Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V SoC on FireSim

Farzad Farshchi[§], Qijing Huang[¶], Heechul Yun[§]

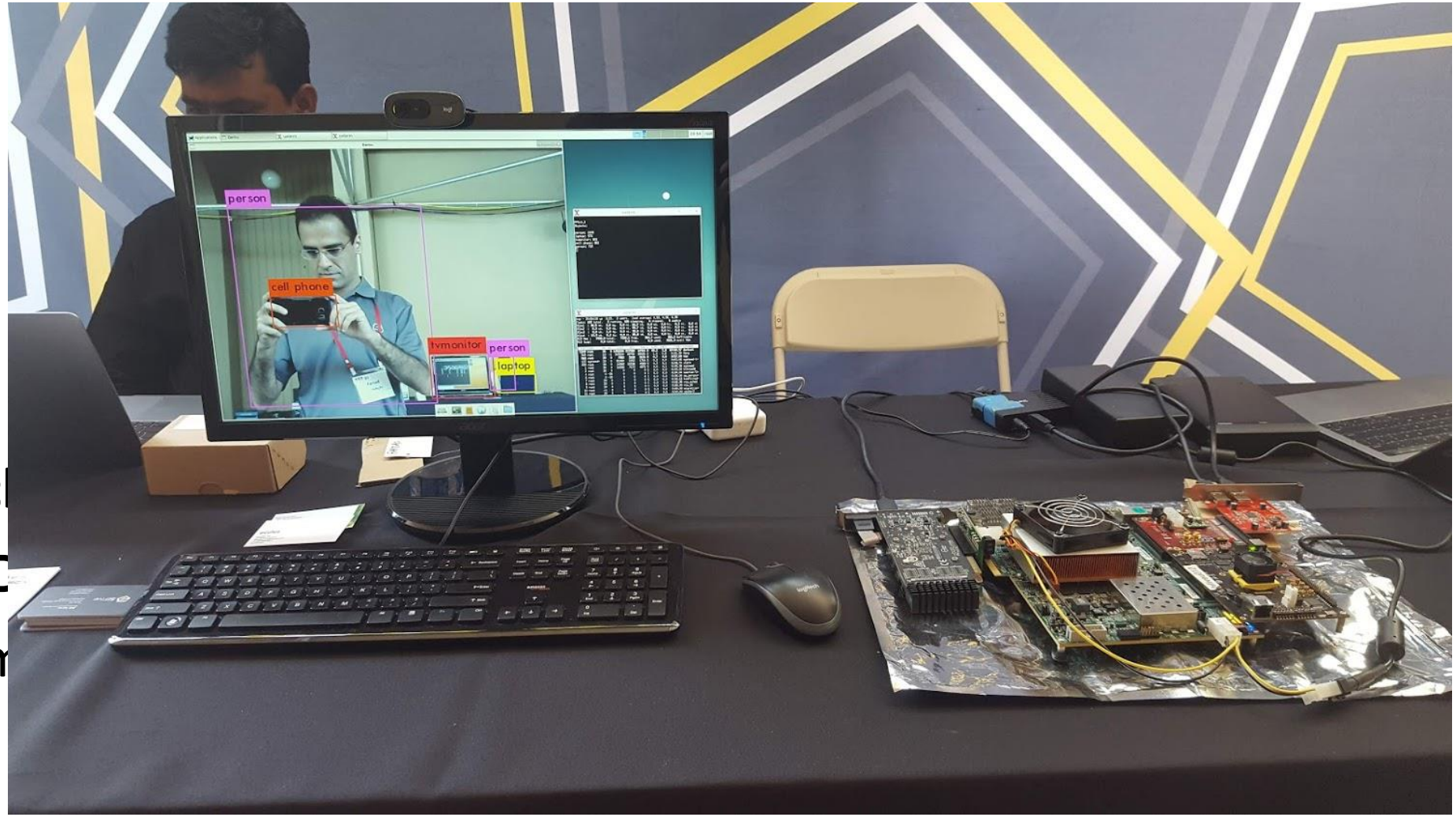
[§]University of Kansas, [¶]University of California, Berkeley

