Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 20(3): 166-173, Sep. 2022 Regular paper

An Edge AI Device based Intelligent Transportation System

Youngwoo Jeong, Hyun Woo Oh, Soohee Kim, and Seung Eun Lee* ***Member*, *KIICE***

Department of Electronic Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea

Abstract

Recently, studies have been conducted on intelligent transportation systems (ITS) that provide safety and convenience to humans. Systems that compose the ITS adopt architectures that applied the cloud computing which consists of a high-performance general-purpose processor or graphics processing unit. However, an architecture that only used the cloud computing requires a high network bandwidth and consumes much power. Therefore, applying edge computing to ITS is essential for solving these problems. In this paper, we propose an edge artificial intelligence (AI) device based ITS. Edge AI which is applicable to various systems in ITS has been applied to license plate recognition. We implemented edge AI on a field-programmable gate array (FPGA). The accuracy of the edge AI for license plate recognition was 0.94. Finally, we synthesized the edge AI logic with Magnachip/Hynix 180nm CMOS technology and the power consumption measured using the Synopsys's design compiler tool was 482.583mW.

Index Terms: Edge AI device, Embedded system, Intelligent transportation system, License plate detection, Character recognition

I. INTRODUCTION

An intelligent transportation system (ITS) is a next-generation transportation system that consists of advanced public transportation, traffic management and automatic parking management systems. The key feature of ITS is that the system has to handle a vast amount of data collected from various roads in real-time [1]. Accordingly, many data centers based on cloud computing with high-performance general-purpose processors and graphics processing units are required to store and process the data [2-3]. In addition, because edge devices send raw data to data centers, high network bandwidth is essential [2]. These characteristics may cause the system to decrease power efficiency, increase processing time, and weaken security.

To overcome these weaknesses, utilizing the concept of edge computing is one of the solutions [4-5]. Edge computing is a distributed computing paradigm that processes data locally and sends the processed data to the central data center. This method not only reduces the amount of data exchanged between the data centers and the edge device, but also improves the security of the systems because the original data is not transmitted to the data centers. Generally, edge devices have limited power resources and areas owing to the features of the environment in which the devices are installed [6]. Therefore, edge devices should be applied to an embedded system that reduces resource wastage by performing certain functions to derive the maximum performance.

In this paper, we propose an edge AI device based ITS. The main contribution of this paper is to increase the power efficiency and security compared to conventional ITS by applying edge AI devices. As one of the necessary elements for implementing ITS is to identify the position and speed of the vehicle in a real-time environment, we applied an edge

Received 14 April 2022, **Revised** 30 May 2022, **Accepted** 07 July 2022
***Corresponding Author** Seung Eun Lee (E-mail: seung.lee@seoultech.ac.kr, Tel: +82-2-970-9021)
Department of Electronic Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea

Open Access https://doi.org/10.56977/jicce.2022.20.3.166 print ISSN: 2234-8255 online ISSN: 2234-8883

This paper is an extension of work, originally reported in Proceedings of the 14th International Conference on Future Information & Communication Engineering 2022.

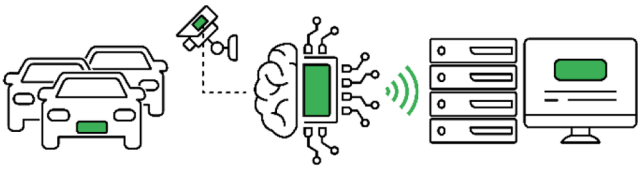

Fig. 1. The concept of the edge AI based ITS.

AI device for license plate recognition in various systems. Figure 1 represents the concept of edge AI based ITS. Accordingly, ITS is possible to collect vehicle information automatically while operating 24h a day with low maintenance costs. We optimized the edge AI device for the license plate recognition by simulation to improve accuracy, as the proposed edge AI device consists of a reconfigurable AI feature [7]. Moreover, we measured the device utili-zation of edge AI on a field-programmable gate array (FPGA) to analyze the total logic elements and total memory bits. Finally, we synthesized the edge AI with Magnachip/Hynix 180 nm CMOS technology and measured the power consumption of the edge AI utilizing Synopsys's design compiler tool of Synopsys. As a result, we verified that edge AI is suitable for ITS.

