PNNPU: A 11.9 TOPS/W High-speed 3D Point Cloud-based Neural Network Processor with Block-based Point Processing for Regular DRAM Access

Sangjin Kim, Juhyoung Lee, Dongseok Im, and Hoi-Jun Yoo
School of EE, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon, 305-701, Republic of Korea
Email: sangjinkim@kaist.ac.kr

Abstract

An efficient and high-speed 3D point cloud-based neural network processing unit (PNNPU) is proposed using the block-based point processing. It has three key features: 1) page-based point block memory management unit (PMMU) with linked list-based page table (LLPT) for on-chip memory footprint reduction, 2) hierarchical block-wise farthest point sampling (HFPS), and block skipping ball-query (BSBQ) for fast and efficient point processing, 3) Skipping-based max-pooling prediction (SMPP) for throughput enhancement. The PNNPU is fabricated in 65nm CMOS process and evaluated on the 3D object detection (3D OD) application. As a result, it shows 84.8 fps at 266.8mW power consumption and achieving 6.6-11.9 TOPS/W energy efficiency.

Introduction

Recently, point cloud-based neural network (PNN) shows high performance in 3D perception such as classification, segmentation, hand pose estimation, and OD [1,2] (Fig. 1). PNN, however, is difficult to realize in edge devices because it requires memory-intensive point processing such as farthest point sampling (FPS) and ball query (BQ). The FPS iteratively finds the farthest point from the sampled point set to sample the centroid point most evenly, and BQ calculates the distance and finds neighboring points for constructing a local region set. Moreover, the convolution layers (CL) after point processing load input with *gather* operation which causes irregular and frequent external memory accesses with small chunks. As a result, the limited burst access and bank confliction induce low bandwidth utilization, so external access consumes 54.4% of total latency. Therefore, even in server-scale GPU [3], PNN shows low framerate (~10 FPS) [2].

Previous PNN processors [4] attempted approximation of point processing through regular sampling and grouping in the 2D projected point cloud. However, in the large scale 3D architecture, its approximation caused an uneven distribution of sampled points in 3D space and resulting in poor accuracy.

This paper presents PNNPU, an efficient high-speed PNN processor with block-based point processing (BPP). BPP partitions the point cloud into multiple blocks then processes sampling and grouping within block-level. Its limited searching area reduces the cost of point processing. Moreover, block-wise access to partitioned point cloud in memory regularize the external DRAM access pattern and reduce latency with burst access. The PNNPU with BPP is realized with the following three key features; 1) page-based point block memory management unit (PMMU) with liked list-based page table (LLPT) for 62.7× memory-efficient point partitioning, 2) hierarchical block-wise FPS (HFPS) and block skipping-BQ (BSBQ) for 18.0× efficient and 22.5× faster sampling and grouping, 3) skipping-based max-pooling prediction (SMPP) to reduce 49.9% of CL operation.

Proposed PNN Processor

The PNNPU consists of 4 DNN core clusters and three types of point processing cores, partitioning core (PTC), sampling core (SPC), and grouping core (GPC) (Fig. 2). The PTC partition the point cloud loaded from external and store in point-MEM. SPC and GPC accelerate sampling and grouping with block-based point processing. DNN core clusters consist of 6 IOSC with 8×8 PE array and accelerate CL with skipping zero in input or output feature map.

1. *The PMMU with LLPT* reduces the memory footprint for partitioning by managing point-MEM through paging (Fig. 3). In PTC, the block classifier divides point cloud into block according to 3D coordinate, and stores in point-MEM. However, pre-allocation of point-MEM with fixed memory area cause fragmentation and consume a huge memory footprint due to the imbalanced block size by non-uniform distribution of point cloud.

