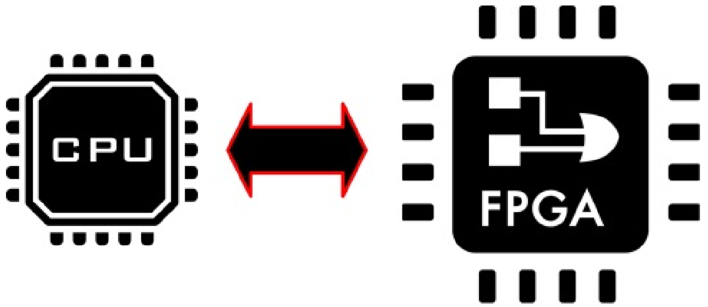
Do OS abstractions make sense in FPGAs?

Dario Korolija
Timothy Roscoe
Gustavo Alonso

Systems Group, Dept. of Computer Science, ETH Zurich

Introduction

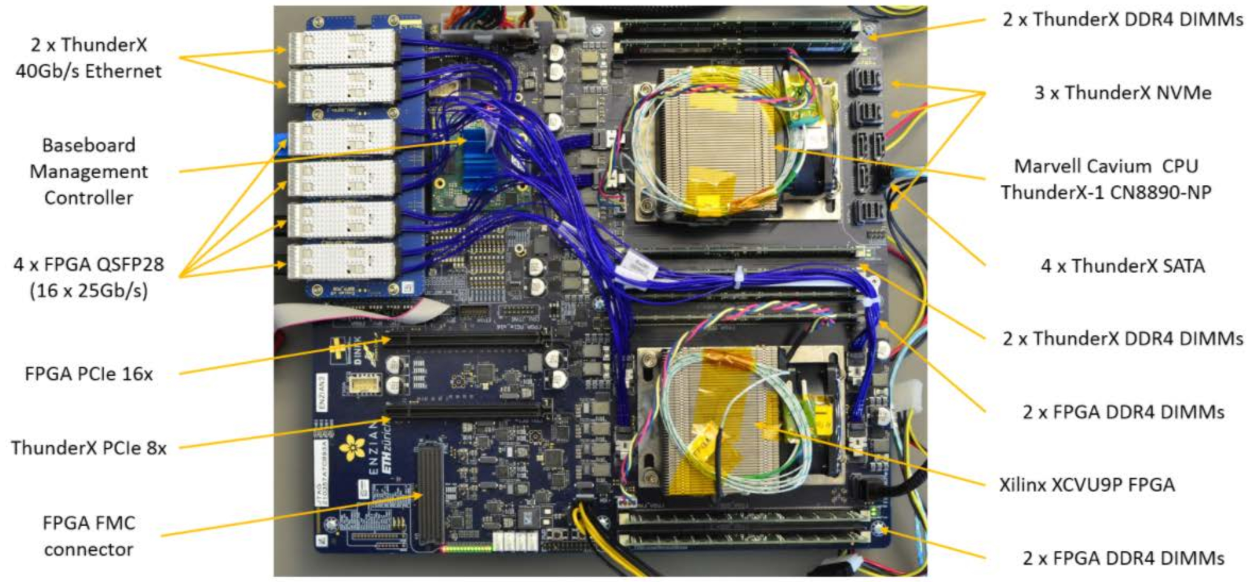
- Hybrid computing systems (*Amazon F1, Microsoft Catapult, Intel HARP, Alibaba F3*...)



- At ETH we have built one of these:

