# *An Energy-Efficient 3D Point Neural Network Accelerator with Fine-grained LiDAR-SoC Pipeline Structure*

# An Energy-Efficient 3D Point Neural Network Accelerator with Fine-grained LiDAR-SoC Pipeline Structure

Bokyoung Seo†, Jueun Jung†, Donghyeon Han‡, and Kyuho Jason Lee†
†UNIST, Ulsan, Republic of Korea, ‡MIT, Cambridge, USA
{kyoung8523, jueunjung, kyuho.jsn.lee}@unist.ac.kr, hdh4797@mit.edu

## ABSTRACT

3D point neural network (PNN) segmentation using LiDAR data has emerged as a fundamental stage of high-level intelligence algorithms for autonomous applications such as SLAM, path planning, object detection, etc. However, previous processors were not feasible for real-time and low-power 3D PNN systems since they wasted ~100 ms of LiDAR's sensing time and required 107.3 mW of external memory access before PNN processing. Furthermore, their compute-intensive bin partitioning and point sampling methods were not suitable for large-scale outdoor data, causing significant computing power. Therefore, the *entire system*, from sensing to processing, must be taken into account for 3D PNN processor implementation. This paper proposes L-PNPU, an energy-efficient 3D PNN segmentation processor optimized with the unique mechanical characteristics of LiDAR. It is designed with three key features: 1) Azimuthal bin partitioning to reduce power and latency, 2) Modified PNN algorithm co-optimized with heterogeneous architecture to remove redundant operation and reduce energy, and 3) Fine-grained LiDAR-System-on-Chip (SoC) pipeline structure to enhance the system energy and throughput. At 250 MHz and 1.0V, L-PNPU achieves 1.27M points/s of throughput and 0.51 μJ/point of energy efficiency.

## CCS CONCEPTS

• **Hardware → Application-specific VLSI designs**

## KEYWORDS

LiDAR, point cloud, 3D PNN segmentation, PNN accelerator

**ACM Reference format:**
Bokyoung Seo, Jueun Jung, Donghyeon Han, and Kyuho Jason Lee. 2024. An Energy-Efficient 3D Point Neural Network Accelerator with Fine-grained LiDAR-SoC Pipeline Structure. In ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '24), August 5–7, 2024, Newport Beach, CA, USA. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3665314.3670846

## 1 INTRODUCTION

The 3D point cloud neural network (PNN) that recognizes the surrounding environment has aggressively progressed over the past decade [1] as an essential function for autonomous driving technology in unmanned vehicles, drones, mobile robots, etc. With the advancement of LiDAR technologies, the performance of 3D perception has been propelled by directly using accurate depth information of surrounding objects with high resolution [2]. As a result, the LiDAR-based PNN segmentation became an integral part of accurate high-level intelligence tasks such as simultaneous localization and mapping (SLAM), path planning, and obstacle detection [3, 4, 5].

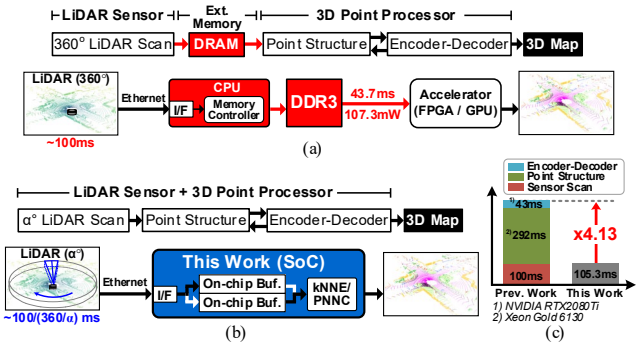



**Figure 1: The system dataflow of (a) previous works and (b) the proposed L-PNPU, and (c) the system latency comparison.**

The point cloud is a set of $\{x, y, z\}$ data points scattered in a 3D space with *unstructured* and *unordered* nature [1]. Nevertheless, early PNNs [6, 7] adopted the feature extractor of traditional Convolution Neural Network (CNN), disregarding the nature of the point cloud data. To fit the input dimensions of CNN, they inevitably required a pre-processing stage of point cloud transformation via 3D projection or discretization, even though such methods lose point data. In addition, transformation needs the full 360°-scanned LiDAR data to process, which is impractical for a real-time operation since LiDAR's sensing time solely takes ~100 ms [8]. Therefore, recent PNN algorithms [9, 10] take raw point data as input and perform segmentation on each point, providing detailed and accurate objects' boundaries. They can also immediately compute with only a partial angle of LiDAR data, facilitating a parallelized LiDAR-System-on-Chip (SoC) process. They require specific functions such as farthest point sampling or random sampling for *point sampling*, k-nearest neighbor (kNN) searching for *point grouping*, and shared multi-layer perception (MLP) at each layer because of the *unordered* and *unstructured* properties of the point cloud. However, its implementation on low-power SoC encounters problems because kNN searching and shared MLP operations considerably increase power consumption due to their complex computations and large amounts of internal and external memory access.


