Reconfigurable Dataflow Optimization for Spatiotemporal Spiking Neural Computation on Systolic Array Accelerators

Jeong-Jun Lee
Electrical and Computer Engineering
University of California
Santa Barbara, CA, USA
jeong-jun@ucsb.edu

Peng Li
Electrical and Computer Engineering
University of California
Santa Barbara, CA, USA
lip@ucsb.edu

***Abstract*—Spiking neural networks (SNNs) offer a promising biologically-plausible computing model and lend themselves to ultra-low-power event-driven processing on neuromorphic processors. Compared with the conventional artificial neural networks, SNNs are well-suited for processing complex spatiotemporal data. Despite its significance, dataflow optimization of spiking neural accelerator architectures has not been extensively studied. Recognizing the need for efficient processing of complex spatiotemporal data while considering the all-or-none nature of spiking activities, we propose holistic reconfigurable dataflow optimization for systolic array acceleration of spiking convolutional networks (S-CNNs). A novel scheme is introduced for parallel acceleration of computation across multiple time points, which further allows for systemic optimization of variable tiling for a large performance and efficiency gains. We show how variable tiling, in particular, the positioning of the temporal dimension, can be targeted to optimize data movement, throughput, and energy efficiency. Furthermore, we explore joint layer-dependent dataflow and accelerator hardware optimization to further boost performance and energy efficiency. To support systemic design space exploration, we develop an SNN dataflow simulator capable of analyzing the throughput and energy dissipation of systolic array accelerators for any targeted S-CNN while considering the inherent spatiotemporal characteristics of spiking neural computation. The proposed techniques deliver orders of magnitude of improvements on throughput, energy efficiency, and delay-energy product for accelerating deep Alexnet and VGG-16 SNNs.**

***Index Terms*—spiking neural networks, neuromorphic computing, dataflow, systolic array, hardware accelerator**

I. Introduction

Spiking neural networks (SNNs) have emerged as the third generation of artificial neural network (ANN) models that are more biologically plausible than conventional ANNs. SNNs are particularly well-suited for processing complex spatiotemporal data based on a variety of rate and temporal codes and lend themselves to ultra-low-power event-driven processing on neuromorphic hardware [1, 2, 35-37].

Hardware acceleration of conventional ANNs, in particular deep neural networks (DNNs), offers a viable solution to deployment of models that are deeper and more powerful. Previous works have proposed dataflows to maximize throughput and energy efficiency of DNN accelerators [3-8], particularly for the widely adopted deep convolutional neural networks (CNNs) [9-11].

Clearly, there is also a strong demand for efficient dataflows and microarchitectures for SNN accelerators. Despite its significance, dataflow optimization of SNN accelerator architectures has not been extensively studied [13-15, 31]. The unique temporal aspect of SNN operations and the all-or-none nature of spiking activities introduce new challenges in dataflow optimization. One common approach is to perform SNN acceleration in a temporally sequential manner while following an optimized non-spiking ANN dataflow, ignoring the complex spatiotemporal tradeoffs [10,13].

Recognizing the need for developing SNN specific dataflows, we propose holistic reconfigurable dataflow optimization for systolic array acceleration of spiking convolutional networks (S-CNNs). As in many recent DNN accelerator works [17-19], we leverage systolic array architectures for efficient parallel processing with high spatiotemporal locality and compute density [25,26]. Our main contributions are:

- A novel scheme is introduced for parallel acceleration of computation across multiple time points, which further allows for systemic optimization of variable tiling for a large performance and efficiency gains.
- We show that parallel spatiotemporal computation over the six-dimensional space of input, weight, output, and temporal data involves significant complications in balancing between data reuse/storage of different data types and processing element utilization.
- We investigate how the tiling strategy, in particular the positioning of the temporal dimension, significantly impacts data movement, throughput, and energy efficiency. We propose the *first* method for mapping feedforward SNNs on a systolic array with optimized output and weight stationary dataflow.
- We explore joint layer-dependent dataflow and accelerator hardware optimization to further boost performance and energy efficiency.
- We develop an SNN dataflow simulator capable of analyzing the throughput and energy dissipation of systolic array accelerators to support systematic design space exploration while considering the inherent spatiotemporal characteristics of spiking neural computation.

The proposed techniques deliver orders of magnitude of improvements on throughput, energy efficiency, and delay-energy product for accelerating deep Alexnet and VGG-16 SNNs.

II. Challenges from Spiking Neural Computation

A. Data representations

In conventional ANNs, all data types, including input feature map (IFmap), filter, and output feature map (OFmap) data, are typically multi-bit. 8 to 16-bit precision is widely used in CNN hardware inference accelerators [3-8]. As shown in Fig. 1, the IFmaps and OFmaps of an SNN layer correspond to binary-valued input and output spikes, respectively. Binary input/output spikes are primarily of no concern because of low data volume. However, filter weights and intermediate results Psums are multi-bit. Furthermore, once computed, Psums may have to be stored over an extended period of time before being accumulated to produce binary output spikes, resulting in data movement challenges. The disparity in data representation shall be exploited in dataflow optimization towards mitigating the challenges brought by the high-volume filter and Psum data.

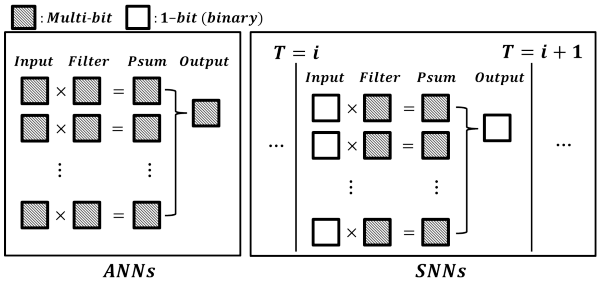

Fig. 1: Computation of a convolutional layer in ANNs vs. in SNNs. The latter involves temporal processing with binary-valued input and output spikes.

