Cymric: Toward Flexible Exploration of Processor-Near-Memory Architectures

Abstract

This paper reports on the status of Cymric – an integrated compute-memory system architecture for massively parallel, irregular, data intensive applications. The major components are a customizable bulk synchronous data parallel architecture integrated with a stacked DRAM memory system and interconnected with a flexible network and package model. Given the nascent state of processor-near-memory (PNM) architectures, Cymric has adopted an approach based on the ability to rapidly develop and analyze a space of potential architectural solutions. The proposed environment is novel in the ability to integrate application binaries, cycle level timing models, and physical models (power, thermal) into a single simulation model. In particular, we focus on heterogeneous near-memory architectures employing multi-layer (stacked) DRAM technologies with in-stack or in-package multithreaded, data-parallel processing elements running both traditional and irregular CUDA and OpenCL workloads. The environment couples a customizable C++-based data parallel architecture synthesis/generation framework with microarchitecture level full system simulation models. We report on the current status, capabilities and future plans.

1. Introduction

Data movement is responsible for a large portion of the power budget of modern supercomputers. Techniques for mitigating this cost have focused on reducing the amount, frequency, and cost of long-distance communication, but this still does not address the basic problem of access to data in DRAM. For memory-intense applications with certain access patterns, the DRAM bandwidth is readily saturated.

A historical approach to solving this problem was PIM, processor-in-memory; placing compute logic on the same die as DRAM. This provides unhindered access to the entire row buffer, enabling unprecedented bandwidth between the on-memory processing elements and the RAM. This technique ultimately proved unfeasible in cost-oriented commodity DRAM processes. They were simply unable to provide fast enough MOSFETs to justify moving any amount of processing into the DRAM. This frustrating state of affairs received a perturbation when manufacturers began demonstrating successful 3D stacked memory products. These allow processors to be implemented in suitable processes and stacked under DRAM layers, either directly with through-silicon vias or in the same package.

When a disruptive technology such as 3D-stacked RAM appears, computer architects race to incorporate it into new architectures and characterize them. The near-memory processing enabled by these stacked RAMs is no different. This paper describes the set of architectures our group is exploring and the set of tools we are using to evaluate them.

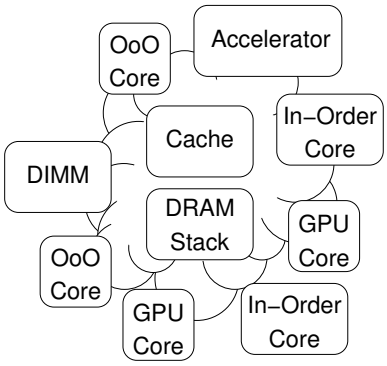

Figure 1: Thanks to the flexibility of the simulation infrastructure, the system architecture of Cymric can remain defined only as a set of components. Where these should located and in what topology they should be connected is the topic of future research.

This document is structured as follows: First we will describe the sets of architectures employing near-memory processing that we are exploring, the programming models exported by these machines, and the benchmark applications we intend to evaluate on them. In the second section, we will describe the evaluation and prototyping environment we are using to explore these architectures. We will conclude a quick look at planned future work.

2. Architectures

Figure 1 illustrates the architecture of a Cymric system; notable for its lack of structure. It is not yet clear at this early point in the development of PNM architectures exactly what their high-level structure should be. With a flexible simulation infrastructure we will be able to explore a variety of possible system-level architectures, including combinations of:

- Networks-on-chip.
- In-package integration with DRAM stacks.
- 3D-stacked integration with DRAM stacks.

Micron's Hybrid Memory Cube (HMC) [4] is one of the most promising emerging stacked DRAM technologies. FPGA products with HMC interfaces have recently been announced. [1] This combination of FPGA and stacked memory is an attractive target for prototype near-memory architectures. Prototypes built in such a manner would hardly qualify as having PNM architectures, but their designs would share much more with PNM architectures than they would with DDR-based systems.

3. Evaluation Infrastructure

The Structural Simulation Toolkit [11] is central to our evaluation methodology. SST provides a distributed simulation framework with a versatile, configurable memory system model, including VaultSim, a model of 3D stacked DRAM. To SST we couple MacSim [8], a trace-driven processor and GPU simulator, and CHDL, a `C++` hardware design and simulation library.

3.1. MacSim

MacSim [8] is a trace-driven and cycle-level heterogeneous architecture simulator. It models architectural behaviors such as detailed pipeline stages, multi-threading, and memory systems.

Micron's Hybrid Memory Cube (HMC) [4] is one of the most promising emerging technologies supporting the implementation of near memory architectures.