*ISLPED '24*, August 5–7, 2024, Newport Beach, CA, USA

ACM ISBN 979-8-4007-0688-2/24/08.
https://doi.org/10.1145/3665314.3670846

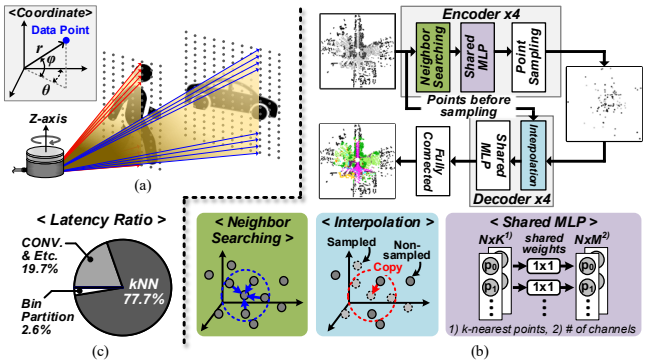



**Figure 2: (a) The coordinate system and laser scan pattern of LiDAR. (b) The visualization of PNN architecture and special functions. (c) Latency ratio of PNN process.**

Previous 3D perception processors [11, 12, 13, 14, 15] aimed for real-time PNN processing, but they neglected both the property of LiDAR and the massive number of points in outdoor environments. They partitioned the input point cloud into Cartesian coordinates to limit the processing area and reduce point processing cost. When using their partitioning method, the fully scanned LiDAR data must be stored in the external memory after unproductively wasting at least 100 ms for LiDAR sensing. In addition, 43.7 ms of latency and 107.3 mW of power consumption for external memory access are required, as depicted in Figure 1(a). Furthermore, they have employed the farthest point sampling as a point sampling method of PNN. However, the farthest point sampling has the computational complexity of $O(M^2N)$ when sampling M points from a point cloud with N points, which is feasible only with the lowest resolution of LiDAR (< 16K points). If farthest point sampling is applied to the higher resolution of LiDAR (< 133K points) to sample 10% of points, the computation increases by 574.4× than random sampling. Therefore, previous works with Cartesian-based bin partitioning and farthest point sampling are impossible to implement a low-power and real-time 3D PNN system with large-scale LiDAR data.

To efficiently perform segmentation on large-scale point clouds, this paper accelerates RandLA-Net [10], which utilizes random point sampling for high efficiency of both computation and memory and mitigates the accuracy loss with a local feature aggregation module. Unlike the farthest point sampling, random point sampling requires the constant time complexity of O(M) regardless of the number of points, ensuring real-time operation and low power consumption. However, even with the high-end CPU and GPU, the serial operation of LiDAR sensing and PNN acceleration consumes 432 ms of system latency, as shown in Figure 1(c). Therefore, this paper proposes L-PNPU, the energy-efficient 3D PNN processor optimized for LiDAR. The entire system of L-PNPU removes unnecessary external memory access between LiDAR and the processor by directly utilizing the LiDAR's sensing time, as shown in Figure 1(b). The L-PNPU has the following hardware features: 1) Azimuthal bin partitioning to reduce 90.5% of power and 90.8% of latency; 2) Modified PNN algorithm co-optimized with heterogeneous architecture to remove redundant computation, reducing energy by 20.1%; 3) Fine-grained LiDAR-SoC pipeline structure to support the parallel processing of the entire system, reducing 76.4% of processing time and 99.9% of system energy.

## 2 PRELIMINARIES & MOTIVATION

To design the 3D PNN accelerator optimized to LiDAR, it is essential to have a thorough understanding of both LiDAR sensor and 3D PNN segmentation. Figure 2(a) illustrates the mechanical characteristics of LiDAR. LiDAR rotates along the azimuth direction while firing laser points at each angle as much as vertical resolution (e.g., 16, 32, 64 channel). Even though LiDAR has a strict firing pattern, the number of obtained points varies due to the object's reflectance or environmental conditions, resulting in the point cloud's *spatially sparse*, *unstructured*, and *unordered* properties [1]. Therefore, the points are not arranged in specific sequences as in a 2D-pixel array, and 3D PNN has a different structure from traditional 2D CNNs, as shown in Figure 2(b). Before processing PNN, the $\{r, \theta, \varphi\}$ of LiDAR data is converted into $\{x, y, z\}$ coordinates, which is the typical format of the point cloud dataset. The encoder layer processes point structure, shared MLP, and point sampling in series. At first, PNN must define the structure among points by neighbor searching algorithm (i.e., kNN) due to the *unstructured* nature of the point cloud. Then, it extracts the local features from the individual point using a shared MLP which is 1×1 convolution. Finally, point sampling is applied to reduce the number of large-scale points for efficient computation. The decoder layer serially performs interpolation and shared MLP to obtain the segmented results. Interpolation also utilizes kNN but finds the structure between points before and after sampling in the encoder layer.

