

Hand-Written Number Classification by Hardware Neural Network

Jialong Chen

University of California, Los Angeles, United States of America

cain011231@ucla.edu

Abstract:

This paper delves into the innovative integration of Very Large Scale Integration (VLSI) with machine learning by developing a perceptron-based digital recognition model tailored for handwritten number classification. This model capitalizes on the perceptron algorithm—a seminal neural network form adept at binary classification via the computation of a weighted sum of inputs followed by a nonlinear activation function. The implementation of VLSI technology underpins the model's architecture, enabling the amalgamation of multiple logic functions onto a singular chip. This consolidation significantly diminishes the size and cost of the electronic components while concurrently elevating performance and energy efficiency. The paper thoroughly explores each phase of the model's development, from its initial conceptualization and algorithmic formulation through to simulation and final hardware implementation, highlighting the intricate processes and meticulous adjustments required for optimization. The study aims to showcase not only the technical feasibility but also the extensive practical advantages and potential applications of melding traditional circuit design techniques with contemporary machine learning methodologies in digital recognition systems.

Keywords: Neural networks; VLSI design; Number classification.

1. Introduction

The application of machine learning technologies has been profoundly transformative across various sectors, heralding significant advancements in automation and data analysis. This dynamic field draws upon algorithms to discern patterns and make predictions from vast data arrays, greatly enhancing decision-making processes and operational efficiencies [1]. Today, machine learning's impact is pervasive,

extending from natural language processing and image recognition to complex applications in finance, healthcare, and beyond [2]. These technologies not only foster innovation but also drive the development of cutting-edge applications that redefine traditional industry practices.

However, the rapid evolution of machine learning necessitates advancements in underlying hardware technologies to sustain its expanding complexity and computational demands. Enter Very Large Scale In-

tegration (VLSI), a pivotal semiconductor technology that synergizes numerous logic functions onto a single chip. This integration facilitates significant reductions in device size and cost while simultaneously boosting performance and energy efficiency [3]. The development of VLSI has been instrumental in advancing the computational capabilities required to handle intricate machine learning algorithms effectively.

This paper explores a novel integration of VLSI with machine learning through the development of a perceptron-based digital recognition model specifically for handwritten numbers. The research focuses on crafting a light-weight neural network model optimized for VLSI, aimed at enhancing the speed and accuracy of digital recognition tasks. The model employs basic perceptron algorithms for binary classification, optimized through innovative circuit design and simulation methodologies. The study outlines the full development trajectory of this model, from conceptualization and algorithmic adjustments to simulation on platforms like MATLAB and MODELSIM, and final hardware implementation using FPGA technology. By detailing this process, the paper not only demonstrates the technical feasibility but also highlights the broader practical implications of merging traditional VLSI designs with modern machine learning techniques in the realm of digital recognition [4-6].

The paper is organized into five main sections. The second section provides an overview of the development of digit recognition technology based on FPGA hardware design. The third section introduces the relevant technologies used in this paper, including the architecture and components of neural networks, training of neural networks on the MATLAB platform, and hardware simulation setup on the MODELSIM platform [7]. The fourth section presents the detailed simulation process and results. The fifth section discusses the future development and applications of digit recognition technology using the FPGA-based perceptron model introduced in the paper. Finally, the last section summarizes the key points and contributions of the paper.

2. Relevant Technologies

2.1 Perceptron

In this section, the principles, primary structure, and operational process of the perceptron neural network used for digit recognition will be thoroughly discussed. The focus will be on explaining how the perceptron functions, detailing its architecture, and describing the step-by-step procedure involved in its execution.

2.1.1 Binary classification

The perceptron is a linear classification model designed for binary classification tasks and belongs to the category of supervised learning algorithms. Its input consists of feature vectors representing instances, while the output corresponds to the class label of the instance, typically taking values of +1 or -1. The primary objective of the perceptron is to identify a separating hyperplane that can partition the sample space into two distinct classes. To achieve this, the perceptron employs a loss function based on misclassified samples and utilizes gradient descent to optimize this loss function, ultimately determining the optimal model parameters.

Given a dataset of m samples, each characterized by n -dimensional features and a binary class label:

$$\left(x_1^{(0)}, x_2^{(0)} \dots x_n^{(0)}, y_0\right), \left(x_1^{(1)}, x_2^{(1)} \dots x_n^{(1)}, y_1\right), \dots \left(x_1^{(m)}, x_2^{(m)} \dots x_n^{(m)}, y_m\right).$$

The need is to find a hyperplane:

$$w_0 + w_1x_1 + \dots + w_nx_n = 0 \quad (1)$$

Where w_n is n -dimensional row vector.

