A Real-time Object-recognition System Based on PCNN Algorithm

Trong-Thuc HOANG† Ngoc-Hung NGUYEN† Xuan-Thuan NGUYEN† Trong-Tu BUI†

†The University of Science, Ho Chi Minh City 227 Nguyen Van Cu St., Dist.5, Ho Chi Minh City, Vietnam
E-mail: †trongthuc.hoang@gmail.com †{nnhung, nxthuan, bttu}@fetel.hcmus.edu.vn

Abstract In this paper, the authors present two hardware architectures of Pulse-Coupled Neural Network (PCNN) which can be applied to real-time object-recognition systems. They are RAM-based model and pipelined model. Both models can generate a time-signal vector at the speed of one vector per video frame. Two architectures have been successfully implemented on FPGA chips. Based on these models, a complete recognition system including a camera, a PCNN core, a search engine and a DVI controller has been built and tested so as to verify the operation of the models. The experiment results show that the percentage of recognition is 92.8%.

Keyword PCNN, Pulse-Coupled Neural Network, Object recognition, Real-time, hardware.

1. Introduction

Pulse-Coupled Neural Network (PCNN) is biologically inspired neural network based on cat's visual cortical neurons. Since introduced by Eckhon in 1990 [1], PCNN model has proven its vital role in digital image processing, such as image segmentation [2], image thinning [3], motion detection [4], pattern recognition [5], face detection [6] etc.

Although the PCNN is suitable for many fields of image processing, it is difficult to achieve real-time processing by software implementation. Hardware implementation is a solution to overcome such an issue. Javier et al [7] presented an FPGA system that can operate at high speed, but it is only suitable for those applications that require one-time operation. In the object recognition system, it is required iterative processing of PCNN. Ranganath et al [8] proposed such a hardware design, but it needs multiple PCNN layers and dynamic parameters. In our previous work [9], an implementation using PCNN algorithm was proposed for feature extraction at high speed. However, the hardware time-signal vector generated by hardware was not perfectly identical to the ideal vector generated by software implementation. In this paper, we present two PCNN hardware architectures which can generate time-signal vector highly similar to the ideal one. They are RAM-based model and pipelined model. The RAM-based model utilizes RAMs to simulate the neurons network. By an N-cycle processing, the RAM-based model generates one N-element time-signal vector, which can be used asa feature vector. The pipelined model divides the operation into N stages; each stage generates one of N elements. The resource costs of RAM-based model depend on the image size and that of the pipelined model mainly depend on the number element of time-signal vector. Both models can generate time-signal vector at the speed of one vector per video frame. Based on these architectures, a real-time object-recognition system has been built, including camera, PCNN core, search engine, and DVI controller. The system can recognize an object based on the image time-signal vector generated by the PCNN core. Each image generates a unique vector. However, if two images contain similar objects, they will generate similar vectors.

The remainder of this paper is organized as follows. Section 2 briefly reviews the PCNN model and shows an implementation of a neuron. Section 3 presents the object-recognition system. Section 4 proposes two PCNN hardware architectures. Section 5 shows the experiment results of two models and the system performance. The conclusion is given in section 6.

2. The PCNN model

2.1. A neuron structure

A single neuron structure is shown in Fig. 1 (Fig. 1 in [10]). The neuron consists of three parts: feeding field (the receptive field or dendritic tree in some references), linking modulation, and pulse generator. The PCNN model is described as iteration by the following equations.

$$F_{ij}[n] = S_{ij} + F_{ij}[n-1]e^{-\alpha_F} + V_F \sum_{kl} M_{ijkl} Y_{kl}[n-1] \quad (1)$$

$$L_{ij}[n] = L_{ij}[n-1]e^{-\alpha_L} + V_L \sum_{kl} W_{ijkl} Y_{kl}[n-1] \quad (2)$$

$$U_{ij}[n] = F_{ij}[n]\left(1 + \beta L_{ij}[n]\right) \quad (3)$$

$$T_{ij}[n] = T_{ij}[n-1]e^{-\alpha_T} + V_T Y_{ij}[n-1] \quad (4)$$

$$Y_{ij}[n] = \begin{cases} 1 \rightarrow U_{ij}[n] > T_{ij}[n] \\ 0 \rightarrow otherwise \end{cases} \quad (5)$$

According to those equations, the operation of a single neuron at step n depend on its corresponding pixel (S_{ij}), previous pulses from surrounding neurons ($Y_{kl}[n-1]$), and the last internal activities of the neuron itself ($F_{ij}[n-1]$, $L_{ij}[n-1]$, and $T_{ij}[n-1]$). M and W are two synaptic weight matrixes. V_F, V_L, V_T, α_F, α_L, α_T, and β are the PCNN parameters and should be considered as constants.