The abovementioned characteristics bring new challenges to designing a 3D PNN processor optimized to LiDAR. First, the LiDAR data is unevenly distributed over space and becomes sparser as the distance from LiDAR increases. Thus, the identical size of bin partitioning triggers a severe imbalance of data across bins, reducing power efficiency by hindering the clock-gating of idle cores. Second, the pipeline between LiDAR and SoC must be preserved even with different resolutions of LiDAR. If the computation time exceeds the sensing time, the inevitable pipeline stall between LiDAR and processor ruins real-time processing and energy efficiency. Third, kNN with its time complexity of $O(N^2)$ is performed twice for each pair of encoder-decoder layers, which is eight sets of kNN operations in total. Hence, the kNN operation takes 77.7% of the overall processing time of PNN, as shown in Figure 2(c), preventing the low-energy computation. Therefore, this paper suggests the novel bin partition method to preserve low-power consumption with a balanced workload, and the modified PNN to reduce the kNN operation. In addition, the fine-grained LiDAR-SoC pipeline structure with the dedicated heterogeneous architecture reduces system energy as well as latency.

## 3 ALGORITHM FEATURES

### 3.1 Azimuthal Bin Partitioning

Previous PNN accelerators [11, 12] adopted a Cartesian-based bin partitioning to reduce the computation cost of point structure by

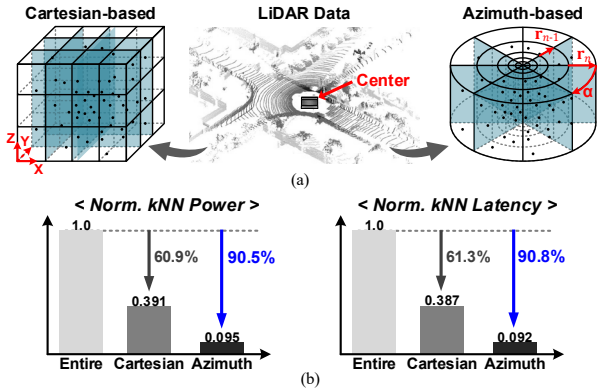



**Figure 3: (a) Example of cartesian-based and azimuth-based bin partitioning. (b) kNN performance comparison: 1) Entire data without partitioning, 2) Cartesian-based partitioning, and 3) Azimuth-based partitioning.**

limiting the searching area. However, [16, 17] observed the pattern of LiDAR data distribution in which points are densely concentrated near the LiDAR, resulting in a severe workload imbalance among bins with the prior method. As mentioned in Section 2, LiDAR scans the surroundings along with azimuth direction with uniform vertical channels and horizontal intervals. Therefore, this paper employs azimuthal bin partitioning to balance point distribution across bins. Figure 3(a) illustrates the concept of the previous bin partitioning and the proposed azimuthal bin partitioning in 3D point space using LiDAR data from the semanticKITTI [18]. The center of the 3D space is the location of the LiDAR. The proposed method partitions the 3D space with α° azimuth angle and divides the range into five intervals. The optimal degree of α is found by exploring the semanticKITTI and nuScenes [19] described in Section 6. The range values ($r_n \in \{r_1, r_2, r_3, r_4, r_5\}$) are determined using the Reinforcement Learning (RL) algorithm that finds the optimal values by iteratively assigning either reward or penalty, which is described by thresholding the accuracy and the evenness of data distribution. RL is applied not only to ensure even data distribution among bins but also to minimize PNN accuracy loss. Figure 3(b) shows the kNN performance results of three cases: without bin partitioning, Cartesian-based bin partitioning, and the proposed bin partitioning. Compared to the kNN on entire data of 360° without bin partitioning, the azimuthally partitioned data reduces 90.5% of computation power and 90.8% of latency in kNN operation. Also, the proposed method requires 76.3% less computation than Cartesian-based bin partitioning by balancing the workload among bins. Besides its advantages in kNN computation, the proposed method also enables sensor-level pipelining and utilizes the sensing time described in Section 6.