SiFive Internship

- **Rocket Chip**: open-source RISC-V SoC
- **NVDLA**: open-source DNN inference engine
- Demoed the integration at Hot Chips'18

SiFive Internship

- **Rocl**
- **NVD**
- Dem

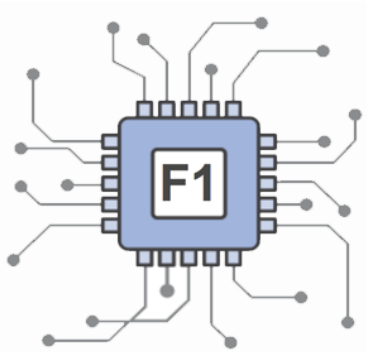


Motivation

- Useful platform for research
- Limitations
  - No L2
  - Fast DRAM, slow SoC
  - Expensive: $7k FPGA board
- Let’s integrate NVDLA into FireSim

FireSim

- Fast, cycle-exact full system simulator, runs on FPGA in the cloud
- Simulated design is derived from Rocket Chip RTL
- Decouples target from FPGA DRAM
  - Adds its own DRAM and LLC model
- Easy-to-use. Very good documentation.



How FireSim Works?

- Transforms RTL to target model
  - Inserts queues at I/O ports of target
  - Creates a token-based simulator
- In each cycle a token is consumed by model
- What if token queue is empty?
  - The model has to wait

➡ Stall the target pipeline

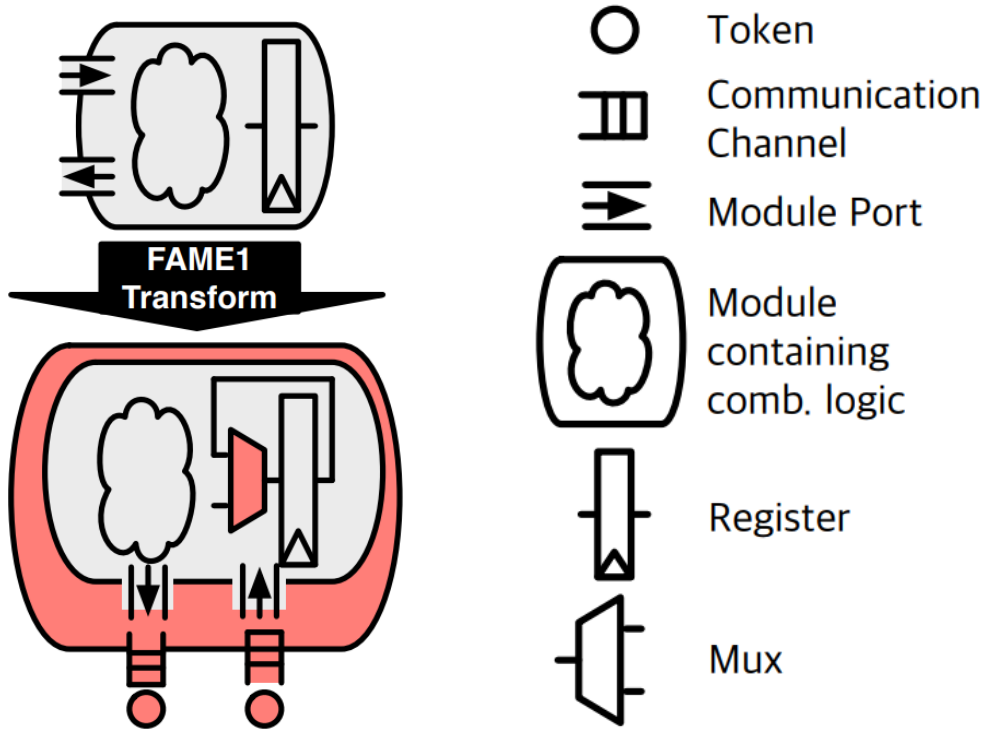
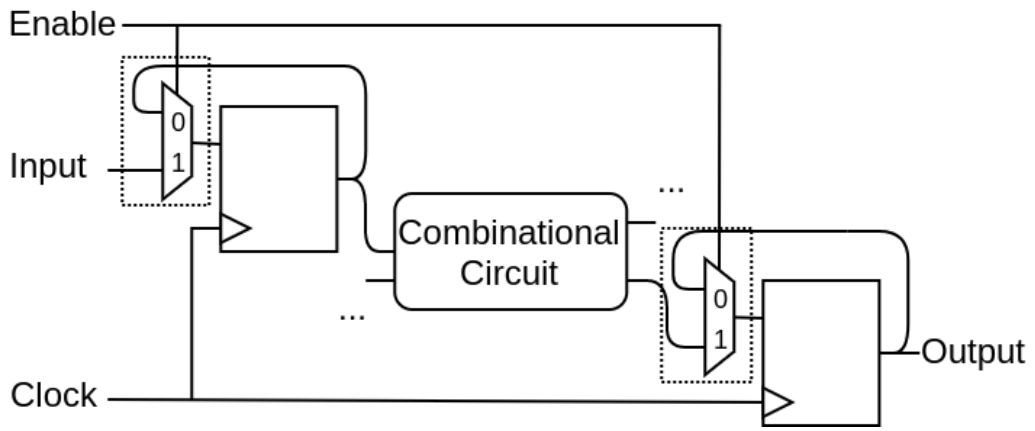


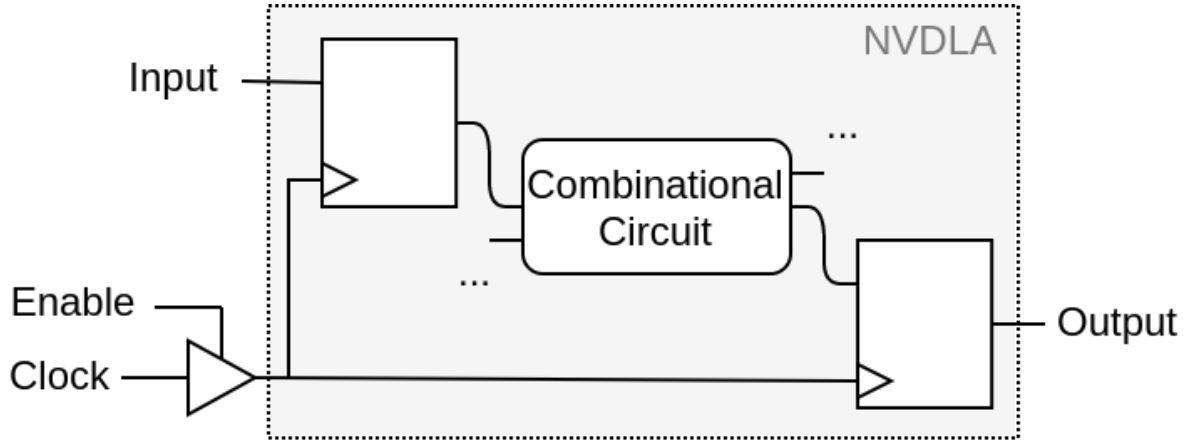
Figure credit: Donggyu Kim et al. “Strober: Fast and Accurate Sample-Based Energy Simulation for Arbitrary RTL”

How to Stall The Target Pipeline?

- For Chisel code:
  - Rocket Chip is written in Chisel

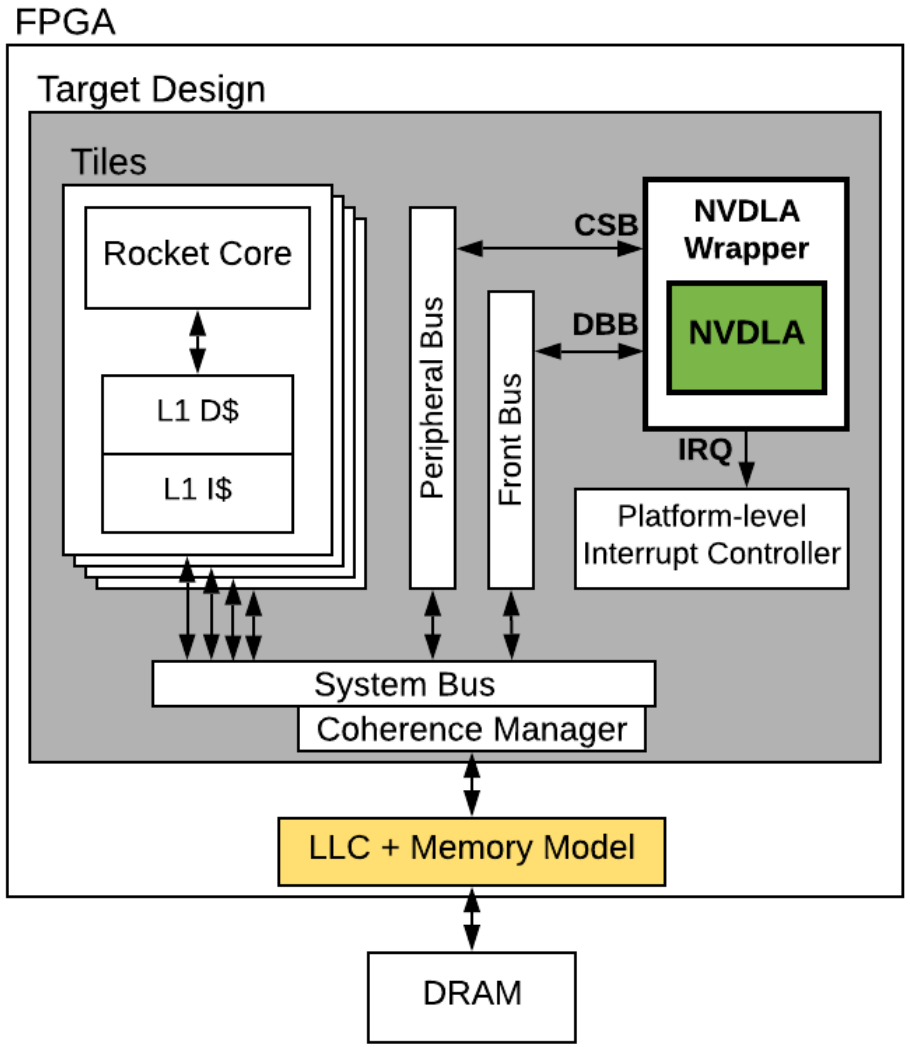


- For Verilog (we added):



