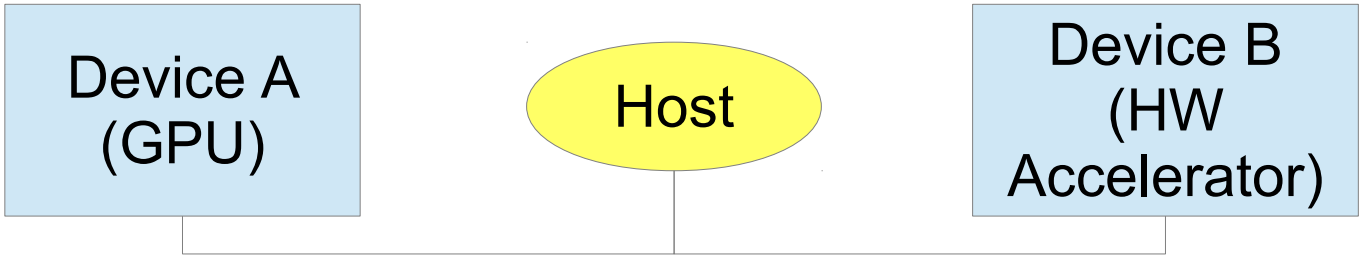
Exploring Pipe Implementations using an OpenCL Framework for FPGAs

Vincent Mirian and Paul Chow
University of Toronto
{mirianvi, pc}@eecg.toronto.edu

Why OpenCL?

- Manage heterogeneous system



- Altera release SDK in 2013, Xilinx in 2014
- Evolving standard
  - Implementation supports constructs
  - OpenCL 2.0 introduce *pipe*
- How to implement a *pipe* in modern FPGAs?

But first... what's a *pipe*?

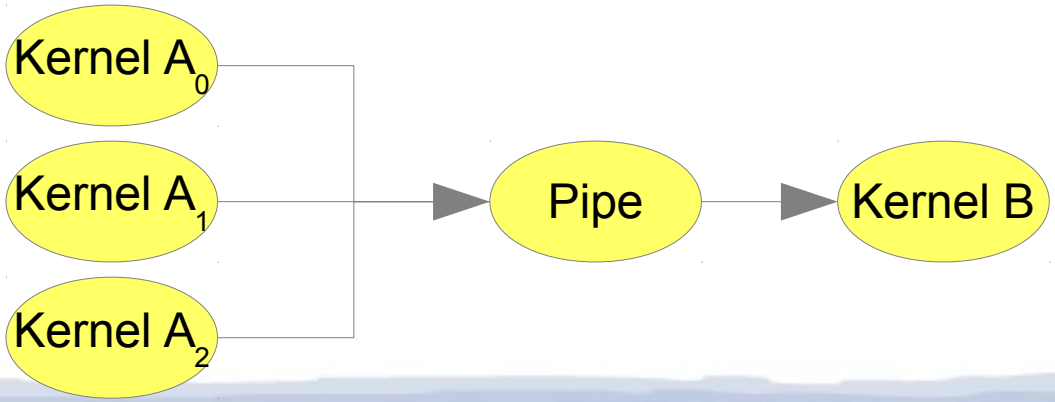
- Memory Object enabling kernel-to-kernel communication
  - 2 kernels: 1 kernel for reading + 1 for writing
  - Access to pipe using OpenCL API
- OpenCL Packet == software data structure

- Functionalities: FIFO and RAM-like behavior
  - FIFO: default functionality
  - RAM: use reservation ID to allocate space in pipe, limited # of reservation IDs per pipe

Applications using a pipe

- Streaming data (imaging):
  - FIFO behavior

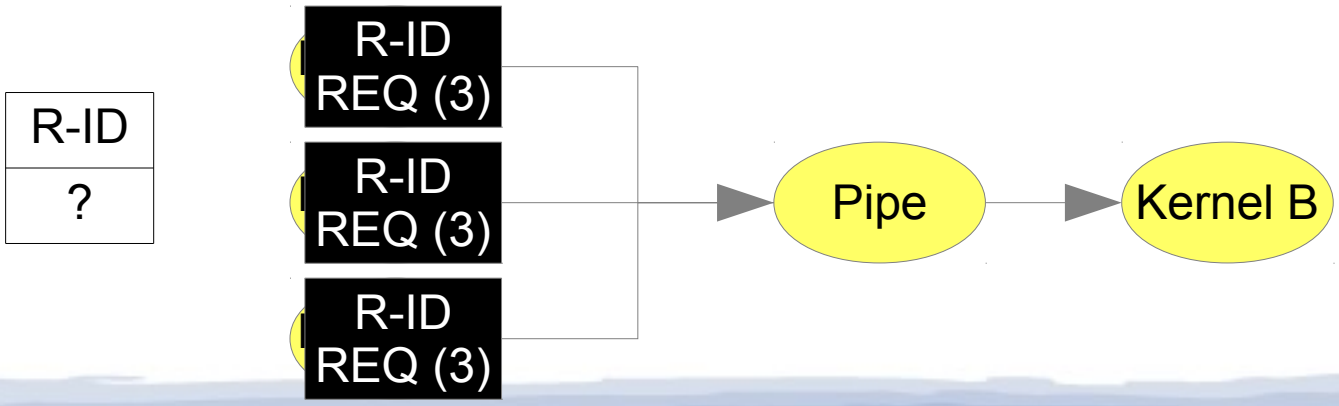
- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



