Reconfigurable 2D Systolic Array based AI Accelerator with Architectural variants and Functional optimizations

Little Mac Team

Daniel Tran, Ryan Lee, Arian Torshizi, Mitali Agrawal, Anjana Manoj, John Hsu

Abstract

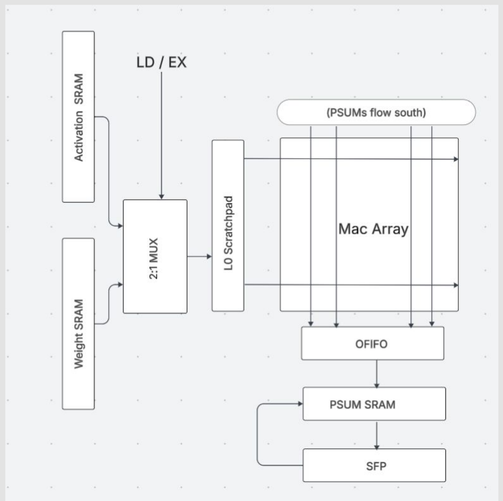
Neural network workloads demand hardware accelerators that are both **high-performance and flexible** to efficiently handle diverse computational requirements.

This project presents a **2D systolic array neural network accelerator** designed to explore multiple **architectural and functional alphas**, including tiling strategies, IFIFO-based partial sum reuse, and configurable activation functions.

By supporting different design variants within a single framework, the accelerator enables systematic evaluation of trade-offs between **throughput, resource usage, and energy efficiency**.

Implemented on FPGA, AlphaMAC demonstrates the potential of a **reconfigurable architecture** that balances performance and flexibility while allowing experimentation with architectural innovations.

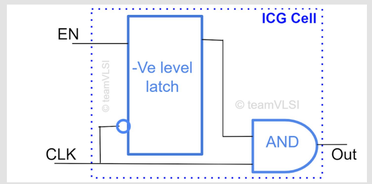
2D Systolic Array



Alpha 1: Clock gating to turn off blocks unused in each phase of operation.

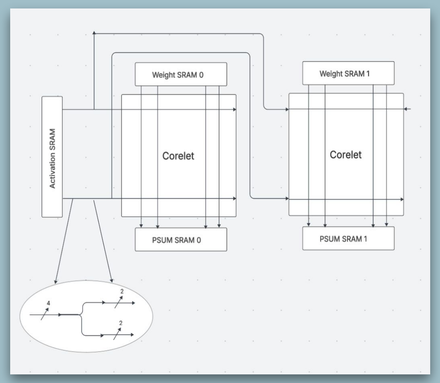
The significant dynamic power consumed by the SRAM blocks are gated based on standard chip enable signals.

Weight SRAM, Activation SRAM and PSUM SRAM are turned off and the sparsity aware mac_tile design avoids the heavy combinational logic of the multiplier, which also saves significant dynamic power on sparse data.



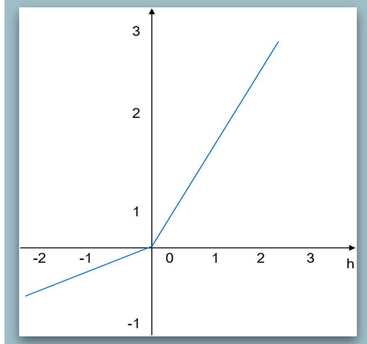
Dynamic power - 28 mW (almost 20% decrease from vanilla)

Alpha 2: Corelet to facilitate output channel tiling (16 output channels tiled into 8). Configurable between 2-bit and 4-bit activations



Dual-core instantiation, configurable between 2-bit and 4-bit activations. Can switch between the two on the layers. In the 2-bit activation, the dimensions are effectively 16x16. In the 4-bit activation mode, the dimensions are effectively 8x16. If hardware resources allow for it, this allows for 2x throughput *and* is flexible enough to accommodate two different types of layers without re-instantiation.

Alpha 3: Configurable activation function. Options are ReLU and LeakyReLU, where the negative scaling must be a value from [0, 0.5, 0.25, 0.125]. Scaling implemented by arithmetic right-shift



Leaky ReLU is a common alternate activation function. We mimic this with an arithmetic right-shift in the SFP unit, achieving the same effect with only 17,052 logic elements - a < 2% increase from vanilla.

Alpha 4: 16 x 16 parameterizable version + verification for 16 x 16 array. (No tiling, all operations done in one "round").

The original “vanilla” model was designed to be 8x8 which meant hardcoding some of the values in the mac array & row to work only with 8 input channels and 8 output channels

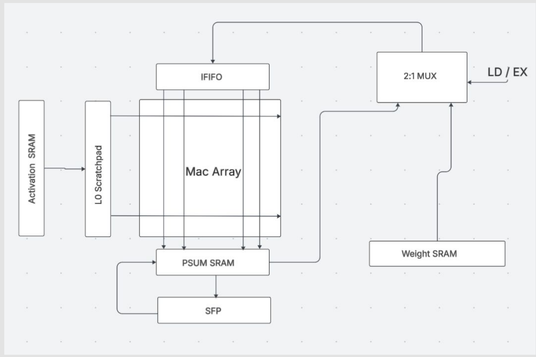
We generated activation and weight files for different input and output channel dimensions (16x16, 8x16, 16x8, 8x8)

We then configured a row and col parameter for our design and with minimal alterations to just the testbench, we were able to instantiate the design with different parameters and run the test bench

We observed that we were able to create different mac array dimensions that passed the testbench with generalizable and parameterized dimension

This means for our weight stationary design, we are able to handle any layer regardless of the layer input and output dimensions easily

Alpha 5: Row Tiling



Feeds data from PSUM SRAM back into MAC array (OFIFO not shown) for row tiles. On last tile, instead feeds to SFP for final ReLU. Accumulation is also done in-place via IFIFO feedback.

FPGA Mapping (Cyclone IV GX)

OPs	128
Frequency	127.36 MHz
Dynamic Power	33.36 mW
TOPs/s	0.0163
TOPs/W	0.489
Logic Elements	16,859

Alpha 6 In-Place Acc:

PSUMs are fed back in through the IFIFO for every kij (see Alpha 5’s diagram).

This allows us to achieve as much as 16x memory savings in depth, as we no longer need to hold PSUMs for every value of kij and can instead accumulate them in-place.