Toward Energy-Efficient Intelligence in Power-/Area-Constrained Hardware: End-to-End Hardware Accelerator and Coarse-Grain Memory Compression for Deep Learning Algorithms

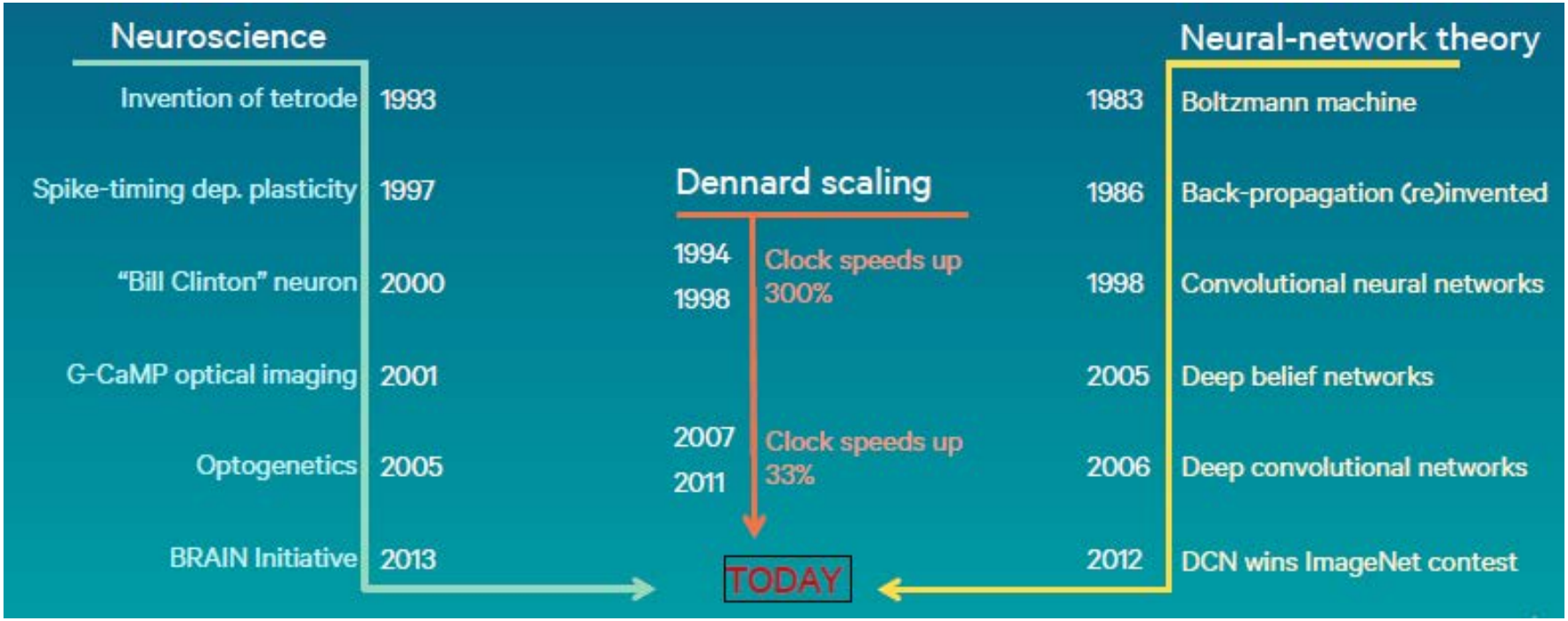
Jae-sun Seo

School of ECEE
Arizona State University

@ ASPDAC Tutorial

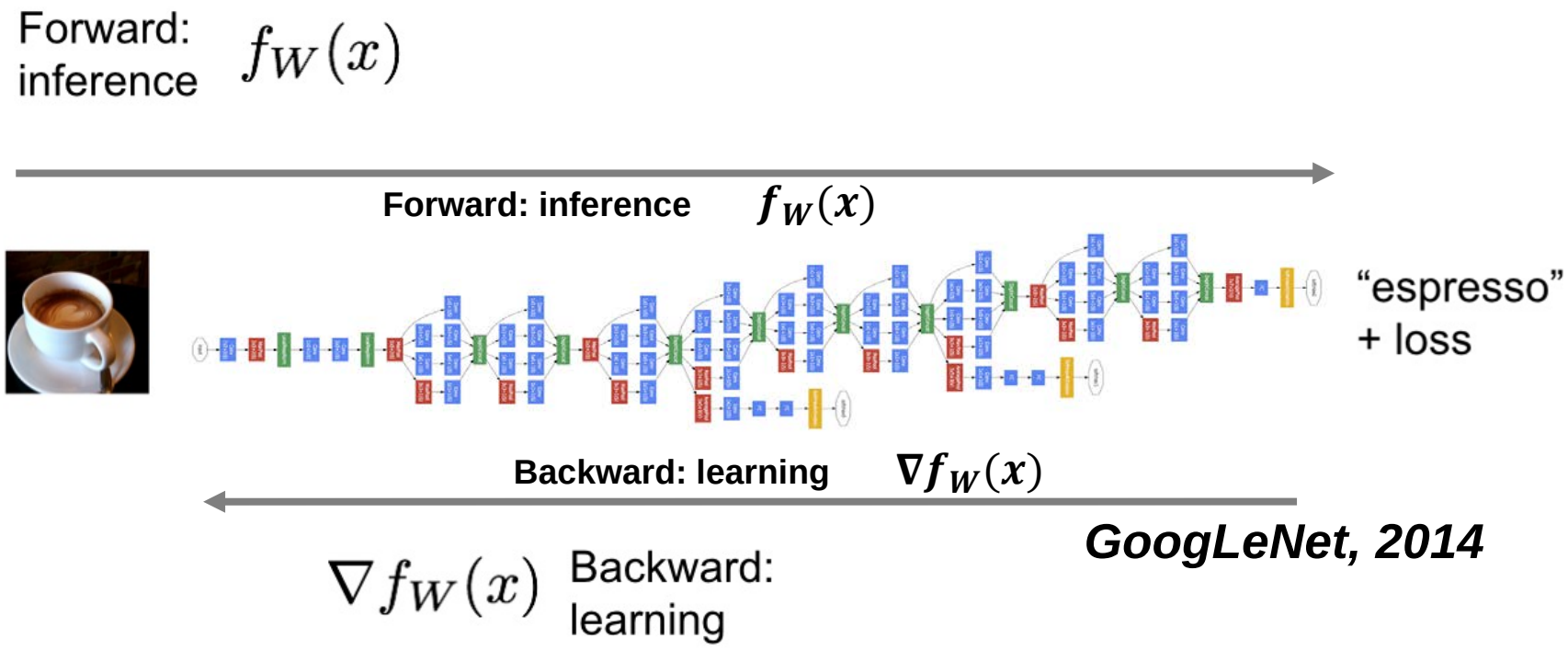
January 16, 2017

Machine Learning Hardware – Why Now?



