Motivation

- FPGA are hard to program
- HLS (High-Level Synthesis) has made it better
- Tools use lots of non-standard language extensions
- Tools are usually split source
- SYCL for FPGA is trying to make it simpler
- Single-source
- Uses the usual compiling process of C++
- Pure modern C++, can be implemented as a normal library for host only execution
- Standard
- SYCL is a single-source C++ DSL targeting a large variety of accelerators in a unified way by using different back-ends.
- SYCL is supported by Xilinx Vitis 2020.2 with some LL VM passes from tSyCL.
- The FPGA device configuration is generated by Xilinx Vitis 2020.2 fed with LLVM IR SPIR and Xilinx XRT.
- We present an experimental SYCL implementation targeting Xilinx Alveo FPGA cards by merging different back-ends.

Implementation

- Based on Intel’s oneAPI DPC++ because:
  - Open-source
  - Upgrading the IR to LLVM latest ToT
  - Generates configuration for the backend
  - Run optimizations
  - Implies better support of the SYCL standard
  - Allow better control on the design and performances
  - Expose more hardware details

- Tools are usually split source
- Cards use lots of non-standard language extensions
- Single-source
- Uses the usual compiling process of C++
- Pure modern C++, can be implemented as a normal library for host only execution
- Standard

FPGA-specific extensions to the SYCL standard

- Pipeline annotations
  - For (...) sycl::pipeline([[ ... ]]);
  - The for loop will get each stage pipelined in hardware and this will speed up the loop at the cost of using a little bit more hardware and latency.
- DDR bank accessor property
  - sycl: accessor accessor
  - The memory accessed by accessor will be placed in the DDR memory bank 1

Future Work

- Focus more on performance
- Expose more hardware details
- Give more control over HLS to the user
- Better adapt the optimizations to FPGA
- Add support for more Xilinx hardware
- Test the implementation on more and bigger applications