Implementation of a VAE-Based Circuit for Image Compression and Anomaly Detection and Its Potential Use in Edge Computing

1st Yuki Imamura
Kyushu Institute of Technology
School of Computer Science and Systems Engineering
Department of Computer Science and Networks
Fukuoka, Japan
imamura.yuuki475@mail.kyutech.jp

2nd Taiga Kawasaki
Kyushu Institute of Technology
School of Computer Science and Systems Engineering
Department of Computer Science and Networks
Fukuoka, Japan
kawasaki.taiga711@mail.kyutech.jp

***Abstract*—With the recent evolution of wireless communication technology and the spread of 5G, there is a need to create a new communication environment that takes advantage of the characteristics of low latency and multiple simultaneous connections. On the other hand, there is a potential problem due to the increased burden of cloud processing. Edge computing is attracting attention as a method to solve this problem. In this study, we developed a system that compresses and analyzes acquired images by implementing VAE using FPGA to realize efficient data processing on an edge server. As a result, the system was able to perform image compression and anomaly detection. However, the system's performance is expected to be further improved by utilizing higher-accuracy FPGAs and by incorporating color image processing.**

***Index Terms*—VAE, FPGA, Edge Computing, Image Compression, Anomaly Detection**

I. Introduction

In recent years, wireless communication technology has improved dramatically, and 5G communications are becoming increasingly popular. Unlike conventional communication standards such as 4G, 5G has the three characteristics of "high speed and large capacity," "low latency," and "multiple simultaneous connections," of which "low latency" and "multiple simultaneous connections" are important axes for building a new communication environment. Conventional 4G communications have focused on smartphones and cell phones used by people. However, 5G assumes that IoT devices such as vehicles, drones, and sensors will be connected to the network in large numbers.

In addition, cloud computing is the mainstream in the IT industry today [4]. As shown in the left side of Fig.1, the cloud performs the processing from the edge device and notifies the edge device of the result. However, there is a limit to the amount of data acquired from many devices that can be processed only by the cloud, and the amount of traffic will increase, making it difficult to enjoy the benefits of 5G.

To solve these problems, edge computing has been gaining attention in recent years. Edge computing is a technology that performs part of the processing that is conventionally performed in the cloud, such as data processing at a base station or server located near the user terminal (smartphone or IoT device) [2] [3]. Using this technique, it is possible to distribute the processing that is required in the cloud to edge computing, reduce the amount of communication traffic, and take advantage of the "low latency" that is one of the features of 5G.

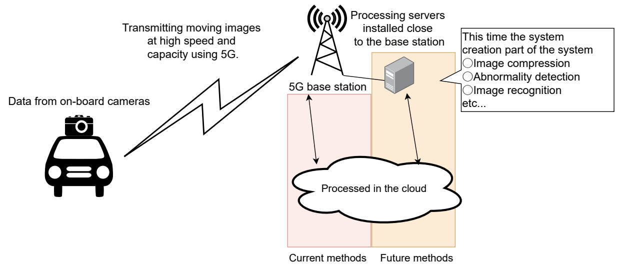

Transmitting moving images at high speed and capacity using 5G.
Processing servers installed close to the base station
This time the system creation part of the system
○Image compression
○Abnormality detection
○Image recognition
etc...
5G base station
Data from on-board cameras
Processed in the cloud
Current methods
Future methods

Fig. 1: Image of edge computing and the part of the system created this time

Therefore, we considered the use of VAE and FPGAs to realize edge computing. A system for detecting foreign objects on the road has already been created using VAE, and certain results have been obtained [5]. Therefore, in this project, we aimed to create a part of the process on the right side of Fig.1 by processing images acquired from an onboard camera using an FPGA equipped with a VAE.

II. Method

The tools and their versions used in the creation of this system are shown in Table I. ZYNQ-7010 with DIGILENT SoC is used as the FPGA evaluation board.

A. System Structure

This VAE-equipped system has the following two functions.

1 image compression
  The dimensional compression capability, one of the features of VAE, is applied to images.

TABLE I: Tools used in this development

Usage	tools used
For VAE simulation	MATLAB 2024b
Hardware simulation	MATLAB/Simulink 2022a
HDL Code Generation	HDL Coder
FPGA Design Software	Vivado 2022.1
Hardware acceleration	Vitis 2022.1
Evaluation Board	DIGILENT ZYNQ-7010

2 Anomaly detection
Another feature of VAE, anomaly detection is performed by comparing the original image and the generated image.