J. Gehlhaar, ASPLOS 2014 Keynote

DNN Training & Classification



- **Training (back-propagation): typically done off-line w/ labeled data**
  - **Large networks take multiple days – weeks using powerful GPUs**

- **Classification (feed-forward): target real-time operation**
  - **For portable applications, power/area constraints pose limits**

Outline

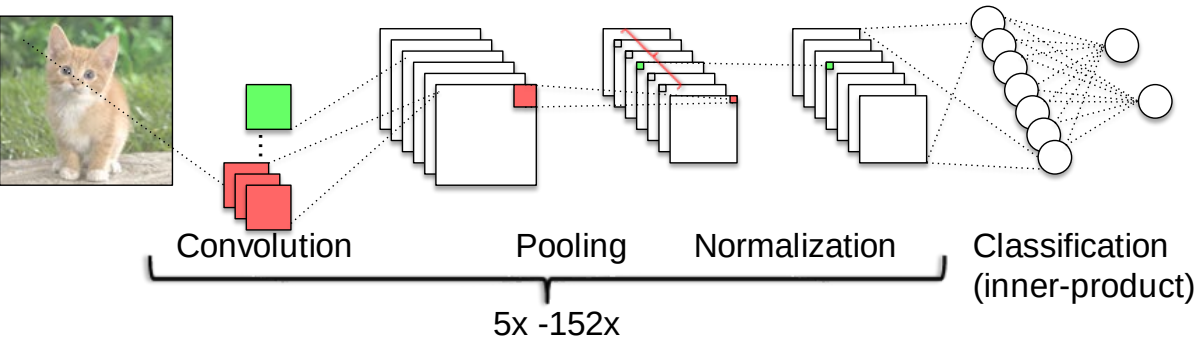
- Introduction
- Image: FPGA Design on Convolutional Neural Networks
  - FPGA 2016, FPL 2016, FPGA 2017
- Speech: Deep Neural Networks w/ Coarse-Grain Sparsity
  - ICCAD 2016
- Summary

Outline

- Introduction
- **Image: FPGA Design on Convolutional Neural Networks**
  - **FPGA 2016, FPL 2016, FPGA 2017**
- Speech: Deep Neural Networks w/ Coarse-Grain Sparsity
  - ICCAD 2016
- Biomedical: Compressed Neural Networks for Wearables
  - 65nm prototype chip for ECG Biometric Authentication
- Summary

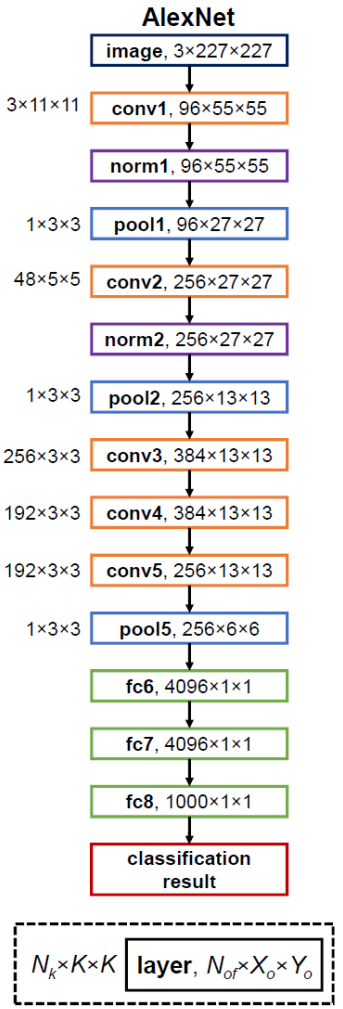
ImageNet: Image Classification & Localization

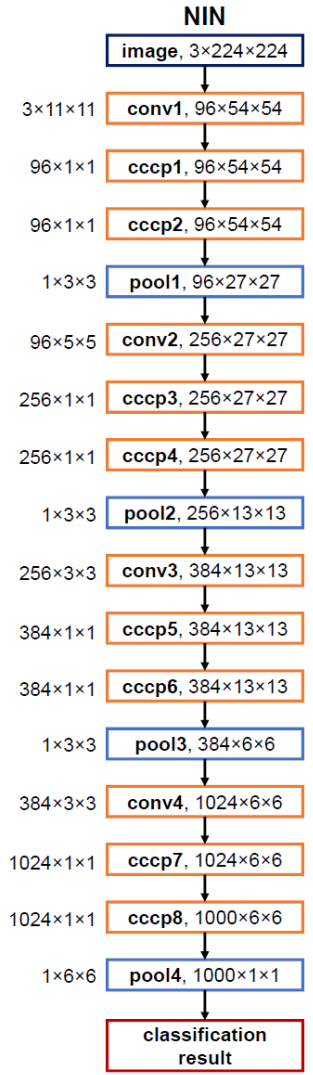
- **ImageNet contest: 1,000 image categories**
  - **Top-1 & top-5 classification accuracy**
  - **Localization: exact area of the object**



many layers (deep)

- **Alexnet: 5 conv, 3 pool, 2 norm, 3 full-conn**
- **NIN: 4 conv, 8 cccp, 4 pool**
- **VGG: 16 conv, 4 pool, 3 full-conn**





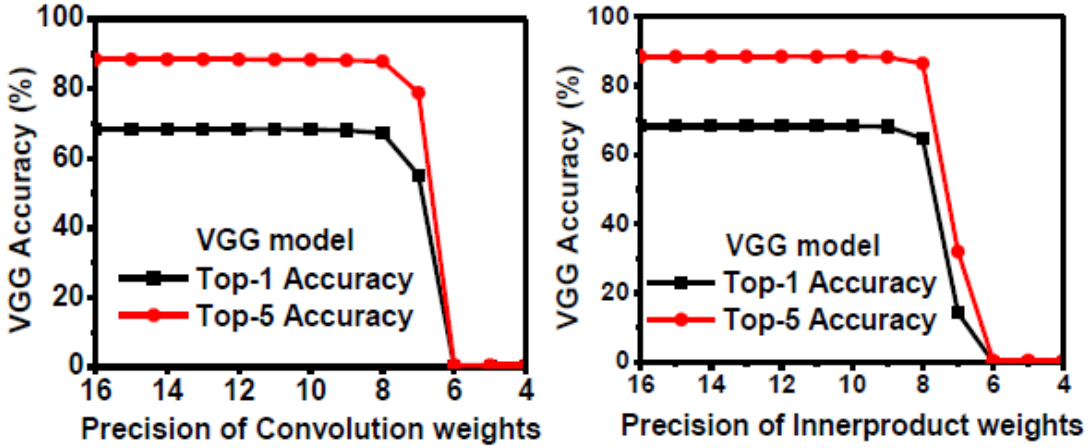
Our Recent Research on Deep CNN

- FPGA 2016 (Int. Symp. on Field Programmable Gate Arrays)
  - N. Suda, et al.
  - "Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks"
- FPL 2016 (Int. Conf. on Field Programmable Logic and Applications)
  - Y. Ma, et al.
  - "Scalable and Modularized RTL Compilation of Convolutional Neural Networks onto FPGA"
- FPGA 2017, in press
  - Y. Ma, et al.
  - "Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks"