## 3.2 Modified Network Architecture

This work proposes the modified PNN architecture to reduce the computation, power, and memory overhead of kNN without any accuracy drop. As shown in Figure 4, it priorly performs point sampling to mask whether the point is sampled or non-sampled. Then, a unified kNN (U-kNN) algorithm is processed to find index results of neighbor searching in the encoder layer and

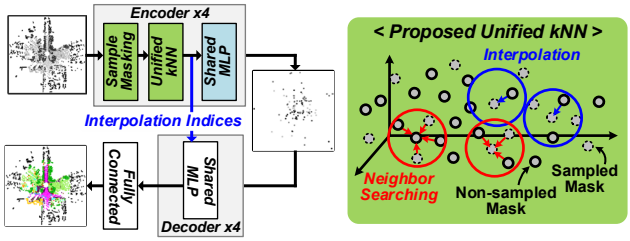



**Figure 4: The architecture of the modified PNN algorithm (left) and visualization of the proposed unified kNN (right).**

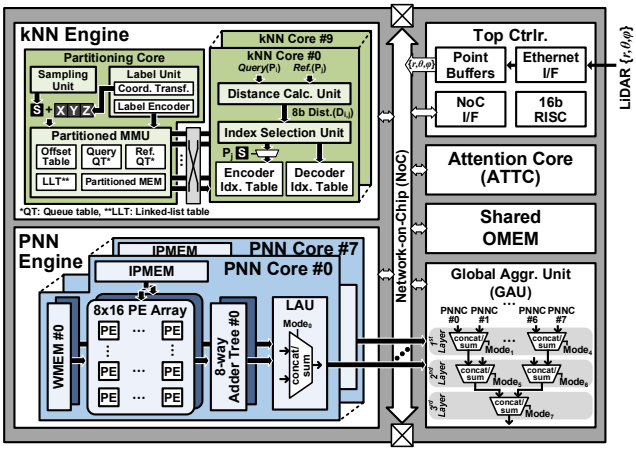



**Figure 5: Overall architecture of the proposed processor.**

interpolation in the decoder layer simultaneously using the sampling mask. As the random sampling process randomly reduces the number of point data, changing the order of sampling and kNN does not affect accuracy. In addition, the kNN operation in the decoder finds the neighbor information between points before and after sampling, as explained in Section 2. Therefore, the kNN indices of the decoder are also obtainable while performing kNN at the encoder stage.

# 4 HARDWARE FEATURES

## 4.1 Overall Architecture

The overall architecture of L-PNPU is described in Figure 5. It consists of kNN engine (kNNE), attention core (ATTC), global aggregation unit (GAU), top controller, shared OMEM, and PNN engine (PNNE). Top controller contains the Ethernet interface and point buffers for data transactions from LiDAR, NoC interface for the interconnection of cores, and a 16-bit RISC to control the overall process of PNN. For fast and efficient point structure processing, kNNE is composed of two units: 1) partitioning core with partitioned memory management unit (MMU), sampling unit, and label unit, and 2) ten kNN cores (kNNC) with index selection unit (ISU) and distance calculation unit (DCU). PNNE contains eight PNN Cores (PNNCs), and each PNNC includes two 8×16 PE arrays with two adder trees, two weight memories (WMEM), input point memory (IPMEM), and local aggregation unit (LAU). It accelerates the shared MLP layer and the FC layer of 3D PNN by supporting *weight stationary* data flow. GAU has three aggregation layers to compute the outputs of eight PNNCs. LAU and GAU perform in either "concatenation" mode or "sum" mode

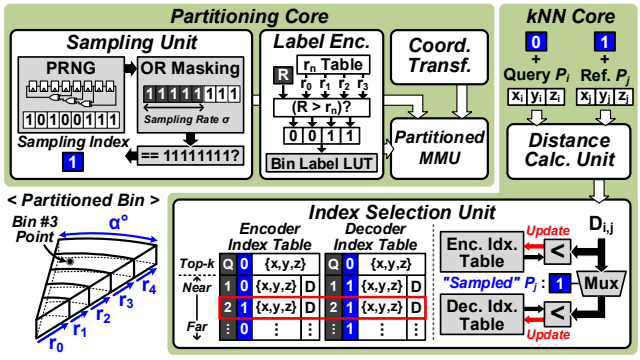



**Figure 6: Details of partitioning core and kNNC in kNNE.**

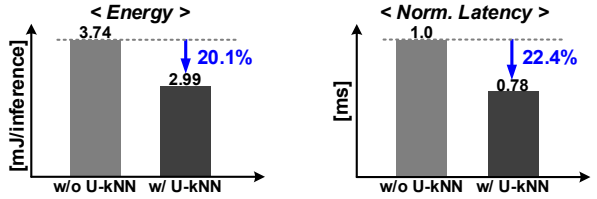



**Figure 7: The performance comparison without and with unified kNN (U-kNN) using the PRNG-based sampling circuit.**

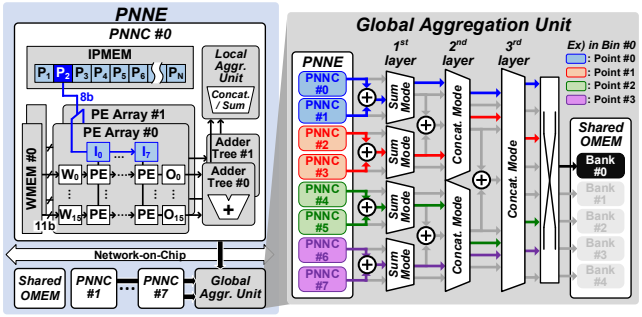



**Figure 8: Details of PNNC and GAU.**

managed by top controller. ATTC performs max pooling and softmax operations to extract weighted features of each point.

