

AOCL Release Notes

Publication Number: 63855 Revision: 5.0

Date: October 2024

AMD

Contents

Chapter 1: Release Highlights....3

Chapter 2: AOCL-BLAS....4

Chapter 3: AOCL-Compression....6

Chapter 4: AOCL-Cryptography....7

Chapter 5: AOCL-Data Analytics....8

Chapter 6: AOCL-FFTW....9

Chapter 7: AOCL-LAPACK....10

Chapter 8: AOCL-LibM....11

Chapter 9: AOCL-LibMem....12

Chapter 10: AOCL-RNG....13

Chapter 11: AOCL-ScaLAPACK....14

Chapter 12: AOCL-Sparse....15

Chapter 13: AOCL-Utils....16

Appendix A: Additional Resources and Legal Notices....17

A.1 Revision History....17

Appendix B: Notices....18

B.1 Trademarks....18

Chapter 1: Release Highlights

This document lists the highlights and known issues of this release.

Chapter 2: AOCL-BLAS

- Turin specific tuning for the following APIs:
  - D/ZGEMM
  - DTRSM
  - DNRM2
- AVX512 improvements for the following APIs:
  - ZGEMV
  - D/ZAXPYF
  - D/ZDOTXF
  - ZDOTV
  - C/ZSCALV
  - DNRM2
  - S/D/ZCOPY
  - S/D/C/ZAXPBYV
  - DTRSV
  - DGEMMT
  - D/ZTRSM
  - D/ZGEMM
- LPGEMM improvements
  - New APIs
    - aocl_gemm_S8S8S32o<s32|S8>
    - aocl_gemm_s8s8s16o<s16|s8>
    - aocl_gemm_bf16s4f32o<f32|bf16>
  - Performance improvements
  - Transpose support for all AVX512 APIs
  - JIT based micro-kernel for bf16
  - Optimal LPGEMV kernels for m=1 or n=1
  - New post-ops
    - SWISH
    - mat_add
    - mat_mul
  - Added support for applying same post-op multiple times

  - aocl_gemm_eltwise bf16|f32 APIs to perform only element wise operations
  - Improvements to AOCL_ENABLE_INSTRUCTIONS functionality.
- Minor bug fixes

Chapter 3: AOCL-Compression

- Performance improvements:
  - ZLIB and BZIP2 single-threaded compression performance improvements.
  - LZ4, ZSTD, ZLIB, and Snappy multi-threaded performance improvements.
  - AOCL_ZLIB_QUICK_MODE added in ZLIB for higher compression speeds.
  - AOCL_DECOMPRESS_FAST mode added in LZ4 and ZSTD for higher decompression speeds.
  - LZ4 optimizations extended to LZ4_compress_*, and LZ4_decompress_safe_* APIs.
- Features:
  - LZ4HC multi-threaded support.
  - Support for asymmetric multi-threaded compression and decompression.
  - LZ4 baseline upgraded to version 1.9.4.
- Testing:
  - Improved test coverage through gtest, fuzzer tests, and third-party test benches.
- Documentation and usage guidelines:
  - Integrated Sphinx and Breathe documentation.
  - Examples added.
- Bug fixes:
  - Fixes to ensure reentrancy in single-threaded and multi-threaded builds.
  - Multiple bug fixes.

Chapter 4: AOCL-Cryptography

- OpenSSL provider support for Ciphers, CMAC, Poly1305, and RSA.
- Chacha20-Poly1305 AEAD Cipher.
- Bug fixes across all Cipher algorithms and performance improvements in AES CFB and CBC algorithms for vaes512 kernel.
- Cipher API Homogenization and C++ interfaces added.
- Padding support in AES CBC, CTR, OFB, and CFB algorithms.
- RSA Encrypt, Decrypt algorithms with PKCS padding.
- RSA Sign and verify algorithms with PSS and PKCS padding.
- Performance improvements in RSA OAEAP Encrypt and Decrypt algorithms.
- Bug fixes and performance improvements in HMAC, CMAC, and Poly1305.
  - SHA3 Shake Squeeze and Digest Context copy functions added.
  - Performance improvements in SHA2 and SHA3 algorithms.
- Bug fixes and improvements in dynamic dispatcher.
- Build system improvements in supporting documentation generation, mem-check tools, code coverage tools, ninja build system, and compilers till GCC13.1.0 and AOCC 5.0.
- Test and benchmarking framework updates to support fuzz testing, misaligned input tests, lifecycle tests, and refactoring and clean up in all modules.
- Memory errors fixed across various algorithms.

