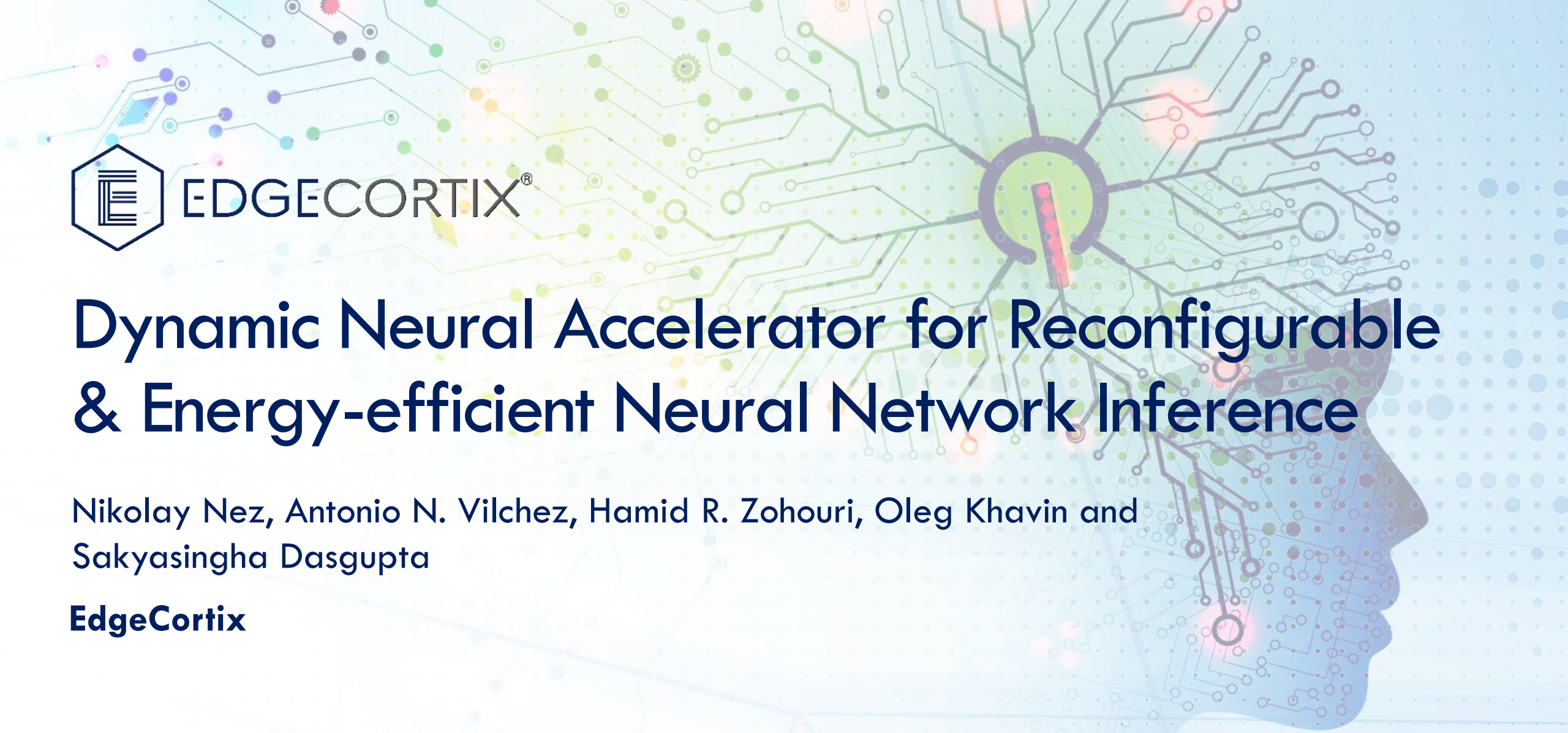


Dynamic Neural Accelerator for Reconfigurable & Energy-efficient Neural Network Inference

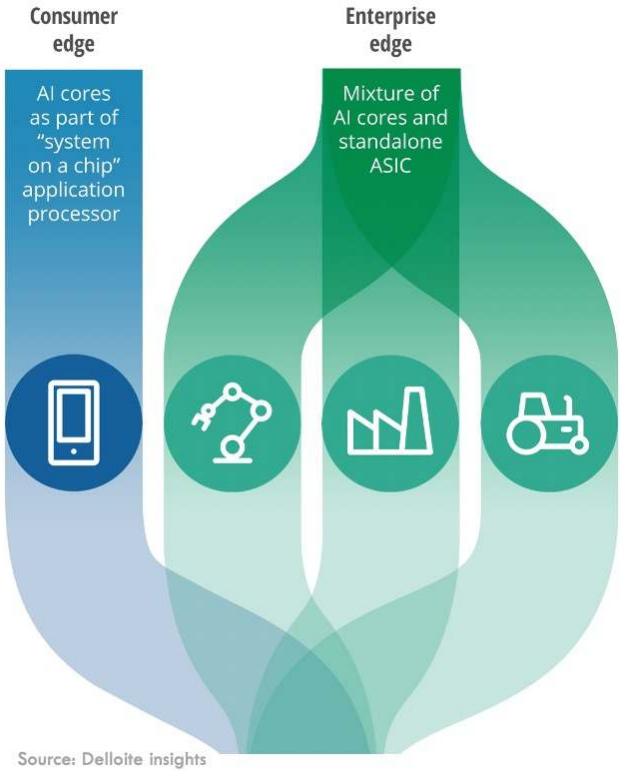
Nikolay Nez, Antonio N. Vilchez, Hamid R. Zohouri, Oleg Khavin and Sakyasingha Dasgupta