B. Temporal data processing

Fig. 1 compares the computation of convolutional layers in ANNs and SNNs. Apart from operating based on binary-valued input/output spikes, another key characteristic of SNNs is temporal processing. A spiking neuron integrates its inputs over time and generates a spike whenever the membrane potential surpasses a firing threshold (e.g., according to the widely adopted leaky integrate-and-fire model [16]). SNN hardware acceleration must consider the added temporal dimension, which causes complex spatiotemporal interactions in data movement/computation and also opens up the opportunity of parallel processing of computation at different time points as pursued in this work.

III. Temporal parallel processing in spiking convolutional layers

A. Proposed parallel processing in temporal dimension

The computation of a spiking neuron consists of: 1) spike input integration, 2) membrane potential accumulation, and 3) spike output generation, as shown in Fig. 2(a) [16, 32-34]. At each time point t, all presynaptic inputs to the given postsynaptic neuron are integrated and then added to the postsynaptic membrane potential at the previous time $t-1$ in order to compute the potential at the present time t. Finally, if the updated membrane potential exceeds a prescribed firing threshed, a binary output spike is generated, and the membrane potential is reset. The spike input integration step completely dominates the total computational cost due to the typically high degree of presynaptic connectivity. As in Fig. 2(a), our key observation is that the most expensive spike input integration step can be parallelized over multiple time steps for all spiking neurons in the layer under processing given the spike outputs from the preceding layer, which act as the inputs to the current layer. This is because spike input integrations at different time points are independent of each other. Upon the completion of the first step, membrane potential accumulation and spike output generation shall be performed in a temporally ascending order due to the temporal dependency introduced by the evolution of each membrane potential and reset. Our key idea is to explore the added parallelism brought by temporal parallel processing of spike input integration, which further allows inclusion of the temporal dimension as part of the variable tiling scheme. As shown later, systematic variable tiling optimization can be explored to effectively optimize or tradeoff between data movement, throughput, and energy dissipation in spiking based spatiotemporal processing.

B. Computation in spiking convolutional layers

Spiking convolutional (CONV) layers dominate the computational cost of a given S-CNN and are accelerated by the targeted systolic array accelerators in a layer-by-layer manner. The computation of a CONV layer can be expressed as:

$$\begin{aligned}&\mathbf{P}[t][n][m][x][y][i][j][c] = \\ &\mathbf{I}[t][n][c][Ux+i][Uy+j] \times \mathbf{W}[m][c][i][j] \\ &\mathbf{P}[t][n][m][x][y] = \sum_{c=0}^{C-1}\sum_{i=0}^{R-1}\sum_{j=0}^{R-1} \mathbf{P}[t][n][m][x][y][i][j][c]\end{aligned} \tag{1}$$

$$\mathbf{P}'[t][n][m][x][y] = \mathbf{P}[t][n][m][x][y] + \mathbf{P}_{Final}[t-1][n][m][x][y] \tag{2}$$

$$\begin{aligned}&\mathbf{O}[t][n][m][x][y] = f(\mathbf{P}'[t][n][m][x][y]) \\ &\mathbf{P}_{Final}[t][n][m][x][y] = \begin{cases} 0 & \text{if } \mathbf{O}[t][n][m][x][y] = 1 \\ \mathbf{P}'[t][n][m][x][y] & \text{else} \end{cases}\end{aligned} \tag{3}$$

$0 \le x,y < E,\ E=(H-R+U)/U,\ 0 \le c < C,\ 0 \le m < M,\ 0 \le t < T,\ 0 \le n < N$

where **O**, **I**, and **W** are the matrices of the OFmaps, IFmaps, and filters, respectively; U is a given stride size, and f is a spike generation function. (1), (2), and (3) correspond to each of the three steps discussed above. Table I lists the shape parameters of a layer and the number of time steps (T). Out of the eight dimensions of the shape parameters, five are independent, which, when combined with the temporal dimension T, give rise to 6-D spatiotemporal space. Since