Known issues

- IPP Compat library is in experimental state in the 5.0 release.
- RSA and EC algorithms are unsupported in pre-AVX2 architectures.
- Behavior might be undefined if AVX512 is disabled in the BIOS configuration on the Zen5 platform.

Chapter 5: AOCL-Data Analytics

AOCL-Data Analytics is a new data analytics library providing optimized building blocks for data analysis and classical machine learning.

- Key functionality available in AOCL-Data Analytics:
  - Loading data from memory by reading CSV files.
  - Preprocessing data by removing missing values, standardizing, and selecting subsets of data.
  - Processing data (using, for example, principal component analysis, linear models, etc.)
- AOCL-Data Analytics can be called in the following ways using:
  - The C-compatible interface to enable seamless integration with multiple programming languages.
  - C++ overloads for increased data abstraction.
  - Python API.
  - scikit-learn patch to enable existing scikit-learn users to benefit from AOCL-Data Analytics' performance with minimal code changes.
- Data processing algorithms available with 5.0:
  - Linear models including logistic regression, lasso, and ridge regression.
  - K-Means Clustering.
  - Principal component analysis and singular value decomposition.
  - K-Nearest Neighbors.
  - Decision Trees and Random Forests.
  - Nonlinear Least Squares Fitting.
  - Basic statistics.

Chapter 6: AOCL-FFTW

- Support for linking user defined library path for OpenMP at compile time in Windows.
- Refined and efficient wisdom support added for AMD Application Layer that can be enabled with `--enable-amd-app-opt`.
- Minor bug fix for clang compilation error when using the config option `--enable-threads` with dynamic dispatcher.

Chapter 7: AOCL-LAPACK

- Improved performance of the following APIs through AVX2 & AVX512 SIMD instructions:
  - Double Precision SVD (DGESVD).
  - LU Factorization/Solver routines for general matrices (DGETRF, ZGETRF, DGETRS, and DGESV).
  - Matrix inverse routine DGETRI for small sizes.
  - Least Square solver DGELS for small sizes.
  - Double Precision Auxiliary routine, DLARFG.
- Improved performance of the following APIs using local AOCL-BLAS optimized kernels:
  - LU Factorization/Solver routines for band storage matrices (DGBTRF, DGBTRS).
- Option to set specific ISA code path at runtime through the `AOCL_ENABLE_INSTRUCTIONS` environment variable.
- Sphinx-based AOCL-LAPACK API documentation.
- LAPACK API modifications:
  - Updated AOCL-LAPACK APIs return type to match with the corresponding netlib subroutine prototypes.
  - Removed xerbla and lsame definition from AOCL-LAPACK. Applications have to invoke these functions from BLAS library.
- pkgconfig is now supported on Linux with cmake builds.
- Test suite framework enhancements:
  - Improvements to accuracy tests including testing with different input generation mechanisms.
  - Addition of extreme values, negative, and corner test cases.
  - Addition of cases to test numerical stability.
  - Support for LAPACKE interface test.
- Minor Bug fixes and code quality improvement changes.

Note:

1. AOCL-Utils library has libstdc++ library dependency. As AOCL-LAPACK is dependent on AOCL-Utils, applications running on Linux must additionally link with libstdc++(-lstdc++).
2. The pre-built AOCL-LAPACK binaries are compiled with the `ENABLE_AOCL_BLAS` option. Hence, applications must link with AOCL-BLAS when using the prebuilt AOCL-LAPACK library.