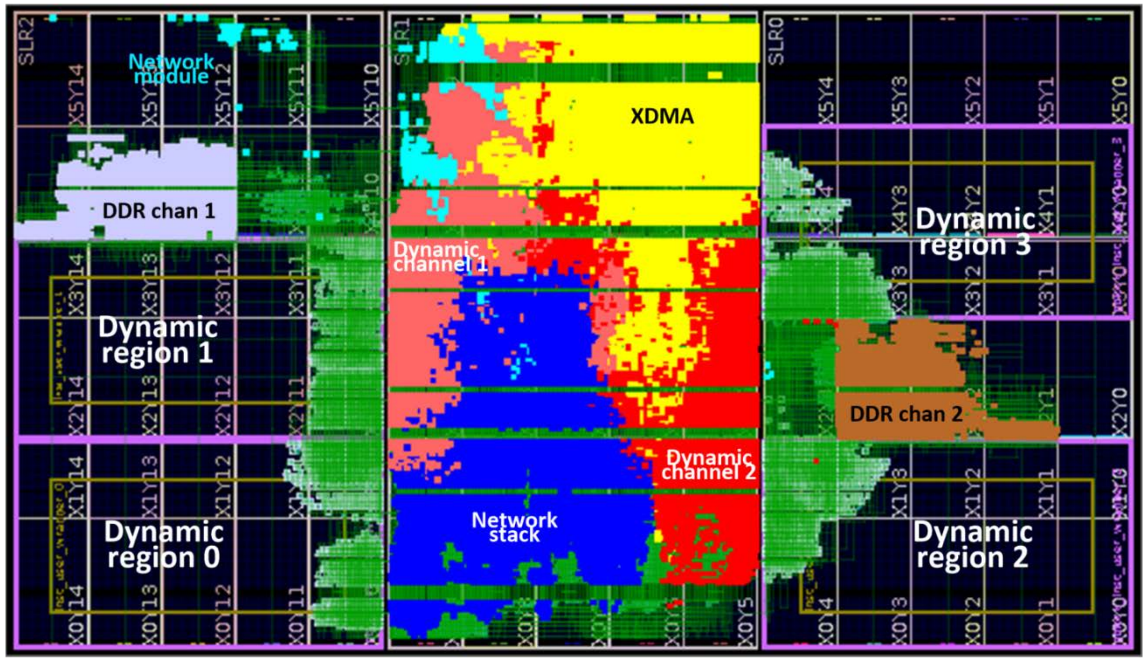
ENZIAN

http://enzian.systems



Hybrid computing systems are complex to program

- CPU-FPGA interaction
- Difficult to program
- No standard execution environment
- Lack of portability
- **Lack of proper abstractions**



FPGA layout
PCIe, memory, storage, network…

OS functionality is starting to appear

- ❖ Shells for hybrid systems:
  - Catapult, HARP, SDAccel, Vitis
- ❖ Memory Management:
  - CoRAM
- ❖ Virtualization and scheduling :
  - Vital, Optimus, AmorphOS...

Focus on particular subsets of functionality only!

A Hypervisor for Shared-Memory FPGA Platforms

Jiacheng Ma[*] Gefei Zuo[*] Kevin Loughlin[*] Xiaohe Cheng[§] Yanqiang Liu[†]
Abel Mulugeta Eneyew[‡] Zhengwei Qi[†] Baris Kasikci[*]
[*]University of Michigan [§]Hong Kong University of Science and Technology
[†]Shanghai Jiao Tong University [‡]Addis Ababa Institute of Technology

Sharing, Protection, and Compatibility for Reconfigurable Fabric with AMORPHOS

Ahmed Khawaja[1], Joshua Landgraf[1], Rohith Prakash[1],
Michael Wei[2], Eric Schkufza[2], Christopher J. Rossbach[3]
[1]The University of Texas at Austin [2]VMware Research Group
[3]The University of Texas at Austin and VMware Research Group

The Feniks FPGA Operating System for Cloud Computing

Jiansong Zhang[§] Yongqiang Xiong[§] Ningyi Xu[§] Ran Shu[§†] Bojie Li[§‡]
Peng Cheng[§] Guo Chen[§] Thomas Moscibroda[§]
[§] Microsoft Research [†]Tsinghua University [‡]USTC
{jiazhang,yqx,ningyixu,v-ranshu,v-bojli,pengc,guoche,moscitho}@microsoft.com

A Cloud-Scale Acceleration Architecture

Adrian M. Caulfield Eric S. Chung Andrew Putnam
Hari Angepat Jeremy Fowers Michael Haselman Stephen Heil Matt Humphrey
Puneet Kaur Joo-Young Kim Daniel Lo Todd Massengill Kalin Ovtcharov
Michael Papamichael Lisa Woods Sitaram Lanka Derek Chiou Doug Burger