This section describes the overall concept. The images used in this project are grayscaled for simplicity. The image is divided into blocks like a JPEG, and VAE is used for each block. The block size is set to 16×16. PSNR is used to compare the original image with the compressed image for each block. Image compression and anomaly detection are processed using PSNR. For image compression, if the PSNR is above a set threshold, the compressed latent space is used as the compressed data, since there is no problem in using the compressed latent space. For anomaly detection, if the PSNR value is less than the threshold value, the system is set to judge the image as an anomaly because there is a high possibility that it is not a road.

In order to realize the above functions, the structure of VAE is described inII-A1. The details of the FPGA structure are explained in II-A2, and finally the SoC FPGA structure is explained in II-A3.

1) VAE Structure: A schematic of the VAE structure is shown in Fig.2. Since 16×16 images are used, 256 input dimensions and 256 output dimensions were used in the design. The latent space was designed to have 16 dimensions to allow for a certain degree of discrimination even after compression. The encoder part uses a ReLU function for the mean and a soft plus function for the variance. The decoder part uses a sigmoid function.

$$f(x) = x \qquad : ReLU function \tag{1}$$

$$f(x) = \log(1 + e^x) \qquad : Softplus function \tag{2}$$

$$f(x) = \frac{1}{1 + e^{-x}} \qquad : Sigmoid function \tag{3}$$

The VAE learning method is shown in Fig.3. First, a large number of road-only blocks are prepared. The blocks are used as teacher data to train VAE. The learning conditions are summarized in table II. In this case, we prepared a photo showing a road (the upper left image in Fig.3) and prepared the teacher data by converting the part showing only the road (the upper right image in Fig.3) into a block. The VAE was trained in MATLAB, and the weights and parameters output from the training were used to control the VAE mounted on FPGA.

2) FPGA Structure: The board used was ZYNQ-7010 manufactured by DIGILENT. In this design, $X(16)$, $W(16 \times 2)$, and $b(16)$ were prepared as input data and $Z(16)$ as output

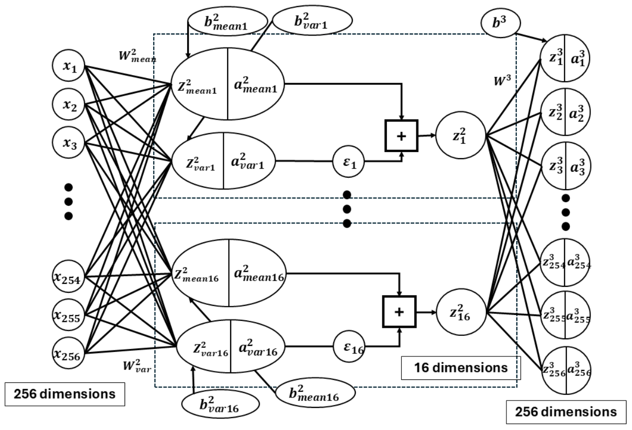


Fig. 2: Structure of $256 \times 16 \times 256$VAE

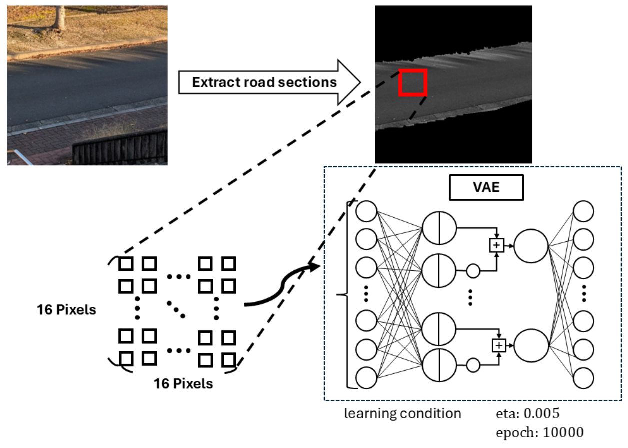


Fig. 3: How to study VAE

as shown in Fig.4 (the number of dimensions in parentheses). The unit enclosed in the red box, where $Output_1$ is the output, performs the operation of multiplying the input and weight parameters, as shown in Equation 4.

$$Output_1 = X_1 \times W1_1 \tag{4}$$

Sixteen such units are shown in the blue box. The final Z output is the computation of Equation 5.

$$Z_1 = \sum_{i=0}^{16} X_i \times W1_i + b1 \tag{5}$$

Initially, we wanted to put 256 input and 256 output operations on the FPGA, but there was a capacity limitation. Therefore, we designed the VAE of $256 \times 16 \times 256$ to have 16 inputs for X to make it easier to realize the VAE of $256 \times 16 \times 256$ that we designed this time.

