14.5 Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks

Yu-Hsin Chen[1], Tushar Krishna[1], Joel Emer[1,2], Vivienne Sze[1]

[1]Massachusetts Institute of Technology, Cambridge, MA,
[2]Nvidia, Westford, MA

Deep learning using convolutional neural networks (CNN) gives state-of-the-art accuracy on many computer vision tasks (e.g. object detection, recognition, segmentation). Convolutions account for over 90% of the processing in CNNs for both inference/testing and training, and fully convolutional networks are increasingly being used. To achieve state-of-the-art accuracy requires CNNs with not only a larger number of layers, but also millions of filters weights, and varying shapes (i.e. filter sizes, number of filters, number of channels) as shown in Fig. 14.5.1. For instance, AlexNet [1] uses 2.3 million weights (4.6MB of storage) and requires 666 million MACs per 227×227 image (13kMACs/pixel). VGG16 [2] uses 14.7 million weights (29.4MB of storage) and requires 15.3 billion MACs per 224×224 image (306kMACs/pixel). The large number of filter weights and channels results in substantial data movement, which consumes significant energy.

Existing accelerators do not support the configurability necessary to efficiently support large CNNs with different shapes [3], and using mobile GPUs can be expensive [4]. This paper describes an accelerator that can deliver state-of-the-art accuracy with minimum energy consumption in the system (including DRAM) in real-time, by using two key methods: (1) efficient dataflow and supporting hardware (spatial array, memory hierarchy and on-chip network) that minimize data movement by exploiting data reuse and support different shapes; (2) exploit data statistics to minimize energy through zeros skipping/gating to avoid unnecessary reads and computations; and data compression to reduce off-chip memory bandwidth, which is the most expensive data movement.

Figure 14.5.2 shows the top-level architecture and memory hierarchy of the accelerator. Data movement is optimized by buffering input image data (Img), filter weights (Filt) and partial sums (Psum) in a shared 108KB SRAM buffer, which facilitates temporal reuse of loaded data. Image data and filter weights are read from DRAM to the buffer and streamed into the spatial computation array allowing for overlap of memory traffic and computation. The streaming and reuse allows the system to achieve high computational efficiency even when running the memory link at a lower clock frequency than the spatial array. The spatial array computes inner products between the image and filter weights, generating partial sums that are returned from the array to the buffer and then, optionally rectified (ReLU) and compressed, to the DRAM. Run-length-based compression reduces the average image bandwidth by 2×. Configurable support for image and filter sizes that do not fit completely into the spatial array is achieved by saving partial sums in the buffer and later restoring them to the spatial array. The sizes of the spatial array and buffer determine the number of such 'passes' needed to do the calculations for a specific layer. Unused PEs are clock gated.

Figure 14.5.3 shows the dataflow within the array for filter weights, image values and partial sums. If the filter height (R) equals the number of rows in the array (in our case 12), the logical dataflow would be as follows: (1) filter weights are fed from the buffer into the left column of the array (one filter row per PE) and the filter weights move from left to right within the array; (2) image values are fed into the left column and bottom row of the array (one image row per PE) and the image values move up diagonally; (3) partial sums for each output row move up vertically, and can be read out of the top row at the end of the computational pass. If the partial sums are used in the next pass, they are fed into the bottom row of the array from the buffer at the beginning of the next computational pass.

In order to maximize utilization of a fixed-size array for different shapes, the mapping may require either folding or replication if the shape size is larger or smaller than the array dimension, respectively. Replication results in increased throughput as compared to the purely logical dataflow described above. Cases II, III, IV, and V in Fig. 14.5.3 illustrate the replication and folding of image values for various layers of AlexNet. The same data values are shown in the same color. Across the six example cases, which include physical mapping of filter weights, image values and partial sums onto the fixed-size spatial array, we see the logical dataflow patterns translating to myriad physical dataflow patterns that need to be supported. Furthermore, the same data value is often needed by multiple PEs, whose physical location in the array depends on the data type (filter, image or partial sum) and layer.

Since different layers have different shapes and hence different mappings, a design-time fixed interconnect topology will not work. Every PE can potentially be a destination for a piece of data in some particular configuration, and so a Network-on-Chip (NoC) is needed to support address based data delivery. However, traditional NoC designs with switches at every PE to buffer/forward data to one or multiple targets would result in multi-cycle delays. A full-chip broadcast to every PE could work, but would consume enormous power.