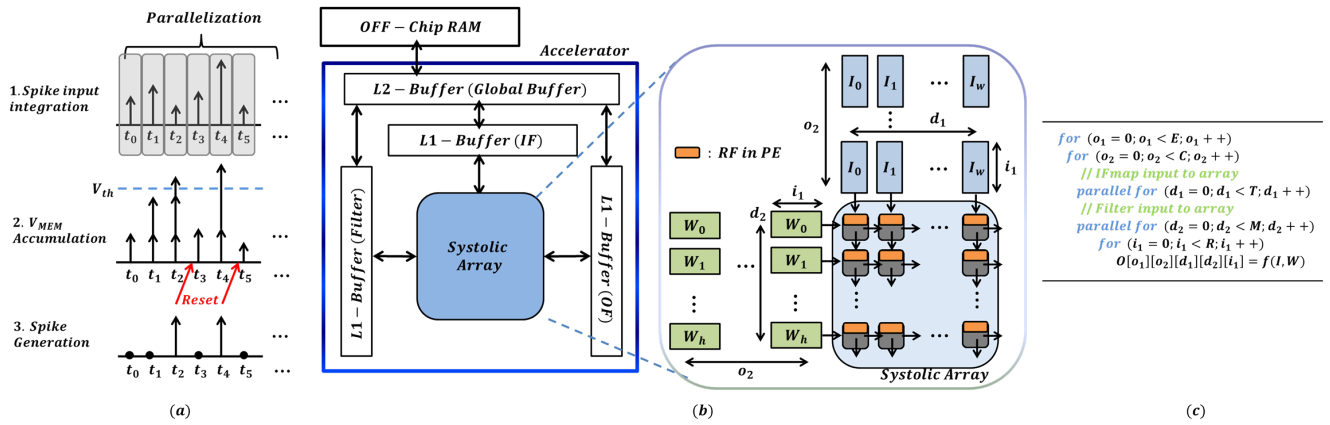

Fig. 2: Overview of the proposed work: (a) parallelization in temporal dimension, (b) systolic accelerator design, and (c) generalized loop representation of the mapped tiling strategy.

we use a 2-D systolic array as a computing substrate, the computation associated with two of these dimensions will be parallelized in a simultaneous row-wise and column-wise manner. We map a variable tiling scheme into the loop nest shown in Fig. 2 (c).

TABLE I: Shape parameters of a CONV layer in SNNs

Shape Parameter	*Description*
H / H	ifmap width / height
E / E	ofmap width / height
R / R	filter width / height
C	# of ifmap (= # of filter channels)
M	# of ofmap channels
T	# of time steps

IV. Dataflow Optimization for Spiking CNNs

Dataflows specify data scheduling via variable tiling and directly impact the overall runtime and energy dissipation of a DNN accelerator [3-7, 11, 12]. However, dataflows for SNNs have not been extensively studied. Mapping the computations of a spiking conv layer involves directly associating two parameters (dimensions) with the 2-D systolic array to naturally exploit spatial-locality and data reuse. These two parameters are represented as d1 and d2 in Fig. 2. Ultimately, an optimized tiling strategy over the six-dimensional space of input, weight, output, and temporal data shall be adopted to maximize data reuse, energy efficiency, and throughput.

One common dataflow approach is to perform SNN acceleration in a temporally sequential manner while following an optimized non-spiking ANN dataflow [10,13]. Clearly, this approach is unable to parallelize computation along the temporal dimension and ignores the complications and opportunities in dealing with complex spatiotemporal processing in SNNs. We propose in-depth dataflow optimizations considering data stationarity, variable tiling, and layer/network dependencies for spiking CNNs (S-CNNs).

A. Stationary schemes for S-CNNs

Three stationary dataflows, namely, input stationary (IS), weight stationary (WS), and output stationary (OS), have been studied for non-spiking CNN accelerators [3-8]. In each stationary scheme, one type of the data (i.e., IFmap, filter weight, or Psum data) stays stationary in each processing element (PE) of the array to minimize the movement of the corresponding data. For SNNs, our key observation is that the disparity among different data types must be considered, particularly with respect to the low volumes of input/output spike data for each spiking neuron. The IFmap data (input spikes) are binary and hence require low data bandwidth and are not the main bottleneck in data movement. Similarly, since the final activation of each spiking neuron is binary, the output spike data are not the bottleneck of data movement either. As such, retaining the low-volume IFmap data in the systolic array by choosing the input-stationary dataflow is not effective. The OS and WS stationary flows can minimize the data movement of high-volume multi-bit Psum and filter weight data and shall be specifically focused on for SNN accelerators.

B. Variable Tiling

As discussed in Section II-B, the proposed parallelization across multiple time points allows for systematic dataflow optimization via a proper choice of variable tiling. Nevertheless, optimization of the variable tiling is a multi-faceted problem and shall be based upon a careful balancing between data reuse/movement, throughput, and energy efficiency involved in processing spatiotemporal data. Under a given hardware resource constraint and resource allocation between the memory and compute units, a systolic array accelerator may be either compute-bound or memory-bound, which is further dependent on the variable tiling in conjunction with the characteristics of the CNN layer to be processed.

While the dataflow simulator proposed in Section V supports the evaluation of variable tiling in both the compute-bound and memory-bound regimes, we discuss the impacts of different tiling strategies under memory-bound operations with respect to the high-volume multi-bit filter and Psum data, the two main bottlenecks in data movement. In particular, the positioning of temporal variable t has significant impacts on the tradeoffs between runtime (throughput) and energy dissipation.

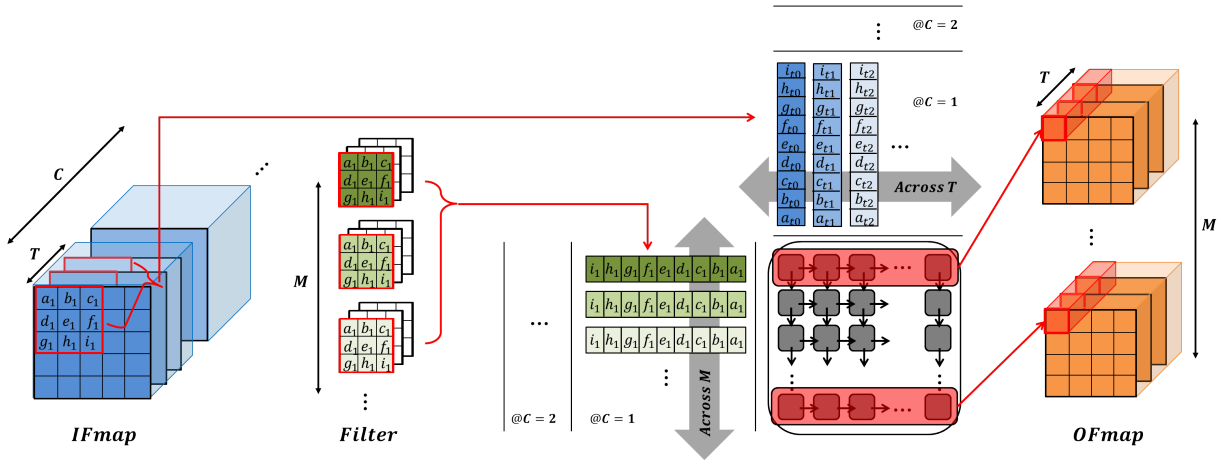

Fig. 3: Mapping of a *Psum-friendly* output-stationary dataflow onto a systolic array accelerator.