Precision Optimization

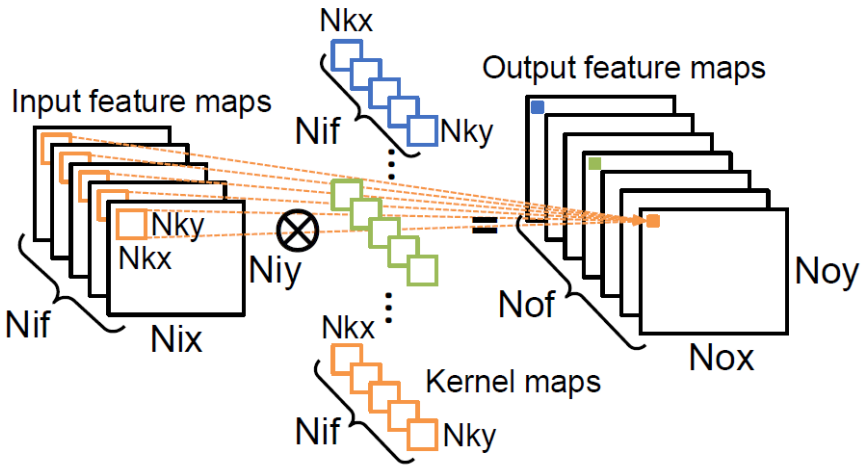
- **Accuracy vs. precision (*w/ ImageNet ILSVRC2012_val 1~200 images*)**

	Floating-point	Integer/fraction fixed-point									
Data bits	-	13/2	13/2	13/2	13/2	13/2	13/2	13/2	13/2	13/2	13/2
Weight bits	-	0/15	0/14	0/13	0/12	0/11	0/10	0/9	0/8	0/7	0/6
VGG-19 Top1 error	32.0%	31.0%	31.5%	31.0%	31.0%	31.0%	32.5%	32.5%	33%	37.0%	94.0%
VGG-19 Top5 error	12.0%	12.0%	12.0%	12.5%	12.0%	12.0%	12.5%	12.5%	12.5%	17.0%	88.5%
VGG-16 Top1 error	35.5%	35.5%	34.5%	35.5%	35.0%	35.5%	35.5%	35.5%	36.5%	39.5%	95.0%
VGG-16 Top5 error	13.0%	13.0%	13.0%	13.0%	13.0%	13.0%	13.0%	13.0%	13.0%	15.0%	81%



- **Better to use diff. integer vs. fractional bits for neurons vs. weights**

Convolution Loops in Deep CNNs

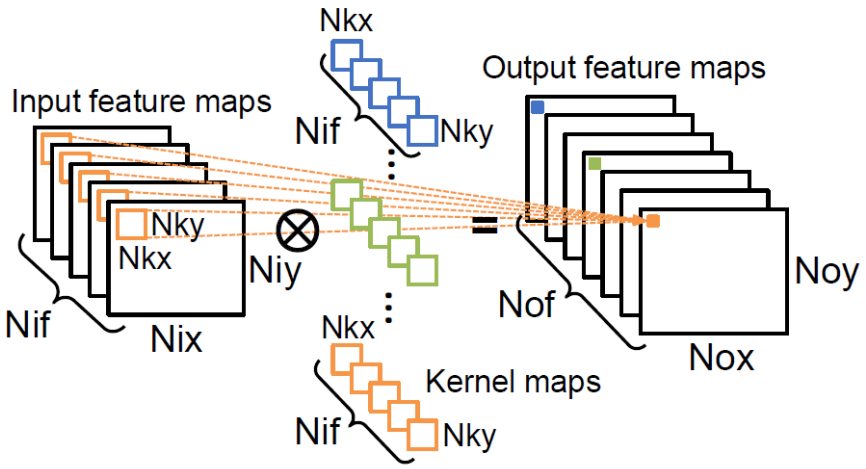


Across the output feature maps of N_{of} **Loop-4**
Scan within one input feature map with $X \times Y$ **Loop-3**
Across the input feature maps of N_{if} **Loop-2**
MAC within a kernel window of $K \times K$ **Loop-1**

```
for (no = 0; no < Nof; no ++)                                   ---> Loop-4
    for (y = 0; y < Noy; y =+ S)
        for (x = 0; x < Nox; x =+ S)                            ---> Loop-3
            for (ni = 0; ni < Nif; ni ++)                       ---> Loop-2
                for (ky = 0; ky < Nky; ky ++)
                    for (kx = 0; kx < Nkx; kx ++)               ---> Loop-1
                        pixel_L(no; x, y) += pixel_L-1(ni; x + kx, y + ky) × weight_L-1(ni, no; kx, ky);
            pixel_L (no; x, y) = pixel_L(no; x, y) + bias(no);
```

- **Four level of loops exist per image computation, unrolling and order of loops are crucial for memory, comm.**

Convolution Loop Optimization

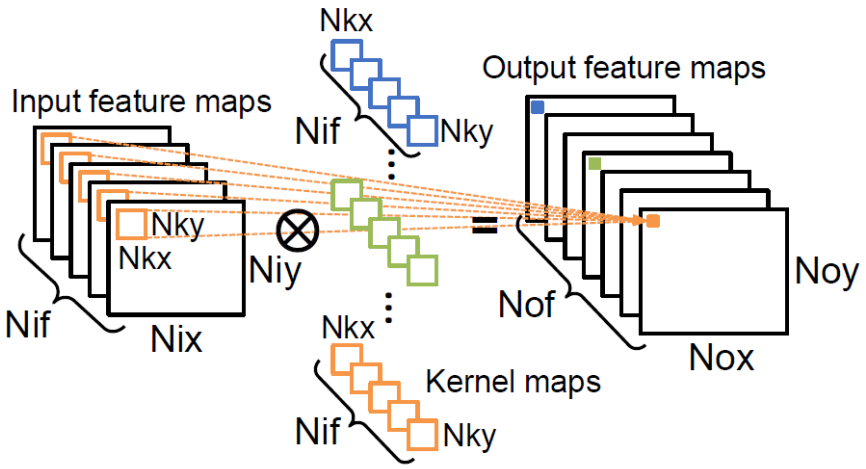


- **Loop unrolling**
  - **Loop 1: parallelize k*k conv.**
- **Loop tiling**
  - **On-chip memory vs. DRAM**
- **Loop interchange**
  - **Sequential order of loops**

	Kernel Window (width/height)		Input Feature Map (width/height)		Output Feature Map (width/height)		# of Input Feature Maps	# of Output Feature Maps
Convolution Loops	**Loop-1**		**Loop-3**		**Loop-3**		**Loop-2**	**Loop-4**
Convolution Dimensions	Nkx	Nky	Nix	Niy	Nox	Noy	Nif	Nof
Loop Tiling	*Tkx*	*Tky*	*Tix*	*Tiy*	*Tox*	*Toy*	*Tif*	*Tof*
Loop Unrolling	*Pkx*	*Pky*	*Pix*	*Piy*	*Pox*	*Poy*	*Pif*	*Pof*