TABLE II: VAE learning conditions

epoch	10000
eta	0.0005
Layer2(number of latent spaces)	16

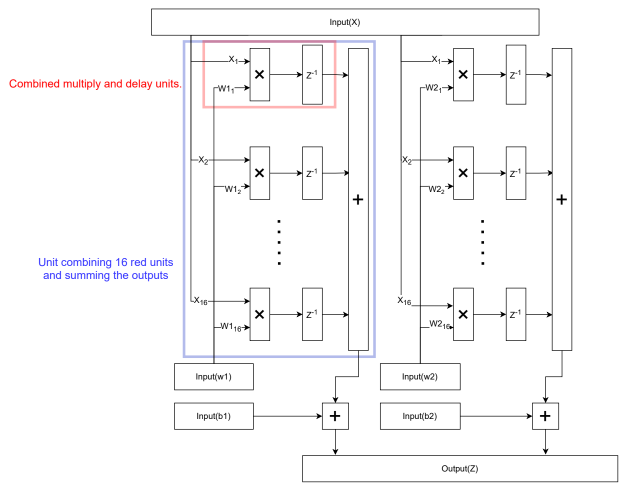


Fig. 4: FPGA structure

TABLE III: FPGA Resource Utilization

Resorce	Utilization	Available	Utilization[%]
LUT	3301	17600	18.76
LUTRAM	62	6000	1.03
FF	3297	35200	9.37
DSP	64	80	80.0
IO	12	100	12.0
BUFG	1	32	3.13

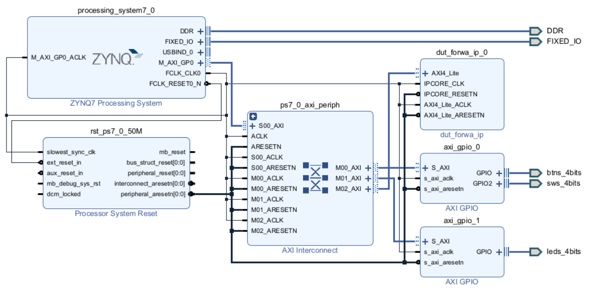


Fig. 5: SoC FPGA Configuration Diagram

3) SoC FPGA Structure: Fig.5 shows an overview of the SoC FPGA system structure. The processing **processing_system7_0** performs various processes through the bus, and plays the role of CPU. The **dut_forwa_ip_0** part is the part of the FPGA created this time. The resource utilization evaluation after implementation is shown in Table III.

Next, an overview of the SoC FPGA processing is shown in Fig.6. The SD card contains weights and parameters stored in CSV files and image data in RAW format. The CPU reads these data. The CPU then processes the input data according to the created FPGA and sends the data to the FPGA. The FPGA executes the data as soon as it is stored and stores the output results. The CPU reads the output result and processes the output data. These processes are repeated.

This VAE is $256 \times 16 \times 256$, and the FPGA structure created in II-A2 is 16×2. The part of the FPGA that can be processed by running the designed FPGA once is shown in Fig.7. The encoder assigns the computation of the two outputs to z^2_{mean} and z^2_{var}. Since the FPGA can process 16 out of 256 input dimensions in a single use, it is necessary to use the FPGA 16 times to compute a single latent space. The output z^2_{mean} and z^2_{var} are designed to be stored in b. Since there are 16 latent spaces, the encoder calculation runs a total of $16 \times 16 = 256$ times. When computing the decoder part, only one FPGA run is needed to obtain 2 out of 256 dimensions of the output of the third layer Z^3_i. Therefore, the FPGA is run 128 times in the decoder. This kind of control was created using a CPU.

The z^2_{mean}, z^2_{var}, and z^3 obtained from the output are used to calculate a^2_{mean}, a^2_{var}, and a^3 using the active functions in the software. Furthermore, z^2 is also calculated on the software using a^2_{mean} and a^2_{var}. Finally, PSNR is measured by comparing the input and output a^3 values.

