Verification Continuum™

VC Verification IP
ACE Performance Metrics Supported Through Verdi

Version S-2021.06, June 2021

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Preface

About This Document

This document provides information about the performance metrics of ACE supported with Verdi.

Web Resources

- Documentation through SolvNet: https://solvnetplus.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): http://www.synopsys.com/keys

Customer Support

To obtain support for your product, choose one of the following:

1. Go to https://solvnetplus.synopsys.com and open a case.

   Enter the information according to your environment and your issue.
2. Send an e-mail message to support_center@synopsys.com
   - Include the Product name, Sub Product name, and Product version for which you want to register the problem.
   - If applicable, provide the information noted in Appendix A, "Reporting Problems" on page 59.
3. Telephone your local support center.
   - North America:

     Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
   - All other countries:

     http://www.synopsys.com/Support/GlobalSupportCenters

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

Performance Metrics

The following is the list of ACE Performance Metrics and its description:

1 ACE Metrics Description

- `ace_trans_cleaninvalid_latency`: ace_trans_cleaninvalid_latency metric computes total time taken by the CLEANINVALID transaction to complete.
- `ace_trans_cleanshared_latency`: ace_trans_cleanshared_latency metric computes total time taken by the CLEANSHARED transaction to complete.
- `ace_trans_cleanunique_latency`: ace_trans_cleanunique_latency metric computes total time taken by the CLEANUNIQUE transaction to complete.
- `ace_trans_dvmcomplete_latency`: ace_trans_dvmcomplete_latency metric computes total time taken by the DVMCOMPLETE transaction to complete.
- `ace_trans_dvmmessage_latency`: ace_trans_dvmmessage_latency metric computes total time taken by the DVMMESSAGE transaction to complete.
- `ace_trans_evict_latency`: ace_trans_evict_latency metric computes total time taken by the EVICT transaction to complete.
- `ace_trans_makeinvalid_latency`: ace_trans_makeinvalid_latency metric computes total time taken by the MAKEINVALID transaction to complete.
- `ace_trans_makeunique_latency`: ace_trans_makeunique_latency metric computes total time taken by the MAKEUNIQUE transaction to complete.
- `ace_trans_readbarrier_latency`: ace_trans_readbarrier_latency metric computes total time taken by the READBARRIER transaction to complete.
- `ace_trans_read_byte_count`: ace_trans_read_byte_count metric computes total byte count of the READ type transaction. This metric is computed for every READ type transaction individually.
- `ace_trans_readclean_latency`: ace_trans_readclean_latency metric computes total time taken by the READCLEAN transaction to complete.
- `ace_trans_read_latency`: ace_trans_read_latency metric computes total time taken by all the read transactions to complete.
- `ace_trans_readnosnoop_latency`: ace_trans_readnosnoop_latency metric computes total time taken by the READNOSNOOP transaction to complete.
- `ace_trans_readnotshareddirty_latency`: ace_trans_readnotshareddirty_latency metric computes total time taken by the READNOTSHAREDDIRTY transaction to complete.
- `ace_trans_readonce_latency`: ace_trans_readonce_latency metric computes total time taken by the READONCE transaction to complete.

- `ace_trans_readshared_latency`: ace_trans_readshared_latency metric computes total time taken by the READSHARED transaction to complete.
- `ace_trans_readunique_latency`: ace_trans_readunique_latency metric computes total time taken by the READUNIQUE transaction to complete.
- `ace_trans_writeback_latency`: ace_trans_writeback_latency metric computes total time taken by the WRITEBACK transaction to complete.
- `ace_trans_writebarrier_latency`: ace_trans_writebarrier_latency metric computes total time taken by the WRITEBARRIER transaction to complete.
- `ace_trans_write_byte_count`: ace_trans_write_byte_count metric computes total byte count of the WRITE type transactions.
- `ace_trans_writeclean_latency`:ace_trans_writeclean_latency metric computes total time taken by the WRITECLEAN transaction to complete.
- `ace_trans_writeevict_latency`: ace_trans_writeevict_latency metric computes total time taken by the WRITEEVICT transaction to complete.
- `ace_trans_write_latency`: ace_trans_write_latency metric computes total time taken by all the write transactions to complete.
- `ace_trans_writelineunique_latency`: ace_trans_writelineunique_latency metric computes total time taken by the WRITELINEUNIQUE transaction to complete.
- `ace_trans_writenosnoop_latency`: ace_trans_writenosnoop_latency metric computes total time taken by the WRITENOSNOOP transaction to complete.
- `ace_trans_writeunique_latency`: ace_trans_writeunique_latency metric computes total time taken by the WRITEUNIQUE transaction to complete.
- `ace_ctrans_avg_read_latency`: ace_ctrans_avg_read_latency metric computes average latency of READ type transactions at a given port instance.
- `ace_ctrans_avg_write_latency`: ace_ctrans_avg_write_latency metric computes average latency of WRITE type transactions at a given port instance.
- `ace_ctrans_max_read_latency`: ace_ctrans_max_read_latency metric computes maximum latency from all READ type transactions at a given port instance.
- `ace_ctrans_max_write_latency`: ace_ctrans_max_write_latency metric computes maximum latency from all WRITE type transactions at a given port instance.
- `ace_ctrans_min_read_latency`: ace_ctrans_min_read_latency metric computes minimum latency from all READ type transactions at a given port instance.
- `ace_ctrans_min_write_latency`: ace_ctrans_min_write_latency metric computes minimum latency from all WRITE type transactions at a given port instance.
- `ace_ctrans_read_byte_count`: ace_ctrans_read_byte_count metric computes total byte count of READ type transactions at a given port instance. This metric displays total byte count of all READ type transaction at a given port instance and also byte count of each transaction at a given port instance.
- `ace_ctrans_read_outstanding_count`: ace_ctrans_read_outstanding_count metric computes total number of READ type request which did not complete [or outstanding] at any given point in time at a given port instance. Basically this metrics is used during interactive mode of debug to see how many transactions are outstanding.