LiDAR sensor transmits data packets to L-PNPU through the Ethernet interface. The interface circuit analyzes the packet, which contains point cloud data along with the channel of LiDAR in 16-bit dynamic fixed-point (FXP) $\{r, \theta, \phi\}$ format. The input points are stored in the point buffers in a double-buffering manner. It facilitates L-PNPU in performing sensor-level pipeline because L-PNPU can compute with the points from the occupied buffer while the points in the following range are stored in the other buffer space (See Figure 1(c)). After the α ° points are stored in the buffer, top controller of L-PNPU begins PNN operation.

## 4.2 Details of kNNE, PNNE, and GAU

Figure 6 represents the detailed architecture of partitioning core and kNNC that supports the U-kNN operation of modified PNN. At first, coordinate transformation unit, label encoder, and sampling unit are performed simultaneously. The coordinate transformation unit converts 16-bit dynamic FXP $\{r, \theta, \phi\}$ data into 8-bit dynamic FXP $\{x, y, z\}$ data by the trigonometry ALU. At the same time, the label encoder determines the bin label by the range value ($r_n$), allocating an input point to one of the five

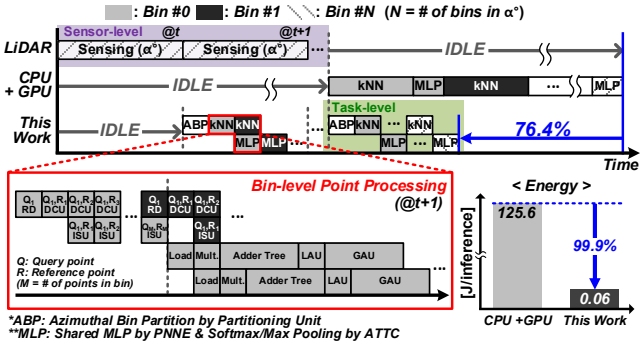



**Figure 9: The comparison of CPU+GPU system without pipelining and proposed fine-grained pipeline structure.**

bins. Meanwhile, a pseudo-random number generator (PRNG) in the sampling unit, controlled by the sampling ratio of network configuration, randomly masks the point as either sampled (1) or non-sampled (0). As a result, the masking operation of PRNG provides information about the points before and after random sampling. Finally, the $\{x, y, z\}$ data, bin labels, and sampling indices are stored in the linked-list page table-based [11] partitioned MEM by partitioned MMU. After the partitioning unit is operated, kNNCs perform kNN per bin. To support the parallel processing of kNN, the partitioned MMU transmits ten different query points and the same reference point within a bin to each kNNC. Then, DCU of kNNC computes the L2 distance between the received query point and reference point. The distance value from DCU is used as input of ISU to obtain the encoder kNN index and decoder kNN index at the same time during the nearest distance sorting process. The encoder index is determined by searching the top-k nearest points for the whole sampled and non-sampled reference points within the bin, and the decoder index is acquired by selecting the top-k nearest points among the sampled reference points. Therefore, L-PNPU supports the U-kNN by utilizing the PRNG-based sampling circuit and ISU. As shown in Figure 7, the energy consumption and latency are decreased by 20.1% and 22.4% compared to the entire kNN operations.

Since shared MLP applies the same set of weights to all the input points, the SIMD PE arrays can quickly accelerate parallel execution. As shown in the left of Figure 8, each PE array has its own WMEM, while both arrays share IPMEM. The 8-bit dynamic FXP input points fetched from shared OMEM are stored in IPMEM, and 11-bit dynamic FXP weights are stored in WMEM. Due to PNNE also operating in bin-level point processing, IPMEM broadcasts multiple channels of the same input data to each column of the PE arrays. Therefore, WMEM unicasts different weights for each row of the PE arrays, sending weight channels to the corresponding column. Then, the eight-way adder tree adds the multiplied results of each PE. LAU aggregates the results of two PE arrays; then GAU hierarchically aggregates the results of eight PNNCs according to the *concat/sum mode* selected by the top controller. Each aggregation layer of GAU can operate in a different mode, resulting in one or multiple data outputs. The right of Figure 8 describes the example of GAU operation. If two PNNCs share the same point data with different channels, the first layer of GAU works in *sum* mode and the others in *concat* mode.