By using the Formula (1), data could be divided into two groups, as shown in Formula (2) and (3) respectively.

$$w_0 + w_1x_1 + \dots + w_nx_n > 0 \quad (2)$$

$$w_0 + w_1x_1 + \dots + w_nx_n < 0 \quad (3)$$

In some cases, to simplify the representation of the separating hyperplane, it is a custom to augment each sample with an additional feature dimension. This is achieved by appending a constant value x_0 (typically 1) to each feature vector, allowing the bias term to be incorporated into the weight vector as an additional parameter. With this modification, the hyperplane can be expressed in a more concise form, simplifying the mathematical formulation and the computation of the decision boundary during the training process. Therefore, hyperplane is expressed:

$$\sum_{i=0}^n w_i x_i = 0 \quad (4)$$

There model of perception is expressed:

$$y = \text{sign}(w \bullet x) \quad (5)$$

$$\text{sign}(x) = \{-1, x < 0; 1, x > 0\} \quad (6)$$

Assuming the training dataset is linearly separable, the objective is to identify a hyperplane that can perfectly distinguish between positive and negative instances, thereby determining the parameters w and b of the perceptron model. To achieve this, it is necessary to define a learning strategy by specifying an empirical loss function, which is subsequently minimized to optimize the model.

While the total number of misclassified points may seem like a natural choice for the loss function, it is not contin-

uously differentiable and thus challenging to optimize. As a result, the perceptron adopts a loss function based on the total distance of the misclassified points from the hyperplane, facilitating a more tractable and efficient optimization process.

Assume hyperplane is $h = w \cdot x + b$, thus the distance from sample point x to hyperplane is:

$$d = \frac{w \cdot x + b}{\|w\|} \quad (7)$$

Where $\|w\|$ is L2 norm of w , b is the bias.

Let the set of misclassified points with respect to the hyperplane S be denoted as M . The total distance of all misclassified points from the hyperplane S can then be expressed as:

$$-\frac{1}{\|w\|} \sum_{x_i \in M} y_i (w \cdot x_i + b) \quad (8)$$

By simplifying Formula (8), the final loss function formula is received:

$$L(w, b) = \sum_{x_i \in M} y_i (w \cdot x_i + b) \quad (9)$$

Subsequently, in order to minimize the error function, the initial weights and bias need to be continuously updated. By computing the current value of the loss function and adjusting the parameters accordingly using an appropriate step size and learning rate, the process continues until the loss function is minimized. The perceptron learning algorithm is theoretically guaranteed to converge, provided that the data is linearly separable. For linearly separable data, the perceptron can find a decision boundary after a finite number of iterations, ensuring that all samples are classified correctly.

2.1.2 Multi-classification

Unlike binary classification tasks, where inputs are often simple vectors, digit recognition tasks involve pixel matrices where each pixel's brightness value serves as an input feature. For instance, a handwritten digit image with dimensions 28x28 pixels can be flattened into a one-dimensional vector with 784 input nodes. In digit recognition, the perceptron's output must not be limited to binary classification but should instead differentiate between each digit category (0 to 9). Consequently, one can employ either 10 separate perceptron models, each dedicated to identifying a specific digit, or utilize multiple output

nodes to handle multi-class classification.

In multi-class classification tasks, the perceptron's loss function is typically based on the number of misclassified samples. The objective in such scenarios is to minimize the number of errors across all classes, thereby ensuring that the model correctly classifies each input image into its appropriate digit category. The loss function is designed to reflect this goal, focusing on minimizing classification errors throughout the process.

$$L(w, b) = -\sum_{i=1}^N \sum_{k=1}^K y_{ik} (w_k \cdot x_i + b_k) \quad (10)$$

Where N is the number of training sample. K is the number of categories. y_{ik} is the real label of sample x_i for category k .

Then take the same solution to update the weights and bias to minimize the error function.

2.2 MATLAB Simulation

This section mainly describes the process of training the perceptron in MATLAB and exporting the training data. First, the original training data and parameter files need to be imported into the MATLAB platform. Then, through training and iteration of the perceptron, the corresponding weight parameter files are generated and exported. Additionally, the training data files need to be processed, and test files for random digit samples are generated and exported. Before these files are input into the hardware simulation system for testing, a preliminary simulation can be conducted on the MATLAB platform using the generated weight parameters to test their digit recognition accuracy. If the results meet the expected standards, the files are then imported into the hardware simulation system for further simulation to verify the feasibility of the hardware system.

2.3 Hardware Design

In this section, the implementation process for constructing the hardware necessary for achieving the functionality of a perceptron-based digit recognition system will be introduced. This includes the handling of data inputs, performing multiplications and additions, selecting the index of the maximum value, designing the sequential logic, and optimizing the model, among other aspects. The general illustration of the function of number classification is shown below. As show in the Fig. 1.

