Use better input/output value choices for SIMD tests

04/21/2017 04:05 PM - Berk Hess

Status: Closed
Priority: Normal
Assignee: Erik Lindahl
Category: core library
Target version:
Difficulty: uncategorized

Description
Many of the SIMD unit tests use input and output values that have 0 for all bits beyond single precision. Thus they won't catch any accidental double/float conversion errors.

Related issues:
- Related to GROMACS - Bug #2162: Several SIMD4 double precision reduce are act... Closed
- Related to GROMACS - Bug #2164: SIMD sqrt in double-precision build does not ... Closed

Associated revisions
Revision b9057925 - 08/28/2017 10:17 PM - Erik Lindahl
Improved SIMD test data to use all bits

The SIMD test data now uses all available bits in the current precision floating-point numbers, to catch errors where we lose precision or have mistakes in single/double conversions (e.g. return values). Since some operations now depend on the least significant bit, a few tests have been relaxed to allow a tolerance rather than require exact binary matching.

The new tests caught a precision loss in a reduction function for the SIMD-mimicking scalar functions, but this has never been used apart from testing, and should be harmless.

Fixes #2163.

Change-Id: I6d8f19d2aafeee8f42a2034c6100fcb10f6d2e81

Revision e34ead15 - 09/04/2017 11:41 PM - Erik Lindahl
More SIMD math argument checking, added unsafe options

This change adds more argument checking and safeguards for sqrt, exp2, and exp-related SIMD math functions, and properly documents allowed values. These functions now have an (optional) template parameter that makes it possible to avoid the checks where it is important to save every cycle, and the developer is certain that this usage is fine. For now we only use the unsafe versions in the nonbonded kernels.

The SIMD function test code has also been extended with options to allow denormals to be considered zero.

Fixes #2164.
Refs #2163.

Change-Id: I93ddadf74dd0fa013f61cf27fd1993f11cde28bc

Revision 6d32275c - 09/09/2017 09:24 AM - Berk Hess
Avoid inf in SIMD double sqrt()

Arguments >0 and <float_min to double precision SIMD sqrt() would produce inf on many SIMD architectures. Now sqrt() will return 0 for arguments in this range, which is not fully correct, but should be unproblematic.

Updated the tests to check for this range and to produce output

02/21/2020 1/3
that checks all double precision mantissa bits.

Fixes #2164.
Refs #2163.

Change-Id: lc6d2c6d4102d602703b40e7e8bcc1974a7283f7c

History

#1 - 04/21/2017 04:06 PM - Berk Hess
- Related to Bug #2162: Several SIMD4 double precision reduce are actual single precision added

#2 - 04/21/2017 11:14 PM - Mark Abraham
- Related to Bug #2164: SIMD sqrt in double-precision build does not work correctly added

#3 - 04/22/2017 10:03 AM - Erik Lindahl
- Tracker changed from Bug to Feature

Dropping to feature.

There nothing wrong with the current tests, but of course it might be possible to test more things by always using input data that cannot be reproduced exactly in single precision (but they were never designed to do this).

#4 - 04/22/2017 11:43 AM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #2163.
Uploader: Berk Hess (hess@kth.se)
Change-Id: gromacs~release-2016~lc6d2c6d4102d602703b40e7e8bcc1974a7283f7c
Gerrit URL: https://gerrit.gromacs.org/6599

#5 - 04/24/2017 09:17 AM - Berk Hess
- Subject changed from Bad input/output value choices of SIMD tests to Use better input/output value choices for SIMD tests

Changed the title to reflect the change to feature.
Many test values are such that we only test 3 mantissa bits. This means that e.g. not doing any Newton-Raphson iterations will not be caught. This is not good for tests that test math functions. We should test that we reach full claimed accuracy.

#6 - 04/24/2017 09:28 AM - Erik Lindahl
Agreed. I'll try to implement it in master, but it will likely trigger some false failures as we apply trial-and-error to see exactly how conservative we can make the thresholds for various compilers/hardware (which in turn will require more testing), so I prefer that we don't do it in the released branch.

#7 - 05/25/2017 03:09 PM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #2163.
Uploader: Erik Lindahl (erik.lindahl@gmail.com)
Change-Id: gromacs~master~f93ddadf74dd0f0a013f61cf27d1993f11cde28bc
Gerrit URL: https://gerrit.gromacs.org/8661

#8 - 05/25/2017 08:07 PM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #2163.
Uploader: Erik Lindahl (erik.lindahl@gmail.com)
Change-Id: gromacs~master~16d8f19d2aaeeeef8f42a2034c6100fc10f6d2e81
Gerrit URL: https://gerrit.gromacs.org/8662

#9 - 08/28/2017 10:30 PM - Erik Lindahl
- Status changed from Accepted to Resolved

Applied in changeset b905792595f066c1c088719f6bbf29d0c5f6e666.

#10 - 08/29/2017 01:09 PM - Aleksei lupinov
Insufficient tolerance on ARM NEON?

http://jenkins.gromacs.org/job/Matrix_PostSubmit_master/143/OPTIONS=gcc-5%20simd=ARM_NEON%20ASIMD%20release%20host=bs_jetson_tx1,label=bs_jetson_tx1/testReport/(root)/SimdVectorOperationsTest/cprod/
/home/jenkins/workspace/Matrix_PostSubmit_master/1ce3e58a/gromacs/src/gromacs/simd/tests/simd_vector_operations

02/21/2020 2/3
Failing comparison between refoX and cX
Requested abs tolerance: 0
Requested ulp tolerance: 2
(And values should not differ in sign unless within abs tolerance.)
Reference values: { -2.61181, -41.9362, -57.6741, -2.61181 }
SIMD values: { -2.61181, -41.9362, -57.6741, -2.61181 }
Abs. difference: { 1.90735e-06, 0, 3.8147e-06, 1.90735e-06 }
Ulp difference: { 8, 0, 1, 8 }

#11 - 12/31/2017 12:07 AM - Erik Lindahl
- Status changed from Resolved to Closed

Fixed several months ago.