The remainder of this paper is organized as follows. Section 2 discusses related work on license plate recognition. Section 3 shows the system overview, system architecture, and flow of image preprocessing. Section 4 presents experimental results. Section 5 summarizes the proposed method and experimental results.

II. RELATED WORK

A. Embedded System

The study in [8] presented license plate recognition based on YOLOv4. The experiment was conducted on images with 256 × 256 and 608×608 resolutions with the NVIDIA Jetson TX2 board. The accuracy for over 3000 license plate images was 0.983 and 0.981, respectively. Additionally, the frame per second (FPS) was 9.5, and 2.1. On the one hand, the study in [9] proposed FPGA-based Optical Character Recognition system (OCR) which is designed and tested with imperfect and noisy license plate images. The OCR system is based on a feedforward neural network, that uses efficient and precise neurons. The neuron transfer function is based on an approximation of the hyperbolic tangent activation function. The logic element (LE) of the proposed system is 13,909, and the total number of memory bits is 3,519,000. The accuracy of the dataset consisting of 665 characters is 0.982. An FPGA accelerator based on a convolutional neural network (CNN) for license plate recognition was proposed in [10]. Grayscale processing, binarization processing, and threshold settings were applied to reduce the number of parameters. The LE of the proposed CNN accelerator was 49,088, and Flip-Flop (FF) was 54,047. The proposed CNN accelerator is deployed on an Artix-7 development board. The accuracy was 0.987 and the recognition time was 0.21s. [11] proposed a system that detected license plate with Tiny YOLOv3 and identified the characters using an ensemble of CNN models. The proposed system was implemented on a Raspberry Pi3 and showed an average time of 4.88*s* to process an image with 1024×768 resolution. The accuracy of the proposed system is 0.995. The power consumption was measured at 3.12W when processing uninterruptedly.

B. High-performance Processors

The study in [12] performed license plate detection using an Improved Bernsen Algorithm and Connected Component Analysis models, character segmentation with optimal-K means clustering, and character recognition with a CNN. The proposed model was accelerated by applying it to a PC with configurations such as i5, 8th generation, and 16 GB RAM. It achieved a maximum overall accuracy of 0.981 on the Stanford Cars dataset. License plate detection and character segmentation using YOLOv3 were proposed in [13]. To apply YOLOv3 to more complex scenes, the last number of layers was changed in the proposed method. Character recognition was performed using a convolutional recurrent neural network (CRNN) designed for text recognition. The accuracy of the proposed method was 0.961 for approximately 2049 images with NVIDIA 1080Ti. The time required to process the license plate detection, character segmentation, and character recognition were 10.211*ms*, 2.31*ms*, and 1.59*ms* each.

III. SYSTEM MODEL AND METHODS

A. System Overview

The system flow for license plate recognition is divided into three stages: license plate detection, character segmentation, and character recognition. In general, complex image processing algorithms that cause long latency are employed for license plate detection and character recognition in the license plate recognition process. However, we reduced the processing time by reducing the character recognition algorithm by employing the characteristics of license plates. We applied a k-Nearest Neighbor (k-NN) based edge AI to the character recognition and edge AI performs training and recognition. The edge AI based on structureless k-NN to overcome memory limitations and reduce computational complexity [7]. The system flow of the license plate recognition designed by workload analysis is shown in Fig. 2. The

