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

---

Farzad farshchi, Qijing huang, and Heechul yun

Energy Efficient Machine Learning and Cognitive Computing (EMC2), 2019

Presenter: Constant (Sang-Soo) Park

http://esoc.hanyang.ac.kr/people/sangsoo_park/index.html

September 23, 2020

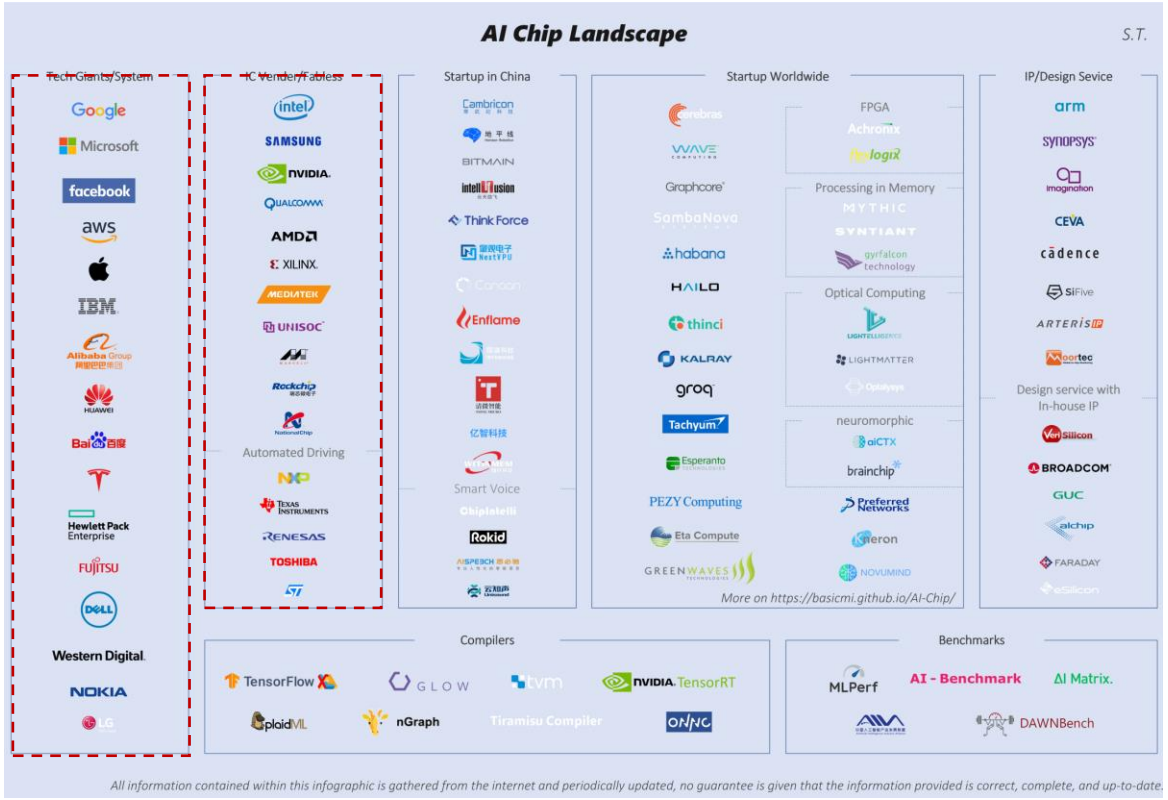
Neural Network Acceleration Study Season #3

Contents of presentation

- **Introduction and Background**
  - Motivation, Background knowledge
- **Key contribution: LLC + Memory Model**
  - System-on-Chip (SoC) architecture
  - Memory configuration for optimized computing performance
- **Performance**
- **Conclusion and Discussion**

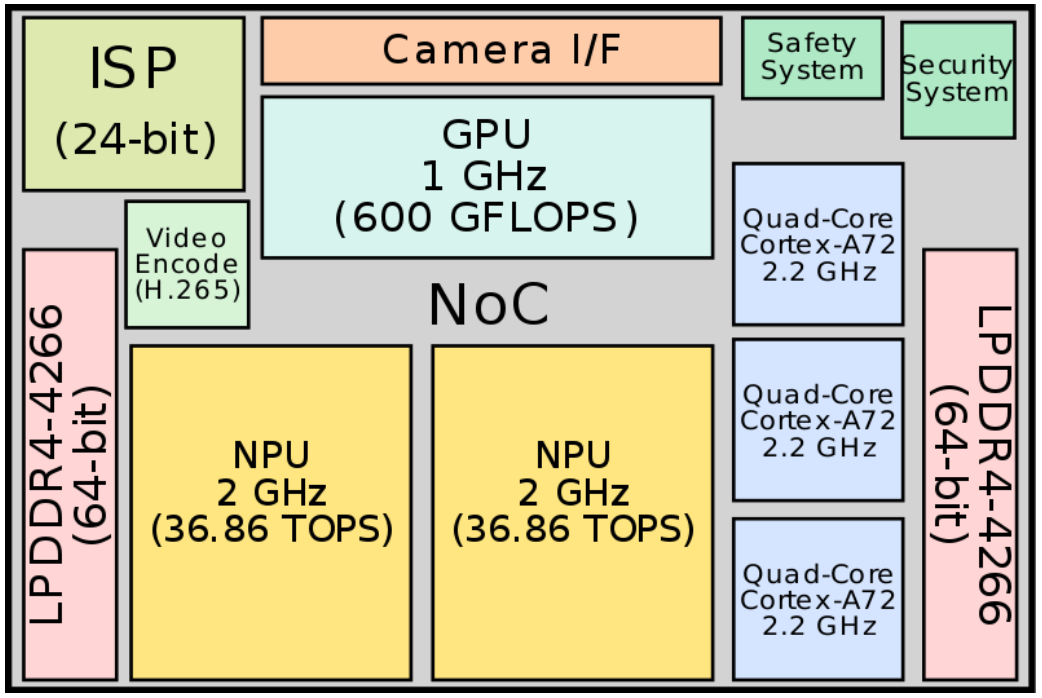
Now is the era of NPU #1

- **Neural Processing Unit (NPU)**
  - Dedicated accelerator for inference/training neural network
  - Three mainstreams (ASIC/FPGA, PIM, Neuromorphic)