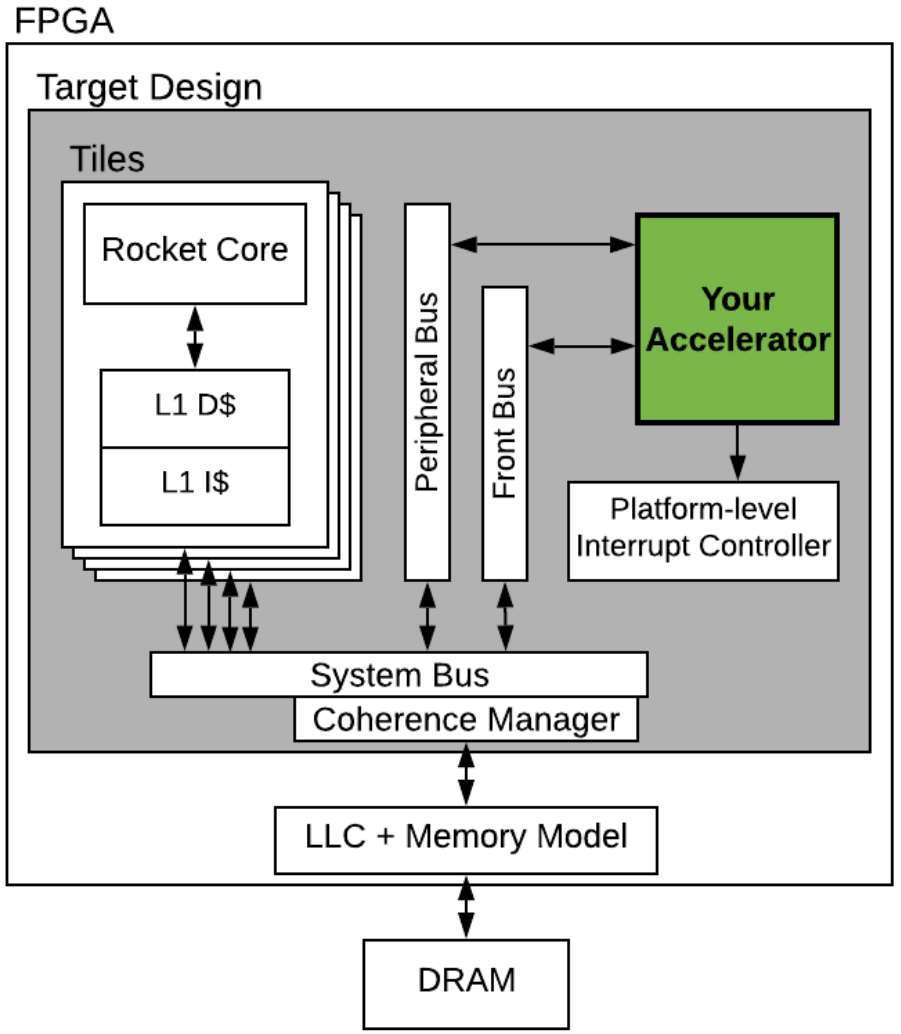
Overall System Architecture

- NVDLA is integrated in target
- LLC + Memory Model: Not part of the target. Added by FireSim.
  - Supports multiple models e.g. DDR3, constant latency
  - Runtime configurable LLC: different set, way, block sizes. No need to rebuild FPGA image



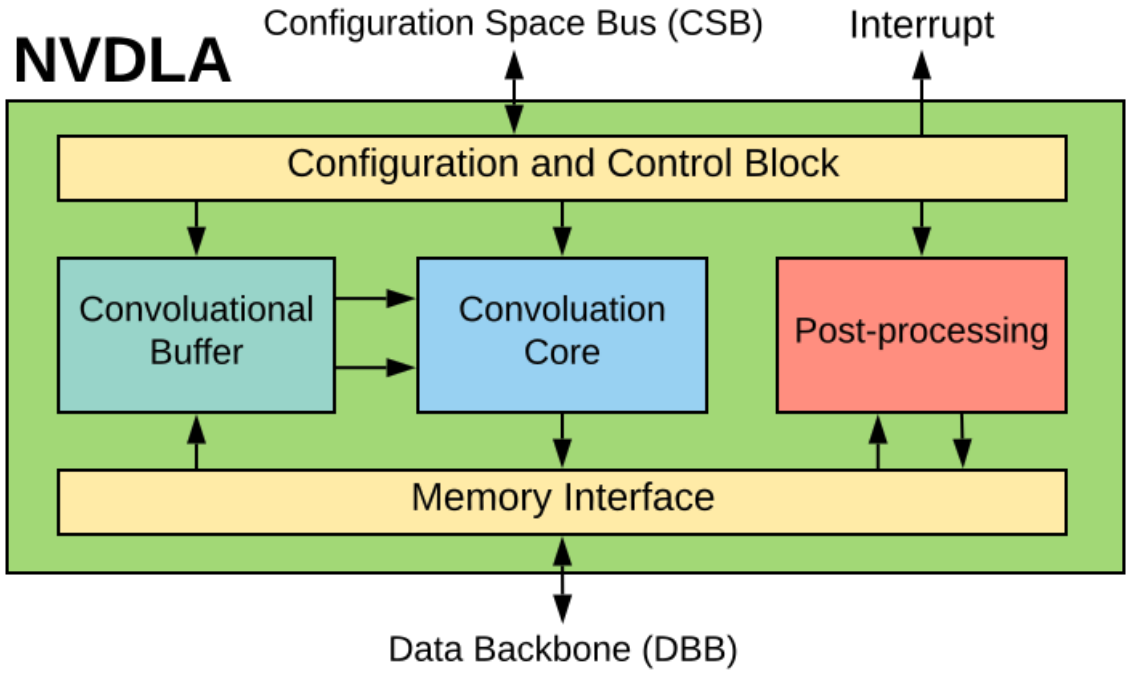
Integrate Your Own Accelerator

- Any accelerator can be integrated (if it fits inside FPGA)
- Develop and test software for your accelerator in Linux environment before having the chip in hand
- Get fast and accurate performance results



NVDLA

- **Scalable**: nv_small, nv_medium, nv_large
- We used **nv_large**: 2048 MACs
- Convolutional core: matrix-matrix multiplication
- Post-processing: activation function, pooling, etc.



