GROMACS - Bug #2323

avx512 implementation of operator << requires immediate with gcc 6.4

12/03/2017 12:40 PM - Mark Abraham

Status:	Closed		
Priority:	Normal		
Assignee:			
Category:	core library		
Target version:			
Affected version - extra info:		**Difficulty:**	uncategorized
Affected version:	2018-beta1		

Description

With gcc 6.4 on dev-purley01, I get

```
FAILED: /opt/tcbsys/gcc/6.4/bin/g++   -DGMX_DOUBLE=0 -DHAVE_CONFIG_H -DUSE_STD_INTTYPES_H -Dlibgro
macs_EXPORTS -isystem ../src/external/lmfit -Isrc -isystem ../src/external/thread_mpi/include -I..
/src -isystem /opt/tcbsys/cuda/9.0/include -isystem /nethome/mabraham/progs/include -isystem /opt/
tcbsys/fftw/3.3.6-pl1-sse2/include -I../src/external/tng_io/include -Itng/include -mavx512f -mfma
   -std=c++11  -Wundef -Wextra -Wno-missing-field-initializers -Wpointer-arith -Wmissing-declarati
ons -Wall  -g -fno-inline  -fPIC   -fopenmp -MMD -MT src/gromacs/CMakeFiles/libgromacs.dir/listed-
forces/pairs.cpp.o -MF src/gromacs/CMakeFiles/libgromacs.dir/listed-forces/pairs.cpp.o.d -o src/gr
omacs/CMakeFiles/libgromacs.dir/listed-forces/pairs.cpp.o -c ../src/gromacs/listed-forces/pairs.cp
p
In file included from /opt/tcbsys/gcc/6.4/lib/gcc/x86_64-linux-gnu/6.4.0/include/immintrin.h:45:0,
                 from ../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512_general.h:39,
                 from ../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512.h:40,
                 from ../src/gromacs/simd/simd.h:126,
                 from ../src/gromacs/pbcutil/pbc-simd.h:51,
                 from ../src/gromacs/listed-forces/pairs.cpp:58:
../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512_simd_float.h: In function ‘gmx::SimdFInt32 g
mx::operator<<(gmx::SimdFInt32, int)’:
../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512_simd_float.h:517:16: error: the last argumen
t must be an 8-bit immediate
               _mm512_slli_epi32(a.simdInternal_, n)
               ^
FAILED: /opt/tcbsys/gcc/6.4/bin/g++   -DGMX_DOUBLE=0 -DHAVE_CONFIG_H -DUSE_STD_INTTYPES_H -Dlibgro
macs_EXPORTS -isystem ../src/external/lmfit -Isrc -isystem ../src/external/thread_mpi/include -I..
/src -isystem /opt/tcbsys/cuda/9.0/include -isystem /nethome/mabraham/progs/include -isystem /opt/
tcbsys/fftw/3.3.6-pl1-sse2/include -I../src/external/tng_io/include -Itng/include -mavx512f -mfma
   -std=c++11  -Wundef -Wextra -Wno-missing-field-initializers -Wpointer-arith -Wmissing-declarati
ons -Wall  -g -fno-inline  -fPIC   -fopenmp -MMD -MT src/gromacs/CMakeFiles/libgromacs.dir/listed-
forces/bonded.cpp.o -MF src/gromacs/CMakeFiles/libgromacs.dir/listed-forces/bonded.cpp.o.d -o src/
gromacs/CMakeFiles/libgromacs.dir/listed-forces/bonded.cpp.o -c ../src/gromacs/listed-forces/bonde
d.cpp
In file included from /opt/tcbsys/gcc/6.4/lib/gcc/x86_64-linux-gnu/6.4.0/include/immintrin.h:45:0,
                 from ../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512_general.h:39,
                 from ../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512.h:40,
                 from ../src/gromacs/simd/simd.h:126,
                 from ../src/gromacs/pbcutil/pbc-simd.h:51,
                 from ../src/gromacs/listed-forces/bonded.cpp:65:
../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512_simd_float.h: In function ‘gmx::SimdFInt32 g
mx::operator<<(gmx::SimdFInt32, int)’:
../src/gromacs/simd/impl_x86_avx_512/impl_x86_avx_512_simd_float.h:517:16: error: the last argumen
t must be an 8-bit immediate
               _mm512_slli_epi32(a.simdInternal_, n)
```

This is quite similar to a recent issue we fixed for one of the NEON SIMDs, where an immediate was required. The error might be triggered in not enough inlining has taken place before the check.