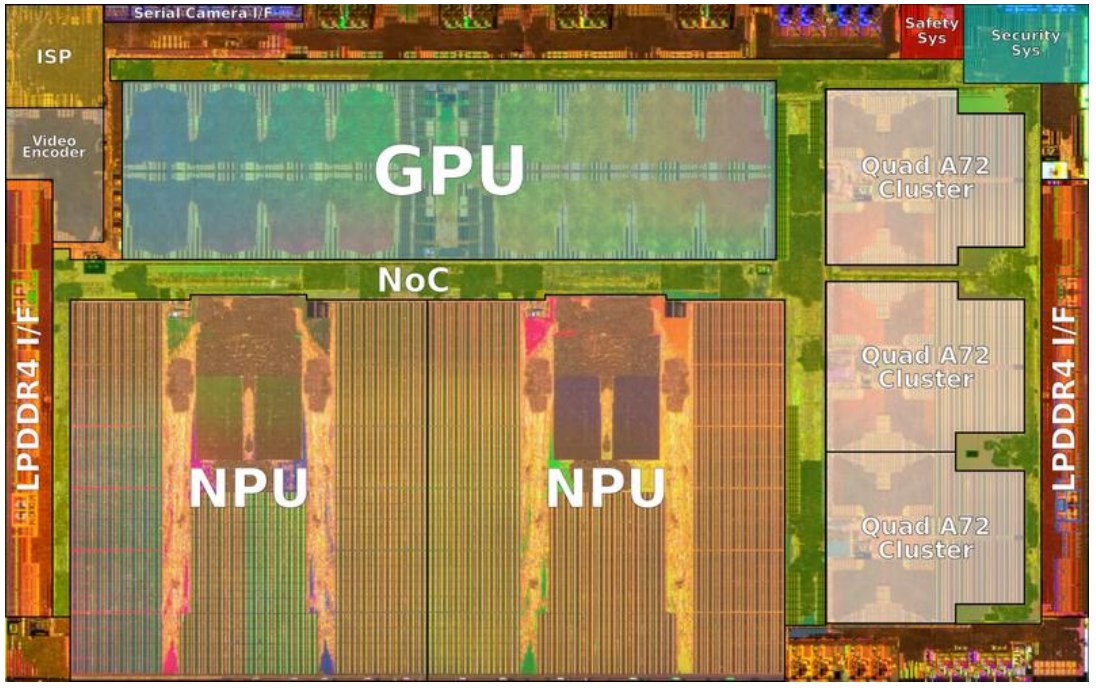


Now is the era of NPU #2

- **NPU in various place in our lives**
  - Self-driving car: TESLA (FSD Chip)[1]
  - Home appliance: Samsung, LG (Robotic vacuum, TV, etc.)[2]
  - Smartphone: AI assistance (Bixby, SIRI), Signal processing



