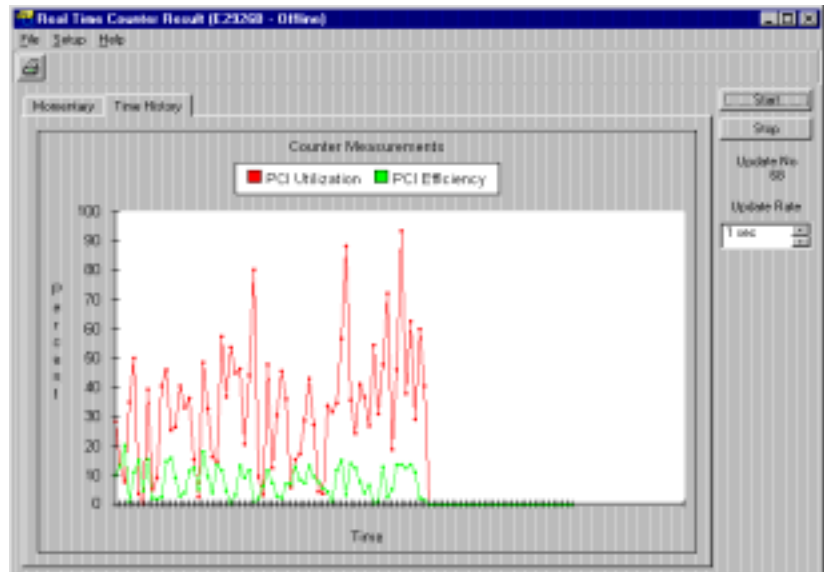


- 2 Press the *Start* button to the right.

By default, the two measures PCI Utilization and PCI Efficiency are displayed in percent. These values are re-calculated and displayed every second.

- 3 To change the update rate, use the arrow buttons next to the text field holding the current update rate.

- 4 To observe the two measures in a chart diagram as a function over time, switch to the *Time History* tab.



**NOTE** In offline/demo mode, running the real-time measurements does not cause an error message but produces random values to be displayed.

**The Available Measures** With the default settings, the Real Time Counter Result window displays two measures at a time. However, the application can employ up to six internal counters plus a reference counter that counts the clock cycles. These counters can be used to calculate any value of interest, whereas four can be displayed at a time.

For convenience, there are four pre-defined measures, that are commonly used when speaking of system or device performance:

- PCI Utilization
- PCI Efficiency
- PCI Throughput
- Retry Rate

How to set up the performance counters for your personal needs, is described in detail in the "*Agilent E2926A/B PCI Analyzer User's Guide*".

# Analyzing the Performance of the PCI Host Bridge

The Agilent E2926A/B Opt. 200 Performance Optimizer can be used to evaluate the performance of a particular device in the system under test. For this example, the PCI host bridge is observed while a master device is accessing the system memory through the host bridge.

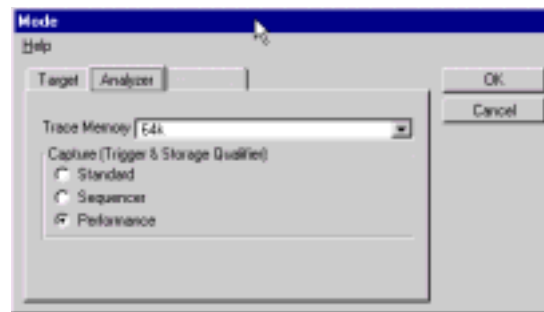
For post-processed analysis, the data traffic on the PCI bus is recorded in the trace memory. The analysis on the stored data is done after the test has finished.

## Loading the Setup File

When preparing the Agilent E2926A/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 *perf.bst* in the path *samples/demo/*.
- 3 Click the *Load* button.

In case your application was not yet in performance mode, it will now switch to performance mode automatically. This causes the buttons in the *Performance* group and the items in the *Performance* menu to be enabled.



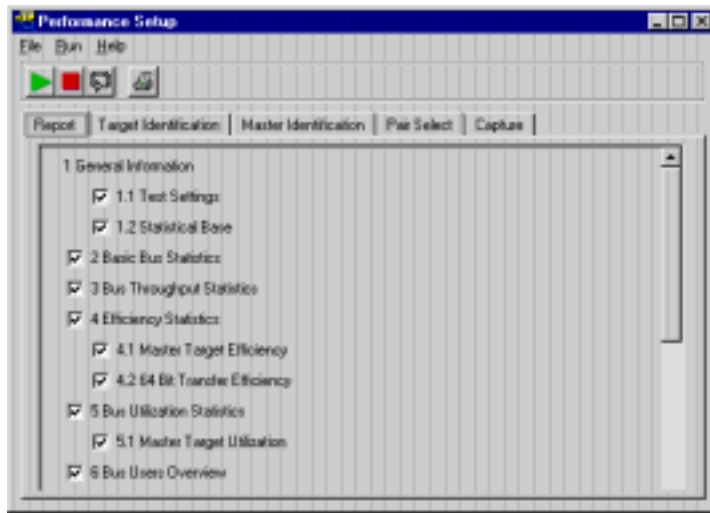
**NOTE** The setup file assumes an Agilent E2926A testcard to be used for the test—at least in offline/demo mode. If this is not the case, you will be informed in a message box. Change the configuration to the testcard required for the example.

## Setting up the Measurements

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 step through the setup and do changes where applicable.

- 1 Select the *Performance Setup* item from the *Performance* menu.

The Performance Setup window opens, displaying the *Report* tab.



After the test has finished, the software provides a textual report of the results in the Performance Report window. With the items found on this tab of the Performance Setup window you can specify the particular results and statistics that are included in the report.

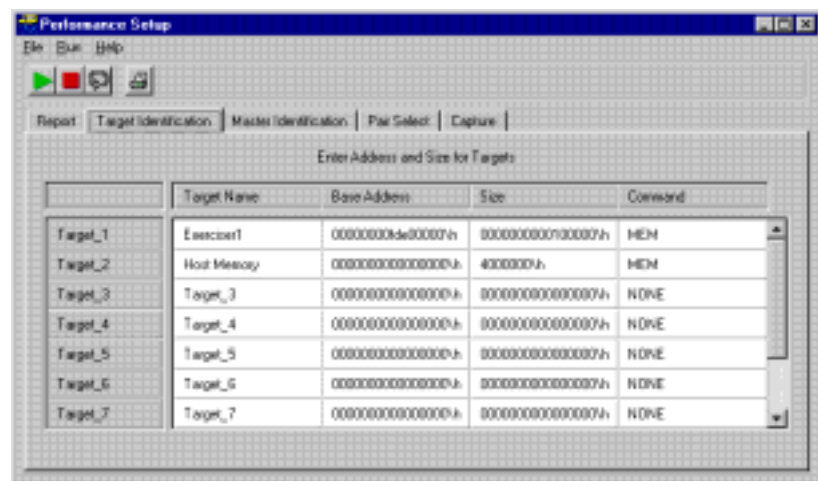
- 2 In our example, we leave all parts of the report included.

## Specifying the Master and Target Identification

In order to enable the Agilent E2926A/B PCI Performance Optimizer to identify particular devices communicating on the PCI bus, you need to specify them in the setup. Masters are identified by their GNT# lines, targets by their address range.

In the example, no particular masters are specified, because the complete traffic via the host bridge is to be observed.

- 1 If the Performance Setup window is not yet open, select *Performance Setup* from the *Performance* menu in the application's main window.
- 2 Switch to the *Target Identification* tab to see the view below.



In this table you can specify up to ten different target devices including their address ranges and the types of PCI commands that are considered in the measurements.

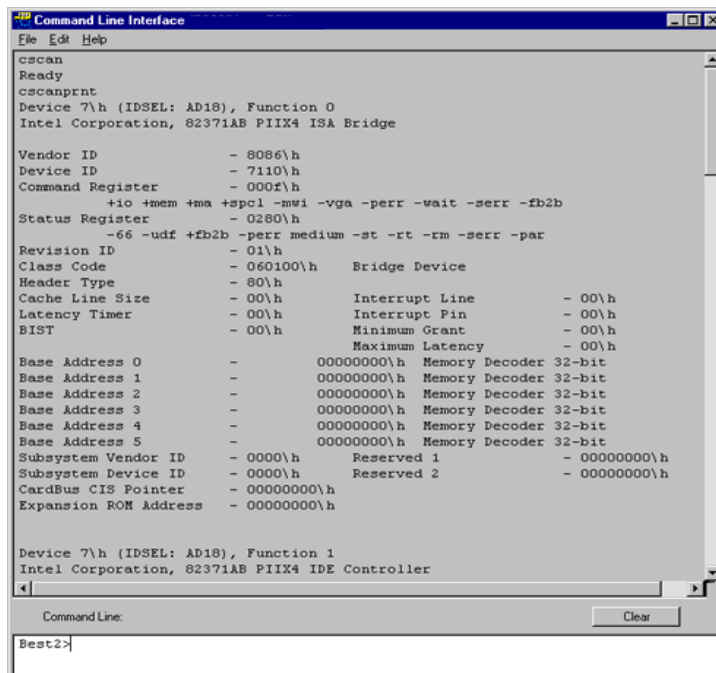
For the test example, two targets are explicitly specified in the table. On one hand there is the built-in target of a Agilent Exerciser card and on the other the host memory.

### Determining the Target Address Range

In case you need to know the address range of a particular target for the test setup, use the Command Line Interface:

- 1 Select *Command Line* from the *Window* menu in the application's main window to open the Command Line Interface window.
- 2 In the command line at the bottom of the window enter **cscan** and then **cscanprnt**, each followed by the return key.

These commands do a configuration scan on the PCI bus. The output area of the window now presents information on all devices that are found on the bus. This also includes the address ranges.



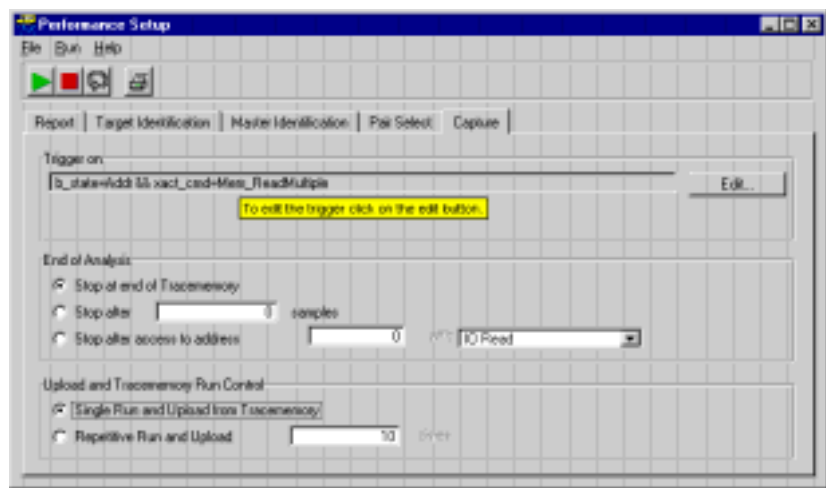
**NOTE** The Command Line Interface only works with a proper data connection to an Agilent Exerciser and Analyzer testcard plugged into the system under test.

The desired address ranges can now be entered in the *Target Identification* tab of the *Performance Setup* window.

## Specifying the Data Capture

The setup of the performance test can not only be focussed on the traffic of particular devices, you can also control the start and the duration of the test. This implies both for the start and the end of the data recording, that you either can wait for a certain amount of time or for a particular event to happen on the bus.

- 1 If the Performance Setup window is not yet open, select *Performance Setup* from the *Performance* menu in the application's main window.
- 2 Switch to the *Capture* tab.



This tab is used to set up a trigger pattern for the start and additional conditions for the behavior of the software regarding the data capture.

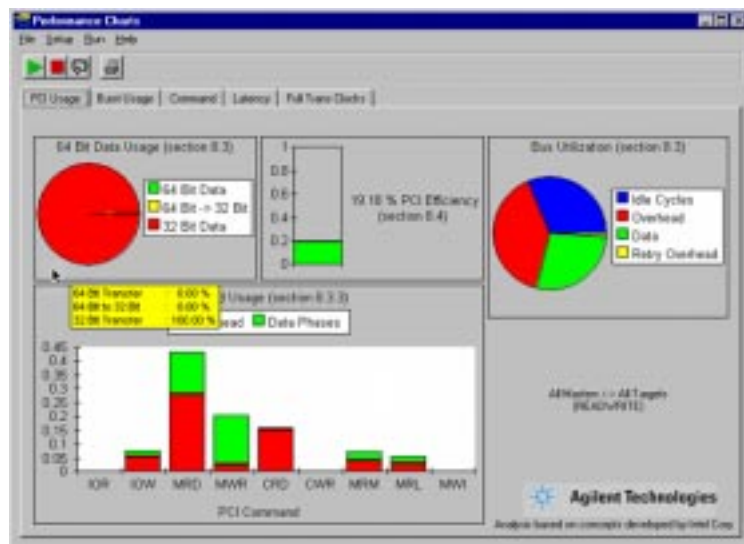
In the test example, the data recording starts at the first occurrence of the PCI command **Memory Read Multiple**. The test stops as soon as the trace memory is completely filled with data.



## 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 graphical charts. Other ways are the textual report or the contents of the bus activity lister.

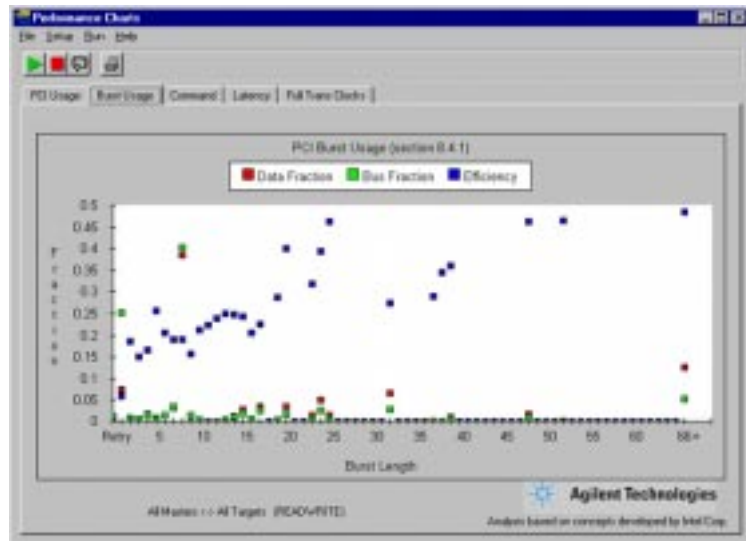
- 1 In the application's main window, select *Performance Charts* from the *Performance* menu, to open the Performance Charts window.



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

- 2 Move the mouse over either pie chart to view the numerical values of the various pie slices.

### 3 Switch to the *Burst Usage* tab.



# Setting Up a PCI Performance Optimizer Test

All steps that are required to set up a PCI performance test with the Agilent E2926A/B Opt. 200 Performance Optimizer are introduced here. It is explained how a proper setup is done, but not the use of the different features as such. See “*Measuring System or Device Performance*” on page 41 for information on the features.

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 E2926A/B testcard. If you need to look up, how to install the testcard in the system under test and how to connect it to the control PC, refer to the “*Agilent E2926A/B PCI Analyzer User’s Guide*”.

The additional steps required for the Performance Optimizer are described in “*Setting Up the Test Hardware*” on page 28.

- The **software setup** includes starting the Performance Optimizer—once per test session—and the individual setup for every single test. Information on these steps is found in “*Preparing the Software for the Test*” on page 30.

# Setting Up the Test Hardware

To evaluate a system's performance with the Agilent E2926A/B Opt. 200 Performance Optimizer, the Agilent E2926A/B testcard must be plugged into the system under test. Refer to *Possible PCI Analyzer Configurations* in the *Analyzer User's Guide* for the different possible configurations.

**Overall System Performance** If you wish to test a system's PCI performance but not the performance of a particular PCI 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 30.

**Performance of Particular Devices** On the other hand, to evaluate the traffic of particular PCI devices, you need to allow the Performance Optimizer to identify these devices.

**Master devices** initiate all read/write traffic on the PCI bus. They request the bus from the arbiter, which grants the use of the bus to them when the bus is free. Master and arbiter use REQ# (request) and GNT# (grant) lines for communication. When the master's request is granted, it first puts an address on the bus. The address is then read and decoded by all **target devices**. The target that has the address space covering the address on the bus replies to the master, and the data transfer may start.

