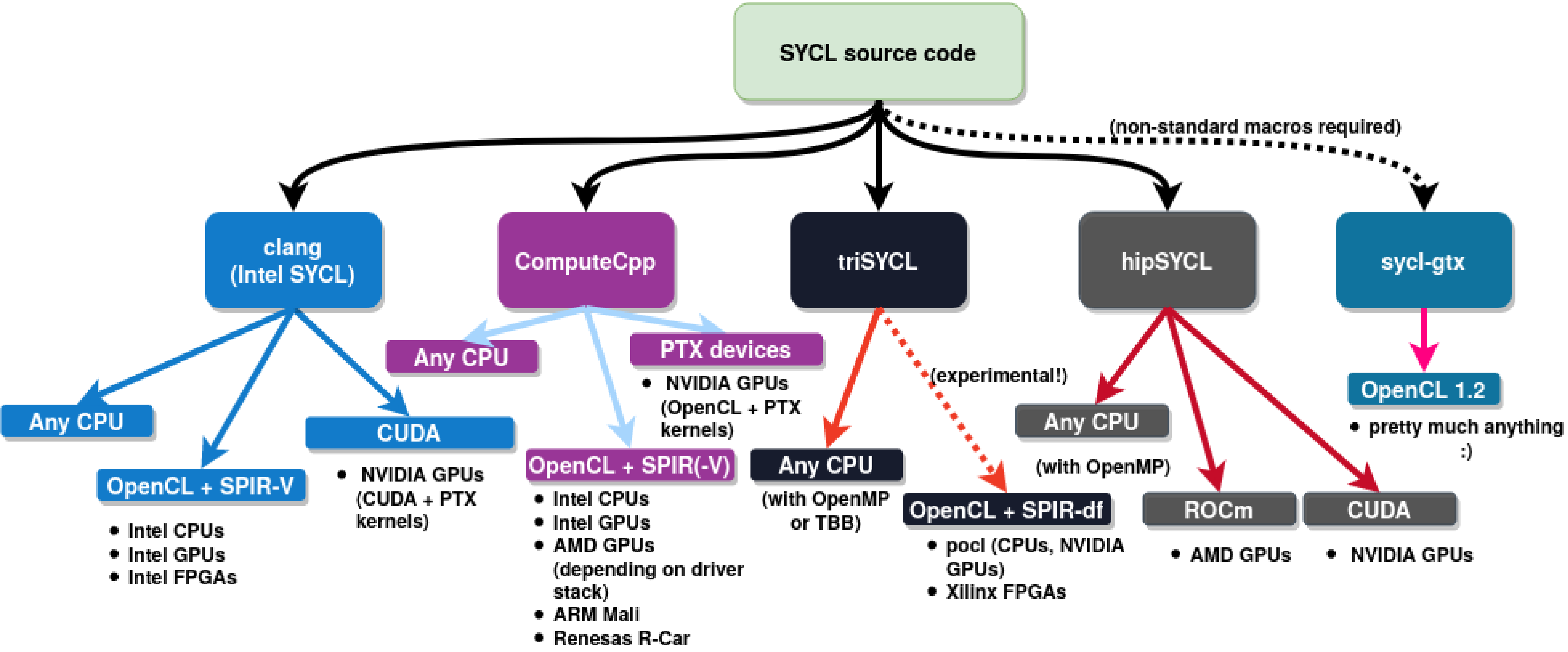
triSYCL

An open-source C++20-based SYCL workbench

Ronan Keryell
Xilinx Research Labs (San José, California) & Khronos SYCL specification editor
2020/04/27

Khronos Group SYCL C++ DSL — current public ecosystem



Aksel Alpay 2020/03/13 https://raw.githubusercontent.com/illuhad/hipSYCL/master/doc/img/sycl-targets.png

(tri)SYCL (short) story

- 2011: SYCL started at Khronos as Higher-Level Model for OpenCL with Codeplay & Qualcomm
- 2014: triSYCL started at AMD PAE to understand/validate the specification & experiment/research/…
- Now mainly backed by Xilinx, with various other contributions (RedHat, Intel…)
- Target CPU first: pure C++ library providing host device
  - Use C++ classes to emulate hardware features without real device compiler
  - Allow ISO C++ experiments (mdspan, ranges, fiber_context, pipes)
- ComputeCpp was (and is still) the strong production implementation
  - But close-source! ☹ Hopefuly they had a Community Edition ☺
- Bootstrap the ecosystem? Open standard SYCL + open-source implementation!
  - triSYCL helped development of other implementations
  - Allow quick prototyping & extensions
  - → SYCL-gtx, hipSYCL, Celerity…
- 2017 prototype for Xilinx FPGA: Clang/LLVM-based device compiler for Xilinx SDx OpenCL stack
- 2019/01/11 Intel SYCL open-source implementation to be up-streamed to Clang/LLVM
  - Branded “oneAPI DPC++”
  - Quite more robust than minimalistic triSYCL device compiler!
- →ReSYCLe Clang/LLVM device compiler from Intel ☺
  - Reuse LLVM passes from triSYCL device compiler & collabrative public code review
  - Reuse triSYCL C++ runtime to target Xilinx Versal ACAP AIE CGRA (coarse-grain reconfigurable array)