Therefore, for avoiding fragmentation, point-MEM is divided into pages and allocated by demand. However, because of the large virtual address for the imbalanced block, a large page table (PT) is required for paging. Therefore, LLPT is proposed for compressing the page table by utilizing a sequential access pattern within each block. LLPT stores the sequence of page ID (PID) for each block in a linked list format. The current PID for each block is saved in the recent page table (RPT), and the next PID of each page is saved in the linked list table (LLT). The number of accesses to each block is counted at the offset table(OT). When the current page is over, RPT is updated with the next PID in the LLT. As a result, page mapping is compactly stored in LLPT, further reducing the PTC memory footprint by 68%.

2. *The HFPS and BSBQ* reduce latency through a bounded searching space by block-based processing on a partitioned point cloud (Fig. 4). HFPS consists of two stages: Level 1, which determines the sampling number of each block for even sampling between blocks, and level 2, which performs block-wise FPS for even sampling inside the block. Because point clouds have different densities for each area, sampled points are uneven if the same number of points is sampled for each block. Therefore, in level 1, FPS is performed on the $1/s$ random sampled sparse point cloud and the number of points in each block is counted. Then scaling the counted number to s and the sampling number reflecting the density is obtained. After that, in level 2, FPS is performed block-wise with the obtained number and accelerating sampling by computing only distance within a block. Moreover, through block-wise sampling, HFPS not only accelerates sampling but also maintains a partitioned state for the next BPP.

The BSBQ caching only the neighboring blocks and accelerating BQ by skipping distance comparison with points in distant blocks. Moreover, since only points within the neighboring block are used for grouping, only the feature maps of the neighboring block are cached. Therefore, it is possible to access in burst mode without irregular external access because the feature map of the required point is loaded by block after the current block finish. As a result, the HFPS and BSBQ reduce 94.4% of energy and 95.6% of latency for point processing. (@ 20k point input w/ 16-level partition)

3. *The SMPP* reduces the computation of CL before the max-pooling layer (MPL) by prediction (Fig. 5). Most of the CL operation before MPL can be skipped through MPP. However, MPP proposed in [4] with heterogeneous core suffer from poor utilization in the CL without MPL. Proposed SMPP predicts the MPL using CL with only large value input after thresholding. Therefore, both the prediction CL and the original CL are accelerated by a homogeneous zero-skipping core. As a result, 49.9% of CL computation is reduced and PNNPU achieves 80.3% higher throughput and 69.0% higher efficiency in CL compared to the previous PNN processor [4].

Implementation Results and Conclusions

The PNNPU is fabricated in 65nm CMOS process with 16mm^2 area. (Fig. 6) The processor operates from 50MHz to 200MHz clock frequency with 0.78V to 1.1V supply voltage. Fig. 7 shows the benchmark result of PNNPU in the 3D OD using on SUN-RGBD dataset [6]. The PNNPU show negligible accuracy drop compared to baseline [2], but take only 11.79ms for inference which is ×8.5 times faster than GPU [3]. Furthermore, PNNPU showed an energy efficiency of 6.6-11.9 TOPS/W which is much higher than the previous processor [3-5] including GPU (Table 1) on the benchmark, at 50MHz clock frequency with 0.78V. In conclusion, a PNN processor with block-based point processing is proposed and achieving higher efficiency and lower latency compared to previous hardware.

References

[1] C. R. Qi, *NIPS, 2017.*
[2] C. R. Qi, *ICCV, 2019.*
[3] NVIDIA GP100 White Paper
[4] D. Im, et al. *VLSI Symp., 2019.*
[5] J. Lee, et al. *ISSCC, 2019.*
[6] S. Song, et al. *CVPR, 2015.*

