ARTICLE

Check for updates

Journal of
Sensor Science and Technology

The Korean Sensors Society

A Light-weight Asymmetric Multi-Processing FPGA Architecture for Real-time High-resolution LiDAR

Yong Hwi Kim[1], Junseop Lee[1], Sang-gyun Gi[1], Seungjoo Lee[1], and Jihyuk Cho[1,+]

[1]*IT Convergence System Research Center, Korea Electronics Technology Institute, 226 Cheomdangwagi-ro, Buk-gu, Gwangju, 61011, Republic of Korea*

Cite This: *J. Sens. Sci. Technol. Vol. 34, No. 6 (2025) 660-667*

https://doi.org/10.46670/JSST.2025.34.6.660

ABSTRACT: In this paper, we propose a new Asymmetric Multi-Processing (AMP) based architecture for a raster scanning LiDAR system of a high-resolution. In order to prevent a desynchronization problem occurred in the AMP, we propose a process scheduling technique for LiDAR systems based on a state machine architecture, which uses shared memory in FPGAs. We validate our method using a customized Hardware In-the-Loop System (HILS) emulating a Time-of-Flight (ToF) LiDAR. Experimental results show that the proposed AMP architecture effectively reduces execution time in I/O control tasks by 45% compared to a sequential processing without requiring any hardware modifications. We also show that the reduced execution time leads to performance improvements in LiDAR systems such for the time budget of per-point scan or Frame-Per-Second (FPS) rates.

KEYWORDS: *Lidar, FPGA SoC, Lidar system optimization, Asymmetric multi-processing, Time budget optimization*

1. INTRODUCTION

The Light Detection And Ranging (LiDAR) technology detecting surrounding objects and their return properties such as surface reflectivity, and velocity is widely used in various fields including robotics, autonomous vehicles, and manufacturing automation. To achieve millimeter-level range resolution, LiDAR systems require a high-speed Analog-to-Digital Conversion (ADC) of photo-multiplier signals and Digital Signal Processing (DSP) simultaneously. Therefore, processing units in the system must be capable of executing and scheduling a series of range-sensing task from raw signal acquisition to transmission of the system output. In this context, System-on-Chip (SoC) solutions based on Field-Programmable Gate Arrays (FPGAs) have become dominant in commercial LiDAR systems due to their capability for hardware acceleration of high-speed data pipelines [1]. Processing units in a FPGAs can be categorized into two major components: Programmable Logic (PL) in which users can implement application-specific hardware logics, and Processing System (PS) which manages sub-modules of the system via I/O interface.

In LiDAR applications, the PL is dedicated to a Field-of-View (FoV) work, a sequential data pipeline repeatedly occurring in a scan region. It includes laser firing, ADC readout, DSP of raw signals, and memory access to store results. Meanwhile, the PS manages system-level processes controlling IO peripherals to implement system functions followed by the FoV work. System-level processes are also called as the Blind work since they are commonly executed in blind areas after finishing the FoV work.

The performance of a LiDAR systems is highly co-related to the processing design. Assuming that we have a single laser-photodiode LiDAR system with a 2D-raster scanning [2] and frame-per-second (FPS) of the system is predefined as T_o, the maximum time duration (T_p) allocated to a per-point scan process in PL can determined as:

$$T_p = (T_o - T_b) * \frac{\Delta\theta_H}{(\Delta\theta_{H,max} - \Delta\theta_{H,min})} * \frac{\Delta\theta_V}{(\Delta\theta_{V,max} - \Delta\theta_{V,min})}, \quad (1)$$

where $\theta_{H,min}$, $\theta_{H,max}$ are the minimum and maximum horizontal angle in FoV area, $\Delta\theta_H$ is the horizontal angle resolution, and θ_V indicates the vertical angle. By Equation 1, the Blind work assigned to PS reduces the time budget T_p resulting in that either maximum detection range determined by a time-window of photo detector or duration of DSP

+Corresponding author: jcho@keti.re.kr
Received : Oct. 20, 2025, Revised : Oct. 27, 2025, Accepted : Nov. 3, 2025

pipeline is limited. Otherwise, we should sacrifice the angular performance of the system by reducing resolution $\Delta\theta$, or range itself ($\theta_{max}-\theta_{min}$).

In terms of the capability of the LiDAR system, it is essential to optimize T_b. A large amount of literature demonstrate their own system design for LiDAR applications including following examples. Rodrigo et al. [2] demonstrate a coherent detection LiDAR with a raster scanning to adjust periodic motions of drones. Huang et al. [3] propose a low-cost TDC based 1D scanning LiDAR using FPGA. Lee [4] combines a measurement of characteristics of road surface into a LiDAR system. But these methods do not take into account system optimization for T_b. For the system parameter optimization, Goudreault et al. [5] directly optimize DSP parameters in a scanning LiDAR system to improve a 3D detection performance. Li et al. [6] present an optimization method for real-time calibration of crosstalk effects in SPAD array using a FPGA. However, aforementioned methods rather focus on system parameter optimization for downstream task [5], or dedicated hardware logic improving system reliability [6].

