# Using Dataflow to Optimize Energy Efficiency of Deep Neural Network Accelerators

Yu-Hsin Chen*, Joel Emer*† and Vivienne Sze*
*Massachusetts Institute of Technology
†NVIDIA

**Abstract**—The authors demonstrate the key role dataflows play in the optimization of energy efficiency for deep neural network (DNN) accelerators. By introducing a systematic approach to analyze the problem and a new dataflow, called row-stationary, that is up to 2.5 times more energy efficient than existing dataflows in the processing of a state-of-the-art DNN, this work provides guidelines for future DNN accelerator designs.

**Index Terms**—Neural Networks, Accelerators, Dataflow, Energy Efficiency, Spatial Architecture

## 1 Introduction

The recent breakthroughs in deep neural networks (DNNs) are leading to an industrial revolution based on artificial intelligence (AI). The superior accuracy of DNNs, however, comes at the cost of high computational complexity. General-purpose processors no longer deliver sufficient processing throughput and energy efficiency for DNNs. As a result, demands for dedicated DNN accelerators are on the rise in order to support the rapidly growing use of AI.

The processing of a DNN mainly consists of multiply-and-accumulate (MAC) operations (Fig. 1). Most of these MACs are performed in the convolutional layers of the DNN, where multi-channel filters are convolved with multi-channel input feature maps (ifmaps, such as images). This generates partial sums (psums) that are further accumulated into output feature maps (ofmaps). Since the MAC operations have little data dependency, DNN accelerators can use high parallelism to achieve high processing throughput. However, this processing also requires a significant amount of data movement: each MAC performs 3 reads and 1 write of data access. As moving data can consume more energy than the computation itself [1], optimizing data movement becomes the key to achieving high energy efficiency.

Data movement can be optimized by exploiting data reuse in a multi-level storage hierarchy. By maximizing the reuse of data in the lower-energy-cost storage levels, e.g., local scratchpads, thus minimizing data accesses to the higher-cost levels, e.g., DRAM, the overall data movement energy consumption is minimized.

In fact, DNNs present many data reuse opportunities. First, there are three types of input data reuse: (1) filter reuse, where each filter weight is reused across multiple ifmaps, (2) ifmap reuse, where each ifmap pixel is reused across multiple filters, and (3) convolutional reuse, where both ifmap pixels and filter weights are reused due to the sliding window processing in convolutions. Second, the intermediate psums are reused through the accumulation of ofmaps. If not accumulated and reduced as soon as possible, the psums may pose additional storage pressure.

These data reuse opportunities are exploited by finding the optimal MAC operation *mapping*, which determines both the temporal and spatial scheduling of the MACs on a highly-parallel architecture, so that data in the lower-cost storage levels can be reused by as many MACs as possible before replacement. However, due to the limited amount of local storage, input data reuse (ifmaps and filters) and psum reuse cannot be fully exploited simultaneously. For example, reusing the same input data for multiple MACs generates psums that cannot be accumulated together and, as a result, consume extra storage space. Therefore, the system energy efficiency is maximized only when the mapping balances all types of data reuse in a multi-level storage hierarchy.

The search for the mapping that maximizes system energy efficiency thus becomes an optimization process. This optimization has to take into account the following factors: (1) the available data reuse opportunities, which is dictated by the DNN shape and size (e.g., number of filters, number of channels, size of filters, etc.), (2) the energy cost of data access at each level of the storage hierarchy, and (3) the available processing parallelism and storage capacity. Factors (2) and (3) are a function of the specific accelerator implementation.

Because of implementation trade-offs, previous proposals for DNN accelerators have made choices on the subset of mappings that can be supported. Therefore, for a specific DNN accelerator design, the optimal mapping can only be selected from the subset of supported mappings instead of the entire mapping space. The subset of supported mappings is usually determined by a set of mapping rules, which also characterizes the hardware implementation. Such a set of mapping rules defines a *dataflow*.

Since state-of-the-art DNNs come in a wide range of shapes and sizes, the corresponding optimal mappings also vary. The question is, can we find a dataflow that accommodates the mappings that optimize data movement for various DNN shapes and sizes?

In this article, we explore different DNN dataflows to answer this question in the context of a spatial architecture

Digital Object Identifier 10.1109/MM.2017.265085944 0272-1732/$26.00 2017 IEEE

[2]. In particular, we will present the following key contributions [3]:

- An analogy between DNN accelerators and general-purpose processors that clearly identifies the distinct aspects of operation of a DNN accelerator, which provides insights into opportunities for innovation.
- A framework that quantitatively evaluates the energy consumption of different mappings working for any DNN shapes and sizes, which is an essential tool for finding the optimal mapping.
- A taxonomy that classifies existing dataflows from previous DNN accelerator projects, which helps to understand a large body of work despite differences in the lower-level details.
- A new dataflow, called Row-Stationary (RS), which is the first dataflow that optimizes data movement for superior system energy efficiency. It has also been verified in a fabricated DNN accelerator chip, *Eyeriss* [4].

We evaluate the energy efficiency of the RS dataflow and compare it to other existing ones from the taxonomy. The comparison uses one of the most popular state-of-the-art DNN model, AlexNet [5], with a fixed amount of hardware resources. Simulation results show that the RS dataflow is 1.4× to 2.5× more energy efficient than other dataflows in the convolutional layers; it is also at least 1.3× more energy efficient in the fully-connected layers for batch sizes of at least 16. These results will provide guidelines for future DNN accelerator designs.

## 2 An Analogy To General-Purpose Processors

The operation of DNN accelerators is analogous to that of general-purpose processors as illustrated in Fig. 2. In conventional computer systems, the compiler translates the program into machine-readable binary codes for execution; in the processing of DNNs, the mapper translates the DNN shape and size into a hardware-compatible mapping for execution. While the compiler usually optimizes for performance, the mapper optimizes for energy efficiency.

The dataflow is a key attribute of a DNN accelerator that is analogous to one of the parts of the architecture of a general-purpose processor. Similar to the role of an ISA or memory consistency model, the dataflow characterizes the hardware implementation and defines the mapping rules that the mapper has to follow in order to generate hardware-compatible mappings. Later in this article, we will introduce several existing dataflows that are widely used in implementations.

Other attributes of a DNN accelerator such as the storage organization also correspond to parts of the general-purpose processor architecture, such as scratchpads or virtual memory. We consider these attributes as a part of the architecture, instead of microarchitecture, since we believe they are going to largely remain invariant across implementations. Although, like GPUs, the distinction between architecture and microarchitecture is likely to blur for DNN accelerators due to its rapid development.

The implementation details, including access energy cost at each level of the storage hierarchy and latency between processing elements (PE), are analogous to the microarchitecture of processors because a mapping will be valid despite changes in these characteristics. However, they play a vital part in the energy efficiency of a mapping.

The goal of the mapper is to search in the mapping space for the best one that optimizes data movement. The size of the entire mapping space is determined by the total number of MACs, which can be calculated from the DNN shape and size. However, only a subset of the space is valid given the mapping rules defined by a dataflow. For example, the dataflow can enforce the following mapping rule: all MACs that use the same filter weight have to be mapped on the same PE in the accelerator. Then, it is the mapper's job to find out the exact ordering of these MACs on each PE by evaluating and comparing the energy efficiency of different valid ordering options.

As in conventional compilers, performing evaluation is an integral part of the mapper. The evaluation process takes a certain mapping as input and gives an energy consumption estimation based on the available hardware resources (microarchitecture) and data reuse opportunities extracted from the DNN shape and size (program). In the next section, we will introduce a framework that can perform this evaluation.

## 3 Evaluating Energy Consumption

Finding the optimal mapping requires evaluation of the energy consumption between various mapping options. In this work, we will be evaluating energy consumption based on a spatial architecture [2], since many of the previous works can be thought of as instances of such an architecture. The spatial architecture (Fig. 3) consists of an array of processing elements (PE) and a multi-level storage hierarchy. The PE array provides high parallelism for high throughput; the storage hierarchy can be used to exploit data reuse in a four-level setup (in decreasing energy cost order): DRAM, global buffer, network-on-chip (NoC, for inter-PE communication), and register file (RF) in the PE as local scratchpads.

In this architecture, we assume all data types can be stored and accessed at any levels of the storage hierarchy. Input data for the MAC operations, i.e., filter weights and ifmap pixels, are moved starting from the most expensive level, i.e., DRAM, to the lower-cost levels. Ultimately, they are mostly delivered from the least expensive level, i.e., RF, to the ALU for computation. The results from the ALU, i.e., psums, move generally in the opposite direction. The orchestration of this movement is determined by the mappings for a specific DNN shape and size under the mapping rule constraints of a specific dataflow.