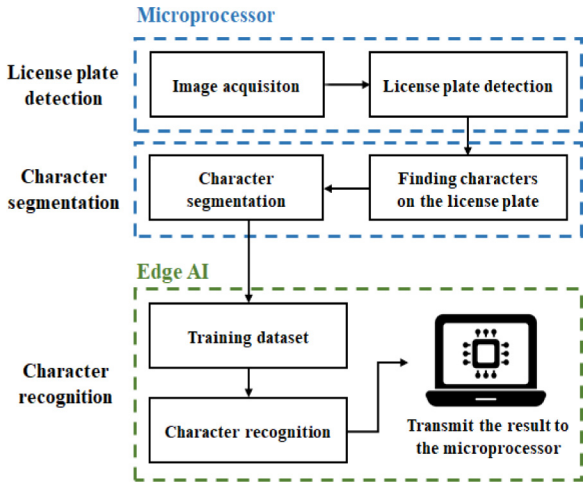

Fig. 2. The system flow of the license plate recognition with the edge AI device.

license plate recognition begins when the edge AI device receives the entire image through a camera module. A microprocessor of the edge AI device detects the license plate on the entire image and segments the characters on the license plate by utilizing OpenCV-Python. The segmented characters are transmitted to the edge AI which is employed for the character recognition. The edge AI compares the recognition data with training data and outputs a result. In order to increase the accuracy, we utilized the format of license plates. Currently, Korean license plates have seven or eight number of characters. The license plates with seven characters have Hangul at the 3rd digit and the license plates with eight characters have Hangul at the 4th digit. According to the format, numbers are trained only when comparing numbers and Hangul is trained only when comparing Hangul. This increases the accuracy because the edge AI does not confuse Hangul with numbers.

B. System Architecture

Figure 3 shows a block diagram of the ITS with the edge AI device. The edge AI device consists of a microprocessor and edge AI [14]. The microprocessor includes a camera module to collect the image data, a system core that performs image preprocessing, and a serial peripheral interface (SPI) for data communication with edge AI. Edge AI consists of a SPI interface for communication with a microprocessor and an AI processor for training and recognition. The proposed system architecture increases the security of the ITS because the original data are not restored based on the resulting data. The interface of edge AI receives external signals from the system core and forwards the results processed by the AI processor to the system core. Modules at the interface interpret external signals composed of training or recognition data and command signals. The command signal consists of 0x60 which is a learning command, 0x40 which is a recognition of 0x40, and 0x83 and 0x84 for receiving the results. The instruction decoder decodes the data and sends command signals to the neuron core. The neuron core includes a scheduler and Neuron cells (N-cell). The N-cells are registers that store the training data. Since the sizes and number of N-cell are variable, the edge AI is optimized according to the application to which the AI processor is applied. Depending on the command types, the scheduler controls the N-cell. In the training process, the scheduler sequentially inputs the training data into the N-cell from the first one. The scheduler counts how many N-cells have been trained each time when the scheduler performs the training command. When all the existing N-cells have been trained, the scheduler sends a signal that the N-cell is full out of the neuron core. During the recognition process, the scheduler controls all N-cells to compare the recognition data with the training data. The types of comparative compu-

