SIEMENS EDA

Algorithmic C (AC) Math Library **Release Notes**

Software Version v3.4.5
November 2022

SIEMENS

Copyright 2018 Siemens

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.

Table of Contents

Release 3.4.5

The following topics describes the changes that were made to the *ac_math* library since the last release. This release provides enhancements and/or bug fixes.

Enhancements

Support for ac_std_float was added to the ac_tanh_pwl and ac_sigmoid_pwl functions.

Corrected Issues

- **CAT-30998** – Fixing ac_hcordic.h issues.
  Added usage examples for all cordic functions in ac_hcordic.h, not just ac_log2_cordic.
  Added static_asserts to allow for easier debugging.
  Removed limitations on input integer width for ac_exp_cordic and ac_exp2_cordic.
  Added template parameters to let users override bitwidths of temporary variables.

Release 3.4.3

The following topics describes the changes that were made to the *ac_math* library since the last release. This release provides enhancements.

Enhancements

- CAT-29881- Support for Microsoft Visual Studio C++ 2019
  The AC Math headers have been updated to compile correctly with MS Visual Studio 2019. Note that although compilation and execution is now possible on the Windows platform, Catapult HLS is only available on Linux.

Release 3.4.2

The following topics describes the changes that were made to the *ac_math* library since the last release. This release provides enhancements and bug fixes.

Enhancements

(CAT-29828, CAT-28828): Files: ac_pow_pwl.h, ac_hcordic.h – Added floating point support.

Corrected Issues

- CAT-29549
  Changed ac_cholinv toolkit and mathlib regressions.
  - gen_matrix function always produces a positive definite matrix.
  - Input precision depends partially on M.
  - The parameter N is no longer used.
  - Regression test example can now avoid PWL error checking through the use of a boolean flag.
  ac_cholinv_tb.cpp and rtest_ac_cholinv_tb.cpp:
  - ac_random library is not included, instead we include the random and limits headers provided by C++11, so as to use the uniform_real_distribution function to generate a randomized positive definite matrix.
  - Input and output typenames now displayed.
  ac_cholinv_tb.cpp only:
  - Removed some debug statements, added #ifdef DEBUG guards around others to reduce noise.
- ac_inverse_sqrt_pwl_vhd_lutgen.cpp
  - LUT file now doesn't take into account input signedness, because inputs to the inverse square root header are always signed.
- ac_sqrt.h
  - Added special input (NaN and -0.0) handling to ac_sqrt.h. Modified both the ac_sqrt unit and regression tests to check for the same.
  - Odd exponent handling for the ac_float ac_sqrt() implementation changed. Instead of multiplying sqrt_mant by root2, we now left-shift the input for odd expoents.
  - rtest_ac_sqrt.cpp - Increased precision of some of the unit tests.

Release 3.4.1

The following topics describes the changes that were made to the *ac_math* library since the last release. This release provides new functionality, enhancements and bug fixes.

Enhancements

N/A

Corrected Issues

Corrected the banner text for each file.

Release 3.2.3

The following topics describes the changes that were made to the *ac_math* library since the last release. This release provides new functionality, enhancements and bug fixes. This version of ac_math was included in Catapult releases 10.4a, 10.4b, 10.4c, 10.5 and 10.5a.

Removed Using-Declarations

Using-declarations to the std namespace for some of the ac_math header files were removed. This was done to avoid any unwanted pulling of namespaces into any implementation files where the headers might be included, and thereby avoid any ensuing ambiguity. The ac_math header files where this was applicable were those that used cout statements to display debug information. The following files now use the scope resolution operator to access members of the std namespace:

- ac_atan_pwl.h
- ac_chol_d.h
- ac_determinant.h
- ac_hcordic.h
- ac_inverse_sqrt_pwl.h
- ac_log_pwl.h
- ac_normalize.h
- ac_pow_pwl.h
- ac_qrd.h
- ac_reciprocal_pwl.h
- ac_sigmoid_pwl.h
- ac_sincos_lut.h
- ac_sqrt_pwl.h
- ac_tan_pwl.h
- ac_tanh_pwl.h

Added High-Accuracy Files

Associated Jira ticket: CAT-24022.
High-accuracy versions of the ac_atan_pwl and ac_reciprocal_pwl header files were added to facilitate high-accuracy calculation of the arctangent function. The new versions use more fractional bits and PWL segments. The following are the names of the files added:

- ac_atan_pwl_ha.h
- ac_reciprocal_pwl_ha.h

Corrected Issues

The following issues were fixed in this release:

- **(CAT-24269, CAT-24362)**: File: ac_qrd.h – QR decomposition calculations for complex inputs were rectified to give the correct output.