Chapter 8: AOCL-LibM

- New functions added:
  - vrd2_powx, vrd4_powx, vrd8_powx, vrs4_powxf, vrs8_powxf, and vrs16_powxf.
  - vrd2_linearfrac, vrd4_linearfrac, vrd8_linearfrac, vrda_linearfrac, vrs4_linearfracf, vrs8_linearfracf, vrs16_linearfracf, and vrsa_linearfracf.
- Vector functions added:
  - vrd2_sqrt, vrd4_sqrt, vrd8_sqrt, vrs4_sqrtf, vrs8_sqrtf, and vrs16_sqrtf.
  - vrd2_fabs, vrd4_fabs, vrd8_fabs, vrs4_fabsf, vrs8_fabsf, and vrs16_fabsf.
  - vrs4_sincosf, vrs8_sincosf, and vrs16_sincosf.
- Vector array functions added:
  - vrda_pow and vrsa_powf.
  - vrda_log, vrs4_logf, vrda_log2, vrsa_log2f, vrda_log10, vrsa_log10f, vrda_log1p, and vrsa_log1pf.
  - vrda_exp, vrsa_expf, vrda_exp2, vrsa_exp2f, vrda_exp10, vrsa_exp10f, vrda_expm1, and vrsa_expm1f.
  - vrda_sin, vrsa_sinf, vrda_cos, and vrsa_cosf.
  - vrda_cbrt, vrsa_cbrtf, vrda_sqrt, vrsa_sqrtf, vrda_fabs, and vrsa_fabsf.
- Optimized the performance of:
  - vrd2_pow and vrd4_pow vrd2_exp, vrd4_exp and vrs8_expf.
  - vrd2_log, vrd4_log, and vrs8_logf.
  - vrd2_cos, vrd4_cos, and vrs8_cosf.
  - vrd2_sin and vrd4_sin.
- Improved accuracy of vrd4_pow.
- Introduced a CMake build system for Linux for building AOCL-LibM. See the AOCL 5.0 User guide for more details and usage instructions. The Scons based build system is maintained as before; will be deprecated in the near future in favor of CMake.
- From release 5.0 onwards, AOCL-LibM links to au_cpuid binaries of AOCL-Utils. More details provided in the AOCL 5.0 User guide.

Known Issues

- Behavior might be undefined if AVX512 is disabled in the BIOS configuration on the Zen5 platform.

Chapter 9: AOCL-LibMem

- Support added for new functions: memchr, strcat, and strstr.
- Improved avx2 memory functions performance for smaller sizes.
- Minor bug fixes.

Note: Different binaries are available for AVX-512 and AVX2. For more information, see the AOCL-LibMem webpage.

Known Issues

- Behavior might be undefined if AVX512 is disabled in the BIOS configuration on the Zen5 platform.

Chapter 10: AOCL-RNG

- Added AVX2 based optimized kernel for double precision MRG32K3A base generator.
- Added Sobol quasi-random number generator as new base generator.
- Support the`AOCL_ENABLE_INSTRUCTIONS` environment variable to select specified ISA kernels.
- Added integer arithmetic based native implementation for double precision MRG32K3A base generator.
- Minor bug fixes and improved documentation.

Known Issues

- MRG32K3A generator tests may fail for distributions other than uniform and Gaussian for specific sizes.
- Behavior might be undefined if AVX512 is disabled in the BIOS configuration on the Zen5 platform.

Chapter 11: AOCL-ScaLAPACK

- Ability to build ScaLAPACK library without building test framework.
- Enabled Tracing and Logging feature for all APIs including single precision.
- Test suite framework enhancements:
  - Improvements to accuracy tests including testing with different input generation mechanisms.
  - Addition of extreme values, negative, and corner test cases.
- Minor bug fixes and code quality improvement changes.

Chapter 12: AOCL-Sparse

- Incorporated new APIs:
  - Level 3: sp2md, spmmd, syrk, syrkd, and sypr.
  - Sparse preconditioner: sorv.
- Support for:
  - Symmetric and Hermitian matrices in csrmm API.
  - Strided dense vector in TRSM and TRSV.
  - Symmetric Gauss Seidel.
  - Customized SpMV and TRSV APIs for HPCG.
- Performance improvements:
  - Level 1: dot, gthr, sctr, roti.
  - Level 2: SpMV for complex general matrices.
- Multi-threading support:
  - Level 3: Sp2M, SpMM, and SpAdd.
- Benchmarking framework improvements:
  - Enhanced statistics
  - Support for new APIs
  - Random matrix generation capability (Hermitian, Diagonally Dominant)
- Framework improvements.

Chapter 13: AOCL-Utils

- New CPUID APIs to identify CPU architecture and CPU features.
- New thread pinning APIs.
- Deprecated old APIs.
- Option to enable and disable building static/shared libraries.

Appendix A: Additional Resources and Legal Notices

A.1 Revision History

Date	Revision	Description
October 2024	5.0	Created this document for the 5.0 release.

Appendix B: Notices

© Copyright 2022-2024 Advanced Micro Devices, Inc.

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED "AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

B.1 Trademarks

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.