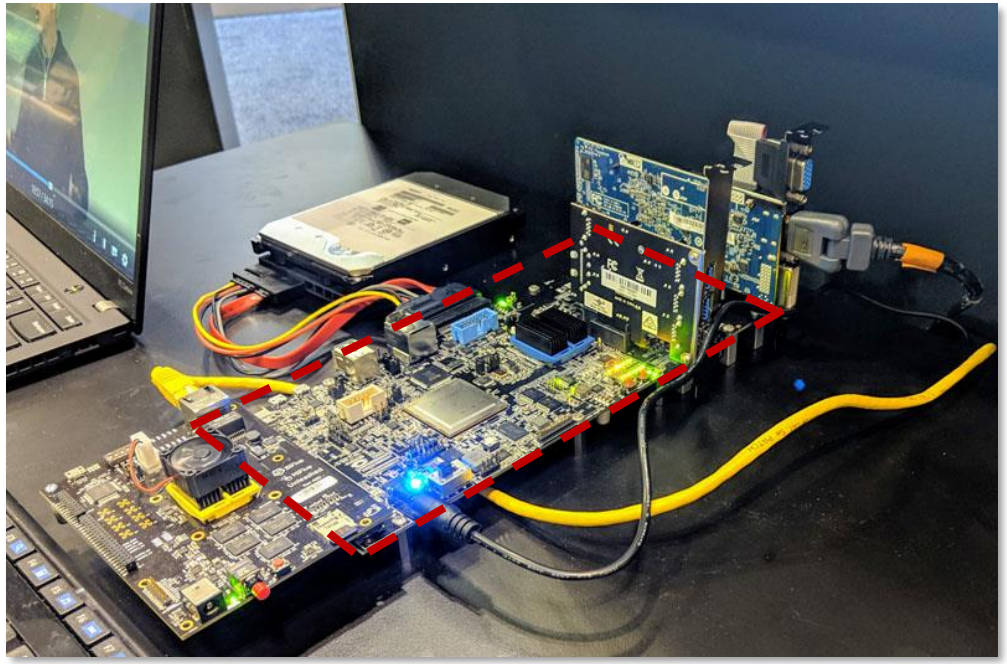
FSD Chip Block Diagram

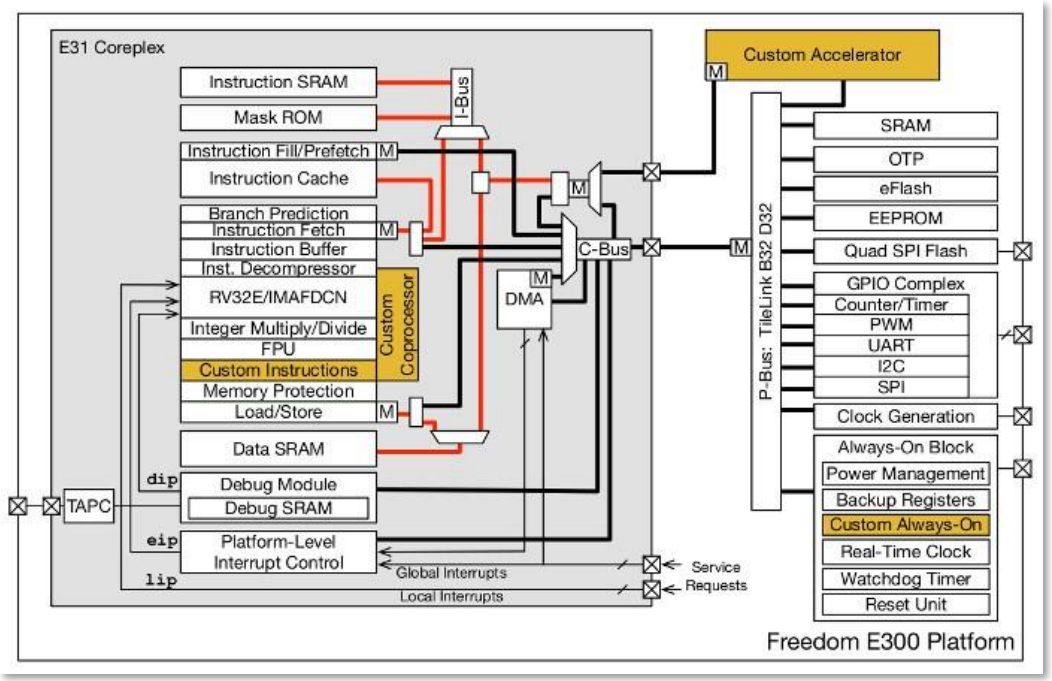


FSD Chip Die Photo

Motivation of paper

- **Useful platform for research**
  - **SoC platform**: integration of NVDLA and MCU (RISC-V)
  - Limitation
    - High cost of Field programmable gate array (**FPGA)** (about 7k cost) → **FireSim**
    - Opensource SoC platform not support an **L2 cache** (limited configuration) → **Chisel**





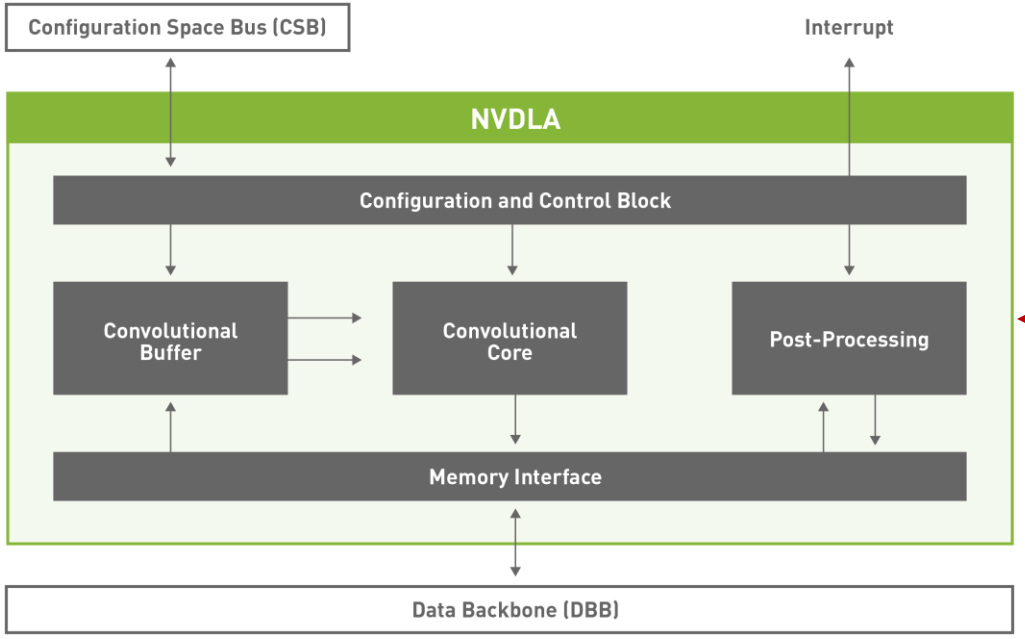
SoC platform running on FireSim

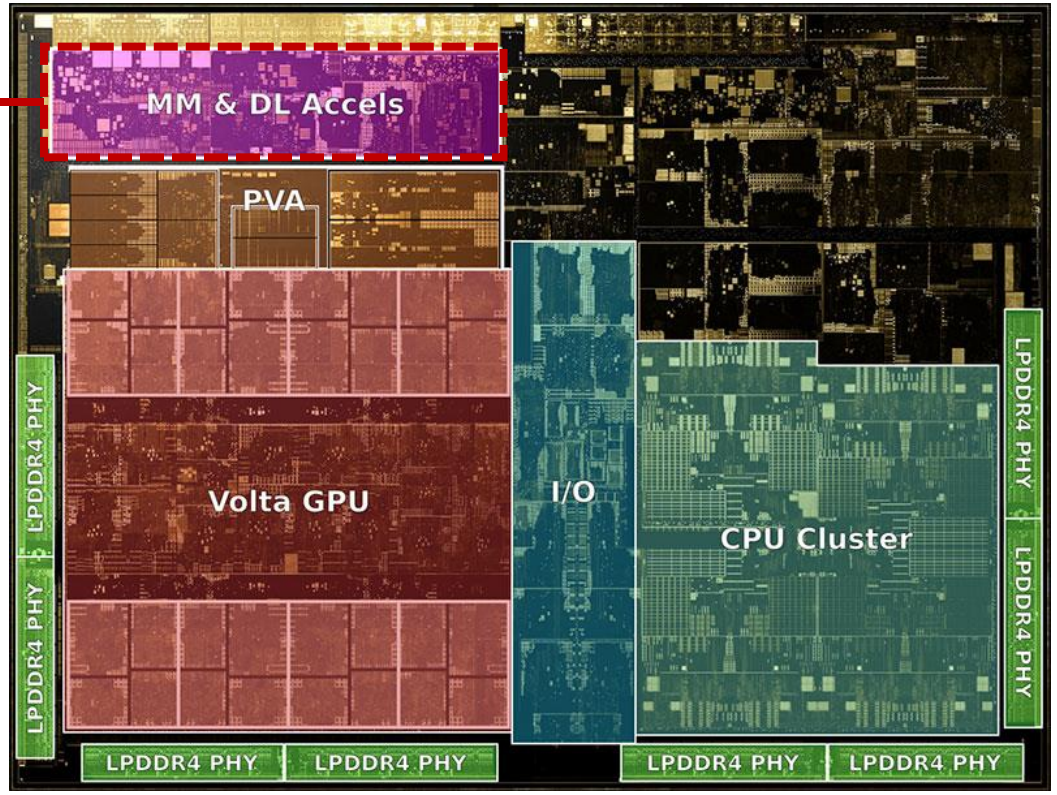
- **DNN accelerator being incorporated into embedded system-on-chips (SoCs)**
  - NVDLA (NVIDIA Deep Learning Accelerator) inside Xavier SoC platform
  - SiFive's RISC-V SoC (Rocket Chip), Freedom U540 SoC platform
  - FireSim: FPGA-accelerated full-system simulator running Amazon cloud FPGAs