- **Design objectives:**
  - **minimize computing latency, maximize data re-use, (min. on-chip mem. access)**
  - **minimize partial sum storage, minimize DRAM access**

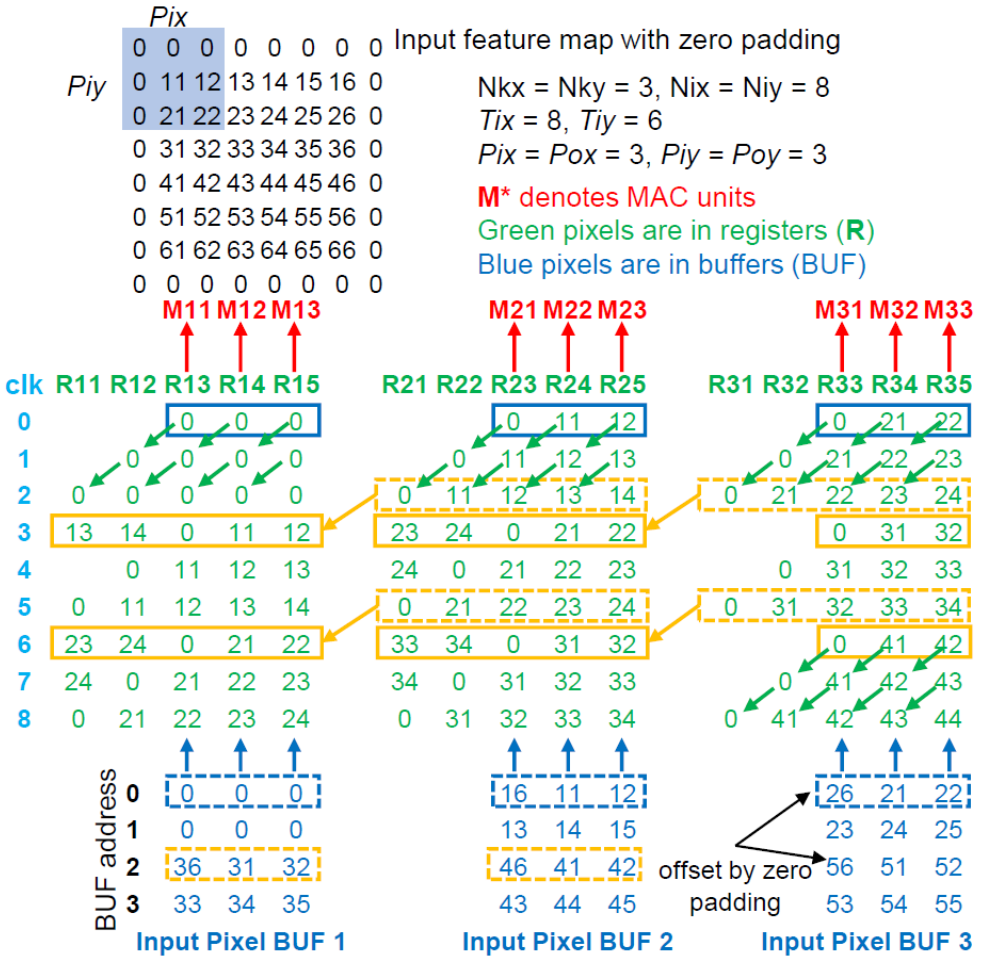
Convolution Loop Optimization



Across the output feature maps of N_{of} — **Loop-4**
Scan within one input feature map with $X \times Y$ — **Loop-3**
Across the input feature maps of N_{if} — **Loop-2**
MAC within a kernel window of $K \times K$ — **Loop-1**

- **Proposed strategy**
  - **Unroll Loop 3 & 4 to max. pixel / weight reuse, reduce on-chip buffer acc.**
  - **First compute intra-tiling Loop 1 & 2, serially, to minimize # of partial sums**
    - **Store the Loop 1 & 2 data in on-chip buffer**
  - **Compute Loop 4 before 3 in inter-tiling order, to re-use pixels in buffer**
- **Related work: Eyeriss (ISSCC 2016, ISCA 2016)**
  - **Unrolled Loop 1 & 3**
    - **Further pixel / weight reuse by unrolling Loop 4 cannot be achieved**
    - **Loop 1 unrolling: insufficient parallelism, kernel size difference**