The time-signal vector can be generated by summing the number of pulses at the output of PCNN, which is shown by the equation (6), where n is the iteration step. With N steps, we have the N elements time-signal vector (vector G), is unique for each image.

$$G[n] = \sum_{ij} Y_{ij}[n] \quad (6)$$

2.2. A neuron implementation

If the field of surrounding neurons is too large, the PCNN loses its segmentation quality, and the time-signal vector is going to saturate in several steps. With saturated vector, there is difficult to identify the difference among images. However, choosing a small field will reduce the feature extraction ability of PCNN. A matrix of 3x3 surrounding neurons is the appropriated field. Two synaptic weight matrixes M and W are selected equally.

The top-view of hardware implementation for a neuron is shown in Fig. 2. The input is 8-bit grayscale image. The signal iS_data is the corresponding pixel. iY_data are the pulses from surrounding neurons and iF_data, iL_data and iT_data represent for the internal variables in the previous step. oY_data transmits the neuron pulse while the others transmit the internal variables. The details of the implementation are shown in Fig. 6. These multiple parameters aF, aL and aT represent for $\exp(-\alpha_F)$, $\exp(-\alpha_L)$ and $\exp(-\alpha_T)$ in above equations, respectively.

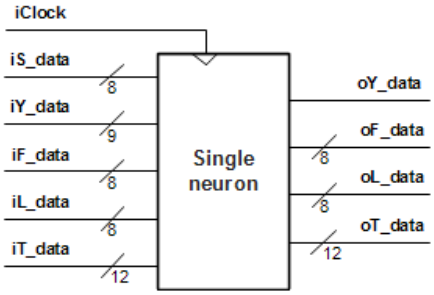

Fig. 2. Top-view of a neuron implementation.

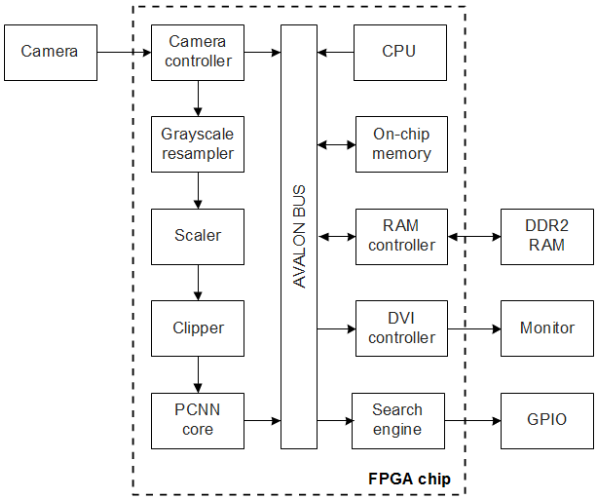

Fig. 3. The system block diagram.

3. The object-recognition system

The block diagram of the object-recognition system has shown in Fig. 3. A camera transmits the image data to the system with 640x480 image size and 24-bit color per pixel. The camera controller writes the image data into DDR2 RAM and Grayscale resampler by its memory-mapped interface and streaming interface, respectively. The DVI controller reads raw image data from DDR2 RAM and displays them on a monitor. Grayscale resampler converts 24-bit color to 8-bit grayscale. Scaler and clipper reduce the image size into 128x128 pixels. After this, the PCNN core extracts time-signal vector based on the received