Thus, there are different ways of identifying the two different types of PCI devices:

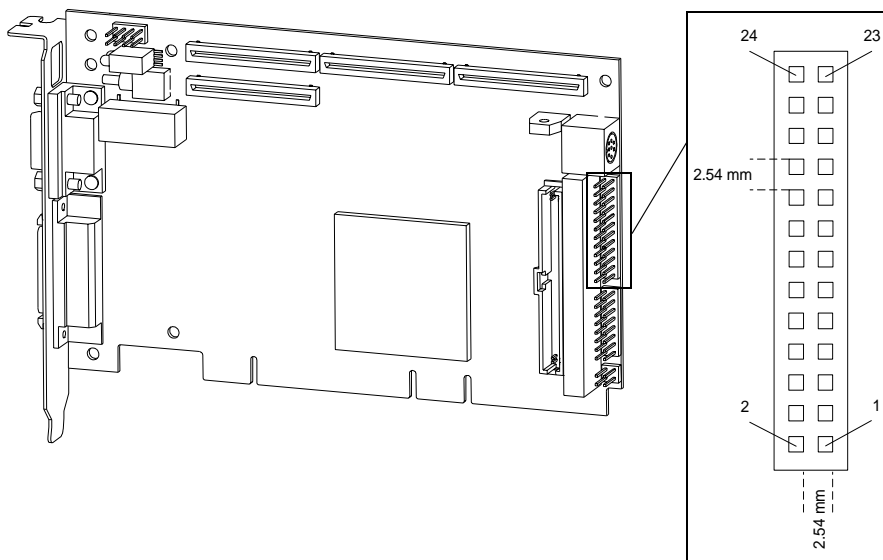
- Master devices are identified by their GNT# lines. How to specify the master devices and their GNT# lines in the Performance Optimizer is described in *"Master Identification"* on page 29.
- Target devices are identified by their address spaces. How to specify the target devices and their address spaces is described in *"Target Identification"* on page 30.

## Master Identification

If you want to measure and evaluate the PCI traffic initiated by particular master devices, you need to set up the Performance Optimizer to identify these masters. For this purpose, the masters' GNT# lines must be made accessible to the Agilent E2926A/B Exerciser and Analyzer card. These lines must be connected to the external trigger input pins of the testcard—if necessary, by soldering a wire to it.

**NOTE** If you also want to run latency tests, the REQ# lines of the respective masters need to be connected to the testcard as well. It is advisable to do this now together with the GNT# lines.

The following figure shows the trigger lines of the Exerciser and Analyzer card. It indicates where to find the pins for the external trigger signals. “*Setting up the Target Identification*” on page 34 describes how to specify the pin assignment in the software.



## Target Identification


To identify the traffic to a particular target device no wiring is necessary. Every target in a PCI system gets a unique address range assigned to it during system startup. A target may have several decoders and may, therefore, use several different address spaces at once. The PCI Performance Optimizer considers the different address spaces of such PCI devices as independent devices.

The easiest way to find out the address ranges that the system under test assigns to the various target devices, is to use the Command Line Interface of the Agilent E2926A/B Exerciser and Analyzer. Open the Command Line Interface from the *Window* menu in the main window. To display a list of all installed targets, first enter the command *cscan* and then *cscanprnt*.

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 Performance Optimizer. See “*Preparing the Software for the Test*” on page 30 for more information.

## Preparing the Software for the Test

This section covers all information needed to set up the Agilent E2926A/B Opt. 200 PCI Performance Optimizer. It is assumed that the required hardware setup has already been done. If not, refer to “*Setting Up the Test Hardware*” on page 28.

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:

- **Enabling the Performance Optimizer.**

Before you can program any test setup and run any test, you need to enable the PCI Performance Optimizer. How to do this is described in *“Enabling the PCI Performance Optimizer” on page 32.*

- **Report tab.**

The software will create a textual report on the evaluation results of the system under test. To set up this report for your individual needs, see *“Setting Up the Report” on page 33.*

- **Master Identification tab and Target Identification tab.**

In case you want to run performance measurements on particular PCI devices, you need to specify the respective master and target devices. This step is explained in *“Setting up the Target Identification” on page 34.*

- **Pair Select tab.**

To restrict the performance test on a single device or a single master/target pair, see *“Selecting a Master/Target Pair” on page 36* 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 37* for details.

Once you have enabled the PCI Performance Optimizer, you can do your setup in any order you like.

## Enabling the PCI Performance Optimizer

The PCI Performance Optimizer is a software package that can be regarded as an add-on part of the PCI Analyzer. For this reason, it is also included in the Analyzer Overview window.

Performance measurements can only yield correct values if all bus cycles are evaluated and no filters set. However, when starting the Agilent E2920A software GUI, the analyzer mode *standard* is active by default. This standard mode allows to program a storage filter to omit unwanted bus cycles and is, therefore, of no use for performance tests. Thus, the analyzer mode needs to be changed.

To enable the PCI Performance Optimizer, proceed as follows:

- 1 Select *Enable* from the *Performance* menu in the main window.  
A message box appears telling you, that you need to change the capture mode of the PCI Analyzer to *Performance*.
- 2 Click *OK* to open the Mode window.



- 3 In the *Capture (Trigger & Storage Qualifier)* group on the *Analyzer* tab, select *Performance*.
- 4 Click *OK* to close the Mode window again.

Now the items in the *Performance* menu are enabled as well as the buttons in the *Performance* group of the main window and in the Analyzer Overview window.




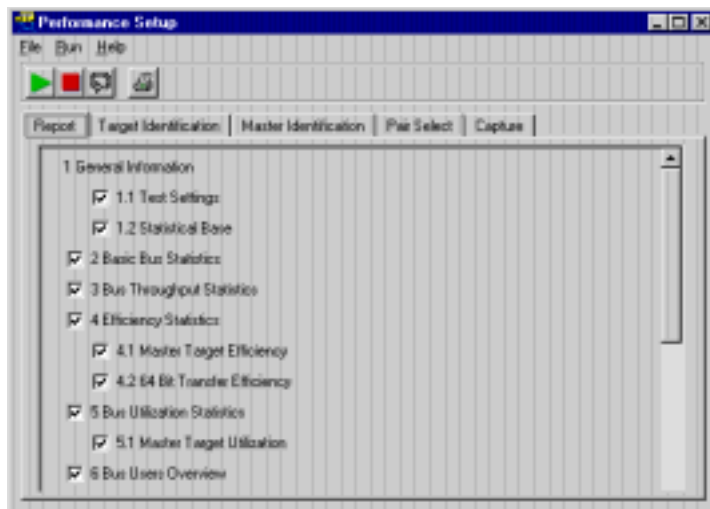


## Setting Up the Report

The Agilent E2926A/B Opt. 200 PCI Performance Optimizer creates a textual report with every test. This report contains general information, overall system performance, and detailed traffic statistics. It also contains hyperlinks for navigation between related topics in the report.

By default all available information will be included in the report. However, you can customize the contents of the report for your individual needs:

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



- 3 Check all topics that you want to have included in the report, uncheck those, that you want to omit.

With this selection, only those sections will appear in your report that are relevant for your test.


## Setting up the Target Identification

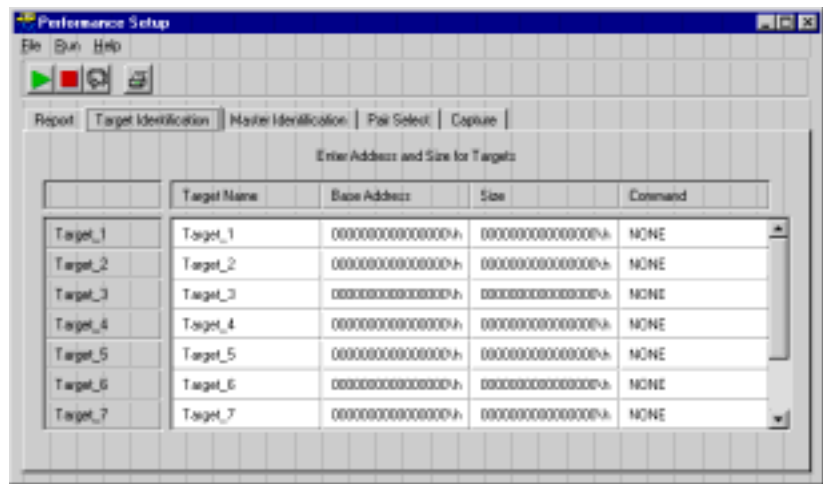
The PCI Performance Optimizer allows to evaluate the performance of single PCI devices or communicating pairs of devices. The target identification is used to specify the target devices of interest. Every target is identified by the address range that it decodes. You can monitor up to ten different targets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

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

**NOTE** If you need to know the address range of a target device, open the Command Line Interface (CLI) from the *Window* menu in the main window and enter the commands *cscan* and then *cscanprnt*, each followed by the return key. The CLI then lists all devices found on the PCI bus including their address spaces.

To define the target identification proceed as follows:

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




- 3 If you wish to have a particular name for a target in the report and in the graphical displays, click into a *Target Name* field and enter a name. Alternatively, you can use the default name.
- 4 Click into the *Base Address* field of the same row and enter the base address of this target's memory. The ending *h* marks hex values.

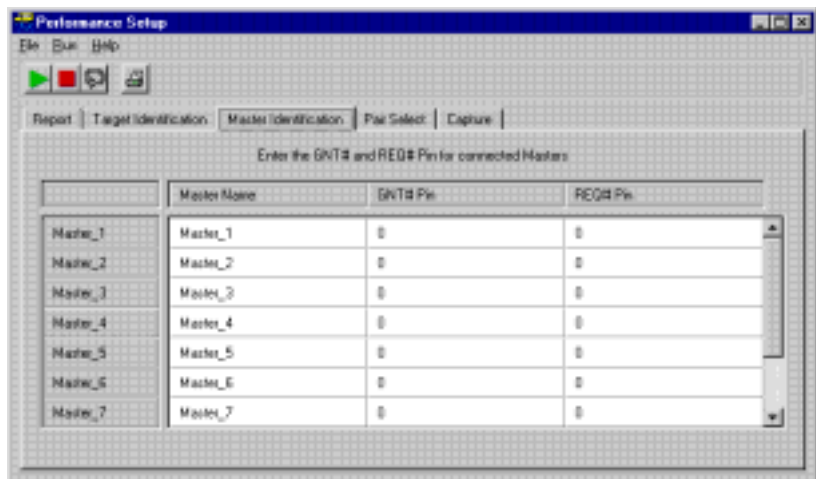
- 5 Click into the *Size* field of this row and enter the size of the target's memory. The ending *h* marks hex values.
- 6 Click into the Command field of this row and select one of the commands from the list. With this selection you can restrict your measurements to traffic using either I/O commands or memory commands, or ignore it at all.
- 7 Repeat the steps for all targets you want to be considered in the performance measurements.

## Setting up the Master Identification

The master identification is used for performance measurements on one or more particular master devices. Every master is identified by its GNT# pin. You can monitor up to ten different masters at a time.

To define the master identification proceed as follows:

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



- 3 If you wish to have a particular name for a master in the report and in the graphical displays, click into a *Master Name* field and enter a name. Alternatively, you can use the default name.
- 4 Click into the *GNT# Pin* field of this row and enter the trigger port number that this master's GNT# line is connected to. Zero (0) means not connected.

- 5 Click into the *REQ# Pin* field of this row and enter the trigger port number that this master's REQ# line is connected to. Zero (0) means not connected.


If the REQ# pin of a master is not connected to any trigger port of the Agilent E2926A/B Exerciser and Analyzer card, arbitration considerations and latency measurements cannot be done for this master.

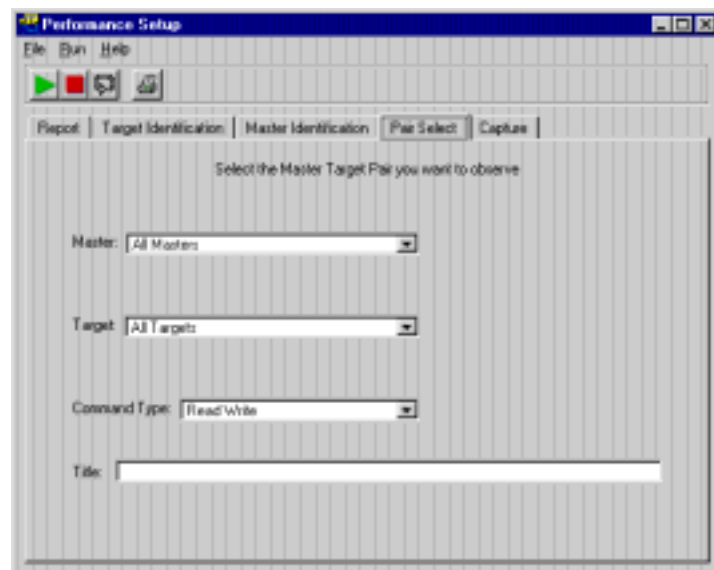
- 6 Repeat the steps for all masters you want to be considered in the performance measurements.

## Selecting a Master/Target Pair

Besides the more general performance measurements for the specified PCI 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, proceed as follows:

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



- 3 From the *Master* list and the *Target* 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 master's view. Thus, *Read* represents the traffic from target to master, and *Write* from master to target.


- 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.

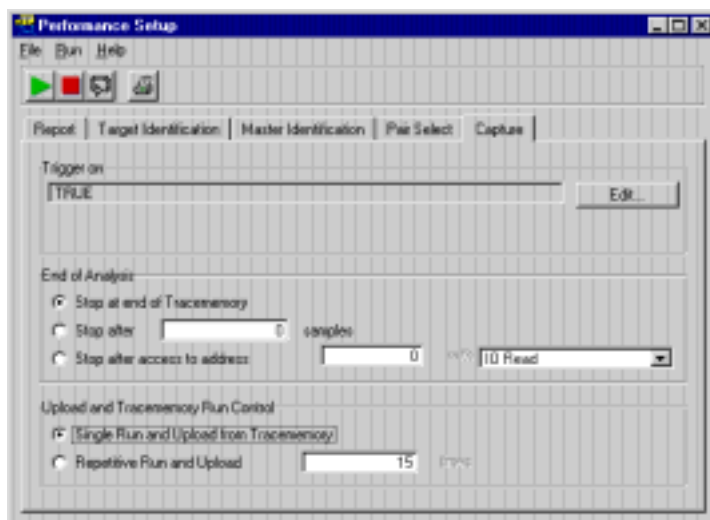
## Setting Up the Data Capture

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 device, it is required to record the data traffic for post-processed analysis only when certain events 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, proceed as follows:

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



- 3 Specify 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 should be known from the Agilent PCI 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 E2926A/B PCI 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. The size of the trace memory depends on the settings in the *Analyzer* tab of the Mode window, which is a feature of the Agilent E2926A/B PCI Analyzer. Performance measurements based on a larger amount of samples can be gained with the use of a Agilent E2995A 155 x 4M Trace Memory board.

- *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 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.

5 The bottom group in the *Capture* tab of the Performance Setup window features the *Upload and Tracememory Run Control*. You can select one of two options:

- *Single Run and Upload from Tracememory* (default).

With this option the performance test runs only once.

- *Repetitive Run and Upload*.

Select this option if you need a large amount of samples to gain more reliable measurement results. The statistical calculations are done on the accumulated values. Enter the number of test repetitions in the text field.

If the measurements should run too long, you can always manually terminate the tests. The results presented then are calculated 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 E2926A/B Opt. 200 PCI Performance Optimizer can be used to measure the performance of a complete PCI system as well as to analyze particular PCI devices. The presented results and statistics allow to debug the test devices and to optimize their performance in terms of data throughput, efficiency, and bus utilization.

After you have set up the test correctly (as described in *“Setting Up a PCI Performance Optimizer Test”* on page 27), 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 42.
- After the test is finished, the Performance Charts window presents the results in a graphical manner on several window tabs. These charts are explained in *“Interpreting the Result Charts”* on page 43.
- An introduction to the contents of the performance report can be found in *“Interpreting the Report Output”* on page 48.
- 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 50.
- Another example containing statistics on the data traffic of a particular device is found in *“Using the Report Output for Performance Optimization”* on page 55.

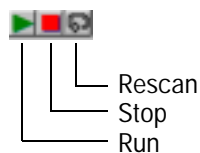
# Running the Performance Measurements

With the Performance Optimizer there are two ways how to evaluate the performance of the devices under test:

**Real-Time Measurements** You can define up to four variables—or use pre-defined ones—that represent measurement values as bus utilization, efficiency, etc. With these variables a real-time measurement can be run, where you watch these values while the system is running. These test options can be found in the Performance menu of the application, although they are also possible with the Agilent E2926A/B PCI Analyzer only. For a description of real-time measurements, refer to the “Agilent E2926A/B PCI Analyzer User’s Guide”.

**Post-Processed Analysis** The other option is to record detailed information on the bus traffic in the trace memory and evaluate them after the test is finished. This approach is called post-processed analysis and is explained here.

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



**Start the Test** In order to start the test, click on the Run button in either 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 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 finished, the contents of the trace memory will be evaluated and the results will be presented in the Performance Charts window, the Performance Report window and the Bus Activity Lister window.

