SemiEngineering Blog
April 6, 2019

Multi-Layer Processing Boosts Inference Throughput/Watt

The focus in discussion of inference throughput is often on the computations required.

For example, YOLOv3, a power real time object detection and recognition model, requires 227 BILLION MACs (multiply-accumulates) to process a single 2 Mega Pixel image! This is with the Winograd Transformation; it's more than 300 Billion without it.

And there is a lot of discussion of the large size of the weights required for the model: 62 Million in the case of YOLOv3.

What is often overlooked is the size of the intermediate activations.

YOLOv3 has slightly over 100 layers.

The input to the first layer is the 2 Megapixel image: 2M x 3 RGB bytes = 6MegaBytes (MB).

The output of the first layer is an activation of 64MB!

Looking at the activation sizes across all the layers we find the activation size output by layer:
1 layer outputs 64MB
2 layers output 32MB
4 layers output 16MB
15 layers output 8MB
24 layers output 4MB
20 layers output 2MB
8 layers output 1MB
2 layers output 0.5MB

So if the inference chip has 8MB of on-chip SRAM for activation storage, it needs to write activations to DRAM for 7 layers for a total of 192MB and then reading it immediately back.

The time to write to DRAM can be significant and is likely longer than the time for the execution of the layer, in which case the inference engine stalls.

And the energy to write to DRAM is approximately 100x the energy to write to local SRAM.

The size of these activations makes batching unattractive because of the large storage and power requirements.

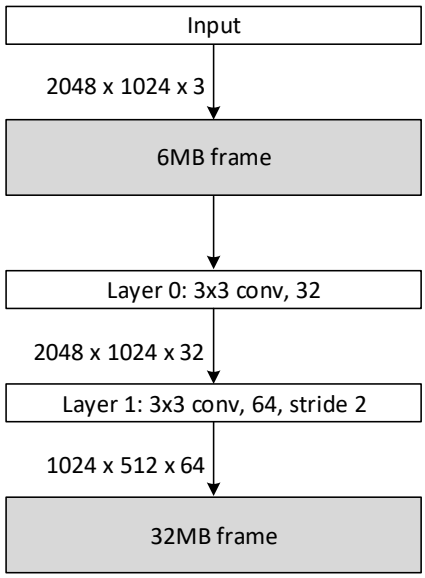
Multi-Layer Inference

The above analysis assumes each layer is processed one step at a time at batch = 1.

SemiEngineering Blog
April 6, 2019

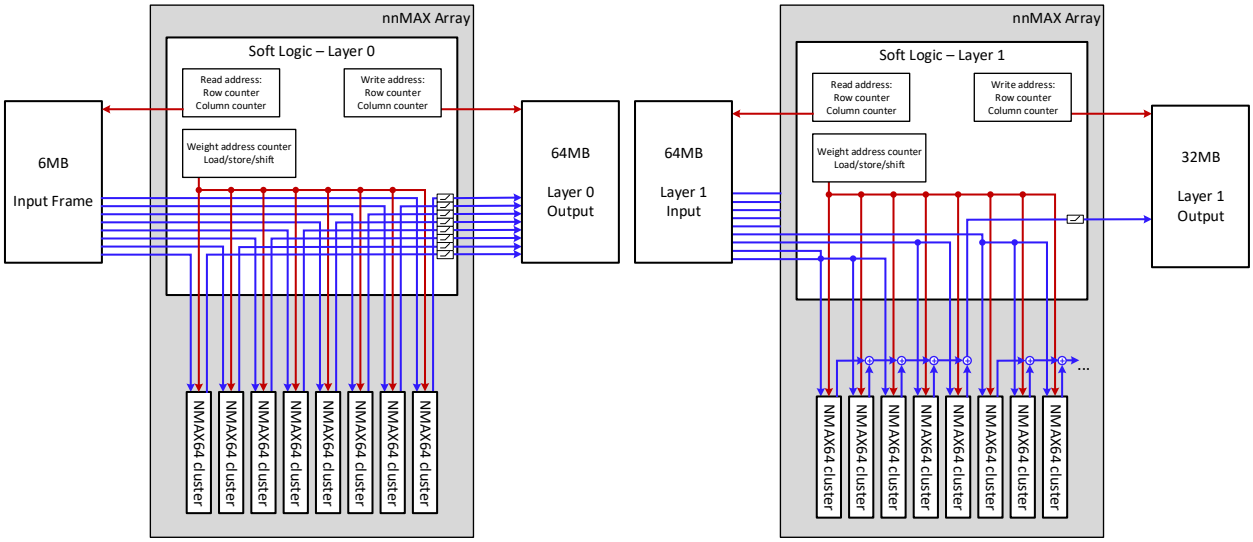
What if the inference architecture can process more than 1 layer in hardware simultaneously?

Let us consider the following fragment of a model:



This is the first two layers of YOLOv3.