Applications using a pipe

Kernel A requests a write reservation ID

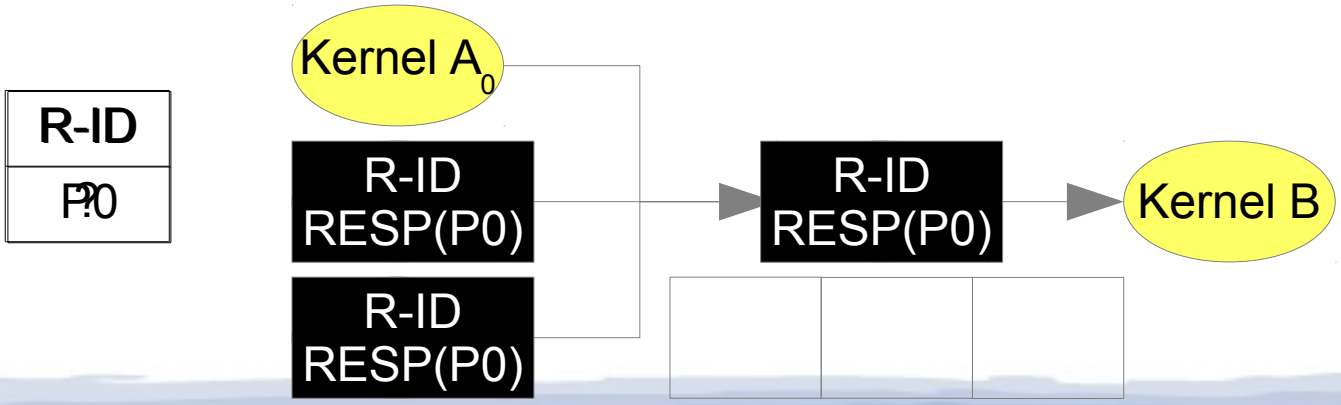
- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



Applications using a pipe

Kernel A requests a write reservation ID

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)

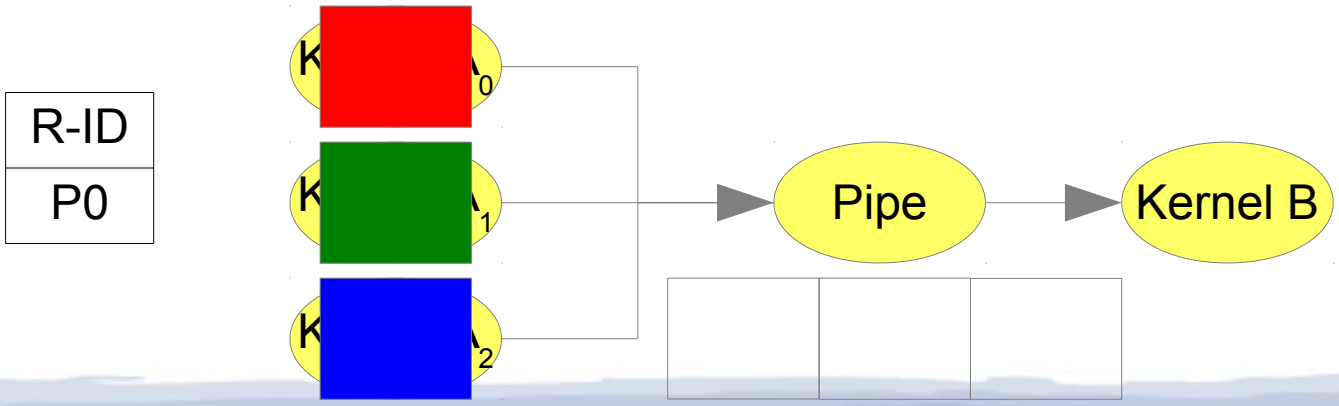


Applications using a pipe

Kernel A requests a write reservation ID

Kernel A writes to pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)