We believe SiFive's integration of NVDLA is especially a **useful platform for conducting research** thanks to its opensource nature.

NVDLA (NVIDIA Deep Learning Accelerator)

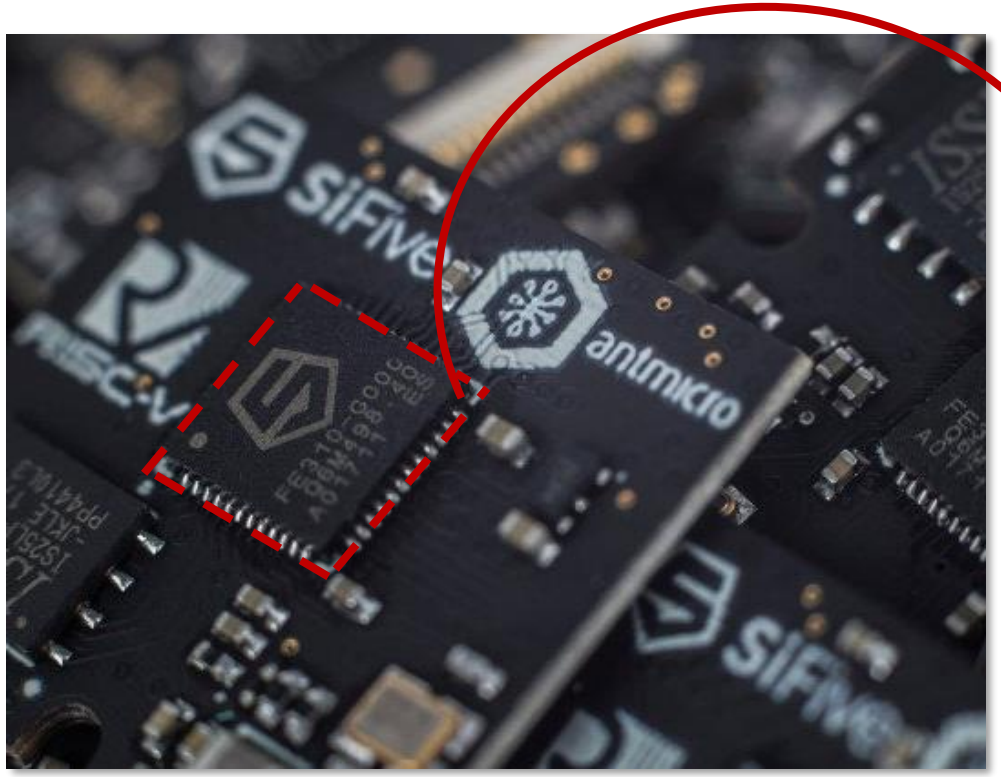
- **Microarchitecture designed by NVIDIA for acceleration DNN**
  - Specifically optimized for CNN (workloads deal with images and video)
  - Targeting edge devices, IoT application, and other lower-power inference designs

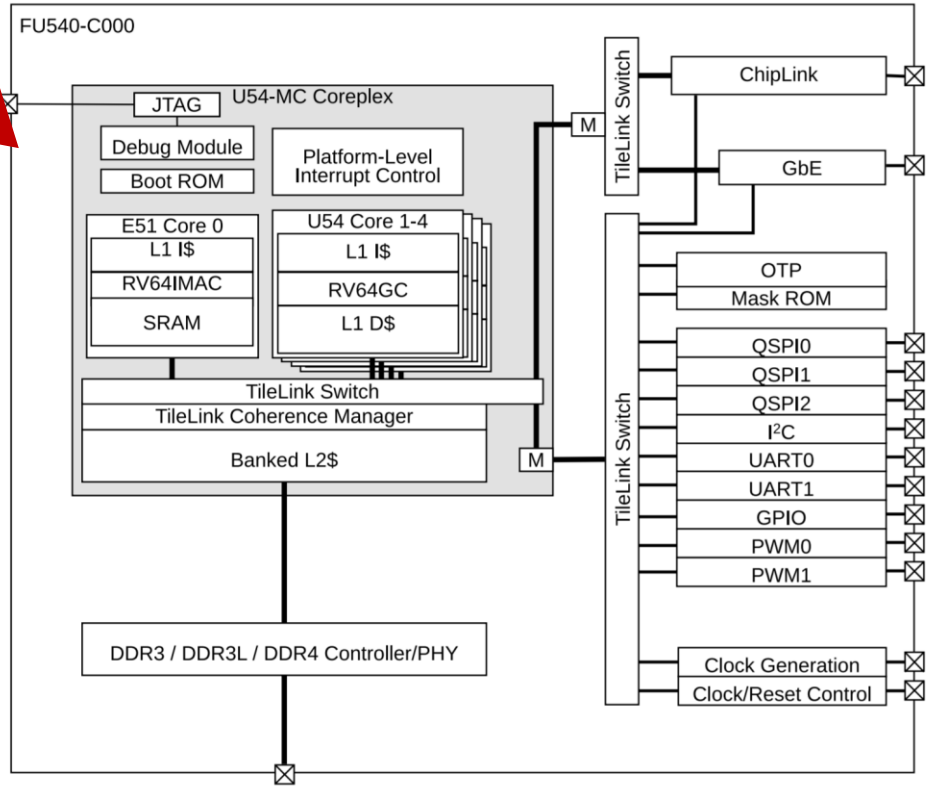




Rocket Chip (NVIDIA Deep Learning Accelerator)