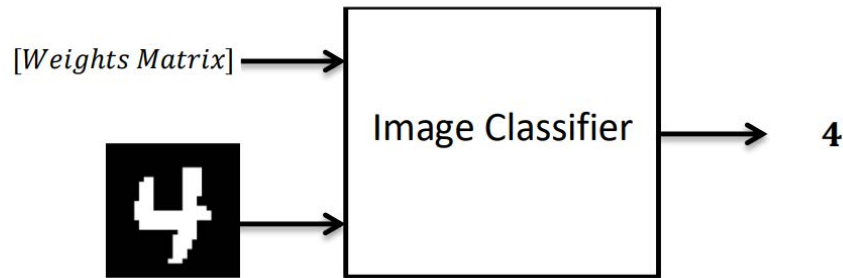


Fig. 1 Function of number classification (Photo credit: Original).

2.3.1 Handling of data inputs

The hardware system needs to receive 10 sets of weight information, each consisting of 785 entries with a bit width of 19, where the final entry corresponds to the bias term. In order to take the matrix multiplication and addition of weight information and pixel information, biased terms need to be separated first. To prevent data overflow, the hardware internally processes this weight information using a 32-bit width. Similarly, for the 10 sets of 28x28 pixel data, each with a bit width of 10, the system adopts a 24-bit width for reception. This prepares the system for subsequent processing and operations.

2.3.2 Matrix Multiplication and addition

For the multiplication and addition of the weight matrix and the pixel matrix, the hardware employs an ele-

ment-wise multiplication followed by an accumulation process. These operations are performed sequentially over two clock cycles, where the first clock cycle completes the multiplication, and the second clock cycle handles the accumulation. It is important to note that, prior to these operations, the hardware system performs another bit-width adjustment on the input data. During the generation of the test dataset, it was observed that only the central portion of the image data is essential for processing, according to Fig. 2. To optimize the computational flow, the bit-width of both the weights and pixels is adjusted. Specifically, the bit-width of the weights is reduced from 32 bits to 24 bits, and the bit-width of the pixels is reduced from 24 bits to 8 bits. This transformation enhances processing efficiency while maintaining the necessary precision for accurate computation. As show in the Fig. 2.

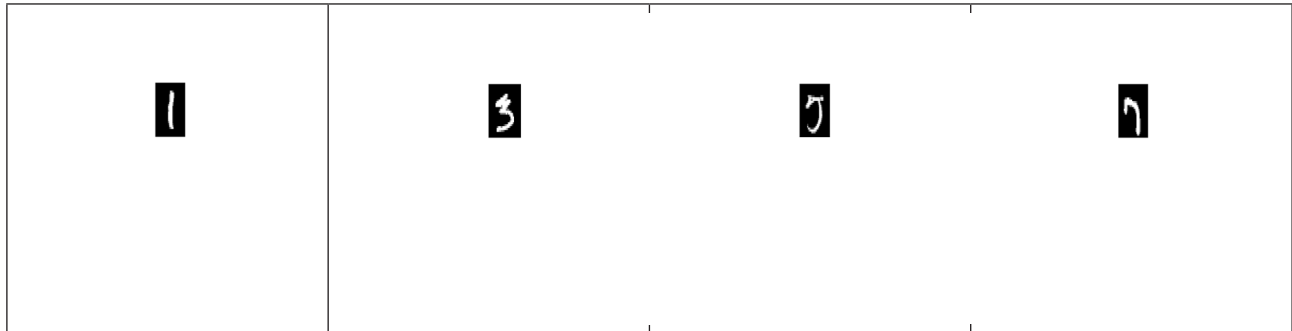


Fig. 2 Test dataset of image data (Photo credit: Original).

2.3.3 Bias term disposal

During data input, to incorporate the bias term into each set of weights, an additional data element with a value of 1 is appended after the 784 pixels of each image. This additional element ensures that the bias term is correctly represented during matrix multiplication. After completing the matrix multiplication and accumulation operations, the previously separated bias term is added to the accumulated sum, forming the complete weight value. This complete value is then fed into the selector for further processing.

This process allows the bias term to be treated similarly to

the other weights during multiplication, while still being accounted for in the final accumulated output, ensuring accurate representation of the bias during the entire operation.

2.3.4 Index of maximum number selection

For the ten final weight values, the hardware system needs to select the index corresponding to the maximum weight. This system employs a four-layer tree structure consisting entirely of two-input comparators, with a total of 9 comparators, as shown in Fig. 2. Compared to a global search using a single comparator, the tree structure significantly

reduces the time required for the maximum value selection.