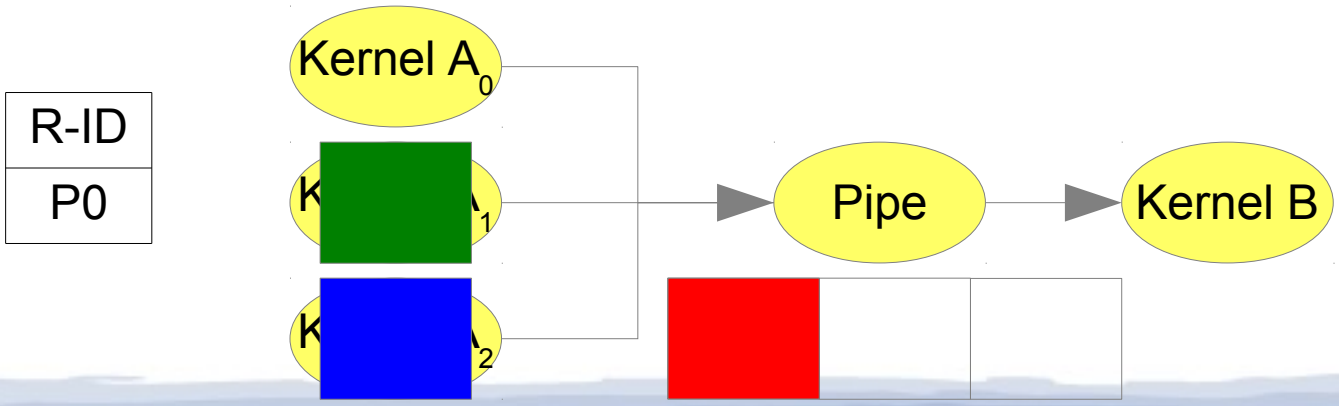


Applications using a pipe

Kernel A requests a write reservation ID

Kernel A writes to pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)

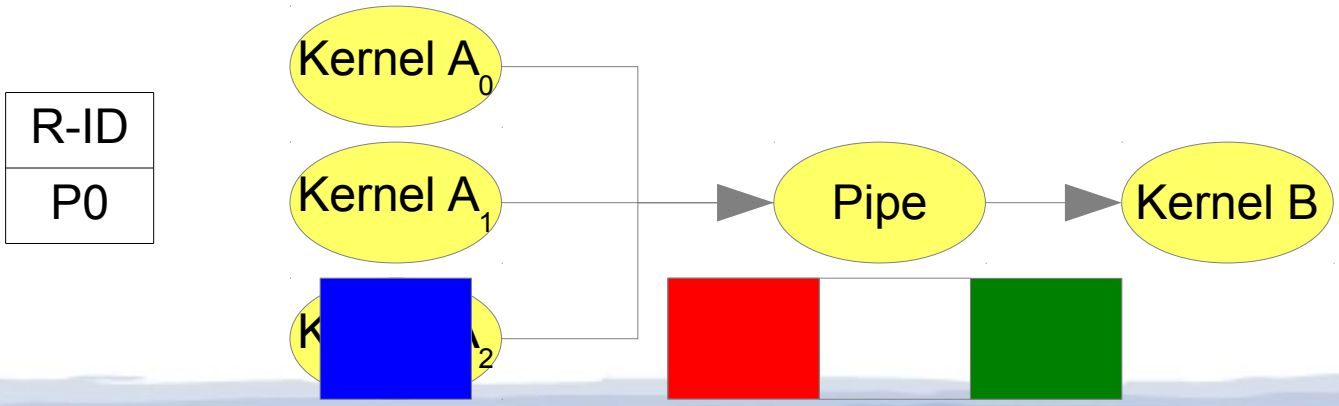


Applications using a pipe

Kernel A requests a write reservation ID

Kernel A writes to pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)