1) Memory access/energy efficiency: The energy dissipated in data movement can be orders of magnitude higher than that of the corresponding MAC operation specifically when moving data across the chip boundary [3-10]. In SNNs, Psums are the most important contributor to data movement.

As the time parameter t goes deeper into the loop nest, it changes more frequently. In other words, there would be more incomplete computations performed over a larger number of time points, producing a higher volume of Psums to be stored. Although the final OFmap data (i.e., spike outputs) is binary, the Psums are multi-bit. On the other hand, placing parameter t in an outer-loop position helps reduce the amount of Psums. This is because, under this case, the computation of spike output of each processed spiking neuron spans over a shortened period of time (clock cycles), leading to faster conversion (summation) of multi-bit Psums to binary spike outputs. This helps reduce data movement and hence improves energy efficiency. We refer to dataflows in which t is placed at an outer-loop position as a *Psum-friendly dataflow*.

2) Throughput: Due to the overlap between computation and memory access, the throughput (runtime) of a systolic array accelerator depends on the more dominant delay component of the two [21,22]. In the memory-bound regime, the filter weight access time dominates the overall runtime of an accelerator.

In output stationary (OS) dataflows, placing parameter t in an inner-loop position of the variable tiling opens up the possibility for parameters related to the filter data to be placed in outer-loop positions, leading to less frequent access to multi-bit filter weight data and more filter data reuse. Ultimately, doing so leads to a greater throughput or speedup of the accelerator. We refer to such dataflows as *filter-friendly* dataflows. Conversely, placing parameter t in an outer-loop position forces the parameters related to the filter data to be placed in inner-loop positions. This may degrade runtime as it requires more frequent filter data transfers, hindering the data loading before starting the systolic array computation. Alternatively, throughput can be improved by adopting a weight stationary (WS) dataflow as it immediately maximizes the filter data reuse. However, this is typically at the cost of higher energy dissipation due to the reduced Psum data reuse.

The above observations suggest that a careful trade-off between throughput and data reuse (i.e., energy dissipation) must be made while optimizing the dataflow for which the positioning the time parameter t in variable tiling is one of the key considerations. One of the proposed tiling strategy, which enables the parallel processing of the temporal dimension, is shown in Fig. 3, where data in a spatial 2D space such as filter weight data associated with the R parameter are vectorized. The dataflow in Fig. 3 is an example of Psum-friendly dataflow since all Psums generated in the array are for one specific output position in the OFmap. At the same time, data processing associated with parameters (T, M) is parallelized in a simultaneous row-wise and column-wise manner. In general, this dataflow is expected to have advantages in terms of energy efficiency but results in greater runtime due to frequent filter data transfers.

C. Layer-dependent dataflow reconfiguration

In a deep S-CNN, early conv layers tend to have larger lateral IFmap and OFmap dimensions and a fewer number of channels compared with late layers. For example in VGG-16, H=224/R=3/C=3/M=64 for the first conv layer CONV1, and H=14/R=3/C=512/M=512 for the 11-th conv layer CONV11. If the binary spike output for each spiking neuron cannot be computed quickly, all its Psums must be stored over an extended period of time, producing more Psum data movement. To prevent this from happening, one may place the IFmap channel dimension C and filter spatial dimension R at the inner-most loop positions. However, this may not be the optimal strategy since it can lead to worsened filter data reuse and PE utilization. This Psum data movement problem gets more pronounced for large OFmaps for which there is a tendency for storing Psum data for a larger number of spike outputs.

As a result, early layers are deemed more Psum intensive, pushing the tiling optimization more towards mitigating the impact of Psum data movement. Conversely, the processing

of later conv layers with more filters is more severely bottlenecked by the need to access to a larger amount of filter data. Clearly, adopting a fixed variable tiling across different layers or SNNs will lead to non-optimal results. Reconfiguring variable tiling during runtime in a layer-dependent manner may further improve the overall accelerator performance. Comprehensive dataflow optimization considering all these intertwined factors will be supported by the proposed SNN dataflow simulator described next.

V. SNN Dataflow Simulator

To support dataflow optimization over a large design space while considering complex tradeoffs in the six-dimensional space of input, output, weight, and temporal data, we introduce an analytic simulator for SNN dataflows. As shown in Fig. 4, the simulator takes user-specified hardware configuration, including technology node and area utilization for compute units and memory, variable tiling, and targeted S-CNN as inputs. It produces a detailed analysis of runtime (throughput), memory access/data movement, and energy dissipation of the systolic-array accelerator.