Dataflow in Convolution Layers

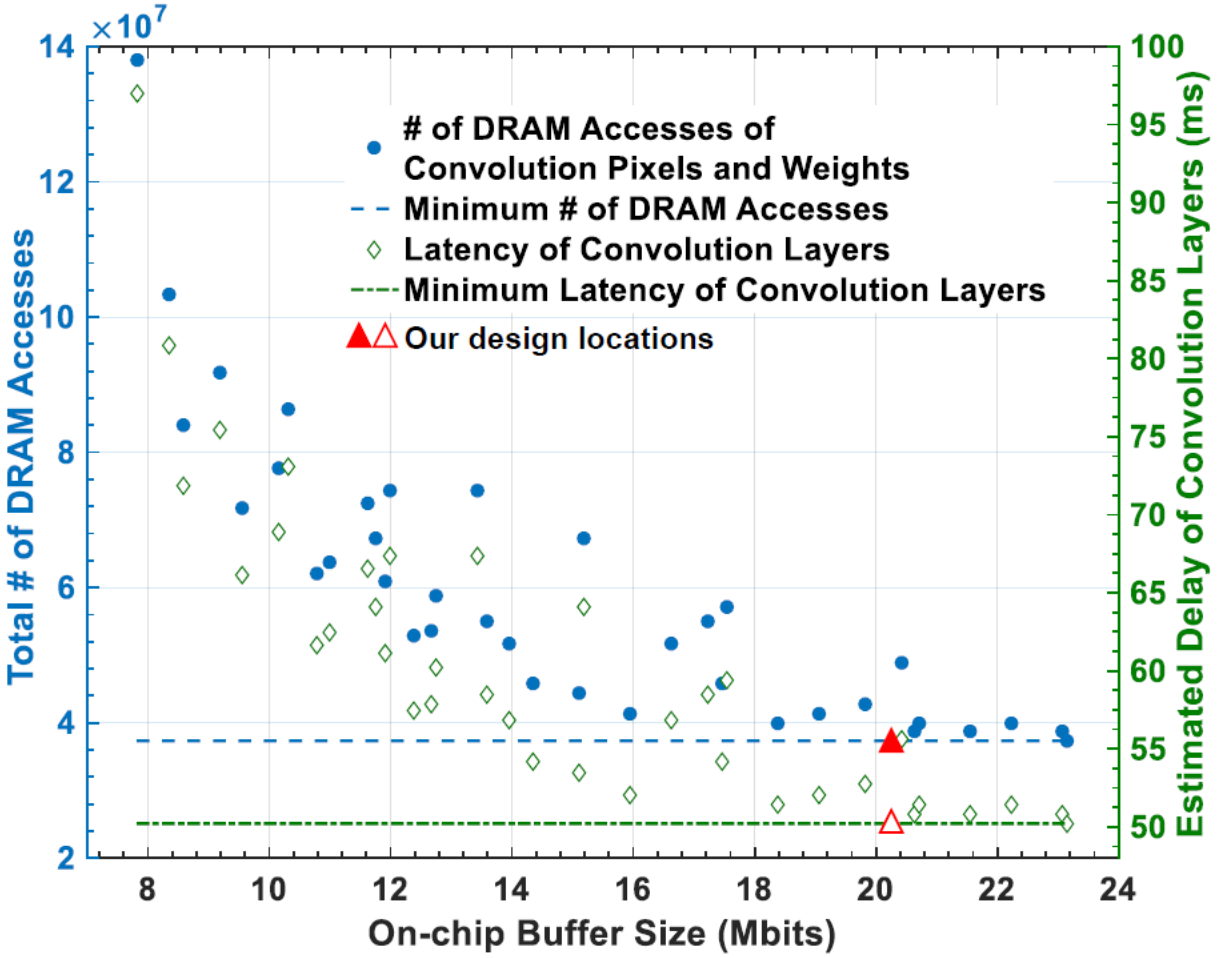


Number (e.g. 14) denote (y,x) location of pixels (e.g. y=1, x=4)

Cycle 1/2/4/5/7/8: pixel shift
→ sliding overlapped pixels reused
Cycle 3/6: new pixel transfer

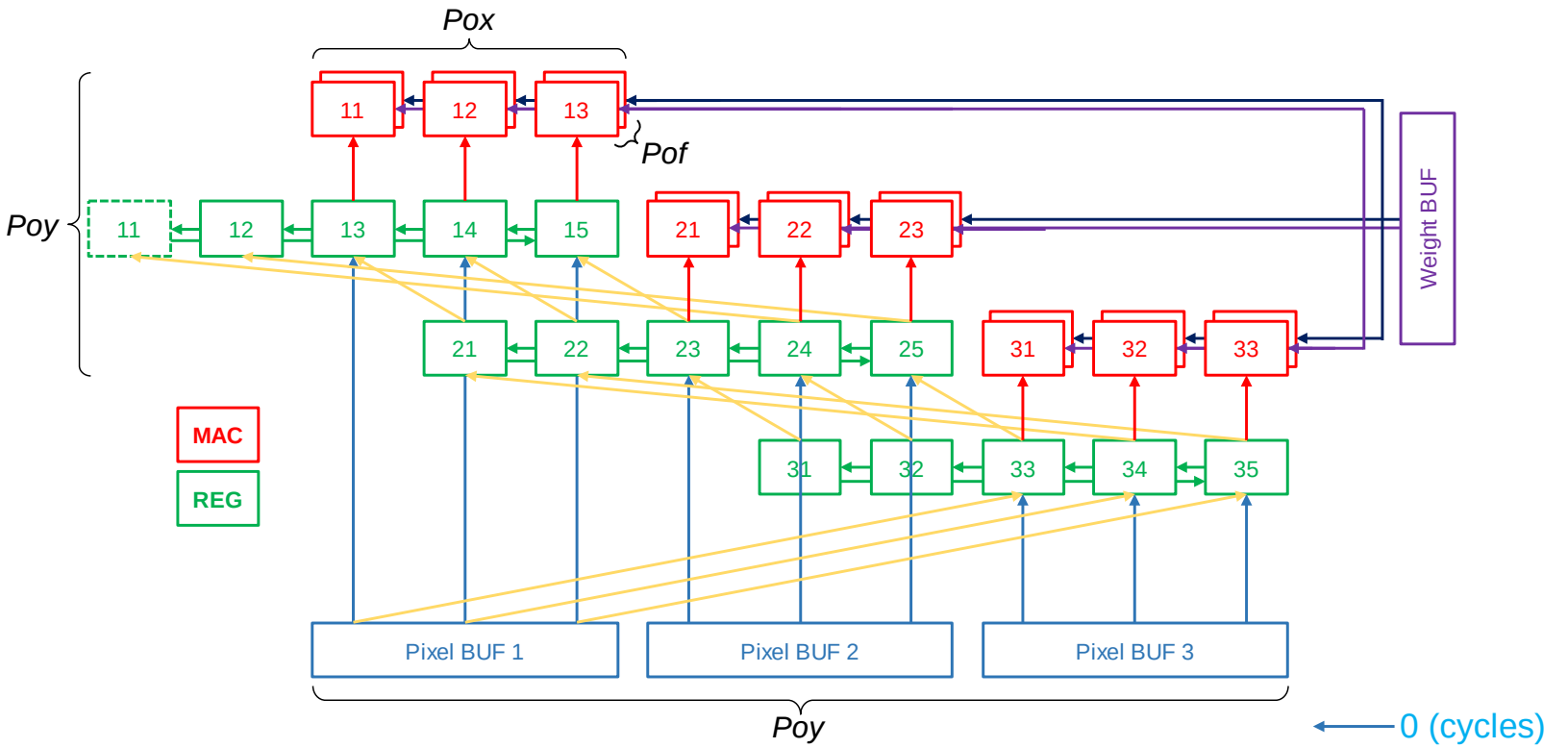
- **Loop-1: Nkx×Nky cycles**
- **Loop-1 & Loop-2: Nkx×Nky×Nif cycles**

On-Chip Storage vs. Off-Chip Comm.



- Estimated results shown for various *Toy* & *Tof* values
- Altera Arria-10 FPGA (GX570-1150): 36-54Mb on-chip memory