Given a specific mapping, the system energy consumption is estimated by accounting for the number of times each data value from all data types (ifmaps, filters, psums) is reused at each level of the four-level memory hierarchy, and weighing it with the energy cost of accessing that specific storage level. Fig. 4 shows the normalized energy consumption of accessing data from each storage level relative

Digital Object Identifier 10.1109/MM.2017.265085944 0272-1732/$26.00 2017 IEEE

to the computation of a MAC at ALU. These numbers are extracted from a commercial 65nm process, and are used in our final experiments.

Fig. 5 uses a toy example to show how a mapping determines the data reuse at each storage level, and thus the energy consumption, in a 3-PE setup. In this example, we have the following assumptions: (1) each ifmap pixel is used by 24 MACs, (2) all ifmap pixels can fit into the global buffer, and (3) the RF of each PE can only fit 1 ifmap pixel at a time. The mapping first reads an ifmap pixel from DRAM to the global buffer, then from the global buffer to the RF of each PE through the NoC, and reuses it from the RF for 4 MACs consecutively in each PE. The mapping then switches to MACs that use other ifmap pixels, so the original one in the RF is replaced by new ones due to limited capacity. Therefore, the original ifmap pixel has to be fetched from the global buffer again when the mapping switches back to the MACs that use it. In this case, the same ifmap pixel is reused at the DRAM, global buffer, NoC, and RF for 1, 2, 6, and 24 times, respectively. The corresponding normalized energy consumption of moving this ifmap pixel is obtained by weighing these numbers with the normalized energy numbers in Fig. 4, and then adding them together, i.e., 1×200+2×6+6×2+24×1 = 248. For other data types, the same approach can be applied.

This analysis framework can be used not only to find the optimal mapping for a specific dataflow, but also to evaluate and compare the energy consumption of different dataflows. In the next section, we will describe the various existing dataflows.

## 4 A Taxonomy of Existing DNN Dataflows

Numerous previous efforts have proposed solutions for DNN acceleration. These designs reflect a variety of trade-offs between performance and implementation complexity. Though with their differences in low-level implementation details, we find that many of them can be described as embodying a set of rules, i.e., *dataflow*, that define the valid mapping space based on how they handle data. As a result, we are able to classify them into a taxonomy of three dataflow categories. In the rest of this section, we will provide a high-level overview on each of the three dataflows.

- **Weight-Stationary (WS) dataflow**: WS keeps filter weights stationary in the RF of each PE by enforcing the following mapping rule: all MACs that use the same filter weight have to be mapped on the same PE for processing contiguously. This maximizes the convolutional and filter reuse of weights in the RF, thus minimizing the energy consumption of accessing weights (e.g., [6, 7]). Fig. 6a shows the data movement of a common WS dataflow implementation. While each weight stays in the RF of each PE, the ifmap pixels are broadcast to all PEs, and the generated psums are then accumulated spatially across PEs.
- **Output-Stationary (OS) dataflow**: OS keeps psums stationary by accumulating them locally in the RF. The mapping rule is: all MACs that generate psums for the same ofmap pixel have to be mapped on the same PE contiguously. This maximizes psum reuse in the RF, thus minimizing energy consumption of psum movement (e.g., [8–10]). The data movement of a common OS dataflow implementation is to broadcast filter weights while passing ifmap pixels spatially across the PE array (Fig. 6b).
- **No Local Reuse (NLR) dataflow**: Unlike the previous two dataflows that keep a certain data type stationary, NLR makes no data stationary locally so it can trade RF off for a large global buffer. This is to minimize DRAM access energy consumption by storing more data on-chip (e.g., [11, 12]). The corresponding mapping rule is: at each processing cycle, all parallel MACs need to come from a unique pair of filter and channel. The data movement of NLR dataflow is to single-cast weights, multi-cast ifmap pixels, and spatially accumulate psums across the PE array (Fig. 6c).

The three dataflows show distinct data movement patterns, which imply different trade-offs. First, as is evident in Fig. 6a and Fig. 6b, the cost for keeping a specific data type stationary is to move the other types of data more. Second, the timing of data accesses also matters. For example, in the WS dataflow, each ifmap pixel read from the global buffer is broadcast to all PEs with properly mapped MACs on the PE array. This is more efficient than reading the same value multiple times from the global buffer and single-casting it to the PEs, which is the case for filter weights in the NLR dataflow (Fig. 6c) due to its mapping restriction. In the next section, we will present a new dataflow that takes these factors into account for optimizing energy efficiency.