To optimize data movement, it is important to exploit spatial reuse, where a single buffer read can be used by multiple PEs (i.e. multicast). Fig. 14.5.4 shows our NoC that supports configurable data patterns, and provides an energy-efficient multicast to a variable number of PEs within a single-cycle. The NoC comprises one Global Y bus, and 12 Global X buses (one per row). Each PE is *configured* with a (row, col) ID at the beginning of processing via a scan chain. Multicast to any subset of PEs is achieved by assigning the *same* ID to *multiple* PEs. Data from the buffer is tagged with the target PEs' (row, col) ID, and multicast controllers at the input of each X bus and each PE deliver data only to those X buses and PEs, respectively, that match the target ID to avoid unnecessary switching. Data is sent on the buses only if *all* target PEs are ready (i.e., have an empty buffer) to receive. To support high bandwidth, we use separate input NoCs for filter, image, and partial sums. The partial sum NoC has a separate set of output links to the buffer to write the final partial sums. The NoC data delivery for four of the cases from Fig. 14.5.3 is shown in Fig. 14.5.4.

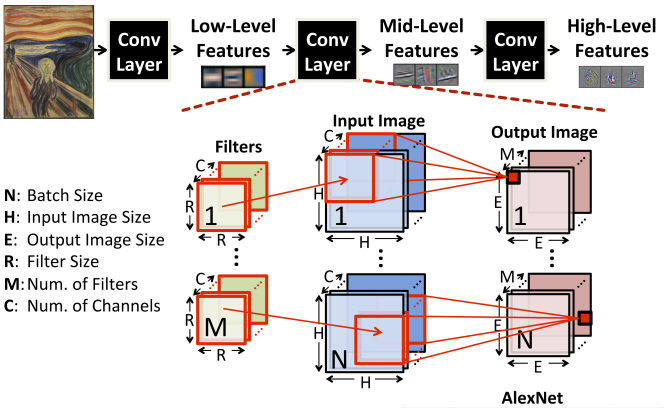
Each processing engine, shown in Fig. 14.5.5, is a three-stage pipeline responsible for calculating the inner product of the input image and filter weights for a single row of the filter. The sequence of partial sums for the sliding filter window is computed sequentially. The partial sums for the row are passed on a local link to the neighboring PE (see Fig. 14.5.4), where the cross-row partial sums are computed. Local scratch pads allow for energy-efficient temporal reuse of input image and filter weights by recirculating values needed by different windows. A partial sum scratch pad allows for temporal reuse of partial sums being generated for different images and/or channels and filters. Data gating is achieved by recording the input image values of zero in a 'zero buffer' and skipping filter reads and computation for those values resulting in a 45% power savings in the PE.

The test chip is implemented in 65nm CMOS. It operates at 200MHz core clock and 60MHz link clock, which results in a frame rate of 34.7fps on the five convolutional layers in AlexNet and a measured power of 278mW at 1V. The PE array, NoC and on-chip buffer consume 77.8%, 15.6% and 2.7% of the total power, respectively. The core and link clocks can scale up to 250MHz and 90MHz, respectively. This enables us to achieve a throughput of 44.8fps at 1.17V. Fig. 14.5.6 shows the performance at each layer, including compression ratio, power consumption, PE utilization, and memory access to highlight the reduction in DRAM bandwidth, efficiency of the reconfigurable mapping and reduced data access due to data reuse, respectively. A die photo of the chip and the range of the shapes it can support natively are shown in Fig. 14.5.7.

Acknowledgements:
This work is funded by the DARPA YFA grant N66001-14-1-4039, MIT Center for Integrated Circuits & Systems, and a gift from Intel. The authors would also like to thank Mehul Tikekar and Michael Price for their technical assistance.

References:
[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems,* pp. 1097–1105, 2012.
[2] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," CoRR, abs/1409.1556, 2014.
[3] S. Park et al., "A 1.93TOPS/W Scalable Deep Learning/Inference Processor with Tetra-parallel MIMD Architecture for Big Data Applications," *ISSCC Dig. Tech. Papers*, pp. 80-81, 2015
[4] S. Chetlur et al., "cuDNN: Efficient Primitives for Deep Learning," CoRR, abs/1410.0759, 2014.

