18-643 Lecture 14:
CoRAM Memory Architecture for 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
  – 10/21 (due noon): Lab 3, Proposal slides
  – 10/{25, 26, 27}: proposal presentations

• Readings
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

ctrlthread() {
    cohandle c0 = get_sram("c0");
    1. c_coram_write(c0, sram_address, global_address, size);
    2. c_fifo_write(1);
}

module verilog_top(…);
    coram c0 (/*ports*/);
    cofifo f0 (/*ports*/);
    …
endmodule

Control Actions

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);
collective_write(coh coram, void *offset, void *memaddr, int bytes);

Edge Memory

Single CoRAM

2 CoRAMs (concatenated)

Src CoRAM

Dst CoRAM

2 CoRAMs (interleaved)
**Design Case Study: MMMult**

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

\[ C = AB \]

<table>
<thead>
<tr>
<th>A SRAM</th>
<th>B</th>
<th>C SRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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(...);
            }
}
```

**MMMutl 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(...);
            }
}
```
Beneath the Abstraction

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

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
  - insolate logic to one-side
CoRAM Reconstructed as “APIs”  
[Gabe Weisz, 2015]

CoRAM++  
Data-Structure-Specific Interfaces

**Control Thread Commands**
- `write_stream()`
- `read_stream()`
- `check_stream()`

**Hardware Kernel Ports**
- `FIFO-style ports`
- `SRAM or FIFO-style ports`
- `FIFO-style ports`

---

<table>
<thead>
<tr>
<th>Stream Interface</th>
<th>Array Interface</th>
<th>Linked List Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>write_stream()</code></td>
<td><code>write_array()</code></td>
<td><code>list_to_stream()</code></td>
</tr>
<tr>
<td><code>read_stream()</code></td>
<td><code>read_array()</code></td>
<td><code>list_from_stream()</code></td>
</tr>
<tr>
<td><code>check_stream()</code></td>
<td><code>write_sequence()</code></td>
<td><code>merge_lists()</code></td>
</tr>
<tr>
<td></td>
<td><code>read_sequence()</code></td>
<td><code>check_list()</code></td>
</tr>
</tbody>
</table>
CoRAM “Classic” Linked List to Stream

CoRAM++ Linked List to Stream
CoRAM for Cache Coherent Shared Memory (sneak peek)

FPGA as First-Class, Equal Partner

Diagram:
- CPU
- CPU
- FPGA
- Memory Bus / Interconnect
- DRAM
- DRAM
- Core
- L2
- Core
- L2
- FPGA Fabric
- LLC Cache
- DRAM
Thread-API for CPU-FPGA Interaction

- Memory
- Control threads
- PCIe

Thread-API for CPU-FPGA Interaction

- FPGA’s outward visible personality
- CPU
- Memory
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 teaches something new and not obvious?
  – strengths: what makes the paper standout?
  – 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