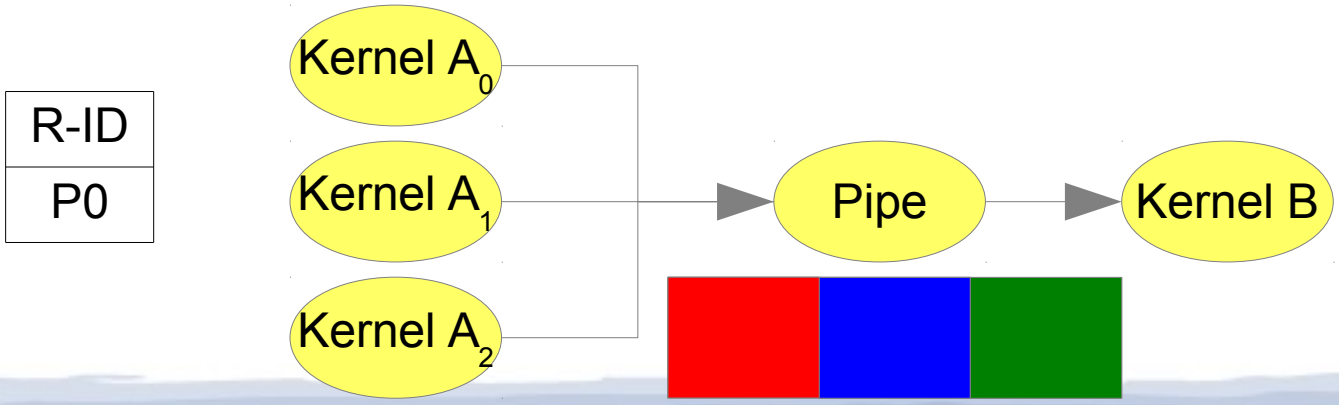


Applications using a pipe

Kernel A requests a write reservation ID

Kernel A writes to pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



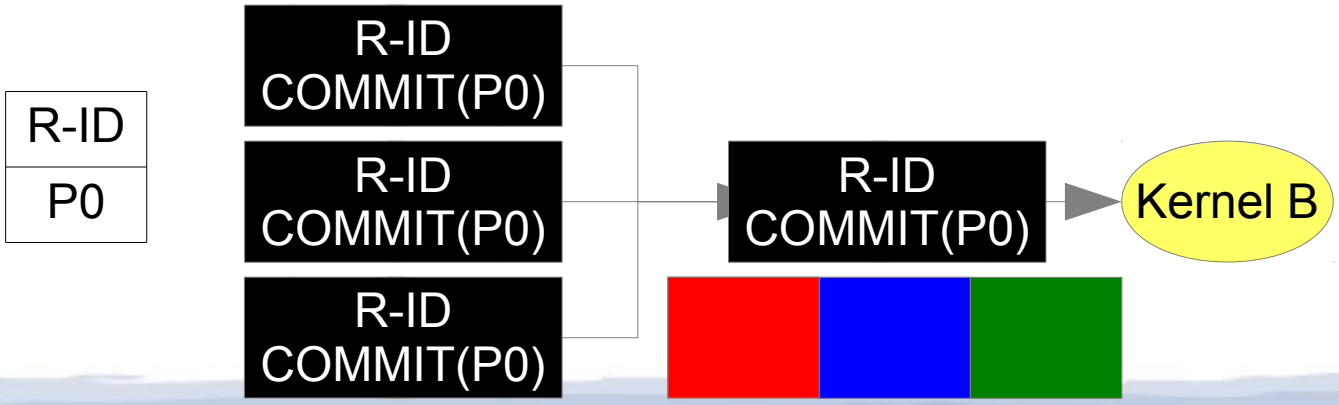
Applications using a pipe

Kernel A requests a write reservation ID

Kernel A writes to pipe

Kernel A commits the write reservation ID

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



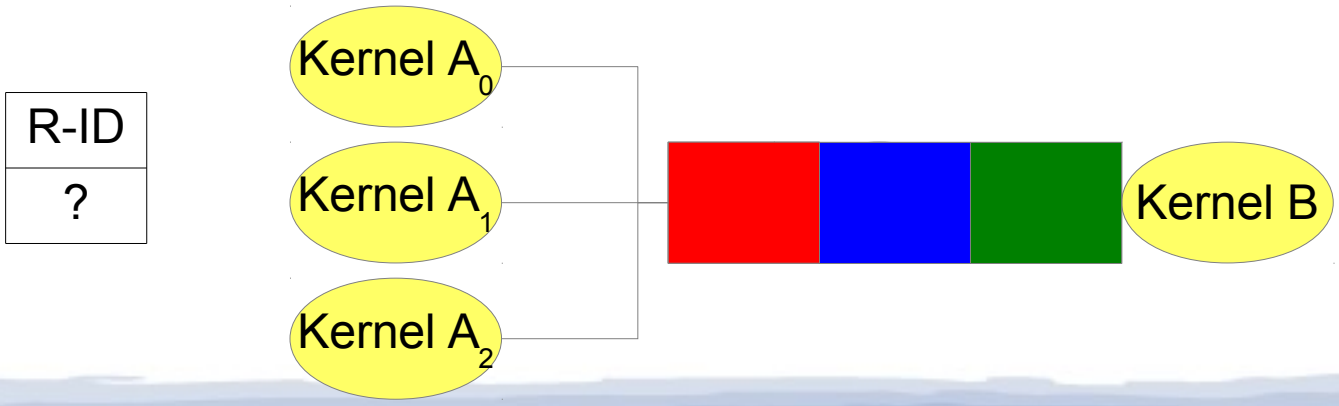
Applications using a pipe

Kernel A requests a write reservation ID

Kernel A writes to pipe

Kernel A commits the write reservation ID

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



Applications using a pipe

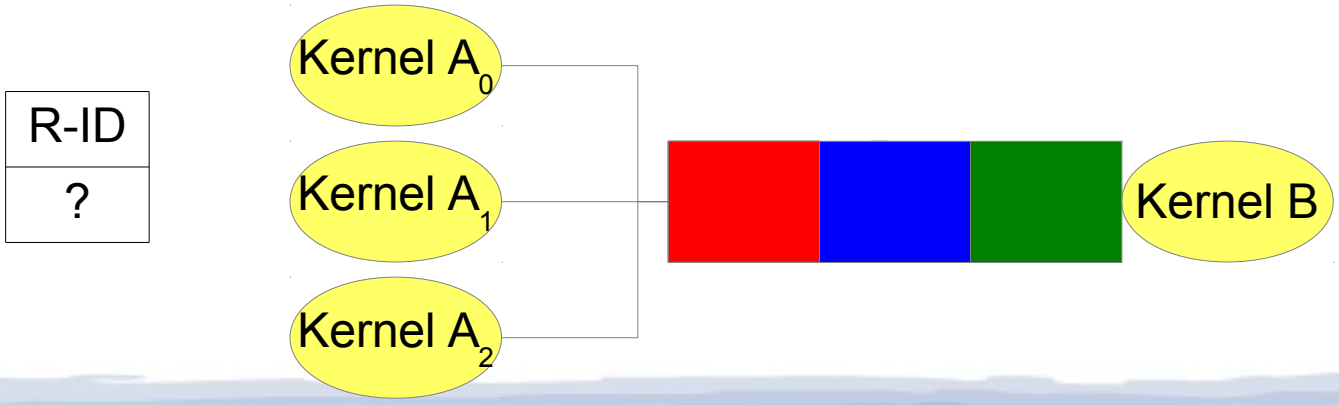
Kernel A requests a write reservation ID