 978-1-4673-9467-3/16/$31.00



CNN	LeNet	AlexNet	VGG16
Year	1989	2012	2014
# Conv Layer	2	5	13
# Weights in Conv Layer	50k	2.3M	14.7M
Ratio (Memory)	1×	46×	294×
# Connection in Conv Layer	322k	666M	15.3G
Ratio (Computation)	1×	2067×	47660×

AlexNet

Layer	R	M	C	Stride
1	11x11	96	3	4
2	5x5	256	48	1
3	3x3	384	256	1
4	3x3	384	192	1
5	3x3	256	192	1

Figure 14.5.1: Deep CNNs are large with varying shapes.

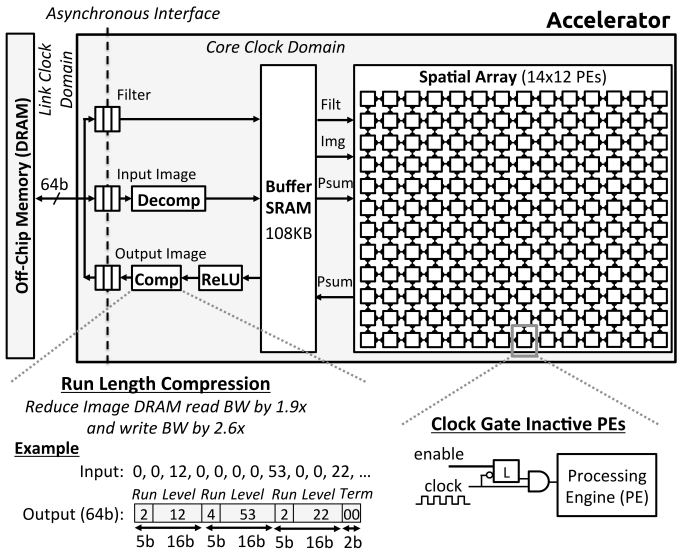


Figure 14.5.2: Top-level architecture with 168 PEs.

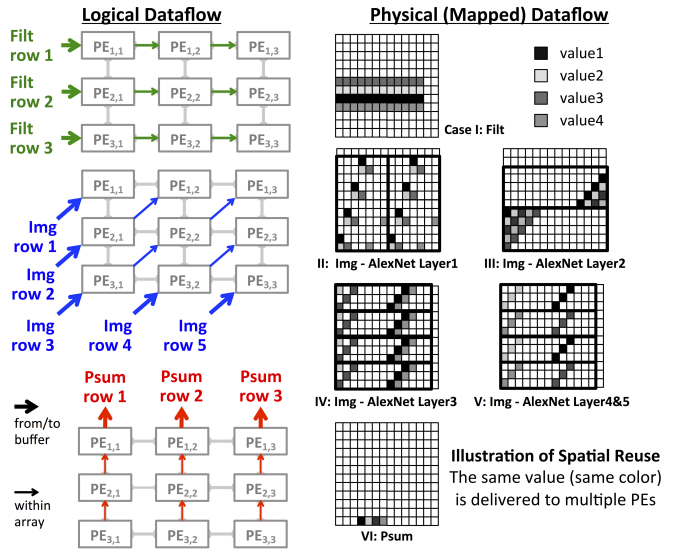


Figure 14.5.3: Logical and physical dataflows.

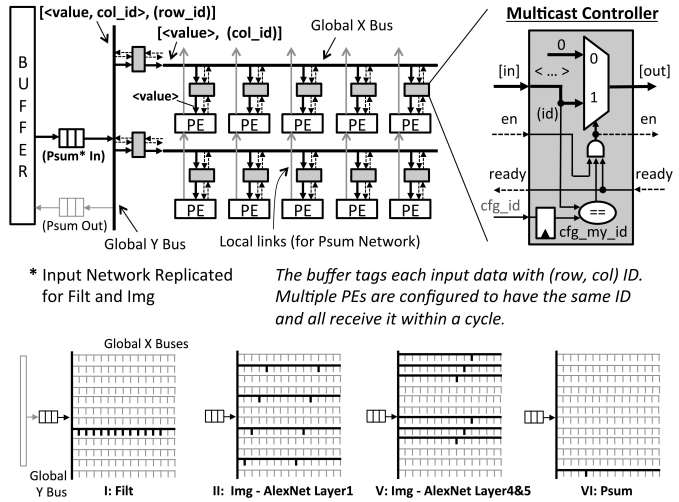


Figure 14.5.4: Network-on-Chip (NoC) for multicasting.

