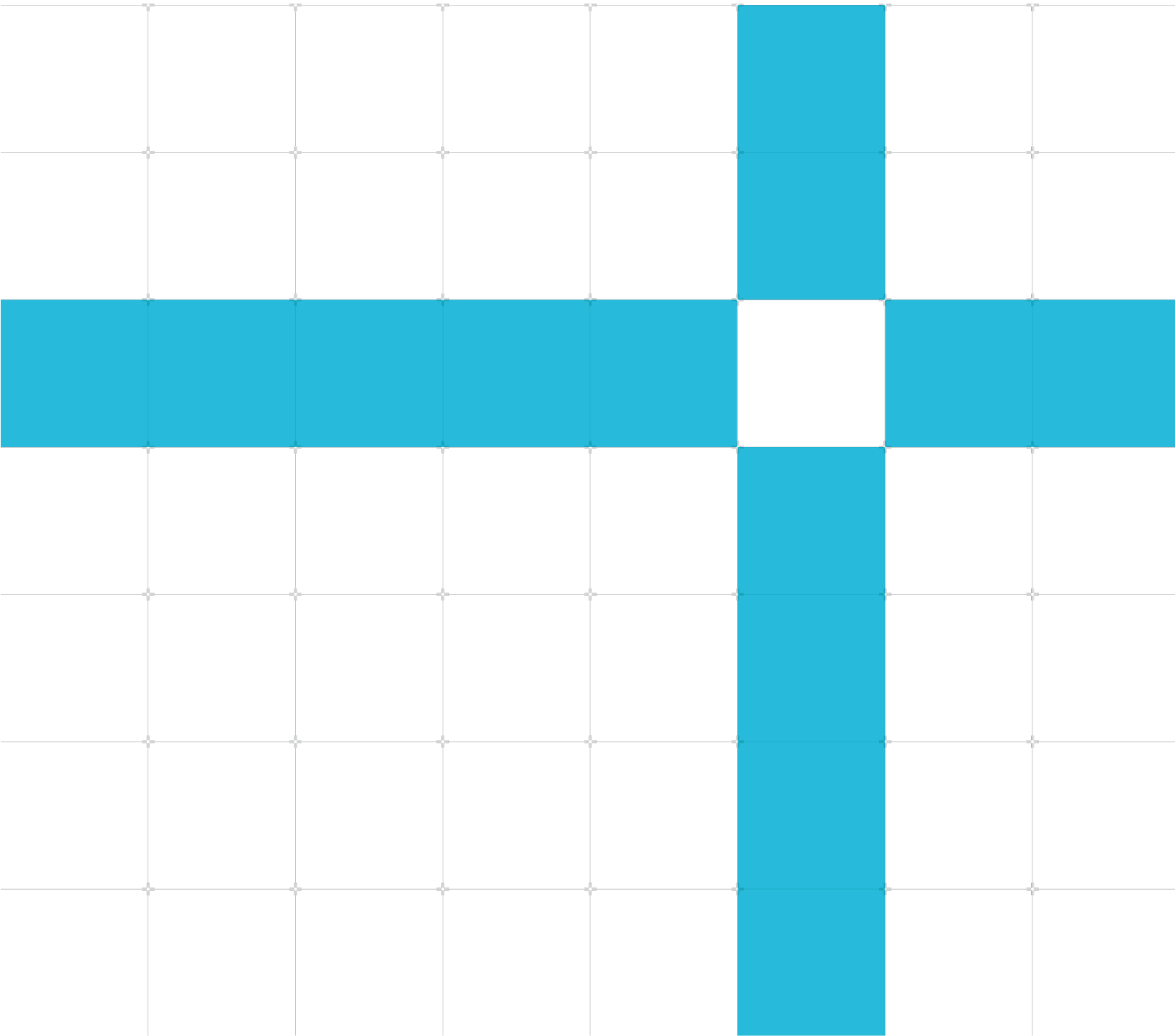
Arm RAN Acceleration Library

Product revision: 22.04

Release Note

Non-Confidential

Document ID:
107561



Arm RAN Acceleration Library

Release Note

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
(LES-PRE-20349)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm RAN Acceleration Library, create a ticket on **https://support.developer.arm.com**.

To provide feedback on the document, fill the following survey:
https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

To report offensive language in this document, email **terms@arm.com**.

Contents

1 Release overview

The following sections describe the product that this release note describes and its quality status at time of release.

Use of Arm RAN Acceleration Library is subject to the terms and conditions of the applicable End User License Agreement (“EULA”). A copy of the EULA can be found in the `license_terms` folder of your product installation.