In this paper, we propose a new Asymmetric Multi-Processing (AMP)-based FPGA architecture for real-time operation of a high resolution LiDAR system with raster scanning. The proposed method directly reduces T_b by parallelizing I/O peripheral control processes in the Blind work using AMP units, so as to maximize the capability of LiDAR applications.

In comparison of Symmetric Multi-Processing (SMP), where a single operating system (OS) manages all processing cores, an AMP based PS architecture has inherent advantages for process time optimizations [7]. Since each processor core in the AMP is regarded as an independent bare-metal instance with/without an OS, it can excel an concurrent process scheduling without a heavy OS overheads produced by a process scheduler. Instead, AMP architecture can be exposed of race conditions due to asynchronous memory access by multi-cores.

In order to prevent desynchronization problem at system resources, we introduce a LiDAR-state-machine-based process scheduling technique. More specifically, the scheduling of asynchronous threads ran in multi-cores is done by monitoring a current state of the application. For this, we use an On-Chip-Memory (OCM) in FPGAs as a shared memory containing state/action indicators and other global variables, such as mutex, used to prevent race conditions.

In summary, our AMP architecture have two strengths:

(1) Asymmetric threads excel the I/O peripheral controls for LiDAR applications while performing hardware accelerated ADC acquisitions by PL.

(2) Synchronization of asymmetric threads based on a LiDAR state machine enables to implement a light-weight scheduling of LiDAR application processes.

The structure of this paper is as follows. Section 2 discusses an application-level design for a canonical LiDAR system including its state machine. Section 3 describes our AMP-based architecture and asymmetric thread scheduling procedures. Section 4 presents the implementation of a customized FPGA SoC and the Hardware-in-the-Loop System (HILS) for validation of our method. Section 5 analyzes the simulation results obtained through the HILS. And we conclude the paper in Section 6.

2. SYSTEM DESIGN

Fig. 1 illustrates our hardware system design for the LiDAR application, placing emphasis on data stream of processing units in FPGA. The system comprises a FPGA SoC, I/O peripherals including an ADC driver, Laser driver, Scanner driver, and a client receiving system outputs. The FPGA SoC consists of a processing system (PS), programmable logic (PL), I/O interfaces for peripheral control, and memory components (e.g., SRAM, DRAM). In a scanning LiDAR application, the Laser driver, ADC driver, and Scanner driver are procedurally controlled by PL and PS in FPGA chip. For instance, the Laser driver periodically receive a laser trigger at the start of data acquisition time window via high-speed serial interface. Within the time window, ADC driver digitizes analog signals acquired from photodetectors in combination with optical systems. The Scanner driver manipulate the optical path of LiDAR application for transmitting/receiving light under the control of FPGA. Lastly, a client communicates with the FPGA SoC to receive application outputs, and to handle user-command for system parameter adjustments via Ethernet interface.

In the FPGA SoC, PS and PL access an internal memory (SRAM) embedded in the FPGA chip as well as an external

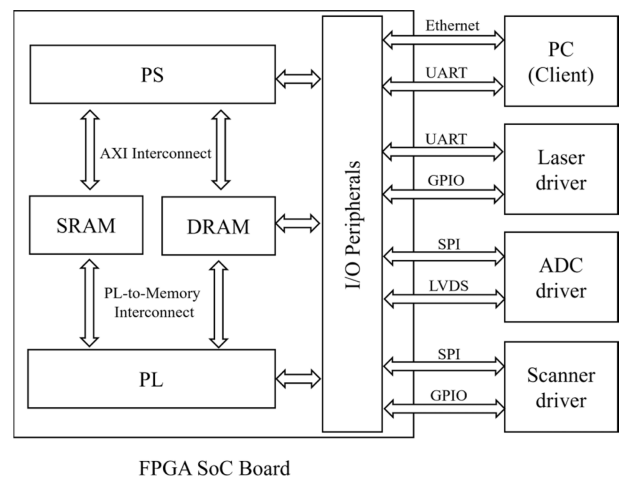

Fig. 1. Hardware system design of LiDAR application

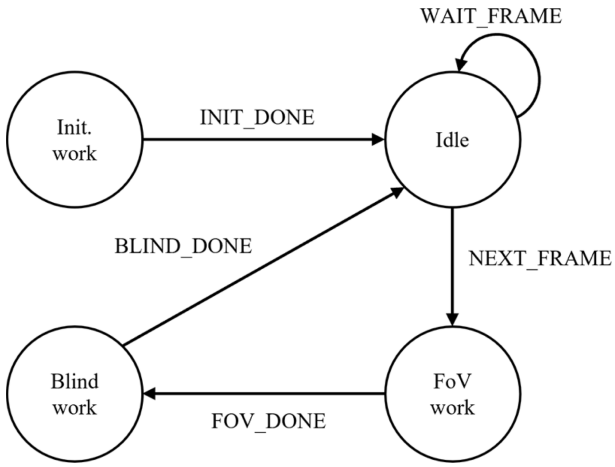

Fig. 2. Lidar state machine. The FoV work state involves ADC acquisition process in a scan area. The Blind work state indicates a post-process handling per-frame management of I/O peripherals of LiDAR applications. The Idle state is periodically repeated until the 'NEXT_FRAME' action occurs