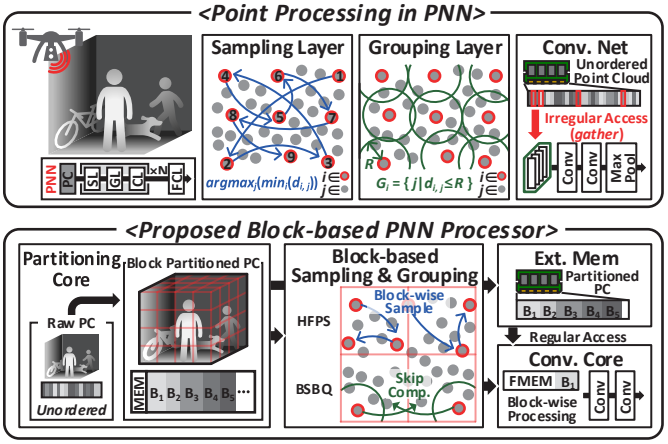


Fig. 1 Operation of PNN and Proposed Block-based Approach

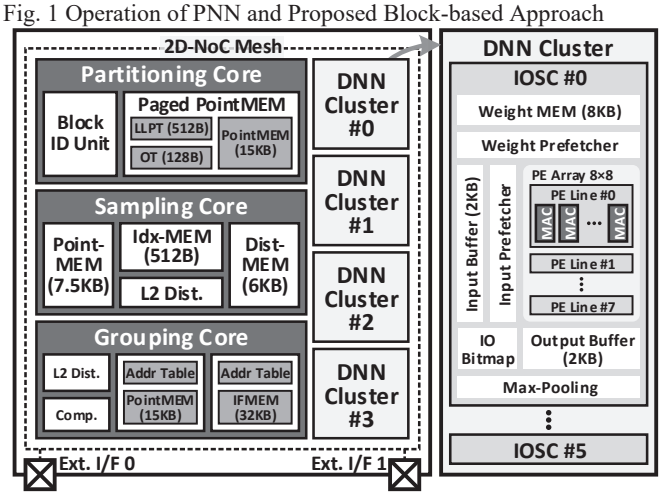


Fig. 2 PNNPU Overall Architecture

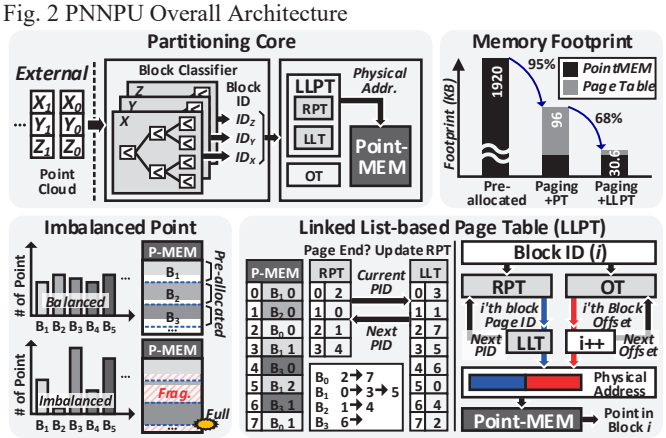
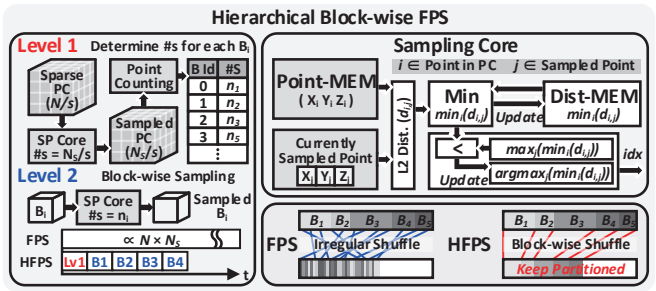