By using this hierarchical approach, each layer of the tree performs pairwise comparisons between two weight values, passing the larger value to the next layer. This process continues until the final comparison yields the maximum

Fig. 3 Tree structure of selector (Photo credit: Original).

2.3.5 Time logic design

For the task of recognizing 10 digits, this project utilizes a serial processing approach to perform continuous operations. For each individual digit recognition task, parallel processing is employed, where 10 pre-processed sets of weight information are simultaneously applied to the same set of pixel data. Specifically, the process operates as follows:

The system serially inputs 784 pixels values. For each pixel, it performs a multiplication with the corresponding weight value, followed by accumulation in the next clock cycle. Consequently, multiplying and accumulating the 784 weight and pixel values takes a total of 785 clock cy-

cles. value and its corresponding index. The tree structure not only optimizes performance by reducing the number of sequential comparisons but also minimizes latency, making it a more efficient design for real-time applications. As show in the Fig. 3.

cles.

Following this, a comparator identifies the index corresponding to the maximum weight value, which requires an additional 4 clock cycles. During the comparison process, the matrix multiplication and addition modules can be initialized, allowing the next set of pixel and weight information to be input simultaneously. This introduces a brief period of parallel processing, thereby enhancing the efficiency of information handling. By overlapping these operations, the system can reduce idle time and improve overall processing throughput, contributing to a more efficient execution of the recognition task. The general processing sketch map is shown below. As show in the Fig. 4.

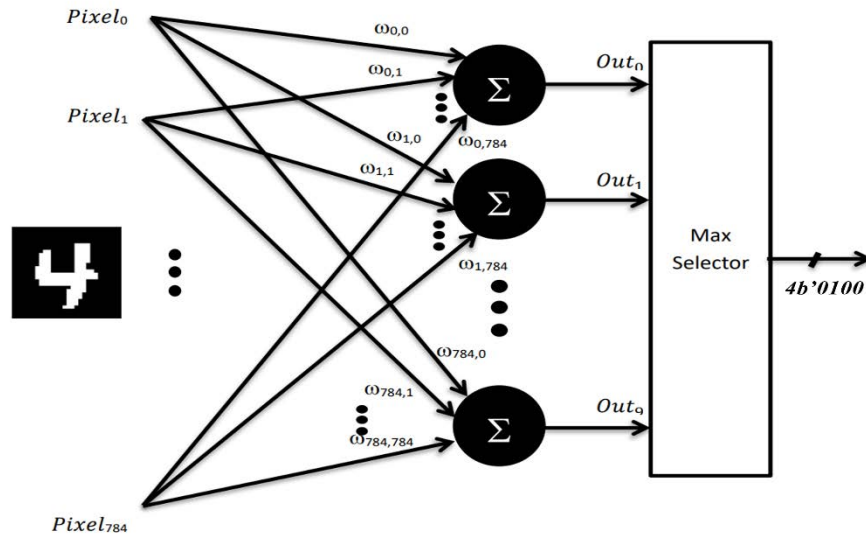


Fig. 4 Sketch map of number classification (Photo credit: Original).

Thus, the complete recognition task for each digit takes 789 clock cycles. There is a 4-cycle parallel processing phase between consecutive digit recognition stages. During this phase, the system simultaneously processes the comparison of the current digit while initializing the matrix multiplication and addition for the next digit. This overlap allows the system to reduce the total processing time, enhancing overall efficiency by utilizing these intermediate cycles for parallel computation. Given that the system is configured with a clock period of 20 nanoseconds, processing all 10 digits recognition tasks requires a total of 157,080 nanoseconds.

This approach balances efficiency and accuracy, utilizing parallelism for concurrent weight-pixel operations while maintaining precise timing for optimal performance in digit recognition tasks.

2.3.6 Model optimization

To further optimize resource utilization, the hardware system can be refined by employing binary image processing instead of grayscale values. By converting the input images to binary format, the system can significantly reduce the computational resources required for operations. Specifically, pixel values greater than 128 are set to 1, while