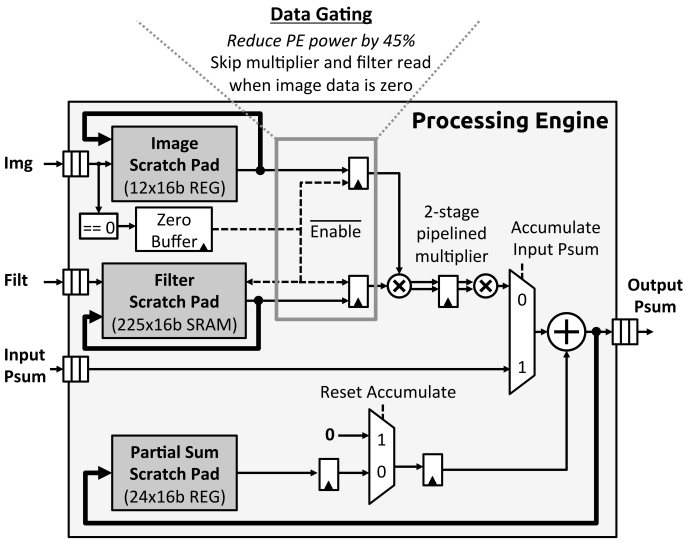
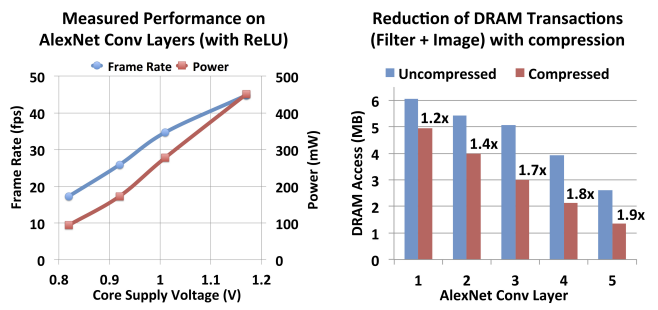


Figure 14.5.5: 3-stage pipelined processing engine.



AlexNet Conv Layer Performance Breakdown (with ReLU)

Image Batch Size = 4 (4 frames of 227x227), Core Clock: 200MHz / Link Clock = 60MHz

Layer	Power (mW)	Latency (ms)	# of MAC (MOPs)	Active # of PEs (%)	# of Filter Weights	Buffer Data Access (MB)	DRAM Data Access (MB)
1	332	20.9	421.6	154 (92%)	34.8k	18.5	5.0
2	288	41.9	895.7	135 (80%)	307.2k	77.6	4.0
3	266	23.6	598.1	156 (93%)	884.7k	50.2	3.0
4	235	18.4	448.6	156 (93%)	663.6k	37.4	2.1
5	236	10.5	299.0	156 (93%)	442.4k	24.9	1.3
Total	**278**	**115.3**	**2663**	**148 (88%)**	**2332.7k**	**208.5**	**15.4**

Figure 14.5.6: Performance of AlexNet convolutional layers.

Technology	TSMC 65nm LP 1P9M
Core Area	3.5mm×3.5mm
Gate Count	1852 kGates (NAND2)
Total SRAM Size	181.5 KB
On-Chip Buffer	108 KB
# of PEs	168
Scratch Pad / PE	0.5 KB
Supply Voltage	0.82 – 1.17 V
Core Frequency	100 – 250 MHz
Peak Performance	16.8 – 42.0 GOPS (1 OP = 1 MAC)
Word Bit-width	16-bit Fixed-Point
Filter Size*	1 – 32 [width] 1 – 12 [height]
# of Filters*	1 – 1024
# of Channels*	1 – 1024
Stride Range	1–12 [horizontal] 1, 2, 4 [vertical]

*** Natively Supported**

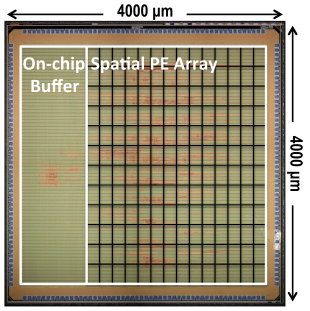


Figure 14.5.7: Chip specifications and die photo.

978-1-4673-9467-3/16/$31.00