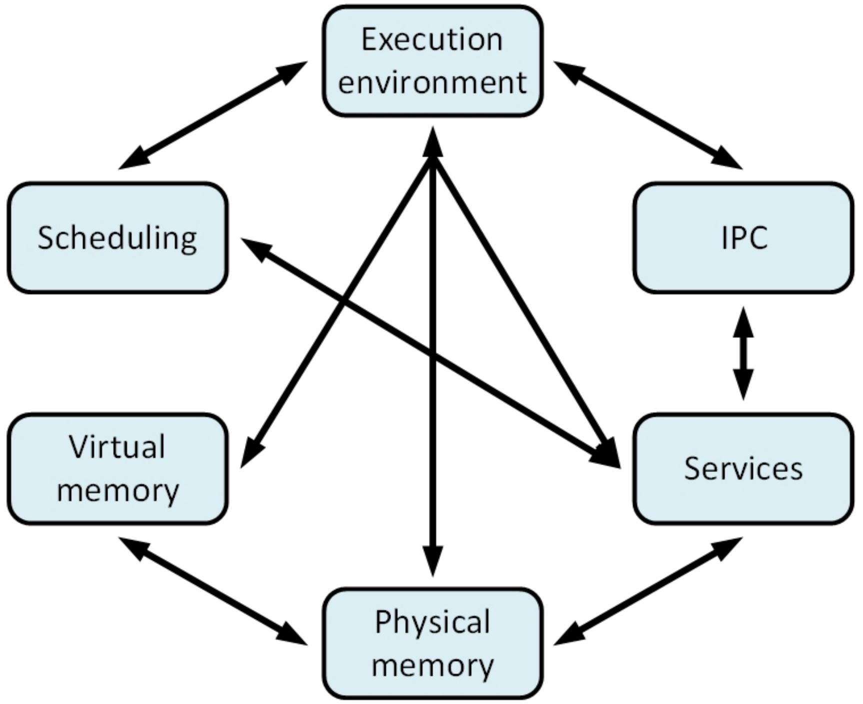
Microsoft Corporation

Coyote

Hybrid computing system

- Coyote provides a **complete minimal core set** of essential features above which other services can be based