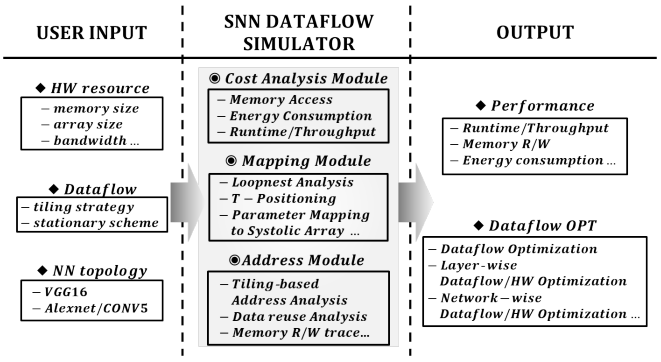

Fig. 4: An overview of SNN dataflow simulator framework.

A. Modeling of systolic array and memory

1) Array configurations: The most commonly used square-shaped systolic arrays are considered. Each systolic array comprises a set of processing elements (PEs), each containing an accumulate (AC) unit and a scratchpad memory (a register file). Note that since the spike inputs are binary, the more complex MAC units are not needed.

2) Memory configurations: The memory hierarchy design is critical to the overall performance and energy consumption of spiking neural computation due to the SNN's memory-intensive nature. As a standard practice, three levels of the memory hierarchy is assumed, as shown in Fig. 2: an external memory (DRAM), on-chip L2/L1 buffers, and a small scratchpad memory in each PE. We model the L2 and L1 buffers as double buffers, allowing simultaneous reads/writes to hide access latency.

3) Compute and memory tradeoffs: Under a total accelerator chip area constraint, the area of the compute units vs. that of the memory can be varied. For example, the array size may be set to 8x8, 16x16, 24x24, 32x32, 40x40, or 48x48, which leaves the bulk of the remaining chip area to memory.

B. Performance/overhead analysis

With a given tiling strategy, the simulator generates the corresponding addresses for IFmap/filter/OFmap data access. The simulator estimates runtime and memory access by generating cycle-accurate read/write traces for each level of memory. We follow the methods presented in [21,22] for runtime/energy estimation.

1) Runtime/throughput: During each systolic array processing iteration, the worst-case delay between computation/data access determines the per iteration runtime. The worst-case delay is accumulated over all iterations in the systolic array to estimate the total runtime of the accelerator. The runtime of one array iteration is modeled by the array computation cycle number which is equal to the sum of the array height/width and length of the input spike train since the data is fed from the left and right edges of the array and propagate via uni-directional links to the right/bottom end of the array. We model the data access delay based on the bandwidth and read/write traces at each level of memory.

2) Memory access and energy dissipation: Energy dissipation comes primarily from two sources: 1) compute energy and 2) memory access energy. Compute energy is evaluated by multiplying the number of AC operations with energy per AC operation. Memory access energy dominates in the total energy dissipation. If the data needed for array computation is not present in the local scratchpad memories, it will be fetched from the higher-level memories. The simulator traces the number of the read/write accesses to each memory. The dominant energy dissipation due to memory access is evaluated based on the access counts and the CACTI model [20].

TABLE II: Layer-specific comparison of tiling strategies in output stationary (OS) dataflows for the VGG-16 net.

	Normalized Performance for VGG16 CONV1 & CONV11 layers						
	Tiling strategy	Runtime/Energy		EDP[c]		EDP I[d]	
		C1[a]	*C11*[b]	*C1*	*C11*	*C1*	*C11*
ref**	T/M/E/C/R	367/32.4	10K/20K	119	20k	0.84X	0.005X
ref*	T/R/E/M/C	100/100	100/100	100	100	1X	1X
	E/R/T/M/C	37.6/0.48	62.4/64.6	0.18	40.3	555X	2.5X
	M/T/E/C/R	359/32.4	10K/10K	117	1M	0.85X	0.0001X
	E/T/C/M/R	215/7.19	50K/50K	15.5	20M	0.06X	0.000005X
	C/M/E/T/R	2K/4K	931/10K	90K	100K	0.001X	0.001X
	M/E/T/C/R	224/1.20	10K/10K	2.69	2M	37X	0.00005X
	T/M/E/R/C	178/14.1	139/166	25.2	229	3.97X	0.43X
① E-T	E/C/T/M/R	30.9/2.88	18.6/138	0.89	25.6	112X	3.9X
② B-T	T/C/E/M/R	22.9/24	14.4/127	5.50	18.2	18X	6X
③ R-T	C/T/E/M/R	22.5/35.7	13.8/106	8.03	14.7	13X	7X

[a]: CONV1. ref*: Existing SNN dataflow [30].
[b]: CONV11. ref**: Non-optimized SNN dataflow.
[c]: Energy-Delay Product. [d]: EDP improvement.

VI. Results

By leveraging the proposed SNN simulator, we demonstrate dataflow optimization under various systolic array configurations and choices of stationary schemes for specific spiking convolutional layers or across one entire spiking CNN network. We focus on the accelerating CONV layers as they account for over 90% of total computation [3, 21-22]. For

demonstration purposes, a 28nm technology node and two total area constraints of 8mm^2 and 16mm^2 are assumed [3, 21]. We evaluate the performance and efficiency of the systolic array accelerators for accelerating the inference of the pre-trained SNN implementations of the Alexnet and VGG-16 networks, two deep learning architectures widely adopted for image classification. These spiking CNNs have demonstrated promising classification accuracy [28,29]. The two spiking network models operate over 200-time steps with time-wise packing of the binary spike inputs/outputs.