## 5 An Energy-Efficient Dataflow

Although the existing dataflows in the taxonomy describe the design of many DNN accelerators, they optimize data movement only for a specific data type (e.g., WS for weights) or storage level (e.g., NLR for DRAM). In this section, we introduce a new dataflow, called Row-Stationary (RS). The RS dataflow aims to optimize data movement for all data types in all levels of the storage hierarchy of a spatial architecture.

The RS dataflow divides the MACs into mapping primitives, each of which consists of a subset of MACs that runs on the same PE in a fixed order. Specifically, each mapping primitive performs a 1D row convolution, which is why we also call it a row primitive, and intrinsically optimizes data reuse per MAC for all data types combined. They are formed with the following rules: (1) the MACs for applying a row of filter weights on a row of ifmap pixels, which generate a row of psums, have to be mapped on the same PE, and (2) the ordering of these MACs enables the use of a sliding window for ifmaps as illustrated in Fig. 7. Convolutional and psum reuse opportunities within a row primitive are fully exploited in the RF given sufficient RF storage capacity.

Even with the RS dataflow as defined by the row primitives, there are still a large number of valid mapping choices. The mapper evaluates these choices to schedule the row primitives on the PE array, both spatially and temporally, so that data movement is optimized across the storage hierarchy. We

Digital Object Identifier 10.1109/MM.2017.265085944 0272-1732/$26.00 2017 IEEE

observe certain mapping patterns that further exploit data reuse opportunities in the lower-cost storage levels:

- One aspect of mapping is how the row primitives are *spatially mapped* onto the PEs in the array. For example, primitives with data rows from the same 2D plane can be spatially mapped on the PE array, which lays out a 2D convolution as shown in Fig. 8. This mapping fully exploits convolutional and psum reuse opportunities across primitives in the NoC: the same rows of filter weights and ifmap pixels are reused across PEs horizontally and diagonally, respectively; psum rows are further accumulated across PEs vertically.
  Another example arises when the size of the PE array is large, and the pattern shown in Fig. 8 can be *spatially duplicated* across the PE array for various 2D convolutions. This not only increases utilization of PEs, but also further exploits filter, ifmap and psum reuse opportunities in the NoC.
- Another aspect of mapping is how the row primitives are *temporally mapped* onto each PE. For example, row primitives from different 2D planes can be *concatenated* or *interleaved* on the same PE. As shown in Fig. 9, primitives with different ifmaps, filters, and channels have filter reuse, ifmap reuse, and psum reuse opportunities, respectively. By concatenating or interleaving their computation together in a PE, it virtually becomes a larger 1D row convolution, which exploits these cross-primitive data reuse opportunities in the RF.
  Another example arises when the PE array size is too small, and the originally spatially-mapped row primitives have to be *temporally folded* into multiple processing passes, i.e., the computation is serialized. In this case, the data reuse opportunities that are originally spatially-exploited in the NoC can be temporally-exploited by the global buffer to avoid DRAM accesses given sufficient storage capacity.

As evident from the preceding list, the RS dataflow provides a high degree of mapping flexibility, such as using *concatenation*, *interleaving*, *duplicating*, and *folding* of the row primitives. The mapper searches for the exact amount to apply each technique in the optimal mapping, e.g., how many filters are interleaved on the same PE to exploit ifmap reuse, to minimize overall system energy consumption.

## 6 Dataflow Comparison

In this section, we quantitatively compare the energy efficiency of different DNN dataflows in a spatial architecture, including the existing ones from the taxonomy and the proposed RS dataflow. We use AlexNet [5] as the benchmarking DNN for the following reasons: (1) it is one of the most popular DNN available, and (2) it consists of 5 convolutional (CONV) layers and 3 fully-connected (FC) layers with a wide variety of shapes and sizes, which can more thoroughly evaluate the optimal mappings from each dataflow.

In order to have a fair comparison, we apply the following two constraints for all dataflows. First, the size of the PE array is fixed at 256 for constant processing throughput across dataflows. Second, the total hardware area is also fixed. For example, since the NLR dataflow does not use RF, it can allocate more area for the global buffer. The corresponding hardware resource parameters are based on the RS dataflow implementation in Eyeriss, a DNN accelerator chip fabricated in 65nm CMOS [4]. By applying these constraints, we fix the total cost to implement the microarchitecture of each dataflow. Therefore, the differences in energy efficiency are solely from the dataflows.