**Rescan the Trace Memory** The Rescan button is thought to be used when doing different performance calculations on the same data. For example, if you want to compare the performance of different master-target pairs during the same test period. In a case like this you do not want to run a new test but to evaluate the data in the trace memory again with different settings. After running the test for the first master-target 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 to 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 Masters/All Targets* needs to be selected. For more information see “*Selecting a Master/Target Pair*” on page 36.

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

- “*PCI Usage*” on page 44.
- “*Burst Usage*” on page 45.
- “*Command*” on page 46.
- “*Latency*” on page 47.
- “*Full Transaction Clocks*” on page 48.

**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 Usage

This tab displays general statistics on the usage of the PCI bus. It contains up to four charts—depending on the test setup—that provide information on the recorded bus traffic: 64-bit data usage, PCI efficiency, bus utilization, and command usage. The selection of the master-target pair observed in the test is displayed in the lower right corner of the window.

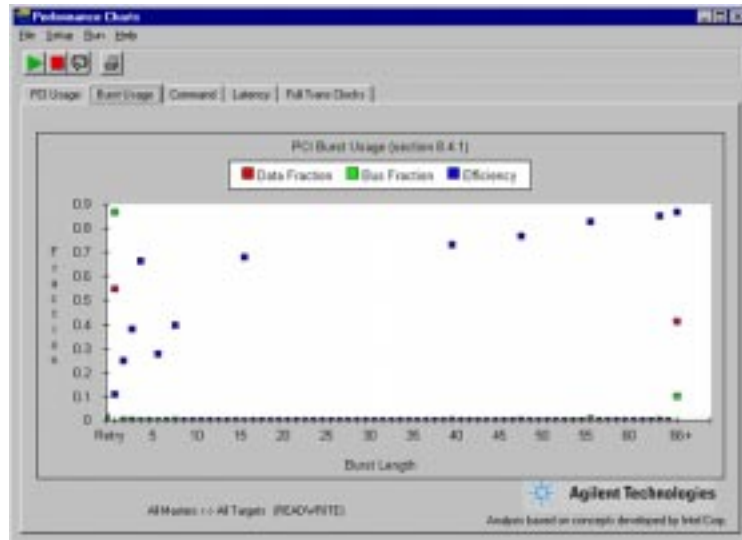


For a detailed description of the charts, see the topic named *PCI Usage Tab* in the online help. The contents of the different charts are explained in the respective sections of the performance report.

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 8.4.1 of the performance report. It shows the distribution of the burst lengths that contributed to the bus traffic. Burst lengths are displayed up to 65, where the length is the number of address phases plus the number of data phases. All longer bursts appear together as 66+.



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

## Command

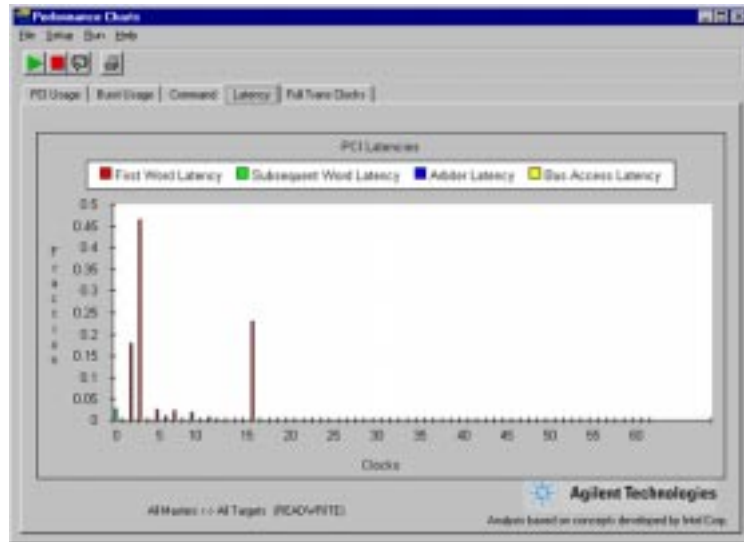
The diagram in this tab corresponds to the section 8.3.6 of the performance report. It displays the distribution of the different bus commands during the traffic (compare the “*PCI Usage*” on page 44). More specifically, it is splitted up with respect to the different burst lengths. Burst lengths are displayed up to 65, where the length is the number of data phases. All longer bursts appear together as 66+.



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

## Latency

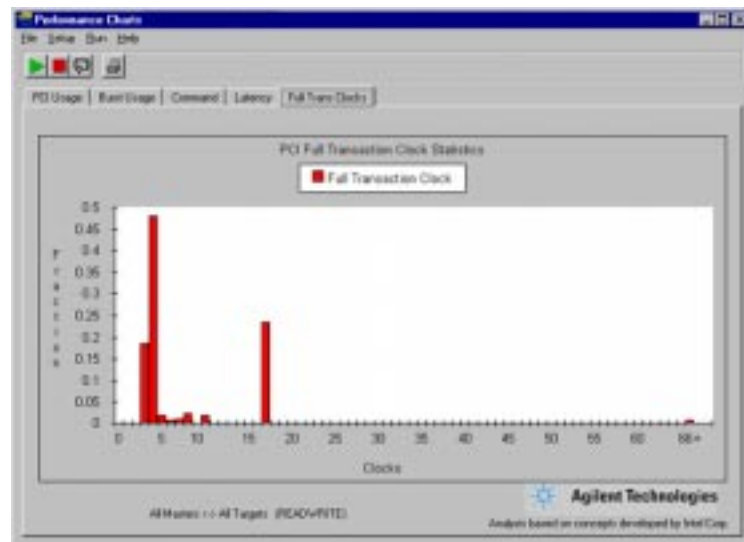
The diagram in this tab corresponds to the section 8.6 in the performance report. It resolves the distribution of the different latencies that occurred during bursts of the different lengths.



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


## Full Transaction Clocks

The diagram in this tab shows the distribution of the total lengths of transactions that were used for the communication between the selected master and target. Total length means the total number of bus cycles including wait states. The sum of all fractions in this diagram equals one.



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


## Interpreting the Report Output

After data traffic from the PCI 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 textual 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 43* for more information.



- Reduced Report Contents** You can choose which parts of the report are shown or omitted before starting the measurements. For this purpose go to the *Report* tab (see “*Setting Up the Report*” on page 33) in the Performance Setup window. This window opens, for example, when selecting the *Setup Window* item from the *Setup* menu in the Performance Report window.
- Navigation in the Report** For convenience, the report is structured in 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 master and target 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. Clicking a link leads 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 50. This example describes the sections 1 to 7.
  - work through the “*Using the Report Output for Performance Optimization*” on page 55. This example describes the section 8.
  - see the “*Report Reference*” on page 81 for a complete and detailed reference of the contents.

# Using the Report Output for Performance Evaluation

This example shows how the report sections 1 to 6 can be used to evaluate the performance of a PCI system or device.

For this example, a graphics card was used as device under test in a 32-bit system and a computer game as a benchmark. When running the benchmark, it seems that the game runs too slowly. Therefore, the PCI performance was examined using the PCI Performance Optimizer. The report sections that lead to the graphics card's shortcomings, are shown and explained in the following. Some of the report sections can also be viewed in charts.

Note, that current PC generations usually use AGP graphics cards instead of PCI graphics cards, because they often interfered with other PCI devices.

## General Information on the System under Test

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

```

1 GENERAL INFORMATION
=====

1.1 Test Settings
-----

Bus Speed ..... 33.00 MHz
Identified Masters:
  1. All Masters
Identified Targets:
  1. All Targets      Addr: 00000000\h
                      Size: FFFFFFFF\h
                      I/O and Mem
  3. Graphics         Addr: 000A0000\h
                      Size: 00010000\h
                      Memory

```

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

## Analyzing the PCI Throughput

The second report section displays the basic bus statistics. Here values like PCI throughput, bus utilization and efficiency are presented.

```

2 BASIC BUS STATISTICS
=====

  PCI Throughput ..... 0.9535 Megabytes/sec
                        ..... 7.6282 Megabits/sec
  PCI Utilization ..... 50.07 %
  Non-retry PCI Utilization ..... 3.60 %
  PCI Efficiency ..... 1.44 %
  Non-retry PCI Efficiency ..... 20.09 %

```

Compare the overall PCI throughput in the basic bus statistics to the throughput in the bus throughput statistics (report section 3).

```

3 BUS THROUGHPUT STATISTICS
=====

Master Target Throughput(MBytes/sec)
-----

                | All Masters
-----
Graphics      |      0.95
Other Target    |      0.00
All Targets     |      0.95

```

The table shows that no other target was involved in the data traffic. Due to the benchmark, the graphics card was the only device using the PCI bus. No other activities took place while the capture was taken.

The measured throughput of 0.9535 MByte/s seems to be the reason for the poor performance. The PCI bus supports 132 MBytes/s peak transfer rate. Therefore, a way for improving the data throughput to the graphics card must be found.

## Analyzing the PCI Utilization

In the basic bus statistics (report section 2), you can find the PCI utilization among others.

```

2 BASIC BUS STATISTICS
=====
PCI Throughput ..... 0.9535 Megabytes/sec
                  ..... 7.6282 Megabits/sec
PCI Utilization ..... 50.07 %
Non-retry PCI Utilization ..... 3.60 %
PCI Efficiency ..... 1.44 %
Non-retry PCI Efficiency ..... 20.09 %

```

The **PCI Utilization** value is 50.07 %. This means that the bus was used only in about half of the available clock cycles. During the remaining time it was idle.

Now compare the PCI Utilization to the Non-retry Utilization. The latter is the bus utilization without retries. Only 3.60 % of the bus cycles were used for data traffic without a retry. This means that the bus was mainly occupied by transporting retry overhead.

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 ..... 2.87 %
Retry Overhead ..... 46.47 %
Data Utilization ..... 0.72 %

5.1 Master Target Utilization
-----
| All Masters
-----
Graphics | 50.04 %
Other Target | 0.03 %
All Targets | 50.07 %

```

From this table, it can be seen that the bus mainly is occupied by retries (46.47 %) and that the graphics card basically is the only target that used the bus during the measurement: 50.04 % compared to a total utilization of 50.07 %.

## Analyzing the PCI Efficiency

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

```

2 BASIC BUS STATISTICS
=====
PCI Throughput ..... 0.9535 Megabytes/sec
                  ..... 7.6282 Megabits/sec
PCI Utilization ..... 50.07 %
Non-retry PCI Utilization ..... 3.60 %
PCI Efficiency ..... 1.44 %
Non-retry PCI Efficiency ..... 20.09 %

```

An efficiency of 1.44 % is very little, especially if there are not many devices requesting the bus concurrently. The Non-retry PCI Efficiency value only takes successful transfers into account and neglects the retries. But this value of 20.09 % still is very poor. 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
=====
PCI Byte Enable Efficiency ..... 99.89 %
PCI Time Efficiency ..... 1.44 %
Non-retry PCI Time Efficiency ..... 20.11 %
Retry Overhead = Non-retry Time Efficiency -
PCI Time Eff. .... 18.66 %

4.1 Master Target Efficiency
-----
| All Masters
-----
Graphics | 1.44 %
Other Target | 1.04 %
All Targets | 1.44 %

```

These statistics show a good PCI Byte Enable Efficiency value of nearly 100 %. This means, that transfers used the full bus width in most cases. Therefore, it is impossible to improve the card behavior with respect to this property.

On the other hand, the analysis of the retry behavior confirms what has already been seen when analyzing throughput and utilization: 18.66 % retry overhead is very much.

**NOTE** When testing a 64-bit system, an additional section is included in the report for efficiency analysis of 64-bit transfers. For more information please refer to “*Efficiency Statistics (Report Section 4)*” on page 88.

## The Bus Users Overview

In report section 6 the Bus Users Overview is displayed. This list leads to the device pair being responsible for the poor performance results.

The Bus Users Overview table contains indices that rate the performance of a device pair. Normally, you can compare the values of the several devices. In this case, however, a direct comparison is not possible because there is not enough traffic from other devices included for reasonable values.

```

6 BUS USERS OVERVIEW
=====

Calculated as: Utilization divided by Efficiency
-----
| All Masters
-----
Graphics      | 34.68
Other Target  | 0.02
All Targets   | 34.70

```

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 graphics card's index is 34.68, which is far too high.

Finally, the report section 7 deals with interrupt statistics and is not relevant for the considerations in this sample walk through.

## Summary for the Performance Evaluation Example

After the report sections 1 to 6 have been analyzed, the performance evaluation results in:

- the bus being used almost only by the graphics card,
- the graphics card using the bus only to the half of its capability,
- the graphics card using the bus mainly to transport retry overhead,
- the graphics card having a poor index in the “Bus Users Overview”

It is obvious that the graphics card should be improved or replaced, for example, by an AGP card.

## Using the Report Output for Performance Optimization

This example explains how the flaws of a particular PCI 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.

To gain this general overview on the system under test, please refer to the *“Using the Report Output for Performance Evaluation” on page 50*. There, the basic bus metrics PCI Throughput, PCI Utilization, PCI Efficiency, and the Bus Users Overview are evaluated with help of the respective sections of the performance report. The result is that the examined graphics card in the system under test shows a very poor performance and should be replaced.

From the report sections 1 to 6, it was derived that the retry behavior of the card is the reason for the poor PCI performance. In order to improve the PCI design of the card, the card’s traffic behavior must be analyzed in more detail.

For this purpose, the report section 8 will be analyzed during this example. The used system under test still is the same as in the other example. Section 8 lists detailed statistics about the communication between the specified master-target pair, in this case the traffic between the graphics card and **All masters**.

## Analyzing the Bus Utilization

The retry behavior of the graphics card was identified as the reason for poor performance. Therefore, the analysis of the bus utilization is useful. Report section 8.3 provides the results for the selected master target pair:

```

8.3 Bus Utilization
-----
Time Overhead ..... 2.85 %
Retry Overhead ..... 46.47 %
Data Phases ..... 0.72 %
-----
Sum of Utilization ..... 50.04 %

```

The **Sum of Utilization** indicates how intensively the selected device pair participated in the whole traffic on the PCI bus. Here, 50.04 % of all sampled clocks were used by this device pair.

Furthermore, this analysis indicates a remarkable retry overhead (46.47 %). However, it should be taken into account that only the traffic between graphics card and its master, the PCI bridge, was analyzed.

## Analyzing the Traffic Overhead

In this step it is examined, whether the master or the target device is responsible for the retry overhead. For this purpose, see the Time Overhead summary (report section 8.3.2). The time overhead analysis breaks down the values of the section 8.3. It points to the causes of the overhead, and it shows what type of overhead occurred.

8.3.2 Time Overhead				
-----				
Average Decode Speed (1 == fast).... <b>2.00 cycle(s)</b>				
	Overhead caused by			
	Master		Target	
			Both	
				Sum
	-----			
a) Address Phase	1.46 %		--	
b) Waits	0.00 %		1.46 %	
	-----			
1) First Word Latency (a+b)	1.46 %		1.46 %	
2) Retry Transactions	--		<b>94.22 %</b>	
3) Subsequent Latency	0.00 %		2.85 %	
	-----			
Sum ((1) + (2) + (3))	<b>1.46 %</b>		<b>98.54 %</b>	
			0.00 %	
				100.00 %



The average address decode speed of 2 clock cycles is all right and does not necessarily limit the performance.

The sum of all different overhead types is always 100 % and corresponds to the time overhead value found in report section 8.3. Out of this total traffic overhead, the master caused only 1.46 %. The graphics card (target), on the other hand, caused 98.54 %. It is obvious that the graphics card is responsible for the poor performance. The reason for the overhead is obvious as well: 94.22 % of the total overhead is caused by target retries.

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

Furthermore, it is remarkable that the target causes overhead by inserting initial waits (1.46 %) and subsequent waits (2.85 %). However, this is insignificant compared with the huge overhead caused by retries. It indicates that an examination of the inserted waits could help to find further ways of improving the device.

## Analyzing the Wait States

To analyze the source for the wait states during the communication, see the **Wait Histogram** (report section 8.3.5):

### 8.3.5 Wait Histogram

Waits	Master	Target
0	<b>50.00 %</b>	0.00 %
1	0.00 %	<b>2.75 %</b>
2	0.00 %	<b>3.47 %</b>
3	0.00 %	<b>43.77 %</b>
4	0.00 %	0.00 %

... continued ...

Every wait sequence that was inserted by one device—either master or target—is counted as a wait sequence of length zero for the other device. Therefore, all wait sequences appear in both columns. As a result the sum of both columns equals 50 %. The closer the value in the first row reaches 50 %, the less wait sequences this device has caused.

In this case, the table shows, that no wait states are inserted by the master. The target, however, inserted sequences of wait states, mainly three clocks long. These are probably waits inserted when performing retries. The target should be examined to see whether it can be prevented from performing retries.