EdgeCortix

Hot Chips 33, August 2021

Unique Challenges for AI Inference Hardware at the Edge

Diverse edge AI hardware workloads



Source: Delloite insights

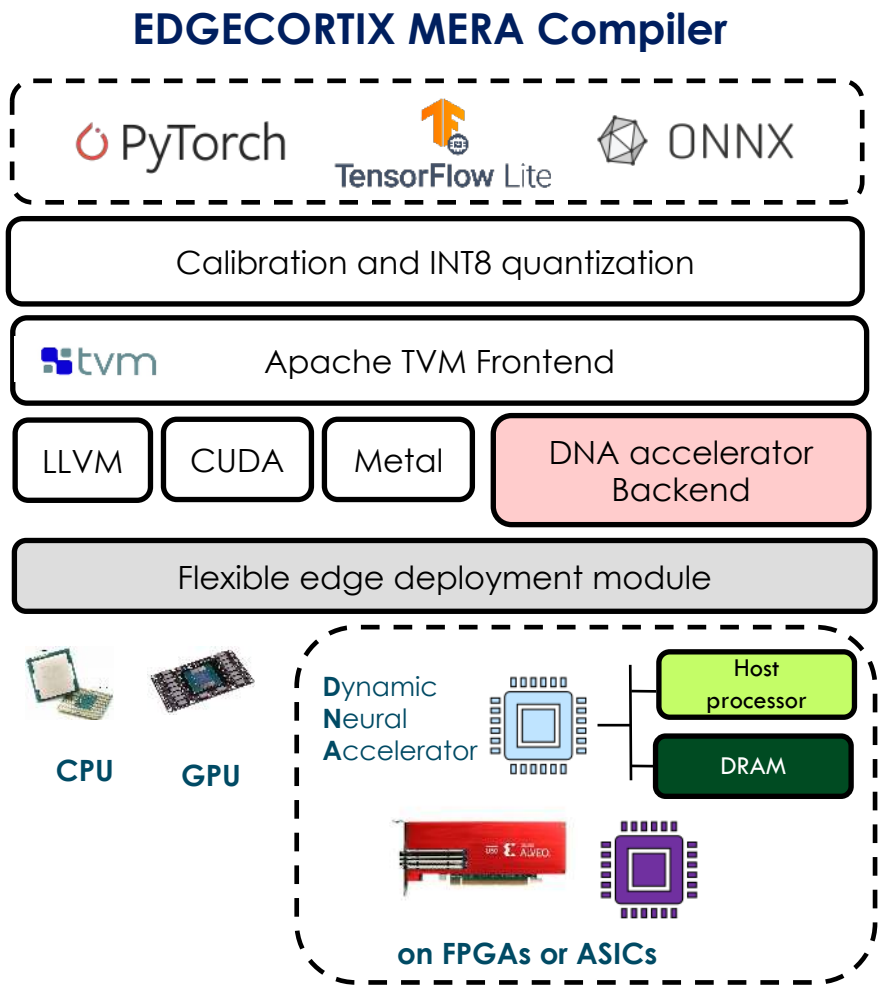
- Peak TOPS or TOPS/Watt are not ideal measures of performance at the edge. Cannot prioritize performance over power efficiency (throughput/watt)
- Many AI Hardware rely on batching to improve utilization. Unsuitable for streaming data (batch size 1) use-case at the edge
- AI hardware architectures that fully cache network parameters using large on-chip SRAM cannot be scaled down easily to sizes applicable for edge workloads.
- Need adaptability to new workloads and the ability to deploy multiple AI models
- AI-specific accelerator needs to operate within heterogenous compute environments
- Need for efficient compiler & scheduling to maximize compute utilization
- Need for high software robustness and usability

Software-defined Approach to AI-Specific Hardware Acceleration

- Part I - Software: Multi-module Efficient Reconfigurable Accelerator (MERA) compiler
  - Efficient scheduling across multiple compute modules → **Multi-module Efficient**
  - Minimize on-chip ⇔ off-chip data movement
  - Early simulation and performance estimation
  - Seamless user experience

- Part II - Hardware: Dynamic Neural Accelerator (DNA) architecture (IP series)
  - Run-time reconfigurability → **Dynamic** Neural Accelerator
  - High compute utilization and power efficiency
  - Performance and power efficiency scalability

Part I - MERA Compiler and SDK



- Extends the Apache TVM deep learning compiler
- Common for FPGA and ASIC
- Python and C++ interfaces
- No custom quantization required: PyTorch and TFLite post-training quantization directly supported
- Automatic offloading of unsupported operators to CPU
- Built-in simulator: slow but accurate performance estimation
- Built-in interpreter: fast functional simulation

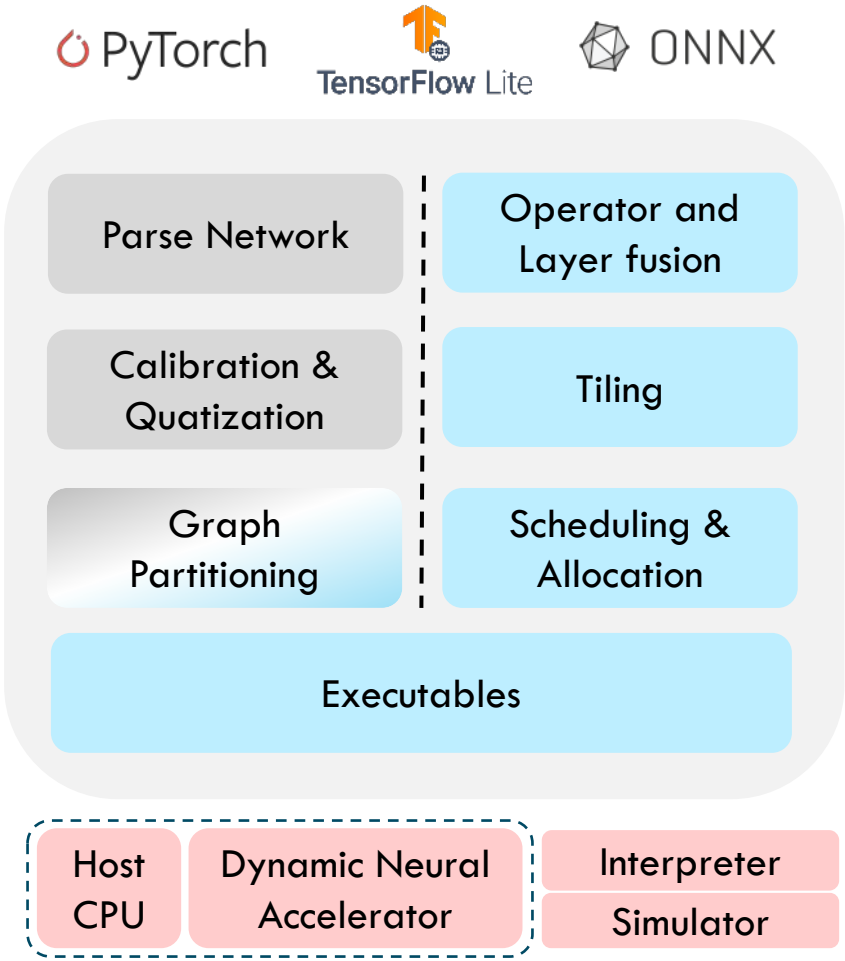
High-level Breakdown of the MERA Compiler Flow

MERA Independent

- Parse PyTorch / TFLite model
- Calibration and quantization
- High-level graph partitioning (Apache TVM)