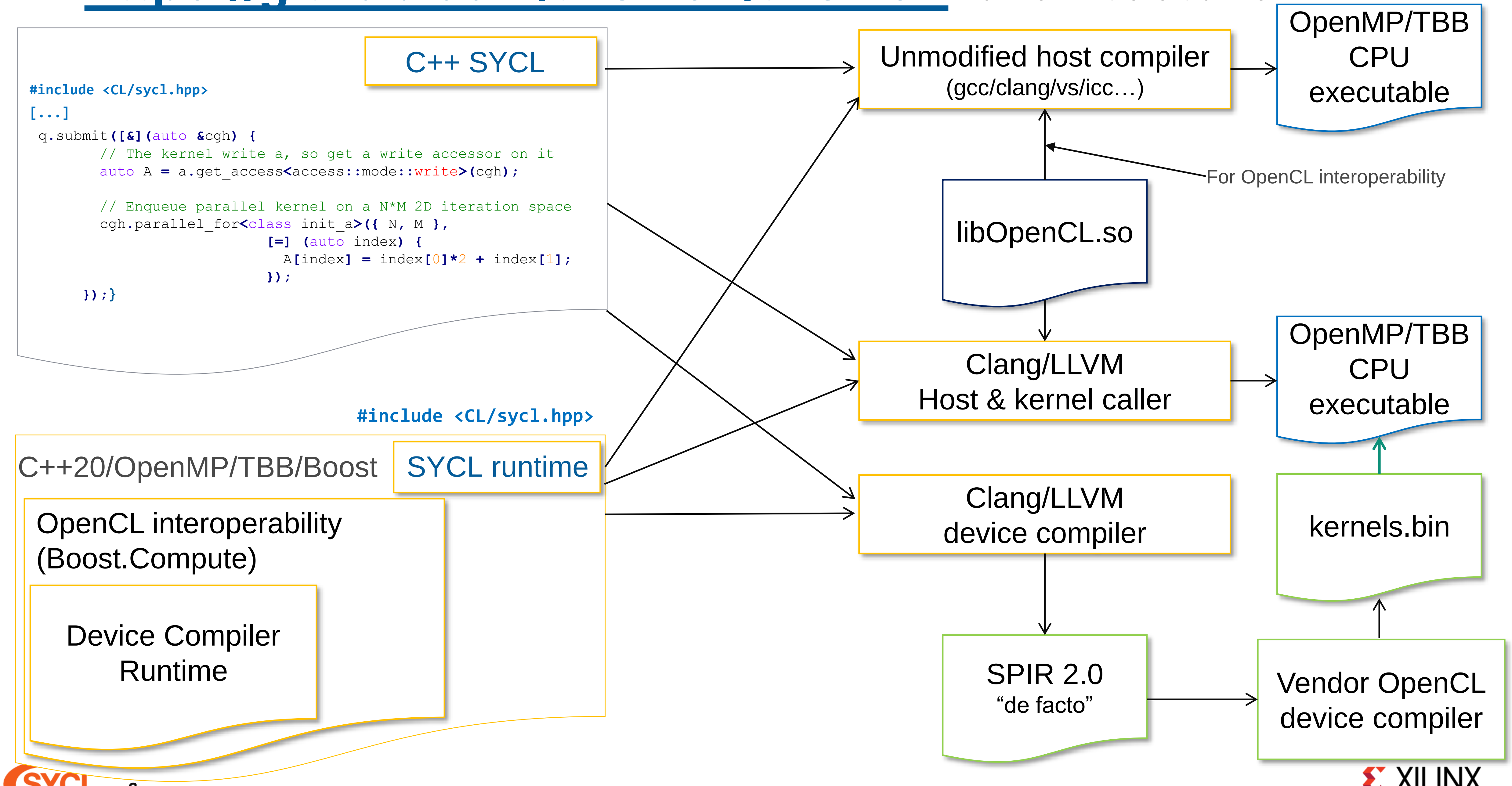
triSYCL

- Open Source (incomplete) SYCL implementation of 1.2.1 (with some [deprecated] ~~2.2~~ features)
  - https://github.com/triSYCL/triSYCL
- Clean-room open-source SYCL implementation without compromise
- Header only templated classes, easy to integrate in existing project
- C++20
  - Do not waste time writing old C++… It is hard enough with C++20! ☺
  - Better usability for SYCL programmer
  - Give feedback to ISO C++
- Use existing libraries to fight Not Invented Here (NIH)/Not Written Here (NWH) syndromes
  - CPU execution OpenMP, TBB, `std::thread`,… + automatic vectorization from host C++ compiler
  - Boost.Compute for OpenCL interaction
- Helped Khronos Group to refine the SYCL and [deprecated] ~~OpenCL C++ 2.2~~ standards
  - Languages are now too complex to be defined without implementing...

SYCL 1.2.1 OpenCL interoperability mode

- Requirement of SYCL 1.2.1 & does not require a device compiler (non single-source mode...)
- Implemented with Boost.Compute in triSYCL
- Allow using existing OpenCL kernels (OpenCL C source or binaries) in nice SYCL way
  - No need for OpenCL host API boilerplate
  - Communications/computations overlapping for free
  - SYCL is useful for OpenCL programmers!
- OpenCL built-in kernels are *very* common in FPGA world
  - Written in Verilog/VHDL or Vivado HLS C++
    - But with SDAccel OpenCL kernel interface
  - Kernel libraries
  - Linear algebra
  - Machine learning
  - Computer vision
  - Direct access to hardware: wire-speed Ethernet…
- Version of TensorFlow SYCL with Xilinx FPGA kernels (but no longer maintained ☹)

https://github.com/triSYCL/triSYCL architecture



```
#include <CL/sycl.hpp>
[...]
 q.submit([&](auto &cgh) {
        // The kernel write a, so get a write accessor on it
        auto A = a.get_access<access::mode::write>(cgh);

        // Enqueue parallel kernel on a N*M 2D iteration space
        cgh.parallel_for<class init_a>({ N, M },
                           [=] (auto index) {
                             A[index] = index[0]*2 + index[1];
                           });