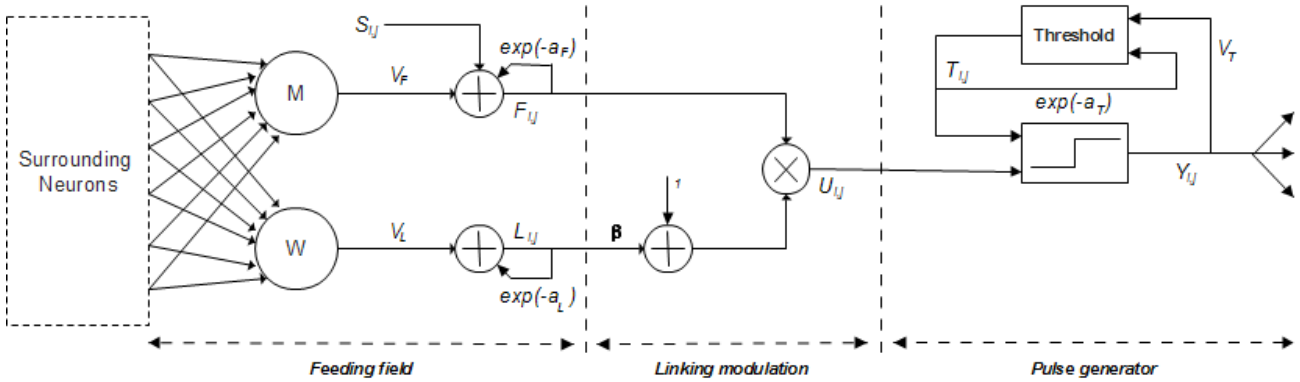

Fig. 1. A neuron structure.

image, and then transmits it to search engine through Avalon bus. The search engine computes the image entropy by Equation (7) and (8). The search engine also consist the templates which contain sample object vectors. The comparison between the computed entropy and the templates is done by the MSE values. The MSE values are computed by Equation (9). As soon as the search engine had finished its work, it notices the results by GPIO (i.e. LEDs, LCD, etc.).

$$p_n = \frac{G[n]}{P_{HV}} \tag{7}$$

$$E_n(p) = -p_n log_2 p_n - (1-p_n) log_2 (1-p_n) \tag{8}$$

$$MSE = \frac{1}{N}\sum_n [E_n(p) - E'_n(p)]^2 \tag{9}$$

where, $G[n]$ is the vector, P_{HV} is the number of total image pixels, p_n is the probabilities when neurons are activated, i.e. $Y_{ij}[n] = 1$, $E_n(p)$ is the image entropy, and $E'_n(p)$ is the sample entropy, respectively.

4. Proposed hardware architectures

The 8-bit grayscale image has HxV resolution. P_{HV} denotes the number of the total pixels in the image, i.e. H*V. The output vector G has N elements.

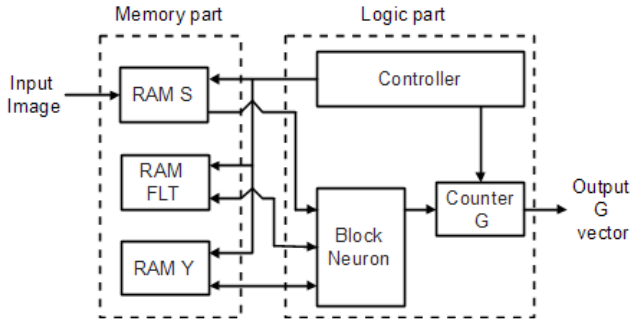

Fig. 4. PCNN RAM-based block diagram.

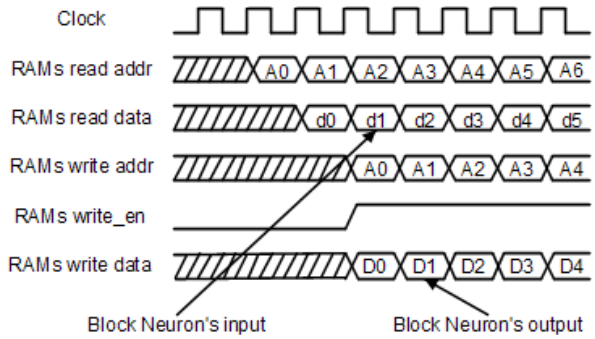

Fig. 5. The operation of RAMs.

4.1. RAM-based architecture

PCNN model is a two-dimensional network with a 1:1 corresponding between image pixels and network neurons. Instead of implementing a network that contains P_{HV} node neurons, the cost can be reduced by some neurons for computing and RAMs for storing the variables of the network while the system processing.