Kernel A writes to pipe

Kernel A commits the write reservation ID

Kernel B reads from pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



Applications using a pipe

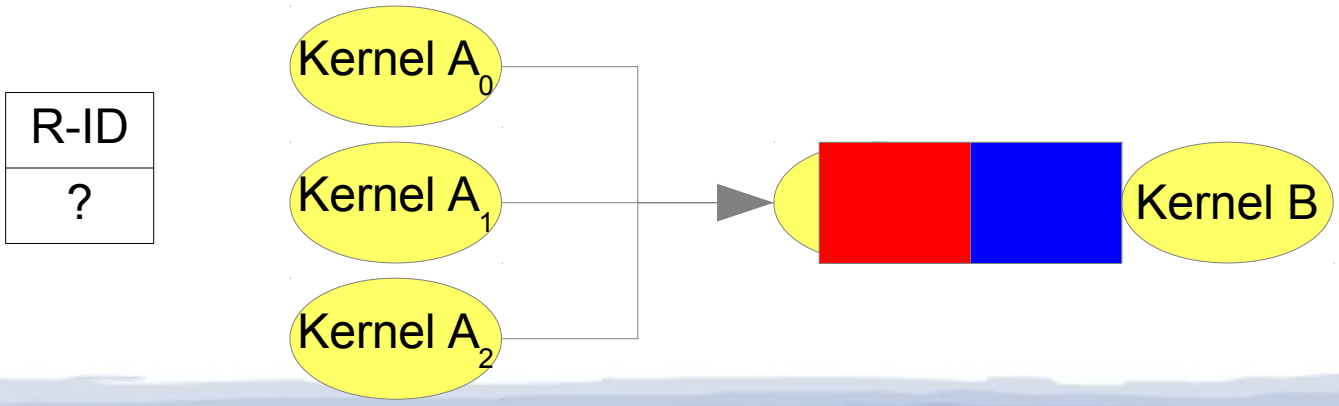
Kernel A requests a write reservation ID

Kernel A writes to pipe

Kernel A commits the write reservation ID

Kernel B reads from pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



Applications using a pipe

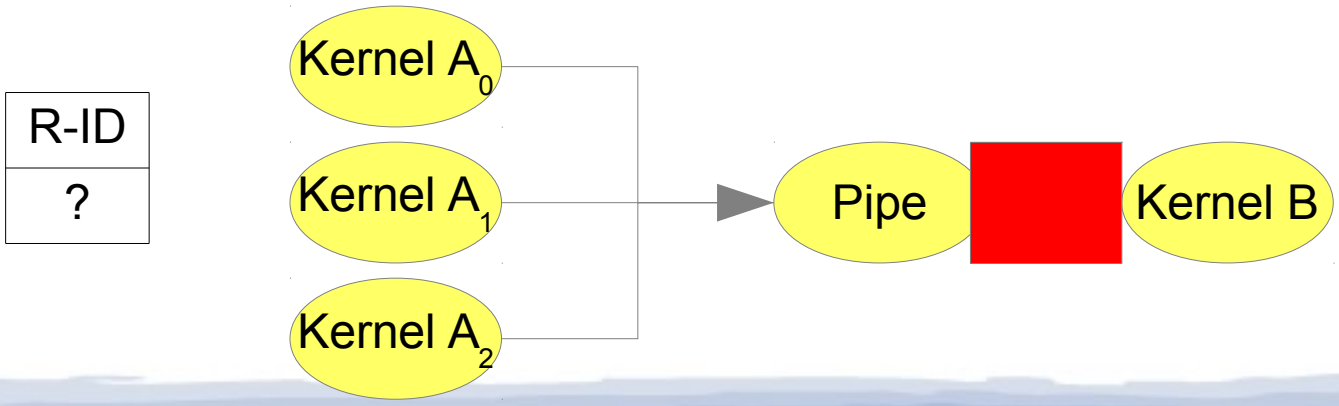
Kernel A requests a write reservation ID

Kernel A writes to pipe

Kernel A commits the write reservation ID

Kernel B reads from pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



Applications using a pipe

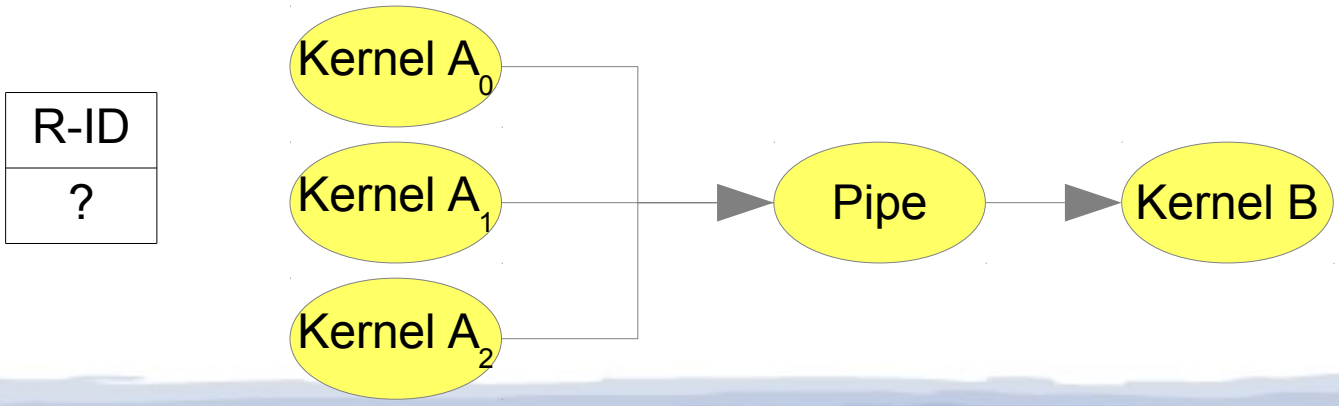
Kernel A requests a write reservation ID