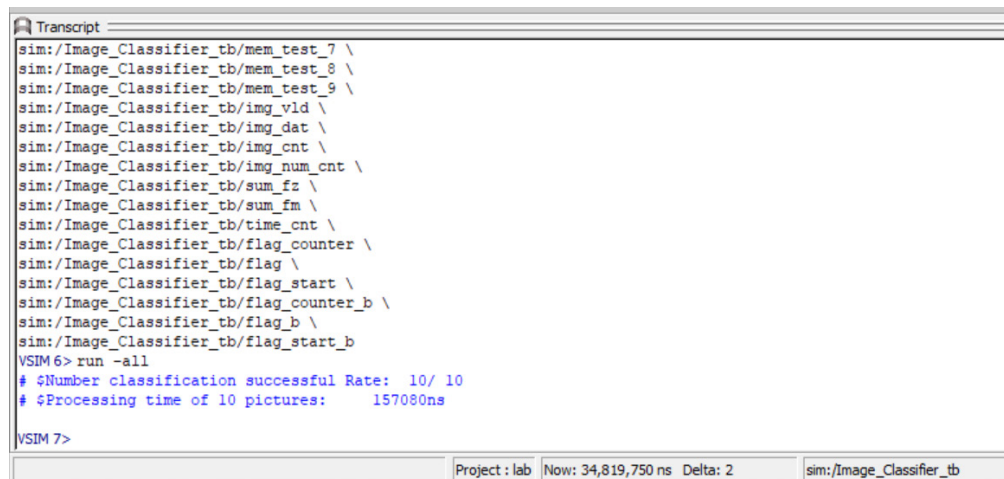
those below 128 are set to 0. This approach eliminates the need for traditional multipliers and omits the matrix multiplication process. Since the binary image consists solely of 0s and 1s, the system can directly accumulate the corresponding weight values for pixels with a value of 1.

The optimized hardware system maintains the same timing logic as the original design, with the matrix multiplication phase being replaced by the binary operations, all of which require only one clock cycle per operation. However, the binary conversion process substantially reduces the level of detail in the image, which can result in the loss or blurring of important features, particularly in areas with fine edges or strokes, potentially affecting classification accuracy. Nonetheless, the recognition accuracy remains within an acceptable range, making the trade-off between a slight reduction in recognition accuracy and significant resource savings a worthwhile compromise.

3 Stimulation Result

The simulation is implemented on WINDOWS 11 using MATLAB 2024a, VSCODE 2024, and MODELSIM 2016 platforms. Initially, the raw training and testing data files are imported into the MATLAB environment. The imported training data is then used to train the perceptron model. After training, the 10 sets of weight information are exported, and ten groups of test datasets are randomly generated using the original test files, which are then also exported. These datasets are subsequently imported into the hardware model code through file read and write operations. The code development process is entirely conducted within the VSCODE platform. Finally, the completed code is imported into MODELSIM for simulation, where the waveform, recognition success rate, and simulation time are analyzed and presented.

The stimulation results before binary operation are shown below. As show in the Fig. 5, Fig. 6 and Fig. 7.



```
Transcript
sim:/Image_Classifier_tb/mem_test_7 \
sim:/Image_Classifier_tb/mem_test_8 \
sim:/Image_Classifier_tb/mem_test_9 \
sim:/Image_Classifier_tb/img_vld \
sim:/Image_Classifier_tb/img_dat \
sim:/Image_Classifier_tb/img_cnt \
sim:/Image_Classifier_tb/img_num_cnt \
sim:/Image_Classifier_tb/sum_fz \
sim:/Image_Classifier_tb/sum_fm \
sim:/Image_Classifier_tb/time_cnt \
sim:/Image_Classifier_tb/flag_counter \
sim:/Image_Classifier_tb/flag \
sim:/Image_Classifier_tb/flag_start \
sim:/Image_Classifier_tb/flag_counter_b \
sim:/Image_Classifier_tb/flag_b \
sim:/Image_Classifier_tb/flag_start_b
V$IM6> run -all
# $Number classification successful Rate: 10/ 10
# $Processing time of 10 pictures:      157080ns
V$IM7>
```

Project : lab Now: 34,819,750 ns Delta: 2 sim:/Image_Classifier_tb

Fig. 5 Stimulation result of grayscale type (Photo credit: Original).