Adopted from “The Nvidia Deep Learning Accelerator”, https://goo.gl/Znyba5

Performance Analysis (I)

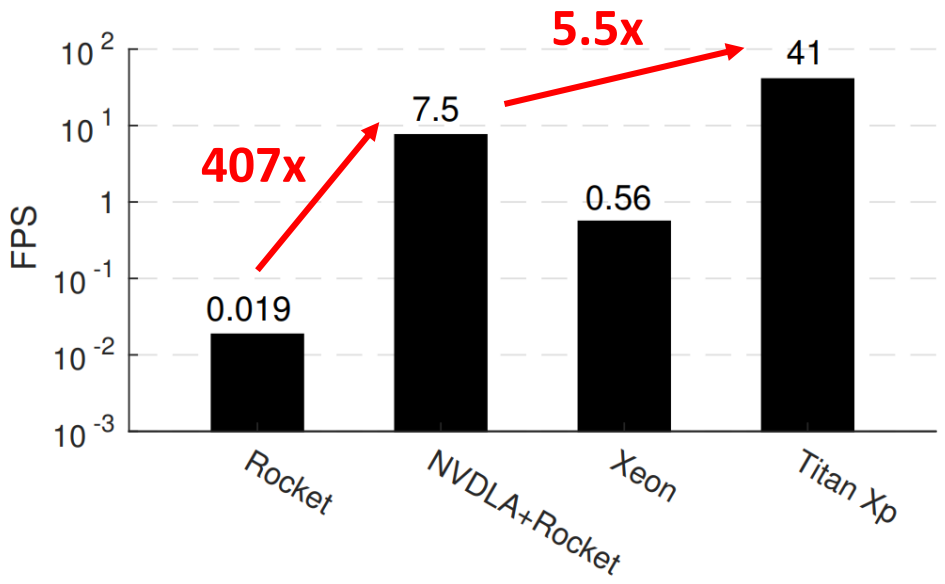
- Baseline config:
  - Quad-core **Rocket Core**, 3.2 GHz
  - **NVDLA**: 2048 INT8 MACs, 512 KiB conv. buffer, 3.2 GHz
  - **LLC**: Shared 2 MiB, 8-way, 64 B block
  - **DRAM**: 4 ranks, 8 banks, FR-FCFS
- YOLOv3: 416 x 416 frame, 66 billion operations

Performance Analysis (II)

- Frame process time: 133 ms (**7.5 fps**)
  - 67 ms on NVDLA
  - 66 ms on processor, multithreaded with OpenMP
- Layers not supported by NVDLA are running on processor
  - Custom YOLO, upsampling, FP ⇔ INT8
- Make common DNN algorithm run very fast ✔
- Computations not supported by the accelerator can make you slow ✘