- **Opensource RISC architecture**
  - Expandable instruction set architecture (ISA) for RISC processor
  - Inside SSD controller, 5G modem, IoT sensor, power-management IC (PMIC)
  - Implemented by chisel language

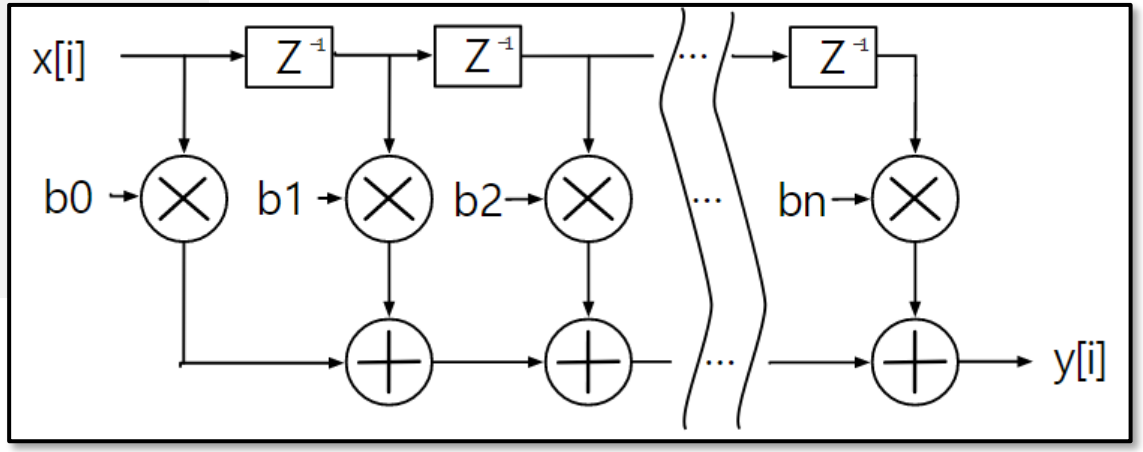




Chisel

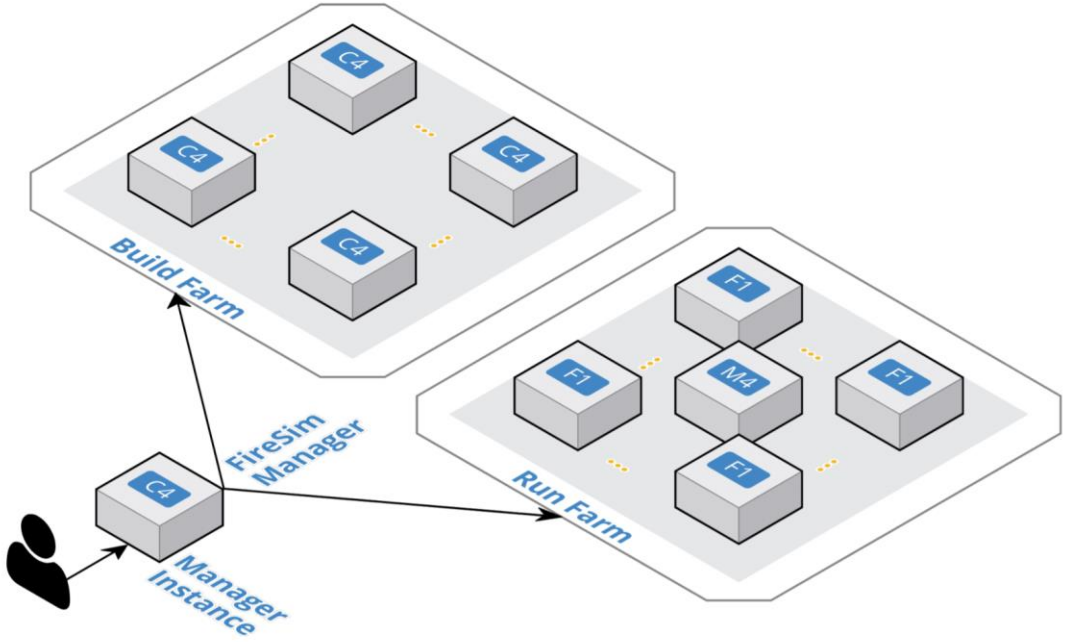
- **HW description language for advanced circuit generation and design reuse**
  - Productive Hardware description language (HDL)
    - Instance port mapping, Instance reuse
    - Designer can design hardware architecture like high-level programming (Scala)