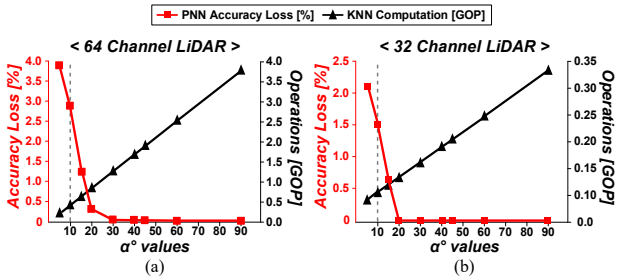



**Figure 10: The KNN computation and PNN accuracy loss along with different values of azimuth angle (α°) at the (a) semanticKITTI [18] and (b) nuScenes [19].**

**TABLE 1: ABLATION STUDY**

| Ablation | | | Accuracy [%] | | Latency [ms] | |
|---|---|---|---|---|---|---|
| Baseline | Azimuth | RL | [18] | [19] | [18] | [19] |
| ✓ | | | 88.97 | 89.95 | 7583.7 | 1436.7 |
| ✓ | ✓ | | 88.60 | 89.95 | 436.3 | 86.4 |
| ✓ | ✓ | ✓ | 86.46 | 89.58 | 100.0 | 22.8 |
| | | Performance | -2.51 | -0.37 | ×**75.8** | ×**63.0** |

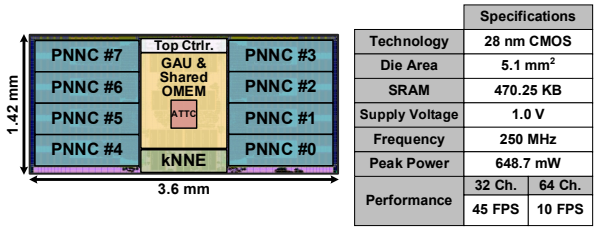
As the shared OMEM has five banks for five bins, GAU stores the intermediate data in the bank assigned to the bin. The proposed hierarchical operation of LAU and GAU accommodates the various layer configurations in the encoder-decoder network architecture. The hierarchical combination allows PNNCs to work in parallel as a group by the power of two or to be used as one to efficiently process the encoder-decoder network.

## 5 Fine-grained Pipeline Structure

The pipelined heterogeneous core architecture aims to enhance throughput by employing efficient data scheduling methods between cores and parallelizing serial operations. Figure 9 represents the data flow of the proposed fine-grained pipeline with three levels: sensor-level, task-level, and bin-level. The sensor-level pipeline primarily processes the α°-scanned data without waiting for LiDAR sensor to complete the full 360° scan. The task-level pipeline parallelizes the sequential operations of PNN through kNNE and PNNE. Finally, the bin-level pipeline operates kNNCs and PNNE for each bin. When all three pipeline levels are fully utilized, L-PNPU can operate without any stall between LiDAR sensor and the PNN processor. The timing diagram shows that the azimuthal bin partitioning *(ABP)* must be completed before the kNN operation. Therefore, the partitioning unit and kNNCs have to operate sequentially. After the partitioning unit completes processing all incoming points, kNNCs and PNNE perform bin-level point processing of kNN and shared MLP, as shown in the red box. As a result, the fine-grained pipelining reduces system latency by 76.4% and energy by 99.9%, and enhances throughput by 4.2× compared to the baseline.

## 6 EVALUATION & IMPLEMENTATION

In Section 6, the evaluation of the azimuthal bin partitioning and L-PNPU processor is conducted. The experiment for optimal angle value and ablation study for RL-based ranges are performed




| | Specifications | |
|---|---|---|
| Technology | 28 nm CMOS | |
| Die Area | 5.1 mm² | |
| SRAM | 470.25 KB | |
| Supply Voltage | 1.0 V | |
| Frequency | 250 MHz | |
| Peak Power | 648.7 mW | |
| Performance | 32 Ch. | 64 Ch. |
| | 45 FPS | 10 FPS |

**Figure 11: Layout photograph and specifications of L-PNPU.**

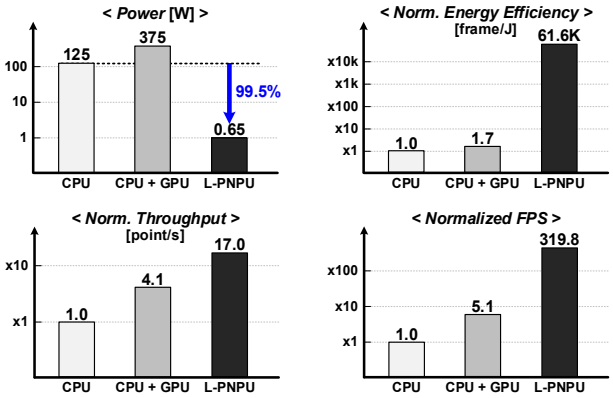



**Figure 12: The performance comparison of CPU, CPU+GPU, and L-PNPU.**

on the semanticKITTI and nuScenes datasets using raw 3D points $\{x, y, z\}$ obtained by Velodyne HDL-64E and HDL-32E in each case. Then, the performance analysis of L-PNPU compared to high-end CPU+GPU and state-of-the-art processors is presented.