memory (DRAM) additionally mounted on the SoC board. In the DRAM, a Direct Memory Access (DMA) controller manages a high-speed input/output stream through the AXI interconnect and the PL-to-Memory interconnect. PL and PS are synchronized with the DRAM using interrupt signals of the DMA controller. The SRAM, implemented as On-Chip Memory (OCM) within the FPGA, serves as shared memory in the AMP architecture. For the synchronization between PL and PS processes, state flags representing the execution status of each process are used.

Each peripheral connected to the FPGA SoC has two data interfaces. Except for the PC client, the upper interface is linked to PS processes in the Blind work, and the lower interface is dedicated to a high-speed serial communication for the FoV work by PL. The UART port to the PC client is an auxiliary port to debug the application status.

Fig. 2 presents a schematic diagram of the application state machine. The LiDAR process can be mainly divided into three phases: a high-speed data acquisition process that periodically occurs at each scan point in the FoV work, and a system control process, referred to as blind work, after finishing the FoV work and the idle period between consecutive LiDAR output frames.

The LiDAR application begins at the initialization state that prepares all the system components to be ready for application processes. Once all system components are transited to be operational, the application proceeds to the idle state with the 'INIT_DONE' action. In the Idle state, the PL generates the 'NEXT_FRAME' action with a master clock, that indicates the start of the data acquisition process, and the application moves to the FoV work state. During the FoV work state, DMA interrupts are generated upon completion of the data acquisition pipeline for each scan point. The PS monitors the interrupt counter in the SRAM, and it issues the 'FOV_DONE' action when the counter reaches a pre-defined number of scan points. Subsequently, in the Blind work state, the PS manages system components through frame-level feedback control via the I/O interface. Once all Blind work processes are completed, the application transitions back to the Idle state upon receiving the 'BLIND_DONE' action. The 'WAIT_FRAME' action then repeatedly occurs until the next 'NEXT_FRAME' is generated, maintaining the application in the Idle state until the start of the subsequent frame.

3. PROPOSED METHOD

Fig. 3 illustrates a process block diagram of the proposed AMP architecture. As described in Section 2, the application assigns the FoV work to the PL and the Blind work to the PS, based on the state machine. During the FoV work, the PL