Proposed Accelerator Architecture



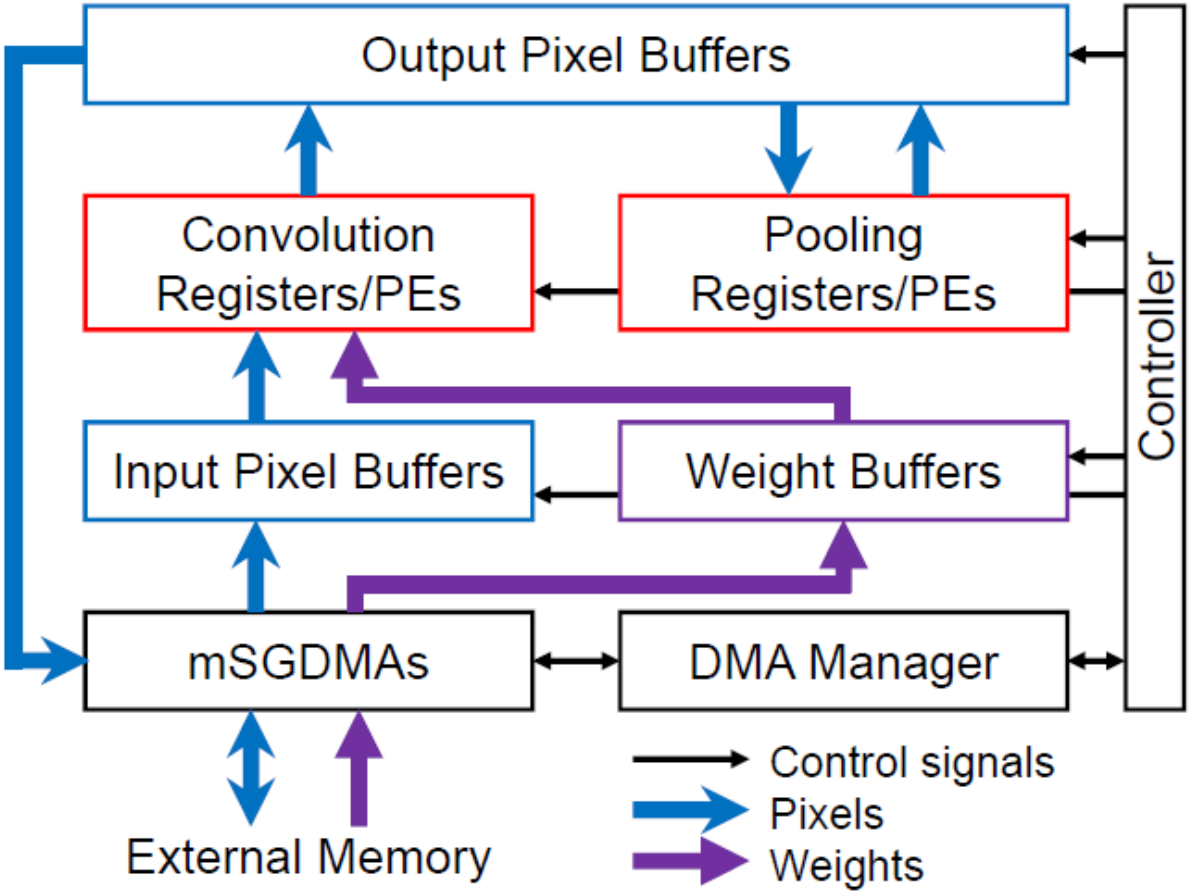
# of MACs = *Pox* × *Poy* × *Pof*

# of Registers = ((*Pox*-1)S+K) × *Poy*

One MAC is comprised of one multiplier followed by an accumulator.

- ***Pox*Poy*Pof*** **PEs and *Poy* input pixel registers**
- **Maximize input data re-use across different output feat. maps**

Top-Level CNN Acceleration System



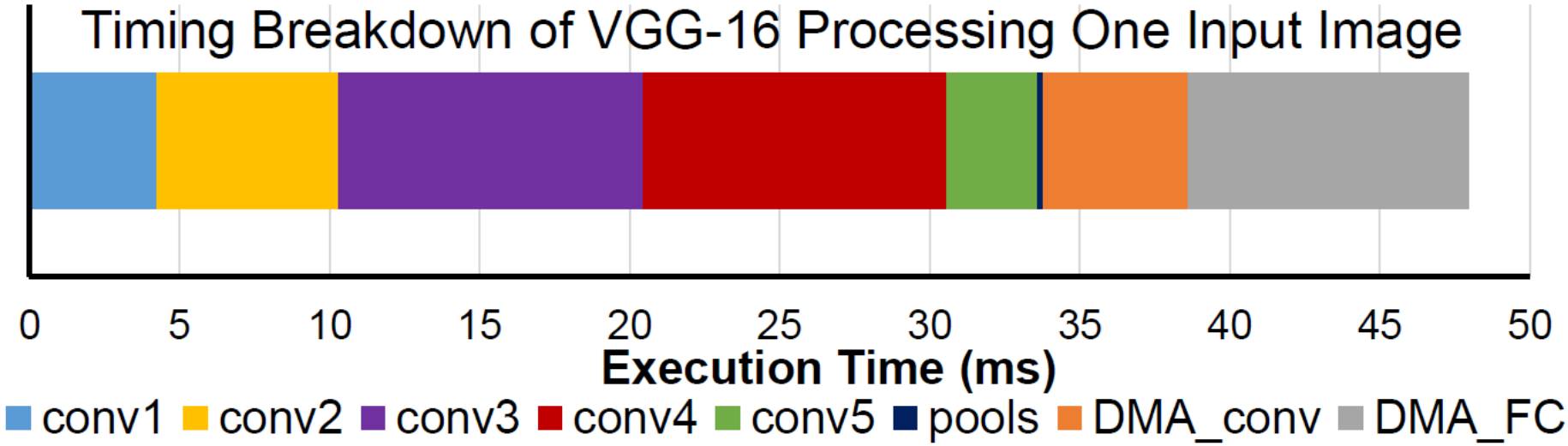
- **Implemented with parametrized Verilog scripts**
- **On-board: 2 banks of DDR3L SDRAM**

Comparison with Prior Works

	[10] VGG	[9] VGG	[8] VGG	This work: VGG
FPGA	Zynq XC7Z045	Stratix-V GSD8	Virtex-7 VX690t	**Arria-10 GX 1150**
Frequency (MHz)	150	120	150	**150**
# Operations (GOP)	30.76	30.95	30.95	**30.95**
Number of Weights	50.18 M	138.3 M	138.3 M	**138.3 M**
Precision (all fixed)	16 bit	8-16 bit	16 bit	**8-16 bit**
DSP Utilization	780 (89%)	1,963[d]	3,600[d]	**1,518 (100%)**
Logic Utilization[a]	183K (84%)	262K[d]	693K[d]	**161K (38%)**
On-chip RAM[b]	486 (87%)	2,567[d]	1,470[d]	**1,900 (70%)**
Latency/Image (ms)	224.6	262.9	151.8	**47.97**
Throughput (GOPS)	136.97	117.8	203.9	**645.25**

- **VGG-16: 13 convolution layers, 5 pooling layers, 3 FC layers, 138.3 million parameters**
- **For VGG model, lowest latency & highest throughput of 645 GOPS achieved for end-to-end operation**

End-to-End Latency Breakdown



- **Convolution layers dominates the total latency (70.0%)**
- **DMA_conv (SDRAM transaction delay of conv. weights & input/output pixels): 10.1% of the overall latency,**
- **FC computing time is hidden by FC weights transfer delay through DMA (DMA_FC)**

Outline

- We are mainly focusing on the *sorting* function
  - Hardware and memory constraints
  - Distance based clustering [JSSC'13]
  - Pre-processing techniques for autonomous learning [DAC'15]
  - Boundary based clustering [VLSI'16]
  - Neuromorphic clustering [VLSI'16]

