dMazeRunner: Optimizing Convolutions on Dataflow Accelerators

Shail Dave[1], Aviral Shrivastava[1], Youngbin Kim[2], Sasikanth Avancha[3], Kyoungwoo Lee[2]

[1] Compiler Microarchitecture Lab, Arizona State University
[2] Department of Computer Science, Yonsei University
[3] Parallel Computing Lab, Intel Labs

45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2020)

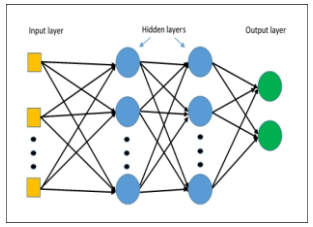
Arizona State University

CML

Must-Accelerate Applications in ML Era

Widely Used ML Models

Multi Layer Perceptrons



http://yann.lecun.com/exdb/lenet/

Convolution Neural Networks

http://vision03.csail.mit.edu/cnn_art/index.html
https://pjreddie.com/darknet/

Sequence Models

http://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
https://deeplearning.mit.edu/

Reinforcement Learning

AlphaGo.
https://www.nature.com/articles/nature24270

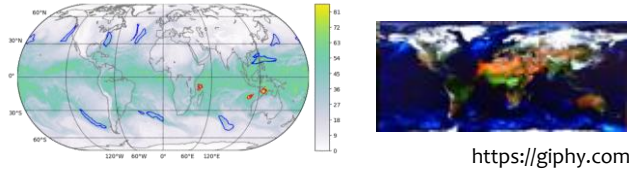
Graph Neural Networks

Points of Interest | Delaunay Triangulation

YOW! Data 2018 Conference.
https://www.youtube.com/watch?v=lDRb3CjESmM

Popular Applications

- **Object Classification/Detection**
- **Media Processing/Generation**
- **Large-Scale Scientific Computing**



https://giphy.com

Tropical Cyclon Detection
https://insidehpc.com/2019/02/gordon-bell-prize-highlights-the-impact-of-ai/

- **Designing Software 2.0**

Google shrinks language translation code from 500k LoC to 500

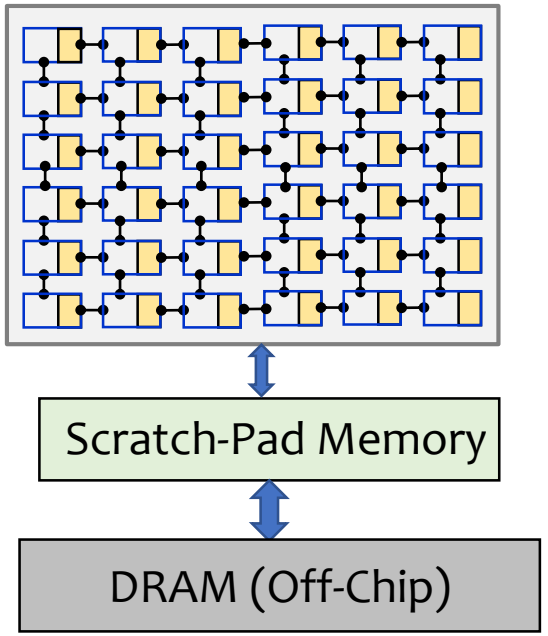
Designing Computer Systems for Software 2.0.
Kunle Olukotun, NeurIPS 2018 Invited talk.

- **and more …**

Compute Intensive,
Needs Energy-efficient Acceleration

Dataflow Accelerators: Promising Solution

Known variations include - Systolic Arrays,
- Spatially Programmable Architecture,
- Coarse-Grain Reconfigurable Array (CGRA)