MERA Dependent

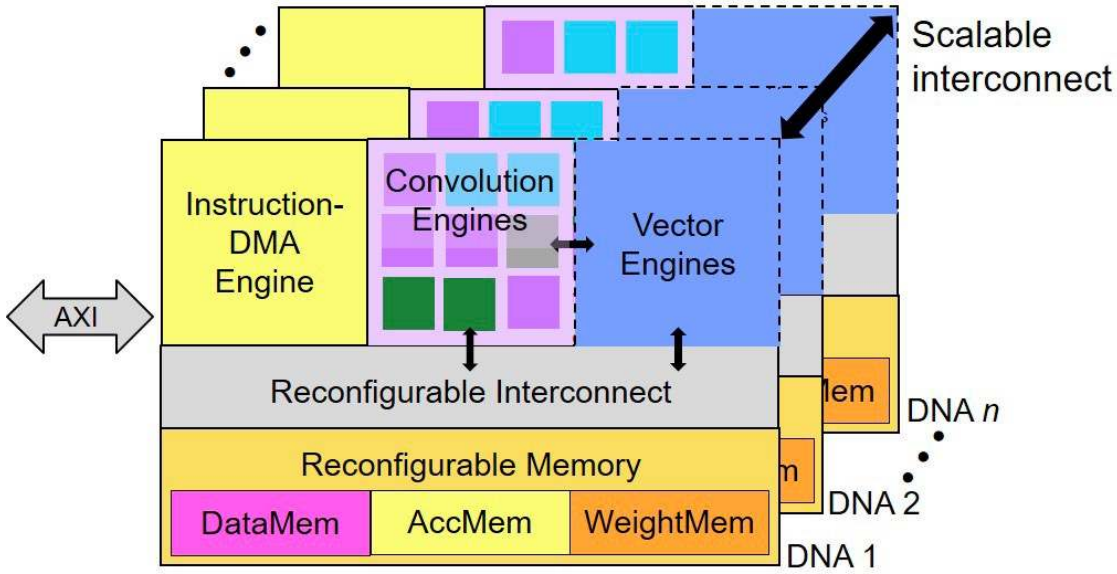
- Low-level graph partitioning
- Target-dependent optimizations
  - Operator and layer fusion
  - Tiling (channel, height, width)
- Scheduling
  - Exploits multiple forms of parallelism
  - Maximizes compute utilization
- Allocator
  - Minimize data spill to external memory



Part II - Dynamic Neural Accelerator Architecture

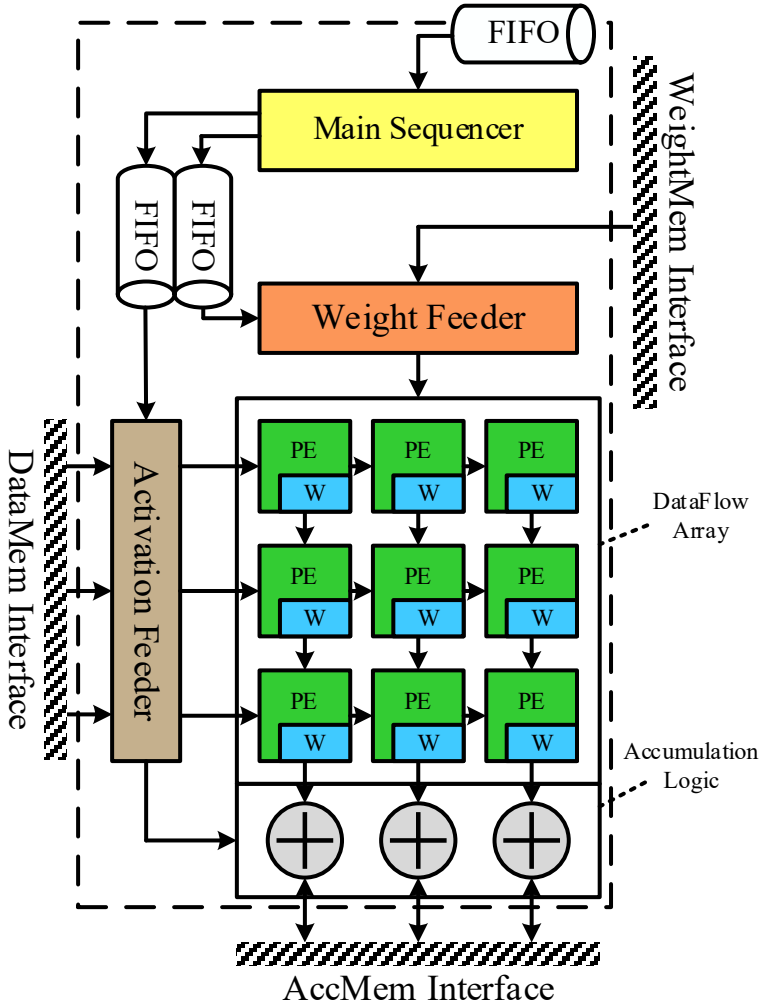
- Designed for low-latency and low-power with high compute utilization.
- Optimized instruction set for INT-8 bit
- Configurable and scalable compute and power
- Parallelism at model, tile, channel, and filter dimensions
- **Optimized for streaming data (batch size 1)**
- **Run-time memory and interconnect reconfigurability**
  - Adapt to varying network layer & model parallelism
- Support for external memory
  - Avoid limiting network support by on-chip memory size

Dynamic Neural Accelerator Overview



Peak Under the Hood of the Convolution Engines - I

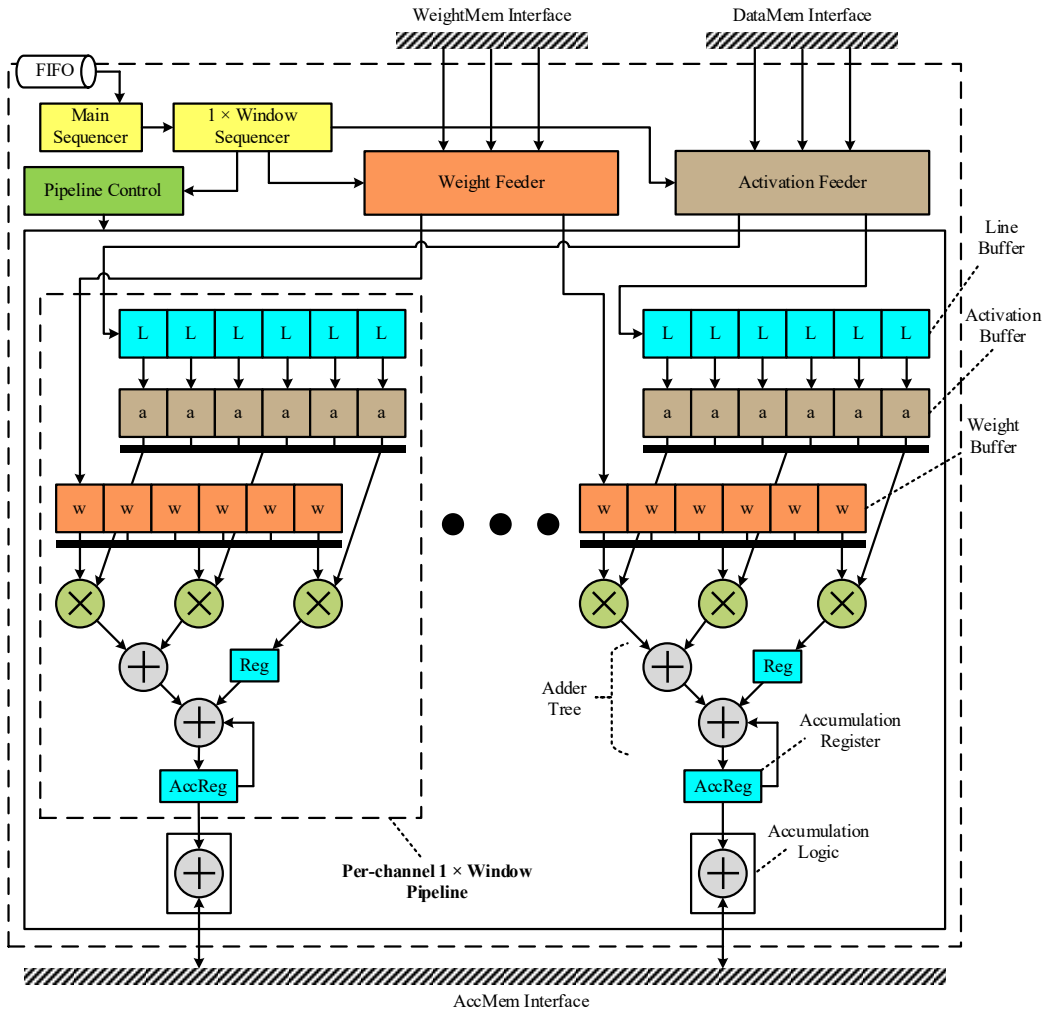
- DataFlow array-based architecture
- Parallelism in input and output channel, and filter row
- Array node has one PE and two weight registers
- Configurable parameters
  - Size of DataFlow array
  - Number of engines
- Example configurations
  - 1 engine of 32x32: 1.6 TOPS @ 800 MHz
  - 8 engines of 64x64: 52 TOPS @ 800 MHz