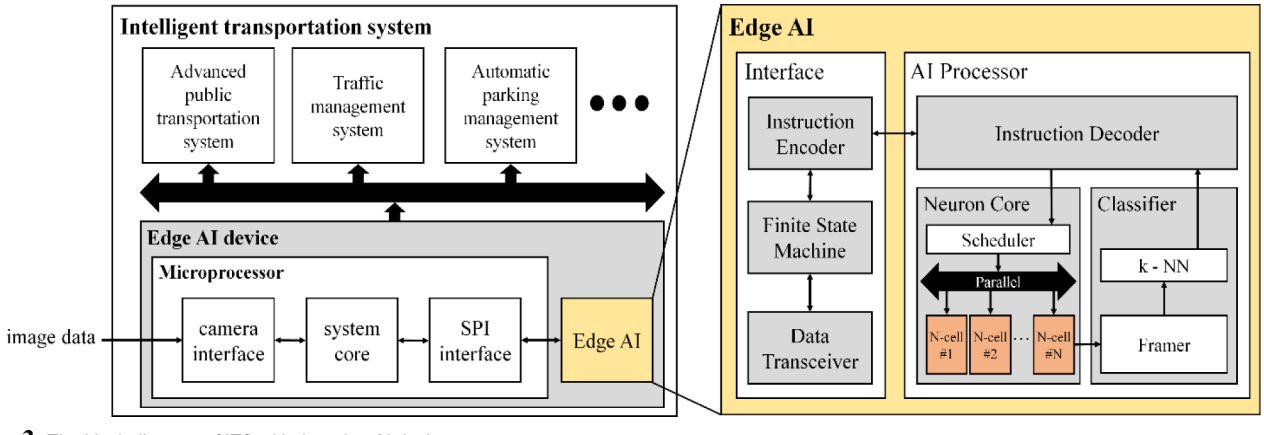

Fig. 3. The block diagram of ITS with the edge AI device.

tation are divided into Manhattan distance calculation and Euclidean distance calculation. Equations 1 and 2 represent the Manhattan distance calculation and Euclidean distance calculation, respectively, which are the methods of comparative operations. Equation 1 requires only the subtractor, but Equation 2 requires the subtractor and multiplier. The processing time of the multiplier is longer than the processing time of the subtractor, and the logic size of the multiplier is larger than that of the subtractor. Accordingly, the multipliers are not suitable for edge AI which is applied to embedded systems. Therefore, the Manhattan distance calculation is appropriate for edge AI.

$$d_1 = |a_1 - b_1| + |a_2 - b_2| \quad (1)$$

$$d_2 = \sqrt{(a_1 - a_2)^2} + \sqrt{(b_1 - b_2)^2} \quad (2)$$

After the comparison process, each N-cell outputs a category and a distance. A classifier aggregates these results and outputs the results through the k-NN. The results are transmitted to the system core in reverse order of the data coming in.

Python library code is utilized between the microprocessor and the edge AI device to communicate with the SPI protocol. A function for the learning process compares whether the size of the learning data is the same as that of the N-cell. Subsequently, the microprocessor transmits 0x60 which is the learning command, the length of the learning data expressed in 16 bits, the learning data, and the learning category to edge AI in order. Similarly, the function of the recognition process compares whether the size of the recognition data is the same as the size as that of the N-cell. Afterwards, 0x40 which is the recognition command, the recognition data, 0x83 and 0x84 which are the commands to read the results, are transmitted to receive the result data distance and category.

C. Image Preprocessing

Figure 4 shows the entire process of image preprocessing. To decrease the latency of image preprocessing, the microprocessor performs grayscale conversion which reduces the amount of data that causes performance degradation. After that, the microprocessor applied a Gaussian filter with a 3 × 3 kernel size to the image and performed threshold processing to clarify the image. The microprocessor finds contours and draws rectangles about the contours like Figure 4(b) and Figure 4(c) based on the clarified image. Figure 4(d) shows the rectangles that can be the characters on the license plate by comparing the size, location, and arrangement of the rectangles. The sizes of the license plates of the two vehicles were different from this perspective. The proposed license plate detection method only recognizes the license plate of the vehicle in front. By analyzing the angle of the detected license plate, the microprocessor rotates the image as if it were taken from the front like Figure 4(e). In order to segment the characters from the license plate, the characteristic of Hangul has to be consi-dered which are different from English and numbers. As Hangul is a combina-

