18-643 Lecture 14: CoRAM Memory Architecture or FPGA Computing

James C. Hoe
Department of ECE
Carnegie Mellon University

Housekeeping

- Your goal today: 2nd try at transition to “seminar” mode for the 2nd half of the course
- Notices
  - 10/18: Midterm
  - 10/20: Marie Nguyen Lecture: Partial Reconfig.
  - 10/21 (due noon): Lab 3, Proposal slides
  - 10/{25, 26, 27}: proposal presentations
- Readings
Moore’s Law for FPGA

What can make programming easier (and more portable)

• High-level programming abstraction
  – general-purpose robs efficiency
  – special-purpose restricts applicability

• Platform virtualization
  What does a FPGA programmer need to know beyond the fabric boundary?

• Infrastructural library
  What is “C-library” and “system calls” for FPGAs?
2-Part Outline

- CoRAM “Architecture”
- CoRAM++ “API”

Computing on Bare-Metal FPGA

User responsible for:
- application and data
Computing on Bare-Metal FPGA

User responsible for:
- application and data
- platform I/O and memory interfaces
- data distribution
- sequence and control logic

Kinder-Gentler FPGA

- Hard logic for memory subsystem
Kinder-Gentler FPGA

- Hard logic for memory subsystem
- Hard logic for data distribution (NoC)
- “Software”-managed memory hierarchy

CoRAM FPGA Architectural View

- Hard logic for memory subsystem
- Hard logic for data distribution (NoC)
- “Software”-managed memory hierarchy
- Simple, portable abstraction for user
**Specification Syntax**

```verilog
module verilog_top(…);
    cohandle c0 = get_sram("c0");
    c_coram_write(c0, sram_address, global_address, size);
endmodule
```

**Control Actions**

- **Edge Memory**
  - `coram_write(coh coram, void *offset, void *memaddr, int bytes);`
  - `collective_write(coh coram, void *offset, void *memaddr, int bytes);`
  - `coram_copy(coh src, coh dst, void *srcoffset, void *dstoffset, int bytes);`

- **2 CoRAMs (concatenated)**
  - `collective_write(coh coram, void *offset, void *memaddr, int bytes);`

- **2 CoRAMs (interleaved)**
  - `collective_write(coh coram, void *offset, void *memaddr, int bytes);`
### Design Case Study: MMMult

#### MMM in hardware
- N compute engines (1 per row of matrix)
- custom data network

![Diagram of MMMult](image)

#### Mmult with CoRAM

**Control Thread Program**

```c
void ctrl_thread() {
    for (j = 0; j < N; j += NB)
        for (i = 0; i < N; i += NB)
            for (k = 0; k < N; k += NB) {
                c_fifo_read(...);
                for (m = 0; m < NB; m++) {
                    c_collective_write(ramsA, m*NB, A + i*N+k * m*N, NB*dsz);
                    ...
                    c_fifo_write(...);
                }
            }
}
```

![Diagram of CoRAM](image)
**Beneath the Abstraction**

Architecture

Microarchitecture

CoRAM FPGA

Control Unit

Outer Unit

Fabric

Control Unit

Outer Unit

Fabric

control thread programs

Bulk Data Distribution (Network-on-Chip)

Memory Interfaces (DRAM or Cache)

---

**Control Threads: Soft or Hard?**

**Soft methods**
- only rely on RL fabric
- must share logic with app
- efficiency/perf depends on quality of C-based synthesis and compilation

**Hard method**
- dedicated silicon for cores
- high perf/efficiency
- consumes area whether used or not
### RTL Prototyping on FPGA

**Diagram:**
- **I/O Subsystem and Drivers**
- **DRAM Interfaces**
- **Caches**
- **NoC and Clusters**
- **MMMultiply Compute Elements**

### CoRAM++ “API”
Somewhere we changed our mind

- Architecture design is an exercise in compromise
  i.e., enough but not too much
- Do we really need an FPGA “architecture” just for computing?

CoRAM “Architecture” Revisited

- Must have hardened memory interface and NoC
- I like the decoupled computation paradigm
- Do I need to harden CoRAM?
  - fixed set of commands
  - isolate logic to one-side
CoRAM Reconstructed as “APIs”
[Gabe Weisz, 2015]

CoRAM++
Data-Structure-Specific Interfaces

<table>
<thead>
<tr>
<th>Control Thread Commands</th>
<th>Hardware Kernel Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>write_stream()</td>
<td>FIFO-style ports</td>
</tr>
<tr>
<td>read_stream()</td>
<td></td>
</tr>
<tr>
<td>check_stream()</td>
<td></td>
</tr>
<tr>
<td>write_array()</td>
<td>SRAM or FIFO-style ports</td>
</tr>
<tr>
<td>read_array()</td>
<td></td>
</tr>
<tr>
<td>write_sequence()</td>
<td></td>
</tr>
<tr>
<td>read_sequence()</td>
<td></td>
</tr>
<tr>
<td>check_array()</td>
<td></td>
</tr>
<tr>
<td>list_to_stream()</td>
<td>FIFO-style ports</td>
</tr>
<tr>
<td>list_from_stream()</td>
<td></td>
</tr>
<tr>
<td>merge_lists()</td>
<td></td>
</tr>
<tr>
<td>check_list()</td>
<td></td>
</tr>
</tbody>
</table>
CoRAM “Classic” Linked List to Stream

CoRAM++ Linked List to Stream

Terasic DE4
CoRAM for Cache Coherent Shared Memory (sneak peek)

FPGA as First-Class, Equal Partner
Thread-API for CPU-FPGA Interaction

FPGA’s outward visible personality
Parting Thoughts

• Performance is hard and efficiency is harder
  It is the same for SW . . . .

• CoRAM wants to avoid needless “hard” in common-denominator infrastructure
  – standardizing common API helps with programmability AND portability
  – standardization and specialization do not conflict

• New work motivated by fine-grain CPU-FPGA collaboration in accelerating irregular parallelism

18-643 Review Template: Summary

• What to look for in a paper/presentation
  – what is the question/problem?
  – why is this question/problem important?
  – why is this question/problem hard?
  – what is the answer/solution offered by the paper?
  – what is new, novel about the answer/solution?
  – how does the paper argue/support the answer/solution is correct/good?

• If you don’t know the answers, you didn’t “read” the paper, OR, the paper/presentation is bad
18-643 Review Template: Critic

• What to think about while reading
  – soundness: is the paper's answer/solution correct/good?
  – impact: is the paper's answer/solution important?
  – novelty: does the paper teach something new and not obvious?
  – strengths: what makes the paper stand out?
  – weaknesses: what could be improved?

• If you don’t know the answers, you didn’t “read” the paper
  
  Don’t accept what a paper says unless you agree