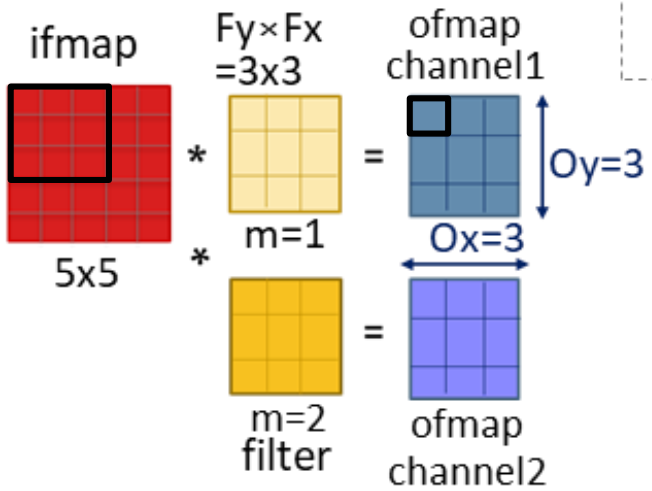
- **Massive arrays of processing elements (PEs).**
  - Simple: absence of complex out-of-order pipeline and decoding.
  - Programmable: accommodate executing all operations within a loop.
- Private and shared memory for PEs sustain data reuse.
- PEs can be busy performing computations while data is being communicated from lower memories.
  - Taken care by effective data management i.e., software prefetching, data distribution, data allocation.

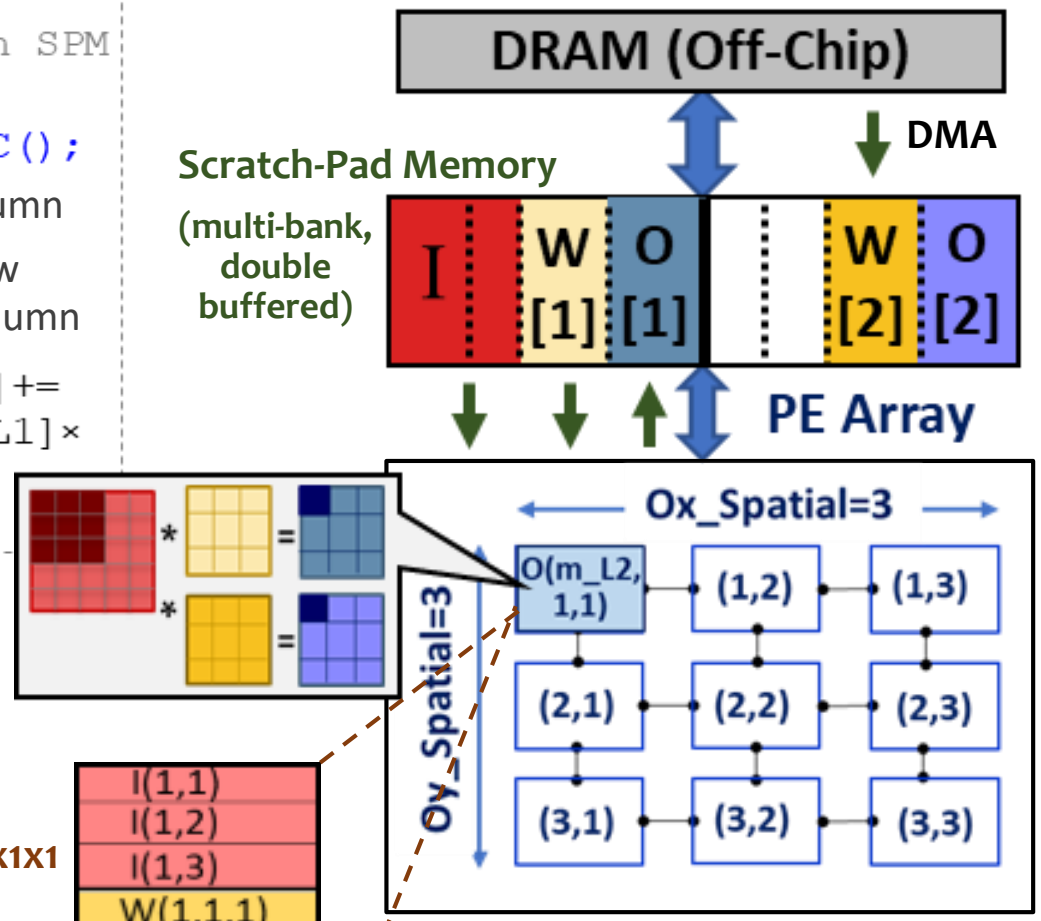
[1] Norman Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In ISCA 2017. [TPU v1]
[2] Yu-Hsin Chen et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep cnns. In JSSC 2016. [Eyeriss System, MIT]
[3] Kamran Khan. 2018. Xilinx DNN Processor (xDNN), Accelerating AI in Datacenters. [Xilinx xDNN]
[4] Bruce Fleischer et al., A Scalable Multi-TeraOPS Core for AI Training and Inference. In VLSI 2018. [IBM AI Accelerator]
[5] Dataflow Processing Unit from Wave Computing. In HOTCHIPS 2017. [Wave DPU]
[6] M. Thottethodi and T. N. Vijaykumar. Why GPGPU is Less Efficient than TPU for DNNs. ACM SIGARCH Blog, Jan 2019. (online)