The Python library code: SPI communication between the microprocessor and the edge AI device.

```
import spidev as spi
spi = spi.SpiDev()
spi.open(0, 0)
spi.mode = 3
spi.max_speed_hz = 6000000
learning_command = 0x60
recognition_command = 0x40
read_dist = 0x83
read_cat = 0x84
def learning(train_data, cat):
    data_len = len(train_data)
    if (data_len != SIZE_OF_N_CELL):
        print('vector size does not match')
    else:
        len_cmd1 = (data_len - 1) >> 8
        len_cmd2 = (data_len - 1) & 0xFF
        spi.xfer([learning_command])
        spi.xfer([len_cmd1])
        spi.xfer([len_cmd2])
        spi.xfer(train_data)
        spi.xfer([cat])
def recognition(test_data):
    data_len = len(test_data)
    if (data_len != SIZE_OF_N_CELL):
        print('vector size does not match')
    else:
        len_cmd1 = (data_len - 1) >> 8
        len_cmd2 = (data_len - 1) & 0xFF
        spi.xfer([recognition_command])
        spi.xfer([len_cmd1])
        spi.xfer([len_cmd2])
        spi.xfer(test_data)
        spi.xfer([read_dist])
        spi.xfer([0])
        distance1 = spi.xfer([0])
        distance2 = spi.xfer([0])
        distance = (distance1[0] << 8) | distance2[0]
        spi.xfer([read_cat])
        spi.xfer([0])
        category = spi.xfer([0])
        category = (category[0] << 8) + spi.xfer([0])[0]
```

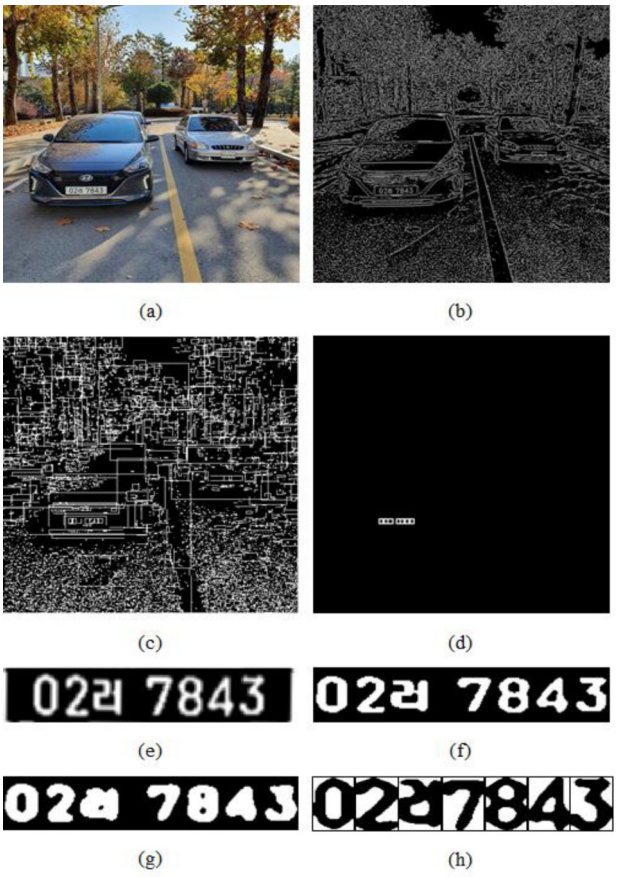

Fig. 4. Entire process of image preprocessing.

tion of consonants and vowels, there are cases in which consonants and vowels are separated. Therefore, when contours are found in this case, they are searched separately. To solve this problem, the following steps have to be implemented. Initially, the microprocessor equalizes the size of the extracted license plate images. Second, the microprocessor processes a Gaussian filter with a large kernel size for the image to segment the separated characters into one character. These processes cause the consonants and vowels to be always attached even if the microprocessor processes the same Gaussian filter. The microprocessor finds the contours and draws rectangles about the contours again, which is a previously performed task. Based on these rectangles, the microprocessor segments the characters one by one in the image using a Gaussian filter with a small kernel size. Figure 4(f) shows an image with a Gaussian filter with a small kernel size of 3×3, and Figure 4(g) shows an image with a Gaussian filter with a large kernel size. The segmented characters have to be transmitted to the edge AI sequentially. Therefore, the microprocessor sorts the characters based on the x-value of the rectangles, crops the characters and resizes the images. At last, the microprocessor processes a Gaussian filter with a large kernel size on the image and performs threshold processing on the image to make the characters clear, making it similar to the training data. Figure 4(h) shows the image of each character image transmitted to the edge AI. The proposed image preprocessing method is feasible for application to license plates with complex surroundings or license plates from other countries.