The system is mainly composed of two parts: the memory part (MP) and the logic part (LP) as shown in Fig. 4. MP contains three RAMs to stores the input image (RAM_S), the internal variables (RAM_FLT) and the output binary image (RAM_Y). Each RAM has the same amount of memory cells, i.e. P_{HV} cells. However, each RAM has the different cell width. RAM_S, RAM_FLT, and RAM_Y have 8 bits, 28 bits and 1 bit per memory cell, respectively. The total RAM capacity can be calculated in Equation (10). In LP, the controller controls the RAM address lines while RAM data lines are connected to the block neuron. Counter G counts the pulse produced by the block neuron to computes the point value.

$$B_{RAM} = 37HV\ (bit) \tag{10}$$

The operation divided into two phases: inactive phase and active phase. In the inactive phase, the video frame is written into RAM_S. When the writing process is finish,

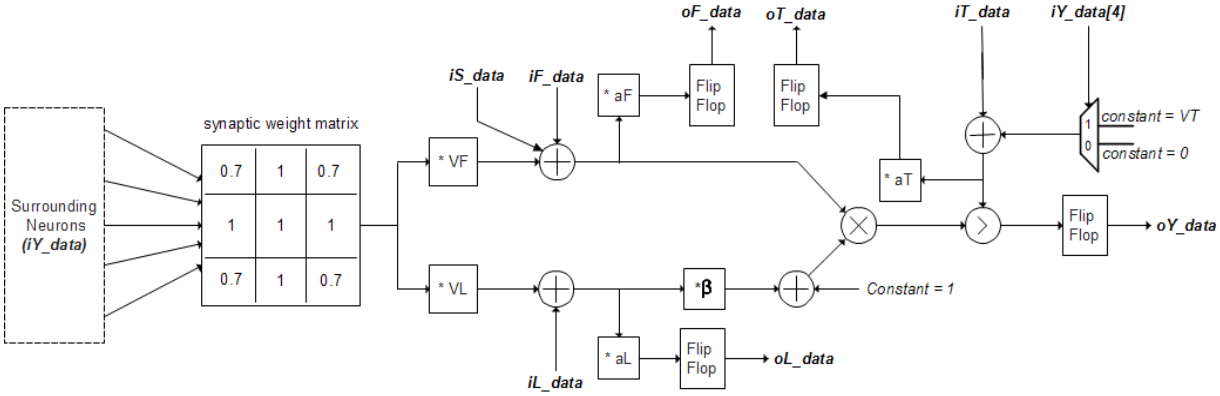

Fig. 6. A neuron implementation

the system turns into active phase to generate a vector G. To avoid the overlapping of adjacent frames, RAM_S has to be read-only during the active phase. RAM_FLT and RAM_Y have the initial value of zeros. There are P_{HV} nodes in the network and their variables are stored in RAMs. LP accesses to RAMs and iterates the processing N cycles in order to generate an N-element vector G. In the basic model, the block neuron is considered as a single neuron. With one neuron, LP can process one node per clock cycle. The details of communication between LP and RAMs are described in Fig. 5. A_i are the address of node[i]. d_i are the last data of node[i], and D_i are the new data of node[i] that will be written back into RAMs. After P_{HV} clock cycles, LP generates one of N elements of vector G. After N-cycle processing, the system generates a full vector G, and the active phase is completed. The system returns to the inactive phase.

Since block neuron contains a single neuron, LP can process only one node per clock cycle. With a group of neurons implemented in block neuron instead of one, the system can speed up the processing time. For example, if block neuron includes a matrix of 2x2 neurons, the system can lower the processing time four times. The LP resource costs are affected by the number of neurons in block neuron. However, the total RAM capacity in the MP remains constantly. The number of neurons in block neuron depends on the length of vector G. In order to generate vector at the speed of one vector per frame, the number of neurons has to be N.

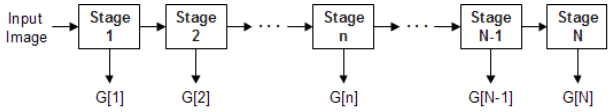

Fig. 7. The PCNN system with N stages pipelined.

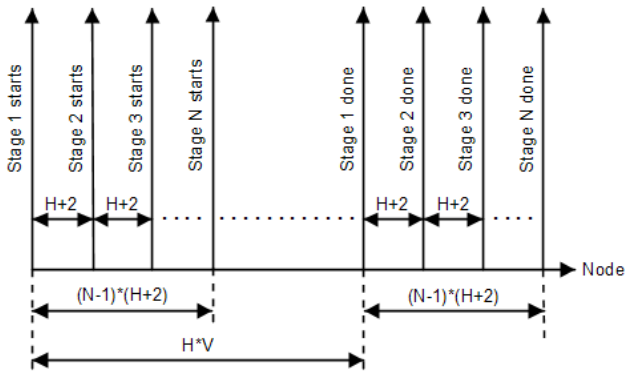

Fig. 8. The processing times of N stages are alternated.