## Analyzing the Byte Enable and Burst Behavior

This step examines the byte enable and burst behavior of the graphics card in order to find further ways for improvements.

The detailed analysis of the Byte Enable Efficiency shows that almost always the maximum of 4 bytes per data phase was transferred. This has been detected earlier in the bus statistics analysis in report section 4 (see *“Analyzing the PCI Efficiency” on page 53* for details).

This shows a good PCI design of the master device, in this case, the motherboard’s PCI bridge. It initiates write transfers to the graphics card and puts the data on the bus. The graphics card, as being the target, has no influence on the distribution of byte enables.

Now, regard the average burst length in the data communication of the selected device pair. For this purpose, see the report section 8.3.1:

```

8.3.1 Data Phase
-----
0 Bytes ..... 0.00 %
1 Byte ..... 0.00 %
2 Bytes ..... 0.00 %
3 Bytes ..... 0.00 %
4 Bytes ..... 100.00 %
-----
Average Byte Enable Efficiency ..... 100.00 %
Average Burst Length ..... 1.00 data phase(s)

```

As the diagram shows, the average burst length is only 1 data phase. Basically, the longer the bursts are, the better the performance. Their length is limited by the capabilities of the participating devices. Thus, this average burst length indicates very poor design and is a root cause for poor performance. The burst length depends on the behavior of both master and target and should, therefore, be analyzed in more detail.

## Analyzing the Bursts

Proper PCI design results in the devices being capable of performing burst lengths greater than 1. Because the average burst length in our case is only 1, it is obvious that the graphics card under test cannot handle bursts properly.

A detailed list of the different PCI bus commands used with bursts of a particular length is found in report section 8.3.6:

8.3.6 Burst Length over Command									
-----									
Brst	IO_REA	IO_WRI	MEM_RE	MEM_WR	CONFIG	CONFIG	MEM_RE	MEM_RE	MEM_WR
Len	D	TE	AD	ITE	_READ	_WRITE	ADMULT	ADLINE	ITEINV
-----									
0	0.00 %	0.00 %	0.00 %	<b>94.18 %</b>	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
1	0.00 %	0.00 %	0.00 %	<b>5.82 %</b>	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
2	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
3	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
... continued ...									

The **Burst Length over Command** table is particularly useful when different commands are used in the data traffic. This table shows which commands were used with higher burst length and, therefore, achieved the better performance.

In this example, however, only the plain Memory Write command was used and 94.18 % of all bursts had a length of 0, which means the target stopped the transaction with a retry. The remaining 5.82 % are bursts of length 1. Apparently, the card did not perform any bursts with more than 1 data phase.

In order to examine the quality of the transactions with different burst length, see the report section 8.4.1:

8.4.1 Efficiency over Burstlength			
-----			
Burstlength	Efficiency	Data Fraction	Bus Fraction
-----			
0	0.00 %	0.00 %	92.86 %
1	20.20 %	99.83 %	7.13 %
2	<b>33.33 %</b>	<b>0.02 %</b>	0.00 %
3	--	0.00 %	0.00 %
4	<b>66.67 %</b>	<b>0.09 %</b>	0.00 %
5	<b>71.43 %</b>	<b>0.06 %</b>	0.00 %
6	--	0.00 %	0.00 %
7	--	0.00 %	0.00 %
8	--	0.00 %	0.00 %
... continued ...			

This table reveals, that some bursts with lengths of 2, 4 and 5 cycles occurred, but there were so few of them that they do not appear in the **Burst Length over Command** table (report section 8.3.6). Note that the PCI efficiency is higher for these longer bursts.

The main result, that can be read from this table, is that the graphics card actually *is* capable of bursts longer than one (probably under certain conditions only) and that the efficiency would remarkably increase if the card was able to handle these bursts more often.

From section 8.3.6 it can be seen that the only used command is “Memory Write”. Although this is not an extended command, it is commonly used for graphics cards. The transaction was terminated after the first data phase in most cases. 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 commands in the device communication can also be the reason for poor performance. Graphics cards are passive and usually only receive data from a master. Therefore, they have no chance to select the commands and there is no way to improve the graphics card in this respect.

However, the behavior of the graphics card in response to different PCI commands during the data transfers affects PCI performance. The following table displays, how the different transactions were handled.

8.3.4 Command Termination				
-----				
Command	Fraction of Terminations			
	Retry	Disc. w/o data	Disc. w/ data	Normal
-----				
SPECIAL	0.00 %	0.00 %	0.00 %	0.00 %
IO_READ	0.00 %	0.00 %	0.00 %	0.00 %
IO_WRITE	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_4	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_5	0.00 %	0.00 %	0.00 %	0.00 %
MEM_READ	0.00 %	0.00 %	0.00 %	0.00 %
<b>MEM_WRITE</b>	<b>94.18 %</b>	<b>5.63 %</b>	<b>0.09 %</b>	<b>0.10 %</b>
RESERVED_8	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_9	0.00 %	0.00 %	0.00 %	0.00 %
CONFIG_READ	0.00 %	0.00 %	0.00 %	0.00 %
CONFIG_WRITE	0.00 %	0.00 %	0.00 %	0.00 %
MEM_READMULTI	0.00 %	0.00 %	0.00 %	0.00 %
MEM_READLINE	0.00 %	0.00 %	0.00 %	0.00 %
MEM_WRITEINVA	0.00 %	0.00 %	0.00 %	0.00 %
-----				
Total:	<b>94.18 %</b>	5.63 %	0.09 %	<b>0.10 %</b>

This table shows, that only 0.10 % of all transfers finished normally. This is a very poor result. Any successful transaction without premature termination is listed in the “normal” column. In this case, however, 99.9 % of all transfers were terminated with retries or disconnects.

5.63 % of the terminations were “Disconnects without data”. This means that the target was unable to receive the last transferred data cycle. On the other hand, “Disconnect with data” indicates, that the target did receive the last data cycle successfully, but had to terminate the transaction for some reason. The latter is less critical, because the master does not need to resend the last transferred data. This happened in only 0.09 % of the cases.

The existence of disconnects proves, that the master *did* try to use burst lengths greater than 1. 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 master not being responsible for the poor performance.

To identify the device that terminated the transactions, refer to the termination statistics (report section 8.5):

8.5 Termination Statistics			
-----			
Average Number of retries if retried ..... <b>16.20</b>			
8.5.1 Termination Burst Histogram			
-----			
Burstlength	Transfer was stopped by ...		
	Master	Target	Arbiter
-----			
0	0.00 %	<b>94.18 %</b>	--
1	0.10 %	<b>5.72 %</b>	--
2	0.00 %	0.00 %	--
3	0.00 %	0.00 %	--
... continued ...			
63	0.00 %	0.00 %	--
64	0.00 %	0.00 %	--
65	0.00 %	0.00 %	--
66+	0.00 %	0.00 %	--
-----			
Sum	<b>0.10 %</b>	<b>99.90 %</b>	--

The **Average Number of retries** of 16.20 means that the master had to retry more than 16 times in average until a successful data transfer could be performed. This is a remarkably poor value.

In the termination burst histogram (report section 8.5.1) is listed which bursts were terminated by which device. The last row shows the sum for all bursts. These values, of course, are the same as in the Command Termination table (report section 8.3.4).

From the last row, it can be seen that all “normal” terminations (representing the successful transactions) were caused by the master. All other terminations were caused by the target. A termination at burst length 0 means, that the transaction was stopped already after the address phase. This is a target retry in most cases and occurred in 94.18 % of all transactions in the test.

Another 5.72 % of the terminations were caused by the target at burst length 1. This corresponds to the figures in the Command Termination table, where 5.72 % of terminations with burst length 1 are target disconnects with (0.09 %) or without (5.63 %) data.

As a result, these two tables show that the target stops too many transfers by retries and disconnects at the first data phase. If it disconnects, it uses “Disconnect without data” although the target basically is capable of using “Disconnect with data”, too.

## Summary of the Performance Optimization Example

The selected maser-target pair in this example was the graphics card as target and “All Masters”. However, the only master accessing the graphics card is the PCI bridge. The graphics card was found to be the device with the worst performance in the system under test.

After analyzing the data traffic to the graphics card in detail with the Agilent E2926A/B PCI Performance Optimizer, the following conclusions can be drawn:

- The target answers the vast majority of transactions with a retry.
- The remaining transactions are terminated with a target disconnect mostly during the first data phase and mostly without any data being transferred.
- Furthermore, the target inserts wait states (initial and subsequent wait states).
- It is unable to make use of its burst handling capabilities.

This results in massive performance problems. Possible reasons could be, for example:

- A slow graphic chipset (due to its design: no dual ported RAM, too small buffers, etc.).
- The card may have no FIFO to buffer data received from the PCI bus.
- The card needs too much time to process the data it receives from the PCI bus.





# Re-Using 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 E2926A/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 re-used.

**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, change 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 E2926A/B Opt. 200 Performance Optimizer is described in *“Printing and Exporting Test Results” on page 66.*

**Re-Using Data Files** How to view or re-use previously exported files is explained in *“Re-Using Previously Saved Data” on page 69.*


# Printing and Exporting Test Results

After successfully running the test measurements, you probably 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 re-run the same test after the device vendor or manufacturer has re-designed 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 textual report in the Performance Report window and the contents of the Bus Activity Lister.

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

The Print button will send the complete report as listed in the report window to the printer, not only the currently displayed range.

- Bus Activity Lister window

Only the range of bus cycles is printed that currently is visible in the Bus Activity Lister window.

## Exporting the Report

The textual report of the Agilent E2926A/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, proceed as follows:

- 1 In the Performance Report window, select *Export to File* from the *File* menu.
- 2 In the file dialog box, enter a name for the text file and click *Save*.

## Exporting the Trace Memory

The performance of your system under test is calculated 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. This is done together with trigger information, hardware and software versions and some more.

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,
- Bus Activity Lister 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 E2926A/B software.

**NOTE** If “Repetitive Run and Upload” is enabled, only the data of the most recent upload are stored in the waveform file. Later analysis of this file will return other results than those of the analysis of the complete set of uploads. See “*Setting Up the Data Capture*” on page 37 for details on repetitive runs.

## Exporting the Bus Activity Listing

The Bus Activity Lister presents the contents of the trace memory with respect to the single bus transactions along with their individual performance values. Instead of saving the pure contents of the trace memory you can also choose to export this listing including the performance evaluations.

To export the contents of the Bus Activity Lister, select *Export to file* from the *File* menu.

The data will be stored in ASCII format and can be viewed with any editor. However, to identify the different values in the data sets, it is recommended to view the files within the Bus Activity Lister.

## Exporting the Test Settings

For communicating your test results with other people, 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 of the application, 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 bus, and the connected master and target devices. Furthermore, this file contains all settings done regarding the device behavior, including their REQ# and GNT# lines and the selected names of the devices. And all the settings needed for the performance measurements are stored in this file as well.

# Re-Using Previously Saved Data


The instructions found here explain how to re-use 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 E2926A/B software is running in offline/demo mode and the PCI Performance Optimizer option is enabled.

## Loading Setup Files

The setup files (.bst files) contain the complete setup of the Agilent E2926A/B PCI Performance Optimizer. This includes the used test hardware and software, the properties of the PCI bus and the connected master and target devices. Furthermore, these files contain all settings done regarding the device behavior, their names, address spaces, and their REQ# and GNT# lines.

To re-use a setup file, click the Load button  in the main window of the GUI, save the current settings if required, 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 do 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, Bus Activity Lister 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 three result windows of the Performance Optimizer. Thus, you basically do not need to separately save the text files containing the report and the bus activity listing.

**Loading Report Files** The report files contain the results of previous tests in a textual manner. They are calculated from the data found in the respective setup files and waveform files. Report files cannot be imported into the GUI directly. You can use any text editor instead.

**Loading Bus Activity Lister Files** These files, finally, keep the contents of the bus activity lister from previous tests in a textual manner. They cannot be imported into the GUI directly. You can use any text editor instead.

# Going Further Into Details

The Agilent E2926A/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 system to examine any device that communicates via the PCI bus.

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

Assuming you specified a proper test setup, the basic information is found in the Performance Chart window. The different charts present the performance of the selected devices:

- If the selected master-target pair is *All Masters -> All Targets*, the window displays the different performance properties of the whole system under test.
- If the master-target pair is selected that you want to examine in particular, the charts in the window only contain this information.

Compare the two results to see whether existing performance problems only affect 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.

- 64-Bit Data Usage

For good performance, the fraction of 64-bit data transfers should be as high as possible. However, some bus commands as, for example, I/O transfers require 32-bit transfers, resulting in slower traffic.

- Command Usage

Extended commands like *Memory Read Line*, *Memory Read Multiple*, and *Memory Write and Invalidate* are more efficient than others. The usage of I/O commands, however, should be avoided where possible, because they consume precious CPU time.

Also very important for good performance is a high fraction of data phases for each command and as little overhead as possible.

- Bus Utilization

Again, 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 time (blue).

- PCI Efficiency

The efficiency is a measure that directly represents the quality of the data communication between the regarded PCI devices. It is affected by every change of the behavior of the participating devices. Thus, optimizing the device behavior will result in the optimum efficiency.

- Burst Length (on the *Burst Usage* tab)

For good performance, the preferred burst length should be adjusted to the requirements of the participating PCI 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 system is whether all devices always stick to the rules for efficient data traffic (shortly introduced in “*Rules for Proper PCI 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 system.

**Long Bursts** The most important rule for high performance is to use 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.

**Extended Commands** When developing a PCI device, it is recommended to implement extended commands rather than plain commands:

- Use *Memory Write and Invalidate* to write multiple cachelines of aligned data.
- Use *Memory Read Multiple* for data held in more than one cache line (to read more than 1 memory line).
- Use *Memory Read Line* to read one complete cacheline from a memory.

If the extended bus commands are not used in the communication, the master did not attempt to use them. If the master uses these commands but they show no effect, the target does not support them.

**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 precious 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 target behavior like reducing latencies or avoiding wait states, see “*Improving Target Behavior*” on page 75.

## 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 test your own PCI chipset or card design, you can verify the efficient behavior in any environment by checking the rules. And when finding 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 giving 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 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, regard the *“Bus Users Overview (Report Section 6)” on page 91* to see which devices have to handle the majority of the traffic.

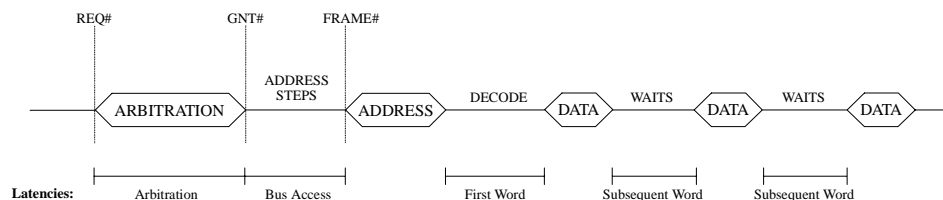
# Improving Target Behavior

In many cases traffic problems occur in PCI systems because a master attempts to initiate a transfer to or from a target that cannot handle the PCI command, the burst length, etc. This gives rise to improve the target behavior to be able to communicate at the highest speed.

## Minimize Latencies

One of the basic rules to speed up the traffic is to minimize the latencies. Proper PCI design provides rapid replies between the devices and avoids the insertion of waits during transactions.

**Latency Definitions** The following figure schematically shows the several latency types that can occur during a data transaction. These latencies are displayed (in different colors) on the *Latency* tab in the Performance Charts window.



In chronological order there are:

- *Arbiter Latency* (blue).

This is the time between a master asserting the REQ# signal and the arbiter giving the GNT# signal to this master.

A large number of clocks wasted for arbiter latency either indicates the bus being permanently busy (high bus utilization) or a slow arbiter with a poor arbitration algorithm.

- *Bus Access Latency* (yellow).

This is the time between the arbiter asserting the GNT# signal to a master and this master driving FRAME# and the address onto the bus.

The bus access latency also depends on the latency counters of the other PCI devices. A device may use the bus until its latency counter has expired—the other devices have to wait to access the bus.

Other reasons for high bus access latency may be, that the master needs time to drive the address onto the bus (master received GNT# unexpectedly fast) or performs address stepping. Address stepping means that the master drives the address onto the bus using more than one clock cycle (to avoid crosstalk and/or current peaks).

**NOTE** Both arbitration latency and bus access latency require the REQ# lines and the GNT# lines to be connected to the Agilent E2926A/B testcard as described in “*Master Identification*” on page 29.

- *First Word Latency* (red).

This 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 target and/or many initial waits, for example due to locked target resources.

Note, that retries are not considered in this latency. For these cases there is an extra column in the latency histogram in section 8.6 of the performance report that is called “First Word Latency with Retries”.

- *Subsequent Word Latency* (green).