     });}
```

```
#include <CL/sycl.hpp>
```

C++ SYCL
C++20/OpenMP/TBB/Boost
SYCL runtime
OpenCL interoperability (Boost.Compute)
Device Compiler Runtime
Unmodified host compiler (gcc/clang/vs/icc…)
OpenMP/TBB CPU executable
For OpenCL interoperability
libOpenCL.so
Clang/LLVM Host & kernel caller
OpenMP/TBB CPU executable
Clang/LLVM device compiler
kernels.bin
SPIR 2.0 “de facto”
Vendor OpenCL device compiler

Configurable SYCL namespace

- No size fits all… Even for SYCL
- #include "CL/sycl.hpp"
  - Traditional `::cl::sycl`
- #include "triSYCL/sycl.hpp"
  - Innovative `::trisycl`
  - Allow cohabitation with other main implementation in `::cl::sycl`
  - Independently of what is included, always accessible to access triSYCL extensions
- #include "SYCL/sycl.hpp"
  - `::sycl`
  - Spare 4 characters… because you deserve some laziness too ☺

Resources

- Open-source (mainly Xilinx contributions)
  - https://github.com/triSYCL/triSYCL
- Open-source fusion triSYCL+Intel SYCL to Xilinx FPGA
  - https://github.com/triSYCL/sycl
- Open-source fusion triSYCL+Intel SYCL to Xilinx Versal ACAP AIE CGRA
  - To be open-sourced… Stay tuned!

Conclusion

- Not a product for normal SYCL programmers: incomplete implementation ☹
  - Good news: there are many other SYCL implementations today! ☺
- Research project for future SYCL, extensions, new back-ends (FPGA, CGRA, NNP…), future C++…
- Simple architecture using latest C++
  - Pure modern C++ DSL as pure header library
  - Allow interacting with other implementations
    - No size fits all: probably required for extreme heterogeneous programming CPU+GPU+CGRA+FPGA+DSP+MPI+…
  - Can build abstractions for any device: FPGA, ACAP++… See other presentation
  - Use host fall-back mode for target emulation, debug & co-design on CPU for free
- Independent from OpenCL with concept of backends
  - OpenCL: Xilinx FPGA: SPIR "de-facto" + OpenCL host API for resource control
  - Non-OpenCL: Xilinx Versal ACAP AIE (CGRA): LLVM IR + C++ runtime for resource control (not open-sourced yet)
- Contribute to good open-source collaboration across SYCL community

Thank You