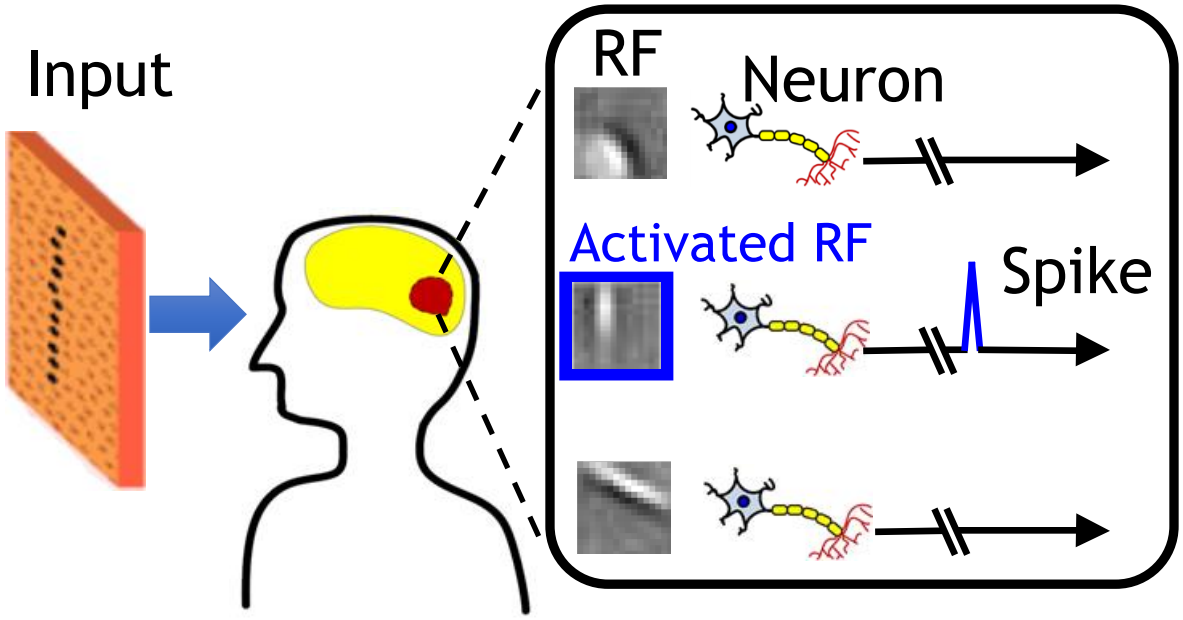
- **Some of remaining questions in this area**

- **Conclusions**

- **Reference**

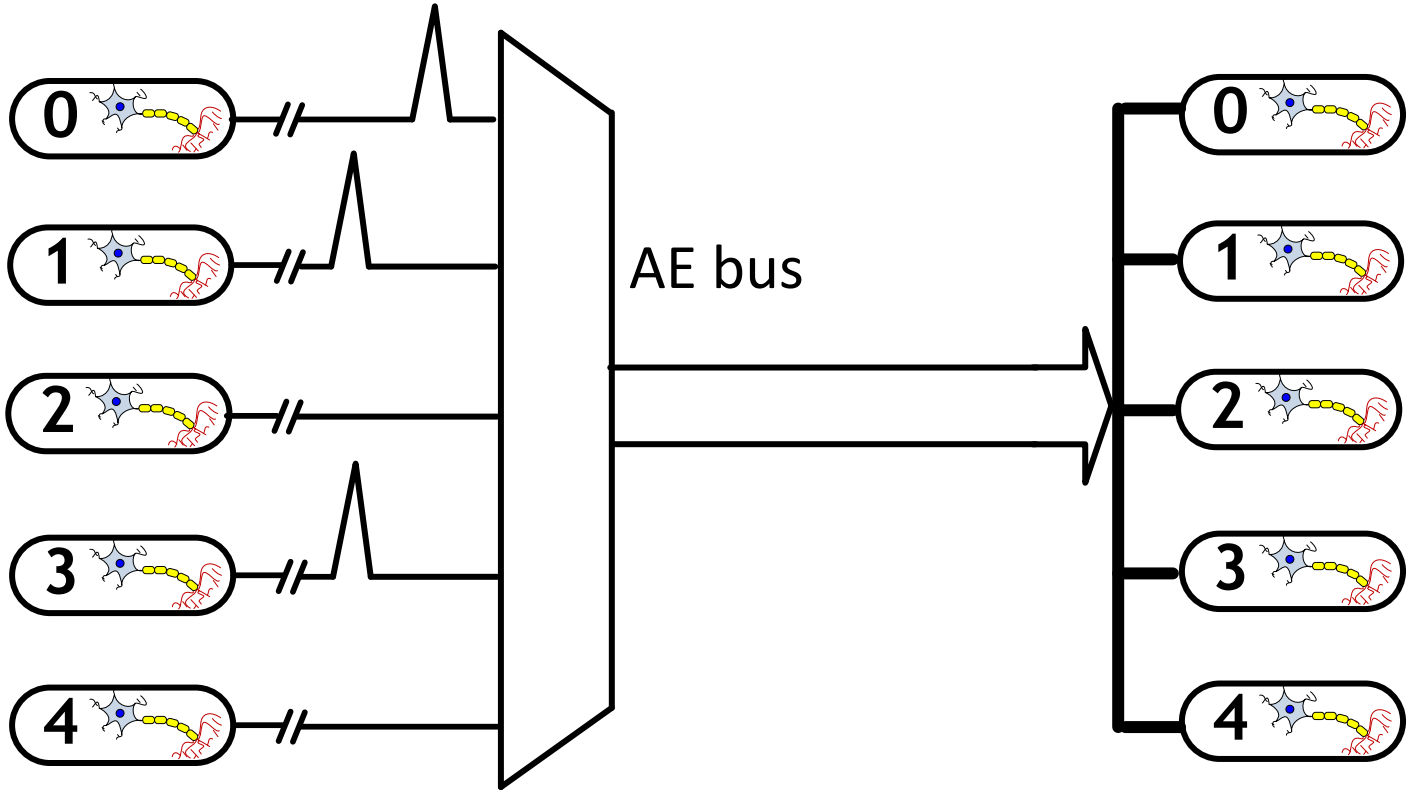
Concept of Sparse Coding

- Learn an overcomplete dictionary by unsupervised training
- Encode an input using a small set of dictionary elements (sparse feature extraction)



