Accepted for FCCM 2020

ZRLMPI: A Unified Programming Model for Reconfigurable Heterogeneous Computing Clusters

Burkhard Ringlein*†, Francois Abel†, Alexander Ditter*, Beat Weiss†, Christoph Hagleitner†, and Dietmar Fey*
†IBM Research Europe, *Friedrich-Alexander University Erlangen-Nürnberg
{ngl, fab, wei, hle}@zurich.ibm.com, {burkhard.ringlein, alexander.ditter, dietmar.fey}@fau.de

***Abstract*—Over the past two decades, the Message Passing Interface (MPI) has evolved as the de-facto standard for programming High-Performance Computing (HPC) clusters. Its widespread utilization led to the rapid development of applications and high reusability. Meanwhile, energy- and compute-efficient devices such as Field-Programmable Gate Arrays (FPGAs) are stepping into modern data centers and HPC clusters to address the nearing end of technology scaling. This combination of traditional CPU servers and FPGA nodes leads to Reconfigurable Heterogeneous HPC (ReH^{2}PC) systems that are particularly cumbersome to program because of the absence of a standard programming model. This work advocates the use of MPI to program such ReH^{2}PC clusters and presents a proof of concept based on a cross-compiler, a High-Level Synthesis library, a C++ library, an FPGA- and a CPU-runtime environment. The result is a one-click solution, which compiles a standard MPI application for a ReH^{2}PC cluster.**

I. Programming ReH^{2}PC Clusters

Today's High-Performance Computing (HPC) systems can be classified into three classes. The first and traditional HPC class solely consists of CPU servers, while the second class, typically referred to as Reconfigurable HPC (ReHPC), is only comprised of Field-Programmable Gate Arrays (FPGAs) nodes. The third class is named Reconfigurable Heterogeneous HPC (ReH^{2}PC) because it comprises a mixture of the CPU servers from the first class and the FPGA nodes from the second class. Unfortunately, despite many attempts, no standard has yet emerged for the programming of such heterogeneous clusters. This absence of agreement hinders the rapid development of applications using FPGAs in HPC, and motivated us to reconsider the use of the Message Passing Interface (MPI) for ReH^{2}PC platforms. MPI is widely adopted in the HPC community and we want to demonstrate that, with its standardized syntax and semantics, it also fits as a single programming model for ReH^{2}PC clusters. To avoid re-coding every application for every specific heterogeneous cluster, we propose a High-Level Synthesis (HLS) approach, where the application code (for e.g. C/C++) is turned into a hardware design description at some point in the compilation flow. An HLS design is typically coded as a set of processes interconnected with object-oriented stream constructs (e.g. AXI4 streams). Since MPI already defines the parallel execution and communication of the node processes, we think that it is a proper forking point to enter the FPGA HLS synthesis.

Our ambition is to take existing MPI-based applications that were developed for CPU clusters, and execute them on a ReH^{2}PC cluster without any code modifications. Our proof of concept uses virtual machines for the CPUs and a set of network-attached FPGAs managed by the framework described in [1].

```
int msg[1];
int next_node = (rank + 1) % size;
int previous_node = rank -1;
if(rank == 0) {
  msg[0] = 0xcaffee;
  MPI_Send(&msg[0], 1, MPI_INTEGER, 1, 0, MPI_COMM_WORLD);
  MPI_Recv(&msg[0], 1, MPI_INTEGER, size-1, 0, MPI_COMM_WORLD, &status);
} else {
  MPI_Recv(&msg[0], 1, MPI_INTEGER, previous_node, 0, MPI_COMM_WORLD, &status);
  MPI_Send(&msg[0], 1, MPI_INTEGER, next_node, 0, MPI_COMM_WORLD);
}
```

Listing 1. Snippet of an MPI message ring example. Rank 0 is executed on the CPU, all other cases on FPGAs. A `rank` is a unique id per MPI node.

II. ZRLMPI: MPI for ReH^{2}PC

The goal of ZRLMPI is to bring CPUs and FPGAs to work together efficiently using a single source of code. As an example, consider the MPI code of Listing 1, which forwards a message around a ring of multiple nodes from a sender (`rank 0`) back to that same node. In such a programming approach, the user is not expected to annotate the MPI code or to use HLS tools her/himself in order to bring the program to a ReH^{2}PC cluster. This step is automated by our cross-compiler (`ZRLMPIcc`) that identifies the parts of the program that will be executed on FPGAs and transforms these parts from the original C code to synthesizable HLS code. To identify these parts, `ZRLMPIcc` uses a user-defined `rankfile` that maps every rank to a specific physical node. This is analogous to the *affinity* concept of MPI. To implement the MPI synchronization and collective routines via the underlying cluster communication protocol, we developed an HLS core called `Message Passing Engine` (MPE). This MPE is merged with the application HLS code by `ZRLMPIcc` and is synthesized to a partial bitstream. In parallel, the CPU specific parts are also emitted by `ZRLMPIcc` and compiled together with the ZRLMPI software runtime library (`ZRLMPIlib`). This `ZRLMPIlib` is the software counterpart of the MPE that synchronizes CPU and FPGA nodes. To distribute the partial bitfiles and software binaries as specified by the `rankfile`, we've developed a deployment framework (`ZRLMPIrun`) using the FPGA management runtime of platform [1].

References

[1] B. Ringlein, F. Abel, A. Ditter, B. Weiss, C. Hagleitner, and D. Fey, "System Architecture for Network-Attached FPGAs in the Cloud using Partial Reconfiguration," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, Barcelona, Spain: IEEE, 2019, pp. 293–300. DOI: 10.1109/FPL.2019.00054.

This is the accepted version of the article published by IEEE: DOI 10.1109/FCCM48280.2020.00051