Fig. 10a and Fig. 10b show the comparison of energy efficiency between dataflows in the CONV layers of AlexNet with an ifmap batch size of 16. Fig. 10a has the breakdown in terms of storage levels and ALU; Fig. 10b has the breakdown in terms of data types. First, the ALU energy consumption is only a fraction of the total energy consumption, which proves the importance of data movement optimization. Second, even though NLR has the lowest energy consumption in DRAM, its total energy consumption is still high since most of the data accesses come from the global buffer, which are more expensive than from the NoC or RF. Third, while WS and OS dataflows clearly optimize the energy consumption of accessing weights and psums, respectively, they sacrifice the energy consumption of moving other data types and therefore do not achieve the lowest energy consumption. This shows that DRAM alone does not dictate energy efficiency, and optimizing the energy consumption for only a certain data type does not lead to the best system energy efficiency. Overall, the RS dataflow is 1.4× to 2.5× more energy efficient than other dataflows in the CONV layers of AlexNet.

Fig. 11 shows the same experiment results as in Fig 10b except that it is for the FC layers of AlexNet. Compared to CONV layers, FC layers have no convolutional reuse and use much more filter weights. Still, RS dataflow is at least 1.3× more energy efficient than the other dataflows, which proves that the capability to optimize data movement for all data types is the key to achieving highest overall energy efficiency. It should be noted that FC layers account for less than 20% of the total energy consumption in AlexNet. In recent DNNs, the number of FC layers have also been greatly reduced, making their energy consumption even less significant.

## 7 Summary

Research on architectures for DNN accelerators is getting very popular for its promising performance and wide applicability. This work has demonstrated the key role of dataflows in DNN accelerator design, and shows how to systematically exploit all types of data reuse in a multi-level storage hierarchy for optimizing energy efficiency with a new dataflow. It challenges the conventional design approaches, which focus more on optimizing parts of the problem, and shifts it towards a global optimization that takes all relevant metrics into consideration.

The taxonomy of dataflows condenses the low-level implementation details into categorical pros and cons, which can be used as guidelines to improve future designs. While these dataflows are currently implemented on distinct architectures, it is also possible to come up with a *union ar-*

Digital Object Identifier 10.1109/MM.2017.265085944 0272-1732/$26.00 2017 IEEE

*chitecture* that can support multiple dataflows simultaneously. The questions are: (1) how to choose a combination of dataflows that maximally benefit the search for optimal mappings? (2) how to support these dataflows with the minimum amount of hardware implementation overhead? Answers to these questions will guide the direction of future DNN architecture designs.

This work has also pointed out how the concept of DNN dataflows and the mapping of a DNN computation onto a dataflow can be viewed as analogous to a general-purpose processor's architecture and compiling onto that architecture. We hope this will open up space for computer architects to approach the design of DNN accelerators by applying the knowledge and techniques from a well-established research field in a more systematic manner, such as methodologies for design abstraction, modularization, and performance evaluation.

For instance, a research trend for DNNs that is currently gaining a strong momentum is to exploit data statistics. Specifically, different proposals on quantization, pruning, and data representation have all shown promising results on improving the performance of DNNs. Therefore, it is of prime importance that new architectures can also take advantage of these findings. As compilers for general-purpose processors can take the profile of targeted workloads to further improve the performance of the generated binary, the mapper for DNN accelerators can also take the profile of targeted DNN statistics to further optimize the generated mappings. This is an endeavor we will leave for future work.

## References

[1] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE ISSCC*, 2014.

[2] A. Parashar, M. Pellauer, M. Adler, B. Ahsan, N. Crago, D. Lustig, V. Pavlov, A. Zhai, M. Gambhir, A. Jaleel, R. Allmon, R. Rayess, S. Maresh, and J. Emer, "Triggered Instructions: A Control Paradigm for Spatially-programmed Architectures," in *ISCA*, 2013.

[3] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *ISCA*, 2016.

[4] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE ISSCC*, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.

[6] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A Dynamically Configurable Coprocessor for Convolutional Neural Networks," in *ISCA*, 2010.

[7] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks," in *IEEE CVPRW*, 2014.

[8] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting Vision Processing Closer to the Sensor," in *ISCA*, 2015.

[9] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision," in *CoRR*, vol. abs/1502.02551, 2015.

[10] M. Peemen, A. A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for Convolutional Neural Networks," in *IEEE ICCD*, 2013.