B. Experimental Procedure

In this case, we prepared three types of images shown in Fig.8. Fig.8a is an image without any falling objects. Fig.8b and Fig.8c are images with falling objects, and were prepared for comparison of anomaly detection. The size of the image is 512×512, and the image is grayed out when passed through VAE. The image is processed by dividing it into $16^times16$ blocks, and the PSNR is measured. The PSNR is compared between the MATLAB output and the FPGA output. The PSNR output from the FPGA is written to a CSV file, and the data is presented in MATLAB.

III. Experiments and Discussions

A. Experimental Results

First, the SoC FPGA execution screen is shown in Fig.9. The CSV file stored in the SD card is read by pressing button 0. The image is passed through the VAE by pressing buttons 1 to 3. Fig.9 shows the state after button 0 is pressed and the parameters are acquired.

1) Comparison of output of image 1 : Fig.10b shows the output in MATLAB and Fig.10c shows the output in FPGA.

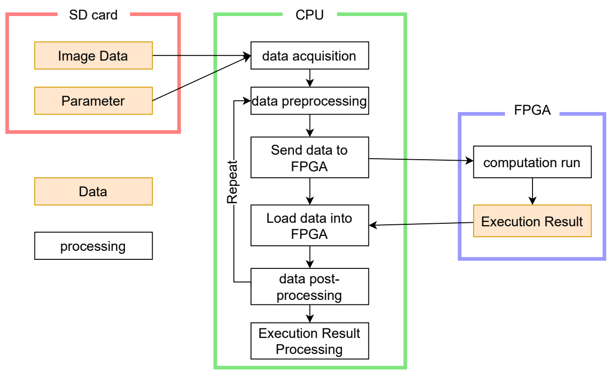


Fig. 6: SoC FPGA Processing Overview

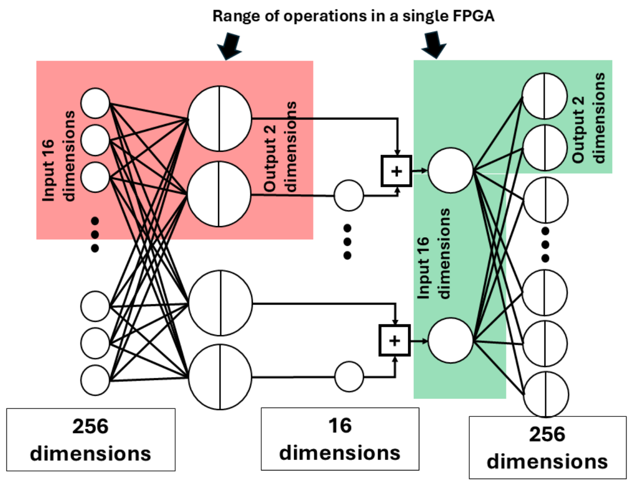
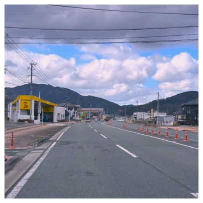
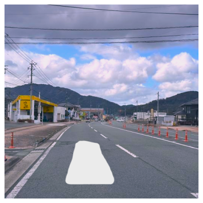
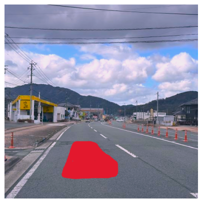


Fig. 7: Image of FPGA use

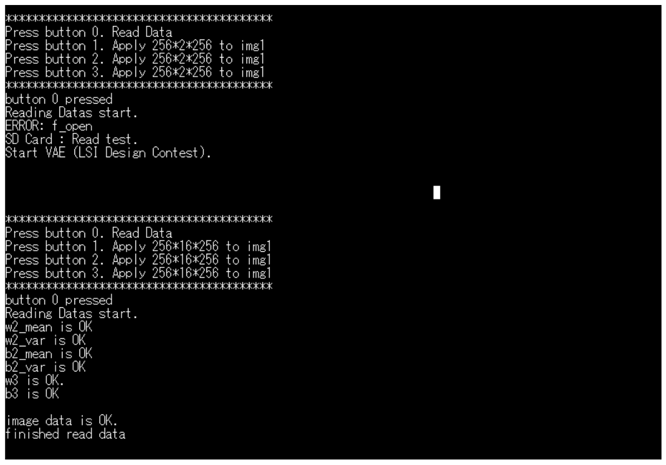




(a) Image 1 Original image **(b)** Image 2 With falling objects **(c)** Image 3 With falling objects

