Motivation

• C++ for OpenCL kernel language [1] brings many C++ language features to OpenCL, while keeping backward compatibility with OpenCL C.
  • Version 1.0 was built on top of C++17 and OpenCL 2.0.
• Evolution of C++ for OpenCL alongside OpenCL standard is key.
• Main design goal of C++ for OpenCL 2021 is compatibility with OpenCL 3.0 (released in Sep 2020).
• clang-14 provides complete experimental support of OpenCL C 3.0.
• C++ for OpenCL 2021 support in clang is built on top of existing clang features for maximal code reuse and backward compatibility guarantee.
C++ for OpenCL 2021 overview

• Key differences to OpenCL C 3.0:
  • Many native C++ language features are enabled:
    – Object oriented or generic programming.
  • In common behaviour:
    – Variadic macros can be used as in C++17.
    – Atomic types can be used with built-in operators if the sequential consistency memory model is supported.
    – Blocks are not supported.
    – `NULL` is defined as `nullptr` rather than `((void*)0)`.
    – C++ for OpenCL limits usage of some C-specific features:
      ▪ implicit type conversions are stricter.
      ▪ `restrict` keyword is not supported.
      ▪ `goto` statements follow the rules from C++17.

• Key differences to C++ for OpenCL 1.0:
  • C++ for OpenCL 2021 provides all optional features from OpenCL C 3.0 including but not limited to:
    – generic address space, program scope variables in global address space, sequential consistency memory model, etc.
  • Address space removal type trait introduced.
Implementation in Clang

- Experimental support released in clang-14.
- Extended command line flag for language version:
  \[ \text{clang++ -cl-std=clc++2021 --target=spirv64 mykernel.clcpp} \]
- Implicitly defined version macros:
  \texttt{__CL\_CPP\_VERSION\_2021__} and \texttt{__OPENCL\_CPP\_VERSION__} set to 202100.
- Further unified with OpenCL C
  - \texttt{getOpenCLCompatibleVersion()} helper performs mapping from C++ for OpenCL or OpenCL C version to a corresponding compatible OpenCL version.
- Extended optionality of generic address space to C++ specific constructs:
  - e.g., implicit pointer to object parameters, special member functions.
  - \texttt{getDefaultOpenCLPointeeAddrSpace()} helper determines whether generic or private address space should be deduced by the compiler.
- Added address space removal utility
  - Based on feedback from LLVM community.
Demonstration – generic address spaces optionality

```c
struct C {
    void foo(int *par);
}

#ifndef __opencl_c_generic_address_space
    // W/o generic address space (GAS)
    // support an overload is needed for
    // objects in __global address space.
    void foo(int *par) __global;
#endif

__global C globC{};
void bar() {
    __private C locC;
    int i;
    locC.foo(&i);
    globC.foo(&i); // error w/o GAS support.
}
```
Demonstration – generic address spaces optionality

```c
struct C {
    void foo(int *par);
#endif __opencl_c_Generic_address_space
    // W/o generic address space (GAS)
    // support an overload is needed for
    // objects in __global address space.
    void foo(int *par) __global;
#endif
};
__global C globC{};
void bar() {
    __private C locC;
    int i;
    locC.foo(&i);
    globC.foo(&i); // error w/o GAS support.
}
```

```c
template<typename T> void helper(T *par) {
    #ifdef __opencl_c_Generic_address_space
        // If GAS is supported T is deduced to it.
        // As local variables can not be declared
        // with GAS an address space qualifiers
        // needs to be removed.
        typename __remove_address_space<T>::type var;
    #else
        T var;
    #endif
}
void C::foo(int *par) {
    helper(par);
}
#endif __opencl_c_Generic_address_space
void C::foo(int *par) __global {
    helper(par);
}
#endif
```
Conclusions and feedback

• More details about the new versions can be found in the unified language documentation:
  • https://www.khronos.org/opencl/assets/CXX_for_OpenCL.html

• C++ for OpenCL 2021 is implemented in clang-14 as an experimental feature and it provides compatibility with OpenCL C 3.0 and C++17.
  • https://clang.llvm.org/docs/OpenCLSupport.html#cxx-for-opencl-impl

• Developers are invited for experimenting and contributing.
  • Feedback helps us to identify bugs, missing features and shape the language evolution.

• Future: more effort is needed on expanding test coverage towards the final release of C++ for OpenCL 2021.
Thank You
Danke
Gracias
谢谢
ありがとうございました
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה