Cymric

A Framework for Prototyping Near-Memory Architectures

Chad D. Kersey[1], Hyesoon Kim[2], Sudhakar Yalamanchili[1]

The rest of the Cymric team:
Nathan Braswell, Jemmy Gazhenko, Prasun Gera, Meghana Gupta,
Hyojong Kim, Joo Hwan Lee, Troy O'Neal, Rahul Rajan

[1] Georgia Tech School of ECE; {cdkersey,sudha}@gatech.edu
[2] Georgia Tech College of Computing; hyesoon@cc.gatech.edu

Summary and Outline

Summary

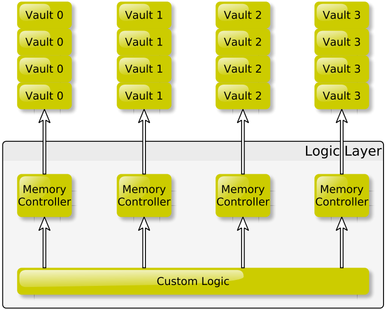
We have built Cymric, which:

- Enables simulations and prototyping of novel heterogeneous architectures, such as those employing stacked DRAM.
- Uses C++ as a common, unified description language.
- Includes prototype logic layer model for stacked memory systems which runs on the Micron EX-800 HMC/FPGA board.

Outline

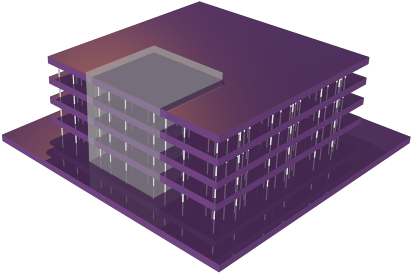
- Background
- The Cymric Infrastructure
- Target Design
- Implementation Platform (EX-800)
- Project Status

Background: Stacked DRAM Architectures



Vault-based stacked DRAM.

- HMC organization: multiple channels (*vaults*).
- One bank per layer per vault.



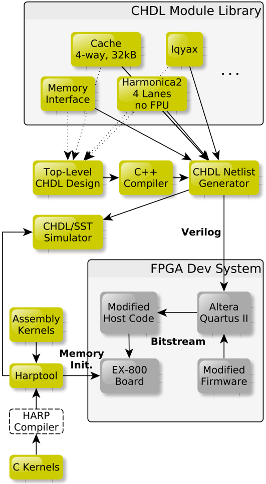
One vault highlighted.

Hybrid Memory Cube(Micron):

- 16 or 32 vaults.
- 4 or 8 DRAM layers.
- 4 160Gb/s (each way) links

Also: HBM, WideIO

Cymric: Overview



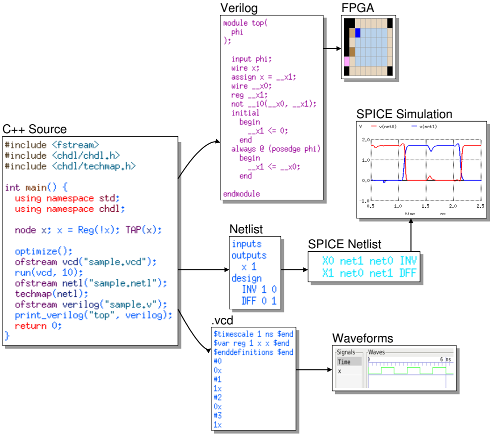
Overview of Cymric.

To evaulate stacked DRAM architectures employing near-memory computation, we have developed Cymric, an open source[1] environment including:

- CHDL hardware design library.
- Caches, scratchpads, and ROMs using CHDL's memory interface format.
- HARP SIMT instruction sets, Harmonica 2 implementation.
- HARP LLVM back-end and run-time library (running OpenCL-like kernels).
- Memory interfaces for Altera's Avalon and Pico's HMC controller.

[1] Except pieces marked in gray.

Implementation Environment: CHDL



```
#include <fstream>
#include <chdl/chdl.h>
#include <chdl/techmap.h>

int main() {
  using namespace std;
  using namespace chdl;

  node x; x = Reg(!x); TAP(x);

  optimize();
  ofstream vcd("sample.vcd");
  run(vcd, 10);
  ofstream netl("sample.netl");
  techmap(netl);
  ofstream verilog("sample.v");
  print_verilog("top", verilog);
  return 0;
}
```

```
module top(
  phi
);

  input phi;
  wire x;
  assign x = __x1;
  wire __x0;
  reg __x1;
  not __i0(__x0, __x1);
  initial
    begin
      __x1 <= 0;
    end
  always @ (posedge phi)
    begin
      __x1 <= __x0;
    end

endmodule
```

```
inputs
outputs
  x 1
design
  INV 1 0
  DFF 0 1
```

```
  X0 net1 net0 INV
  X1 net0 net1 DFF
```

```
$timescale 1 ns $end
$var reg 1 x x $end
$enddefinitions $end
#0
0x
#1
1x
#2
0x
#3
1x
```

CHDL output can be:

- Synthesized to netlist.
- Simulated at logic level.
- Run on FPGA w/ vendor tools.

- Designs in CHDL are netlist generators.
- Ultimately generating basic gates and memory arrays.
- All written in standard C++.

Parameterizability

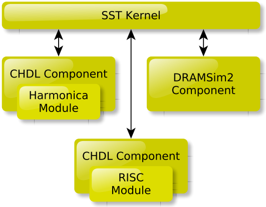
HARP's configurability is enabled by C++'s support for template metaprogramming, brought into the hardware domain by CHDL.

Implementation Environment: CHDL

CHDL provides:

- Structured signals.
- Loadable netlists.

The CHDL *module library* provides a set of pre-built IP-cores along with HTML documentation of the interfaces.



CHDL/SST integration allows simulation of FPGA-ready designs.

Modules with CHDL memory interfaces can be simulated in the Structural Simulation Toolkit[1].

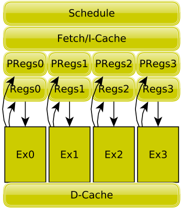
- Harmonica cores in FPGA clocked at 125MHz, with 10000 seconds of static overhead.
- Simulation environment runs at about 100Hz, with 100 seconds of static overhead.
- For under 1M cycles, simulation is faster.
- Simulations allow arbitrary configuration of the memory system.

[1] Rodrigues, et al. 2011 ACM SIGMETRICS

Cymric: HARP Instruction Sets

SIMT architectures:

- Many threads in flight.
- Threads grouped into warps.
- SIMD parallelism within warps.
- Many warps in flight; few hazards, less forwarding.
- Pipeline parallelism between warps.
- Lanes enabled using *active mask*.
- When control flow paths diverge, serialize.
- Due to TLP and large number of warps: high MLP, mem. latency tolerant.



4-lane SIMT pipeline.[1]

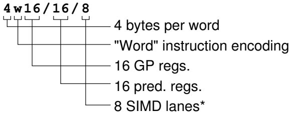
HARP (our SIMT implementation)

- Architecture for a Heterogeneous Architecture Resarch Prototype.
- GPGPU-like. Also provides dynamic thread/warp spawn instructions.

[1] Based on image from nVidia.

Cymric: HARP Instruction Sets

Cymric processor options include Harmonica[1], supporting the HARP family of SIMT instruction sets.



*ignored by assembler/linker

HARP Architecture ID String

- To the assembler and linker, a short string identifies the ISA.
- The fields of the ArchID correspond to parameters of the Harmonica implementation.

Tools available for HARP include:

- HARPTool assembler/linker/emulator.
- LLVM compiler back-end (in progress).

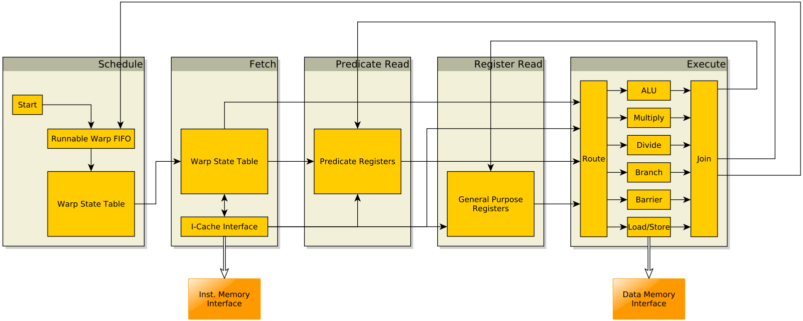
Parameters

HARP ISAs are fully parameterized, including:

- Number of registers.
- Number of SIMT lanes.
- Available instructions and functional units.

[1] Kersey et al. FCCM 2014 (poster)

Implementation Environment: Harmonica

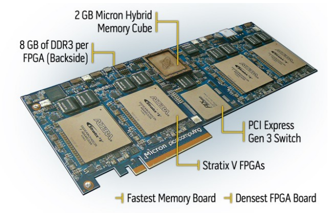


Warps	Lanes	Gates
16	8	285005
16	32	1078223
32	8	328092
32	32	1145118

Harmonica sizes with various parameter values.

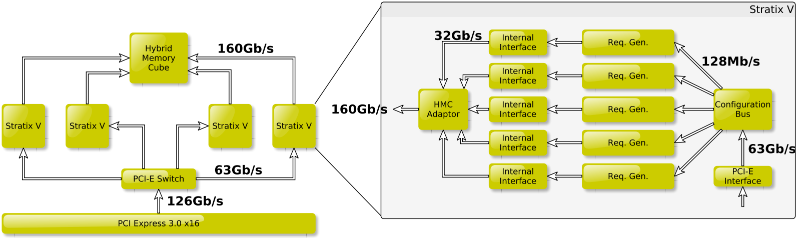
- Custom SIMT design using HARP ISA.
- Warps, lanes, registers: parameters.
- Simple design (1600 lines of C++).
- Doesn't extract ILP within threads.
- Optional floating point support.

Implementation Platform: EX-800



EX-800 HMC board. [1]

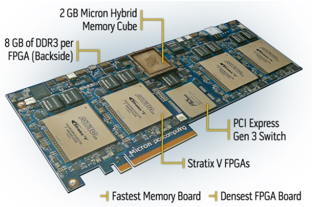
- Running prototype on Micron EX-800 HMC evaluation board.
- 4 Altera Stratix V FPGAs, each with a 160Gbps link, internally split into five 32Gbps links.
- We build our prototype by modifying existing FPGA firmware.



Relevant components annotated with unidirectional bandwidths.

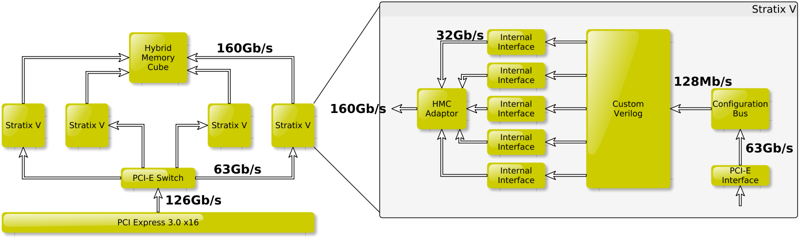
[1] Source: `picocomputing.com`

Implementation Platform: EX-800



EX-800 HMC board. [1]

- Running prototype on Micron EX-800 HMC evaluation board.
- 4 Altera Stratix V FPGAs, each with a 160Gbps link, internally split into five 32Gbps links.
- We build our prototype by modifying existing FPGA firmware.

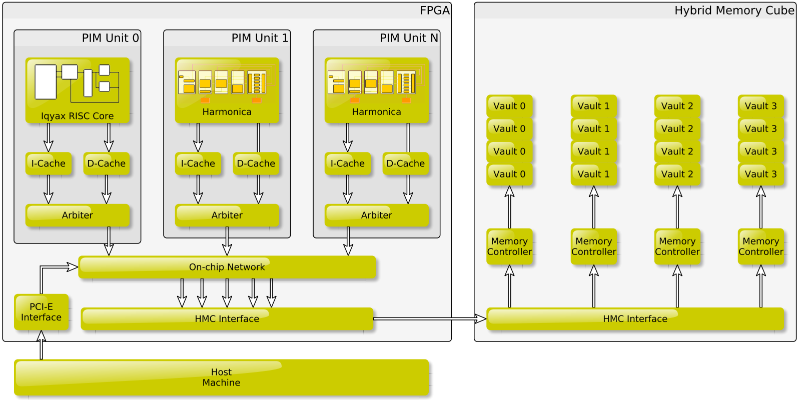


Relevant components annotated with unidirectional bandwidths.

[1] Source: `picocomputing.com`

Prototype: Target Design

Our planned target design:



- Current capability: running single MIPS or Harmonica instance on the EX-800 at 125MHz.
- Next: fully populating the FPGA with cores and running applications; attempting to reach 250MHz.

Prototype: Memory Interface

The crucial modules for prototyping are the memory interface adaptors, converting between the following:

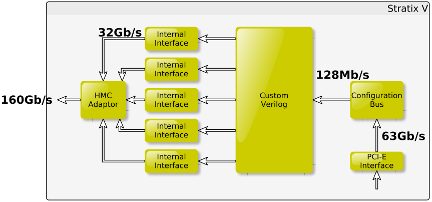
- CHDL `mem_port`.
  - Limited to 1 beat.
  - Masked writes.
  - Parameterized.
- Pico bus. (EX800 config)
  - Synchronous.
  - No flow control.
- EX-800 HMC interface. (x5) [1]
  - 128 bits wide.
  - 6-bit tag.
  - 1 to 8 beats.
  - No masked writes.

Four clock domains:

- Configuration bus. (4MHz)
- HMC interface TX. (250MHz)
- HMC interface RX. (250MHz)
- Half-RX core clock. (125MHz)

CHDL Asynchronous FIFO

- Fully parameterized.
- 59 lines. (150 typ. for vl.)



[1] OpenHMC is being considered as a fully open-source alternative.

Cymric: Status

Designs simulated:

- Harmonica cores in various configurations attached to simulated memory systems, running relational benchmarks (join, select, and project), key/value store, graph BFT, FFT, and matrix multiply.
- Heterogeneous Harmonica/MIPS systems with shared memory.
- Multi-core MIPS systems using SST's simulated coherent caches.

Designs FPGA proven:

- RISC and Harmonica with scratchpad memories on Cyclone II.
- RISC/Harmonica together with external DDR 2 on Stratix 3.
- Harmonica multi-cores with external DDR 3 on Stratix V.

Designs and programs run on the EX-800 so far:

- Scratchpad memory attached to configuration bus.
- One RISC core attached to scratchpad memory attached to configuration bus.
- One small (4 lane, 4 warp) Harmonica core running simple single-threaded test program attached to single HMC interface.

Cymric: Status

Project	Code	Test	Doc	Description
HARP Universe				
harptool	✓	⚠	✓	Assembler/linker/emulator.
compiler	⚠	⚠	⚠	LLVM back-end.
benchmarks	✓	✗	⚠	Assembly language apps.
CHDL Universe				
chdl	✓	⚠	⚠	Hardware design framework/library.
chdl-stl	✓	✓	⚠	Template library for CHDL.
chdl-module	✓	✓	⚠	Use netlists as sub-modules.
iqyax	✓	⚠	✗	MIPS-like in-order core.
harmonica	✓	⚠	✗	SIMT processor core.
harpsim	✓	⚠	✗	Harmonica simulator.
cache	✓	⚠	✗	Set associative cache.
chdl-sst	✓	⚠	✗	Use CHDL in SST framework.
mem-iface	⚠	⚠	✗	Memory interface adaptors.

✗Not started. ⚠In progress. ✓Finished.

Thank You

Questions?