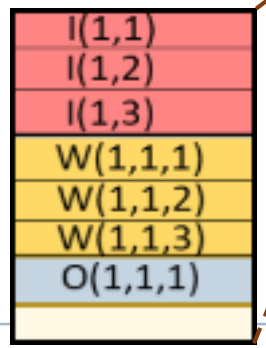
Mapping Computation in Space and Time

```
for m=1:2
 for oy=1:3
  for ox=1:3
   for fy=1:3
    for fx=1:3
     O[m][oy][ox]+=
      I[oy+fy-1][ox+fx-1]
      × W[m][fy][fx];
```

```
% access DRAM
for m_L3=1:2      % filters
 dma()  % prefetch data in SPM
 for fy_L2=1:3    % filter row
  access_SPM_and_comm_NoC();
  for fx_L1=1:3   % filter column
   for oy_S=1:3   % o/p row
    for ox_S=1:3  % o/p column
     O[m_L3][oy_S][ox_S]+=
     W[m_L3][fy_L2][fx_L1]×
     I[oy_S+fy_L2-1]
      [ox_S+fx_L2-1];
```







Data in RF of PE(1,1)
I: 1x3 , W: 1x1x3, O: 1x1x1

Execution on PE(1,1): O[m_L3][1][1] +=
W [m_L3][fy_L2][fx_L1] × I [fy_L2][fx_L1]

PEs compute different outputs

Vast "Execution Method" Space

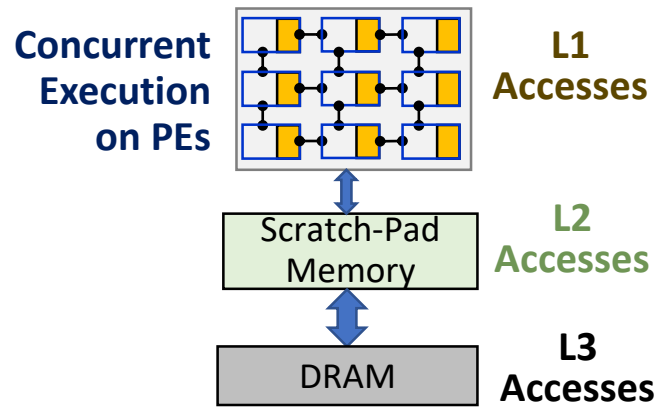
- Many many ways to execute nested loops (of DNN) on a dataflow accelerator
  - Both software and hardware design space
  - **Hardware:** Size, layout, and connectivity of PEs, size of on-chip buffer and registers, etc.
  - **Software:** loop mappings, e.g., **Spatial:** parallelism, data reuse, **Temporal:** ordering and tiling of the loops, data reuse, data buffering, etc.

