SNAP: A 1.67 – 21.55TOPS/W Sparse Neural Acceleration Processor for Unstructured Sparse Deep Neural Network Inference in 16nm CMOS

Jie-Fang Zhang [1], Ching-En Lee [1], Chester Liu [1], Yakun Sophia Shao [2], Stephen W. Keckler [2], Zhengya Zhang [1]
[1] University of Michigan, Ann Arbor, MI, USA; [2] NVIDIA, Santa Clara, CA, USA

Abstract

A Sparse Neural Acceleration Processor (SNAP) is designed to exploit unstructured sparsity in deep neural networks (DNNs). SNAP uses parallel associative search to discover input pairs to maintain an average 75% hardware utilization. SNAP's two-level partial sum reduce eliminates access contention and cuts the writeback traffic by 22×. Through diagonal and row configurations of PE arrays, SNAP supports any CONV and FC layers. A 2.4mm^2 16nm SNAP test chip is measured to achieve a peak effectual efficiency of 21.55TOPS/W (16b) at 0.55V and 260MHz for CONV layers with 10% weight and activation density. Operating on pruned ResNet-50, SNAP achieves 90.98fps at 0.80V and 480MHz, dissipating 348mW.

Introduction

Modern DNNs [1] contain millions of weights and require billions of operations per input frame. By removing weights (Ws) using pruning [2] and zeroing out output activations (OAs) (i.e., input activations (IAs) of the next layer) using rectified linear units (ReLU), a sparse DNN model can be produced (Fig. 1). A sparse DNN accelerator stores Ws and IAs in a compressed form consisting of a data array (zeros removed) and an index array. Compressed W and IA arrays are paired by matching indices, dispatched to a SIMD multiplier (MULT) array, and the resulting partial sums (psums) are accumulated to their respective OAs (Fig. 2). Data sparsity leads to better efficiency, but major challenges remain: 1) MULT underutilization due to an insufficient number of W-IA inputs that can be extracted and dispatched (frontend challenge); 2) massive data traffic and access contention to support accumulation of psums whose destination addresses are seemingly random (backend challenge); and 3) limited support for different kernel sizes and layer types (flexibility challenge).

The latest sparse DNN accelerators are unable to solve all three challenges. SCNN [3] supports only CONV layers. It maximizes MULT utilization at the cost of massive writeback traffic and access contention. STICKER [4] uses 2-way set-associative PEs but is unable to resolve access contention completely. Stitch-X [5] proposes a parallelism discovery unit (PDU) to improve utilization, but it does not solve the backend or the flexibility challenges. We present SNAP to address all three challenges. SNAP uses associative index matching (AIM) to extract a sufficient number of W-IA pairs to maintain an average 75% hardware utilization. SNAP accumulates (i.e., reduces) psums in two levels to the maximum extent before writeback: channel-dimension reduce (C-reduce) followed by pixel-dimension reduce (P-reduce). The two-level reduce eliminates access contention and cuts writeback traffic by over one order of magnitude. With PE array configurations, SNAP supports various kernel sizes and layer types.

SNAP Architecture

The SNAP architecture consists of control, memory and multi-core compute modules (Fig. 3). The memory module is composed of multi-banked IA, OA buffers shared between cores, and W buffers private to each core. Inputs are aligned after fetch, and outputs are compressed before writeback. The 16nm SNAP test chip is built using 4 cores, with 7×3 PEs per core and 3 MULTs per PE for a total of 252 MULTs (16b) together with 280.6 KB SRAM buffer. AIM units are used as the frontend of the PE array to extract W-IA inputs by searching index arrays. In the backend, C-reduce is done at the PE level, and P-reduce is done at the core level by the core reducer. To sustain a high throughput and efficiency, the PE array needs to be maximally utilized by having the AIM produce enough W-IA inputs, and the reduce stages consume as many psums as possible before writeback.