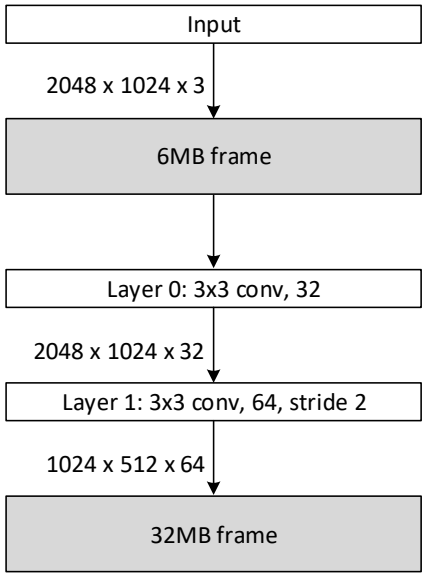
If done one layer at a time, with the nnMAX inference architecture, the result is below:



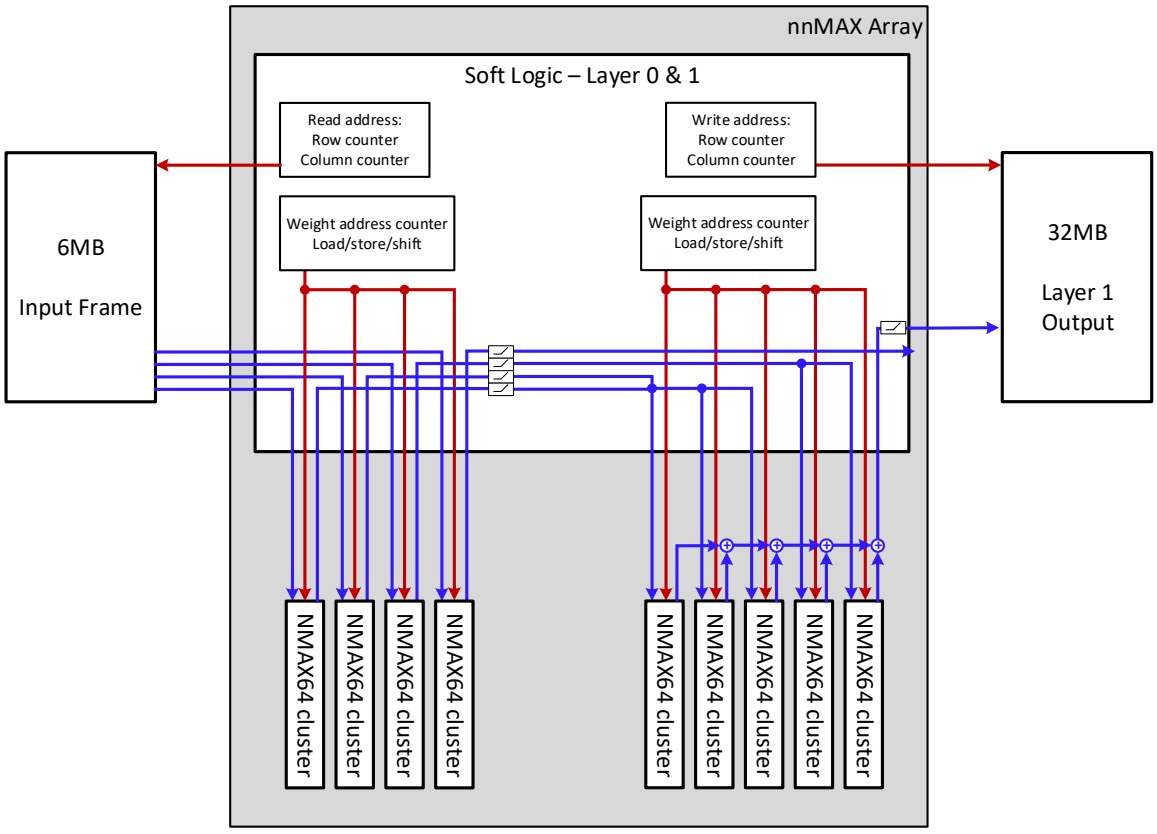
If the local SRAM for activation storage is say 8MB, large activations are written to and read back from DRAM, burning power and likely stalling the inference engine.

SemiEngineering Blog
April 6, 2019

What if you could process both Layer 0 and Layer 1 in hardware at the same time with Layer 0's output feeding directly into Layer 1? It would like like below and eliminates a 64MB write to DRAM and a 64MB read from DRAM.



In the nnMAX architecture, when there are sufficient MACs, say 4K MACs, it is possible to implement Layer 0 and 1 running simultaneously with one feeding directly into the other.



SemiEngineering Blog
April 6, 2019

The benefits are a reduction in DRAM bandwidth, which cuts power, and a simultaneous boost in throughput since the inference engine does not stall.

Depending on the number of MACs more layers could be implemented simultaneously.

For an nnMAX configuration of 4K MACs and 8MB SRAM, the performance increase from multi-layer operation is 15-20% over the performance in single-layer configuration.

Inference architectures which support multi-layer operation will have a significant throughput/watt advantage over those which do not.

Geoff Tate
CEO, Flex Logix