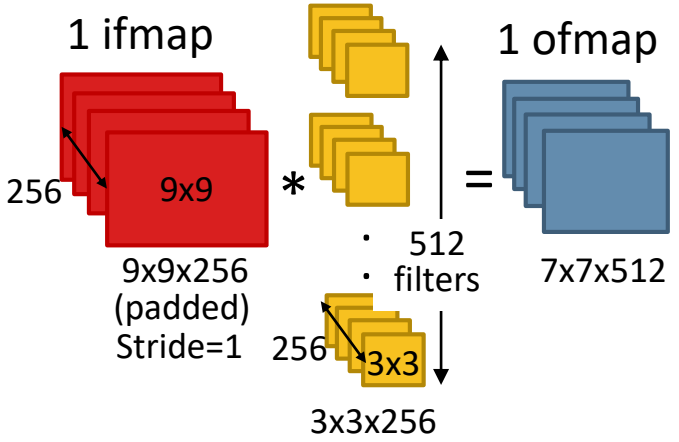
4D Convolution:

```
for n=1:N                % batch size
  for m=1:M              % filters
    for c=1:C            % channels
      for ox=1:Ox        % rows of ofmap
        for oy=1:Oy      % columns of ofmap
          for fx=1:Fx    % filter height
            for fy=1:Fy  % filter width
              O[n][m][ox][oy] +=
                I[n][c][ox+fx-1][oy+fy-1]*
                W[m][c][fx][fy];
```

<N,M,C,Ox,Oy,Fx,Fy> =
<1,512,256,7,7,3,3>

Hardware Accelerator





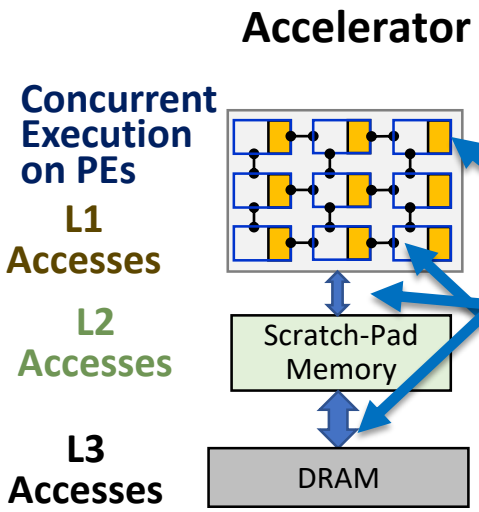
Conv5_1 [ResNet]

Orchestration of Loops

Convolution Operator:

```
for n=1:N              % batch size
  for m=1:M            % filters
    for c=1:C          % channels
      for ox=1:Ox      % rows of ofmap
        for oy=1:Oy    % columns of ofmap
          for fx=1:Fx  % filter height
            for fy=1:Fy  % filter width
              O[n][m][ox][oy] +=
                I[n][c][ox+fx-1][oy+fy-1]*
                W[m][c][fx][fy];
```

7-deep nested loop

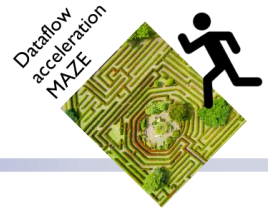



```
for n_L3 = 1:N_DRAM
 for m_L3 = 1:M_DRAM
  for c_L3 = 1:C_DRAM
   for ox_L3 = 1:Ox_DRAM
    for oy_L3 = 1:Oy_DRAM
     for fx_L3 = 1:Fx_DRAM
      for fy_L3 = 1:Fy_DRAM
      {
        dma( );
       for n_L2 = 1:N_SPM
        for m_L2 = 1:M_SPM
         for c_L2 = 1:C_SPM
          for ox_L2 = 1:Ox_SPM
           for oy_L2 = 1:Oy_SPM
            for fx_L2 = 1:Fx_SPM
             for fy_L2 = 1:Fy_SPM
             {
               communicate_data_NoC( );
              for n_L1 = 1:N_RF
               for m_L1 = 1:M_RF
                for c_L1 = 1:C_RF
                 for ox_L1 = 1:Ox_RF
                  for oy_L1 = 1:Oy_RF
                   for fx_L1 = 1:Fx_RF
                    for fy_L1 = 1:Fy_RF
                    {
                     for n_S = 1:N_SPATIAL
                      for m_S = 1:M_SPATIAL
                       for c_S = 1:C_SPATIAL
                        for ox_S = 1:Ox_SPATIAL
                         for oy_S = 1:Oy_SPATIAL
                          for fx_L3 = 1:Fx_SPATIAL
                           for fy_L3 = 1:Fy_SPATIAL
                              O[][][][] +=
                               I[][][][] *
                               W[][][][];
                    }
             }
      }
}
```