Associative Index Matching

To facilitate psum reduce, SNAP streams compressed Ws and IAs in a channel-first ordering. To extract enough W-IA inputs, AIM employs a 2D comparator array to discover W-IA pairs of matching channel index that can be readily multiplied (Fig. 4a). The 2D comparator results are read out in parallel by priority encoders to obtain a list of IA addresses that have matching Ws. A sequence decoder then locates the addresses for dispatching data to the MULT array to compute psums (Fig. 4b, Fig. 4c). The size of the 2D comparator array determines the search depth, and it needs to be sufficiently large to extract enough W-IA inputs to maintain high MULT utilization. However, the area and power of AIM increase quadratically with the size. In the SNAP test chip, we choose a 32×32 AIM to serve a row of 3 PEs in a time-multiplexed fashion, keeping the average MULT utilization above 75% for common workload densities (Fig. 4d) and limiting the AIM area overhead in a core below 12.5%.

Two-Level Partial Sum Reduce

A PE consists of 3 MULTs and computes 3 psums per cycle. In the majority of the cases, the 3 psums are C-reduced to 1 OA. A PE uses a configurable adder tree (Fig. 5) to support the 3:1 C-reduce as well as the cases when the 3 psums belong to different OAs, i.e., when W's and/or IA's pixel address advance. A PE retains psums until C-reduce is complete, cutting the writeback traffic by up to the channel depth.

After C-reduce is complete, a PE passes the final psum and its OA address to the core reducer. The core reducer reduces psums of matching OA address along lanes of the PE array based on the layer type as described in the next section. The core-level reduce cuts the writeback traffic by another 2.3-3.0×. The two-level reduce resolves access contention completely. For common workload densities, the traffic is cut down to 2.79 OA writeback per cycle per core (63 MULTs), which is 22× lower than the SCNN baseline.

PE Array Configuration

SNAP provides two PE array configurations, diagonal and row, to support different kernel and layer types. The diagonal configuration supports CONV layers with P-reduce, e.g., in a 3×3 CONV, compressed IA data of consecutive pixels are broadcast to consecutive PE rows; and the 3×3 kernels are split into 3×1 slices, compressed and broadcast to consecutive PE columns. The streaming order allows the PEs along a diagonal to compute psums of the same OA address for P-reduce (Fig. 6a). Arbitrary R×S CONV kernels are divided into R×3 sub-kernels, each assigned to a core. The row configuration supports layers without P-reduce opportunity, e.g., in a 1×1 CONV, IA and W are partitioned in the C dimension and multicast to PEs, allowing the PEs along each row to compute psums of the same OA address for C-reduce (Fig. 6b). A FC layer can be supported in a similar way by further partitioning in the K dimension.

Chip Measurement Results

A 2.4 mm^2 SNAP test chip (Fig. 7) was implemented in 16nm CMOS. At 0.55V, 260MHz, and in room temperature, the chip is measured to achieve a peak effectual efficiency of 1.67, 5.06 and 21.55TOPS/W (16b) on 100%, 40% and 10% W and IA density benchmarks, respectively (Fig. 8a). Following the pruning technique [2], SNAP's runtime performance and efficiency are evaluated using pruned AlexNet, VGG-16 and ResNet-50, demonstrating effectual efficiency of 3.86, 3.79 and 3.61TOPS/W for the three networks at 0.55V, 260MHz. At 0.8V, 480MHz, SNAP provides an inference throughput of 90.98fps on the pruned ResNet-50, dissipating 348mW (Fig. 8b). Compared to the state-of-the-art dense [6]-[8] and sparse [3], [4] accelerators, SNAP offers competitive performance and efficiency (Tables I and II) by maintaining high utilization and low writeback data traffic.

Acknowledgements

We thank Bill Dally, Joel Emer, and Angshuman Parashar for advice, Cheng Fu and Tim Wesley for assistance with chip design. The Michigan team effort and chip fabrication were supported by Toyota Research Institute and DARPA.