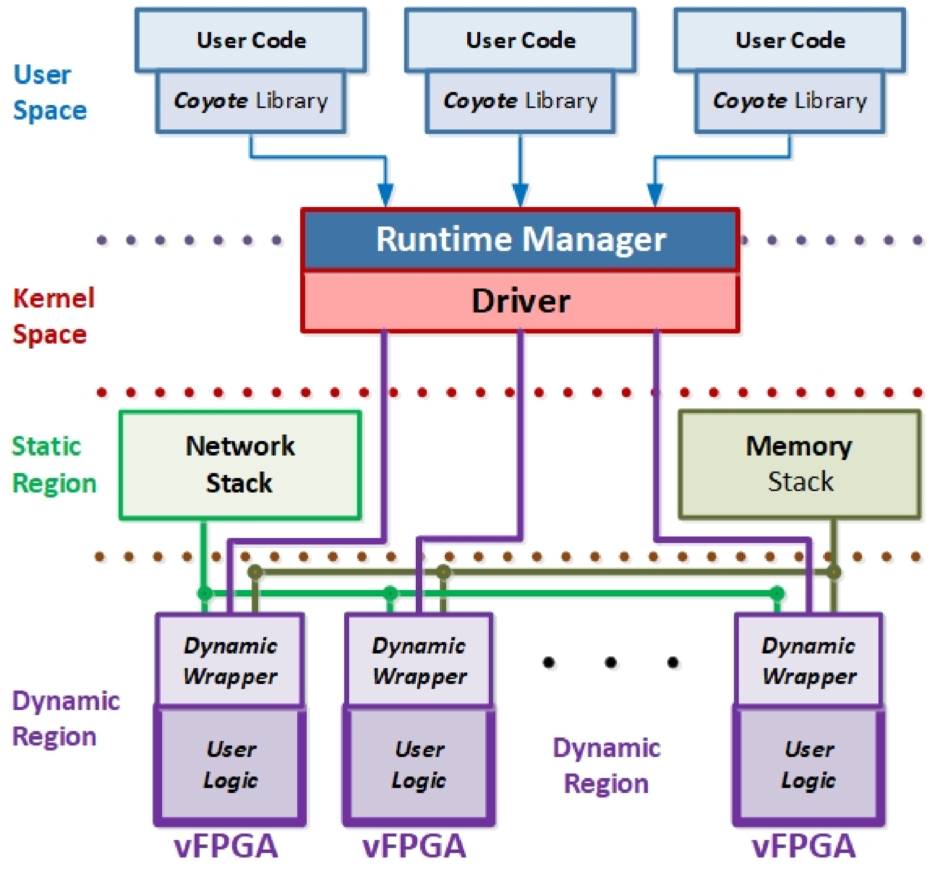
Interdependence



Coyote system foundations

- Hardware split into *static* and *dynamic* regions
  - Dynamic region split into multiple **vFPGAs**
    - **vFPGA** further split into the *User Logic* and the *Dynamic Wrapper*

- Functionality not on the critical path handled by the host OS
  - Kernel driver
  - Runtime scheduler
  - High level API

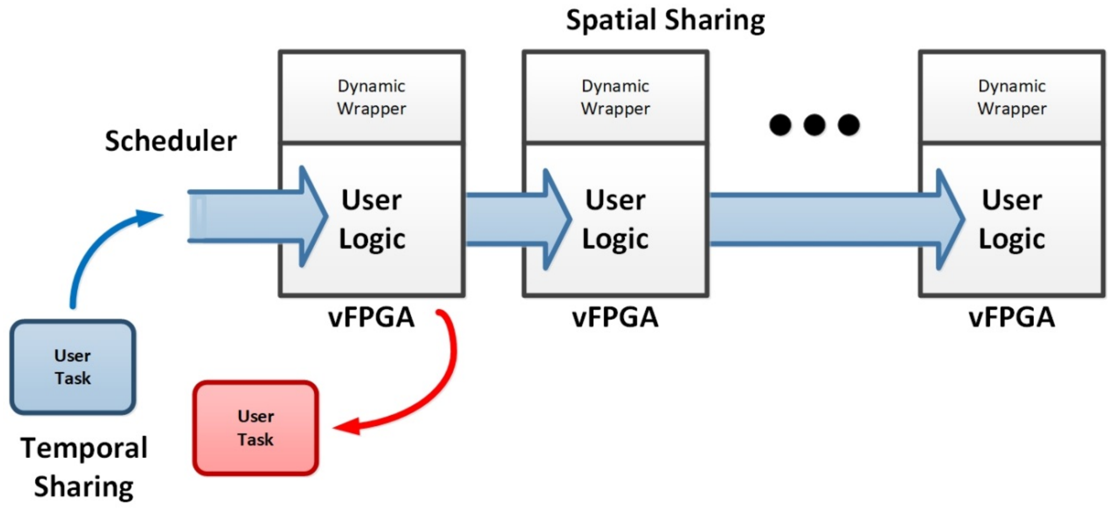


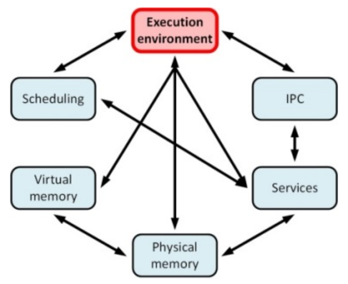
ETH zürich Systems@ETH zürich D INFK

Basic multiplexing abstractions: Processes, threads and tasks

- **FPGAs are fundamentally different**
  - No CPUs or cores
  - Spatial and temporal sharing

- **Coyote:**
  - Combines both approaches
  - Multitasking abstraction for a set of independent, isolated vFPGAs
  - **vFPGAs are equivavalent**