Peak Under the Hood of the Convolution Engines - II

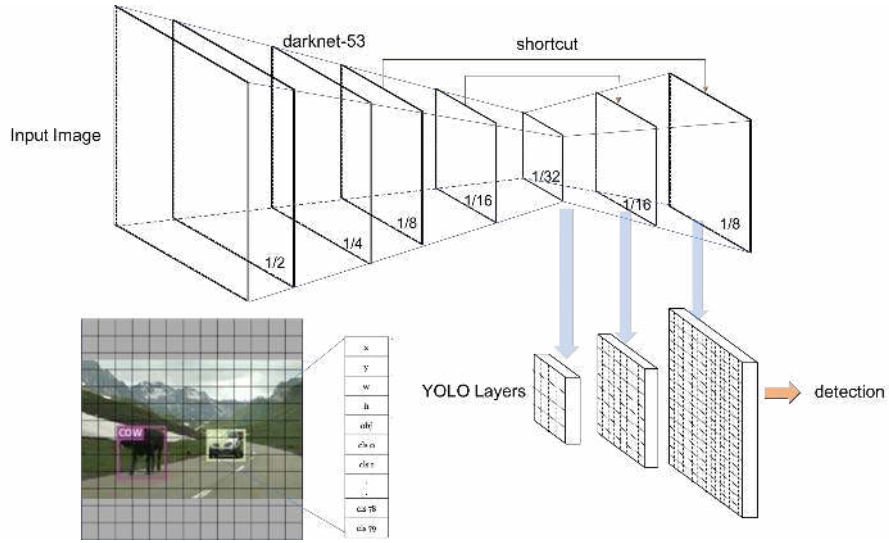
- Multiple-parallel-pipelines design
- Parallelism in output channel and filter row
- Configurable parameters
  - Pipeline width
  - Number of parallel pipelines
  - Number of engines
- Example configuration

  4 engines of 64 3-wide pipelines: 1.2 TOPS @800 MHz



Run-time Reconfigurability in DNA Architecture

- Available degrees of parallelism vary greatly across neural networks and network layers
  - **Early layers**: small channel sizes, large row/column sizes
  - **Middle layers**: moderate channel and row/column sizes
  - **Late layers**: large channel sizes, small row/column sizes
- Fixed hardware parallelism results in low compute utilization in some layers
- Solution: **run-time reconfigurable interconnect and memory structure**
  - Efficiently mix different types of parallelism to improve utilization
- Reconfiguration can happen once per network or for some number of layers

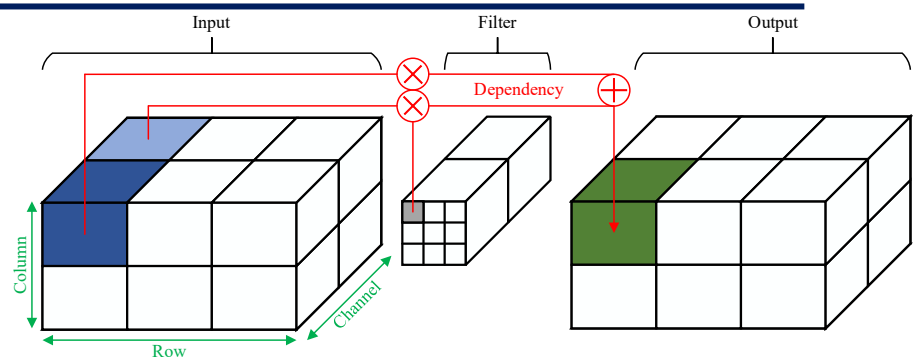


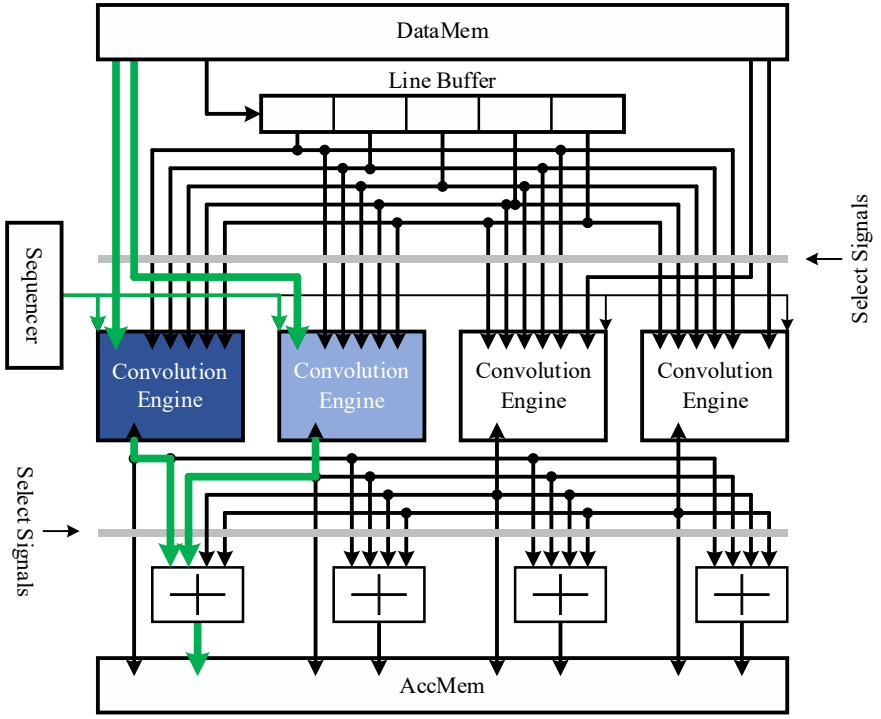
Schematic of the YOLOv3 network architecture