References

[1] K. He, et al., CVPR 2016
[2] S. Han, et al., NIPS 2015
[3] A. Parashar, et al., ISCA 2017
[4] Z. Yuan, et al., VLSI 2018
[5] C.-E. Lee, et al., SysML 2018
[6] Y.-H. Chen, et al., ISSCC 2016
[7] B. Moons, et al., ISSCC 2017
[8] J. Lee, et al., ISSCC 2018

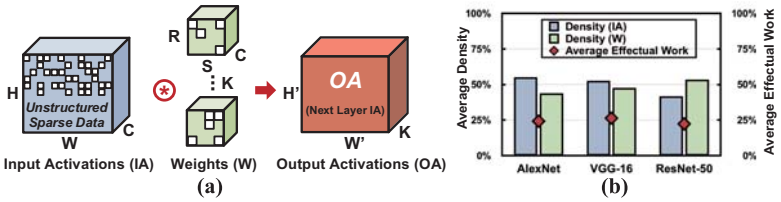

Fig. 1. (a) Convolution of unstructured sparse input activation (IA) and weight (W) in a sparse deep neural network (DNN), (b) Average IA, W densities and effectual work after network pruning.

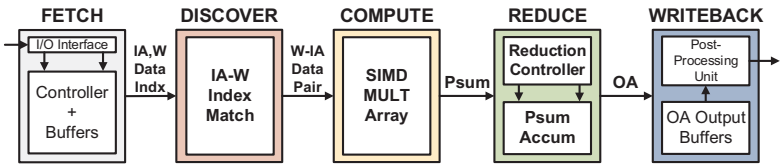

Fig. 2. Sparse DNN processing pipeline.

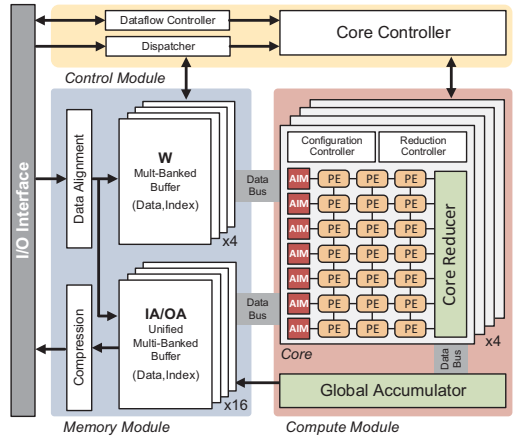

Fig. 3. SNAP system architecture, including a control module, a memory module and a 4-core compute module.

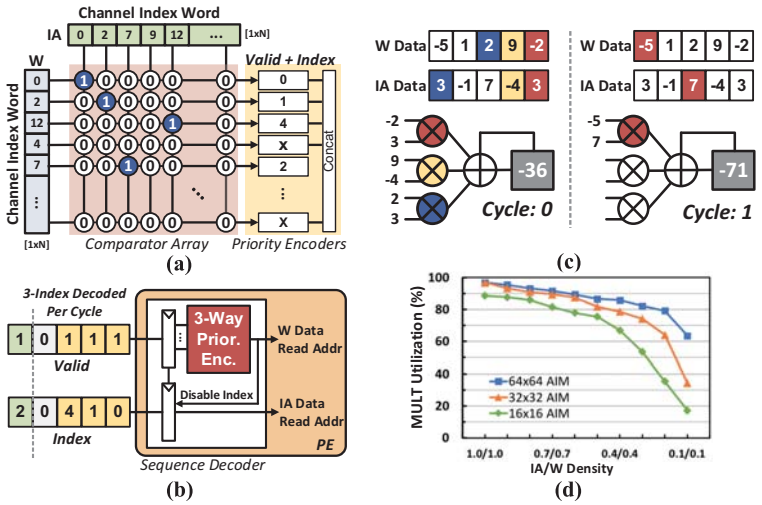

Fig. 4. (a) Associative index matching (AIM) implementation, (b) Sequence decoder implementation, (c) Sequence decoder mechanism illustration, (d) Multiplier array utilization with respect to different sized AIM implementations.

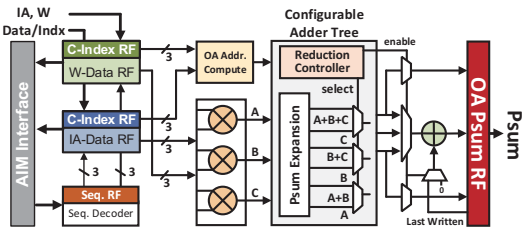

Fig. 5. Processing element (PE) implementation.