[11] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning," in *ASPLOS*, 2014.

[12] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," in *FPGA*, 2015.

**Yu-Hsin Chen** is a PhD student in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. His research interests include energy-efficient multi-media systems, deep learning architectures and computer vision. Yu-Hsin received an MS in electrical engineering and computer science from the Massachusetts Institute of Technology. He is a student member of IEEE. Contact him at yhchen@mit.edu.

**Joel Emer** is a senior distinguished research scientist at Nvidia. He is also a professor of electrical engineering and computer science at the Massachusetts Institute of Technology. His research interests include spatial and parallel architectures, performance modeling, reliability analysis, and memory hierarchies. Emer received a PhD in electrical engineering from the University of Illinois. He is a Fellow of IEEE. Contact him at jsemer@mit.edu.

**Vivienne Sze** is an assistant professor in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. Her research interests include energy-aware signal processing algorithms, and low-power architecture and system design for multimedia applications such as machine learning, computer vision and video coding. Sze received a PhD in electrical engineering from Massachusetts Institute of Technology. She is a senior member of IEEE. Contact her at sze@mit.edu.

Digital Object Identifier 10.1109/MM.2017.265085944 0272-1732/$26.00 2017 IEEE

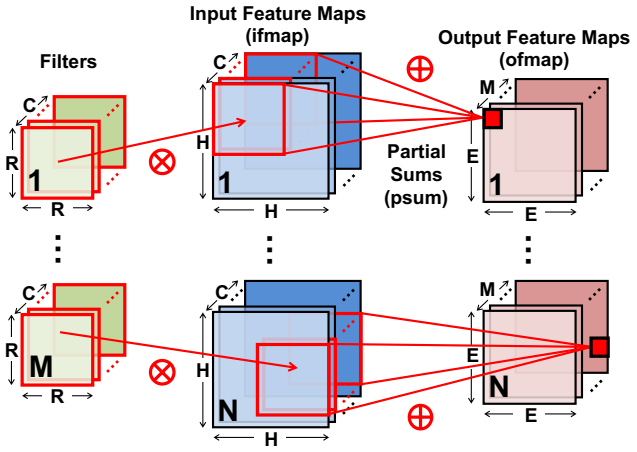



Fig. 1. In the processing of a DNN, multi-channel filters are convolved with the multi-channel input feature maps, which then generate the output feature maps. It consists of a large number of multiply-and-accumulate operations.

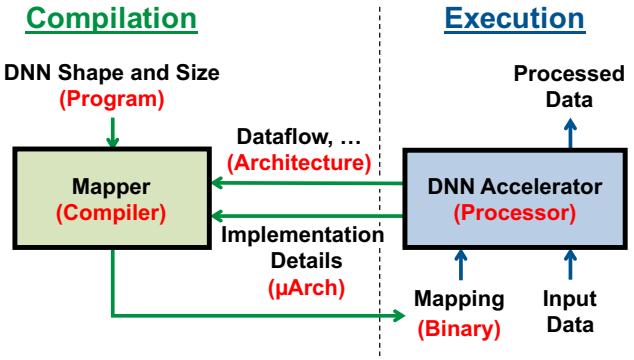



Fig. 2. An analogy between the operation of DNN accelerators (texts in black) and that of general-purpose processors (texts in red).

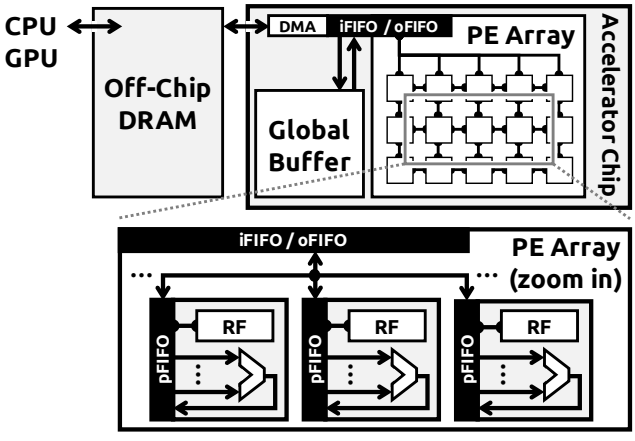



Fig. 3. Spatial Array Architecture. It consists of an array of processing elements (PE) and a multi-level storage hierarchy, including the off-chip DRAM, global buffer, network-on-chip (NoC), and register file (RF) in the PE. The off-chip DRAM, global buffer and PEs in the array can communicate with each other directly through the input/output FIFO (iFIFO/oFIFO). Within each PE, the PE FIFO (pFIFO) controls the traffic going in and out of the ALU, including from the RF or other storage levels.