Reconfigurable Interconnect

Purpose

- Changing module connectivity
- Adding or removing modules to/from data flow

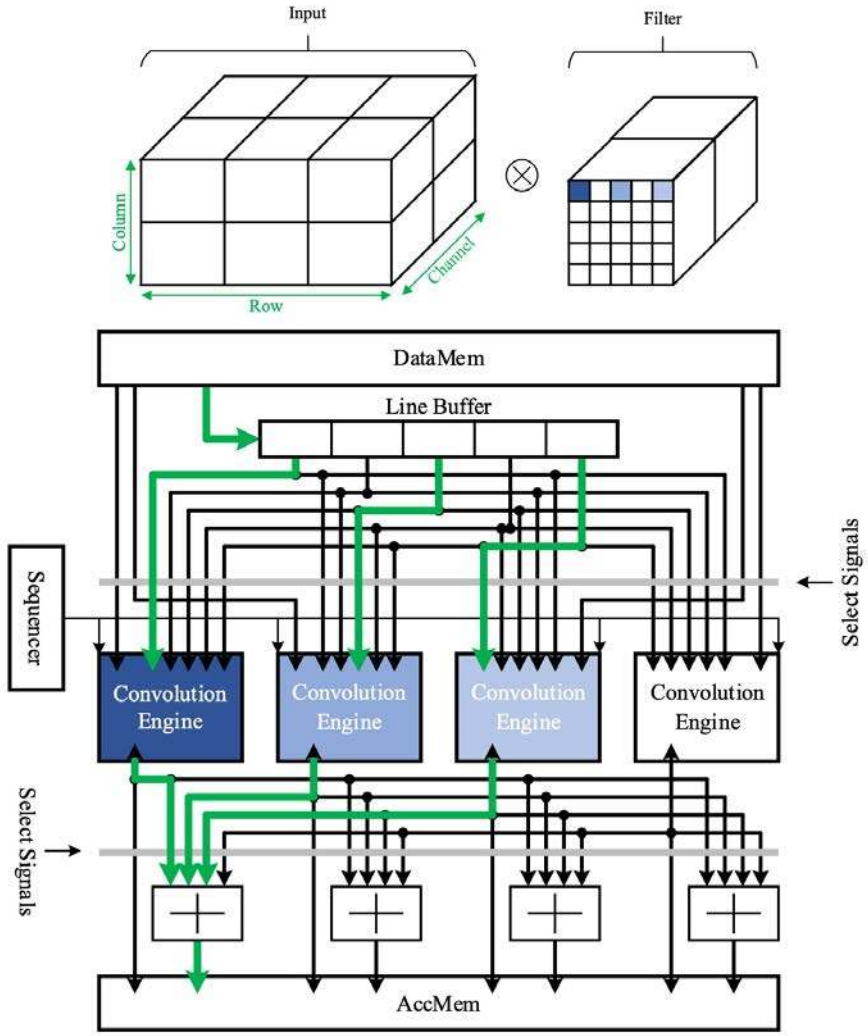




Reconfigurable Interconnect

Purpose

- Changing module connectivity
- Adding or removing modules to/from data flow



Reconfigurable Interconnect

Purpose

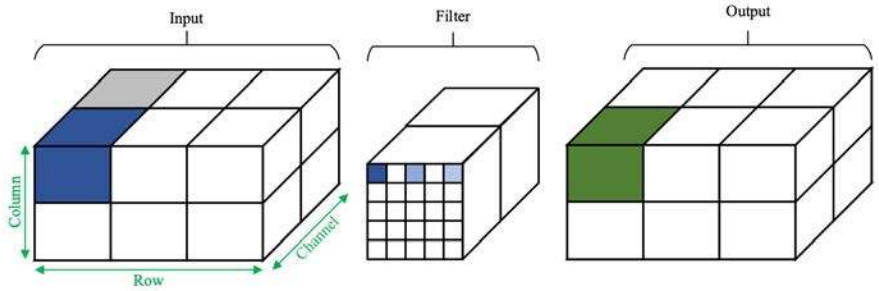
- Changing module connectivity
- Adding or removing modules to/from data flow

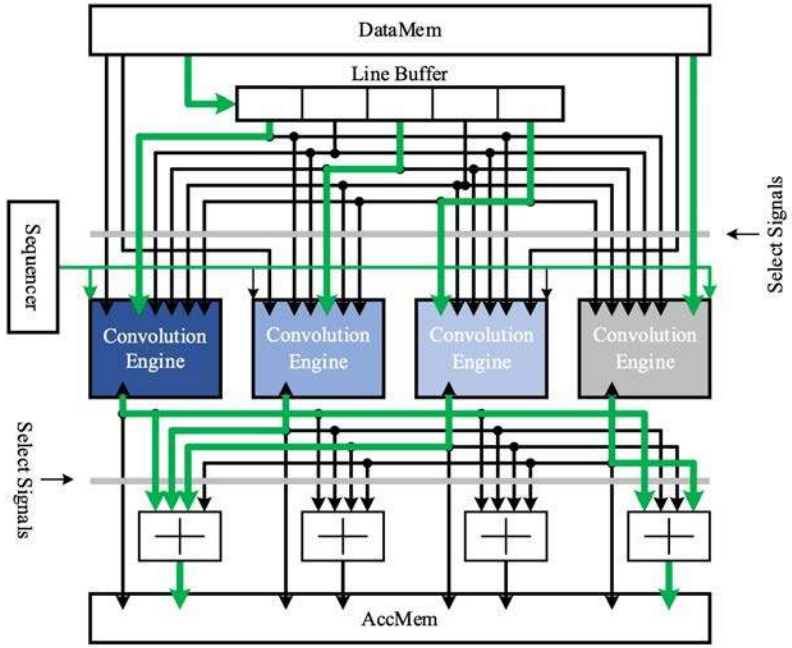
Efficient mixing of different types of parallelism

- Improves compute utilization

Implemented with circuit-switching

- Connectivity determined at compile-time
- Configured at run-time





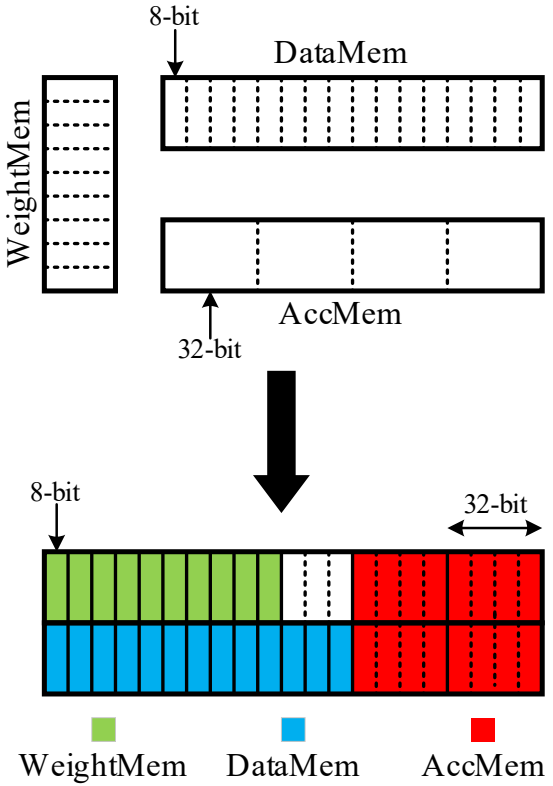
Reconfigurable On-chip Memory Structure

Purpose

- Virtual allocation of different memory types (Weight, Data, Acc.) in same physical memory structure
- Virtual combination of consecutive memory banks to create wider or deeper banks