A. Layer-specific dataflow optimization

We perform comprehensive dataflow optimization for a specific spiking conv layer by comparing a large number of variable tiling strategies based on output stationary (OS) dataflows. The targeted layers are CONV1 and CONV11 from the VGG-16 net, a representative early and late conv layer, respectively. Dataflow optimization has not been targeted in most of the existing SNN works [13] with a few adopting a non-spiking CNN dataflow in which the computations at different time points are performed sequentially and the scheduling of the computation at each time step follows the chosen dataflow [13, 30]. One of the such SNN dataflows from [30] is denoted by ref* and chosen as a baseline reference.

Table II reports a comprehensive comparison of the overall accelerator performance across many tiling strategies. The accelerator performance level varies over several orders of magnitude, indicating the key importance of dataflow optimization. For comparison purposes, one non-optimized dataflow denoted by ref** in Table II is chosen as another reference. It is evident that non-optimal dataflows can result in significantly degraded overall performance.

We identify three near-optimal variable tiling schemes, denoted by ①, ②, and ③ in Table II. We call dataflows ①, ②, and ③ energy-targeted (E-T), energy-runtime balance-targeted (B-T), and runtime-targeted (R-T), respectively. They favor the optimization of energy or runtime or strike a good balance between the two. As presented in Fig. 3, the advantages of the E-T dataflow result from its Psum-friendly nature in that the Psums for each binary spike output are accumulated over a shortened time span, reducing the energy cost of data movement. On the other hand, R-T dataflow shows the advantages of filter reuse in that the parameters related to filters are placed in outer-loop positions, leading to greater throughput.

B. Joint optimization of tiling and stationary flows

It is meaningful to investigate how variable tiling and stationarity of the dataflow can be jointly optimized. Fig. 5(a) shows the throughputs of the E-T, B-T, and R-T variable tiling when paired with the weight stationary (WS) and output (OS) schemes for the VGG-16 CONV1 and CONV11 layers. It can be seen that WS results in an improved throughput compared with OS due to the maximized reuse of the filter data for both layers, as discussed in Section IV-B2. Note that while R-T

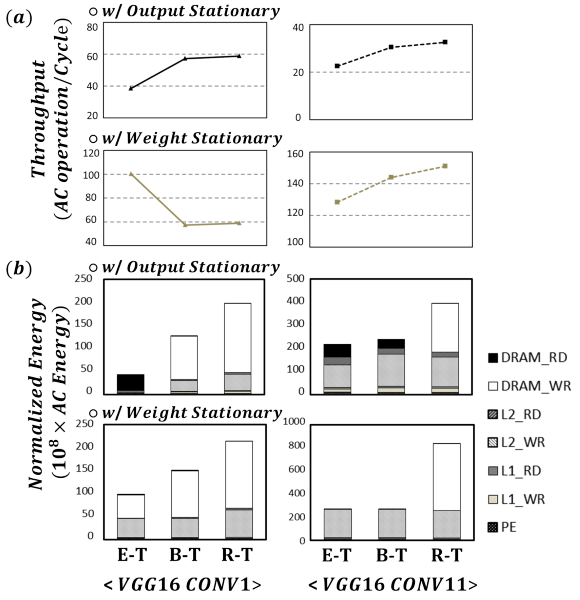

Fig. 5: (a) The throughput of three dataflows for VGG-16 CONV1 and CONV11 with OS and WS. (b) Energy dissipation of various dataflows for VGG-16 CONV1 and CONV11. AC refers to accumulation operation.

when paired with OS provides better throughput than E-T as presented previously, with WS, E-T happens to offer higher throughput than R-T for CONV1. This suggests that for early layers, WS may exert a greater impact on throughput than variable tiling.

Fig. 5(b) reports the breakdown of energy consumption for the three tiling schemes when paired with OS or WS. As discussed before, the Psums data is the most important factor in energy efficiency, which is better handled by OS. As a result, OS improves energy efficiency over WS. Among all the considered tiling and stationarity combinations, paring OS with E-T leads to the best energy efficiency.

C. Joint layer-dependent reconfigurable dataflow and hardware optimization

We demonstrate the additional benefits brought by the proposed layer-dependent configuration of variable tiling during runtime based on OS dataflows. As discussed previously, the three tiling schemes E-T, B-T, and R-T well represent distinct trade-offs that can be made between throughput and energy dissipation. We limit the tiling choice of each layer to these three schemes while our simulator can support wide ranges of design space exploration at a higher computational cost. Furthermore, we explore accelerator hardware optimization by allocating different area budgets for the compute units and on-chip memory under a fixed total chip area constraint. This is done by evaluating the overall performance by changing the systolic array size, which correspondingly alters the amount of on-chip memory under the constant total chip area. Note that optimized accelerator hardware is not runtime reconfigurable except for the variable tiling, which is reconfigured on a layer-by-layer basis.

TABLE III: Joint layer-dependent dataflow and accelerator optimization under different optimization-targets for VGG-16.

VGG-16	CONV Layers (Area = 8mm^2)													Optimized
Target	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	Array Size
Runtime	③	③	③	②	②	②	②	③	③	③	③	③	③	24X24
Energy	①	①	①	①	①	①	①	①	①	①	①	①	①	32X32
EDP	①	①	①	①	①	①	①	①	①	①	②	②	②	32X32

VGG-16	CONV Layers (Area = 16mm^2)													Optimized
Target	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	Array Size
Runtime	③	③	③	③	②	②	②	③	③	③	③	③	③	32X32
Energy	①	①	①	①	①	①	①	①	①	①	①	①	①	40X40
EDP	①	①	①	①	①	①	①	①	①	①	①	①	①	40X40

*①, ②, and ③ represents the E-T (Energy-Targeted) / B-T (energy-runtime Balance-Targeted) / R-T (Runtime-Targeted) dataflow respectively.