### 6.1 Optimal Value of Azimuth Angle (α°)

The optimal value of azimuth angle (α°) is selected to minimize both PNN accuracy loss and kNN computation. While increasing the value of α° can minimize the PNN accuracy loss, the amounts of kNN computation are linearly increased. Figure 10(a) and (b) indicate the trade-off between PNN accuracy loss and kNN computation evaluated with semanticKITTI and nuScenes datasets. When comparing 10° and 90°, the required kNN operations in semanticKITTI and nuScenes escalate by 17.8× and 7.0×. For fast processing, using 10° decreases the required kNN computations, but it drastically loses PNN accuracy by up to 2.9%, which means it cannot perform further optimization with range. Therefore, the optimal angle of the azimuthal bin partitioning is 20° with the margin of PNN accuracy loss and the least kNN computation.

### 6.2 Ablation Study

Table 1 presents the results of ablation experiments on the proposed azimuth-RL-based bin partitioning method for the PNN segmentation with datasets of different resolutions. Because PNN is executed only with the point data in the azimuthally partitioned bin, its accuracy drop is inevitable even though we retrained the model. As a result, the PNN accuracy decreased by 1.10% and 0.37% in the semanticKITTI and nuScenes. Despite the drawbacks, the proposed partitioning method achieves 75.8× and 63.0× faster processing latencies than baseline, even without hierarchical pipelining.

**TABLE 2: COMPARISON TABLE WITH SYSTEM-ON-CHIP DESIGNS**

| | VLSI'21[11] | JSSC'22[13] | DAC'23[12] | This Work |
|---|---|---|---|---|
| Technology | 65 nm | 65 nm | 28 nm | 28 nm |
| Application | OD[1)] | HPE[2)] | SS[3)] | SS[3)] |
| Target Sensor | RGB-D | RGB-D | LiDAR | LiDAR |
| Sensor-Friendly? | No | No | No | **Yes** |
| # of points | 20K | >50K (QVGA) | 16K | **58K/133K** |
| Area [$mm^2$] | 16 | 16 | N/A | 5.1 |
| Frequency [MHz] | 200 | 150 | 200 | 250 |
| Peak Power [mW] | 331 | 273 | N/A | 648.7 |
| Energy [μJ/point] | 1.85 | 0.57 | N/A | **0.51/1.12** |
| Throughput [point/s] | 0.18M | 0.48M | 0.15M | 0.58M/1.27M |
| *Latency [ms] | 11.94 | 4.64 | 8.06 | 1.21/5.29 |

* Remained latency after LiDAR's sensing time (100ms) with EMA (0.4 GB/s of BW).
1) In-door, Object Detection 2) In-door, Hand Pose Estimation 3) Out-door, Semantic Segmentation

## 6.3 Performance Analysis

Figure 11 presents the layout photograph and specifications of L-PNPU. L-PNPU is designed in Verilog HDL and synthesized in Samsung 28nm CMOS technology with a standard cell library. It is verified through RTL simulations to count the exact cycle, and the power consumption is analyzed using Synopsys PrimePower by annotating the switching activity. L-PNPU occupies 5.1 $mm^2$ and consumes 648.7 mW, operating at 1.0 V and 250 MHz. Under these conditions, L-PNPU performs 45 FPS with the 32-channel LiDAR and 10 FPS with the 64-channel LiDAR in each case.

Figure 12 compares the performance of L-PNPU with CPU and CPU+GPU using 64-channel LiDAR data. Intel's Xeon Gold 6130 is used for the CPU, and Nvidia's RTX 2080Ti is used for the GPU. The vertical axis of the graphs is in log scale, and energy efficiency, FPS, and throughput are normalized to the case of the CPU. L-PNPU reduces 99.5% of power consumption compared to the CPU while the CPU+GPU is increased. Also, the proposed work improved the energy efficiency by 61.6K× than that of CPU while CPU+GPU only obtained 1.7× improvement. Finally, the *entire system* improved throughput and speeds-up by 17.0× and 319.8× respectively.

