Chair of Decentralized Systems Engineering
Department of Informatics

LLVM Compiler for Peer-to-Peer Communication on Cloud FPGAs

General information	**Advisor**	Dr. Atsushi Koshiba
	Email	atsushi.koshiba@tum.de
	Date	31.1.2022
Type	Master / Bachelor / Guided Research	
Description	Hardware acceleration with Field Programmable Gate Arrays (FPGAs) is a key technology to satisfy the high-performance demands of cloud workloads under the severe energy constraint in data centers. FPGAs are well suited for data processing as they can be connected directly to storage or network devices so that custom logics on FPGA process the incoming data without going through CPU-side memory. Such peer-to-peer communications can significantly improve the performance and energy efficiency of the computation. Although FPGA has the advantage of accelerating data-intensive workloads in such ways, they are lack programmability and portability. FPGA vendors offer programming interfaces enabling peer-to-peer communications between FPGA and other devices [1, 2]. However, they are realized by vendor-specific pragmas or extensions and are only available on specified FPGA boards. Custom logics written in this manner are not portable among different FPGA vendors/board types. LLVM compilers for High-Level Synthesis (HLS) [3] and Domain-Specific Language (DSL) [4] have been well studied for improving FPGA	

programmability and portability. They interpret software code written in C/C++ or other high-level languages into register transfer level (RTL) code so that they can be synthesized on their target FPGAs as custom logic. However, they only focus on generating a standalone hardware module, and device-to-device communication is not supported.

This project aims to build an LLVM compiler to ease the development of custom logic for gaining FPGA portability and programmability. We propose a platform- and vendor-independent programming model that abstracts the communication layer between custom logics on FPGA and the other devices (memory, disk, network). The LLVM compiler generates an RTL code of custom logic configurable on various FPGA cards. The proposed system will be implemented upon an open-source LLVM compiler offered by Xilinx [5]. We also verify the portability of the generated RTL code with various FPGA cards we have.

Keywords

FPGA, LLVM, heterogeneous computing, portability, near-data processing, networking

Goals

Concrete outcomes

1. Propose a programming model that abstracts the communication layer for FPGA custom logic. The student can either propose a new one or extend an existing model such as OpenCL.
2. Design an LLVM compiler that generates hardware modules portable among different FPGAs.
3. Implement the compiler by extending the Vitis LLVM frontend [5].
4. Evaluate the portability of the generated modules with two different Xilinx FPGA cards: Alveo U50 and Nexys Video.

Bonus points

1. Publish the extended compiler as an open-sourced project.
2. Support an FPGA card from different vendors, i.e., Intel FPGA.

Prerequisites

Compulsory

- Good knowledge and experience of C/C++
- Knowledge of LLVM

- Knowledge of FPGA

Preferred

- Experience in FPGA programming (HDL, HLS, etc.)
- Knowledge of CUDA or OpenCL

References

1. PCIe Peer-to-Peer, https://xilinx.github.io/XRT/master/html/p2p.html
2. Intel FPGA SDK for OpenCL Overview, https://www.intel.com/content/www/us/en/programmable/documentation/mwh1391807965224.html#ewa1411747396740
3. A Survey and Evaluation of FPGA High-Level Synthesis Tools, IEEE Trans on CADICS'16, https://ieeexplore.ieee.org/abstract/document/7368920
4. Generating FPGA-based image processing accelerators with Hipacc, ICCAD'17, https://ieeexplore.ieee.org/document/8203894
5. Xilinx Vitis HLS LLVM, https://github.com/Xilinx/HLS

Application process

Please send an email to the advisor including the following:

- Email subject: "Thesis application (DSE)"
- CV
- A copy of your transcript(s)
- A **motivation statement**, please include samples of your work that you are proud of (e.g., major projects, open-source contributions, Github page, etc.) and/or writing samples (e.g., your technical blog, project reports, etc.)