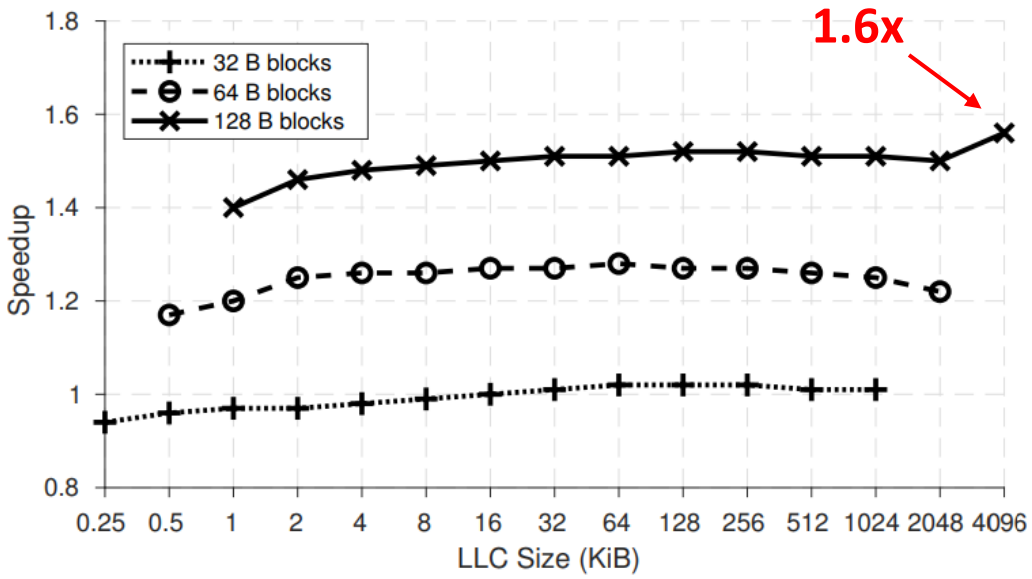
Performance Comparison

- **Rocket**: baseline config, no NVDLA
- **NVDLA+Rocket**: baseline config
- **Xeon**: E5-2658 v3
- **Titan Xp**: Pascal arch, 3840 CUDA cores
- Titan cosumes more power
  - Titan Xp: board TDP **250 W**, 471 mm² in 16nm
  - NVDLA IP: **766 mW** peak, 3.3 mm² in 16nm



Sharing LLC with Accelerator

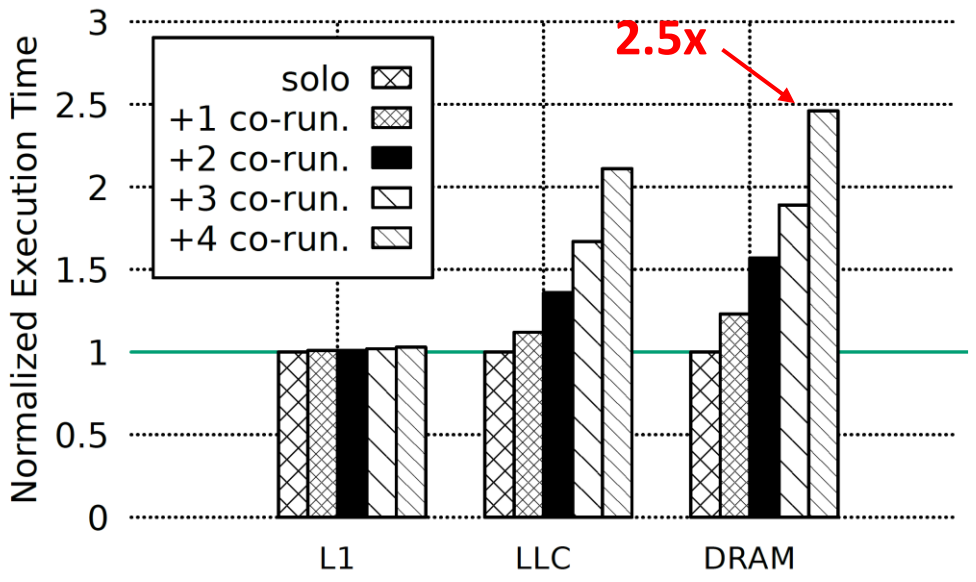
- Sharing the LLC can be a good alternative to scratchpad
  - Consumes less **chip area**
  - Less **programming effort**
- Performance does not vary by changing the **LLC size**
- But varies by changing the **block size**
- **Streaming** access pattern. Not much data reuse left
- NVDLA minimum burst length: 32B
- Hardware prefetcher should help



* Speedup is measured w.r.t design with no LLC

Contention In Memory System

- We care about worst-case execution time in real-time systems
- Synthetic benchmark is running on the CPU stressing the memory system
- NVDLA execution time is measured