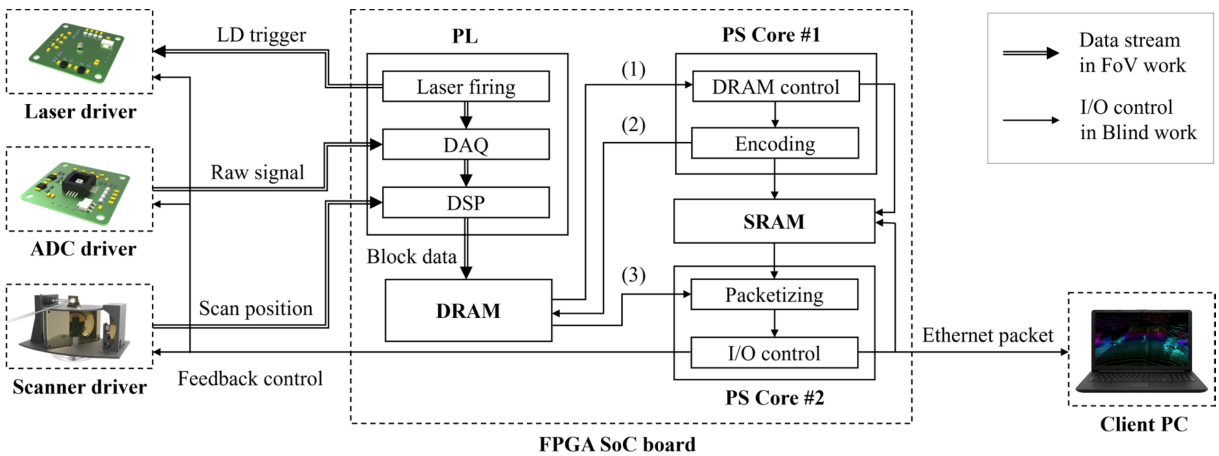

Fig. 3. Process block diagram of our AMP architecture in FPGA SoC

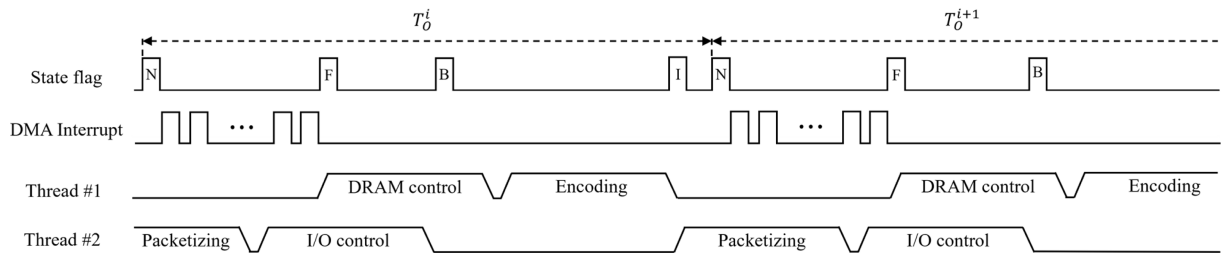

Fig. 4. Asynchronous processing of Blind Work scheduled by State flags

sequentially executes three processes synchronized with a high-speed master clock: Laser firing, Data acquisition (DAQ), and DSP. At the Laser firing process, an LD trigger is given to the Laser driver via serial communication. After a certain time window pre-defined by a system parameter, raw digital signals are acquired from the ADC board. The subsequent DSP process converts them into a block data suitable for DRAM stacking. The DRAM access from PS occurs three times remarked in (1), (2), (3) of Fig. 3. First, the DRAM control process reads stacked block data to interpret them as LiDAR outputs such as range, angles, intensity, and etc. (1), and writes outputs back to the DRAM (2) by the PS Core #1. The PS Core #2 recalls the processed DRAM data to packetize them into fragment of Ethernet packets (3), and sends to the Client PC during the I/O control process.

The proposed AMP architecture executes the four subordinate PS processes in parallel through a dual-thread configuration. The first thread, assigned to Core#1, is responsible for the post-processing of a frame data acquired from the FoV work. This thread manages flash memory in DRAM, and schedules remaining process based on state flags in the SRAM to prevent race conditions between PL and PS cores (DRAM control). Frame data stored in the DRAM is converted into LiDAR outputs during the Encoding process. In parallel, the second thread, assigned to Core#2, performs the Packetizing process, transforming the LiDAR outputs into Ethernet packets, and the I/O Control process, which maintains system stability by performing feedback control of peripheral devices.

Meanwhile, the SRAM is accessed by PS cores for asymmetric synchronization of PS processes. For example, the DRAM control process consistently monitor a DMA interrupt counter indicating whether the FoV work is successfully done. When the DMA interrupt counter reaches to a number of scan points, it update a state flag as the Bind work state in Fig. 2. In a similar manner, the PS Core #1 queues the processes in the PS Core #2. Once the encoding process is done, the PS Core#1 updates another state flag indicating the start of packetizing process in the PS Core#2. And the PS Core#2 informs the PS Core#1 of the end of the Blind work in Fig. 2 using the SRAM. Overall, the PL, PS Core#1, and PS Core#2 run in parallel to reduce the lead time of Blind work T_b.

Since the Blind work processes from the DRAM to I/O control are segmented by two PS cores, we can maximize T_p for per-point scan process in the FoV work. Recalling Equation 1, T_b can be reformulated as a inequality form:

$$T_o > T_f + T_b, \tag{2}$$

where $T_f = T_p * \frac{(\theta_{H,max} - \theta_{H,min})}{\Delta\theta_H} * \frac{(\theta_{V,max} - \theta_{V,min})}{\Delta\theta_V}$.

Equation 2 emphasizes that the sum of the lead time of FoV work and Blind work must not exceed a time gap between consequent frames T_o determined by FPS of LiDAR outputs. Our AMP design potentially extends the length of T_b by parallelizing the PS processes, which leads to reduction in duration time of T_b:

$$T_o > T_f + T_b - \Delta T_b, \tag{3}$$

where ΔT_b is a superposition time between dual cores in PS. Equation 3 shows that the theoretical performance of LiDAR applications can be enhanced by reducing T_b by ΔT_b. For instance, we can increase a scan resolution $\Delta\theta$ without sacrificing T_p, and vice versa. This characteristic is especially necessary for a high-resolution LiDAR [8] with fast scan frequency, or FMCW LiDAR [9] which requires a long acquisition window to capture frequency modulated signals.

Fig. 4 illustrates a process diagram of our AMP architecture, depicting asynchronous processing between two cores during i-th frame time T_o^i. The top line represents the state flag, which indicates the application's operation state according to the state machine shown in Fig. 2. The state flag letters N, B, and F correspond to the actions 'NEXT_FRAME', 'BLIND_DONE', and 'FOV_DONE', respectively. An additional state flag 'I' is used to maintain the synchronization of two Cores. In the second line, the DMA interrupt is generated whenever the PL completes the per-point scan pipeline. Once the DMA interrupt counter reaches to the number of points, the 'F' state flag occurs, and the Core#1 starts the post-processing of frame data. At the end of Core#1 processes, the state flag is changed as 'I', and the Core#2 run remaining processes in parallel. The

'B' state indicates the end of the Blind work so that the application is ready for the next frame process.