Kernel A writes to pipe

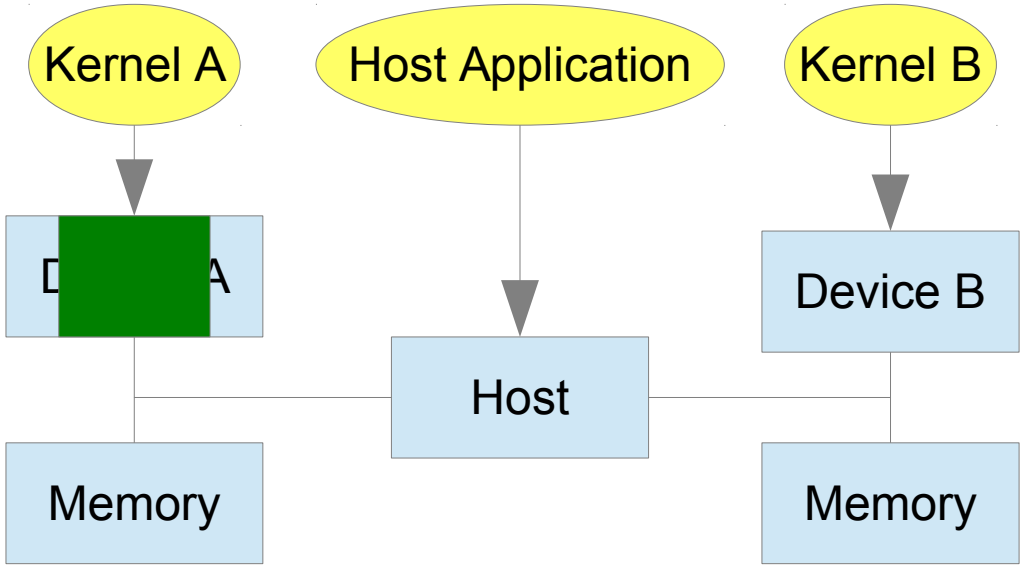
Kernel A commits the write reservation ID

Kernel B reads from pipe

- Preparing messages (networking)
  - Parallel-to-serial buffer (RAM-like behavior)