Efficient on-chip memory allocation

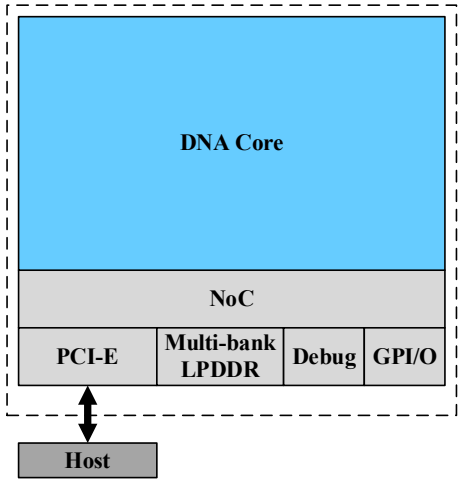
- Directly improves on-chip memory utilization
- Indirectly improves compute utilizations



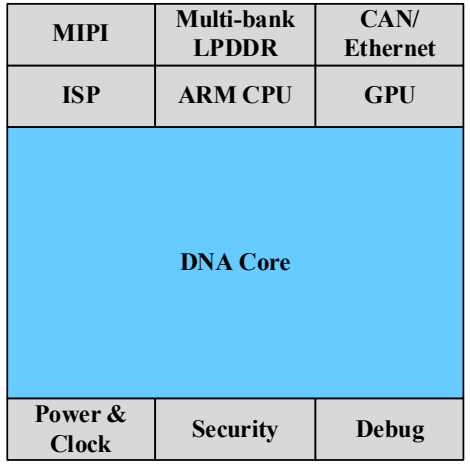
One flexible memory type

DNA IP for ASIC in Different Configurations

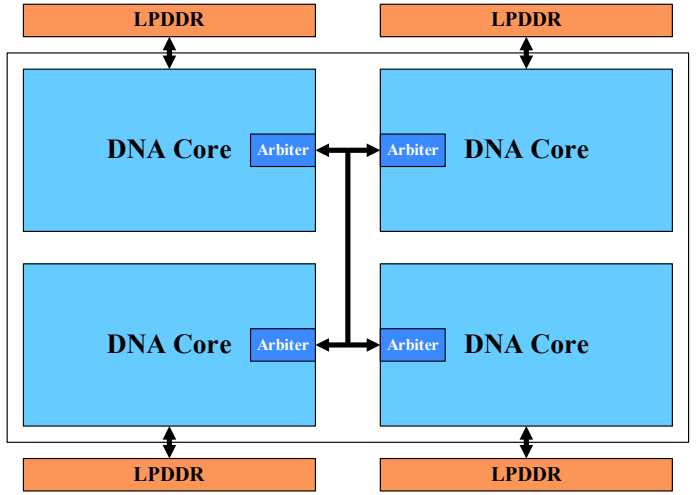
- DNA as a modular and configurable IP series for ASICs
- Typical configurations are provided as part of the DNA-A series
  - Different performance and power efficiency points
  - Ranging from 1.8 TOPS for under 0.6 Watts, to 54 TOPS for under 8 Watts @ 800 MHz (single core DNA)
- Multi-core and multi-chip scalability



PCI-E-based



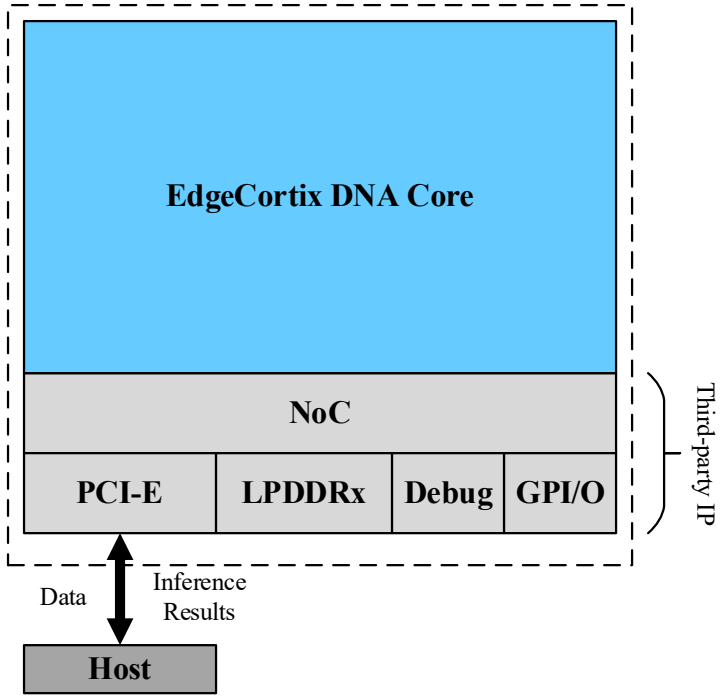
System on Chip



Multi-core

*Grey boxes are third-party IPs

PCI-E Based Dynamic Neural Accelerator Demonstrator Chip

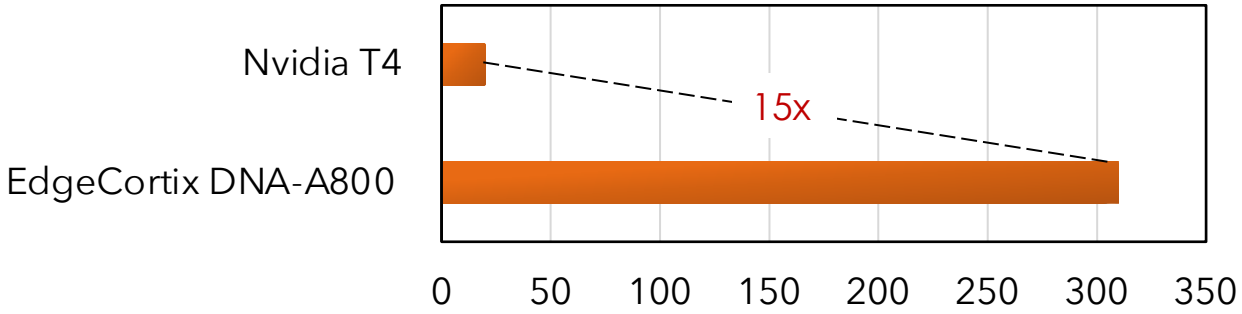


Coming Soon!