4. IMPLEMENTATION

This section presents the implementation of the Hardware-In-the-Loop System (HILS) used to validate the proposed AMP architecture. Fig. 5(a) shows a custom FPGA SoC board, while Fig. 5(b) and 5(c) depict an ADC simulation board and a Scanner simulation board, respectively. The FPGA SoC board, built on the Zynq-7020 platform, features 256 KB of SRAM (OCM) and a dual-core ARM processor operating at 666 MHz, and includes a reference clock generated from a crystal oscillator. In addition to the SRAM, a 4 GB DRAM is mounted on the SoC to handle frame data with a DMA controller. For a client, an RJ45 connector supporting up to 1 Gbps bandwidth Ethernet is included, along with two UART ports, two SPI interfaces, one LVDS link, and several GPIO interfaces. The scanner driver simulation board (Fig. 5(b)) connects to the FPGA SoC via two GPIO ports. It emulates the scanner position feedback based on the control input signals transmitted from the PS through the SPI interface. The emulated position feedback is then transferred to the PL via GPIO.

The ADC simulation board (Fig. 5(c)) emulates the Time-to-Digital Conversion (TDC) signal, which measures the time difference between the laser pulse triggered by the PL reference clock and the corresponding analog pulse signal received by a photodetector. In this simulation, 16-channel TDC signals are sequentially transmitted to the PL via LVDS communication using four TDC groups. Fig. 6 presents the simulation results of the TDC signals generated by the ADC simulation board. Fig. 6(a) shows the block diagram of the TDC signal simulation process, which proceeds as follows:

The PL provides both the reference clock for TDC timing and the PL system clock (LCLKIN) to the simulation board.

When the laser is triggered, the PL generates a 'STOP' signal with a pre-defined time interval to mark the end of the analog signal acquisition.

As illustrated in Fig. 6(b), the ADC simulation board produces a virtual pulse signal at each 'STOP' event, thereby generating a specific distance pattern for each scan point.

The simulated TDC data are transmitted to the PL through a Serial Data Output (SDO) interface, sequentially stored in the BRAM, and then transferred to the DRAM via the DMA controller.

In this simulation, single SDO has a total length of 32 bits, where the upper 12 bits represent the TDC channel index and the remaining 20 bits encode the virtual pulse position. The

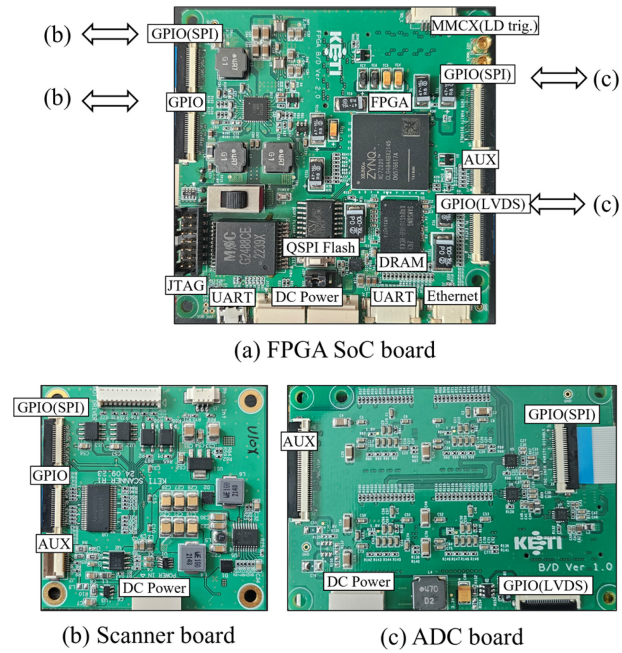

Fig. 5. HILS for LiDAR simulations. (a) FPGA SoC board based on Xilinx's Zynq-7020 (b) Scanner simulation board (c) ADC simulation board

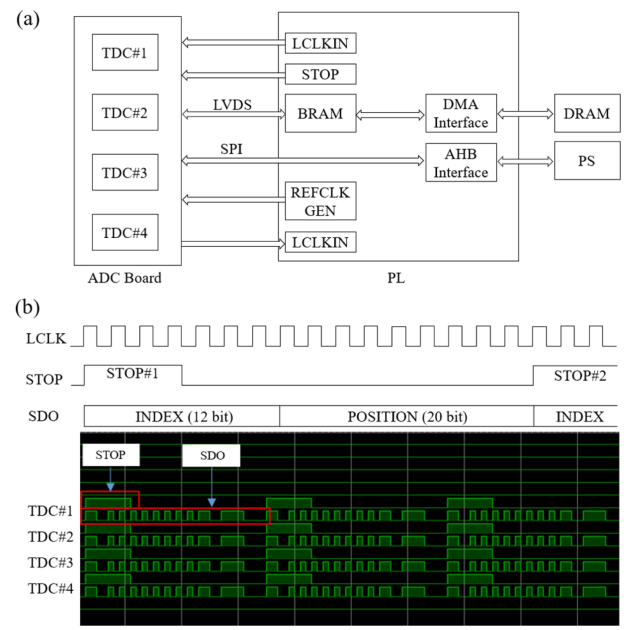

Fig. 6. TDC simulation result. (a) Block diagram of TDC-PL data pipeline (b) TDC test pattern acquired by PL

lower portion of Fig. 6(b) presents the measurement results of the simulated TDC data captured by a dedicated logic analyzer within the PL. As shown, the SDO signals are measured at the end of each per-point scan according to the 'STOP' signal, confirming that the simulated SDO data are consistently captured across all TDC channels.