28-deep nested loop

The problem of exploring the "execution methods" for mapping 7-deep nested loop onto dataflow accelerator becomes the problem of exploring all the possibilities of tiling and ordering factors in the 28-deep nested loop

dMazeRunner Demo



Framework Features

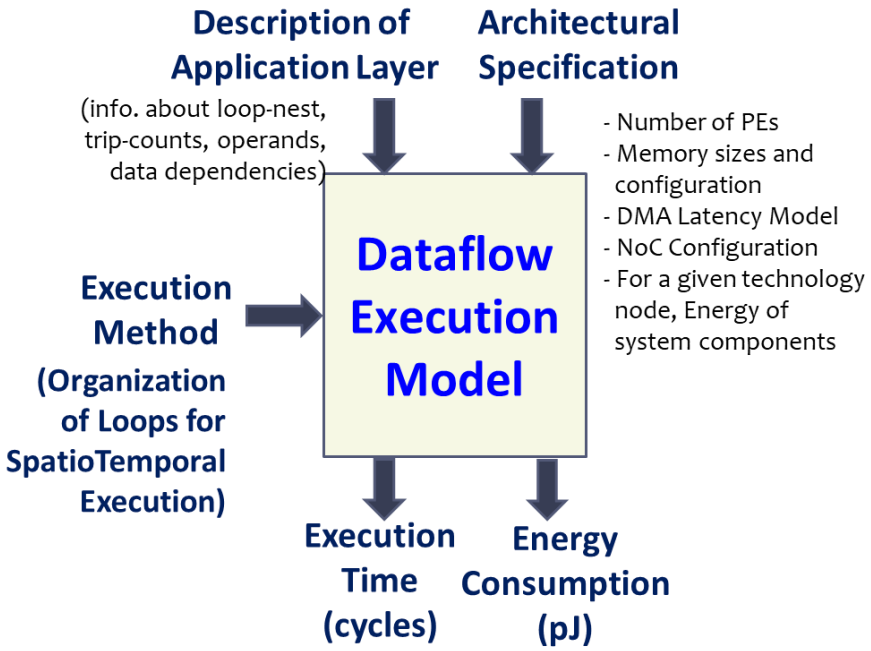
1. Estimate performance and energy-efficiency of individual execution method for a specified accelerator hardware and layer dimensions

2. Optimize specific or all layers of common CNN networks

3. Explore efficient accelerator designs for DNNs

Release: https://github.com/MPSLab-ASU/dMazeRunner

Execution Modeling of Dataflow Accelerators

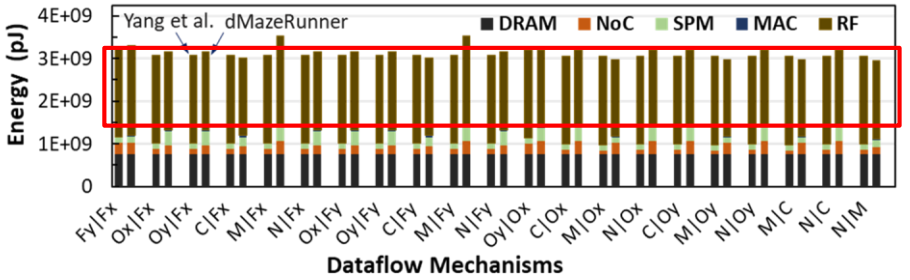


Features with detailed modeling of

- ✓ Analyze **arbitrary perfectly nested loops.**
- ✓ **miss penalty and stall cycles** (during PE execution and in managing PE's local or shared memory).
- ✓ **inter-PE communication.**
- ✓ **temporal/spatial data reuse.**
- ✓ Integrated **support common ML libraries** MXNet/Keras/... (thanks TVM! – leveraging front-end)

Validation against DNN Optimizer of Yang et al.