We can easily explore many points in our architectural design space including ones that exploit HMC as a near memory dram cache alongside an off-chip memory by varying MacSim and VaultSim configurations and assembling different memory hierarchies in the SST configuration.

We can configure MacSim and VaultSim on top of the SST framework so that it models architectures that are of interest. For example we can form a variety of topologies of VaultSims to get an idea of how different characteristics of HMC as opposed to current DDR-based memory can affect overall system performance when they are employed in different formation. Specifically we can organize more than one VaultSims so that one of them directly talks to MacSim (Host CPUs or GPUs) while others talk to each other.

Besides, we can vary the number of GPUs, the number of VaultSims, typical DRAM parameters, TSV width, link bandwidth, transaction size, etc to evaluate the sensitivity to parameters in different configurations so that we can get an idea how each and every parameter influences on performance as well as which is the optimal configuration and topology pair before diving into deep exploration.

3.2. HARP and CHDL

An open source `C++`-based hardware design library called CHDL [7] forms the core of the prototyping and gate-level simulation efforts. CHDL designs are created by writing a high-level description in `C++`. These descriptions can be incorporated into a program which, when compiled and run, produces any of a variety of outputs, which can include

- Value Change Dump (`.vcd`) files representing simulation output.
- Optimized synthesizable Verilog source code.
- Intermediate netlists suitable for circuit-level simulation or place and route.

A very brief CHDL example is shown in Figure 3 along with a variety of output formats.

Bits	Regs	Lanes	Gates	Crit. Path (Gates)
32	8	1	31811	99
32	16	1	36459	99
32	16	2	62295	99
32	16	8	217358	99
32	16	16	424117	99
64	16	8	544835	105
32	16	32	864220	99

Table 1: Harmonica implementation of HARP ISA with various parameters. All of these numbers represent integer-only cores with gshare branch prediction.

Our ongoing effort to produce a heterogeneous architecture research prototype (HARP) has succeeded in producing a set of instruction set architectures collectively known as the HARP ISA, an assembler/linker/emulator tool, and a parameterized in-order, single-issue, register transfer level implementation called Harmonica. The HARP ISA is parameterized fully predicated SIMD instruction set intended to be used in a wide range of processor designs. Because of the parameterized nature of the HARP ISA and the Harmonica implementation thereof, it is versatile, being able to fill multiple roles in the so-called compute hierarchy.

Harmonica is a pipelined processor that issues instructions in-order and allows them to complete out-of-order. It is built in a modular fashion so that functional units implementing various sets of instructions can be added or omitted as desired. The minimum pipeline depth is four, but depending on the number of functional units installed and the number of outstanding instructions each might have, the total instructions in flight can be much higher. Table 1 provides an example of the range of core sizes and parameters possible with Harmonica. These are on the small side, lacking floating point units and having relatively small register files. In the figure, "gates" is the total post-optimization number of nand gates, D flip-flops, and inverters in the design.

This work does not exist in isolation from the higher-level simulation work. Using a set of generic interface functions, a CHDL processor model has been demonstrated running within SST, using SST's MemHierarchy interface as its only memory system. This integration will allow a seamless transition between high-level trace-driven models and low-level synthesizable hardware models and has the added advantage, since SST is a distributed simulation framework, of allowing multiple CHDL processor implementations to be simulated within SST in parallel.

The eventual goal of the HARP project is to run unmodified GPU kernels generated from the PTX intermediate form as well as non-SIMD benchmarks compiled from C. For now, all of the applications are kernel benchmarks written directly in HARP assembly language.