While experimenting with that fix, I did implement a fancy thing that ensures we do propagate an immediate to the point of call, but

we chose to use the commit 54042f4ba49b388064806cd60a497bf3d24a4e3a instead.

Associated revisions

Revision 9437181e - 12/11/2017 09:00 AM - Mark Abraham

Remove SIMD shift operators

These were almost unused, and caused problems in debug or clang builds
when the intrinsics required immediate operands that were not always
understood by the compiler to be available, because the function
argument was a variable. This could be fixed (see #2323), but there is
almost no known use for this functionality. For AVX-512, the fastMultiply
function is now implemented with explict intrinsics.

Fixes #2323

Change-Id: Ide45bde08deb425c18f35cd2adb263e566d643a1

History

#1 - 12/06/2017 01:08 AM - Roland Schulz

Nothing fancy is needed. One can simply pass std::integral_constant instead of int as an argument to the operator. Implementation which don't care about runtime/compile don't need to change. Others need to take std::integral_constant. Either as a 2nd overload or as the only option depending on whether runtime is supposed to be supported.

It seems we don't use << or >> anywhere outside the tests and internal helper function fastMultiply.
If we don't think we actually need them we could replace it with the slower _mm512_sll_epi32 and use _mm512_slli_epi32 directly in fastMultiply. Or of course simply delete them. Having SIMD functions we don't use is wasting time.

I'm in favor of deleting them. Opinions?

#2 - 12/07/2017 05:55 AM - Mark Abraham

If we have hopes of eventually writing source-compatible plain-C++ and SIMD-C++ kernels, then having shift operators is attractive.

Even if not, we probably want a function that can do a shift in SIMD - I don't remember whether these operators wrap such a function or not, but it too would need std::integral_constant.

#3 - 12/07/2017 06:00 AM - Roland Schulz

No there is no function. The operator is directly implemented with the intrinsic. I agree the source-compatibility issue would arise if we replace the operator with a function. But that has no advantage (as you say it still would need std::integral_constant) and isn't what I suggest. Instead I'm proposing to remove it outright because bit-shifting isn't used at all.

#4 - 12/11/2017 03:10 AM - Mark Abraham

I see three options:

1) remove the bit-shift operators entirely (implementing fastMultiply, which is only implemented for AVX-512 at the moment, with slli calls)
2) in Debug mode (when inlining is less likely to have seen an immediate operand propagated), implement the bit-shift operators with intrinsics that don't require immediate operands (we need the tests to pass, we don't need the Debug mode tests to test the same intrinsic as Release mode, that's what Release mode test builds are for; downside: RelWithAssert in Jenkins won't be a useful test of Release builds)
3) implement the shift operators in overloaded and templated flavours that can accept a std::integral_constant or an int, so that the caller has the option of explicitly specifying the immediate when they want that, and the variable when they need that (downside: work for no known gain; upside: would make a possible future spin-off SIMD library slightly more useful)

I think 1) is pretty clear right now, and Roland agrees. Will fix.

#5 - 12/11/2017 03:39 AM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2323.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~release-2018~Ide45bde08deb425c18f35cd2adb263e566d643a1
Gerrit URL: https://gerrit.gromacs.org/7321

#6 - 12/11/2017 08:54 AM - Berk Hess

- Status changed from New to Fix uploaded

#7 - 12/11/2017 09:45 AM - Mark Abraham

- Status changed from Fix uploaded to Resolved

Applied in changeset 9437181eacbb7194ef6f2a4d21489b25db6b2661.

#8 - 12/11/2017 11:54 AM - Erik Lindahl

- Status changed from Resolved to Closed