IV. RESULTS

The size of the entire image affects the detection of license plates during image preprocessing. Considering the performance of the camera used in the edge AI device, the image sizes to be tested were determined. Figure 4 shows the accuracy of detecting license plates when the number of data points is 500. The results showed that the maximum accuracy was 0.994 with 720 × 720 resolution images.

We also analyzed the accuracy of license plate recognition according to changes in the size and number of N-cells. As the license plate consisted of a combination of 10 numbers and 40 Hangul, the number of N-cells was fixed at 64. The size of the N-cell is the size of the character images that are segmented through image preprocessing and transmitted to the edge AI. Therefore, the size of the N-cell affects the accuracy of the edge AI for the characters. We simulated various N-cell sizes to optimize the edge AI for license plate recognition. After the characters were segmented by image preprocessing, the number of numeric data was 3024 and the number of Hangul data was 497. We set different kernel sizes for the Gaussian filter for each character image on the license plate and analyzed the maximum accuracy of each character image. Table 1 shows the simulation results for various N-cell sizes of the edge AI. The accuracy for numbers was about 0.99, but the accuracy for Hangul showed a significant difference. The highest accuracy of the edge AI device for the license plate recognition was 0.94 when the size of N-cell was 32 × 64. Additionally, the size of the N-

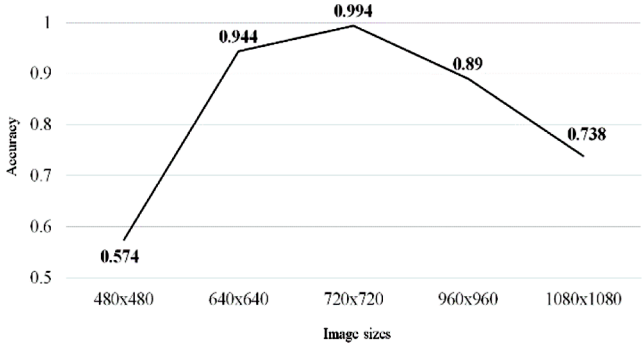

Fig. 5. The accuracy of license plate detection according to the image sizes.

Table 1. Figures of simulation results.

N-cell size	Number accuracy	Hangul accuracy	Total accuracy	License plate detection time	Character segmentation time	Character recognition time	Processing time	FPS
8 × 16	0.985	0.734	0.678	30.406*ms*	0.61*ms*	9.926*ms*	40.91*ms*	24.443
16 × 32	0.992	0.963	0.928	28.544*ms*	0.612*ms*	13.566*ms*	42.61*ms*	23.468
32 × 64	0.992	0.981	0.94	25.394*ms*	0.599*ms*	19.872*ms*	45.865*ms*	21.803
64 × 128	0.991	0.971	0.93	28.087*ms*	0.606*ms*	47.548*ms*	81.695*ms*	12.24
128 × 256	0.991	0.965	0.922	39.538*ms*	0.702*ms*	207.746*ms*	247.986*ms*	4.032

Simulated with Intel core i5-4590 CPU and 4GB RAM

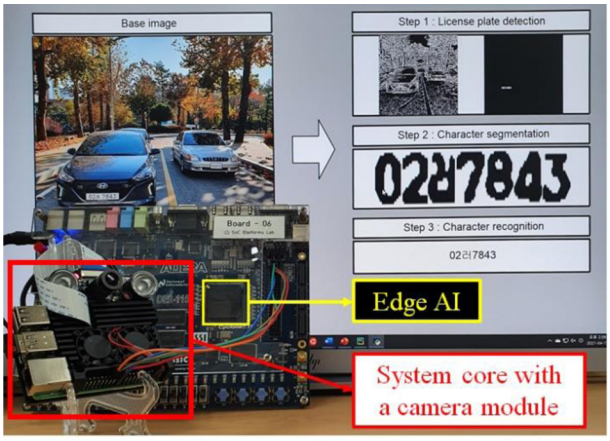

Fig. 6. Experimental environment.