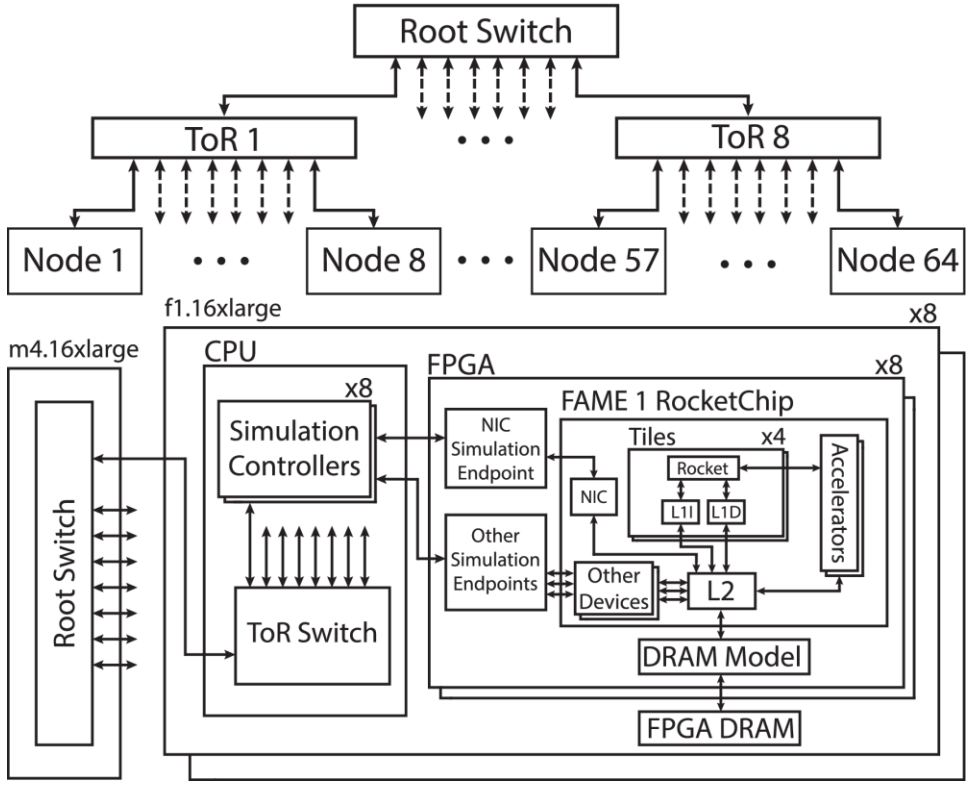
```scala
class FirFilter(bitWidth: Int, coeffs: Seq[UInt]) extends Module {
  val io = IO(new Bundle {
    val in = Input(UInt(bitWidth.W))
    val out = Output(UInt(bitWidth.W))
  })
  // Create the serial-in, parallel-out shift register
  val zs = Reg(Vec(coeffs.length, UInt(bitWidth.W)))
  zs(0) := io.in
  for (i <- 1 until coeffs.length) {
    zs(i) := zs(i-1)
  }

  // Do the multiplies
  val products = VecInit.tabulate(coeffs.length)(i => zs(i) * coeffs(i))

  // Sum up the products
  io.out := products.reduce(_ + _)
}
```



FireSim

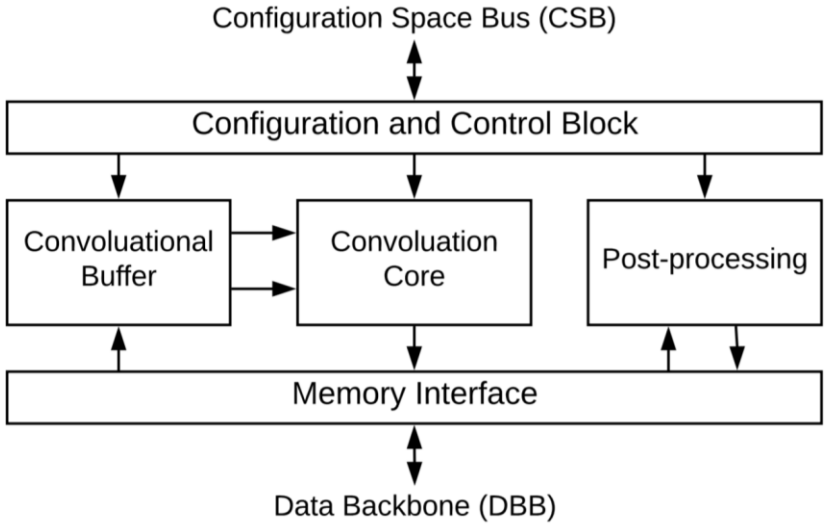
- **FPGA-accelerated cycle-accurate HW simulation in cloud**
  - Running HW design at near-FPGA-prototype speeds on cloud FPGAs
  - Obtaining cycle-accurate performance results (10~100s of MHz frequency)
  - Rate for on-demand access ($1.65 per hour)

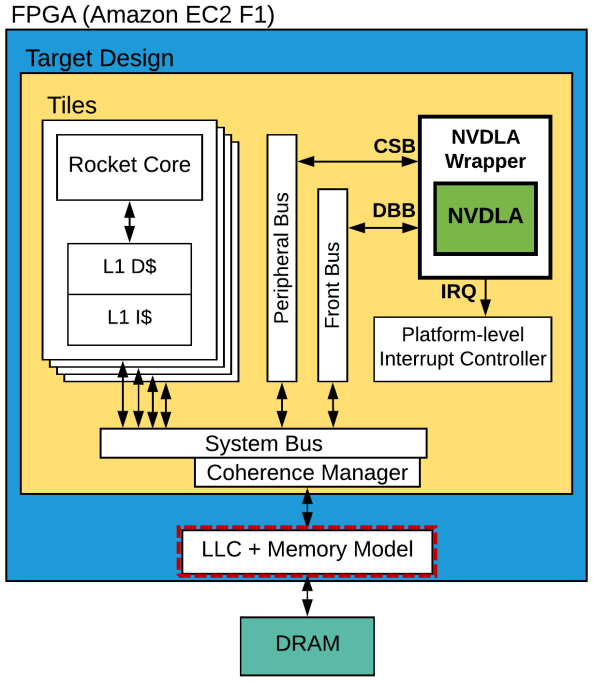




Key contribution: NVDLA integration

- **NVIDIA DLA (NVDLA)**
  - Buffer (feature map, weight), Core (MAC units), Post-processing (activation, pooling)
  - AMBA* based design, Control signal (CSB), Data signal (DBB)
- **NVDLA integration**
  - TileLink based bus protocol, Bride connection (Peripheral, Front)
  - **Configurable LLC and Memory Model (# of sets, ways, and block size)**