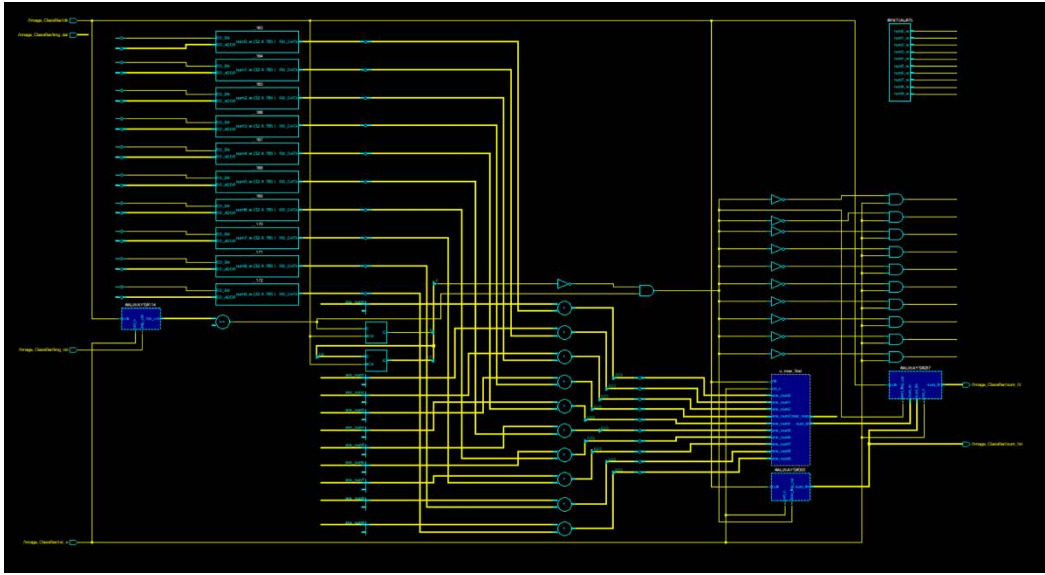


Fig. 6 Schematic of circuit of grayscale type (Photo credit: Original).

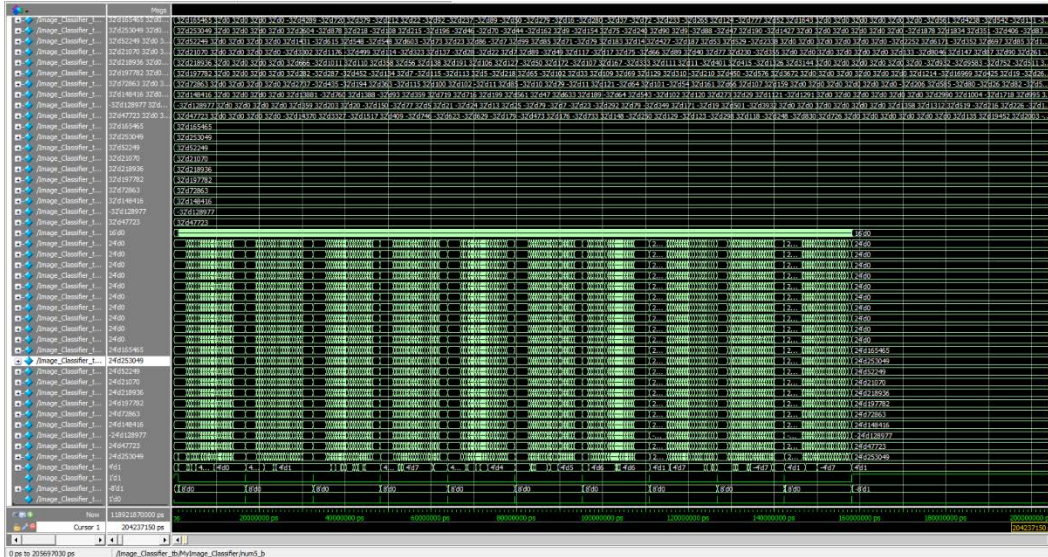


Fig. 7 Wave form of grayscale type (Photo credit: Original).

The stimulation results after binary operation are shown below. As show in the Fig. 8, Fig. 9 and Fig. 10.

```
Transcript
sim:/Image_Classifier_tb/MyImage_Classifier/ans_num6 \
sim:/Image_Classifier_tb/MyImage_Classifier/ans_num7 \
sim:/Image_Classifier_tb/MyImage_Classifier/ans_num8 \
sim:/Image_Classifier_tb/MyImage_Classifier/ans_num9 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num0 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num1 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num2 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num3 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num4 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num5 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num6 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num7 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num8 \
sim:/Image_Classifier_tb/MyImage_Classifier/end_ans_num9 \
sim:/Image_Classifier_tb/MyImage_Classifier/max_num \
sim:/Image_Classifier_tb/MyImage_Classifier/num_id \
sim:/Image_Classifier_tb/MyImage_Classifier/last_flag \
sim:/Image_Classifier_tb/MyImage_Classifier/last_flag_dly \
sim:/Image_Classifier_tb/MyImage_Classifier/last_flag_up
VSIM 4> run -all
# $Number classification successful Rate: 9/ 10
# $Processing time of 10 pictures: 157080ns
VSIM 5>
```

Fig. 8 Stimulation result of binary type (Photo credit: Original).