This is the sum of all wait states that occurred between the first and the last data phase of a transaction. These wait states can be inserted both by the master and the target.

You can see from the latency diagram, for which burst lengths additional bus clocks were inserted by which device. This allows you to draw conclusions about the shortcomings of the devices: for example, whether master, target, or arbiter is responsible for waits, and what the reasons might be.

**NOTE** Unfortunately, the names for the latency types may differ or may be used differently in other software or literature. When reporting test results with latency statistics, it is recommended to include the definitions listed above.

## Command Termination

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

To view the different types of transaction terminations see the *“Command Termination (Report Subsection 8.3.4)” on page 102.*

### 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.

- Normal

This is the preferred command termination meaning the transaction was finished successfully as initiated by the master.

- Retry

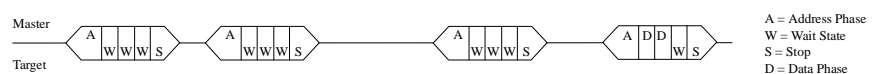
Retries are used by the targets during the bus address phase if they cannot supply or accept the data. See *“Target Retries” on page 77* for details.

- Disconnect

If a target faces problems during the data phases of a transaction it can terminate the transfer with several types of disconnects. See *“Target Disconnects” on page 78* for details.

## Target Retries

A target terminates a transaction with a retry if it cannot supply or accept data, for example because it is too slow. The following figure shows how retries are typically performed on the bus:



The master tries to start a transfer and drives the address onto the bus. The addressed target answers with three waits (as an example) and finally stops the transaction with a retry. From this behavior the master notices that the target cannot transfer any data at the moment. Therefore, the master tries again later, but receives (in the example) two more times the same reply: three waits and a termination. At the fourth attempt, the target eventually is ready and the master can successfully transfer the data (two data phases in this example).

**Reasons for Target Retries** Possible reasons for the target needing some preparation time are:

- the data must be loaded first, for example, from a disk,
- the target's memory is locked,
- the target is equipped with too small internal buffers,
- the target has a slow internal logic to handle the data.

Basically, forcing retries is the best option for the target in such situations. Alternatively, it 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 target returns a retry as described, other users can access the bus in between, which increases the performance.

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

PCI efficiency and utilization consider the time that is used for retries as busy bus time. **Non-retry efficiency** and **non-retry utilization**, however, exclude retry times from the calculations. Thus, the difference of efficiency and non-retry efficiency is the efficiency loss due to retries. The same applies for the PCI utilization.

## Target Disconnects

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

A disconnect may occur either **with data** (disconnect type A and B, where the current data cycle will be completed before termination) or **without data** (disconnect type 1 and 2, aborts current data transfer). A "Disconnect without data" is worse than a "Disconnect with data", because in the first case the master needs to send/read the last sequence of data again.

**Reasons for Disconnects**

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

- The target is too slow to complete the data phase.

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

- The target 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 specification revisions.

- The transfer crosses the target's address boundary.
- The burst memory transfer crosses the cache line boundary.

## Using the Bus Activity Lister

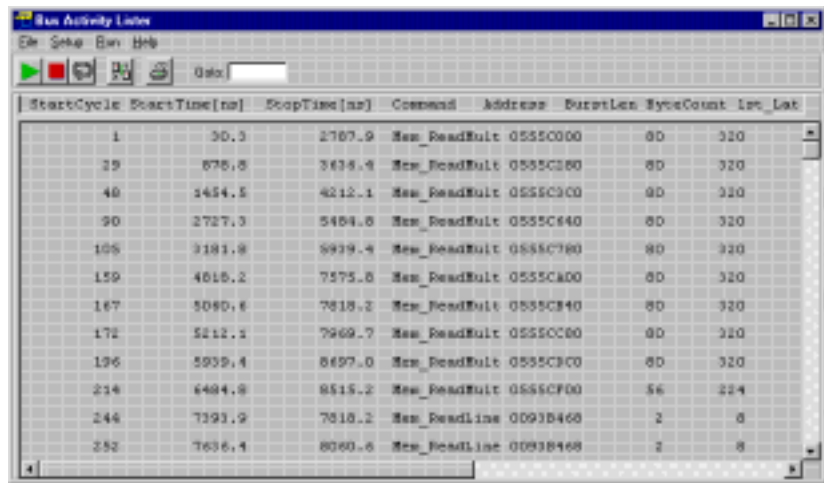
The Agilent E2926A/B Opt. 200 Performance Optimizer provides another feature for the exact analysis of the bus traffic: the bus activity lister. This lister presents the data traffic on the PCI bus in sequential order similar to the transaction lister known from the PCI Analyzer. The important difference is, that the bus activity lister works with respect to the performance properties of the traffic.

The Bus Activity Lister displays all bus traffic that was recorded in the trace memory. The performance values are calculated for each transaction separately, neglecting which devices participated.

Several bus cycles are regarded as one bus activity if they belong to the same transaction. However, for each bus activity, only those properties are displayed in the lister that are of interest in a performance analysis. This excludes, for example, the data bits during the data phases.

To view the contents of the bus activity lister, proceed as follows:

- 1 Fill the trace memory with traffic data, either by loading a waveform file or by running a post-processed analysis.
- 2 Open the Bus Activity Lister window by selecting *Bus Activity Lister* from the *Performance* menu in the main window.



The screenshot shows the 'Bus Activity Lister' window with a menu bar (File, Setup, Run, Help) and a toolbar with icons for running, pausing, and other functions. A 'Goto:' field is present. The main area contains a table with the following columns: StartCycle, StartTime[ns], StopTime[ns], Command, Address, BurstLen, ByteCount, and Lst\_Lst. The table lists 25 entries of bus activity, primarily 'Mem\_ReadBurst' commands to various memory addresses, with the last two entries being 'Mem\_ReadLine' commands.


StartCycle	StartTime[ns]	StopTime[ns]	Command	Address	BurstLen	ByteCount	Lst_Lst
1	30.3	2767.9	Mem_ReadBurst	0555C000	80	320	
29	876.8	3638.4	Mem_ReadBurst	0555C280	80	320	
49	1454.5	4212.1	Mem_ReadBurst	0555C3C0	80	320	
90	2727.3	5484.8	Mem_ReadBurst	0555C640	80	320	
108	3181.8	5939.4	Mem_ReadBurst	0555C780	80	320	
159	4616.2	7575.8	Mem_ReadBurst	0555CA00	80	320	
167	5040.6	7818.2	Mem_ReadBurst	0555CB40	80	320	
172	5212.1	7969.7	Mem_ReadBurst	0555CC80	80	320	
196	5939.4	8697.0	Mem_ReadBurst	0555CEC0	80	320	
214	6484.8	8515.2	Mem_ReadBurst	0555CF00	56	224	
244	7393.9	7818.2	Mem_ReadLine	0D93B468	2	8	
252	7636.4	8060.6	Mem_ReadLine	0D93B468	2	8	

The scroll bars can be used to scroll horizontally and vertically through the list. Alternatively, you can directly display a line by entering a line number in the *Goto:* field and pressing the return key.

For more details on the displayed data, please refer to the online help.

#### Combining the Different Listers

The listers of the PCI Analyzer together with the bus activity lister can be combined with a cross reference functionality. This allows to mark the respective lines in the different listers simultaneously to easier locate and examine the contents.

To use the cross reference function, mark one or more lines in any of the listers and click the Cross Reference button .

After that, all listers display the contents derived from the same range of traffic samples in the trace memory in their own view.

If Repetitive Run and Trace Memory Upload is enabled, the listers always show the samples loaded with the most recent recording.



# Report Reference

The Agilent E2926A/B Opt. 200 Performance Optimizer presents the results of performance measurements in several forms. One of these is the textual 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, that relate to particular sections in the report 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 Performance Optimizer Test” on page 27*. If you need to know how to run the tests and how to interpret the results, see *“Measuring System or Device Performance” on page 41*.

**Example Transfers** Some few example data transactions are introduced in *“Example Transfers” on page 82*. They are used to explain typical data transfers on the PCI bus. Please make yourself familiar with them first, because it will help to understand the meaning of the various measures used in the report.

**Report Sections 1 to 7** The report sections 1 through 7 list general information and statistics about the PCI bus and the connected devices. Explanations of these sections are found in *“Bus Statistics (Report Sections 1 to 7)” on page 83*.

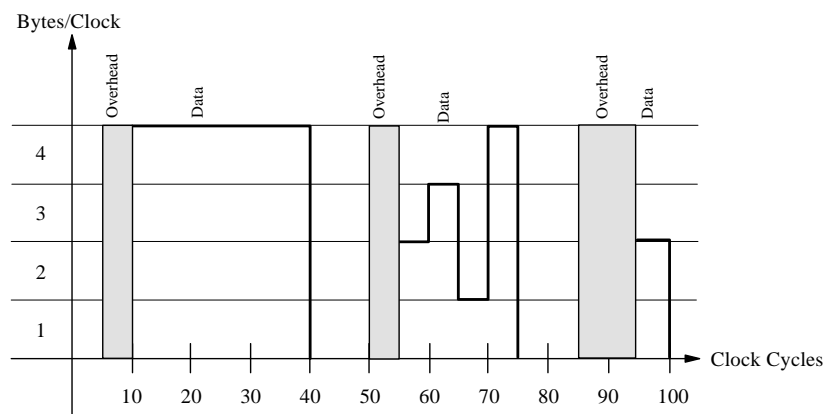
**Report Section 8** The report section 8 contains detailed statistics on the data communication of an arbitrary master or target device or a pair of devices. The explanations of section 8 are found in *“Master-Target Pair Measurements (Report Section 8)” on page 93*.

**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 115*.

# Example Transfers

The example introduced here was used to run a performance test with the Agilent E2926A/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 a sequence of 100 clock cycles that contain three data transactions as they typically occur on PCI buses. A 32-bit system is assumed, but it applies for 64-bit systems as well.



In the diagram above, the clock cycles are indicated horizontally. The vertical axis represents the **address/data** bits. In 32-bit systems a maximum of four bytes can be transferred per clock cycle. The actual number of transferred bytes is determined by the **byte enable** bits.

**The Example Transfers** The figure shows three transfers within a time of 100 clock cycles:

- The first transaction starts on clock cycle 5 and ends on clock cycle 40. Clock cycles 5 to 10 contain **transaction overhead** which is used to initiate the transfer (**address phase** and **address decoding**). For this transfer the full bus width was used during the **data phases**: 4 bytes were transferred in each clock cycle (clock cycles 11 to 40).
- The second transaction starts at clock cycle 50 and is 25 clocks long. The overhead is 5 cycles. In contrast to the first transfer, a full bus width was only used between clock cycle 70 and 75. Only those bytes during a data phase, that are enabled, are counted as transferred **data**.
- The third transaction (clock cycles 85 to 100) uses only 2 bytes of the bus width. The overhead takes 10 clock cycles.

The example does not take into account which devices communicate via the PCI bus. However, the Agilent E2926A/B Opt. 200 Performance Optimizer can evaluate the traffic of different devices separately. Masters are identified by their GNT# lines, targets 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 as the bus throughput, the bus efficiency, the bus utilization, the bus users overview, and the interrupt protocol.

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

## General Information (Report Section 1)

The first report section provides general information on the system under test: the test settings and information about the statistical base. It also holds a list of all identified masters and targets.

```

1 GENERAL INFORMATION
=====
1.1 Test Settings
-----

Bus Speed ..... 33.00 MHz
Identified Masters:
  1. All Masters
Identified Targets:
  1. All Targets      Addr: 00000000\h
                      Size: FFFFFFFF\h
                      I/O and Mem

  2. Other Target     Addr: 00000000\h
                      Size: 00000000\h
                      I/O and Mem

  3. Target_1         Addr: 000A0000\h
                      Size: 00010000\h
                      Memory

Selected Pair:      All Masters -> Target_1

```

In detail there are:

- Bus Speed.

According to the PCI specification this value should be either 33.00 MHz or 66.00 MHz.

- Identified Masters.

In this example, no particular master devices are identified. To identify which transactions belong to which masters, the masters' GNT# lines need to be connected to the Agilent E2926A/B testcard. If any masters were set up to be identified in the test, they would be listed here.

- Identified Targets.

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

The target *Target\_1* including its address space was set up by the user. The statement below the address space indicates, whether only memory or I/O accesses or both are considered in the test.

If the user specified one or more targets, the entry *Other Target* appears in the list, too. This entry represents all targets other than the specified one(s). Therefore, there is no reasonable address range to be displayed here.

- Selected Pair.

The PCI device or device pair, of which the data traffic was to evaluate in the test. In the example test the selection *All Masters -> Target\_1* caused the complete traffic to *Target\_1* to be considered in the measurements.

**Statistical Base** 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 precise. The values in this section are given in absolute numbers. All other report sections (except the statistical base of the master-target pair report in section 8) contain relative numbers to keep the reports of different test sessions comparable.

#### 1.2 Statistical Base

-----

Test covered .....	100 clocks
Test covered .....	0.00000303 sec
No of captured address cycles .....	20 clocks
No of captured data cycles .....	55 clocks
Bytes transferred .....	180 bytes
No of Interrupts occurred .....	0

In detail there are:

- Test covered

This is the 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 33 MHz.

- No of captured address cycles

This is the total number of initiated transactions, no matter whether they were completed successfully or not.

- No of captured data cycles

This is the number of data phases recorded during the test. The ratio between address and data phases is important for the performance evaluation.

- Bytes transferred

This value tells how many bytes actually were transferred between the masters and targets. Only those bytes in the data phases are counted that are indicated by the byte enable bits.

- No of Interrupts occurred

This is the total number of interrupts that occurred during the test.

## 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 Throughput ..... 56,6539 Megabytes/sec
                  ..... 453,2310 Megabits/sec
PCI Utilization ..... 75.00 %
Non-retry PCI Utilization ..... 75.00 %
PCI Efficiency ..... 60.00 %
Non-retry PCI Efficiency ..... 60.00 %

```

In detail there are:

- PCI Throughput.

The throughput is the amount of data transferred per time. In the example, 180 bytes are transferred in 100 bus cycles, 30.3 ns each. The resulting throughput is 56.6 MByte/s or 453.2 Mbit/s.

- PCI Utilization.

The utilization is the ratio between used bus cycles and unused (idle) cycles. In the example, during 75 clocks—of a total of 100 clocks—the bus was busy. Therefore, the utilization is 75 %.

- Non-retry PCI Utilization.

This value omits all transactions that were terminated with a target retry. Hence, only the bus cycles of successful data transfers are considered.

- PCI Efficiency.

The efficiency indicates the quality of the communication on the bus. It is derived from throughput and utilization. In the example, 180 bytes were transferred within the 75 busy clock cycles. Up to four bytes can be transferred per cycle. This makes a theoretical maximum of 300 bytes during the 75 cycles, out of which 180 bytes are 60 %.

- Non-retry PCI Efficiency.

This value is calculated only on those clocks that are not part of transactions terminated with a target retry. In other words, it is the efficiency of the successful transactions.

## Bus Throughput Statistics (Report Section 3)

The report section 3 is called Bus Throughput Statistics and holds a table analyzing the data throughput between all master-target pairs. The values can be compared with each other to find the devices with highest or lowest throughput.

```

3 BUS THROUGHPUT STATISTICS
=====

Master Target Throughput (MBytes/sec)
-----
                        | All Masters
-----
Target_1                |    56.65
Other Target            |    0.00
All Targets              |    56.65

```

The masters (in the top row) and the targets (left column) are the same that are listed in report section 1.1 (*Test Settings*). The entries *All Masters* and *All Targets* 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 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 master-target pairs. Finally, in 64-bit systems, there also is a subsection 4.2 presenting the distribution of 32-bit transfers and 64-bit transfers.