You do not require a license to use this Arm RAN Acceleration Library package.

1.1 Product description

The Arm RAN Acceleration Library (ArmRAL) contains a set of functions for accelerating telecommunications applications such as, but not limited to, 5G Radio Access Networks (RANs).

The Arm RAN Acceleration Library 22.04 package provides a library that is optimized for Arm AArch64-based processors, and also provides optimized routines for processors that support the SVE and SVE2 architecture extensions.

Arm RAN Acceleration Library provides:

- Vector functions

- Matrix functions

- Lower PHY support functions

- Upper PHY support functions

- DU-RU Interface support functions

Arm RAN Acceleration Library includes functions that operate on 16-bit signed integers and 32-bit floating-point values.

1.2 Release status

This is the 22.04 release of Arm RAN Acceleration Library.

These deliverables are being released under the terms of the agreement between Arm and each licensee (the "Agreement"). All planned verification and validation is complete.

2 Release contents

Arm releases can contain documentation and source files such as RTL, testbenches, or software.

The following sections describe:

- Downloading and unpacking the product.
- The contents of this release.
- Any changes since the previous release.
- Any known issues and limitations that exist at the time of this release.

2.1 Downloading and unpacking

You must download the Arm RAN Acceleration Library deliverable from the Arm Developer website, then unpack the contents.

Procedure

1. Go to https://developer.arm.com/solutions/infrastructure/developer-resources/5g/ran/download.
2. Complete the form and click Submit. The package downloads.
3. Locate the downloaded .tar.gz file.
4. Copy the .tar.gz file to the directory where these files are to be built.
5. Extract the tar file contents using a tar utility, type:
   ```
   tar -zxvf arm-ran-acceleration-library-22.04-aarch64.tar.gz
   ```

2.2 Deliverables

The downloaded product includes the deliverables listed in this section.

- Arm RAN Acceleration Library 22.04

- Release Notes (this document)

- Documentation

Product documentation is available on the Arm Developer website at:

https://developer.arm.com/documentation/102249/2204

Documentation, errata and release notes might change between product releases. For the latest documentation bundle, check the product download page.

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of this document when used with any other PDF reader.
A suitable PDF reader can be downloaded from Adobe at **http://www.adobe.com**.

2.3 Differences from previous release

The following subsections describe differences from the previous release of Arm RAN Acceleration Library.

2.3.1 Additions and changes

Describes new features or components added, or any technical changes to features or components in this release.

- Improved the performance of the following implementations:

- Batched 16-bit floating-point matrix-vector multiplication with 64-bit floating-point accumulator (`armral_cmplx_mat_vec_mult_batch_i16`, `armral_cmplx_mat_vec_mult_batch_i16_pa`). The product of the number of matrices and the number of vectors per matrix needs to be a multiple of 12. The number of vectors per matrix must be either one or a multiple of four.
- Batched 16-bit floating-point matrix-vector multiplication with 32-bit floating-point accumulator (`armral_cmplx_mat_vec_mult_batch_i16_32bit`, `armral_cmplx_mat_vec_mult_batch_i16_32bit_pa`). The product of the number of matrices and the number of vectors per matrix needs to be a multiple of 12. The number of vectors per matrix must be either one or a multiple of four.
- 14-bit block float compression (`armral_block_float_compr_14bit`).
- 14-bit block scaling compression (`armral_block_scaling_compr_14bit`).
- 14-bit mu law compression (`armral_mu_law_compr_14bit`).
- Complex 32-bit floating-point singular value decomposition (`armral_svd_cf32`). The input matrix now needs to be stored in column-major order. Output matrices are also returned in column-major order.

- Complex 32-bit floating-point Hermitian matrix inversion for a single matrix or a batch of matrices of size `3x3` (`armral_cmplx_hermitian_mat_inverse_f32`, `armral_cmplx_hermitian_mat_inverse_batch_f32`).
- Polar list decoding (`armral_polar_decoder`) with size of list equal to four. The performance for list size equal to one is slightly reduced, but the list size of four gives much better error correction, and we focus on that.

- Added SVE2 optimized implementations of the following routines:

- Batched 16-bit fixed-point matrix-vector multiplication with 64-bit and 32-bit fixed-point accumulator (`armral_cmplx_mat_vec_mult_batch_i16`, `armral_cmplx_mat_vec_mult_batch_i16_pa`, `armral_cmplx_mat_vec_mult_batch_i16_32bit`, `armral_cmplx_mat_vec_mult_batch_i16_32bit_pa`).
- Complex 32-bit floating-point singular value decomposition (`armral_svd_cf32`).
- Complex 32-bit floating-point Hermitian matrix inversion for a single matrix or a batch of matrices of size `3x3` (`armral_cmplx_hermitian_mat_inverse_f32`, `armral_cmplx_hermitian_mat_inverse_batch_f32`).
- 9-bit and 14-bit mu law compression (`armral_mu_law_compr_9bit`, `armral_mu_law_compr_14bit`).
- 9-bit and 14-bit mu law decompression (`armral_mu_law_decompr_9bit`, `armral_mu_law_decompr_14bit`).

- Added complex 32-bit floating-point general matrix inversion for matrices of size `2x2`, `3x3`, `4x4`, `8x8`, and `16x16` (`armral_cmplx_mat_inverse_f32`). No specific optimizations have been implemented in this release.

- The functions to perform batched matrix-vector multiplications in fixed-point precision (`armral_cmplx_mat_vec_mult_batch_i16`, `armral_cmplx_mat_vec_mult_batch_i16_pa`, `armral_cmplx_mat_vec_mult_batch_i16_32bit`, `armral_cmplx_mat_vec_mult_batch_i16_32bit_pa`) have restrictions added on the number of matrices and vectors in the batch. This is to facilitate optimizations in implementation. The following two conditions need to be met:
    - The product of the number of matrices and number of vectors per matrix must be a multiple of 12.
    - The number of vectors per matrix must either be one or a multiple of four.

- The function to perform fixed-point complex matrix-matrix multiplication with a 64-bit accumulator (`armral_cmplx_mat_mult_i16`) narrows from the 64-bit accumulator to a 32-bit intermediate value, and then to the 16-bit result using truncating narrowing operations instead of rounding operations. This matches the behavior in the fixed-point complex matrix-matrix multiplication with a 32-bit accumulator. It is assumed that the loss in half a bit of precision with each truncation is not important for correct results.

- The function to perform fixed-point complex matrix-vector multiplication with a 64-bit accumulator (`armral_cmplx_mat_vec_mult_i16`) narrows from the 64-bit accumulator to a 32-bit intermediate value, and then to the 16-bit result using truncating narrowing operations instead of rounding operations. This matches the behavior in the fixed-point complex matrix-vector multiplication with a 32-bit accumulator. It is assumed that the loss in half a bit of precision with each truncation is not required for correct results.

2.3.2 Resolved issues

There are no resolved issues in this release.

2.4 Known limitations

There are no known issues with this release.

3 Support

If you have any issues with the installation, content or use of this release, create a ticket on **https://support.developer.arm.com**. Arm will respond as soon as possible.

Support for this release of the product is only provided by Arm to partners who have a current support and maintenance contract for the product.

3.1 Tools

The following points list the tools that are required to build or run Arm RAN Acceleration Library:

- A recent version of a C/C++ compiler, such as GCC. Arm RAN Acceleration Library has been tested with GCC 7.5.0, 8.2.0, 9.3.0, 10.2.0, and 11.1.0.

If you are cross-compiling, you need a cross-toolchain compiler that targets AArch64. You can download open-source cross-toolchain builds of the GCC compiler on the Arm Developer website:
https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-a/downloads
The variant to use for an AArch64 GNU/Linux target is `aarch64-none-linux-gnu`.

- A recent version of CMake (version 3.3.0, or higher).

In addition to the preceding requirements:

- To run the benchmarks, you must have the Linux utility tool `perf` installed and a recent version of Python 3. Arm RAN Acceleration Library has been tested with Python 3.8.5.
- To build a local version of the documentation, you must have Doxygen installed. Arm RAN Acceleration Library has been tested with Doxygen version 1.8.13.
- To generate code coverage HTML pages, you must have `gcovr` installed. The library has been tested with `gcovr` version 4.2.

Arm RAN Acceleration Library runs on AArch64 cores, however to use the CRC functions, you must run on a core that supports the AArch64 PMULL extension. If your machine supports the PMULL extension, pmull is listed under the "Features" list given in the /proc/cpuinfo file.

4 Release history

A full release history (with release notes) for Arm RAN Acceleration Library is available on the Arm Developer website:

https://developer.arm.com/solutions/infrastructure/developer-resources/5g/ran/release-history

5 Conventions

The following subsections describe conventions used in Arm documents.

5.1 Glossary

The Arm Glossary is a list of terms that are used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: **https://developer.arm.com/glossary**.