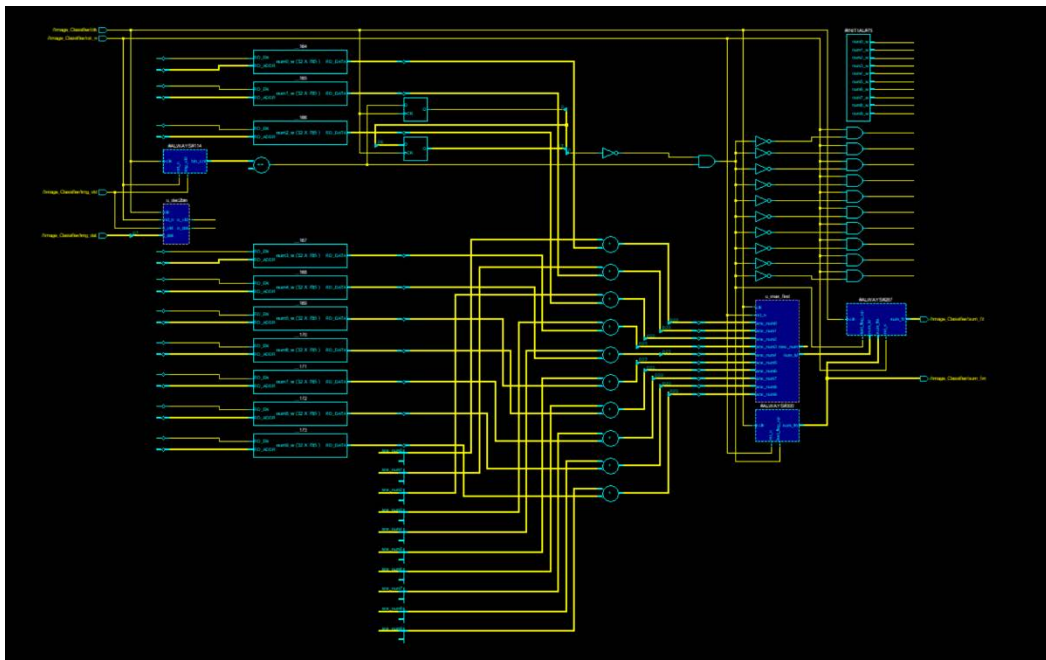


Fig. 9 Schematic of circuit of binary type (Photo credit: Original).

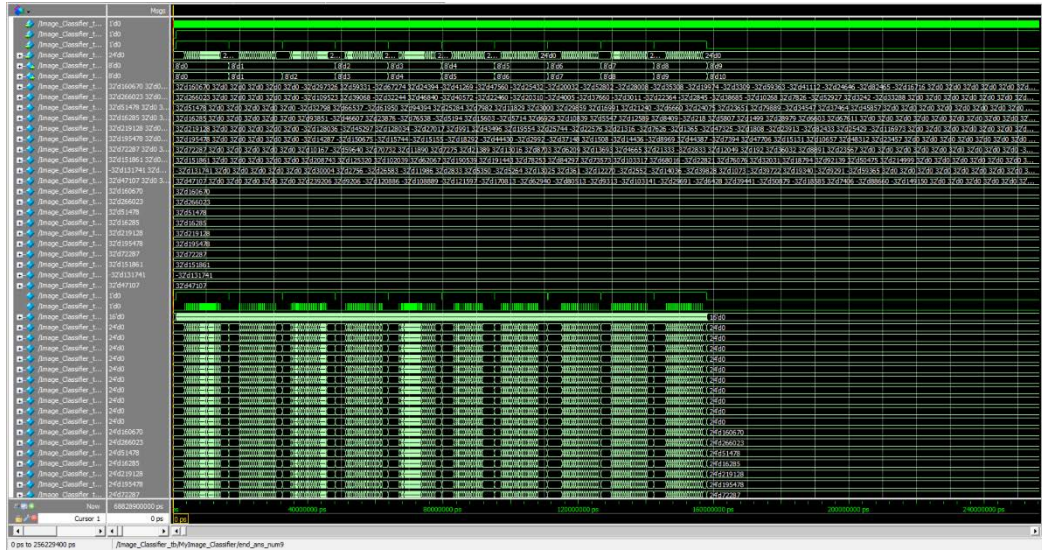


Fig. 10 Wave form of binary type (Photo credit: Original).

From the simulation results, the graphical display indicates that the binarization process caused the recognition task for the digit “1” to fail, resulting in an overall recognition success rate of 90%. However, binarization eliminates the need for multipliers in the matrix multiplication, allowing the hardware system to save the resources of ten multipliers. Therefore, the reduction in recognition accuracy is an acceptable trade-off for the significant resource savings achieved in the system.