```

4 EFFICIENCY STATISTICS
=====

PCI Byte Enable Efficiency ..... 81.81 %
PCI Time Efficiency ..... 73.34 %
Non-retry PCI Time Efficiency ..... 73.34 %
Retry Overhead = Non-retry Time Efficiency -
                  PCI Time Eff. .... 0.00 %

```

In detail, there are:

- **PCI Byte Enable Efficiency**

The byte enable efficiency indicates how many bytes in average the devices could transfer via the PCI bus per clock cycle. The maximum is four bytes (32-bit).

In the example, 180 bytes were sent during 55 data cycles. 180 bytes out of a maximum of  $4 * 55 = 220$  bytes results in 81.81 %. This reveals, that not all transactions did use all four bytes of the bus.

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.

- **PCI Time Efficiency**

The time efficiency shows the average fraction of clock cycles in a transaction that are data cycles. If a 4-cycle transaction contains one address phase and three data phases, the time efficiency is 75 %.

- **Non-retry PCI Time Efficiency**

The non-retry time efficiency simply neglects all transactions that are terminated by the target with a retry. Thus, only those transactions are considered, during which data was transferred successfully.

- **Retry Overhead.**

The retry overhead is the efficiency loss due to target retries. It is calculated as the difference between the non-retry time efficiency and the time efficiency.



**Master-Target Efficiency** The table in subsection 4.1 shows the PCI efficiency of the data traffic between all master-target pairs. The values can be compared in order to find the device pair with the lowest performance.

4.1 Master Target Efficiency	
-----	
	All Masters
-----	
Target_1	60.00 %
Other Target	0.00 %
All Targets	60.00 %

The masters (in the top row) and the targets (left column) are the same that are listed in report section 1.1 (*Test Settings*). The entries *All Masters* and *All Targets* 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 .....	0.00 %
Master tried 64 but Target refused	50.00 %
-----	
32 Bit Data Transfers .(sum).....	50.00 %
64 Bit Data Transfers .....	50.00 %

The 64-bit bus statistics list:

- Pure 32 Bit Data Transfers

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

- Master tried 64 but Target refused

These are the transactions that were initiated by the master as 64-bit transfers, but rejected by the target, 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 bus. Then, in subsection 5.1, a table is displayed that analyzes the bus usage of the specified master-target 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 master-target pairs, that mainly occupy the bus.

```

5 BUS UTILIZATION STATISTICS
=====

Overhead Utilization ..... 20.00 %
Retry Overhead ..... 0.00 %
Data Utilization ..... 55.00 %

5.1 Master Target Utilization
-----
| All Masters
-----
Target_1 | 75.00 %
Other Target | 0.00 %
All Targets | 75.00 %

```

In detail, this section covers:

- Overhead Utilization

The overhead utilization indicates how much of the time the bus was occupied for transferring overhead data. In this value only transactions are considered that were not terminated by a target retry.

In the example, the first and the second transfer use 5 clocks for overhead transfer each, the third uses 10 clocks. This results in 20 % overhead utilization: 20 of 100 clocks were used for overhead transfer.

- Retry Overhead

The retry overhead is the fraction of overhead caused by the retries. In the example no retries occurred. Hence, the retry overhead is zero.

- Data Utilization

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

In the example, the data utilization is 55 % (55 out of 100 bus cycles were data phases).

## 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 Masters	
-----		
Target_1		1.25
Other Target		0.00
All Targets		1.25

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 master-target pair.

A high index indicates a device with a poor PCI 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 1.25 (75 % divided by 60 %).

The masters (in the top row) and the targets (left column) are the same that are listed in report section 1.1 (*Test Settings*). The entries *All Masters* and *All Targets* always appear in this table, the others were specified by the user during test setup.

## Interrupt Latency (Report Section 7)

The interrupt statistics are of special interest for users who need detailed information on BIOS behavior and the interrupt handling routines of the operating system. Any interrupt occurring during the performance test is registered by the Agilent E2926A/B Opt. 200 Performance Optimizer. Although the different interrupts do not resolve which device requested the interrupt, the interrupt handling influences the system performance. Thus, it is a factor to be considered when evaluating a system under test.

### Interrupt Latency

The interrupt latencies are measured in PCI clocks and show the average amount of time an interrupt request took to be acknowledged by the operating system. This requires that PCI interrupt acknowledge is not disabled by BIOS. The values are measured for all PCI interrupts INTA, INTB, INTC, and INTD separately.

#### 7 INTERRUPT STATISTICS

=====

```
Average INTA to Int_Ack Latency .... -none-
Average INTB to Int_Ack Latency .... -none-
Average INTC to Int_Ack Latency .... 9.17 clocks
Average INTD to Int_Ack Latency .... -none-

Average Overall Interrupt Latency .. 9.17 clocks
```

#### 7.1 Interrupt Latency Histogram

-----

Clocks	INT A	INT B	INT C	INT D
0	0.00 %	0.00 %	0.00 %	0.00 %
2	0.00 %	0.00 %	0.00 %	0.00 %
4	0.00 %	0.00 %	0.00 %	0.00 %
8	0.00 %	0.00 %	85.37 %	0.00 %
16	0.00 %	0.00 %	14.63 %	0.00 %
32	0.00 %	0.00 %	0.00 %	0.00 %
64	0.00 %	0.00 %	0.00 %	0.00 %
128	0.00 %	0.00 %	0.00 %	0.00 %
256	0.00 %	0.00 %	0.00 %	0.00 %
512	0.00 %	0.00 %	0.00 %	0.00 %
1024	0.00 %	0.00 %	0.00 %	0.00 %
2048	0.00 %	0.00 %	0.00 %	0.00 %
4096	0.00 %	0.00 %	0.00 %	0.00 %
8192	0.00 %	0.00 %	0.00 %	0.00 %
16384	0.00 %	0.00 %	0.00 %	0.00 %
16384+	0.00 %	0.00 %	0.00 %	0.00 %

First the average interrupt latencies of the four interrupts INTA, INTB, INTC, and INTD are listed followed by the average latency off all interrupts together.

**Interrupt Latency Histogram** Then, in report subsection 7.1, a detailed list is printed that reveals the exact fraction of interrupts that was acknowledged within a certain number of clock cycles as indicated in the first column.

**NOTE** The values presented in this table do not refer to the example data traffic of 100 clock cycles. In the example no interrupts occurred and, therefore, this table is empty in the report.

## Master-Target Pair Measurements (Report Section 8)

In report section 8 and its subsections the detailed analysis of the data traffic between a particular master-target 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 Master/Target Pair” on page 36*. Otherwise, the default pair and traffic direction will be used: “All Masters” and “All Targets”, “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 82*. 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 master's view. The figure below shows an example how the header may look like.

```
8 MASTER - TARGET PAIR:      All Masters <-> Target_1      (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 larger amount of data traffic.

```
8.1 Statistical Basis
-----
This pair was busy on .....          75 clocks
This pair was busy on .....          0.00000227 sec
No of captured address cycles .....    20 clocks
No of captured data cycles .....       55 clocks
Bytes transferred .....               180 bytes
```

In detail, there are:

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

Comparing these values to the statistical basis of the total bus traffic listed in “*General Information (Report Section 1)*” on page 84 allows conclusions on how intensively this device or device pair takes part in the communication on the PCI bus.

## Bus Usage (Report Subsection 8.2)

This report section lists the different waiting times for the selected device pair to get access to the bus after a bus request. It also reports how much the device pair occupied the bus. These waiting times, referred to as the latencies, depend on the algorithm of the bus arbiter.

### 8.2 Bus Usage

```
-----
Bus idle or otherwise busy ..... 25.00 %
Waiting for GNT# with bus idle ..... 0.00 %
Waiting for GNT# with bus busy ..... 0.00 %
GNT# to address phase ..... 0.00 %
Bus occupied ..... 75.00 %
```

In detail, there are:

- Bus idle or otherwise busy  
This is the fraction of time the bus was either idle or used by other device pairs than the selected one.
- Waiting for GNT# with bus idle  
This is the fraction of time the selected master waited for bus access after a request while the bus was idle.
- Waiting for GNT# with bus busy  
This is the fraction of time the selected master waited for bus access after a request while the bus was busy.
- GNT# to address phase  
This is the sum of all times between the arbiter granting the bus to the master and the master driving the address phase on the bus.
- Bus occupied  
This is the time the selected device pair occupied the bus.

The sum of all five values in this report section obviously is 100 %.

A complete analysis of the different latencies over different burst lengths is found in the “*Latency Histogram (Report Subsection 8.6)*” on page 111.

**NOTE** The waiting times after a bus request can be examined only if the REQ# and GNT# lines of the selected master device are connected to the Agilent E2926A/B testcard as described in “*Master Identification*” on page 29.

## Bus Utilization (Report Subsection 8.3)

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

8.3 Bus Utilization	
-----	
Time Overhead .....	20.00 %
Retry Overhead .....	0.00 %
Data Phases .....	55.00 %
	-----
Sum of Utilization .....	75.00 %

In detail, this list contains:

- Time Overhead

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

- Retry Overhead

The retry overhead is the sum of all transactions initiated by the master that were terminated by the target with a retry.

- Data Phases

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

- Sum of Utilization

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

Note, that these values are given in percent of the total test time. Hence, the sum of utilization here equals the bus occupation found in the “*Bus Usage (Report Subsection 8.2)*” on page 95. The remaining time the bus was either idle or the bus was occupied by other devices.

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 master and target, refer to the report subsections 8.3.1 to 8.3.6.



## Data Phase (Report Subsection 8.3.1)

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.

8.3.1 Data Phase	
-----	
0 Bytes .....	0.00 %
1 Byte .....	9.09 %
2 Bytes .....	18.18 %
3 Bytes .....	9.09 %
4 Bytes .....	63.63 %
-----	
Average Byte Enable Efficiency .....	81.81 %
Average Burst Length .....	18.33 data phase(s)

**Bytes per Data Phase** The byte enable bits on the PCI 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.

**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.

**On 64-Bit Systems** Within a **64-bit system**, the data phase analysis shows the byte enable usage for 8 byte enable bits.

```

8.3.1 Data Phase
-----
0 Bytes ..... 0.00 %
1 Bytes ..... 0.00 %
2 Bytes ..... 0.00 %
3 Bytes ..... 50.00 %
4 Bytes ..... 0.00 %
5 Bytes ..... 0.00 %
6 Bytes ..... 50.00 %
7 Bytes ..... 0.00 %
8 Bytes ..... 0.00 %
-----
Average Byte Enable Efficiency ..... 56.25 %
Average Burst Length ..... 2.00 data phase(s)

```

Again the total of the nine values equals 100 %, because all data phases are considered. In this 64-bit example, the average byte enable efficiency turns out as 56.25 % when sending either three or six bytes per data phase—both in equal amount. The average burst length of only two data phases per transaction is very low and indicates devices that probably cannot handle longer bursts.