- **(no bug #)**: File: ac_sqrt.h – An extra set of parentheses were added to remove compile-time warnings and reduce ambiguity.

Release 3.1.2

The following topics describes the changes that were made to the *ac_math* library since the last release. This release provides new functionality and bug fixes. This version of ac_math was included in Catapult release 10.3a.

Improved Bitwidth Calculation

The following PWL functions had the calculations for parameterized bitwidths of ac_fixed variables improved in order to eliminate redundant bits and reduce area:

- ac_inverse_sqrt_pwl.h
- ac_log_pwl.h
- ac_pow_pwl.h
- ac_reciprocal_pwl.h
- ac_sqrt_pwl.h
- ac_tan_pwl.h

Cleanup Sine/Cosine Cordic

The ac_sincos_cordic.h file was updated to rename a typedef so as to avoid a redeclaration conflict.

Removed Direct Access to AC Float Member Data

The ac_reciprocal_pwl(), ac_inverse_sqrt_pwl() and ac_sqrt_pwl() functions now no longer directly access the mantissa and exponent data members in the ac_float class for inputs. Instead, the functions use the .mantissa() and .exp() member functions to indirectly access (read-only) the data members of the input floating point variables.

Added New Power Function

A generic ac_pow_pwl() function was which accepted variable bases as well as exponents.

Default Template Parameter Changed

ac_exp_pwl() function now uses more fractional bits by default for an intermediate variable, to minimize error.

Corrected Issues

The following user-reported problems were fixed in this release:

- **(no bug #)**: File: ac_matrix.h – Transpose() member function was incorrect.
- **(no bug #)**: File ac_sqrt_pwl.h – Fixed bug in output near normalized 1 by adding extra bit to account for upward shifting of segments against the direction of concavity for ac_float types.

- **(no bug #)**: File ac_sincos_cordic.h – Renamed the typedef to avoid redeclaration conflict with ac_atan2_cordic.h

Release 3.1.0

The following topics describes the changes that were made to the *ac_math* library since the last release. This release provides new functionality and bug fixes. This version of ac_math was included in Catapult release 10.3.

Hyperbolic Tangent File Renamed

The file ac_hyperbolic_tan_pwl.h was renamed to ac_tanh_pwl.h for consistency with the other header files.

Improved AC Complex Support

The file ac_reciprocal_pwl.h was changed to use better bitwidth calculation for intermediate variables in the ac_complex<ac_float> version.

Corrected Issues

The following user-reported problems were fixed in this release:

- **(bug #51400):** File: ac_abs.h – Function for ac_int values results in hardware that uses an XW + 1 bit adder instead of an XW + 2 bit adder, where XW is the input bitwidth, thereby reducing area and improving QofR.
- **(bug #51145)**: Files: ac_inverse_sqrt_pwl.h, ac_reciprocal_pwl.h and ac_sqrt_pwl.h – Fixed expressions that directly manipulated the mantissa and exponent for ac_float outputs, replacing it with an ac_float constructor that performed normalization first and then modified the mantissa and exponent.
- **(no bug #)**: File: ac_sqrt.h – Fixed expression that could cause overflow.
- **(no bug #)**: Files: ac_arccos_cordic.h and ac_arcsin_cordic.h – Used different names for the class, function and variable in both the files to avoid name conflicts.

Release 2.0.10

This is the first official open-source release of the *ac_math* library. The following table lists the functions available in this release. For details about how to use the ac_math library consult the AC Math Reference Manual. This version of ac_math was included in Catapult release 10.2d.