Performance Analysis #1

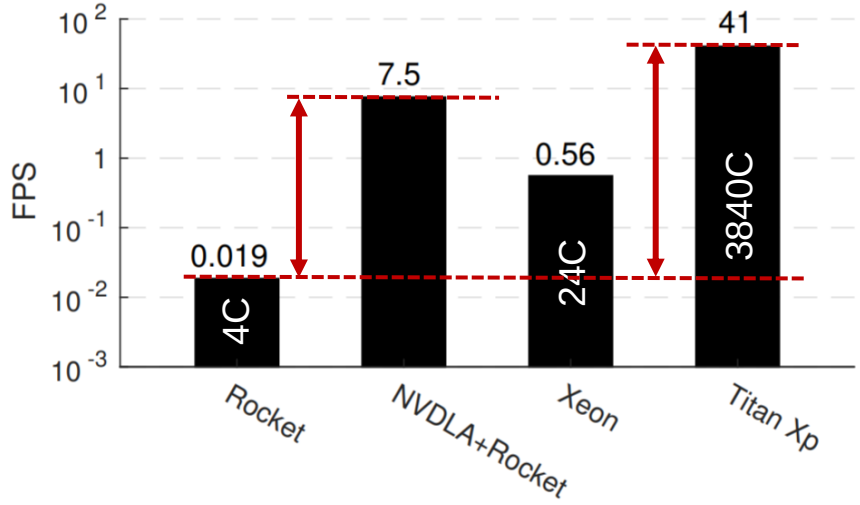
- **Experiment Environment**
  - Benchmark: YOLO v3 object detection algorithm
    - 66 billions operation to process 416 by 416 frame
  - Be operated at lower frequency
    - NVDLA block is clocked at the same frequency with the processor (FPGA)
  - **Baseline config**
    - Quad-core Rocket Core, 3.2GHz with 2048 INT8 MACs (FP <-> INT8)

Processor	Quad-core, in-order, single-issue, 3.2 GHz
NVDLA	2048 INT8 MACs, 512 KiB buffer, 3.2 GHz
L1 I/D$	Private 16/16 KiB, 4-way, 64 B block
LLC	Shared 2 MiB, 8-way, 64 B block
DRAM	16 GiB DDR3, 4 ranks, 8 banks, FR-FCFS

Performance Analysis #2

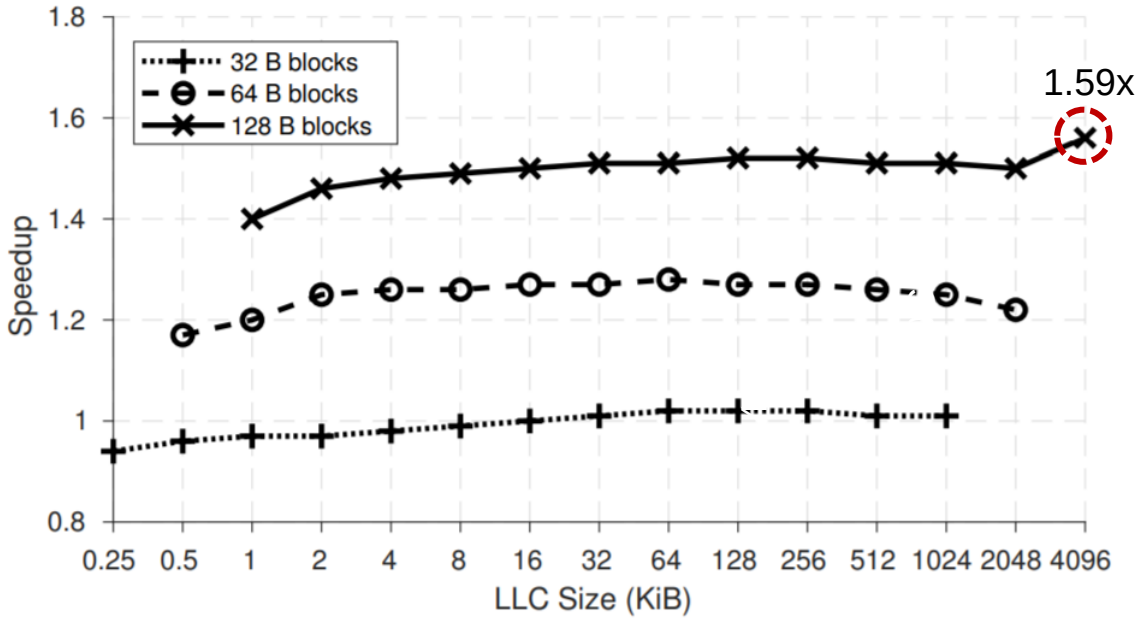
- **Performance comparison**
  - Baseline performance of NVDLA
    - 67ms on NVDLA + 66ms on processor (multithreaded with OpenMP)
    - Layers not supported by NVDLA are running on processor
  - Titan consumes more power
    - Titan Xp: 250W (TDP), 471mm^2 in 16nm technology
    - NVDLA: 766mW, 3.3mm^2 in same technology

Processor	Quad-core, in-order, single-issue, 3.2 GHz
NVDLA	2048 INT8 MACs, 512 KiB buffer, 3.2 GHz
L1 I/D$	Private 16/16 KiB, 4-way, 64 B block
LLC	Shared 2 MiB, 8-way, 64 B block
DRAM	16 GiB DDR3, 4 ranks, 8 banks, FR-FCFS