**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 ..... 0.00 %
Master tried 64 but Target refused . 50.00 %
-----
32 Bit Data Transfers .(sum)..... 50.00 %
64 Bit Data Transfers ..... 50.00 %

```

The different cases distinguished here are:

- Pure 32-bit data transfers.

These are all transactions that were initiated by the master as 32-bit transactions.

- Master tried 64-bit but target refused.

This figure covers all transactions that were initiated by the master as 64-bit transactions, but were performed as 32-bit transfers due to the target'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.

## Time Overhead (Report Subsection 8.3.2)

This report section analyzes the time overhead in the recorded traffic. Every clock cycle in a transaction, that is not a data phase, is considered as overhead: address phases, waits, and latencies. The table reveals which device caused what type of overhead. Entries in the column *Both* indicate the cases where both master and target inserted wait states concurrently.

8.3.2 Time Overhead				
-----				
Average Decode Speed (1 == fast).... 2.12 cycle(s)				
	Overhead caused by			
	Master	Target	Both	Sum
	-----			
a) Address Phase	10.10 %	--	--	10.10 %
b) Waits	0.00 %	83.56 %	6.35 %	89.90 %
	-----			
1) First Word Latency (a+b)	10.10 %	83.56 %	6.35 %	100.00 %
2) Retry Transactions	--	0.00 %	--	0.00 %
3) Subsequent Latency	0.00 %	0.00 %	0.00 %	0.00 %
	-----			
Sum ((1) + (2) + (3))	10.10 %	83.56 %	6.35 %	100.00 %

**Average Decode Speed** The average decode speed in this example is 2.12 clock cycles. Good PCI device design includes targets that decode the address very quickly. However, the decode time cannot become shorter than one clock cycle.

**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 master devices.

- Waits

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

- First Word Latency

The first word latency is the time that the transactions need before the first set of data is transferred. It is calculated as the sum of address phases and initial wait states.

- Retry Transactions

This is the amount of overhead caused by the target due to retries.

- Subsequent Latency

Covered in this row are all delays that occurred in the transactions between the data phases.

- Sum ((1) + (2) + (3))

This row lists the total overhead caused by master and/or target. 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 (Report Subsection 8.6)” on page 111*.

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

## Command Usage (Report Subsection 8.3.3)

The Agilent E2926A/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 bus commands that have been used for the communication between the different devices.

This report subsection lists the distribution of the different PCI 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.

8.3.3 Command Usage					
Command	Fraction of used Clocks:		Fraction of Bytes moved	Fraction of used Clocks	Effi- ciency
	Overhead	Data			
SPECIAL	0.00 %	0.00 %	0.00 %	0.00 %	--
IO_READ	0.00 %	0.00 %	0.00 %	0.00 %	--
IO_WRITE	0.00 %	0.00 %	0.00 %	0.00 %	--
RESERVED_4	0.00 %	0.00 %	0.00 %	0.00 %	--
RESERVED_5	0.00 %	0.00 %	0.00 %	0.00 %	--
MEM_READ	0.00 %	0.00 %	0.00 %	0.00 %	--
MEM_WRITE	26.67 %	73.33 %	100.00 %	100.00 %	60.00 %
RESERVED_9	0.00 %	0.00 %	0.00 %	0.00 %	--
CONFIG_READ	0.00 %	0.00 %	0.00 %	0.00 %	--
CONFIG_WRITE	0.00 %	0.00 %	0.00 %	0.00 %	--
MEM_READMULT	0.00 %	0.00 %	0.00 %	0.00 %	--
MEM_READLINE	0.00 %	0.00 %	0.00 %	0.00 %	--
MEM_WRITEINV	0.00 %	0.00 %	0.00 %	0.00 %	--
Total:	100 %		100 %	100 %	

For each of the available PCI 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 clocks used by this command (i.e. the sum of the fractions for overhead and data).
- The overall efficiency of the transactions that were using this command.

When developing a PCI device, it is advisable to implement extended commands and 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 Usage* tab, the *Command Usage* chart presents the contents of this table in a graphical way. See also “*PCI Usage*” on page 44.

**On 64-bit Systems** Within a **64-bit system**, the table contains an additional column named *Fraction of 64 bit data* showing the percentage of the commands performed as 64-bit transactions.

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

## Command Termination (Report Subsection 8.3.4)

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

8.3.4 Command Termination				
Command	Fraction of Terminations			
	Retry	Disc. w/o data	Disc. w/ data	Normal
SPECIAL	0.00 %	0.00 %	0.00 %	0.00 %
IO_READ	0.00 %	0.00 %	0.00 %	18.62 %
IO_WRITE	0.00 %	0.00 %	0.00 %	28.28 %
RESERVED_4	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_5	0.00 %	0.00 %	0.00 %	0.00 %
MEM_READ	0.00 %	0.00 %	0.00 %	41.92 %
MEM_WRITE	0.00 %	0.00 %	0.01 %	7.02 %
RESERVED_8	0.00 %	0.00 %	0.00 %	0.00 %
RESERVED_9	0.00 %	0.00 %	0.00 %	0.00 %
CONFIG_READ	0.00 %	0.00 %	0.00 %	0.00 %
CONFIG_WRITE	0.00 %	0.00 %	0.00 %	0.00 %
MEM_READMULTI	0.00 %	0.00 %	0.00 %	0.00 %
MEM_READLINE	0.00 %	0.00 %	0.00 %	0.00 %
MEM_WRITEINVA	0.00 %	0.00 %	0.00 %	0.00 %
Total:	0.00 %	0.00 %	0.01 %	95.84 %

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

- 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.

- Disc. w/o data.

This type of termination only occurs during data phases. The transaction gets terminated without the data sent in this clock cycle being successfully transmitted to the recipient.

- Disc. w/ data.

This type of termination only occurs during data phases. The transaction gets terminated, but the data sent in this clock cycle is successfully transmitted to the recipient.

- Normal.

This column contains the fraction of data transfers that were finished as initially scheduled.

Note, that interrupt acknowledge cycles, master and target aborts are not covered in this table.

The last row of the table lists the sum of the respective columns. This means 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 8.5)” on page 109*.

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

## Wait Histogram (Report Subsection 8.3.5)

The wait histogram reveals the source for the sequences of wait states that occurred in the recorded data traffic. The table in this report subsection displays the wait sequences according to their length in clock cycles. The values in the table list the percentage of the total number of wait sequences, not wait states.

8.3.5 Wait Histogram		
-----		
Waits	Master	Target
-----		
0	50.00 %	50.00 %
1	0.00 %	0.00 %
2	0.00 %	0.00 %
3	0.00 %	0.00 %
4	0.00 %	0.00 %
5	0.00 %	0.00 %
6	0.00 %	0.00 %
7	0.00 %	0.00 %
8	0.00 %	0.00 %
9	0.00 %	0.00 %
10	0.00 %	0.00 %
11	0.00 %	0.00 %
12	0.00 %	0.00 %
13	0.00 %	0.00 %
14	0.00 %	0.00 %
15	0.00 %	0.00 %
16	0.00 %	0.00 %
17+	0.00 %	0.00 %

Every wait sequence that was inserted by one device—either master or target—is counted as a wait sequence of length zero for the other device. Therefore, all wait sequences appear in both columns. As a result the sum of both columns equals 50 %. The higher the value in the first row is, the less wait sequences this device actually has caused. In this example, none of the two devices did insert any wait states.



## Burst Length over Command (Report Subsection 8.3.6)

This report subsection explicitly lists every data transaction between the observed device pair according to its burst length and the used PCI bus command.

8.3.6 Burst Length over Command									
Brst Len	IO_REA D	IO_WRI TE	MEM_RE AD	MEM_WR ITE	CONFIG _READ	CONFIG _WRITE	MEM_RE ADMULT	MEM_RE ADLINE	MEM_WR ITEINV
0	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
1	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
2	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
3	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
4	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
5	0.00 %	0.00 %	0.00 %	33.33 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
... continued ...									
20	0.00 %	0.00 %	0.00 %	33.33 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
... continued ...									
30	0.00 %	0.00 %	0.00 %	33.33 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
... continued ...									
64	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
65	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
66+	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Sum	0.00 %	0.00 %	0.00 %	100.00 %	0.00 %	0.00 %	0.00 %	0.00 %	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 target's PCI chipset.

The figures in this table refer to the example transactions introduced in *“Example Transfers”* on page 82. From the last row of this table you can see that all transactions used the bus command Mem\_Read. One of the three transactions had the burst length 5, another length 20, and the third length 30.

**NOTE** This table displays the fraction of the total number of transactions. This is not equal to the fraction of used clock cycles as listed in *“Command Usage (Report Subsection 8.3.3)”* on page 101.

A graphical view of this table is displayed on the *Command* tab in the Performance Charts window. Refer to “*Command*” on page 46 for details.

## Efficiency Statistics (Report Subsection 8.4)

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

8.4 Efficiency Statistics	
-----	
PCI Efficiency of this pair .....	26.67 %
Non-retry Efficiency of this pair .....	26.67 %

The efficiency statistics contain

- PCI efficiency of this pair  
This is the overall PCI efficiency of the complete data traffic of the selected device pair considered in the test.
- Non-retry efficiency of this pair  
This is the PCI efficiency of the remaining transactions, when neglecting all transfers with a retry.

The difference of efficiency and non-retry efficiency is the retry overhead. If there is a remarkable difference between the two, the retry overhead should be analyzed in more detail to determine possible reasons. For this purpose see the “*Time Overhead (Report Subsection 8.3.2)*” on page 99.

In the example data traffic, no retries occurred. Therefore, both efficiency values are equal.

The values shown in this subsection of the report are also displayed on the *PCI Usage* tab of the Performance Charts window. A description to this tab is found in “*PCI Usage*” on page 44.

## Efficiency over Burst Length (Report Subsection 8.4.1)

This report subsection splits up the efficiency statistics for the different burst lengths separately. The table shows which burst lengths 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.

### 8.4.1 Efficiency over Burstlength

Burstlength	Efficiency	Data Fraction	Bus Fraction
0	--	0.00 %	0.00 %
1	--	0.00 %	0.00 %
2	--	0.00 %	0.00 %
3	--	0.00 %	0.00 %
4	--	0.00 %	0.00 %
5	16.67 %	9.09 %	20.00 %
... continued ...			
20	50.00 %	36.36 %	33.33 %
... continued ...			
30	85.71 %	54.54 %	46.67 %
... continued ...			
63	--	0.00 %	0.00 %
64	--	0.00 %	0.00 %
65	--	0.00 %	0.00 %
66+	--	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 burst length. If it does not, the bursts contain large overheads. In practice, 50 % is a good value.

The efficiencies of the three example transfers are:

- The first burst needed 35 clocks to transfer 120 bytes. Within this period 35 times 4 bytes = 140 bytes theoretically could be transferred. Thus, the efficiency is 85.7 % (120 bytes divided by 140 bytes). This transaction had 30 data phases, therefore, it is found in the row where burst length is 30.
- The second transaction—with burst length 20—needed 25 clocks to transfer 50 bytes. Thus, the efficiency is 50 %.
- The third transaction—with burst length 5—needed 15 clocks to transfer 10 bytes. Thus, the efficiency is 16.67 %.

**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 was used in a burst of a particular length.

The first of the example transfers used 30 cycles for data transfer (burst length = 30). The total number of data cycles in the test is 55. Thus, 30 out of 55 cycles were used in bursts of length 30. This yields a fraction of 55 %.

**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 first example transfer occupied the bus for 35 clock cycles. This is 46.67 % out of the total of 75 busy clock cycles.

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 45.

## Termination Statistics (Report Subsection 8.5)

The termination statistics display the average number of retries thrown by the target until the data could be transferred successfully.

### 8.5 Termination Statistics

-----  
Average Number of retries if retried ..... 16.20

This value shows the average number of times the target device stopped a transaction that was initiated by the master, before it finally was able to supply or receive data. Note, that only those data transfers are included, where at least one retry occurred.

For obvious reasons, the number of retries should be as small as possible for good performance.

To see an analysis showing with which PCI bus commands the retries occurred, refer to “*Command Termination (Report Subsection 8.3.4)*” on page 102. A table displaying the devices that terminated the bursts of different length can be found in “*Termination Burst Histogram (Report Subsection 8.5.1)*” on page 110.

**NOTE** The value in the table above was not taken from the report on the example transactions. In the example, no retries occurred. Thus, the average number of retries is zero as well.

## Termination Burst Histogram (Report Subsection 8.5.1)

The termination burst histogram lists the data transfers by their burst lengths and the devices that terminated them. Within this analysis it is not taken into account whether a transaction was finished successfully as initiated or whether it was terminated preliminary.

8.5.1 Termination Burst Histogram			
-----			
Burstlength	Transfer was stopped by ...		
	Master	Target	Arbiter
-----			
0	0.00 %	0.00 %	0.00 %
1	0.00 %	0.00 %	0.00 %
2	0.00 %	0.00 %	0.00 %
3	0.00 %	0.00 %	0.00 %
4	0.00 %	0.00 %	0.00 %
5	33.33 %	0.00 %	0.00 %
... continued ...			
20	33.33 %	0.00 %	0.00 %
... continued ...			
30	33.33 %	0.00 %	0.00 %
... continued ...			
64	0.00 %	0.00 %	0.00 %
65	0.00 %	0.00 %	0.00 %
66+	0.00 %	0.00 %	0.00 %
-----			
Sum	100.00 %	0.00 %	0.00 %

The values in this table show how many of all transfers terminated by a device were bursts of a particular length. Therefore, the sum in the three columns for master, target, and arbiter always is 100 %, or 0 % if this device did not terminate any transactions.

For good performance, preferably long bursts should be used that are then terminated by the master when reaching the initiated burst length. In the example all three transactions were completed successfully and terminated by the master. Thus, each of the three transactions made up one third (33.33 %) of all transfers that were terminated by the master.

**Terminating Devices** The three devices that can terminate a transaction are the master, the target, and the arbiter. The software assumes that the arbiter terminated a transfer if the REQ# signal still is asserted, but the GNT# signal is not asserted anymore. In all other cases the master is assumed to be the terminating device, unless it is a target abort, a disconnect, or a retry.

**NOTE** To observe the arbiter termination with the test software requires the REQ# lines and the GNT# lines to be connected to external trigger lines of the testcard as described in “*Master Identification*” on page 29.

Another representation of the transfer terminations is found in “*Command Termination (Report Subsection 8.3.4)*” on page 102. There the different termination reasons are displayed for the different PCI bus commands that were used in the transactions.

## Latency Histogram (Report Subsection 8.6)

The latency histogram in this subsection lists the total transaction lengths and all occurring latencies in the data traffic in detail. This enables you to determine how much delay in the data traffic was caused by which latencies and which device was responsible for it.

8.6 Latency Histogram						
	Full	First	Subseq.		Bus	First
	Transac.	Word	Word	Arbiter	Access	Word Lat.
Clocks	Clocks	Latency	Latency	Latency	Latency	w/ Retries
0	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
1	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
2	0.00 %	6.75 %	0.00 %	0.00 %	0.00 %	--
3	6.75 %	0.00 %	0.00 %	0.00 %	0.00 %	--
4	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
5	3.96 %	3.96 %	0.00 %	0.00 %	0.00 %	--
6	0.00 %	2.40 %	0.00 %	0.00 %	0.00 %	--
7	2.40 %	3.57 %	0.00 %	0.00 %	0.00 %	--
8	3.57 %	40.94 %	0.00 %	0.00 %	0.00 %	--
... continued ...						
61	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
62	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
63	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
64	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
65	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
66+	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	--
Sum	100.00 %					--

The different columns in the table are:

- Full Transac. Clocks

The full transaction clocks display the total lengths of the transactions in clock cycles. However, this value does not indicate whether long transactions were long bursts (good performance) or whether they contained a lot of overhead (poor performance).

A graphical view of this distribution is found on the *Full Trans Clocks* tab in the Performance Charts window. For a description to this view, refer to “*Full Transaction Clocks*” on page 48.

- 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 target and/or many initial waits, for example, due to locked target resources.

- Subseq. Word Latency

The subsequent word latency is the sum of all wait states that occurred between the first and the last data phase of a transaction. These wait states can be inserted both by the master and the target.

- Arbiter Latency

The arbiter latency is the time between a master asserting the REQ# signal and the arbiter giving the GNT# signal to this master.

A large number of clocks wasted for arbiter latency either indicates the bus being permanently busy (high bus utilization) or a slow arbiter with a poor arbitration algorithm.

- Bus Access Latency

This is the time between the arbiter asserting the GNT# signal to a master and this master driving FRAME# and the address onto the bus.

The bus access latency depends on the latency counters of the other PCI devices. A device may use the bus until its latency counter has expired—the other devices have to wait.

**NOTE** Both arbitration latency and bus access latency require the REQ# lines and the GNT# lines to be connected to the Agilent E2926A/B testcard as described in “*Master Identification*” on page 29.

- First Word Lat. w/ Retries

The first word latency with retries is the total number of clock cycles that the device pair occupied the bus from the first address phase until the first successful data transfer, including all failed attempts (retries).



The values of first word latency, subsequent word latency, arbiter latency, and bus access latency are displayed in a diagram on the *Latency* tab in the Performance Charts window. A description to this view is found in “*Latency*” on page 47.

**NOTE** The values listed in the table above were not taken from the test report of the example transactions. In the example no latencies occurred. Therefore, the latency histogram would only contain interesting data in the *Full Transac. Clocks* column.

## Top Ten List of First Word Latencies with Retries (Report Subsection 8.6.1)

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

8.6.1 Top 10 list of first word latencies with retries					
-----					
Nr	Command	Address	First Word Latency (clocks)	First Word Latency (sec)	Number of retries
-----					
10.	MEM_WRITE	00080010	36 clk	0.00000109 s	3
9.	--	--	--	--	--
8.	--	--	--	--	--
7.	--	--	--	--	--
6.	--	--	--	--	--
5.	--	--	--	--	--
4.	--	--	--	--	--
3.	--	--	--	--	--
2.	--	--	--	--	--
1.	--	--	--	--	--

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

- Command

This is the PCI bus command that was used by the master.

- Address

This is the target 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 (retries).

- Number of Retries

This is the number of times the target terminated the transaction with a retry until it finally was ready to transfer the data.

# Definitions of Used Measures

The definition of used measures is thought to describe a list of names, terms, and phrases that are used in the field of PCI data communication in general and in the Agilent E2926A/B Opt. 200 Performance Optimizer in particular. Many definitions are used commonly through literature, others may differ from other software or literature. 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.

- 64-Bit Data** 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. The 32-bit transfers are distinguished as:
- Pure 32-bit transfers that were initiated by the master instead of 64-bit transfers.
  - 64-bit transfers initiated by the master that were rejected by the target and then implemented as 32-bit transfers.

**Arbitration Latency** The arbitration latency is the time the arbiter needs to grant the bus to the requesting master, measured in clock cycles.

This value is displayed in a diagram on the *Latency* tab in the Performance Charts window. A description to this view is found in “*Latency*” on page 47.

Furthermore, this value is presented in the performance report. See the “*Latency Histogram (Report Subsection 8.6)*” on page 111 for details.

**Average Burst Length** This is the average burst length of the recorded transactions measured in clock cycles. This value include all successful data transfers as well as the transactions that were rejected by the target with a retry.

This value is presented in the performance report. See “*Data Phase (Report Subsection 8.3.1)*” on page 97 for details.

**Average Byte Enable Efficiency** The average byte enable efficiency is the average number of bytes transferred during one data phase. The value is given as the percentage of the maximum transferable during one clock cycle: 100 % means, that all data phases transported 4 bytes per data phase on a 32-bit system or 8 bytes on a 64-bit system.

This value is presented in the performance report for the selected master-target pair. See “*Data Phase (Report Subsection 8.3.1)*” on page 97 for details.

See also “*(PCI) Byte Enable Efficiency*” on page 118.