Yang, Xuan, M. Gao, J. Pu, A. Nayak, Q. Liu, S. Bell, J. Setter, K. Cao, H. Ha, Christos Kozyrakis, and Mark Horowitz. "DNN Dataflow Choice Is Overrated." [arXiv '18]



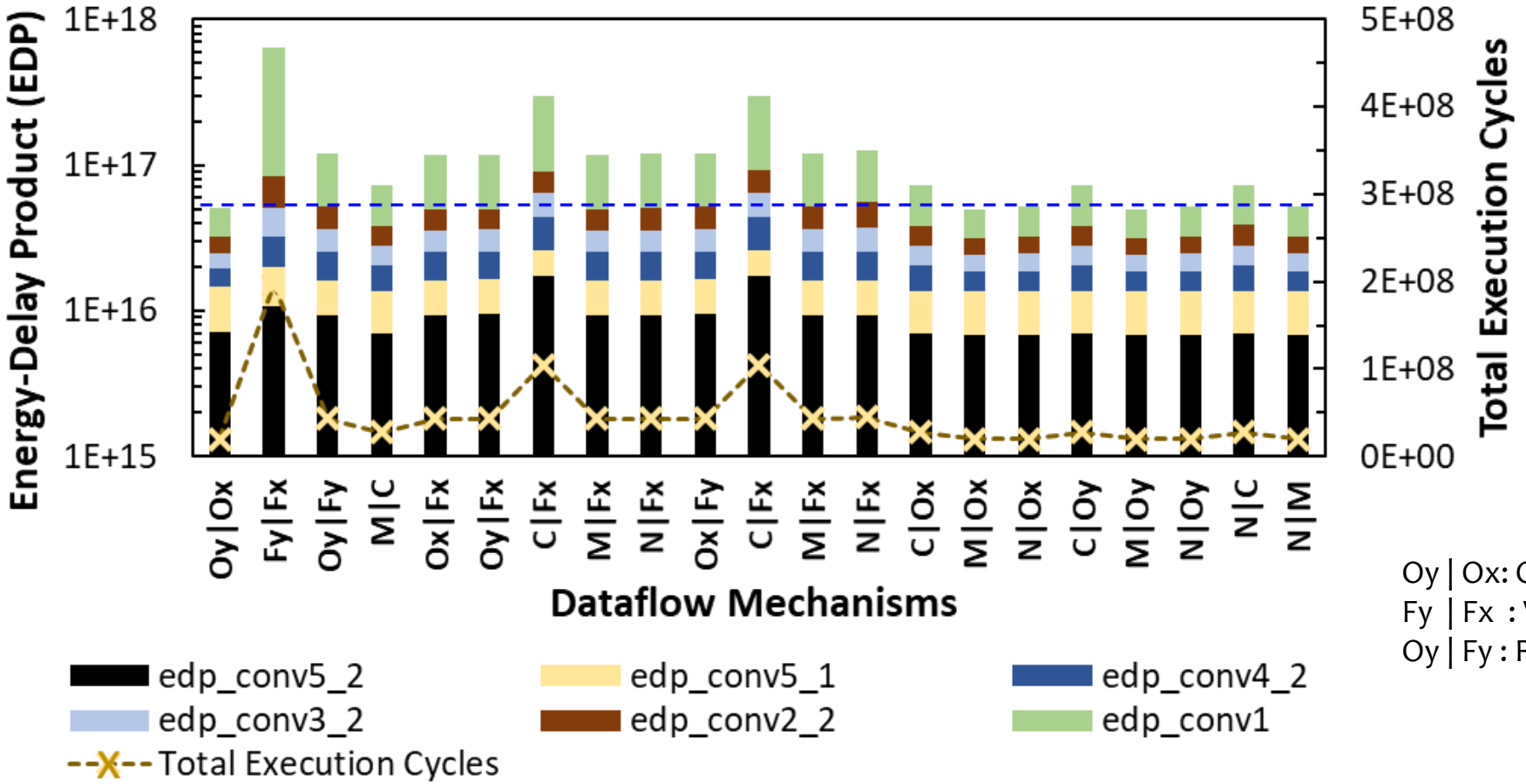
- Energy estimate differs by ~4.2% for variety of execution methods.
- For efficient mappings, major energy spent in RF accesses.