Fig. 8: Evaluation image to be tested this time

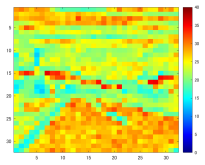
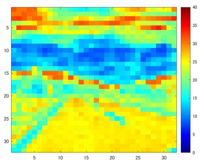
In MATLAB, the PSNR around the road area shows a value of 25 or higher. In addition, parts of the sky and mountains also have a high PSNR. The FPGA output shows a lower overall PSNR compared to MATLAB. However, the PSNR of the road section is around 25, which is not a bad result.

2) Comparison of output of image 2 : Fig.11b shows the output from MATLAB and Fig.11c shows the output from FPGA. Both MATLAB and FPGA outputs show a worse PSNR in the area of the falling objects.



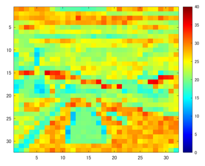
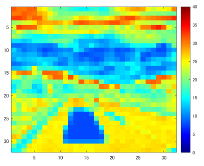
```
****************************************
Press button 0. Read Data
Press button 1. Apply 256*2*256 to img1
Press button 2. Apply 256*2*256 to img1
Press button 3. Apply 256*2*256 to img1
****************************************
button 0 pressed
Reading Datas start.
ERROR: f_open
SD Card : Read test.
Start VAE (LSI Design Contest).



****************************************
Press button 0. Read Data
Press button 1. Apply 256*16*256 to img1
Press button 2. Apply 256*16*256 to img1
Press button 3. Apply 256*16*256 to img1
****************************************
button 0 pressed
Reading Datas start.
w2_mean is OK
w2_var is OK
b2_mean is OK
b2_var is OK
w3 is OK.
b3 is OK

image data is OK.
finished read data
```

Fig. 9: SoC FPGA execution screen

(a) Image 1 to be tested **(b)** PSNR output block by block in MATLAB **(c)** PSNR output block by block in FPGA

Fig. 10: Processing results for image 1

(a) Image 2 to be tested **(b)** PSNR output block by block in MATLAB **(c)** PSNR output block by block in FPGA

Fig. 11: Processing results for image 2

3) Comparison of output of image 3 : Fig.12b shows the output in MATLAB and Fig.12c shows the output in FPGA. Unlike the output in image 2, the PSNR in the area where the falling object is located is conversely improved.

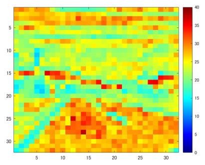
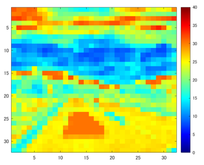
4) Summary of results: Comparing the output images, there were differences between the MATLAB and FPGA output results. However, since the PSNR was higher for the road section in FPGA than for other sections, we do not think that there was a failure.

In addition, it was found that the anomaly detection was affected by the color of the falling object. The PSNR worsened when the object was white, while the PSNR improved when the object was red.

B. Consideration

From the results obtained, the following three points are discussed.

1) The point that PSNR is high except for roads
2) MATLAB and FPGA outputs are different

(a) Image 3 to be tested **(b)** PSNR output block by block in MATLAB **(c)** PSNR output block by block in FPGA

Fig. 12: Processing results for image 3

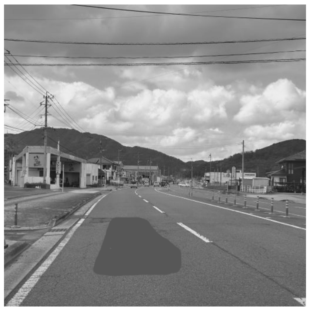


Fig. 13: Image 3 converted to grayscale

3) PSNR is improved even for red objects

(1) can be considered from two points of view: the effect of grayscaling and the effect of VAE characteristics. In particular, when the sky portion is grayscaled, it has fewer features than the road portion. Therefore, the VAE created in this study can represent them, and it is thought that the image restoration capability of the VAE can be demonstrated.

(2) is considered to be caused by the variables used and the fixed-point error. However, since the detailed output has not been confirmed, we will conduct further verification to determine the cause.