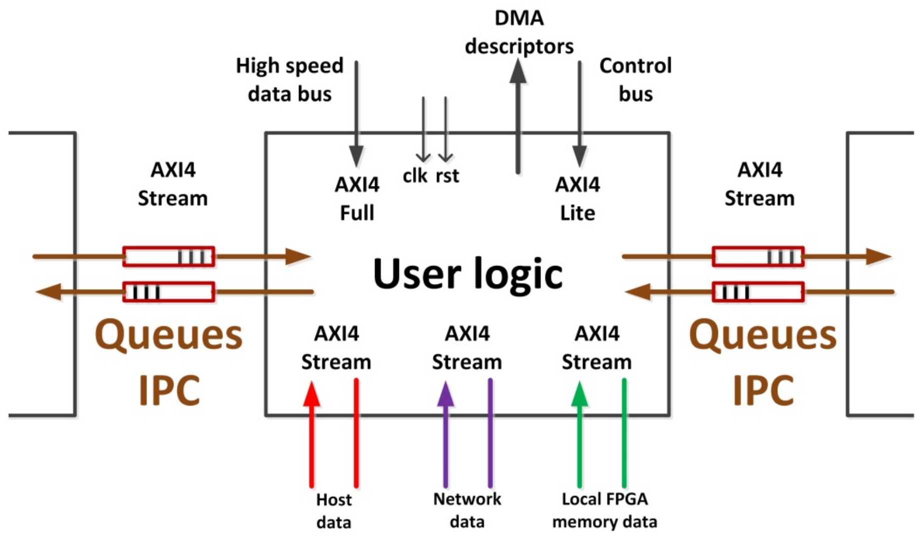


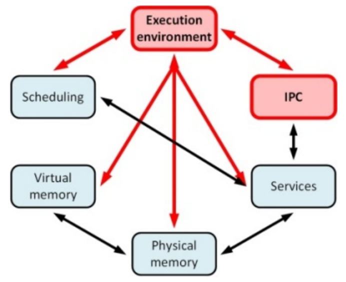


Execution environment: Application portability

- **Don’t really exist across the FPGA platforms**
  - Effort by Xilinx with **Vitis**
  - FPGA not only a pure computational device

- **Coyote:**
  - Single User Logic Interface (***ULI***)
  - Interaction with the complete system
  - Access is low level
  - **Portability and extensibility**