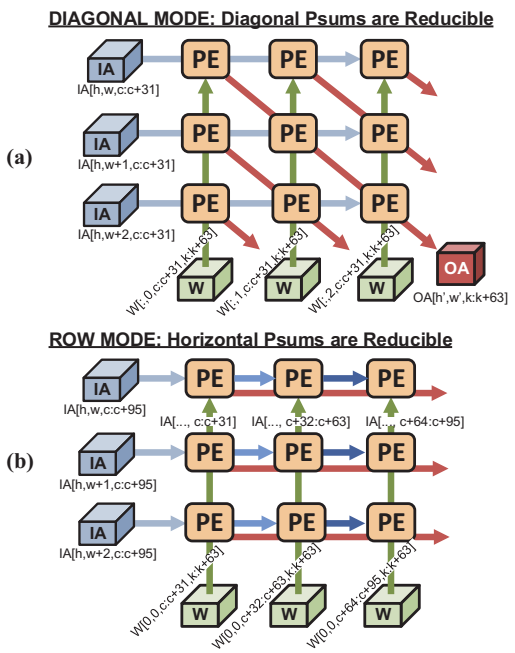

Fig. 6. PE array configuration and data partitioning of (a) a 3x3 CONV with diagonal reduce mode, (b) a 1x1 CONV with row reduce mode.

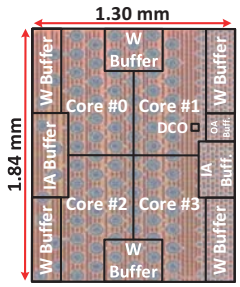

Fig. 7. SNAP test chip microphotograph.

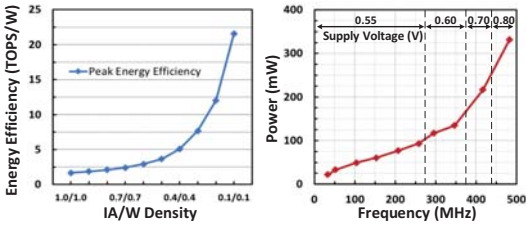

Fig. 8. (a) Energy efficiency for different sparse workloads, (b) Measured power and frequency at room temperature with voltage scaling.

TABLE I: COMPARISON WITH PRIOR WORKS

	Eyeriss [6]	Envision [7]	UNPU [8]	SCNN[a] [3]	Sticker [4]	This Work
Sparsity Mode	D	D	D	IA+W	IA+W	**IA+W**
Layer Type	CONV	CONV+FC	CONV+FC	CONV	CONV+FC	**CONV+FC**
Technology (nm)	65	28	65	16	65	**16**
Die Area (mm^2)	12.25	1.87	16	7.9	7.8	**2.4**
Multiplier Number	168	Nx256	13824[b]	1024	256	**252**
Data Width (bit)	16	1-16	1-16	16	8	**16**
On-Chip Buffer (KB)	108	144	256	1200	170	**280.6**
Supply Voltage (V)	0.82-1.17	0.55-1.1	0.63-1.1	N/A	0.67-1.0	**0.55-0.80**
Frequency (MHz)	100-250	50-200	5-200	1000	20-200	**33-480**
Power (mW)	235-332	7.5-300	3.2-297	N/A	20.5-248.4	**16.3-364**

D: Dense mode only; (a): Simulation work only; (b): 1-bit operations

TABLE II: PEAK EFFICIENCY (TOPS/W) ON SPARSE WORKLOADS

	Sparse Layers			Pruned Networks[a]		
	Dense	Medium	Sparse	AlexNet	VGG-16	ResNet-50
Eyeriss [6]	0.31 (16b)			0.17 (16b)	0.09 (16b)	-
Envision [7]	0.53 (16b)			0.8-3.8[b]	2.0[b]	-
UNPU [8]	3.08 (16b)			4.3 (8b)	4.71 (8b)	-
Sticker [4]	0.5 (8b)	2.5 (8b)	30 (8b)	1.038 (8b)	-	-
This Work	**1.67 (16b)**	**5.06 (16b)**	**21.55 (16b)**	**3.86 (16b)**	**3.79 (16b)**	**3.61 (16b)**

1MAC = 2Ops; Dense: 1.0/1.0, Medium: 0.4/0.4, Sparse: 0.1/0.1 IA/W density;
(a): Network pruned with no accuracy loss; (b): Bit-precision not provided