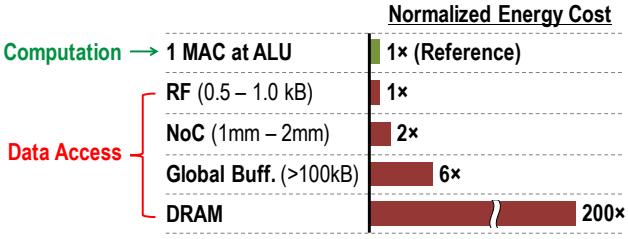



Fig. 4. Normalized energy cost relative to the computation of one MAC operation at ALU. Numbers are extracted from a commercial 65nm process.

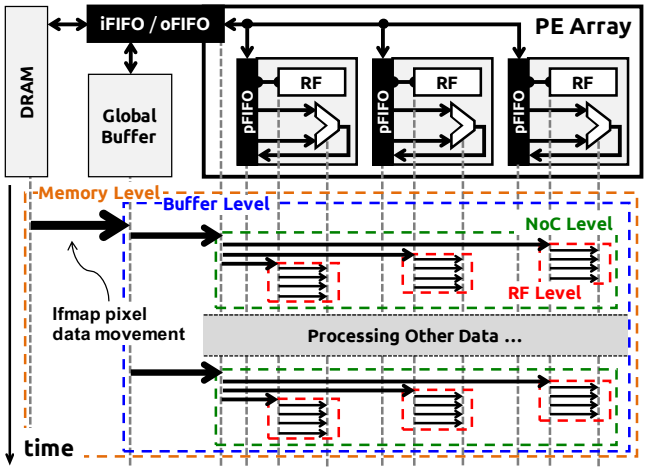



Fig. 5. Example of how a mapping determines the data reuse at each storage level. This example shows the data movement of one ifmap pixel going through the storage hierarchy. Each arrow means moving data between specific levels (or to ALU for computation).

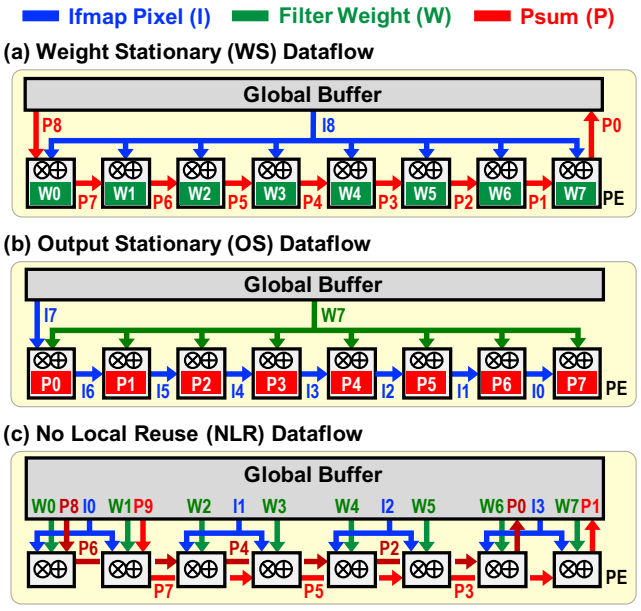



Fig. 6. Dataflow Taxonomy

Digital Object Identifier 10.1109/MM.2017.265085944 

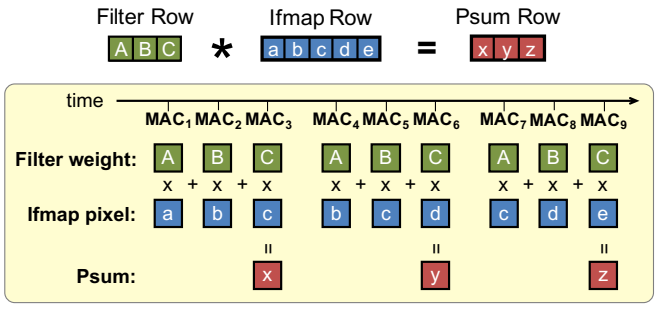



Fig. 7. Each row primitive in the RS dataflow runs a 1D row convolution on the same PE in a sliding window processing order.