Table 2 compares L-PNPU with System-on-Chip designs [11, 12, 13] that directly use point cloud. The target applications of [11, 13] were indoor scenes obtained by RGB-D sensor, which are less complex than outdoor scenes. While [12] targeted semantic segmentation with LiDAR sensor, its performance was limited to 16K points regardless of LiDAR resolution. Also, it achieved only 0.15M points/s of throughput due to ignorance of LiDAR's sensing time. Compared to the previous works, L-PNPU is the only sensor-friendly processor capable of utilizing high-resolution LiDAR sensors (i.e., 32-channel and 64-channel) and processing massive points of outdoor environments. In addition, it entirely eliminates the usage of DRAM between LiDAR and processor, while previous works required external DRAM. As a result, L-PNPU achieves the highest throughput of 1.27M points/s with only 5.29 ms of latency. For the case of 58K points (32-channel), it achieves 0.58M points/s and 1.21 ms. Finally, L-PNPU consumes the lowest energy of 0.51 uJ/point, even for processing a large number of points compared to previous works.

# 7 CONCLUSION

This paper proposes L-PNPU with the fine-grained LiDAR-SoC pipeline structure for the low-power 3D PNN processor. The azimuthal bin partitioning decreases 90.5% of the operating energy compared to using entire data without partitioning and 76.3% of computation than the previous bin partitioning by balancing the workload. The co-optimization of the modified PNN algorithm and heterogenous architecture with fine-grained pipelining reduces entire system latency and energy by 76.4% and 99.9% compared to the fully serialized CPU+GPU. Moreover, it is the first design that enables real-time operation of the entire system, integrating both LiDAR sensor and processor as well as removing external DARM. As a result, the proposed L-PNPU achieves the highest energy efficiency of 0.51 μJ/point and the shortest system latency after sensing time of 1.21 ms among state-of-the-art processors.

# ACKNOWLEDGMENTS

This work was supported by IITP grant funded by the Korea government(MSIT) (No.RS-2020-II201336, Artificial Intelligence Graduate School Program(UNIST)). The EDA tool was supported by the IC Design Education Center (IDEC), South Korea.

# REFERENCES

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun. 2021. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 12 (2021), 4338-4364.
[2] R. Roriz, J. Cabral and T. Gomes. 2022. Automotive LiDAR Technology: A Survey. *IEEE Transactions on Intelligent Transportation Systems* 23, 7 (2022), 6282-6297.
[3] S. Shi, X. Wang and H. Li. 2019. PointRCNN: 3d object proposal generation and detection from point cloud, In *2019 IEEE/CVF CVPR*, 770-779.
[4] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley and C. Stachniss. 2019. SuMa++: Efficient LiDAR-based Semantic SLAM. In *2019 IEEE/RSJ IROS,* 4530-4537.
[5] H. Thomas, B. Agro, M. Gridseth, J. Zhang and T. D. Barfoot. 2021. Self-Supervised Learning of Lidar Segmentation for Autonomous Indoor Navigation. In *2021 IEEE ICRA*, 14047-14053.
[6] D. Maturana and S. Scherer. 2015. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ IROS*, 922-928.
[7] A. Milioto, I. Vizzo, J. Behley and C. Stachniss. 2019. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. In *2019 IEEE/RSJ IROS*, 4213-4220.
[8] "HDL-64E", Mar. 2018, [online] Available: http://velodynelidar.com/.
[9] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. 2017. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.
[10] Q. Hu et al. 2020. Learning Semantic Segmentation of Large-Scale Point Clouds With Random Sampling. In *2022 IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8338-8354.
[11] S. Kim, J. Lee, D. Im and H. -J. Yoo. 2021. PNNPU: A 11.9 TOPS/W High-speed 3D Point Cloud-based Neural Network Processor with Block-based Point Processing for Regular DRAM Access. In *2021 Symp. on VLSI Circuits*, 1-2.
[12] J. Zheng, H. Jiang, X. Nie, Z. Huang, C. Chen and Q. Liu. 2023. TiPU: A Spatial-Locality-Aware Near-Memory Tile Processing Unit for 3D Point Cloud Neural Network. In *2023 60th ACM/IEEE DAC*, 1-6.
[13] D. Im, D. Han, S. Kang and H. -J. Yoo. 2022. A Pipelined Point Cloud Based Neural Network Processor for 3-D Vision With Large-Scale Max Pooling Layer Prediction. *IEEE Journal of Solid-State Circuits* 57, 2 (2022), 661-670.
[14] Y. Lin, Z. Zhang, H. Tang, H. Wang, and S. Han. 2021. PointAcc: Efficient Point Cloud Accelerator. In *54th Annual MICRO*, 449–461.
[15] X. Yang et al. 2023. An Efficient Accelerator for Point-based and Voxel-based Point Cloud Neural Networks. In *2023 60th ACM/IEEE DAC*, 1-6.
[16] Y. Zhang et al. 2020. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *2020 IEEE/CVF CVPR*, 9601-9610.
[17] X. Zhu et al. 2021. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *2021 IEEE/CVF CVPR,* 9939-9948.
[18] J. Behley et al. 2019. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *2019 IEEE/CVF ICCV*, 9296-9306.
[19] H. Caesar et al. 2020. nuScenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF CVPR*, 11618-11628.