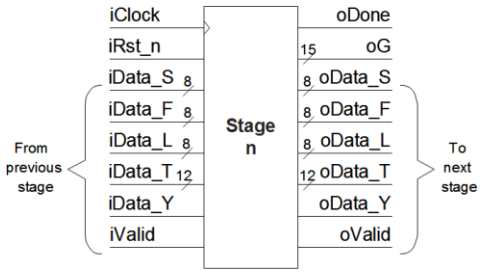

Fig. 9. Top-view of a general stage design.

4.2. Pipelined architecture

Pipelined model is constructed based on the separation of the operation into N independent stages, each stage computes one of N elements vector G.

In order to achieve its goal, a stage needs the information of all node neurons in the network from the previous stage and the input image pixel. The top-view of one stage is shown in Fig. 9. At every clock when iValid is asserted, a stage receives one node information including pixel values (i.e. iData_S), internal variables (i.e. iData_F, iData_L, iData_T), and pulse (i.e. iData_Y). As soon as a stage has finished its work, it asserts oDone signal. The oG signal carries the final point value. The first and the last stage are designed differently with the common design in Fig.9. Owing to the initial value of neuron variables and pulse are zeros, stage 1 needs only the input pixel value to operate. The final stage has not to transmit its network information, and then the output signals are only oDone and oG. Fig. 7 describes the system with N-stage pipelining.

$A_{i,j}[n]$ represents a node in the network where $i \in (1,H)$ is the column number, $j \in (1,V)$ is the row number, and $n \in (1,N)$ is the number of stage. In order to operate, node $A_{i,j}[n]$ requires the pixel value ($S_{i,j}$), internal variables ($F_{i,j}[n-1]$, $L_{i,j}[n-1]$, $T_{i,j}[n-1]$), and pulses from 3x3 matrix surrounding neurons ($Y_{i-1,j-1}[n-1]$, $Y_{i-1,j}[n-1]$, $Y_{i-1,j+1}[n-1]$, $Y_{i,j-1}[n-1]$, $Y_{i,j}[n-1]$, $Y_{i,j+1}[n-1]$, $Y_{i+1,j-1}[n-1]$, $Y_{i+1,j}[n-1]$, $Y_{i+1,j+1}[n-1]$). A stage starts the processing with node $A_{1,1}[n]$ and finishes with node $A_{H,V}[n]$. Node $A_{1,1}[n]$ is at the corner of the network which has four surrounding neurons ($Y_{1,1}[n-1]$, $Y_{1,2}[n-1]$, $Y_{2,1}[n-1]$, $Y_{2,2}[n]$) instead of nine ones. Stage n starts the processing whenever it has enough information to process its first node, so stage[n] has to wait for stage[n-1] first (H+2) nodes. The processing times of N-stage are alternated as shown in Fig. 8.

The system is ready to receive a new input image after stage 1 done, and one vector G is generated each time stage N done. A stage needs buffers to store (H+2) nodes information results from deviation between two adjacent

stages processing time. There are five buffers to store pixel values, internal variables and pulses as shown in Fig. 10. Buffers are constructed by shift registers. The pixel values and internal variables buffers utilize (H+2) registers for each buffer. The pulses of surrounding neurons are 3x3 matrixes leading to three lines buffers are required for storing them. With the different data width of $S_{i,j}$, $F_{i,j}$[n-1], $L_{i,j}$[n-1], $T_{i,j}$[n-1], and Y[n-1], the total buffers capacity between two adjacent stages can be calculated by Equation (11). The entire buffers capacity in the system can be calculated by Equation (12).

$$B = 39H + 72\ (bit) \tag{11}$$

$$B_{Pipe} = (N-1) * B = (N-1) * (39H + 72)\ (bit) \tag{12}$$

Fig. 10 describes a stage design of including buffers, one PCNN neuron, controller, and counter. By counting the iValid signal, the controller knows the status of the previous stage. The controller generates the oValid signal and controls the internal activities of the stage. Counter count the pulse carried by oData_Y output signal to computes the point value of vector G. Counter reset by the controller each time the stage done. Because the first stage needs only input pixel values to operate, its input signals are only iData_S and iValid. Therefore, there are no buffers in the first stage. The last stage is similar with common design in Fig. 10, but the neuron output signal is only oY_data, and the controller does not need to generate the oValid signal.

