Cymric: A Framework for Prototyping Near-Memory Architectures

Chad D. Kersey, Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, 30332
{cdkersey,sudha}@gatech.edu

Hyesoon Kim
College of Computing
Georgia Institute of Technology
Atlanta, GA, 30332
hyesoon@cc.gatech.edu

***Abstract*—Over the past two years, a prototyping infrastructure has grown up around CHDL; a C++ library for designing and simulating hardware, and HARP; a family of instruction set architectures for data parallel computing. Among the fruits of this effort are several processor cores, including a single-instruction-multiple-thread implementation of HARP called Harmonica and a variety of FPGA and simulation oriented memory system components. Collectively, these pieces form Cymric, an infrastructure for prototyping near-memory processor architectures. Here, we report on the current status of our efforts, enumerating the software and hardware components produced and discussing the tools we have found indispensable in their production.**

***Index Terms*—FPGA; GPGPU; SIMT; data parallel; processor near memory; C++**

I. Introduction

Processing-near-memory (PNM) is becoming an attractive solution to reduce energy consumption by reducing the volume of data movement in computing systems. While research in the 1990s addressed the idea of placing processing-in-memory (PIM), the continued evolution of Moore's Law and related architectural advances precluded the need for such architectures. However, the need of energy savings in the big-data era drives shifting from processing-centric computing to memory-centric computing. Understanding the energy and time behavior of PNM architectures is essential to the design of PNM architectures.

To explore the PNM architecture design space with detailed evaluations of time and energy behavior, we have been developing a PNM prototype, Cymric, along with a toolchain and ISA that allows us to explore various PNM architectures. Our new parameterizable ISA, HARP, has been designed to enable exploration of different architecture parameters. The HARP ISA resembles GPGPU ISAs but supports dynamic thread spawn, precise exceptions, and is deeply parameterized. We also developed the CHDL tool chain, a hardware design environment built around a C++ library enabling structured hardware generation and simulation. The layered design of CHDL allows the generation of architectural prototypes from relatively high-level microarchitectural descriptions. We have been developing a software stack that includes compilers, run-time system, and an FPGA-oriented implementation of HARP called Harmonica. Currently the Cymric system which is shown in Figure 2, consisting of a RISC core, several HARP cores, and an on-chip network, runs on Altera Stratix V system and a part of Cymric is currently being tested on a Micron EX-800 board.

II. Cymric System

1) HARP: The basis of Cymric is the HARP (Heterogeneous Architecture Research Prototype) family of instruction set architectures. HARP instructions sets are GPGPU-like; inherently SIMT, and any instruction stream can represent multiple threads of execution with independent register values for any period during which the program counters for all of the threads are the same. When program counters become different, *control flow divergence* has occurred, which is handled using the IPDOM stack-based algorithm. This is a parameterized family of architectures enabling unprecedented flexibility in the implementation of SIMT cores. The flexibility of HARP is in the number of design decisions left to the implementation, including register and datapath width, number of SIMD lanes, which instructions to implement, how many general-purpose registers and predicate registers to provide, and how to encode the instructions and data. To support these instruction sets, we have created a toolchain including an assembler, linker, and emulator, as well as a set of example programs. The HARP ISAs allow the use of a short string to describe the SIMD width, number of general-purpose and predicate registers, and instruction encoding used. Using this string, the toolchain targets the assembly and linking of input programs to the appropriate core type.

2) Harmonica: Current soft cores come in many sizes, from microcontrollers to superscalar, often providing some level of configurability. It is likely that the range of sizes demanded from current soft cores will also be seen in the demand for SIMT soft cores. To answer the need in the research community for soft GPGPU-like cores of many sizes and feature sets, we have developed Harmonica, a configurable SIMT soft-core with a parameterized instruction set architecture, to run GPGPU applications. The current Harmonica core is composed of a configurable number of SIMT lanes and has a 6-stage in-order-scheduled execution pipeline. This core is configurable in the sense that dimensions such as SIMD width and word size can be configured as well as the set of and

types of functional units present. This not only saves space by allowing the removal of unnecessary functional units, but also allows the development of functional units using vendor-provided IP for performance-critical operations like arithmetic.

A. HARP Compiler

The HARP compiler is implemented as a back end for the LLVM/Clang compiler system. The compiler currently supports a subset of OpenCL APIs as well as simple C/C++ programs, with growing support. In its current state, the compiler can translate some simple OpenCL kernels and C programs to HARP assembly. The majority of our test programs, however, are still written in assembly language.