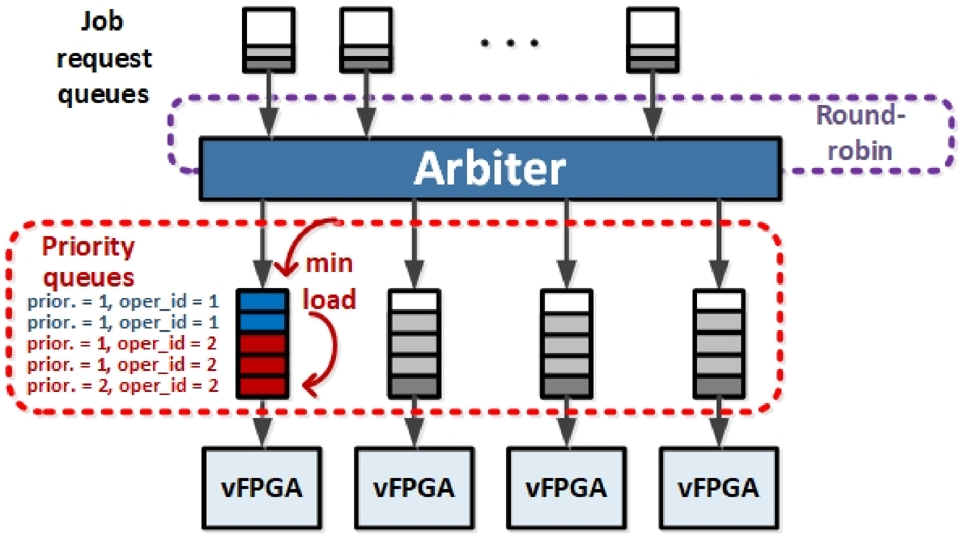


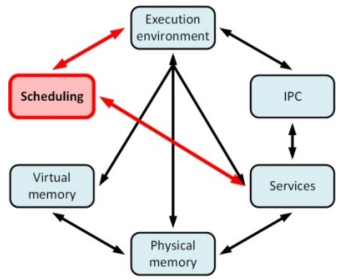


Scheduling: Non-preemptive

- **Basic mechanisms to capture the state of the FPGA don't exist**

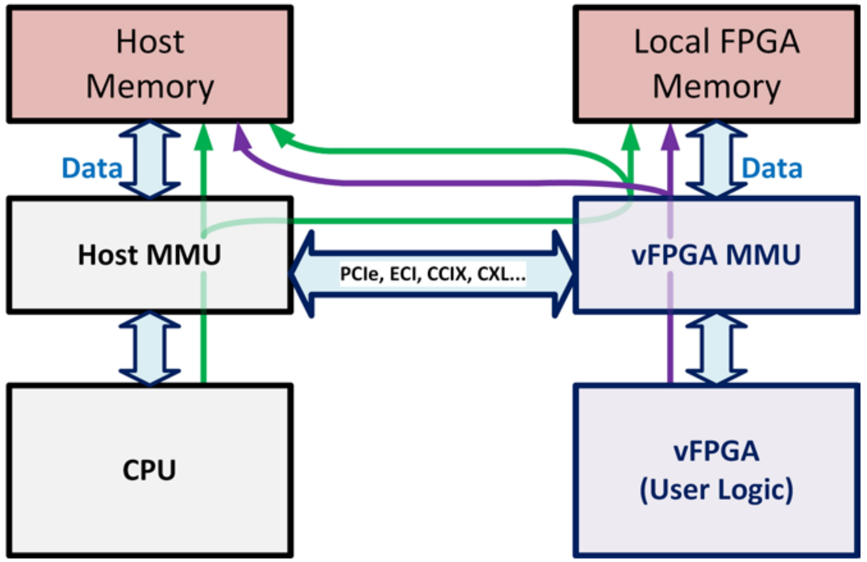
- **Coyote:**
  - Non-preemptive task based approach
  - **Avoid preemption**
    - User applications can't always be trusted
    - Preemption requires cooperation
    - Preemption would impose additional application complexity
    - Additional problems of capturing the state of stateful services

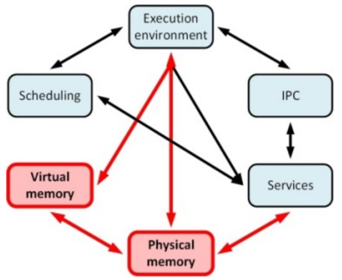




Memory management: Access flexibility

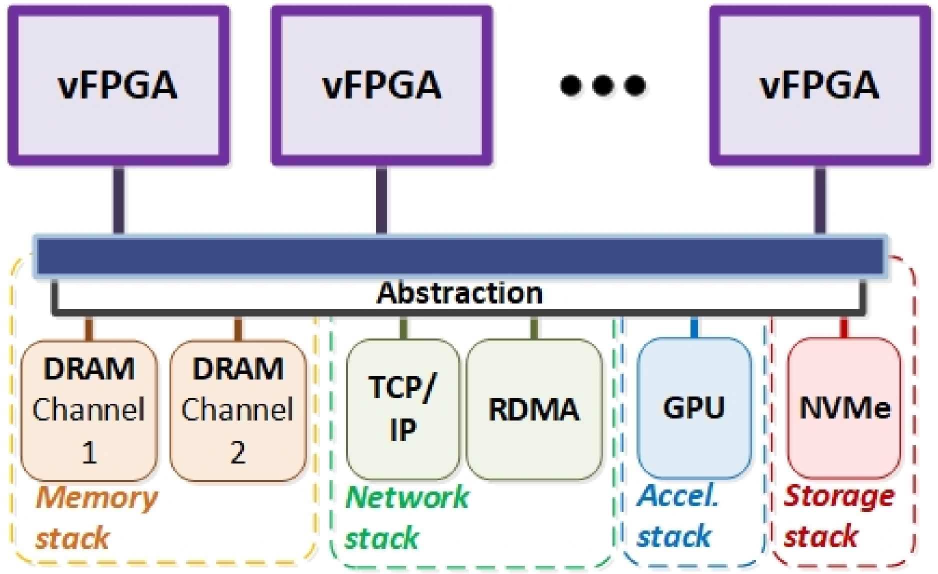
- **Virtual memory tends to be ignored in FPGAs**
- **Coyote implements a flexible approach giving us multiple ways to access both host and local FPGA memory**
- **Unified memory model built on top adds to the programmability**