Table 2. The PCNN parameters.

Parameter	$\propto_F$	$\propto_L$	$\propto_T$	V_F	V_L	V_T	β
Value	0.5	1	1	0.02	0.02	10	0.2

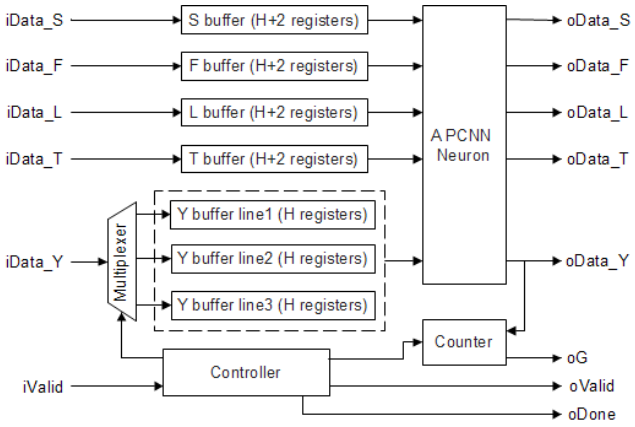

Fig. 10. A stage design.

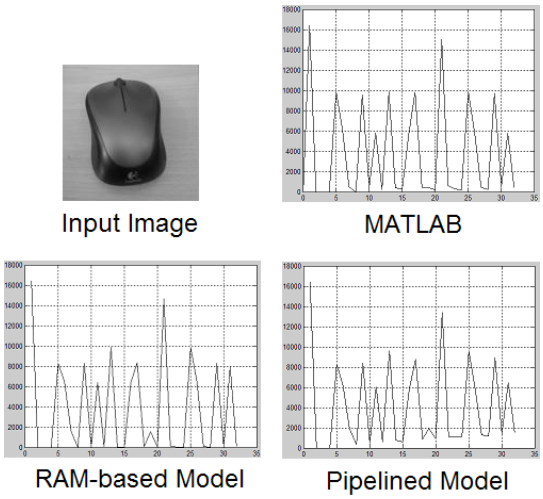

Fig. 11. The comparison between vectors.

5. Experiment results

Both two models have the same processing speed regarding the resource costs. The pipelined model resembles the RAM-based model in that the number of implemented PCNN neuron is the same. The difference of resource costs came from Equation (10) and Equation (12). As can be seen in two equations, the resource costs of RAM-based model depend on the image resolution and that of the pipelined model mainly depend on the element number of vector G. Based on requirements of image resolution and vector length, the better model can be selected by above two equations.

Table 1. Hardware usage results of the PCNN cores and search engine.

	RAM-based model	Pipelined model	Search Engine
ALUTs	12,739	9,423	234
Register	1.84 KB	17.5 KB	7 B
Memory (KB)	72	0	0.44
DSP block	86	38	8
F_{max} (MHz)	54.67	70.61	146.49

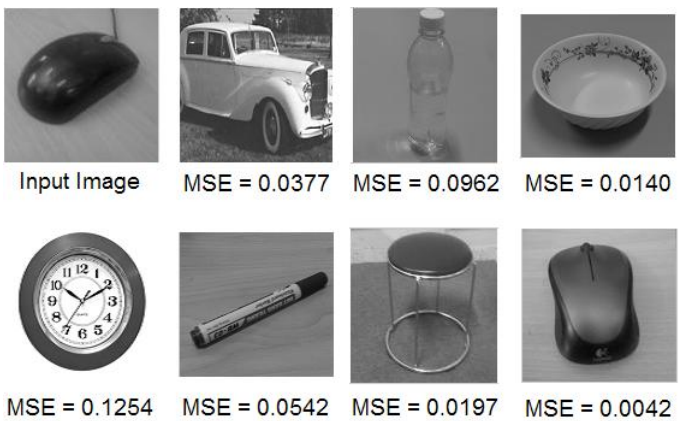

Fig. 12. The MSE values between the input image and the sample images.

Two models are verified by the Modelsim simulator and implemented on Altera Stratix III board with an EP3SL150F1152C2 FPGA chip. The system parameters H, V, and N are 128, 128, and 32, respectively. It means that the image size is 128x128, and time-signal vector length is 32 elements. Tab. 1 shows the resource utilization of two architectures and search engine. From the Tab. 2, PCNN parameters are selected for object-recognition applications. Fig. 11 shows the comparison between three vectors: the ideal vector is calculated by Matlab and others are obtained from hardware designs. It is clear that both models generate vectors which are highly similar to the ideal one.