TABLE IV: Joint layer-dependent dataflow and accelerator optimization under different optimization-targets for Alexnet.

Alexnet	CONV Layers (Area = 8mm^2)					Optimized
Target	L1	L2	L3	L4	L5	Array Size
Runtime	①	②	③	③	③	24X24
Energy	②	①	①	①	②	32X32
EDP	②	③	③	③	③	16X16

Alexnet	CONV Layers (Area = 16mm^2)					Optimized
Target	L1	L2	L3	L4	L5	Array Size
Runtime	②	②	③	③	③	32X32
Energy	②	①	②	②	③	40X40
EDP	②	③	②	①	②	32X32

Table III and Table IV summarize the results of joint reconfigurable dataflow and hardware optimization based on two total chip area constraints for VGG-16 and Alexnet, respectively. The optimal variable tiling for all layers in the network and the optimized accelerator in terms of its array size are reported. The results are based on maximizing one of the three performance targets across all layers of the network: throughput ($1/runtime$), energy, and energy-delay product (EDP). In general, targeting maximizing total energy efficiency predominantly makes the optimal tiling for all layers to be E-T, with some being B-T, which is well expected. Similarly, targeting minimizing overall runtime makes the optimal tiling R-T for most layers. The trend on the optimization of EDP target is less visible, potentially due to the fact that both the variable tiling and accelerator hardware are simultaneously optimized.

Table V details the overall runtime, energy dissipation, and EDP resulted from the optimizations performed in Tables III and IV. We also report the results of using each of the two reference tiling schemes ref* [30] and ref** for all layers. For completeness of comparison, the results based on a fixed default array size (16X16) are reported to demonstrate the impact of hardware optimization. It is evident that the overall performance improves with the chip area. The proposed joint optimization improves EDP by up to 16.7X and by up to $282,000$X when compared with the variable tiling ref* and ref** without hardware optimization, respectively.

VII. Conclusion

This work is motivated by a dearth of a knowledge base for developing efficient SNN dataflows despite its crucial impact on SNN accelerator design. A novel scheme is introduced to enable the parallel acceleration of computation across multiple time points, which further leads to large performance and efficiency gains via optimization of the tiling strategy. We demonstrate how the tiling strategy can be reconfigured on a layer-by-layer basis and jointly optimized with the accelerator hardware to achieve large gains in throughput and energy efficiency. Furthermore, an SNN dataflow simulator has been developed to aid systemic design space exploration. The proposed techniques are able to improve EDP of the accelerator by several orders of magnitude and more than one order of magnitude over a non-optimized and existing SNN dataflow, respectively.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grants No. 1948201 and No. 1940761.

References

[1] Davies, Mike, et al. "Loihi: A neuromorphic manycore processor with on-chip learning." IEEE Micro 38.1 (2018): 82-99.

[2] Akopyan, Filipp, et al. "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip." IEEE transactions on computer-aided design of integrated circuits and systems 34.10 (2015): 1537-1557.

[3] Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." IEEE journal of solid-state circuits 52.1 (2016): 127-138.

[4] Peemen, Maurice, et al. "Memory-centric accelerator design for convolutional neural networks." 2013 IEEE 31st International Conference on Computer Design (ICCD). IEEE, 2013.

[5] 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.

[6] 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.

[7] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "DaDianNao: A Machine-Learning Supercomputer," in MICRO, 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] Yao, Peng, et al. "Fully hardware-implemented memristor convolutional neural network." Nature 577.7792 (2020): 641-646.

[10] Cao, Yongqiang, Yang Chen, and Deepak Khosla. "Spiking deep convolutional neural networks for energy-efficient object recognition." International Journal of Computer Vision 113.1 (2015): 54-66.

TABLE V: Normalized runtime, energy dissipation, and EDP under different optimization targets and hardware area constraints for VGG-16 and Alexnet. The SNN dataflow from [30] is denoted by ref*.

VGG16	**AREA = 8mm^2**					**AREA = 16mm^2**				
Optimization target	**Hardware optimization**	***Result***			**EDP Improvement**	**Hardware optimization**	***Result***			**EDP Improvement**
		Runtime	Energy	EDP			Runtime	Energy	EDP	
[a]None (ref**)	No (16X16)	4079	3703	2x10^5	0.0005X	No (16X16)	4024	3676	2x10^5	0.0003X
[b]None (ref*)	No (16X16)	100	100	100	1X	No (16X16)	89.16	76.91	64.72	1X
Runtime	No (16X16)	11.76	178.64	20.93	4.78X	No (16X16)	11.04	139.98	16.49	3.92X
Energy	No (16X16)	16.84	63.27	10.25	9.76X	No (16X16)	15.89	59.47	9.376	6.98X
Runtime	Yes (24X24)	9.26	174.62	16.33	6.12X	Yes (32X32)	6.04	152.28	8.843	7.32X
Energy	Yes (32X32)	21.27	39.09	7.692	13.0X	Yes (40X40)	16.72	23.84	3.874	16.7X
EDP	Yes (32X32)	20.93	39.81	7.667	13.1X	Yes (40X40)	16.72	23.84	3.874	16.7X

