Coyote

A Comprehensive Study of Coyote Memory System

Yuan Luo
Supervisor: Atsushi Koshiba, Jiyang Chen
Chair of Distributed Systems and Operating Systems
https://dse.in.tum.de/

17.10.2022 – 20.04.2023

Motivation: Research context

- Sharing FPGA between different user logics is difficult due to potential I/O conflicts

- Programming FPGA applications targeting different I/O devices remains challenging due to insufficient support

- An easy-to-use FPGA I/O acceleration framework is needed (**FAIR**)

State of the art

- AmorphOS
  - Focuses only on memory I/O

- Coyote
  - Supprots memory I/O and network I/O
  - Lacks efficient I/O transaction scheduling

We need a smarter FPGA I/O scheduler for different I/O devices

Problem statement

A survey on Coyote to implement FAIR I/O scheduler is required

- How Coyote implements virtual FPGA and virtual memory at the hardware level
- How Coyote memory system works
- How to integrate an I/O scheduler into Coyote memory system

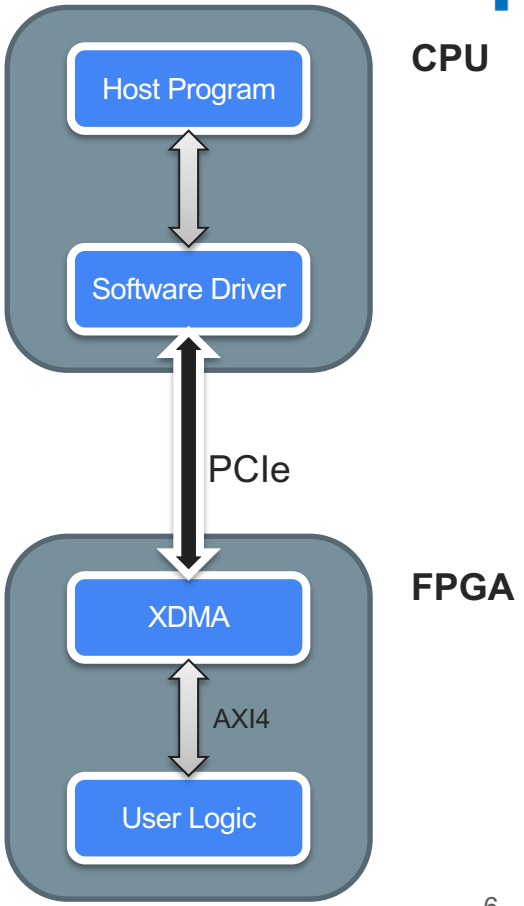
Outline

- Coyote Architecture
- Coyote Memory System
- Debugging Coyote
- Evaluation

Communication between CPU and FPGA

- Driver connects host program with PCIe

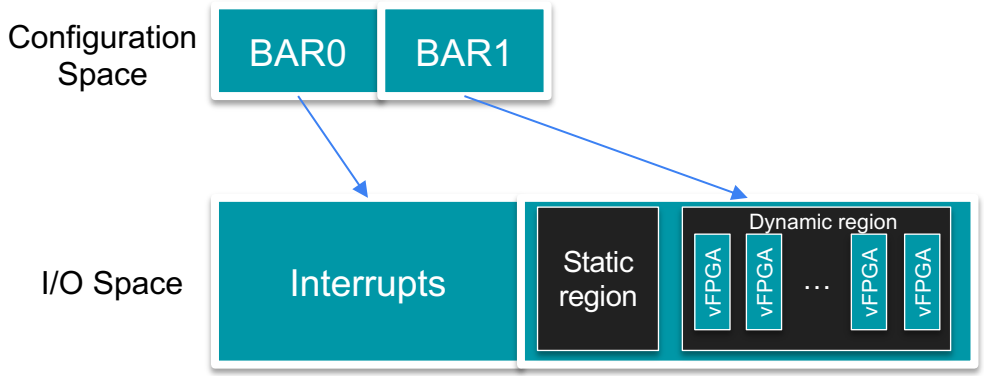
- XDMA configures PCIe and communicates with user logic