- `ace_ctrans_read_request_count`: ace_ctrans_read_request_count metric computes total number of READ type requests received to a given port instance.
- `ace_ctrans_write_byte_count`: ace_ctrans_write_byte_count metric computes total byte count of all WRITE type transactions at a given port.
- `ace_ctrans_write_outstanding_count`: ace_ctrans_write_outstanding_count metric computes total number of WRITE type requests which did not complete [or outstanding] at any given point in time at a given port instance. Basically this metrics is used during interactive mode of debug to see how many transactions are outstanding.
- `ace_ctrans_write_request_count`: ace_ctrans_write_request_count metric computes total number of WRITE type requests received at a given port instance.
- `ace_cinst_read_bus_bandwidth_percentage`: ace_cinst_read_bus_bandwidth_percentage metric computes the percentage of READ type transactions' BANDWIDTH at a given instance from total READ type transactions' BANDWIDTH across all port instances. For ex: If the total READ type transactions' bandwidth is X, and at each instance total READ type transactions' bandwidths are Y1, Y2 and Y3 [in case of 3 masters configuration], then percentage of bandwidth at a given port is calculated as - { (Y1 * 100) / X }. Similarly for Y2 and Y3. This metric can be represented in the form of PIE chart.
- `ace_cinst_read_bus_bandwidth`: ace_cinst_read_bus_bandwidth metric computes total bandwidth of READ type transactions across all port instances. Basically it is computed as bytes per second taking ace_cinst_read_byte_count as the total byte count.
- `ace_cinst_read_byte_count_percentage`: ace_cinst_read_byte_count_percentage metric computes the percentage of READ type transactions' total byte_count at a given instance from total READ type transactions' byte_count across all port instances. For ex: If the total READ type transactions' byte_count is X, and at each instance total READ type transactions' byte_counts are Y1, Y2 and Y3 [in case of 3 masters configuration], then percentage of byte_count at a given port is calculated as - { (Y1 * 100) / X }. Similarly for Y2 and Y3. This metric can be represented in the form of PIE chart.
- `ace_cinst_read_byte_count`: ace_cinst_read_byte_count metric computes total byte count of READ type transactions across all port instances. Basically it is sum of all ace_ctrans_read_byte_count metrics at each port instance.
- `ace_cinst_read_request_count`: ace_cinst_read_request_count metric computes total number of READ type requests received across all port instances. This metrics is computed as sum of all ace_ctrans_read_request_count metrics.
- `ace_cinst_read_request_percentage`: ace_cinst_read_request_percentage metric computes the percentage of READ transaction requests at a given instance from total READ transaction requests across all port instances. For ex: If the total READ transaction requests is X, and at each instance total READ transaction requests is Y1, Y2 and Y3 [incase of 3 masters configuration], then percentage of READ transaction request at a given port is calculated as - { (Y1 * 100) / X }. Similarly for Y2 and Y3. This metric can be represented in the form of PIE chart.
- `ace_cinst_write_bus_bandwidth_percentage`: ace_cinst_write_bus_bandwidth_percentage metric computes the percentage of WRITE type transactions' BANDWIDTH at a given instance from total WRITE type transactions' BANDWIDTH across all port instances. For ex: If the total WRITE type transactions' bandwidth is X, and at each instance total WRITE type transactions' bandwidths are Y1, Y2 and Y3 [in case of 3 masters configuration], then percentage of bandwidth at a given port is calculated as - { (Y1 * 100) / X }. Similarly for Y2 and Y3. This metric can be represented in the form of PIE chart.

- `ace_cinst_write_bus_bandwidth`: ace_cinst_write_bus_bandwidth metric computes total bandwidth of WRITE type transactions across all port instances. Basically it is computed as bytes per second taking ace_cinst_write_byte_count as the total byte count.
- `ace_cinst_write_byte_count_percentage`: ace_cinst_write_byte_count_percentage metric computes the percentage of WRITE type transactions' byte_count at a given instance from total WRITE type transactions' byte_count across all port instances. For ex: If the total WRITE type transactions' bandwidth is X, and at each instance total WRITE type transactions' bandwidths are Y1, Y2 and Y3 [in case of 3 masters configuration], then percentage of bandwidth at a given port is calculated as - { (Y1 * 100) / X }. Similarly for Y2 and Y3. This metric can be represented in the form of PIE chart.
- `ace_cinst_write_byte_count`: ace_cinst_write_byte_count metric computes total byte count of WRITE type transactions across all port instances. Basically it is sum of all ace_ctrans_write_byte_count metrics at each port instance.
- `ace_cinst_write_request_count`: ace_cinst_write_request_count metric computes total number of WRITE type requests received across all port instances. This metrics is computed as sum of all ace_ctrans_write_request_count metrics.
- `ace_cinst_write_request_percentage`: ace_cinst_write_request_percentage metric computes the percentage of WRITE transaction requests at a given instance from total WRITE transaction requests across all port instances. For example, If the total WRITE transaction requests is X, and at each instance total READ transaction requests is Y1, Y2 and Y3 [in case of 3 masters configuration], then percentage of WRITE transaction request at a given port is calculated as - { (Y1 * 100) / X }. Similarly for Y2 and Y3. This metric can be represented in the form of PIE chart.