Fig. 3 Page-based Point Block Management Unit with LLPT



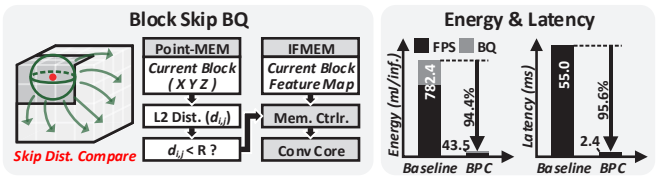


Fig. 4 Sampling Core with H-FPS and Grouping Core with BSG

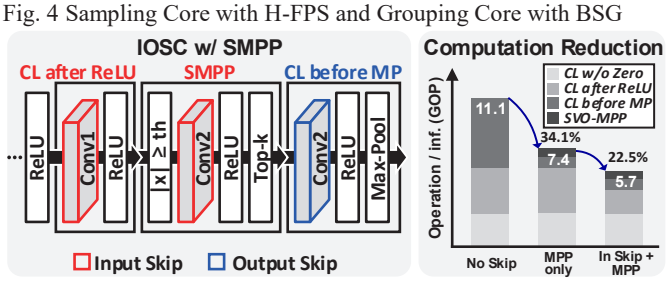
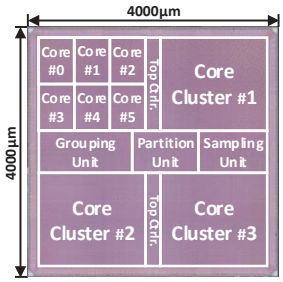


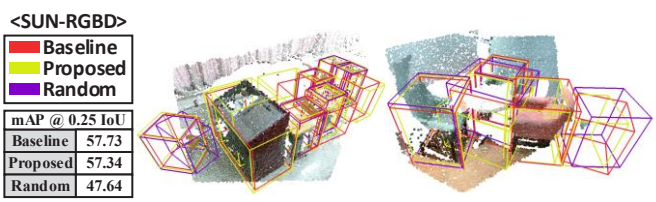
Fig. 5 Skipping-based MPP and Its Latency Reduction



		Specification	
Technology		65nm Logic CMOS	
Die Area		4000µm × 4000µm	
SRAM		364 KB	
Frequency		200MHz @ 1.1V	50MHz @ 0.78V
Peak Power [mW]	SG	23.2	3.3
	CL	237.0[1] - 308.2[2]	27.2[1]-38.5[2]
Peak Performance [GOPS]	SG	23.2[3] - 604.3[4]	5.8[3] – 151.1[4]
	CL	614.4[1]- 2808.7[2]	153.6[1]-702.17[2]
Efficiency @ MAX. GOPS [TOPS/W]	SG	1.00[3] – 26.1[4]	1.74[3]-45.4[4]
	CL	2.59[1]-9.11[2]	5.65[1]-18.23[2]

1) 0% Input Sparsity, 2) 90% Input Sparsity, 3) w/o HB-FPS & BS-BQ, 4) w/ HB-FPS& BS-BQ

Fig. 6 Chip Photograph and Performance Summary



mAP @ 0.25 IoU	
Baseline	57.73
Proposed	57.34
Random	47.64

	L0	L1	L2	L3	L4	L5	L6	L7	L8	Total
Latency [ms]	2.46	4.56	1.75	0.87	0.30	0.58	0.51	0.73	0.03	**11.79**
Power [mW]	296.9	267.4	254.7	253.0	241.3	241.3	246.8	253.6	245.8	**266.8**

Fig. 7 Measurement Result

Table 1. Comparison Table

	ISSCC 19 [5]	GP-100 [3]	S. VLSI 20 [4]	This Work
Technology	65nm	16nm	65nm	65nm
Purpose	CNN, FC, RNN	General	PNN	PNN
Frequency [MHz]	200	1304	150	200
Area [mm^2]	16	610	16	16
Peak Power [mW]	367	300,000	402	331
Energy Efficiency (TOPS/W)	6.87[1]	0.069	5.62[2]	6.6-11.9[3]
PNN Performance				
Point Processing (Sampling / Grouping)	-	FPS / BQ	Random / Window-based	HB-FPS / BS-BQ
3D OD Framerate[3] [fps]	-	10.0	≤ 44.4[4]	84.8

1) Assume 50% Sparsity by ReLU, 2) Conv. Performance,
3) VoteNet @ SUN RGB-D (20k Point Cloud), 4) Estimated with Maximum Performance