Basic Math Functions

Function Type	Function Call	Approximation Method	Supported Data Types		
			ac_fixed	ac_float	ac_complex
Absolute Value	*ac_abs()*	N/A	Yes	Yes	No
Division	*ac_div()*	N/A	Yes	Yes	Yes
Normalization	*ac_normalize()*	N/A	Yes	No	Yes
Reciprocal	*ac_reciprocal_pwl()*	PWL	Yes	Yes	Yes
Logarithm Base e	*ac_log_pwl()*	PWL	Yes	No	No
	ac_log_cordic()	CORDIC	Yes	No	No
Logarithm Base 2	*ac_log2_pwl()*	PWL	Yes	No	No
	ac_log2_cordic()	CORDIC	Yes	No	No
Exponent Base e	*ac_exp_pwl()*	PWL	Yes	No	No
	ac_exp_cordic()	CORDIC	Yes	No	No
Exponent Base 2	*ac_pow2_pwl()*	PWL	Yes	No	No
	ac_exp2_cordic()	CORDIC	Yes	No	No
Generic Exponent	*ac_pow_pwl()*	PWL	Yes	No	No
	ac_pow_cordic()	CORDIC	Yes	No	No
Square Root	*ac_sqrt_pwl()*	PWL	Yes	Yes	Yes
	ac_sqrt()	N/A	Yes	No	No
Inverse Square Root	*ac_inverse_sqrt_pwl()*	PWL	Yes	Yes	Yes
Sine/Cosine	*ac_sincos()*	LUT	Yes	No	No
	ac_cos_cordic()	CORDIC	Yes	No	No
	ac_sin_cordic()	CORDIC	Yes	No	No
	ac_sincos_cordic()	CORDIC	Yes	No	No
Tangent	*ac_tan_pwl()*	PWL	Yes	Yes	No
Inverse Trig	*ac_atan_pwl()*	PWL	Yes	No	No
	ac_arccos_cordic()	CORDIC	Yes	No	No

Function Type	Function Call	Approximation Method	Supported Data Types		
			ac_fixed	ac_float	ac_complex
	ac_arcsin_cordic()	CORDIC	Yes	No	No
	ac_arctan_cordic()	CORDIC	Yes	No	No
Shift Left/Right	*ac_shift_left*	N/A	Yes	No	Yes
	ac_shift_right	N/A	Yes	No	Yes
Hyperbolic Tangent	*ac_tanh_pwl*	PWL	Yes	Yes	No
Sigmoid	*ac_sigmoid_pwl*	PWL	Yes	Yes	No
Softmax	*ac_softmax_pwl*	PWL	Yes	No	No

AC Matrix Class

The class ac_matrix implements a 2-D container class with a template parameter to specify the data type of the internal storage.
The class has member functions to implement some common operations including

- Assignment: operator=()
- Read-Only and Read-Write Element Access: *this(<row>,<col>)
- Comparison: operator!=(), operator==()
- Piecewise Addition: operator+(), operator+=()
- Piecewise Subtraction: operator-(), operator-=()
- Piecewise Multiplication: pwisemult()
- Matrix Multiplication (nested loops): operator*()
- Matrix Transpose: transpose()
- Sum All Elements: sum()
- Scale All Elements: scale(value)
- Formatted Stream Output: ostream &operator<<()

When using the computational functions with AC Datatypes, the form that returns a value is designed in such a way as to determine the full precision required in the output type in order to preserve accuracy during the operation. So using operator+ between two 10 bit ac_fixed matrices will return an 11 bit ac_fixed matrix. If you wish to prevent the bit growth and accept the truncation, you can use the compound operators +=,-=, etc. so that the target object receives the truncated values.
In addition to the built-in member functions, the ac_math library also includes stand-alone functions for more complicated linear algebra operations as described in the next section.

Linear Algebra Functions

The ac_math library includes several linear algebra functions that operate on either ac_matrix or plain C-style arrays. These functions, when used with AC Datatypes, are designed to give the user greater control over the bit precision of internal variables and the return value.

- Matrix Multiplication
- Matrix Determinant
- Cholesky Decomposition
- Cholesky Inverse
- QR Decomposition

Supported Compilers

The PWL functions use default template arguments. This requires using a C++ compiler that support the C++11 or newer standard.