**Average Decode Speed** This is the speed with which the target decodes an address driven onto the bus by the master in the address phase of a transaction. It is measured in clock cycles:

- 1 = fast
- 2 = medium
- 3 = slow

This value is presented in the performance report. See “*Time Overhead (Report Subsection 8.3.2)*” on page 99 for details.

**Average Number of Retries if Retried** This is the average number of times the addressed target terminates an initiated transaction with a retry before it is ready to receive or supply the data. Note, that only those transfers are considered in this calculation, where at least one retry occurred.

This value is presented in the performance report. See the “*Termination Statistics (Report Subsection 8.5)*” on page 109 for details.

See also “*Non-Retry (PCI) Efficiency*” on page 121 and “*Non-Retry (PCI) Utilization*” on page 121.

**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 target address to start at. Thus, transactions with long bursts make more efficient use of the bus than with short bursts.

**Bus Access Latency** This is the time between the arbiter asserting the signal to a master and this master driving the FRAME# signal and the target address onto the bus.

Note, that this value is named bus acquisition latency in some literature, while bus access latency sometimes is defined differently.

This value is presented in the performance report. See the “*Latency Histogram (Report Subsection 8.6)*” on page 111 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 47.

**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. In this context the bus fraction gives the number of clock cycles, that were used for transactions with a particular burst length, out of the total number of clocks, during which the master-target pair occupied the bus.

This value is presented in the performance report. See “*Efficiency over Burst Length (Report Subsection 8.4.1)*” on page 107 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 45.

See also “*Data Fraction*” on page 118.

**Bus Users Overview** The bus users overview is an index evaluating the performance of a considered PCI device. It is derived from utilization and efficiency. The higher this value the worse is the device’s PCI design and the more it affects the whole system performance.

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

This value is presented in the performance report. See the “*Bus Users Overview (Report Section 6)*” on page 91 for details.

**(PCI) Byte Enable Efficiency** The byte enable efficiency is a measure indicating how well the maximum bandwidth of the PCI 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 it is eight. The byte enable efficiency for a 32-bit system is calculated as shown below. To calculate the value for a 64-bit system use eight bytes instead of four.

$$\text{Byte Enable Efficiency} = \frac{\text{Transferred Bytes}}{\text{Clocks Used for Data Phases} \times 4} \quad [\%]$$

The byte enable efficiency of the complete data traffic on the PCI bus is presented in the performance report. See the “*Efficiency Statistics (Report Section 4)*” on page 88 for details.

The byte enable efficiency of the selected master-target pair is found in “*Data Phase (Report Subsection 8.3.1)*” on page 97.

See also “*Average Byte Enable Efficiency*” on page 116, “*(PCI) Efficiency*” on page 119, and “*(PCI) Time Efficiency*” on page 123.

**Data 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. 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 master-target pair.

This value is presented in the performance report. See “*Efficiency over Burst Length (Report Subsection 8.4.1)*” on page 107 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 45.

See also “*Bus Fraction*” on page 117.

**Data Utilization** The data 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.

$$\text{Data Utilization} = \frac{\text{All Successful Data Phases}}{\text{Total Clocks}} \quad [\%]$$

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 90. The data utilization of the selected master-target pair is found in “*Bus Utilization (Report Subsection 8.3)*” on page 96.

The latter is also displayed in the *Bus Utilization* diagram on the *PCI Usage* tab of the Performance Charts window. For a description of this diagram, refer to “*PCI Usage*” on page 44.

**Disconnect with Data** During a transaction, the target can disconnect from the bus in several ways. In the case of *Disconnect with Data* (named *Disconnect-A* and *Disconnect-B* in the PCI specification) the data on the bus is transferred successfully in the last data cycle.

This value is presented in the performance report. Refer to “*Command Termination (Report Subsection 8.3.4)*” on page 102 for details.

**Disconnect without Data** During a transaction, the target can disconnect from the bus in several ways. In the case of *Disconnect without Data* (named *Disconnect-1* and *Disconnect-2* in the PCI specification) the data on the bus is not transferred successfully in the last data cycle.

This value is presented in the performance report. Refer to “*Command Termination (Report Subsection 8.3.4)*” on page 102 for details.

**(PCI) Efficiency** The PCI efficiency is a measure indicating 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 transferred bytes divided by the number of bytes that theoretically could be transferred within the used clock cycles.

In a 32-bit system the maximum number of bytes, that can be transferred per clock cycle is four. In a 64-bit system, it is eight. For a 32-bit system, the efficiency is defined as shown below. For a 64-bit system, divide by eight bytes instead of four.

$$\text{Efficiency} = \frac{\text{Throughput}}{\text{Utilization}} = \frac{\text{Transferred Bytes}}{\text{Used Clocks} \times 4 \text{ Bytes}} \\ = \text{Time Efficiency} \times \text{Byte Enable Efficiency} [\%]$$

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

As a result, an efficiency value near 100 % means that much data has been transferred while the bus was occupied to a minimum.

The efficiency is displayed in diagrams

- for the data traffic of the selected device pair in the *PCI Efficiency* diagram on the *PCI Usage* tab of the Performance Charts window. For a description of this view, refer to “*PCI Usage*” on page 44.

The respective section of the performance report is described in “*Efficiency Statistics (Report Subsection 8.4)*” on page 106.

- for the transactions with different burst length separately on the *Burst Usage* tab of the Performance Charts window. For a description, refer to “*Burst Usage*” on page 45.

The respective section of the performance report is described in “*Efficiency over Burst Length (Report Subsection 8.4.1)*” on page 107.

Also refer to “*(PCI) Byte Enable Efficiency*” on page 118 and “*(PCI) Time Efficiency*” on page 123.

**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 (Report Subsection 8.6)*” on page 111 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 47.

**First Word Latency with Retries** The first word latency with retries is the total number of clock cycles that the device pair occupied the bus from the first address phase until the first successful data transfer, including all failed attempts (retries).

This value is presented in the performance report. See “*Latency Histogram (Report Subsection 8.6)*” on page 111 for details.

**Full Transaction Clocks** The full transaction clocks represent the total length of a transaction measured in clock cycles. However, this value does not indicate whether long transactions were long bursts (good performance) or whether they contained a lot of overhead (poor performance).

This value is presented in the performance report. See “*Latency Histogram (Report Subsection 8.6)*” on page 111 for details.

A diagram displaying this value is found on the *Full Trans Clocks* tab in the Performance Charts window. For a description to this view, refer to “*Full Transaction Clocks*” on page 48.



<b>GNT# to Address Phase</b>	<p>This is the time the master needs from getting the GNT# signal from the arbiter until asserting the FRAME# signal, which introduces a new transaction with the address phase.</p> <p>This value is presented in the performance report. See <i>“Bus Usage (Report Subsection 8.2)” on page 95</i> for details.</p>
<b>Interrupt Latency</b>	<p>The interrupt latency is the time between an interrupt request and its acknowledgment. It is assumed that PCI interrupt acknowledge is not disabled by the BIOS. The interrupt latency is measured in clock cycles.</p> <p>This value is presented in the performance report. See <i>“Interrupt Latency (Report Section 7)” on page 92</i> for details.</p>
<b>Non-Retry (PCI) Efficiency</b>	<p>The non-retry PCI efficiency basically is calculated the same way as the PCI efficiency. The only difference is, that the non-retry PCI efficiency neglects all transactions, that were rejected by the target with a retry.</p> <p>This value is presented in the performance report. See <i>“Basic Bus Statistics (Report Section 2)” on page 86</i> for details.</p> <p>Also refer to <i>“(PCI) Efficiency” on page 119</i>.</p>
<b>Non-Retry (PCI) Time Efficiency</b>	<p>The non-retry PCI time efficiency basically is calculated the same way as the PCI time efficiency. The only difference is, that the non-retry PCI time efficiency neglects all transactions, that were rejected by the target with a retry.</p> <p>This value is presented in the performance report. See <i>“Efficiency Statistics (Report Section 4)” on page 88</i> for details.</p> <p>Also refer to <i>“(PCI) Time Efficiency” on page 123</i>.</p>
<b>Non-Retry (PCI) Utilization</b>	<p>The non-retry PCI utilization basically is calculated the same way as the PCI utilization. The only difference is, that the non-retry PCI utilization neglects all transactions, that were rejected by the target with a retry.</p> <p>This value is presented in the performance report. See <i>“Basic Bus Statistics (Report Section 2)” on page 86</i> for details.</p> <p>Also refer to <i>“(PCI) Utilization” on page 124</i>.</p>

**Overhead Utilization** The overhead utilization is the time the bus was occupied for transferring overhead data, i.e. all clock cycles except data phases. In this value only transactions are considered that were not terminated by a target retry.

$$\text{Overhead Utilization} = \frac{\text{All Non-Data Phases}}{\text{Total Clocks}} \quad [\%]$$

The overhead utilization of the complete bus traffic is presented in the performance report. See the “*Bus Utilization Statistics (Report Section 5)*” on page 90 for details.

For the observed master-target pair, this value is called time overhead. Refer to “*Time Overhead*” on page 123 for more information.

**Retry Overhead** The retry overhead covers all clock cycles that are part of transactions terminated by the target with a retry. This includes the address phase and all wait states until the retry is set.

$$\text{Retry Overhead} = \frac{\text{All Retry Phases}}{\text{Used Clocks}} \quad [\%]$$

Another way to calculate the retry overhead is the difference between the non-retry time efficiency and the time efficiency.

The retry overhead of the complete bus traffic is listed in the performance report. See the “*Bus Utilization Statistics (Report Section 5)*” on page 90 for details.

The retry overhead of the traffic of the selected master-target pair is displayed in “*Bus Utilization (Report Subsection 8.3)*” on page 96.

Also refer to “*Non-Retry (PCI) Efficiency*” on page 121 and “*Non-Retry (PCI) Utilization*” on page 121.

**Subsequent Word Latency** The subsequent word latency is the sum of all wait states that occur between the first and the last data phase of a transaction. These wait states can be inserted both by the master and the target.

This value is presented in the performance report. See “*Latency Histogram (Report Subsection 8.6)*” on page 111 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 47.

**(PCI) Throughput** The throughput is the amount of data transferred per time.

$$\text{Throughput} = \frac{\text{Transferred Data}}{\text{Time}} \text{ [MByte/s]}$$

This value is presented in the performance report. See “*Bus Throughput Statistics (Report Section 3)*” on page 87 for details.

If this value is given as a percentage, 100% 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 and 528 MByte/s in a 66 MHz/64-bit system.

**(PCI) 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.

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

This value only takes into account the occurrence of data phases. It is not considered whether the full bandwidth of the PCI 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 88 for details.

Also refer to “*(PCI) Byte Enable Efficiency*” on page 118 and “*(PCI) Efficiency*” on page 119.

**Time Overhead** All transaction phases except the data phases are regarded as overhead. The time overhead shows how much the bus was occupied by transferring overhead information (address and wait phases). However, transactions terminated with a retry are not included in this value.

The time overhead in the data traffic of the observed master-target pair is presented in the performance report. Refer to “*Bus Utilization (Report Subsection 8.3)*” on page 96 for a description of this report section.

A detailed analysis of the different clock cycles contributing to the overhead is found in “*Time Overhead (Report Subsection 8.3.2)*” on page 99.

For the complete data traffic observed on the bus, this value is referred to as the overhead utilization. See “*Overhead Utilization*” on page 122 for details.

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

$$\text{Utilization} = \frac{\text{Busy Clocks}}{\text{Total Clocks}} \quad [\%]$$

The PCI 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 90 and its subsection for more information.

**Waiting for GNT# with Bus Busy** This is the time the master waits for the GNT# signal from the arbiter after asserting a request (REQ#), while the bus is busy being used by another device.

This value is presented in the performance report. See “*Bus Usage (Report Subsection 8.2)*” on page 95 for details.

**Waiting for GNT# with Bus Idle** This is the time the master has to wait for the GNT# signal from the arbiter after asserting a request (REQ#), although the bus is idle.

This value is presented in the performance report. See “*Bus Usage (Report Subsection 8.2)*” on page 95 for details.

# Index

## #

64-bit Bus Statistics (report section 4.2) 89  
 64-bit Data 115  
     Command Usage Table 102

## A

Address  
     Ranges, Identification 30  
 Analysis Optimization 12  
 Analyzer (tab) 20, 32  
 Analyzer Overview (window) 14  
 Analyzing the Performance of the PCI Host Bridge 20  
 Arbiter Latency 75, 115  
 Average  
     Burst Length 97, 115  
     Byte Enable Efficiency 97, 116  
     Decode Speed 100, 116  
     Number of Retries 116

## B

Basic Bus Statistics (report section 2) 51, 86  
 Burst  
     Analysis 59  
 Burst Behavior 58  
 Burst Length 116  
     Average 115  
     Average Length 97  
     Efficiency (report) 107  
     over Command (report section 8.3.6) 59, 105  
 Burst Usage (tab) 26, 45  
 Bus Access Latency 75, 117  
 Bus Activity Lister 79  
     Export 68  
     Loading Files 70  
 Bus Fraction 108, 117  
 Bus Statistics (report sections 1 to 7) 83  
 Bus Throughput Statistics (report section 3) 51, 87  
 Bus Usage (report section 8.2) 95  
 Bus Users Overview (report section 6) 54, 91  
     Definition 117  
 Bus Utilization  
     Report Section 8.3 56, 96  
 Bus Utilization Statistics (report section 5) 52, 90  
 Byte Enable 58  
     Average Efficiency 97, 116

## C

Capture (tab) 24  
 Command  
     Tab 46  
 Command Line Interface (window) 23  
 Command Termination  
     Types 77  
 Command Termination (report section 8.3.4) 61, 102  
 Command Usage (report section 8.3.3) 101  
 Configuration Scan on the PCI Bus 23

## D

Data Capture 24  
     see also Capture (tab)  
     Setup 37  
 Data Fraction 108, 118  
 Data Phase (report section 8.3.1) 58, 97  
 Data Utilization 118  
 Decode Speed 100, 116  
 Disconnect With/Without Data 119

## E

Efficiency  
     Average Byte Enable Efficiency 116  
     over Burst Length (report section 8.4.1) 60, 107  
     see also PCI Time Efficiency  
     Statistics (report section 4) 53, 88  
     Statistics (report section 8.4) 106  
 Example Transfers 82  
 Existing Data, Re-Using 69  
 Existing Results, Re-Using 69  
 Exporting  
     Bus Activity Listing 68  
     Report 67  
     Test Settings 68  
     Trace Memory 67  
 Extended PCI Commands 15

## F

First Word Latency 76, 120  
     With Retries 113, 120  
 Full Transaction Clocks  
     Definition 120  
     Tab 48

## G

General Information  
     on the System under Test 50  
     Report Section 1 50, 84  
 GNT#  
     to Address Phase 121  
     Waiting with Bus Busy/Idle 124

## H

Hardware Setup 28

## I

I/O Commands 16  
 Identification  
     see Master, Target and Address Range Identification  
 Interpreting  
     Reports 48  
     Results 43  
 Interrupt Latency  
     Definition 121  
     Histogram (report section 7.1) 93  
 Interrupt Statistics (report section 7) 92

## L

Latency  
     Arbitration Latency 115  
     Definitions 75  
     Histogram (report section 8.6) 111  
     Interrupt Latency 121  
     Interrupt Latency Histogram (report) 93  
     Interrupt Statistics (report) 92  
     Subsequent Word Latency 122  
     Tab 47  
     Tests 29  
 Line Diagram 10  
 Loading  
     Bus Activity Lister Files 70  
     Report Files 70  
     Setup Files 20, 69  
     Test Results 26  
     Waveform Files 69  
 Loadmeter Display 10

## M

Master and Target Identification 22  
 Master Device 28  
 Master Identification 29  
     Setup 35  
     Tab 35  
 Master Target

- Pair 36
- Master-Target
  - Efficiency (report section 4.1) 89
  - Pair Measurements (report section 8) 93
- Measurement Setup 21
- Measures Available for Real-Time Measurements 19
- Memory Commands 16
- Methods of Performance Analysis
- Mode (window), Analyzer (tab) 20, 32
- Momentary (tab) 18

## N

---

- Navigation
  - in the Report 49
  - Main Window 13
- Non-Retry
  - PCI Efficiency 121
  - PCI Time Efficiency 121
  - PCI Utilization 121

## O

---

- Optimization
  - Ways of Analysis 12
- Overall System Performance 17
  - Hardware Setup 28
- Overhead Utilization 122
- Overview Window 14

## P

---

- Pair Select (tab) 36
- PCI Byte Enable Efficiency 118
- PCI Design
  - Process 8
  - Rules 15
- PCI Efficiency
  - Analysis 53
  - Definition 119
  - Non-Retry 121
- PCI Host Bridge, Analyzing Performance 20
- PCI Performance Optimizer
  - Enabling 32
  - General Description 7
  - Methods of Analysis 9
  - Running Measurements 42
  - Setting Up a Test 27
  - Steps of Performance Analysis 11
  - User Interface 13
- PCI Throughput 51, 123
- PCI Time Efficiency 123
- PCI Usage (tab) 25, 44
- PCI Utilization 52
  - Definition 124
  - Non-Retry 121
- Performance Button Group 13

- Performance Charts (window)
  - Burst Usage (tab) 26, 45
  - Charts and Report Sections 48
  - Command (tab) 46
  - Data Source 44
  - Full Transaction Clocks (tab) 48
  - Latency (tab) 47
  - PCI Usage (tab) 25, 44

- Performance of Particular Devices (Hardware Setup) 28

- Performance Setup (window)
  - Capture (tab) 24
  - Master Identification (tab) 35
  - Pair Select (tab) 36
  - Report (tab) 21, 33
  - Target Identification (tab) 22, 34

- Performance Windows 14

- Post-Processed Analysis
  - Advantages and Drawbacks 12
  - Definition 9
  - Description 10
  - Performance Tests 42
  - Trace Memory 11
  - Traffic Samples 10

- Printing Test Results 66

## R

---

- Real Time Counter Result (window) 18
  - Available Measures 19
  - Momentary (tab) 18
  - Time History (tab) 19

- Real-Time Measurement
  - Advantages and Drawbacks 12

- Real-Time Measurements
  - Definition 9
  - Description 10
  - Display Options 10
  - How to Run a Test 18
  - Measures Available 19
  - Performance Tests 42

- Repetitive Run and Upload (option) 39

- Report 48
  - Contents 49
  - Export 67
  - Interpreting 48
  - Loading Report Files 70
  - Navigation 49
  - Results Presented as Charts 48
  - Setup 33
  - Tab 21, 33
  - Using the Output for Performance Evaluation 50
  - Using the Output for Performance Optimization 55

- Rescan the Trace Memory 43

- Result Charts 43

- Results, Re-Using 69

- Retries
  - Average Number 116

- Retry Overhead 122

- Re-Using
  - Previously Saved Data 69
  - Test Setups and Results 65

- Rules for Proper PCI Design 15

## S

---

- Selecting the Appropriate Way of Analysis 12

- Setup
  - Data Capture 37
  - Files 20, 69
  - Master Identification 35
  - Measurements 21
  - Report 33
  - Re-Using Test Setups 65
  - Software 31
  - Target Identification 34
  - Test Hardware 28

- Show Analyzer Overview (menu item) 14

- Single Run and Upload from Trace Memory (option) 39

- Software Setup 31

- Statistical Base (report section 1.2) 85

- Statistical Basis (report section 8.1) 94

- Stop after ... Samples (option) 38

- Stop after Access to Address ... with .... (option) 38

- Stop at End of Trace Memory (option) 38

- Subsequent Word Latency 76, 122

- System Performance 17

- System under Test, General Information 50

## T

---

- Target
  - Address Range 23
  - Decode Speed 116
  - Device 28
  - Device Identification 28
  - Disconnect 78
  - Retry 77

- Target Identification 30
  - Setup 34
  - Tab 22, 34

- Terminating Devices 111

- Termination Burst Histogram (report section 8.5.1) 110

- Termination Statistics (report section 8.5) 62, 109

- Test Results 25
  - Loading the Example Results 26
  - Printing 66
  - Re-Use 65

- Test Settings, Export 68

- Time Efficiency
  - see PCI Time Efficiency

- Time History (tab) 19

Time Overhead  
  Definition 123  
  Report Section 8.3.2 56, 99  
Top Ten List of First Word Latencies with  
Retries (Report section 8.6.1) 113  
Trace Memory 11  
  Export 67  
  Rescan 43  
  Run Control Options 39  
Traffic  
  Overhead 56  
  Samples 10

---

**U**

---

Utilization  
  see PCI Utilization

---

**W**

---

Wait Histogram (report section 8.3.5) 57,  
104  
Waiting for GNT# with Bus Busy/Idle 124  
Waveform Files 69