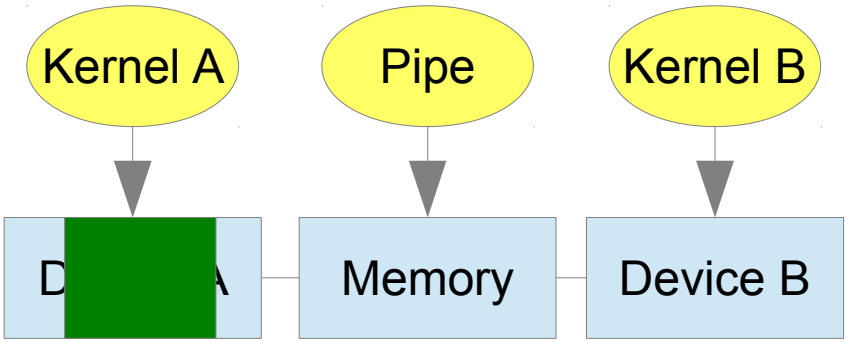
Benefit of using pipes



Actions
4

50% less actions
when using pipe support

Kernel execution without pipe support

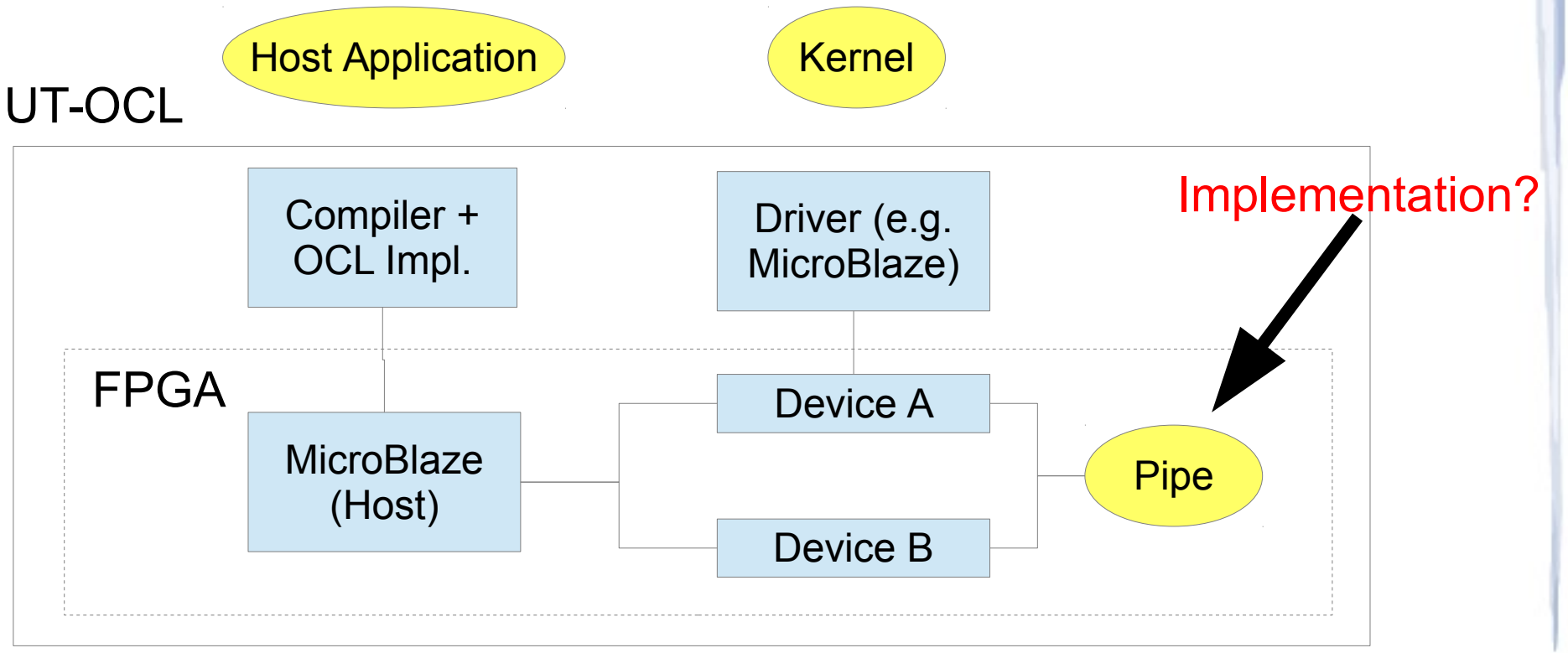


Actions
2

Kernel execution with pipe support

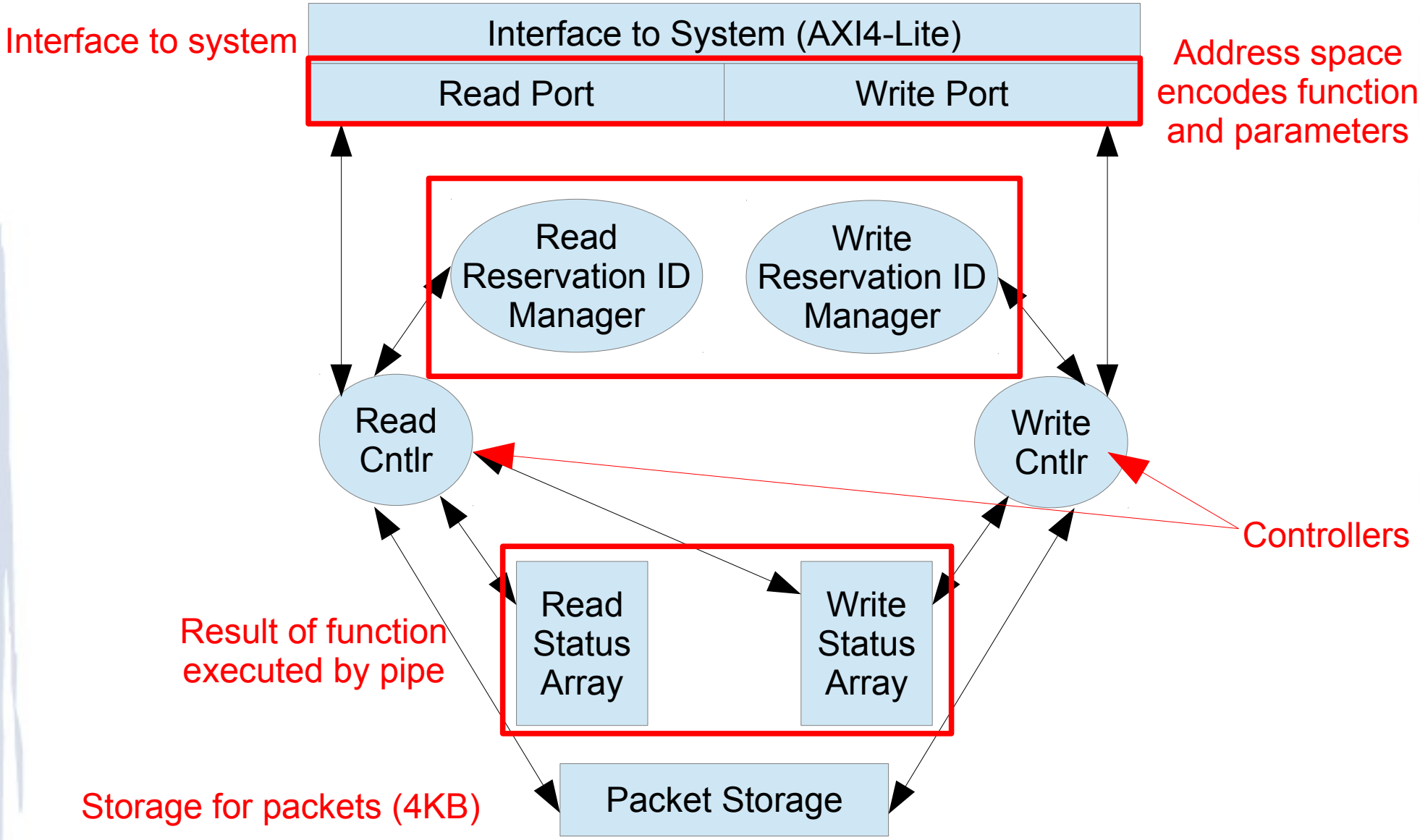
Our Context: Embedded Systems

- UT-OCL (University of Toronto)
  - OpenCL framework for embedded systems using Xilinx FPGAs



Download: http://www.eecg.toronto.edu/~pc/downloads/UT-OCL/

Our Proposed Implementation



A side note:

Pipe Requirements	Altera's Channel [1]	Xilinx's Pipe[2]	Our Implementation
FIFO Behavior	✓	✓	✓
RAM-like Behavior Reservation ID >= 1	✗	✗	✓
OpenCL function signature	✗	✓	✓

- First to introduce a pipe implementation that conforms with the OpenCL Spec.
  - Context: FPGA

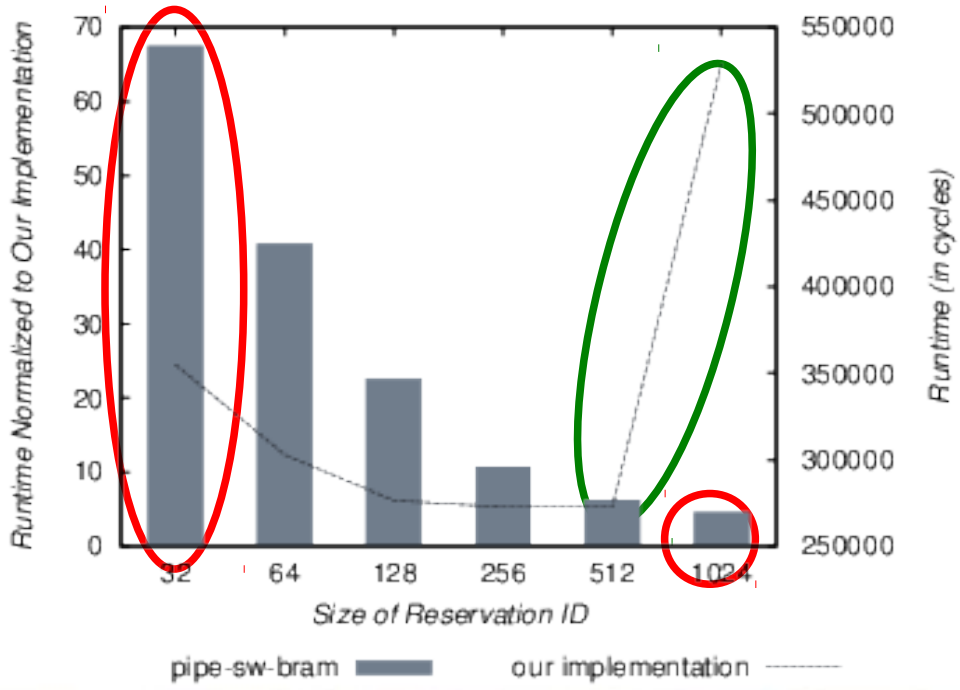
Reference:
[1] Altera Corp., *Altera SDK for OpenCL- Programming Guide (OCL002-15.0.0)*, May 2015.
[2] Xilinx Inc., SDAccel Development Environment: User Guide (UG1023), October 2015.

Results: Parallel-to-serial buffer

- Parity computation for 1K x 4byte packets (4KB)
  - 16 uB writing, 1 uB reading
- SW impl. model buffer: pipe-sw-bram (on-chip storage)

1- Our implementation performs faster

2- Workload imbalance



Results: Resource Utilization

Implementation	FF	LUT	BRAM
Base System	38,998	42,332	69
pipe-sw-bram	39,478 (+1.2%)	42,738 (+1.0%)	70 (+1.4%)
Our implementation	39,677 (+1.7%)	43,235 (+2.1%)	70 (+1.4%)

- Our implementation uses more FFs and LUTs than pipe-hw-bram
  - Additional cost: implement the flags, mutexes, mutex operation and counter found in pipe-hw-bram implementation in hardware

Conclusion

- Pipe:
  - FIFO and RAM-like behavior
  - Reduce kernel-to-kernel communication
- First implementation of a pipe for use within an FPGA
- Observations:
  - Our impl. performs faster than software impl.
  - Tune reservation ID for application
- UT-OCL: Open-source OpenCL framework for embedded systems using FPGAs