Table 1. Specifications of simulated LiDAR application

Specification	Value	Unit
Horizontal/Vertical FoV	120/24.8	Deg
Horizontal/Vertical resolution	0.2/0.2	Deg
Number of points	74,400	-
Number of scans	4,650	-
Per-scan time window	14	ms
Frame-per-second	10	Hz
Per-point buffer size	3	Byte
Total buffer size	223,552	Byte
Number of packet fragments	167	-

5. RESULTS

5.1 LiDAR simulation result

This section shows LiDAR simulation results using the HILS in Section 4. Table 1 summarizes specifications of simulated LiDAR applications. Considering a high-resolution LiDAR application with a raster scanning, the horizontal/vertical resolution is set as 0.2/0.2, respectively for 120/24.8 FoV range. Therefore, the number of points is determined as 120/0.2*24.8/0.2 = 74,400. In our simulation, we use a 4 TDC groups in which each group is consisted of 4-channel TDC arrays (Fig. 5(c)), to parallelize the vertical acquisition process. Therefore, the number of scans is reduced as 74,400/16 = 4,650. An empirically measured per-scan time window is used in this simulation (14.181 micro seconds per scan). We set the FPS of LiDAR output transmitted through 1Gbps Ethernet as 10 Hz, so that $T_f+T_b-\Delta T_b<0.1$ must be satisfied. By the Encoding process in PS Core#1, a 32 bit TDC value per-point is translated into total 3 Byte information (2Byte for range, 1Byte for intensity). Then, the total buffer size is determined as 218 KB including 352 Byte header. We consider the packet fragment data size as 1344. The number of fragments is calculate as 223,552/1344 ≈ 167. We note that the angle information in H./V. scan is natively included in the index of point information with an assumption that the point index is invariant along scan frames.

In our simulation, we implement two types of PS architecture at the same HW platform to demonstrate the effect of PS multi-processing. A first architecture only use single core for the sequential Blind process including DRAM control, Encoding, Packetizing, and I/O control in Fig. 4. Second, we implement our AMP architecture in the same platform described in Fig. 5. Table 2 shows the FoV/Blind work execution time in both cases. The execution time of the sequential processing in Blind work is measured as 0.056 seconds in total (0.012s for DRAM control, 0.015s for Encoding, 0.003s for Packetizing, and 0.026s for I/O control). Compared to the sequential processing, the AMP architecture reduces it as 0.031 seconds, achieving 45% accelerations for the Blind work processes. The FoV execution time is identical in both cases, as the same platform is used.

Table 2. FoV/Blind work execution time in a sequential processing vs. dual processing using the proposed AMP architecture in Blind work

Task	Sequential processing (s)	AMP architecture (s)
FoV work	0.065	0.065
Blind work	0.056	0.031

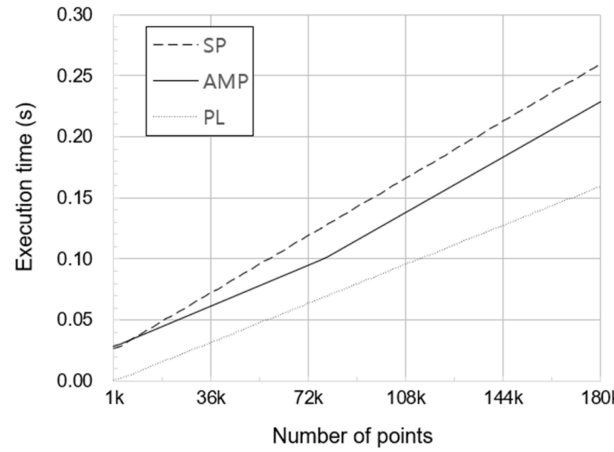

Fig. 7. Total execution time of Sequential processing (SP) and AMP architecture (AMP)

The sequential processing does not meet the condition described in Equation 2. T_f+T_b is measured as 0.121 s per-frame, but it exceeds the pre-defined FPS ($T_o<0.1$). In our result, there exists a frame-drop effect in LiDAR outputs intermittently. In contrast, the AMP architecture successfully operates the entire processes in FoV/Blind work without the frame-drop effect because $T_f+T_b-\Delta T_b$ in Equation 3 does not exceed the T_o limit.