4 Future Development and Application

From the simulation results in Section 4, it is evident that the system must balance between recognition accuracy and resource consumption. However, with the ongoing advancements in machine learning and neural networks, more complex multilayer models, such as convolutional neural networks (CNNs) and artificial neural networks (ANNs), are now widely adopted. Future research can utilize more advanced network models for digit recognition tasks and design corresponding hardware simulation systems, eliminating the need to compromise between accuracy and resource efficiency. Additionally, the application of neural network models based on FPGA hardware design has been expanding beyond simple digit recognition. For example, S. M. Riad Hossain et al. employed an ANN model to recognize handwritten Bengali characters and designed an FPGA-based hardware system to accelerate this recognition task [8]. Similarly, Souha Bel Hadj Salah et al. developed a CNN-based system on FPGA to classify brain activity images, enabling early detection and intervention in Alzheimer’s disease treatment [9]. Furthermore, Muhammad Arsalan et al. utilized a spiking neural network (SNN) combined with radar sensors to recognize

non-contact aerial hand gestures [10]. These application scenarios highlight the promising future of cross-disciplinary developments in digital circuit design and neural network systems.

5 Conclusion

This research has successfully demonstrated the integration of Very Large Scale Integration (VLSI) with machine learning to develop a perceptron-based digital recognition model for handwritten numbers. The lightweight neural network model optimized for VLSI, as presented, significantly enhances the speed and efficiency of digit classification tasks. Through meticulous simulations and detailed phases of development from conceptualization to hardware implementation, this paper underscores the potential of combining traditional circuit design with modern machine learning algorithms in enhancing digital recognition systems. The implementation showcases how VLSI technology can be effectively leveraged to reduce the size and cost of electronic components while boosting performance and energy efficiency. Looking ahead, there are abundant opportunities for future research and development in this domain. As machine learning algorithms continue to evolve and become more complex, the need for more sophisticated hardware implementations will inevitably increase. Future work may focus on integrating multilayer neural network models, such as convolutional neural networks (CNNs) or artificial neural networks (ANNs), which could further improve the accuracy and efficiency of digital recognition systems. Additionally, exploring the application of these advanced neural network models in FPGA-based hardware designs could lead to broader implications for real-world applications beyond

digit recognition. For instance, such technologies could be applied to enhance image processing in medical diagnostics, real-time video surveillance analysis, or even autonomous vehicle navigation systems, thus broadening the impact of this research into new and innovative areas.

References

- [1] Giardino D., Matta M., Silvestri F., Spanò S., Trobiani V. FPGA implementation of hand-written number recognition based on CNN. *International Journal on Advanced Science, Engineering and Information Technology*, vol. 9, no. 1, 2019, pp. 167-171.
- [2] Lin Z. Hand-Written Number Classification Based on Hardware Neural Network. *2023 IEEE International Conference on Electrical, Automation and Computer Engineering (ICEACE)*, 2023, pp. 1447-1451.
- [3] Xu H., Zhu X., Zhao Z., Wei X., Wang X., Zuo J. Research of Pipeline Leak Detection Technology and Application Prospect of Petrochemical Wharf. *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, 2020, pp. 263-271.
- [4] Zhu X., Zhao Z., Wei X., Wang X., Zuo J. Action recognition method based on wavelet transform and neural network in wireless network. *Proceedings of the 2021 5th International Conference on Digital Signal Processing*, 2021, pp. 60-65.
- [5] Ramzan M., Khan H.U., Awan S.M., Akhtar W., Ilyas M., Mahmood A., Zamir A. A survey on using neural network based algorithms for hand written digit recognition. *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 9, 2018.
- [6] Wang R., Zhu J., Wang S., Wang T., Huang J., Zhu X. Multi-modal Emotion Recognition Using Tensor Decomposition Fusion and Self-supervised Multi-tasking. *International Journal of Multimedia Information Retrieval*, 2024, 13(4): 39.
- [7] Chychkarov Y., Serhiienko A., Syrmamiikh I., Kargin A. Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks. *CMIS*, vol. 2864, 2021, pp. 496-509.
- [8] Hossain M.A., Ali M.M. Recognition of handwritten digit using convolutional neural network (CNN). *Global Journal of Computer Science and Technology*, vol. 19, no. 2, 2019, pp. 27-33.
- [9] Zhao Z., Peng Y., Zhu X., Wei X., Wang X., Zuo J. Research on Prediction of Electricity Consumption in Smart Parks Based on Multiple Linear Regression. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2020: 812-816.
- [10] Oraon P., Mangaraj S., Swain A.K., Mahapatra K. Hardware Accelerated Quantized Hand Written Digit Recognition via High Level Synthesis. *Proceedings of the Great Lakes Symposium on VLSI 2024*, 2024, pp. 688-693.