Step-wise equations and analysis:

Shail Dave, Youngbin Kim, Sasikanth Avancha, Kyoungwoo Lee, Aviral Shrivastava, dMazeRunner: Executing Perfectly Nested Loops on Dataflow Accelerators [CODES+ISSS, TECS 2019].

Optimizing Multiple Dataflows

Executing ResNet layers on a 256-PE, 512B RF, 128kB SPM dataflow accelerator



Oy | Ox: Output Stationary
Fy | Fx : Weight Stationary
Oy | Fy : Row Stationary

Adaptable Mappings Yield Better Results

- Adapts to layer / hardware architecture characteristics
  - **Scales for layers and tensors of different shapes**
- Finds non-intuitive mappings that are optimized for various key factors e.g.,
  - ✓ **High Resource (PE/memories) Utilization**
  - ✓ **Maximized Reuse of Multiple Tensors**
  - ✓ **Minimized DRAM accesses**
  - ✓ **Hiding Communication Latency Behind Computations on PEs**

Example Mapping of ResNet Conv5_2 with Output Stationary Dataflow

For data allocated in RFs of PEs,	**MOC**	**dMazeRunner**
PE Compute vs. Data comm. Latency:	144 vs. 648	**576 vs. 576**
Total cycles:	~10,616,832	~2,459,648
Ideal execution cycles for output-stationary:	2,359,296	2,359,296
Reduction in DRAM accesses (ifmaps, weights):	(1x, 1x)	**(4.57x, 2x)**
Perf. improvement (normalized to MOC):	1x	**4.44x**
Energy-Delay-Product reduction (normalized):	1x	**9.86x**

MOC: Simultaneous spatial processing of Multiple Output Channels

[1] S. Gupta et al. Deep learning with limited numerical precision. In ICML, 2015.
[2] Y. Chen et al. Eyeriss: A spatial architecture for energy-efficient dataflow for CNNs. In ISCA 2016.

Achieving Close-to-Optimal Solutions in Seconds

Search Space Reduction for DNN Optimizations

ResNet Conv Layers	Loop Tilings	Loop Orderings	dMzRnr explored Tilings	dMzRnr explored Orderings
1	1.2E+08	7!×7!	46812	3×3
2_2	1.1E+09	7!×7!	122092	3×3
3_2	6.2E+08	7!×7!	53690	3×3
4_2	1.8E+08	7!×7!	10938	3×3
5_1	1.4E+07	7!×7!	877	3×3
5_2	1.7E+07	7!×7!	753	3×3

- **Even domain non-experts can explore the space**

```
python run_optimizer.py --frontend mxnet
       --model resnet18 --layer-index 0
```

- **Does not preclude experts**/programmers from directing the search.

Search Space Exploration on an Intel i7-6700 Quad-core CPU
min: ~1 second, ResNet conv5_2 (753*9 methods)
max: ~122 seconds, ResNet conv2_2 (122092*9 methods)

AlexNet and ResNet18 models in about 18 and 180 seconds, respectively

- Quick exploration:
  - In-built support for a few common opt strategies.
  - Implementation – **multi-threaded, caches commonly invoked routines** of analytical model.
  - Enables effective Design Space Exploration of architecture, e.g., explore impact of scaling PEs and memory sizes on performance and energy efficiency.

CML

Conclusions

- **Dataflow accelerators:** promising for accelerating ML applications.
- **Need to determine efficient "execution method" for spatiotemporal executions** on dataflow accelerators.
- **dMazeRunner: Automated, succinct, and fast exploration of mapping** search space **and** architecture **design space.**
- **Adaptive and non-intuitive mappings** enable efficient dataflow acceleration.

[Release] https://github.com/MPSLab-ASU/dMazeRunner
[Project] https://labs.engineering.asu.edu/mps-lab/ml-accelerators