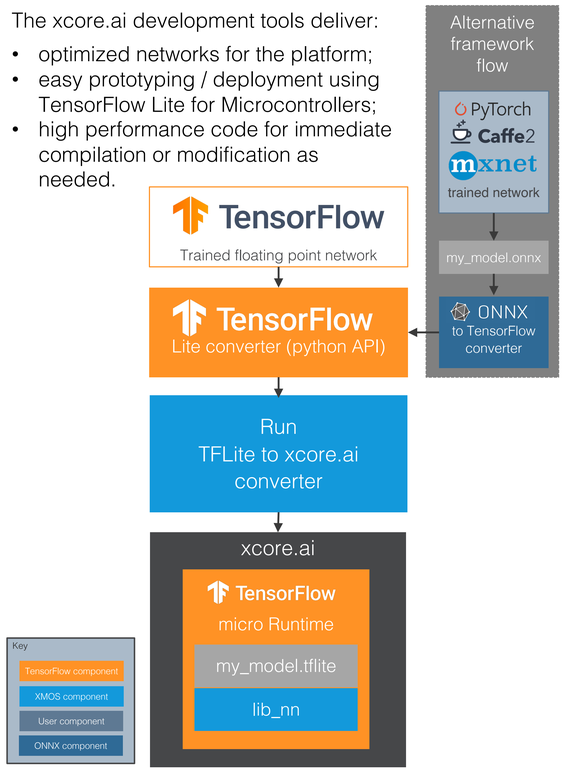
# Fast Neural Network Inference on xcore.ai

Laszlo Kindrat and Andrew Cavanaugh
Find out more at xcore.ai

## xcore.ai neural network porting tool

The xcore.ai development tools deliver:

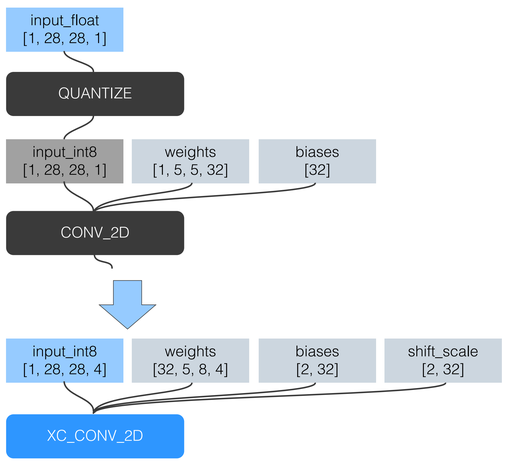
- optimized networks for the platform;
- easy prototyping / deployment using TensorFlow Lite for Microcontrollers;
- high performance code for immediate compilation or modification as needed.




## xcore.ai graph transformations

To take full advantage of the xcore.ai architecture, operations are slightly modified. This includes:

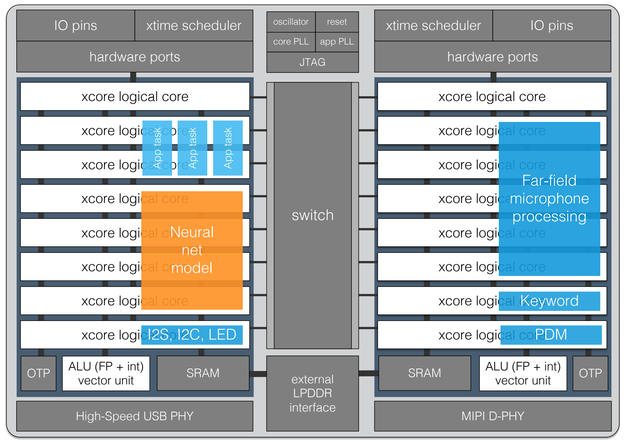
- reshaping and reordering of weight and bias tensors;
- adding xcore.ai specific parameter tensors;
- zero padding parameter or data tensors.




## xcore.ai architecture

xcore.ai implements XMOS' 3rd generation architecture, delivering state-of-the-art performance on edge inference tasks, high speed IO, and flexible, software-defined connectivity, in a single deterministic device.

- Each device includes 2 tiles. Each tile features:
  - Up to 1,600 MIPS, 25.6 GMACCS, 800 MFLOPS;
  - Up to 5 cores can executing simultaneously, each at 160 MHz effective clock rate (or 6-8 cores at lower clock speed down to 100 MHz);
  - 512 kB of memory with single cycle access.
- Each tile has 8 logical cores. Each core features:
  - Up to 320 MIPS, 5.12 GMACCS, 160 MFLOPS;
  - Execution of all (scalar/float/vector) instructions.
  - Vector unit with 256 bit wide registers
    - Operates in 8bit, 16bit or 32bit integer mode
- LPDDR-1 interface (200 MHz)
- USB-2.0 High-Speed PHY and MIPI D-PHY Rx




*Colored components are example software tasks

## xcore.ai benchmarks

To obtain the benchmarks the xcore.ai cycle accurate simulator was used to run inference on a network trained to classify CIFAR-10 images, identical in architecture to the model in [1]. Accuracies (on the CIFAR-10 test set) were 80.6% on our trained floating point model, and 79.7% on the quantized and converted xcore.ai models.