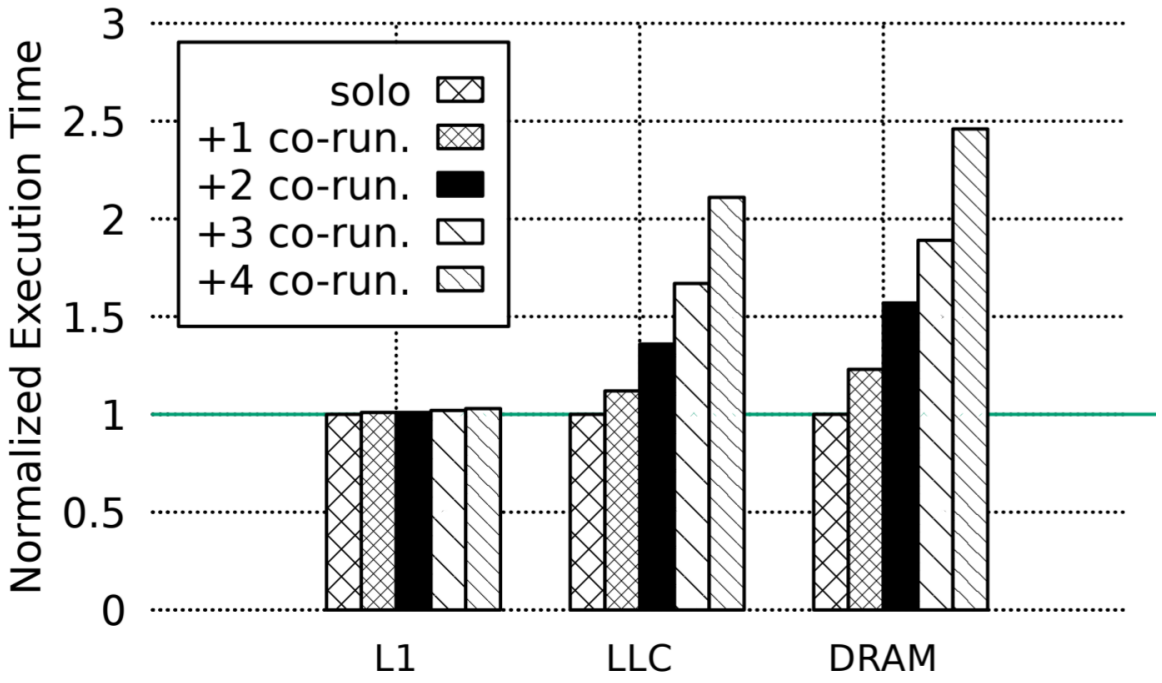
Performance Analysis #3

- **Effect of Last-level Cache on Performance**
  - Sharing LLC can be good alternative to scratchpad
    - Chip area saved, less programming effort
  - Varying LLC size and measuring NVDLA speedup
    - Also, size of cache block (32B, 64B, and 128B)
    - NVDLA is not very sensitive to LLC size
    - **Temporal locality** in NVDLA memory accesses pattern
    - Target>Minimum burst length (32B), reducing latency
  - Hardware prefetching improving NVDLA performance
    - Benefit of sharing LLC capturing spatial locality



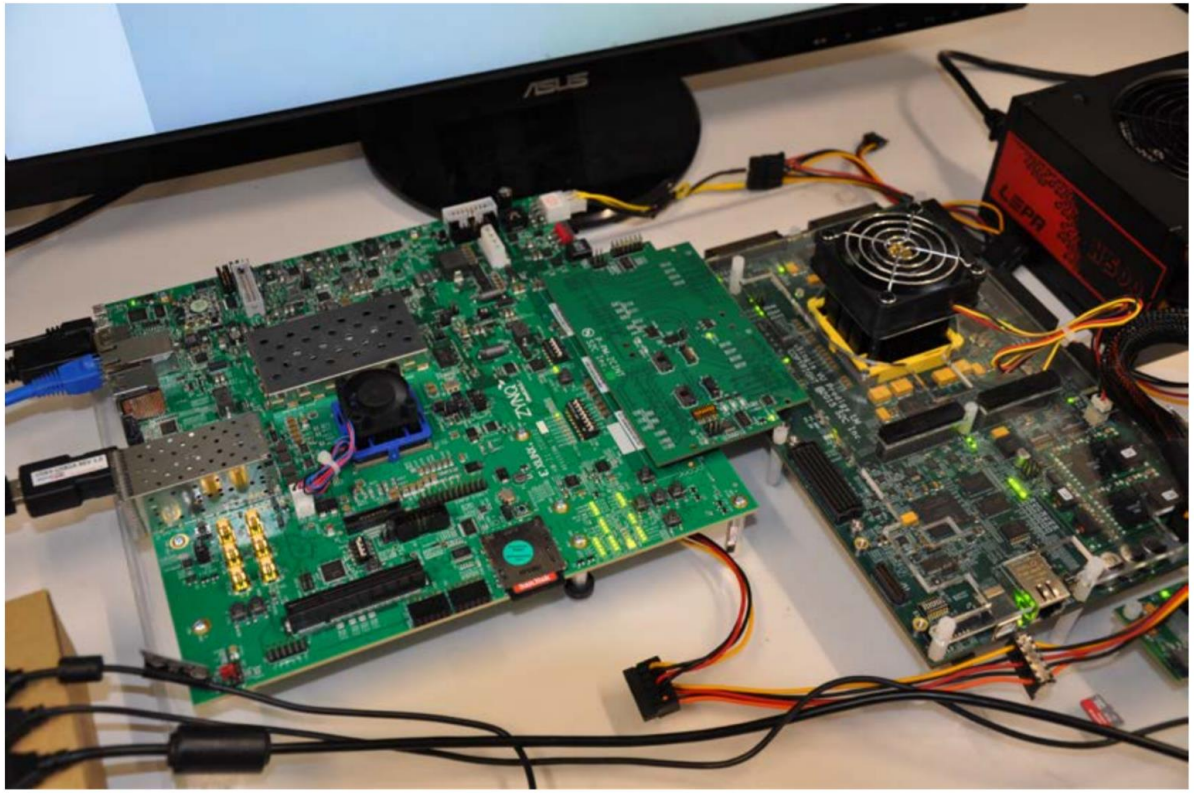
Performance Analysis #4

- **Effect of Shared Memory Interference**
  - NVDLA and CPU are sharing memory system
    - Interference with one another when accessing the memory
    - Unpredictable latency in execution of tasks running on NVDLA
  - Bandwidth Write (BwWrite) benchmark to study interference caused by processor
    - Co-schedule BwWrite with YOLO v3 running on NVDLA
  - L1 cache (No slowdown)
    - Own private data cache and WSS
    - LLC and DRAM: 2.5x for co-runners



Conclusion

- **First integration platform of RISC-V + NVDLA**
  - NVDLA provides good acceleration performance, especially considering its low power consumption
  - Larger cache block size and/or hardware prefetcher is desirable for performance
  - Impact of shared memory interference between CPU and NVDLA is significant



Thank you