Fig. 7 depicts the total execution time of T_f+T_b for both the Sequential processing (SP) and the AMP architecture (AMP) according to the number of points in the HILS. In this graph, we additionally measure the execution time for the FoV work done by PL. As shown in the graph, execution time of both processing methods are linearly increased to the number of points. Our AMP architecture marginally reduces the total execution time compared to the SP. The gap between the SP and the AMP is monotonically increased while the

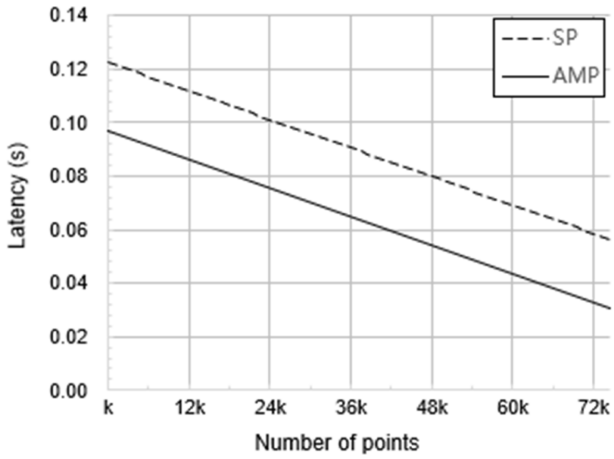

Fig. 8. Latency at each scan point for SP and AMP

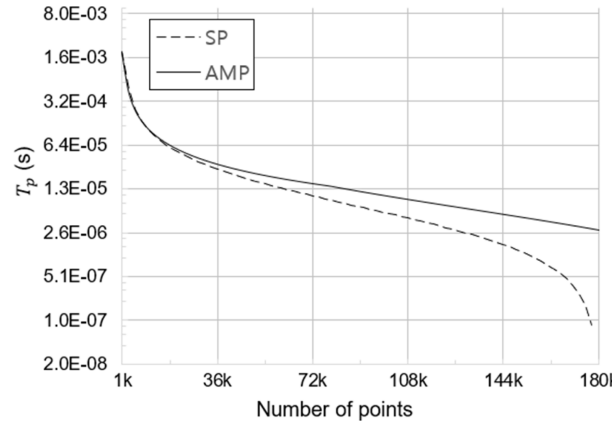

Fig. 9. Maximum time budget according to the number of points while fixing remaining system parameters in Table 1

number of points reaches at 72k. T_f of PL is also proportional to the number of points.

From the perspective of system optimization, processing time is prioritized for the PL. Therefore, lower T_b is better to generate an opportunity for enhancement of the capability of LiDAR applications as shown in Fig. 8. This figure presents the latency measured at each scan point for both the SP and AMP architectures. In a raster-scanning LiDAR application, latency gradually increases along the scan path due to sequential point acquisition. The AMP architecture employs an identical PL pipeline resulting in a uniform latency reduction by ΔT_b.

5.2 System parameter optimization with reduction in T_b

This section shows that we can enhance the capability of a LiDAR application in terms of the maximum time budget $T_{p,max}$ in Equation 1 as well as the maximum FPS $T_{o,max}$ using the HILS. In this result, we calculate the theoretical maximum value of $T_{p,max}$, $T_{o,max}$ from the fixed $T_o = 0.1$ and estimated T_b of two PS architectures. Other system parameters listed in Table 1 remain constant.

Fig. 8 shows the simulation result for $T_{p,max}$ according the number of points. In a low point region (1k ~ 36k), the AMP does not shows a meaningful improvement on T_p. However, For the large point region (36k ~ 180k), the AMP is beneficial to sustain T_p, while T_p of PS drastically converges into 0 in the large point region.

Similarly Fig. 9 shows the simulation result for $T_{o,max}$ with the constant T_p value in Table 1. For a small number of points (<5k), the SP outperforms the AMP due to scheduling overheads for PS cores. However, in the remaining regions (>5k), the AMP outperforms the SP, and this tendency becomes more pronounced in the intermediate

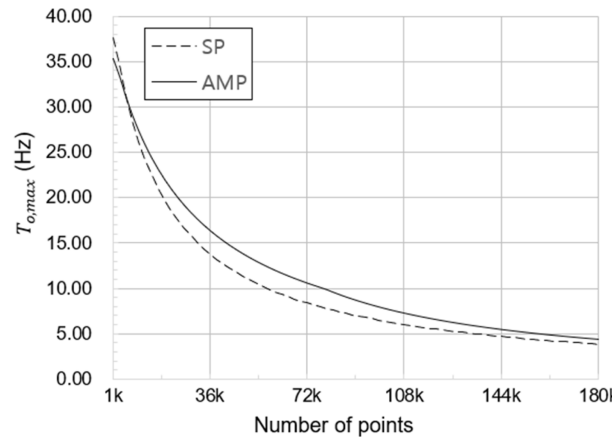

Fig. 10. Maximum FPS according to the number of points while fixing remaining system parameters in Table 1

range (30k to 80k). Over 100k regions, the improvement of FPS is monotonically reduced since the portion of T_f in Equation 2 increases with the number of points as shown in Fig. 7.

