Feedback on P0214

Document Number	P0820R4
Date	2018-06-08
Reply-to	Tim Shen <timshen91@gmail.com> Matthias Kretz <m.kretz@gsi.de>
Audience	LWG

Abstract

We investigated some of our SIMD applications and have some feedback on P0214R9.

The presented change resolves an NB comment on the PDTS

Revision History

P0820R3 to P0820R4

- Removed changes for `simd_abi::deduce` since it's already covered by P0964.
- Remove changes to the `simd_cast` return types (to be reconsidered later).
- Move concat and split related changes to P1118.
- Rebase the mismatched wording onto the Parallelism v2 TS.

P0820R2 to P0820R3

- Rebase onto P0214R9.
- Adapt to P0964R1.
- Changed wording for alias `scalar` and `fixed_size`.

P0820R1 to P0820R2

- Rebased onto P0214R7.
- Extended `static_simd_cast` and `simd_cast` to use `rebind_abi_t`.
- Change `simd_abi::scalar` to an alias.

P0820R0 to P0820R1

- Rebased onto P0214R6.
- Added reference implementation link.

- For concat() and split(), instead of making them return simd types with implementation defined ABIs, make them return `rebind_abi_t<...>`, which is an extension and replacement of original `abi_for_size_t`.
- Removed the default value of `n` in `split_by()`.
- Removed discussion on relational operators. Opened an issue for it (https://issues.isocpp.org/show_bug.cgi?id=401).
- Proposed change to fixed_size from a struct to an alias, as well as guaranteeing the alias to have deduced-context.

simd_abi::scalar and fixed_size<N> are not an aliases

One possible implementation of ABI is to create a centralized ABI struct, and specialize around it:

```
enum class StoragePolicy { kXmm, kYmm, /* ... */ };
template <StoragePolicy policy, int N> struct Abi {};

template <typename T> using native = Abi<kYmm, 32 / sizeof(T)>;
template <typename T> using compatible = Abi<kXmm, 16 / sizeof(T)>;
```

Then every operation is implemented and specialized around the centralized `struct Abi`.

Unlike `native` and `compatible`, `scalar` and `fixed_size` is not an alias. Currently they require extra specializations other than the ones on `struct Abi`.

Wording

Modify [parallel.simd.synopsis] as follows:

```
~~struct~~using scalar ~~{}~~= see below;
template <int N> ~~struct~~using fixed_size ~~{}~~= see below;
```

Modify [parallel.simd.abi] as follows:

```
~~struct~~using scalar ~~{}~~= see below;
template <int N> ~~struct~~using fixed_size ~~{}~~= see below;
```

Modify [parallel.simd.abi] p3 as follows:

`scalar` is an alias for an unspecified ABI tag that is different from `fixed_size<1>`. Use of the scalar tag type requires data-parallel types to store a single element (i.e., simd::size() returns 1). ~~[Note: `scalar` shall not be an alias for `fixed_size<1>`. — end note]~~

Modify [parallel.simd.abi] p5 as follows:

`fixed_size<N>` is an alias for an unspecified ABI tag. `fixed_size` does not introduce a non-deduced context. Use of the `simd_abi::fixed_size<N>` tag type requires data-parallel types to store N elements (i.e. `simd<T, simd_abi::fixed_size<N>>::size()` returns N). `simd<T, fixed_size<N>>` and `simd_mask<T, fixed_size<N>>` with `N > 0` and `N <= max_fixed_size<T>` shall be supported. Additionally, for every supported `simd<T, Abi>` (see [simd.overview]), where Abi is an ABI tag that is not a specialization of `simd_abi::fixed_size`, `N == simd<T, Abi>::size()` shall be supported.

Reference

- The original paper: P0214R9
- Experimental implementation: https://github.com/google/dimsum