Literature Review

Cheng Liu

September 22, 2016

1 C++ Actor Framework Using OpenCL

This paper introduces OpenCL enabled actors to C++ Actor Framework [1, 2]. The design goals are listed as follow:

- Hide complexity of OpenCL management
- Seamless integration into CAF with respect to access transparency as well as location transparency

The concept of the Actor model seems to be interesting. It describes the applications in terms of isolated actor entities which communicate through asynchronous message passing. This abstraction helps to handle complex concurrent processing and can be distributed to larger systems.

Here is a simple explanation about the actor model which I got from Wiki. *An actor is a computational entity that, in response to a message it receives, can concurrently:*

- *send a finite number of messages to other actors*
- *create a finite number of new actors*
- *designate the behavior to be used for the next message it receives*

With this concept, the most straightforward question is whether we can build such an actor framework for FPGA computing system. Although the hardware implementation on FPGAs is relatively limited (for instance, many dynamic behaviors are difficult to be implemented on FPGAs), there is no barrier to support FPGA computing with a similar philosophy.

An actor can be an instance of the hardware accelerator and a number of actors can be implemented on FPGA devices. The actors can communicate with each other through a packed-switched interconnection network efficiently with the well-studied NoC systems. Although it is difficult to spawn new actors on FPGAs at run-time, a tightly coupled processor can help with this and new actors can be scheduled to idle accelerator instances.

Eventually, the users need to provide two things **1)the application organized with the actor framework 2)and hardware implementations using either HDLs or HLS tools for all the different actor instances**. When the

two aspects are ready, the framework is supposed to provide an efficient parallel execution of the application on a CPU-FPGA computing system.

From a hardware designer's point of view, the system works similar to a data driven computing machine. It is just that the communication granularity is now a message instead of a data and the computing granularity is an larger actor instead of an operation.

Major concerns: There are also similar projects in FPGA community. The researchers tried to integrate a hardware accelerator as well as its drivers as a thread. A lot of threads can be implemented on the same FPGAs. A parallel application can be thus abstracted as many threads and the FPGA is taken as a pool of threads. By scheduling different threads in to the FPGAs at runtime, they can also complete larger parallel applications.

What kinds of applications will fit the Actor Framework? Will the applications be too complex for FPGAs?

It seems the major benefit of the actor framework is to automatically parallelize the application and the scheduler is critical to the performance of the resulting system. Then how are we going to implement the scheduler? Will it be too complicate for hardware? It is possible to put the scheduler on a processor, but it can be a bottleneck as the hardware actor may keep waiting for its response.

Each actor may require specific hardware accelerator, and all the actors to be used in the application should be implemented at compilation time. If all the actors are implemented at the same time, the hardware resource can be a key design constrain. If runtime reconfiguration is supported, the reconfiguration time can be a big challenge and performance may degrade a lot. Basically, the performance of the system may not be as good as expected. Or the usage of the system can be limited to very large FPGA system.

Compared to CPU and GPU, FPGA wins on a small set of computing tasks. Thus many Actor may not work well on FPGAs. The user need to decide what should be implement as an actor to be executed on FPGAs. This can still be a difficult hardware/software partition problem. Abstracting the communication messages to NoC packages is another problem yet to explore.

References

[1] Dominik Charousset, Raphael Hiesgen, and Thomas C Schmidt. Caf-the c++ actor framework for scalable and resource-efficient applications. In *Proceedings of the 4th International Workshop on Programming based on Actors Agents & Decentralized Control*, pages 15–28. ACM, 2014.

[2] Raphael Hiesgen, Dominik Charousset, and Thomas C Schmidt. Manyfold actors: extending the c++ actor framework to heterogeneous many-core machines using opencl. In *Proceedings of the 5th International Workshop on Programming Based on Actors, Agents, and Decentralized Control*, pages 45–56. ACM, 2015.