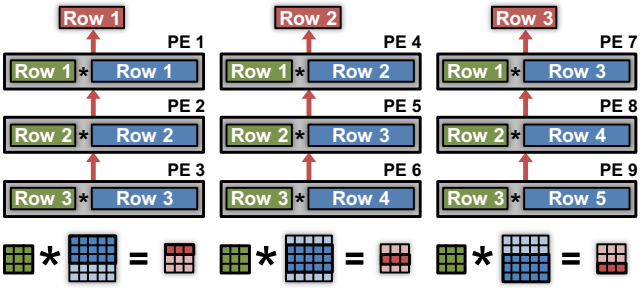



Fig. 8. Patterns of how row primitives from the same 2D plane are mapped onto the PE array in the RS dataflow.

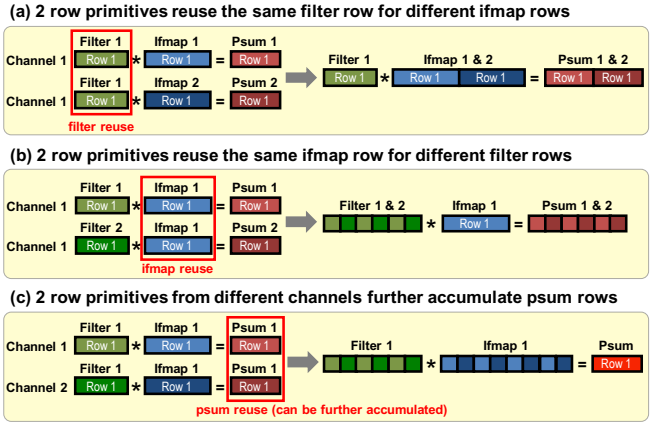
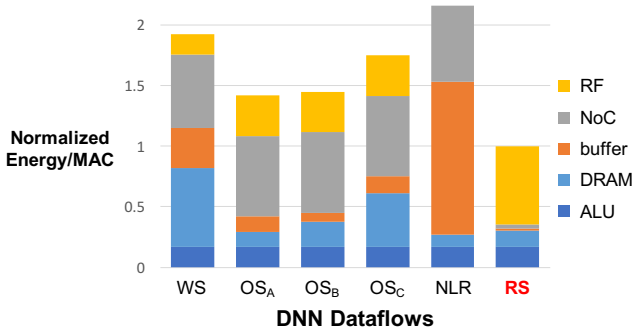
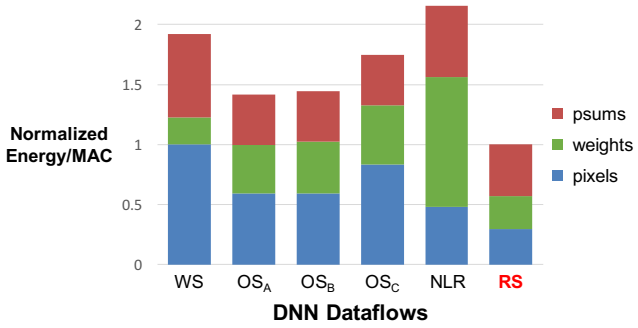



Fig. 9. Row primitives from different 2D planes can be combined by concatenating or interleaving their computation on the same PE to further exploit data reuse at the RF level.




(a)




(b)

Fig. 10. Comparison of energy efficiency between different dataflows in the CONV layers of AlexNet [5]: (a) breakdown in terms of storage levels and ALU, (b) breakdown in terms of data types. OSA, OSB and OSC are three variants of the OS dataflow that are commonly seen in different implementations [3].

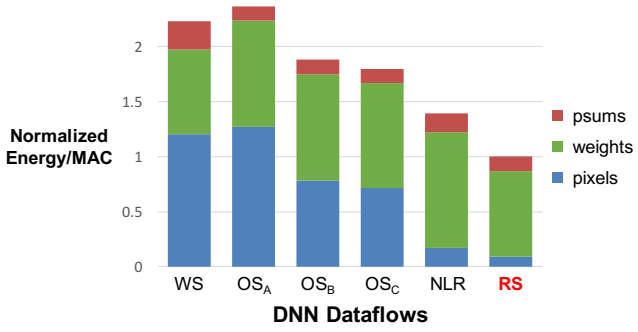



Fig. 11. Comparison of energy efficiency between different dataflows in the FC layers of AlexNet.

Digital Object Identifier 10.1109/MM.2017.265085944 