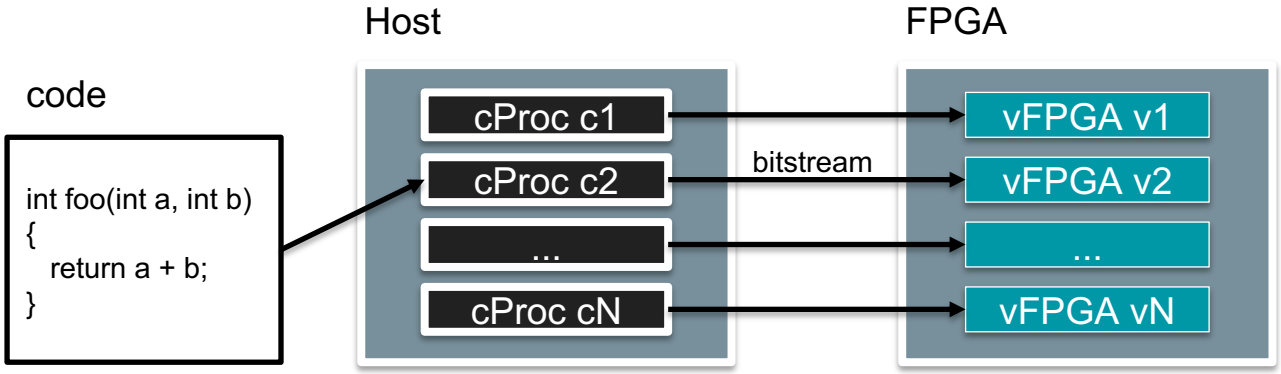
Static region and dynamic region

PCIe memory space contains three address spaces:

- I/O address space
- Memory address space
- Configuration address space



Coyote Process (cProc) and vFPGA



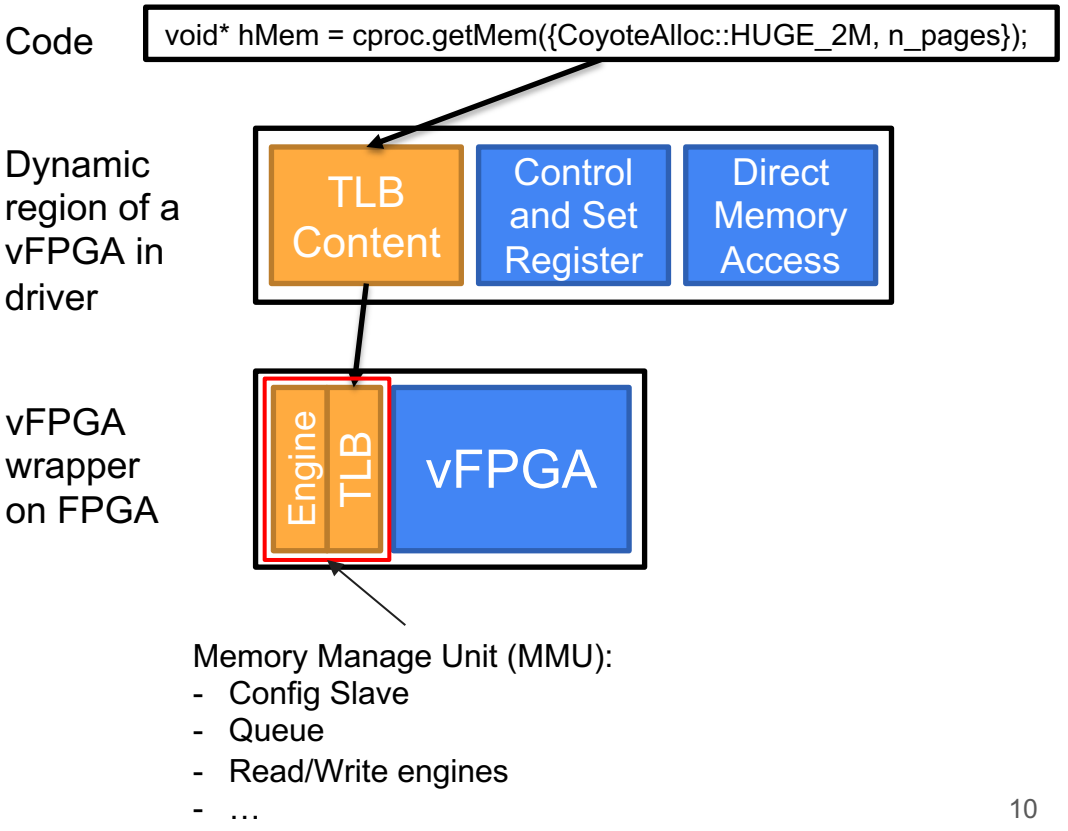
- Program code is first synthesized into bitstream
- cProc loads the bitstream and reconfigures vFPGA