B. Run-time System

The run-time system supports the following two main functionalities: (1) communication between the host core (MIPS) and accelerators (Harmonica). This includes triggering the execution at harmonica cores and also sending its results back to the host cores. (2) Multi-threaded program support: The HARP compiler generates implementations of procedures with no consideration of parallelism and it is the run-time system which spawns multiple SIMT threads. Thread spawn and register initialization prior to program execution are done by the run-time system.

C. CHDL

Harmonica is implemented using CHDL, an open-source C++-based hardware design library partially influenced by JHDL and Chisel [2] [1]. CHDL provides a way to describe the structure of hardware using a common programming language. This structural description can then be either simulated or used to produce netlists in several formats, including Verilog, which can then be parsed by traditional FPGA tool flows. Interfaces within CHDL designs and between them can be described in terms of structured, parameterized signals. This capability has proved invaluable in specifying our designs clearly, enabling e.g. a standard request/response interface format for processor-to-memory interfaces. The modularity enabled by CHDL has enabled a wide variety of memory format interfacing and memory networking functions to be implemented. These provide a high level of flexibility at the top level of design, reducing the assembly of the platform to a few lines of code describing the connectivity of devices attached to the memory system. The use of CHDL enables configurability at compile time with C++ template metaprogramming, ultimately improving performance during elaboration and simulation. The use of C++ also allows CHDL designs to be incorporated into other programs, allowing expansion of simulation and CAD features.

D. Simulation

The availability of FPGA prototypes does not obviate simulation. Given the size of our designs, hour-long compile times are not uncommon, so even with software simulation speeds in the tens of hertz, only runtimes of millions of cycles or more

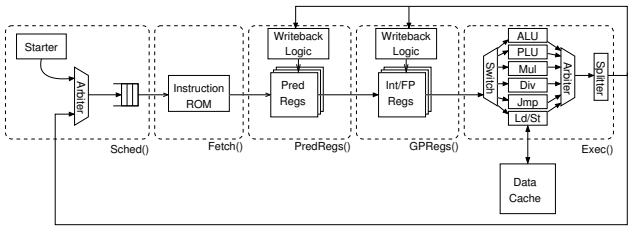

Fig. 1. Harmonica Architecture Pipeline

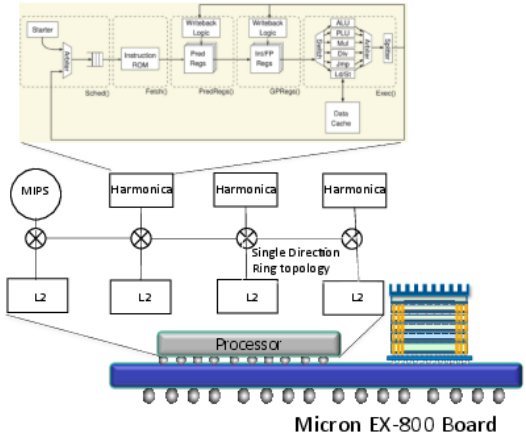

Fig. 2. Overview of Cymric

benefit from the speed of FPGA execution. Our simulation infrastructure is built around the CHDL component for the Structural Simulation Toolkit (SST) [3]. By integrating CHDL with SST, we have enabled the incorporation of CHDL designs into higher-level simulators of memory system components, enabling software validation of our prototype and faster design space exploration than would be available with only FPGA-based design evaluation.

E. FPGA Prototype

Modules interfacing the CHDL memory request/response format to the Avalon bus used by Altera's vendor-supplied memory controllers have been created, enabling a range of DRAM-connected prototypes to be created. In addition to this, there is an ongoing effort to add support for interfacing with Micron's Hybrid Memory Cube stacked DRAM devices.

ACKNOWLEDGEMENTS

This research was supported in part by Sandia National Laboratories and by Intel via the Intel Science and Technology Center for Cloud Computing (ISTC-CC).

REFERENCES

[1] J. Bachrach *et al.*, "Chisel: constructing hardware in a scala embedded language," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 1216–1225.

[2] P. Bellows and B. Hutchings, "Jhdl-an hdl for reconfigurable systems," in *FPGAs for Custom Computing Machines, 1998. Proceedings. IEEE Symposium on*. IEEE, 1998, pp. 175–184.

[3] A. F. Rodrigues *et al.*, "The structural simulation toolkit," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 4, pp. 37–42, 2011.