mPCI-E based Reconfigurable AI acclelerator chip

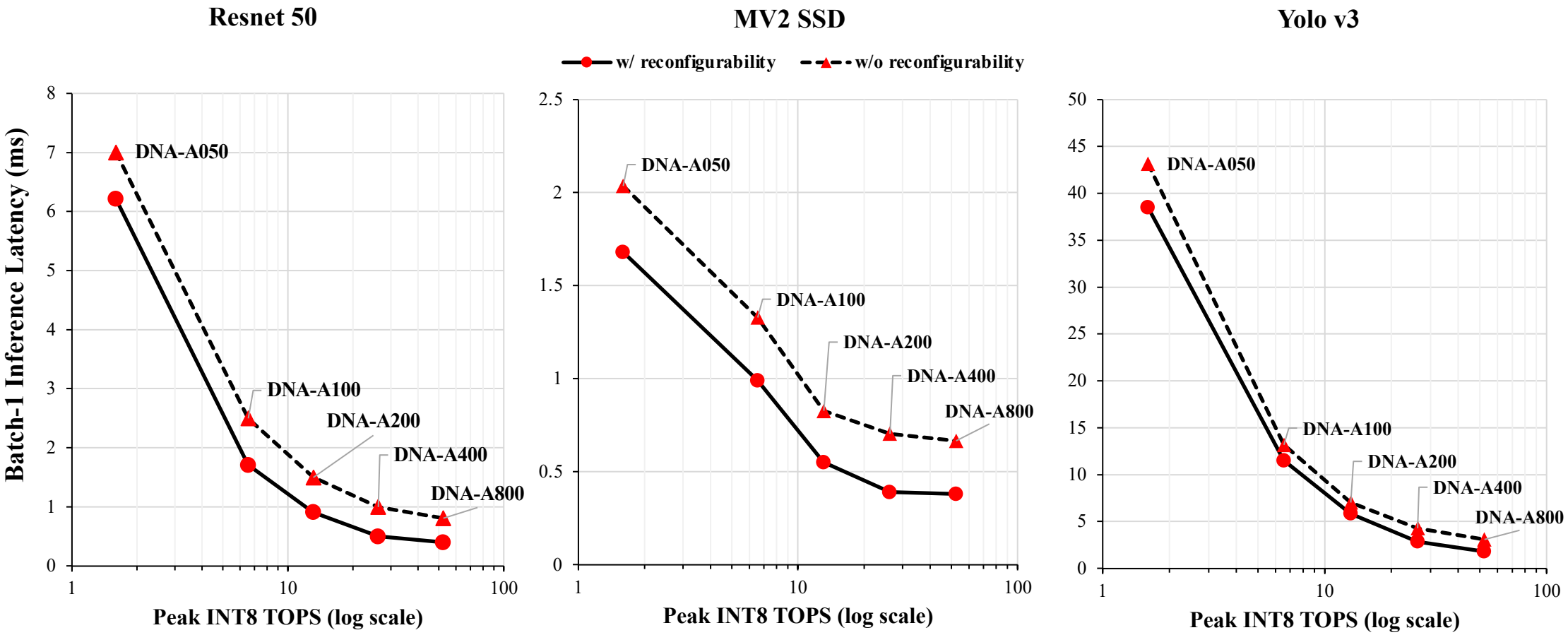
- DNA-A800 Configuration
- TSMC 12 nm @ 800 MHz
- 54 INT8 TOPS peak at under 10 watts
- 16x PCI-E 3.0 and M.2 (4x PCI-E 3.0)
- 2x LPDDR4x 3200 MHz

FPS/Watt at Batch-size 1 (ResNet-50 v1.5)



DNA A-series: Best in Class Batch size 1 Latency

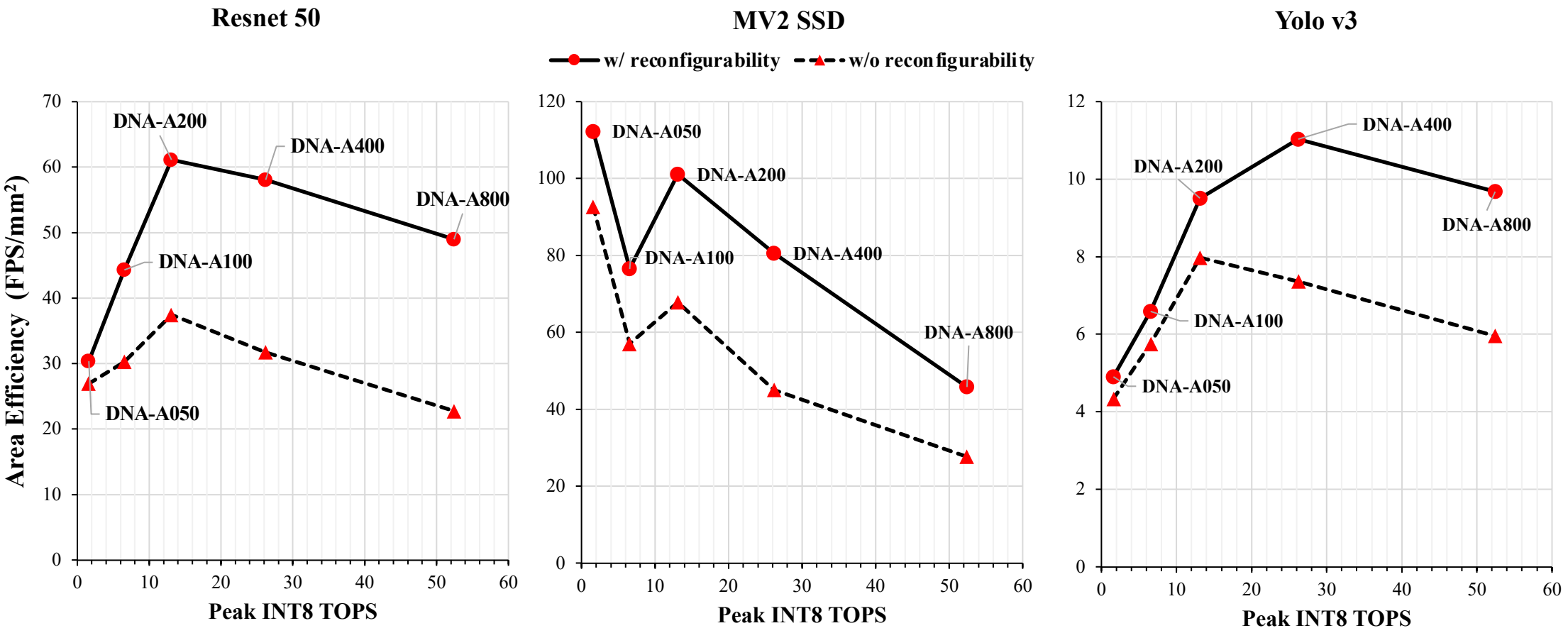
Lower is better



All results verified with Cadence Xcelium Logic Simulation. Post-training quantized, no changes to original neural network model.