Outline

- ~~Coyote Architecture~~
- Coyote Memory System
- Debugging Coyote
- Evaluation

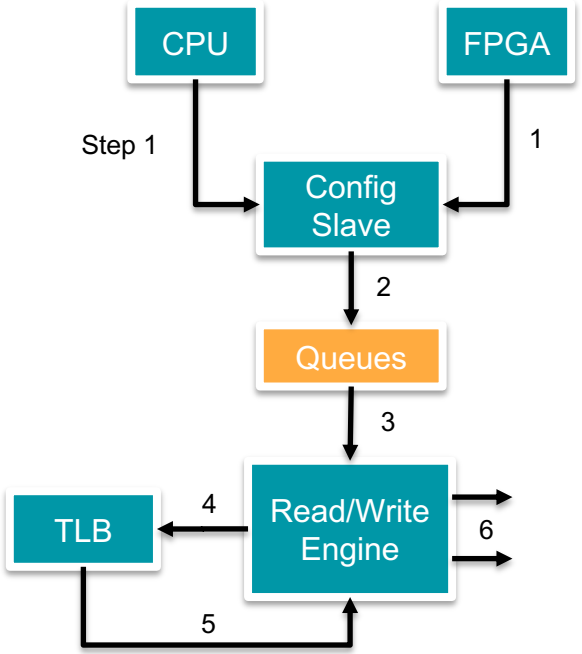
Virtual address

- A strategy similar to CUDA
- The page size can be either 4 KB or 2 MB



Memory management

1. Collect memory access requests
2. Process and distribute requests
3. Forward memory access requests
4. Refer to TLB for possible address translation
5. Return physical address or issue error
6. Issue memory access

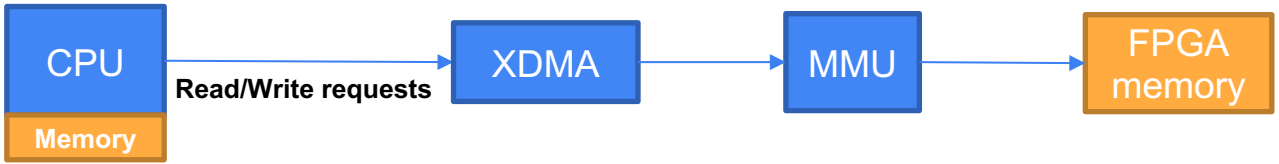


Outline

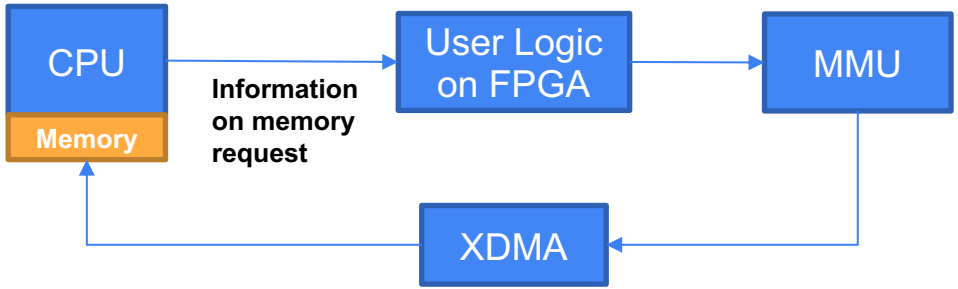
- ~~Coyote Architecture~~
- ~~Coyote Memory System~~
- ~~Debugging Coyote (skipped)~~
- Evaluation

Evaluation

- Host-initiated data transfer
  - A loopback data transfer between host and FPGA