An image has a unique vector which represents for the object in that image. As mentioned above, images contain similar objects will generate similar vectors. The comparison between vectors is done by the MSE value. The computation of MSE value is shown in Equation (9). In object-recognition system, samples vectors are stored in the templates. Receiving an input image from camera, the MSE values can be calculated for each pair of the input image and the sample one. If the input image has an object which is similar with any sample image in the templates, their MSE value will smallest. Therefore, the system can recognize the object by finding the smallest MSE value via winner-take-all circuit. Fig. 12 shows an input image with seven sample images. It is clear that the MSE value between the input image and the computer mouse sample image is the smallest one. Then, the system can recognize the input image as a computer mouse. However, if the input image contains an object which is not in the templates, the system has to dispose of that image. For this reason, an MSE threshold is needed. The system can dispose of the input image if the smallest MSE value is still over the threshold. The MSE threshold is chosen as via experiment.

A completed system has been built successfully on the Stratix III FPGA board. As can be seen in the Fig. 13, the TV's top-left corner contains a vector which represents for the computer mouse in front of the camera.

Fig. 13. Completed system.

6. Conclusion

We have introduced two PCNN hardware architectures and applied them to a real-time object-recognition system. Both two models can generate a time-signal vector at the speed of one vector per frame. The resource costs of RAM-based model and pipelined model primary depend on the image resolution and the time-signal vector length, respectively. Based on the specify application requirements, the better model can be selected. A real-time object-recognition system has been built successfully on FPGA board based on two architectures. The experiment results show that the percentage of recognition is 92.8%.

References

[1] R. Eckhorn, H.J. Reitboeck, M. Arndt, and P.W. Dicke, Feature linking via synchronization among distributed assemblies: Simulation of results from cat cortex, Neural Computation, Vol.2, No.3, pp. 293-307, Fall 1990.

[2] Henrik Berg, Roland Olsson, Thomas Lindblad, and José Chilo, Automatic design of pulse coupled neurons for image segmentation, Neurocomputing, Vol.71, Issues 10-12, pp. 1980-1993, June 2008.

[3] Lifeng Shang, and Zhang Yi, A class of binary images thinning using two PCNNs, Neurocomputing, Vol.70, Issues 4-6, pp. 1096-1101, January 2007.

[4] Jun Chen, Kosei Ishimura, and Mitsuo Wada, "Moving Object Extraction Using Multi-Tiered Pulse-Coupled Neural Network," Proc. SICE Annual Conf., Vol.3, pp. 2843-2848, Sapporo, Japan, August 2004.

[5] Raul C. Muresan, Pattern recognition using pulse-coupled neural networks and discrete Fourier transforms, Neurocomputing, Vol.51, pp. 487-493, April 2003.

[6] Hitoshi Yamada, Yuuki Ogawa, Kosei Ishimura, and Mitsuo Wada, "Face Detection using Pulse-Coupled Neural Network," Proc. SICE Annual Conf., Vol.3, pp. 2784-2788, Fukui, Japan, August 2003.

[7] Javier Vega-Pineda, Mario I. Chacón-Murguía, and Roberto Camarillo-Cisneros, "Synthesis of Pulse-Coupled Neural Networks in FPGA for Real-time Image Segmentation," Proc. Int. Joint Conf. Neural Networks, pp. 4051-4055, Vancouver, Canada, July 2006.

[8] H.S. Ranganath, and G. Kuntimad, Object Detection Using Pulse Coupled Neural Networks, IEEE Trans. Neural Networks, Vol.10, Issue 3, pp. 615-620, May 1999.

[9] Nguyen Ngoc Hung, Nguyen Xuan Thuan, Huynh Huu Thuan, and Bui Trong Tu, "A Hardware Implementation of Pulse Coupled Neural Network for Image Feature Extraction," Proc. Integrated Circuits and Devices in Vietnam (ICDV), pp. 92-97, Hanoi, Vietnam, August 2011.

[10] Zhaobin Wang, Yide Ma, Feiyan Cheng, and Lizhen Yang, Review of pulse-coupled neural networks, Image and Vision Computing, Vol. 28, Issue 1, pp. 5-13, January 2010.