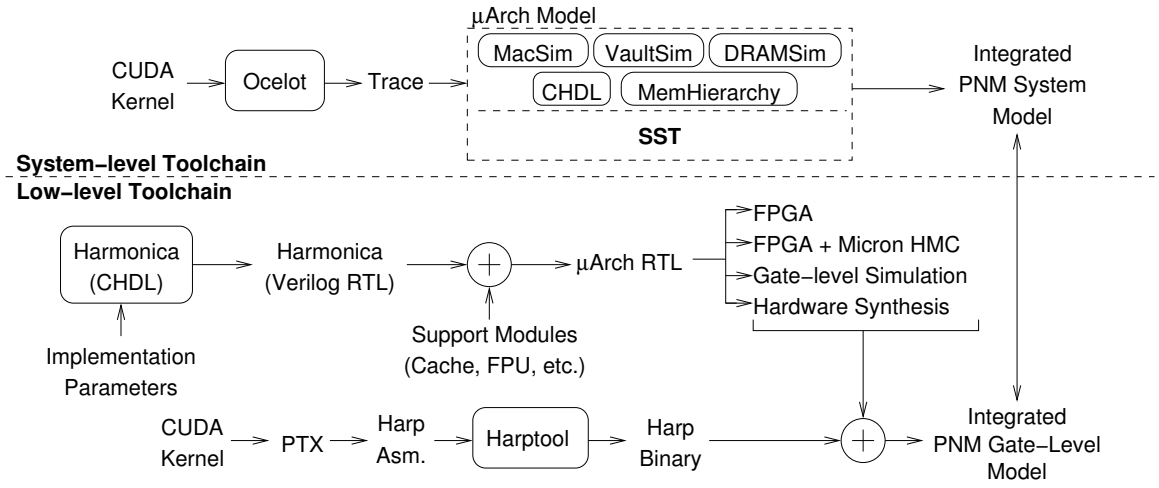

Figure 2: The authors' software infrastructure consists of an RTL CPU design with an enabling hardware design library, a high-level simulation flow, and a low-level implementation flow capable of targeting FPGAs.

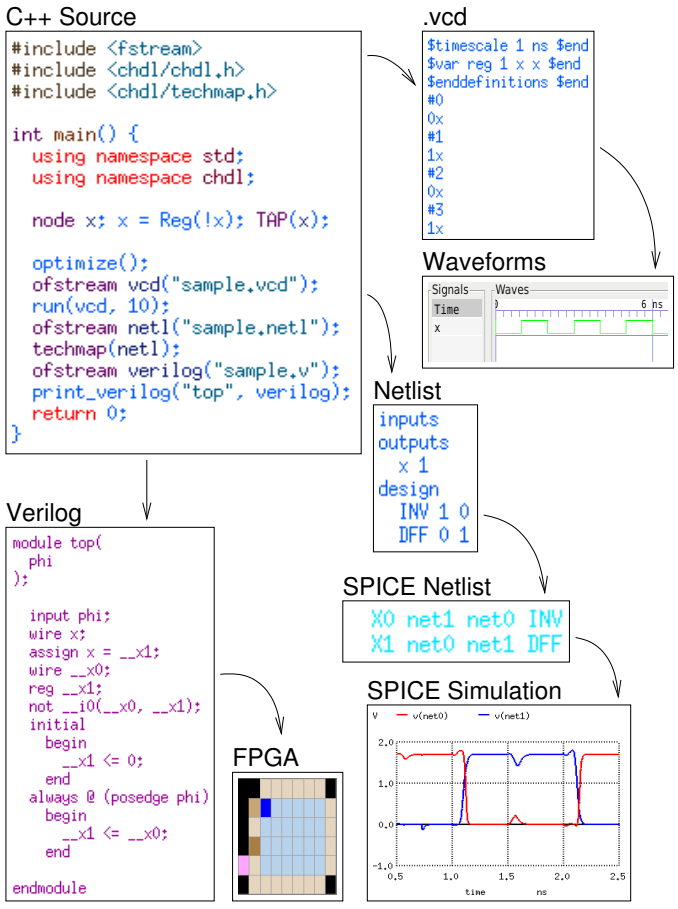

Figure 3: CHDL is a versatile C++ hardware design library supporting a wide range of output formats.

Harmonica Core	
FPGA:	Stratix III; 113,600 LEs
Operating Freq:	90 MHz
RAM(data), ROM(inst):	1024KB, 1024KB
Regs:	8 32-bit Regs
FPGA Utilization:	4% (3,661 / 113,600 ALUTs)

Full system	
Operating Freq:	62.5 MHz
L1 / L2 cache:	16KB / 128KB
Combinational ALUTs:	13% (14,446 / 113,600 ALUTs)
Total block memory bits:	27% (1,505,792 / 5,630,976)

Table 2: The Harmonica core in its default configuration takes up very little FPGA real estate.

3.3. FPGA prototype

One of the goals of the evaluation toolchain is to have the ability to quickly prototype some subset of the systems we are exploring on an FPGA. The tools we have developed and adopted allow us to quickly tune system parameters and implement prototypes on FPGA boards and measure their performance. We can use this process to guide parameter selection in other aspects of architecture simulation.

Our first FPGA board prototype is built around the Harmonica core. Synthesizable Verilog code for the core was generated by compiling the Harmonica source with GCC and linking against the CHDL library. This code was then compiled for an Altera Stratix III FPGA using Altera's Quartus II tool suite. Simple test applications written in HARP assembly like array sum, sieve of Eratosthenes and bubble sort were used to verify the design. The logic consumption of this simple design was tiny as can be seen in Table 2. With this flexible toolchain and a customizable ISA and core we are now able to quickly iterate, rapidly implementing modifications to our instruction set, core, and system architecture. For example, adding support for new instructions, new types of branch predictor, different floating point unit designs, new cache designs and memory controllers, etc. was relatively easy.