Execution time of quantized (int8) models [µs]

| Layer | Layer type | Filter shape | Output shape | Cortex-M7 *(Whole chip @ 600MHz)* | GAP8 (8 RISC-V cores @ 175MHz) | xcore.ai *(1 logical core @160 MHz)* |
|---|---|---|---|---|---|---|
| 1 | Convolution | 5x5x3x32 | 32x32x32 | 11,304 | N/A | 2,544 |
| 2 | Max pooling | N/A | 16x16x32 | 5,76 | N/A | 42 |
| 3 | Convolution | 5x5x32x32 | 16x16x32 | 15,408 | N/A | 1,856 |
| 4 | Max pooling | N/A | 8x8x32 | 144 | N/A | 10 |
| 5 | Convolution | 5x5x32x64 | 8x8x64 | 8,136 | N/A | 767 |
| 6 | Max pooling | N/A | 4x4x64 | 72 | N/A | 5 |
| 7 | Fully connected | 4x4x64x10 | 10 | 36 | N/A | 8 |
| Total execution time [µs] | | | | 35,676 | 8,700 | 5,236 |

- The project scope was constrained to a single logical core on xcore.ai. Measurements presented here should scale well on multicore benchmarks, for example reducing by a factor close to 5 when running inference on 5 cores of an xcore tile.
- For fairness, the Cortex-M7 values in [1] were adjusted to the highest clock speed available for that architecture (600MHz).
- Per-layer execution times for the GAP8 were not released in [2], and we were unable to obtain a dev kit to test it ourselves. Since it is unknown if GAP8 can be scaled to a different technology node, no clock rate adjustments were made for that.

| | Cortex-M7 *(Whole chip)* | GAP8 *(8 RISC-V cores)* | xcore.ai *(1 logical core)* |
|---|---|---|---|
| Energy cost [µJ/inference] | 5946 | 609 | 524 |
| Efficiency [inferences on a 3000mAh AA battery] | 2.7M | 26.6M | 30.9M |

These energy/efficiency numbers are for the CIFAR-10 inference task above. Further assumptions are:

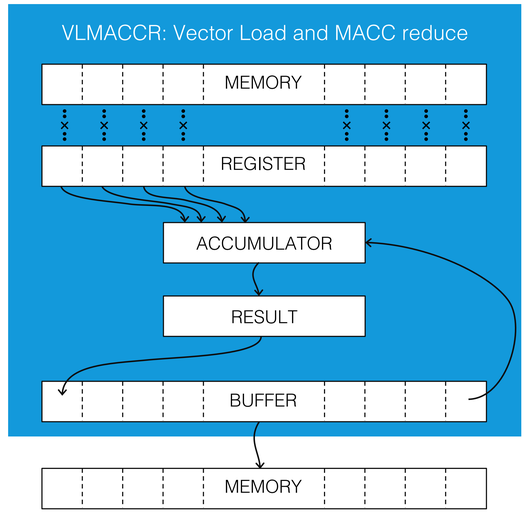
- xcore.ai values are based on cycle accurate simulations and physical extractions.
- GAP8 and Cortex-M7 values are based on active power measurements published in [2].
- Power values were not adjusted for differences in silicon manufacturing processes.

## xcore.ai vector unit

The vector unit implements a hardware ring buffer that allows highly efficient computation of very deep convolutions. In each instruction the vector unit:

- loads a vector of data;
- performs element-wise operations (similarly to SIMD);
- updates a set of result registers.

Instruction results are vectors that can be stored into memory or reduced to scalars.




Binarized neural networks (i.e. bitwise operations on 256-element vectors) are supported by special instructions, which also provide fast implementations of activation functions and pooling operations.

## references & acknowledgements

[1] V. Chandra, New CMSIS-NN Neural Network Kernels Boost Efficiency... (2018)
https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/new-neural-network-kernels-boost-efficiency-in-microcontrollers-by-5x

[2] *GAP8 performance versus ARM M7 on Embedded CNNs* (2019)
https://greenwaves-technologies.com/gap8-versus-arm-m7-embedded-cnns/

[3] D. Situnayake & P. Warden, *TinyML* (2019)
https://www.oreilly.com/library/view/tinyml/9781492052036/

[4] This project received funding from the European Union Horizon 2020 research and innovation programme under Grant Agreement No 849469:
https://ec.europa.eu/programmes/horizon2020/en

This information is not to be reproduced or published in any form without the prior permission of XMOS
marketing@xmos.com

www.PosterPresentations.com