Alexnet	**AREA = 8mm^2**					**AREA = 16mm^2**				
Optimization target	**Hardware optimization**	***Result***			**EDP Improvement**	**Hardware optimization**	***Result***			**EDP Improvement**
		Runtime	Energy	EDP			Runtime	Energy	EDP	
None (ref**)	No (16X16)	4839	36044	3x10^6	0.00003X	No (16X16)	4736	9600	7x10^5	0.0001X
None (ref*)	No (16X16)	100	100	100	1X	No (16X16)	87.20	93.38	75.17	1X
Runtime	No (16X16)	19.87	67.57	11.91	8.40X	No (16X16)	15.76	65.59	9.521	7.90X
Energy	No (16X16)	20.29	66.98	11.84	8.45X	No (16X16)	15.97	64.96	9.434	7.97X
Runtime	Yes (24X24)	19.85	69.88	13.05	7.66X	Yes (32X32)	13.73	65.30	8.137	9.24X
Energy	Yes (32X32)	28.44	55.95	18.70	5.35X	Yes (40X40)	19.65	44.30	10.01	7.51X
EDP	Yes (16X16)	20.29	66.98	11.84	8.45X	Yes (32X32)	14.50	52.03	7.009	10.7X

[a] Without dataflow optimization, using single tiling strategy (ref**) for all layers.
[b] Without dataflow optimization, using single tiling strategy (ref*) for all layers.

[11] Gysel, Philipp, Mohammad Motamedi, and Soheil Ghiasi. "Hardware-oriented approximation of convolutional neural networks." arXiv preprint arXiv:1604.03168 (2016).

[12] Aimar, Alessandro, et al. "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps." IEEE transactions on neural networks and learning systems 30.3 (2018): 644-656.

[13] Wang, Shu-Quan, et al. "SIES: A Novel Implementation of Spiking Convolutional Neural Network Inference Engine on Field-Programmable Gate Array." Journal of Computer Science and Technology 35 (2020): 475-489.

[14] Afifi, A., A. Ayatollahi, and F. Raissi. "Implementation of biologically plausible spiking neural network models on the memristor crossbar-based CMOS/nano circuits." 2009 European Conference on Circuit Theory and Design. IEEE, 2009.

[15] Chen, Ling, et al. "Memristor crossbar-based unsupervised image learning." Neural Computing and Applications 25.2 (2014): 393-400.

[16] Gerstner, Wulfram, and Werner M. Kistler. Spiking neuron models: Single neurons, populations, plasticity. Cambridge university press, 2002.

[17] Wei, Xuechao, et al. "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs." Proceedings of the 54th Annual Design Automation Conference 2017. 2017.

[18] Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. 2019.

[19] Kung, Hsiang-Tsung, Bradley McDanel, and Sai Qian Zhang. "Mapping systolic arrays onto 3d circuit structures: Accelerating convolutional neural network inference." 2018 IEEE International Workshop on Signal Processing Systems (SiPS). IEEE, 2018.

[20] Muralimanohar, Naveen, Rajeev Balasubramonian, and Norman P. Jouppi. "CACTI 6.0: A tool to model large caches." HP laboratories 27 (2009): 28.

[21] Kwon, Hyoukjun, Michael Pellauer, and Tushar Krishna. "MAESTRO: an open-source infrastructure for modeling dataflows within deep learning accelerators." arXiv preprint arXiv:1805.02566 (2018).

[22] Samajdar, Ananda, et al. "Scale-sim: Systolic cnn accelerator simulator." arXiv preprint arXiv:1811.02883 (2018).

[23] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[24] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[25] Kung, H. T., and Charles E. Leiserson. "Systolic arrays (for VLSI)." Sparse Matrix Proceedings 1978. Vol. 1. Society for industrial and applied mathematics, 1979.

[26] Kung, Hsiang-Tsung. "Why systolic architectures?." Computer 1 (1982): 37-46.

[27] Venkatesan, Rangharajan, et al. "Magnet: A modular accelerator generator for neural networks." Proceedings of the International Conference on Computer-Aided Design (ICCAD). 2019.

[28] Sengupta, Abhronil, et al. "Going deeper in spiking neural networks: Vgg and residual architectures." Frontiers in neuroscience 13 (2019).

[29] Zhang, Lei, et al. "Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019.

[30] Tan, Pai-Yu, et al. "A Power-Efficient Binary-Weight Spiking Neural Network Architecture for Real-Time Object Classification." arXiv preprint arXiv:2003.06310 (2020).

[31] Narayanan, Surya, et al. "SpinalFlow: An Architecture and Dataflow Tailored for Spiking Neural Networks." 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020.

[32] Liu, Yu, Yingyezhe Jin, and Peng Li. "Online adaptation and energy minimization for hardware recurrent spiking neural networks." ACM Journal on Emerging Technologies in Computing Systems (JETC) 14.1 (2018): 1-21.

[33] Liu, Yu, Yingyezhe Jin, and Peng Li. "Exploring sparsity of firing activities and clock gating for energy-efficient recurrent spiking neural processors." 2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, 2017.

[34] Wang, Qian, Youjie Li, and Peng Li. "Liquid state machine based pattern recognition on FPGA with firing-activity dependent power gating and approximate computing." 2016 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2016.

[35] Jin, Yingyezhe, Yu Liu, and Peng Li. "SSO-LSM: A sparse and self-organizing architecture for liquid state machine based neural processors." 2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH). IEEE, 2016.

[36] Xu, Changqing, et al. "Boosting Throughput and Efficiency of Hardware Spiking Neural Accelerators Using Time Compression Supporting Multiple Spike Codes." Frontiers in Neuroscience 14 (2020): 104.

[37] Ku, Bon Woong, et al. "Design and architectural co-optimization of monolithic 3D liquid state machine-based neuromorphic processor." 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). IEEE, 2018.