Table 2. Summary of resource utilization.

N-cell spec	Total logic elements	Total memory bits
128 × 64	13,479 / 114,480 (12%)	65,536 / 3,981,312 (2%)
512 × 64	13,995 / 114,480 (12%)	65,536 / 3,981,312 (2%)
2048 × 64	15,345 / 114,480 (13%)	1,048,576 / 3,981,312 (26%)
8192 × 64	15,770 / 114,480 (14%)	8,388,608 / 3,981,312 (211%)

cell does not significantly increase the processing time.

We implemented edge AI logic on the DE2-115 board. Table 2 shows the resource utilization of edge AI. The memory bits of the edge AI exceeded when the size and number of N-cell were 8192 × 64 and 32768 × 64, respectively. Considering the accuracy and device utilization, we fixed the size and number of N-cell to 2048 × 64. The number of logic elements was 15,345. We synthesized edge AI logic with Magnachip/Hynix 180 nm CMOS technology. The total dynamic power measured using the Synopsys's design compiler tool was 482.583 mW when the clock frequency was 50MHz. Based on the experimental results, it can be concluded that the proposed edge AI device is appropriate for license plate recognition.

Figure 6 shows the experimental environment. We connected the microprocessor, Raspberry Pi 4, and the field programmable gate array (FPGA) board, DE2-115, through GPIO. At the start of the experiment, the microprocessor performed the image preprocessing and transmitted the data to the FPGA. Edge AI processed the data and transmitted the recognition result to the microprocessor. The accuracy was the same as that of the software simulation results, and the FPS was 2.01. The increase in processing time was due to the difference in clock frequency and CPU performance. Comparing the experimental results of the edge AI device with the figures of license plate recognition for embedded systems described in the related work, the accuracy was similar while FPS and power consumption were lower. Finally, we simulated the alphabet to analyze the applicability of the edge AI device to English license plates, and the accuracy was 0.904 with a 32×64 resolution.

V. DISCUSSION AND CONCLUSIONS

In this paper, we proposed an edge AI device based ITS. An edge AI device was employed for license plate recognition to increase the power efficiency and decrease the processing time of ITS. We optimized the edge AI for license plate recognition by simulation for the various sizes of N-cells to increase accuracy. When the size of the N-cell was 32 × 64, the accuracy was the highest at 0.94, and the FPS was 21.803. Additionally, we implemented edge AI logic on the FPGA and analyzed the total logic elements and total memory bits. When the size and number of N-cell were 32×64 and 64, respectively, the total number of logic elements was 15,345, and the total number of memory bits was 1,084,576. We synthesized the edge AI with Magnachip/Hynix 180nm CMOS technology and the dynamic power consumption measured using the Synopsys's design compiler tool was 482.583mW. The total accuracy of the hardware verification was the same as that of the software simulation results.

ACKNOWLEDGMENTS

This study was supported by the SeoulTech (Seoul National University of Science and Technology).

REFERENCES

[1] K. Gharehbaghi and M. Myers, "Intelligent system intricacies: Safety, security and risk management apprehensions of ITS," in *8th International Conference on Industrial Technology and Management (ICITM)*, Cambridge, United Kingdom, pp. 37-40, 2019. DOI: 10.1109/ICITM.2019.8710708.

[2] B. Chang and J. Chiou, "Cloud computing-based analyses to predict vehicle driving shockwave for active safe driving in intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 852-866, Feb. 2020. DOI: 10.1109/TITS.2019.2902529.

[3] D. Li, L. Deng, Z. Cai, and X. Yao, "Notice of retraction: intelligent transportation system in macao based on deep self-coding learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3253-3260, Jul. 2018. DOI: 10.1109/TII.2018.2810291.

[4] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1380-1392, 2021. DOI: 10.1109/TMC.2020.2965929.