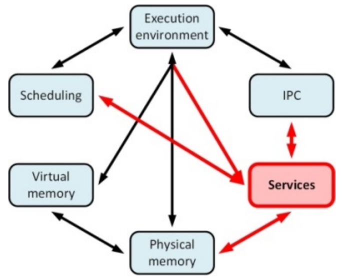


Services

- **Network:** TCP/IP and RDMA network stacks[1] shared between **vFPGAs**.
- **On-board FPGA memory:** Hiding the complexity of multiple local memory channels through *striping* abstraction
- **Further services easily possible**:
  - External storage
  - External accelerators (GPUs, ASICs)



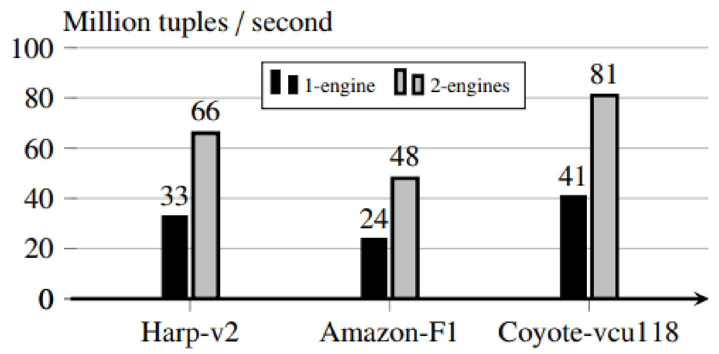
[1] *StRoM: Smart Remote Memory*
David Sidler, Zeke Wang, Monica Chiosa, Amit Kulkarni, Gustavo Alonso



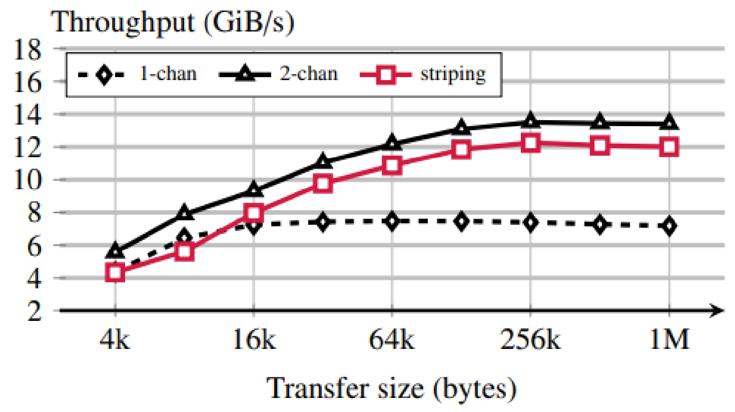
Evaluation

- ❖ Depending on the configuration Coyote shell uses 15-25% of current FPGA resources
- ❖ Compared with commercial systems Coyote achieves comparable or better performance for a real-world use case [1]
- ❖ Microbenchmarks:
  - Sharing of the resources is fair
  - Scheduling tactic reduces overall execution time
  - Abstraction performance penalties are negligible

Decision trees:



Striping:



ETH zürich Systems@ETH Zürich D INFK

Hybrid computing systems are difficult to program
The need for proper abstractions is evident

To find the right abstractions for FPGAs,
a complete set of functions has to be considered

https://github.com/fpgasystems

This work has been made possible through a generous equipment donation from Xilinx
Xilinx Adaptive Compute Cluster (XACC) Program: https://www.xilinx.com/support/university/XUP-XACC.html