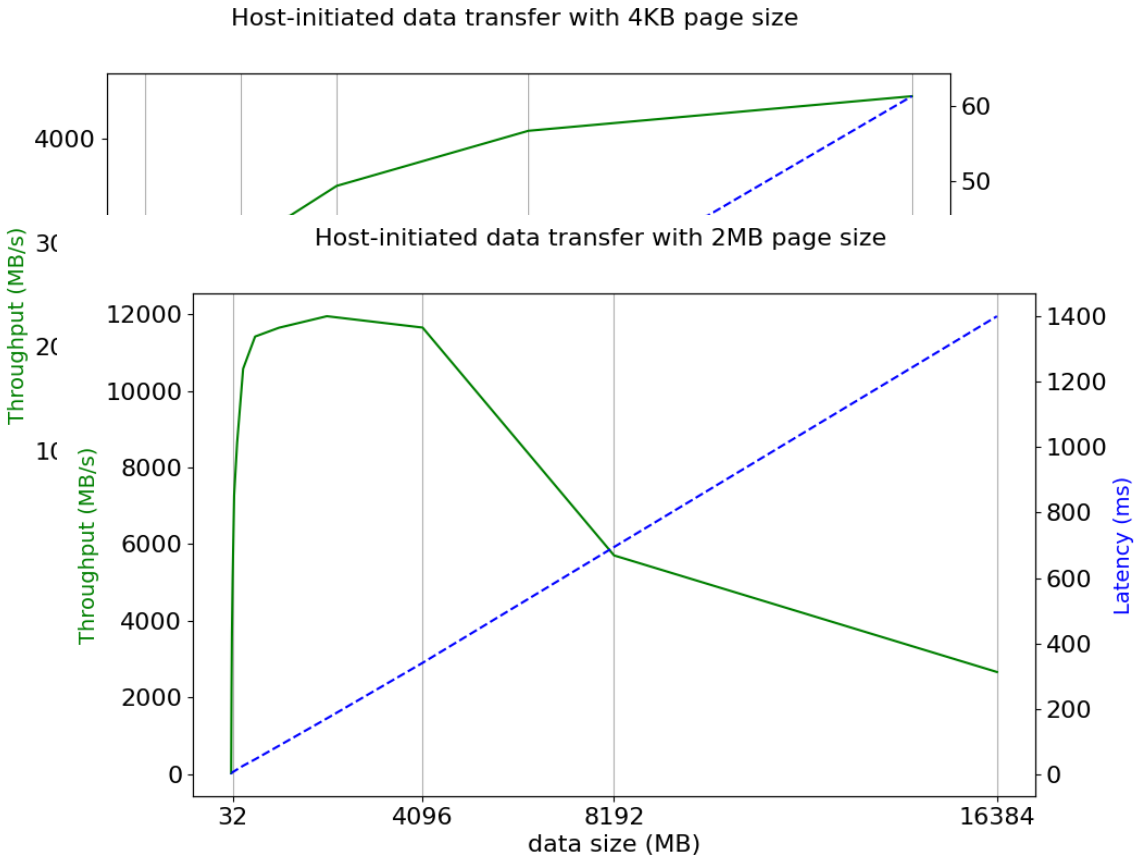
- FPGA-initiated data transfer
  - FPGA tries to access memory on the host



Test Platform (Sakura)
- CPU: Intel(R) Xeon(R) Gold 6238R
- RAM: 256 GB
- FPGA: Alveo U50 (8GB ROM)

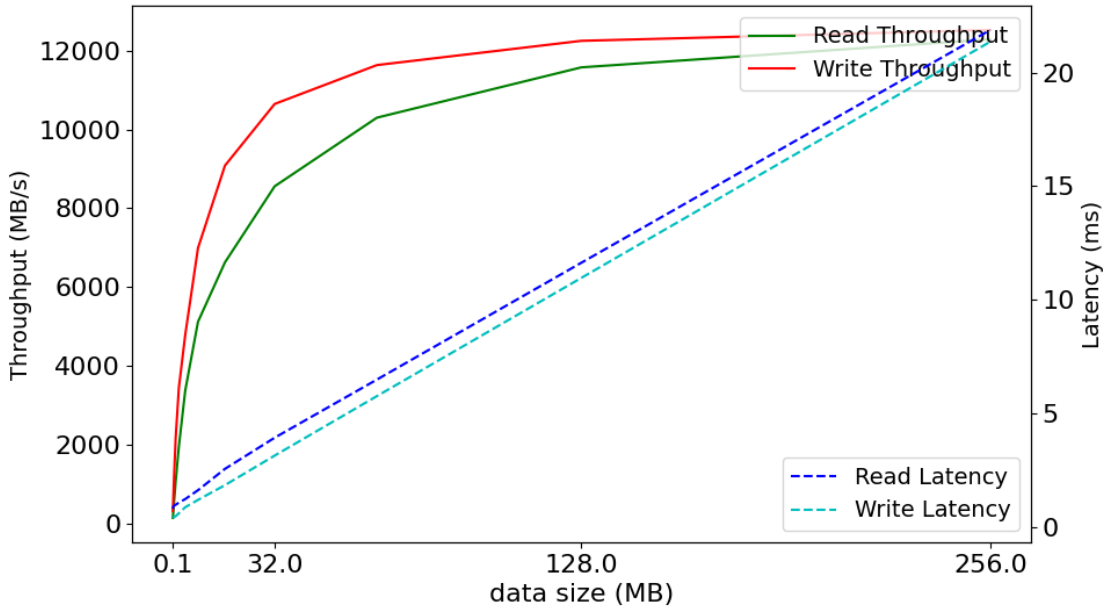
Host-initiated data transfer

- Higher Throughput as payload's size grows
- Alveo U50 has only 8 GB Memory



FPGA-initiated data transfer

- Write operation performs better than read operatior
- Higher priority for I/O scheduling



Summary

- Coyote uses PCIe address spaces to enable host-FPGA communication and process abstraction.

- The memory system enables virtual addresses for FPGA memory on the host side and implements the memory management unit (MMU) for each vFPGA

- A possible priority-based I/O scheduler can be implemented on top of memory access descriptor queues