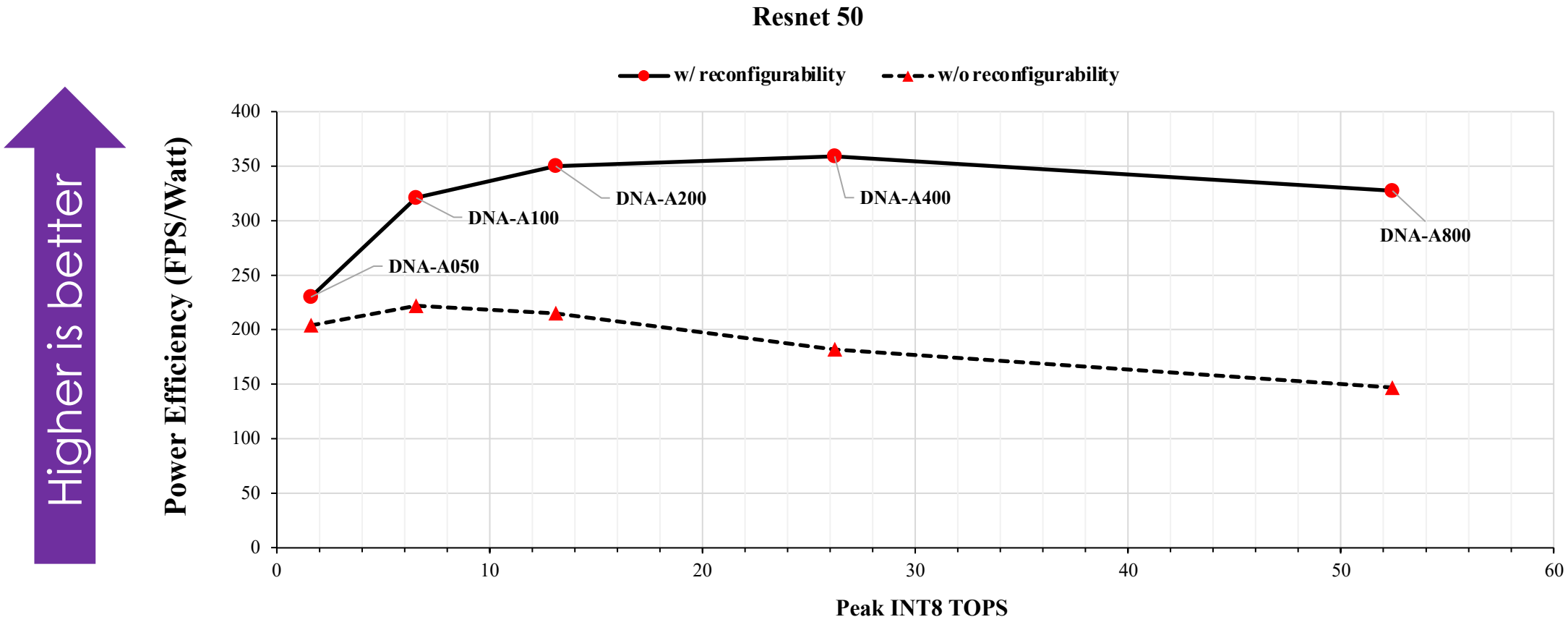
Improved Area Efficiency with Runtime Reconfiguration

Higher is better



FPS – Frames Per Second with batch size 1

Improved Power Efficiency with Runtime Reconfiguration

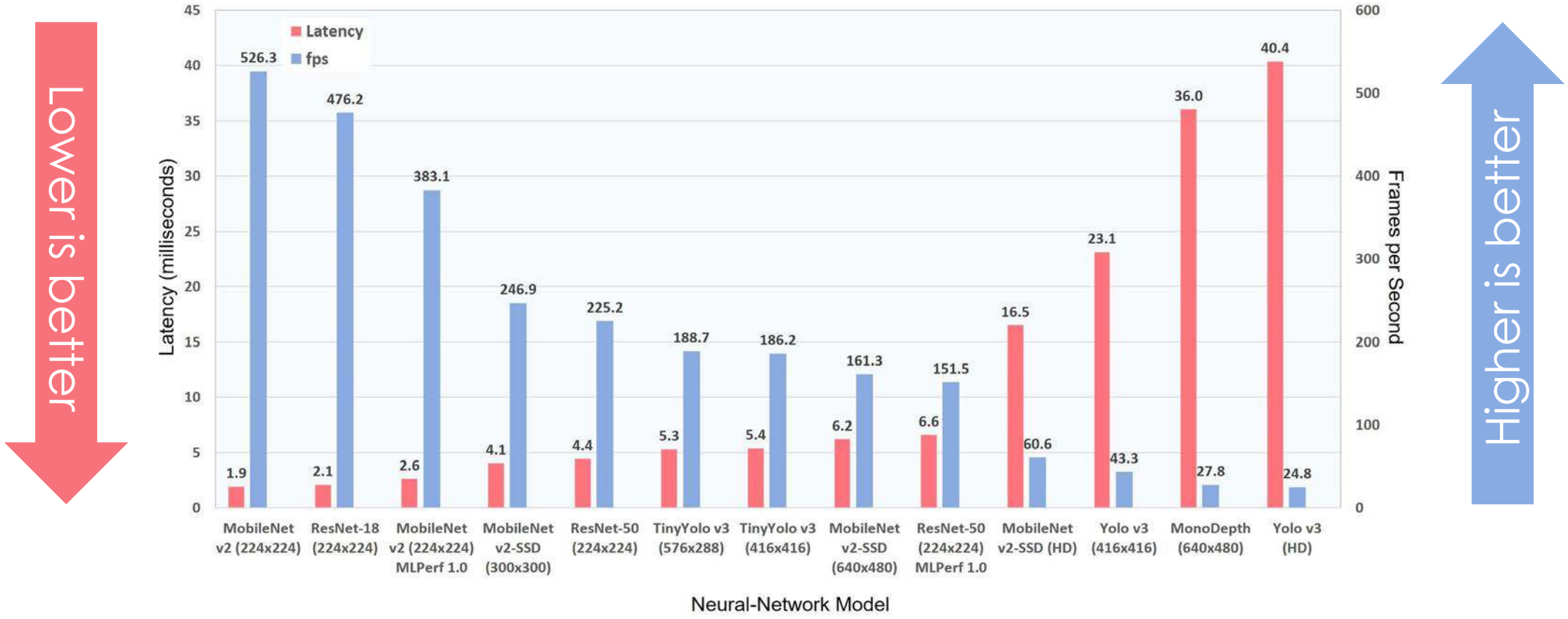


FPS – Frames Per Second with batch size 1

Benchmarking the Performance of DNA IP with FPGAs

Using the DNA-F200v2 configuration optimized for FPGAs delivering 4.9 INT8 TOPS at 300MHz

Latency-optimized inference for batch size 1 DNN (original model without pruning)



Summary

- Peak TOPS cannot be a proxy for performance without considering utilization
- **Compiler efficiency** and **run-time reconfigurability** are crucial to achieve high utilization for AI specific processors.
- Our software-first approach enables:
  - Realizing the compute potential of our IP through efficient graph optimization and scheduling
  - Minimizing user burden when switching from CPUs/GPUs to our IP
- Our run-time-reconfigurable IP enables:
  - Dynamically adapting to varying neural network/layer workloads
  - Maximizing scheduling freedom and minimizing unused compute resources
- **EdgeCortix DNA-A-series IP for ASIC and corresponding family of IP for FPGAs called DNA-F-series is available today,** along with the MERA software stack for AI inference.
- For more details on the Dynamic Neural Accelerator – dna-ip@edgecortix.com