6. CONCLUSIONS

In this paper, we have shown that our AMP architecture effectively reduces the execution time of the Blind work processes, and also this reduction leads to an improvement of the capability of LiDAR applications with system design parameters listed in Table 1. In order to design the scheduling of asynchronous processing in the AMP architecture, we have demonstrated a LiDAR state machine-based process scheduling technique using the internal memory unit, OCM, as a shared memory for PL, and PS cores.

For the validation, we have implemented a custom HILS as described in Section 4. Using the HILS, we simulate a raster scanning LiDAR of a high-resolution (70k points per frame)

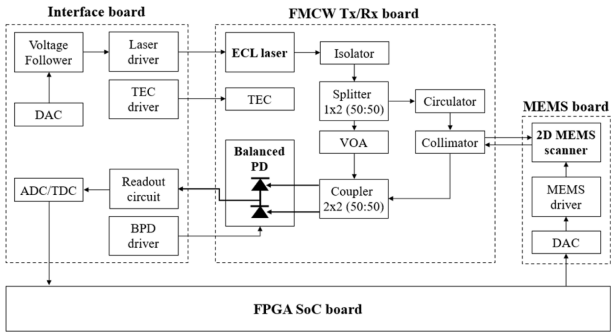

Fig. 11. Block diagram of a FMCW LiDAR system using the FPGA SoC board

with the 4-channel ADC board and the scanning board emulating position feedback according to a register setting. In the simulation result, our AMP architecture effectively reduces the execution time in the Blind work processes by 45% compared to the sequential processing regarded as a baseline method. This reduction can extend the length of the maximum time budget of per-point scan by the PL. In Fig. 8, we have shown the trend of the time budged according to the number of points, resulting that the AMP drastically help to sustain the time budget shrunk as points increases.

For future work, we would like to develop a Frequency-Modulated-Continuous-Wave (FMCW) LiDAR system using our FPGA SoC board. As shown in Fig. 11, This system consists of three submodules: Interface board sending/receiving Laser driving/Readout signals synchronized by PL in FPGA SoC board; FMCW Tx/Rx board extracting a beat signal using a Balanced PD with an optical mixer; and a Micro-Electro-Mechanical System (MEMS) board steering optical path of LiDAR system. Since the FMCW LiDAR inherently demands a relatively long T_p due to chirp signal modulation and DSP in frequency domain, the proposed method can provide advantages in management of an overall FoV processing time.

We also would investigate an advanced optimization method minimizing the Blind work process overheads by adopting additional threading technique such as speculative multi-processing on the PS.

CRediT Authorship Contribution Statement

Yong Hwi Kim: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft. **Junseop Lee:** Validation, Visualization, Software. **Sang-gyun Gi:** Validation. **Seungjoo Lee:** Funding acquisition, Project administration, Resources, Supervision. **Jihyuk Cho:** Conceptualization, Resources, Supervision, Writing – review and editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Technology Innovation Program (RS-2023-00234343, Development of key photonic components and micro photonic integrated circuit module to commercialize high-resolution 4D FMCW MEMS LiDAR for autonomous vehicles) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) and Korea Planning & Evaluation Institute of Industrial Technology (KEIT).

REFERENCES

[1] A.O. Korkan, H. Yuksel, A novel time-to-amplitude converter and a low-cost wide dynamic range FPGA TDC for LiDAR application, IEEE Trans. Instrum. Meas. 71 (2022) 1–15.
[2] P.J. Rodrigo, H.E. Larsen, C. Pedersen, CW coherent detection lidar for micro-Doppler sensing and raster-scan imaging of drones, Opt. Express 31 (2023) 7398–7412.
[3] J. Huang, S. Ran, W. Wei, Q. Yu, Digital integration of LiDAR system implemented in a low-cost FPGA, Symmetry 14 (2022) 1256.
[4] S. Lee, LiDAR Measurement Analysis in Range Domain, J. Sens. Sci. Technol. 33 (2024) 187–195.
[5] F. Goudreault, D. Scheuble, M. Bijelic, N. Robidoux, F. Heide, LiDAR-in-the-loop Hyperparameter Optimization, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2023), Vancouver, Canada, 2023, pp. 13404–13414.
[6] D. Li, R. Ma, X. Wang, J. Hu, Z. Zhu, Optimization of system design and calibration algorithm for SPAD-based LiDAR imager, IEEE Trans. Instrum. Meas. 71 (2022) 1–10.
[7] R.E. Grant, A. Afsahi, Power-performance efficiency of asymmetric multiprocessors for multi-threaded scientific applications, Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes, Greece, 2006, pp. 1–8.
[8] P. Padmanabhan, C. Zhang, E. Charbon, Modeling and analysis of a direct time-of-flight sensor architecture for LiDAR applications, Sensors 19 (2019) 5464.
[9] M. Hauser, M. Hofbauer, FPGA-based EO-PLL with repetitive control for highly linear laser frequency tuning in FMCW LIDAR applications, IEEE Photonics J. 14 (2021) 1–8.