[5] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1840-1852, Mar. 2021. DOI: 10.1109/TITS.2020.3025687.

[6] M. Gusev and S. Dustdar, "Going back to the roots—The evolution of edge computing, an iot perspective," *IEEE Internet Computing*, vol. 22, no. 2, pp. 5-15, Mar./Apr. 2018. DOI: 10.1109/MIC.2018.022021657.

[7] D. H. Hwang, C. Y. Han, H. W. Oh, and S. E. Lee, "ASimOV: A framework for simulation and optimization of an embedded AI accelerator," *Micromachines*, vol. 12, vol. 7, pp. 838, Jul. 2021. DOI: 10.3390/mi12070838.

[8] J. -Y. Sung, S. -B. Yu, and S. -h. P, "Real-time automatic license plate recognition system using YOLOv4," in *IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, Seoul, Korea, pp. 1-3, 2020. DOI: 10.1109/ICCE-Asia49877.2020.9277050.

[9] Y. Jing, B. Youssefi, M. Mirhassani, and R. Muscedere, "An efficient FPGA implementation of optical character recognition for license plate recognition," in *IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, Windsor: ON, Canda, pp. 1-4, 2017. DOI: 10.1109/CCECE.2017.7946734.

[10] Y. Caid, X. Lin, H. Qian, and P. Lu, "FPGA accelerator design for license plate recognition based on 1BIT convolutional neural network," in *Journal of Physics: Conference Series*, Hangzhou, China, vol. 1621, no. 1, 2020. DOI: 10.1088/1742-6596/1621/1/012022.

[11] D. M. F. Izidio, A. P. A. Ferreira, H. R. Medeiros, and E. N. d. S. Barros, "An embedded automatic license plate recognition system using deep learning," *Design Automation for Embedded System*, vol. 24, no. 1, pp. 23-43, Nov. 2019. DOI:10.1007/s10617-019-09230-5.

[12] I. V. Pustokhina, D. A. Pustokhin, J. J. P. C. Rodrigues, D. Gupta, A. Khanna, K. Shankar, C. Seo, G. P. Joshi, "Automatic vehicle license plate recognition using optimal K-means with convolutional neural network for intelligent transportation Systems," *IEEE Access*, vol. 8, pp. 92907-92917, May. 2020. DOI: 10.1109/ACCESS.2020.2993008.

[13] W. Riaz, A. Azeem, G. Chenqiang, Z. Yuxi, Saifullah, and W. Khalid, "YOLO based recognition method for automatic license plate recognition," in *IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, Dalian, China, pp. 87-90, 2020. DOI: 10.1109/AEECA49918.2020.9213506.

[14] Y. H. Yoon, D. H. Hwang, J. H. Yang, and S. E. Lee, "Intellino: Processor for embedded artificial intelligence," *Electronics*, vol. 9, no. 7, pp. 1-12, Jul. 2020. DOI: 10.3390/electronics9071169.

Youngwoo Jeong

received a B.S. degree from the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea in 2022, where he is pursuing the M.S. degree. His current research interests include computer programming, digital system design, and system-on-chip design.

Hyun Woo Oh

received a B.S. degree from the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea in 2021, where he is pursuing the M.S. degree. His current research interests include computer programming, digital system design, and processor for system-on -chip.

Soohee Kim

is undergraduate student in the Department of Electronic and IT Media Engineering at the Seoul National University of Science and Technology, Seoul, Korea. Her research interests include computer programming, digital system design.

Seung Eun Lee

received a Ph.D. degree in electrical and computer engineering from the University of California, Irvine (UC Irvine) in 2008, and B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon in 1998 and 2000, respectively. After graduating, he worked with Intel Labs., Hillsboro, OR, as a Platform Architect. In 2010, he joined the faculty of the Seoul National University of Science and Technology, Seoul. His current research interests include computer architecture, multi-processor system-on-chip, low-power and resilient VLSI, and hardware acceleration for emerging applications.