Time-Multiplexing Bus

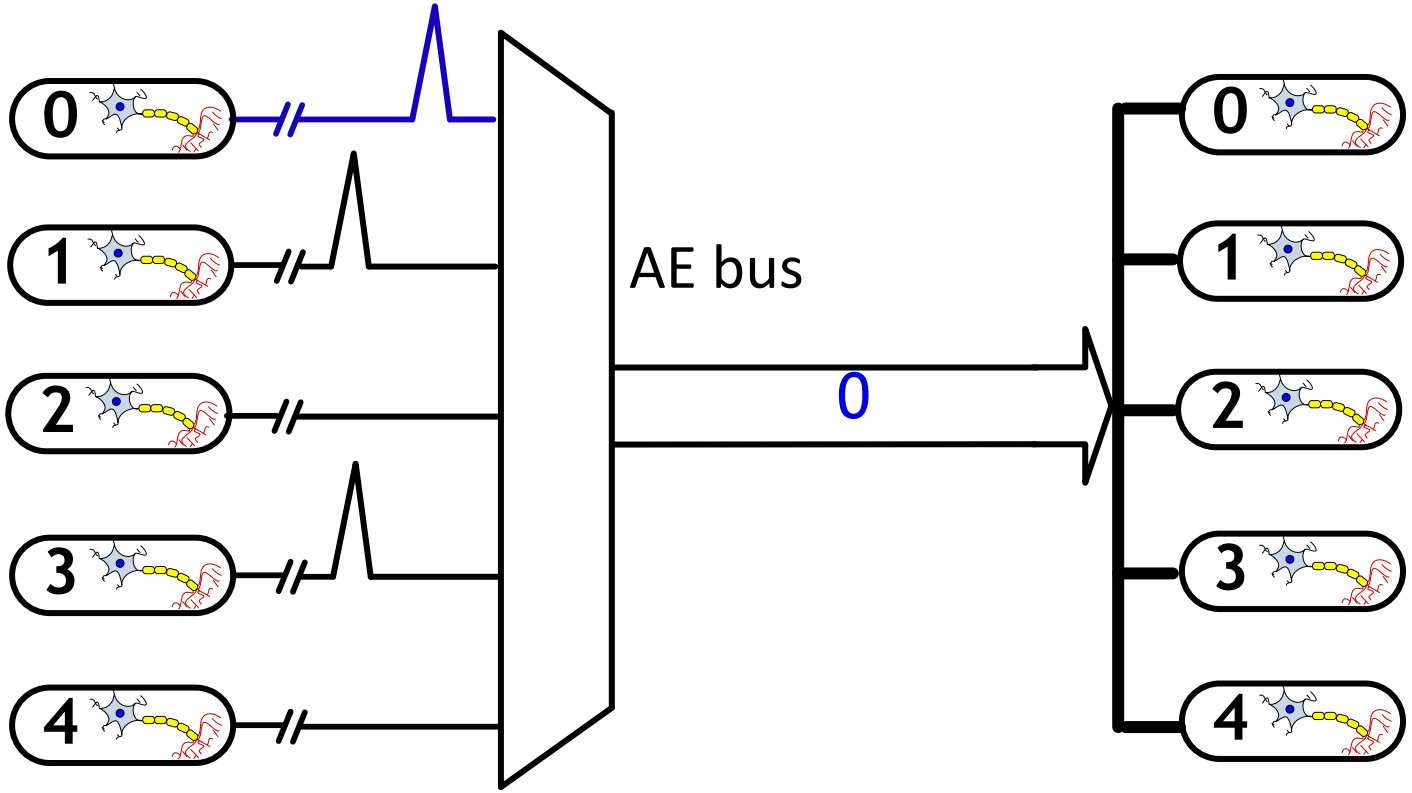
- A neuron spike is encoded by its address, called address event (AE)



M. Mahowald, An Analog VLSI System for Stereoscopic Vision, 1994.

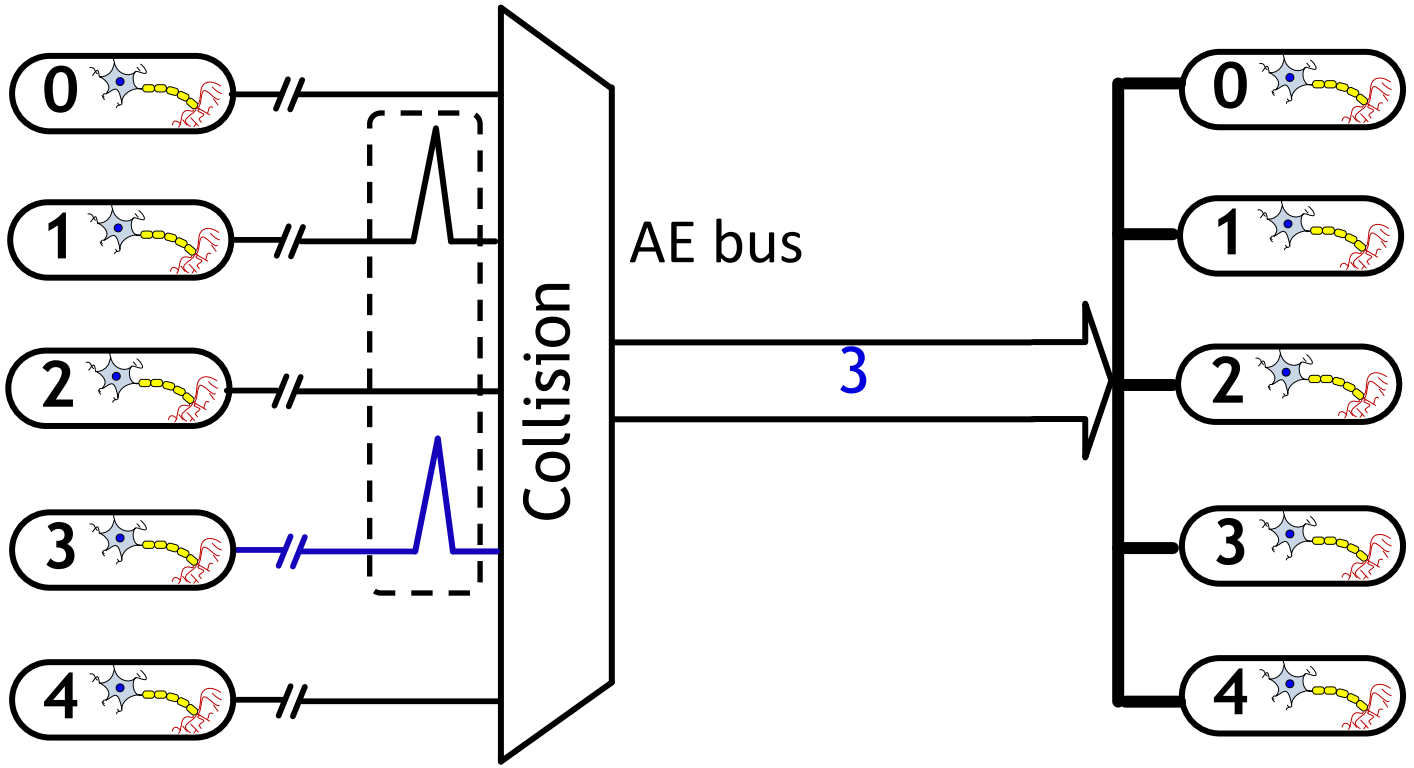
Time-Multiplexing Bus

- Neuron 0 fires



Time-Multiplexing Bus

- Neuron 1 and 3 fire at the same time
- Collision needs to be resolved by an arbiter



Error Tolerance