* Normalized to solo execution time i.e. running in isolation

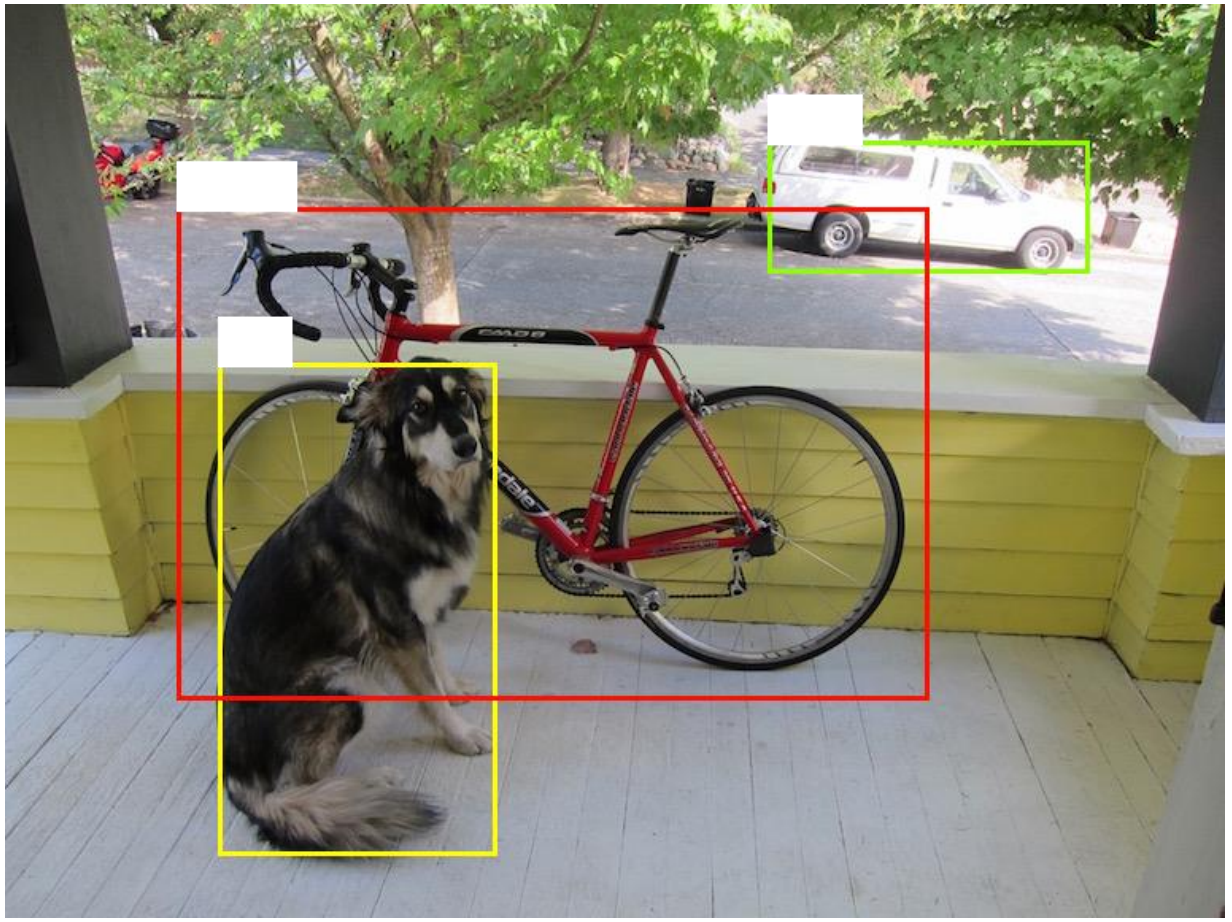
Conclusion

- We integrated NVDLA with a RISC-V SoC on FireSim
  - Fast, easy-to-use
  - No FPGA board needed: runs on the Amazon could
  - Can be used for architectural/system research
- We will be using it for research in real-time embedded systems
- Open-sourced and publicly available at:

https://github.com/CSL-KU/firesim-nvdla/

Google *"firesim nvdla"*

Demo



- Questions?