(3) is thought to be caused by grayscaling. When the red color was converted to grayscale, it was almost the same as the color of the road (Fig.13). This suggests that the color scale should be used to correctly identify falling objects.

1) Evaluating adoption in edge computing: We will evaluate whether it can be adopted as edge computing.

First, we evaluate image compression. This time, the PSNR threshold is set at 25, and judgments are made on the basis that images above the threshold are compressible and those below the threshold are uncompressible. The number of blocks for which the threshold is 25 or higher, the capacity before compression, the capacity after compression, and the compression ratio are shown in Table IV. The size calculation is shown in the formula (6). Since the original image takes values from 0 to 255, each pixel is 8 bits. Since the latent space is compressed data, 8-bit fixed-point numbers are used. The B denotes the number of blocks above the threshold.

$$size = B \cdot 16 \cdot 8[bit] + (32 \cdot 32 - B) \cdot 16 \cdot 8[bit] \tag{6}$$

The compression ratio for all the images is close to 70%. By sending compressed images to the cloud for processing, it is possible to reduce the amount of traffic from the base station to the processing server and to improve the processing speed in the cloud.

Next, anomaly detection is evaluated. As shown in the experimental results, whether an abnormality can be detected depends on the color of the falling object. However, since abnormality detection is possible when the object is white, we believe that it is possible if the processed image is a color image. If the recognition is performed on the area where the vehicle is going to run, it is possible to avoid accidents by immediately notifying the vehicle of the recognition results.

TABLE IV: image compression effect

No.	Blocks	Before[bit]	After[bit]	Com.ratio[%]
1	346	2097152	1432832	68.32
2	305	2097152	1511552	72.08
3	349	2097152	1427072	68.05

Finally, we evaluate real-time performance, which is important in edge computing. The system took about 4 to 5 seconds to input an image and output the PSNR. If the system were to continue at this rate, an accident could occur before an abnormality judgment could be made. The reason for the long processing time is thought to be that data was written and read many times using a time-consuming bus. In this case, the circuit that could be mounted on the FPGA is a part of VAE, and the FPGA is executed many times, and data is transferred using the bus each time. In fact, to process a 512×512 image in VAE, the FPGA is run 393216 times. However, it is highly possible that this problem can be solved by using a higher-precision FPGA.

IV. Conclusions and Future Prospects

In this study, we implemented VAE on FPGA and constructed a system that compresses road images and detects anomalies. The results show that image processing in edge computing using FPGAs is feasible.

Although the system has some problems, we believe that they can be solved with appropriate improvements. Therefore, we believe that the system has potential for future industrial applications.

This contest allowed us to experience LSI development using FPGAs. I felt the future potential of FPGAs and recognized that FPGAs are necessary devices for efficient computation in today's world where AI is used on a daily basis. I would like to continue to develop FPGAs that can demonstrate their capabilities, and since technologies for developing FPGAs, such as high-level synthesis, are evolving day by day, I would like to challenge myself in various ways.

Acknowledgment

Thank you to the LSI Design Contest Executive Committee for organizing this contest, and to the organizers, co-sponsors, sponsors, and supporters for their support. I would also like to thank all the professors who provided various kinds of support even though they were not assigned to my laboratory.

References

[1] H. Morikawa, *5G Jisedai Ido Tsushin Kikaku no Kanosei*, Iwanami Shoten, Tokyo, 2020.
[2] Y. Tanaka, N. Takahashi, and R. Kawamura, "IoT Jidai o Hiraku Edge Computing no Kenkyu Kaihatsu," *NTT Giho Journal*, vol. 27, no. 8, pp. 59-63, 2015.
[3] H. Yokota, S. Oda, T. Kobayashi, D. Ishii, T. Ito, and A. Isozumi, "IoT no Missing Link o Tsunagu Edge Computing Gijutsu," *NEC Giho*, vol. 70, no. 1, 2017.
[4] Ministry of Internal Affairs and Communications, "Reiwa 6-nenban Joho Tsushin Hakusho," 2024.
[5] T. Yamamoto, A. Hashimoto, and H. Okamoto, "Heikin Gazou ni Taisuru VAE Ijou Kenshi no Tekiyo ni Yoru Doro Rakka-mono Kenshutsu," in *Proc. Annu. Conf. Jpn. Soc. Artif. Intell.*, vol. 35, 2021.

[6] ”LSI Design Contest,” Available: http://www.lsi-contest.com/. Accessed: 2025-01-31.