SYCL for Vitis 2020.2: SYCL & C++20 on Xilinx FPGA

Luc Forget¹ Andrew Gozillon² Gautier Harnisch¹ Ronan Keryell¹ Lin-Ya Yu¹ Ralph Wittig¹

¹Xilinx ²University of the West of Scotland

Abstract

SYCL is a single-source C++ DSL targeting a large variety of accelerators in a unified way by using different back-ends. We present an experimental SYCL implementation targeting Xilinx Alveo FPGA cards by merging 2 different open-source implementations, Intel’s oneAPI DPC++ with some LLVM passes from triSYCL. The FPGA device configuration is generated by Xilinx Vitis 2020.2 fed with LLVM IR SPIR and Xilinx XRT is used as a host OpenCL API to control the device.

Motivation

- SYCL for 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

Implementation

- Based on Intel’s oneAPI DPC++ backends:
  - Open-source
  - Using OpenCL
  - Based on LLVM latest ToT
- We use Xilinx’s OpenCL runtime from XRT
- The compilation flow required changes
- Using Vitis’s v = + as back-end compiler
- Needs downgrading from LLVM ToT to LLVM 6.x
- Needs converting SPIR-V builtins to “SPIR-df (de-facto)”
- We tweaked the optimization pipeline

Targets

- We support 3 emulation targets and 1 for real hardware execution
  - Library only
  - SYCL runtime is used as normal C++ library and does not use Vitis at all
  - Any C++ compiler can be used to compile
  - Fastest to compile and execute on CPU
- Kernel code will be executed natively on the host
- Can use runtime checks like: sanitizers, vaig... and usual debuggers
- But it’s the furthest from hardware
- Software emulation
  - Needs to use our custom compiler and use Vitis compiler
  - Kernel code is run natively on the host with the Vitis software emulator
- Kernel code is isolated from host code
- Hardware emulation
  - Needs to use our custom compiler and Vitis RTL simulator
  - Kernel code is isolated from host code
- Hardware execution on FPGA
  - Generates reports about the real resource usage and timings
- Kernel code is executed on the FPGA
- Slowest to compile including Vitis RTL synthesis and FPGA place & route

FPGA-specific extensions to the SYCL standard

- Allow better control on the design and performances
  - Pipeline annotations
    - for (...) sycl: pipeline(a [ ... ]);
    - The for loop will get each stage pipelinised in hardware and this will speedup the loop at the cost of using a little bit more hardware and latency
- DDR bank accessor property
  - sycl: accessor Accessor
    - 
- Hardware execution on FPGA
  - Generates reports about the real resource usage and timings
  - Kernel code is executed on the FPGA
- Slowest to compile including Vitis RTL synthesis and FPGA place & route

Future Work

- Focus more on performance
- Expose more hardware details
- Give more control over HLS to the user
- Better adapt the optimizations to FPGA
- Usability
- Fix more compatibility issues between the SYCL toolchain and Vitis
- Expose more hardware details
- Add support for more Xilinx hardware
- Test the implementation on more and bigger applications