Once the Harmonica core was running on the FPGA we integrated it with more complex modules to create a more

complete system. A 2-level non-blocking write-back cache written in Verilog was created and interfaced with the CHDL-based Harmonica code. This cache is also parameterized. Its size and number of ways, can be changed, and it can be used in multi-level configurations. After integrating the cache we began integrating a DDR2 memory controller generated with Altera's MegaWizard tool with the core and cache. The logic consumption of this design can be seen in Table 2.

In summary, we created a system-level prototype consisting of a highly configurable core, cache and DDR2 memory controller. Each of these components was individually tested before integration. This flow will be helpful for future work in which we plan to use it to explore PNM architectures, e.g. by interfacing the FPGA prototype with Micron's Hybrid Memory Cube. We will only have to obtain a new memory controller IP either from the vendor or add changes to an existing one. This will also us to do various studies like what configuration of the logic layer will be best suited for this type of memory architecture and examine critical paths.

4. Future Work

The work remaining is non-trivial and includes:

- Evaluate system architectures and establish a baseline
- Expanding the feature set of Harmonica
- Performance optimization for both FPGA and silicon targets
- Building a capable PTX to HARP Translator
- Implementing a custom memory controller
- Adding support for HMC to the cache model

5. Related Work

PIM systems have been researched intermittently for a long time now. Most of the work involved major modifications to the DRAM architecture and they used simulators to shows the benefit. Many designs also used SRAM for the near memory [3]. Among the initial works, one version developed by [2] built PIM as a SIMD array of ALUs next to an SRAM. [3] was a popular PIM based architecture where they simulated using RSIM and also created a first prototype. The fabricated prototype ASIC showed encouraging results but again it is time consuming to redesign prototypes for new architecture. It also requires a significant change to DRAM architecture. We did not find much literature which talked about providing enough flexibility to explore different architectures of not just the PIM node but of the overall system. They were either more focused on creating prototypes based on a fixed architecture [3][6][10] or its advantage for certain types of applications at a high level using simulations [9][12][5].

References

[1] A. Corporation. (2013) Hybrid memory cube. Available: http://www.altera.com/technology/memory/serial-memory/hybrid-mem-cubes/mem-cubes.html

[2] M. Gokhale, B. Holmes, and K. Iobst, "Processing in memory: the terasys massively parallel pim array," *Computer*, vol. 28, no. 4, pp. 23–31, 1995.

[3] M. Hall *et al.*, "Mapping irregular applications to diva, a pim-based data-intensive architecture," in *Supercomputing, ACM/IEEE 1999 Conference*, 1999, pp. 57–57.

[4] J. Jeddeloh and B. Keeth, "Hybrid memory cube new dram architecture increases density and performance," in *VLSI Technology (VLSIT), 2012 Symposium on*. IEEE, 2012, pp. 87–88.

[5] J.-Y. Kang, S. Gupta, and J.-L. Gaudiot, "An efficient pim (processor-in-memory) architecture for blast," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 1, 2004, pp. 503–507 Vol.1.

[6] Y. Kang *et al.*, "Flexram: toward an advanced intelligent memory system," in *Computer Design, 1999. (ICCD '99) International Conference on*, 1999, pp. 192–201.

[7] C. Kersey. (2013, Jan.) Announcing chdl. Available: http://www.cdkersey.com/wordpress/?p=26

[8] H. Kim *et al.*, *MacSim: A CPU-GPU Heterogeneous Simulation Framework User Guide*, 2012.

[9] M. Lanuzza, M. Margala, and P. Corsonello, "Cost-effective low-power processor-in-memory-based reconfigurable datapath for multimedia applications," in *Proceedings of the 2005 international symposium on Low power electronics and design*, ser. ISLPED '05. New York, NY, USA: ACM, 2005, pp. 161–166. Available: http://doi.acm.org/10.1145/1077603.1077645

[10] D. Patterson *et al.*, "A case for intelligent ram," *Micro, IEEE*, vol. 17, no. 2, pp. 34–44, 1997.

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

[12] N. Venkateswaran *et al.*, "Memory in processor: a novel design paradigm for supercomputing architectures," in *Proceedings of the 2003 workshop on MEmory performance: DEaling with Applications , systems and architecture*, ser. MEDEA '03. New York, NY, USA: ACM, 2003, pp. 19–26. Available: http://doi.acm.org/10.1145/1152923.1024298