1-17-2013

Gesture Recognition Using Neural Networks Based on HW/SW Cosimulation Platform

Priyanka Mekala
Department of Electrical and Computer Engineering, Florida International University, Miami, FL

Jeffrey Fan
Department of Electrical and Computer Engineering, Florida International University, Miami, FL, Jeffrey.Fan@fiu.edu

Wen-Cheng Lai
Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan: Department of Engineering, Ming Chi University of Technology, Taipei 243, Taiwan

Ching-Wen Hsue
Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan

Recommended Citation
Priyanka Mekala, Jeffrey Fan, Wen-Cheng Lai, and Ching-Wen Hsue, "Gesture Recognition Using Neural Networks Based on HW/SW Cosimulation Platform," Advances in Software Engineering, vol. 2013, Article ID 707248, 13 pages, 2013. doi:10.1155/2013/707248

Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2013, Article ID 707248, 13 pages
http://dx.doi.org/10.1155/2013/707248

Research Article

Gesture Recognition Using Neural Networks Based on HW/SW Cosimulation Platform

Priyanka Mekala,[1] Jeffrey Fan,[1] Wen-Cheng Lai,[2,3] and Ching-Wen Hsue[2]

[1] *Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA*
[2] *Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan*
[3] *Department of Engineering, Ming Chi University of Technology, Taipei 243, Taiwan*

Correspondence should be addressed to Priyanka Mekala; pmeka001@fiu.edu

Received 30 July 2012; Revised 27 December 2012; Accepted 17 January 2013

Academic Editor: Christine W. Chan

Hardware/software (HW/SW) cosimulation integrates software simulation and hardware simulation simultaneously. Usually, HW/SW co-simulation platform is used to ease debugging and verification for very large-scale integration (VLSI) design. To accelerate the computation of the gesture recognition technique, an HW/SW implementation using field programmable gate array (FPGA) technology is presented in this paper. The major contributions of this work are: (1) a novel design of memory controller in the Verilog Hardware Description Language (Verilog HDL) to reduce memory consumption and load on the processor. (2) The testing part of the neural network algorithm is being hardwired to improve the speed and performance. The American Sign Language gesture recognition is chosen to verify the performance of the approach. Several experiments were carried out on four databases of the gestures (alphabet signs A to Z). (3) The major benefit of this design is that it takes only few milliseconds to recognize the hand gesture which makes it computationally more efficient.

1. Introduction

In today's world, the field programmable gate array (FPGA) technology has advanced enough to model complex chips replacing custom application-specific integrated circuits (ASICs) and processors for signal processing and control applications. FPGAs are preferred as higher-level tools evolve to deliver the benefits of reprogrammable silicon to engineers and scientists at all levels of expertise. Taking advantage from the current FPGA technology, this paper proposes a hardware/software cosimulation methodology using hardware description language (HDL) simulations on FPGA as an effort to accelerate the simulation time and performance [1, 2].

The conventional software simulation method has more flexibility in terms of parameters variation. The desired simulation parameters can be changed to study the system behavior under various conditions. The major drawback with the conventional approach is the intolerable simulation time. On the other hand, the complete hardware-based approach can provide significant speedup in examining the system behavior, but the flexibility will be nonetheless compromised. In this paper, we attempt to leverage the merits of the software simulation and hardware emulation to retain both the flexibility and performance by adopting a HW/SW-based platform approach [1].

2. Background/Literature Review

The communication between human and machines or between people can be done using gestures called sign language [3]. The use of sign language plays an important role in the means of communication method for the hearing-impaired community [4]. American Sign Language (ASL) is the 3rd most-used language and the choice for most deaf people in the United States. 500,000 and 2,000,000 people use sign language as their major daily communication tool. It seems that 3.68% of the total population is found to be hard of

hearing and 0.3% of the total population is functionally deaf, out of a total population of about 268,000,000 (2005) in the US [5].

Gesture recognition is generally based on two different approaches. Primarily, glove-based analysis [6–8] where either mechanical or optical sensors are attached to a glove that transforms finger flexions into electrical signals to determine the hand posture [7]. Currently, the vision-based analysis [9–11] is used mostly, which deals with the way human beings perceive information about their surroundings. The database for these vision-based systems is created by selecting the gestures with predefined meaning, and multiple samples of each gesture are considered to increase the accuracy of the system [10]. In this paper, we have used the vision-based approach for our gesture recognition application.

Several approaches have been proposed previously to recognize the gestures using soft computing approaches such as artificial neural networks (ANNs) [12–16], fuzzy logic sets [17], and genetic algorithms [18]. Some statistical models [6] used for gesture recognition include Hidden Markov Model (HMM) [19, 20] and Finite-State Machine (FSM) [21]. ANNs are the adaptive self-organizing [22, 23] technologies that solved a broad range of problems such as identification and control, game playing and decision making, pattern recognition medical diagnosis, financial applications, and data mining [23, 24] in an easy and convenient manner [25, 26].

Murakami and Taguchi in [12] presented the Japanese Sign Language recognition using two different neural network systems. Back Propagation algorithm was used for learning postures, taken using data gloves, of Japanese alphabet. The system is simple and could successfully recognize a word. The proposed automatic sampling and filtering data proved to help improve the system performance. However, the learning time of both network systems was extremely high varying from hours to days. Maung [13] used the real-time 2D hand tracking to recognize hand gestures for Myanmar Alphabet Language. The system was easy to use and no special hardware was required. The input images were acquired via digitized photographs and the feature vector obtained using histograms of local orientation. This feature vector served as the input to the supervised neural networks system built. Implementing the system in MATLAB tool box made the work easy because of the simplicity in design and easy use of toolbox. However, the tradeoff was the speed in execution time as the time consumed for implementation was high compared to that of other languages.

Bailador et al. [15] presented Continuous Time Recurrent Neural Networks (CTRNNs) real-time hand gesture recognition system. The work was based on the idea of creating specialized signal predictors for each gesture class [15]. The standard Genetic Algorithm (GA) was used to represent the neuron parameters, and each genetic string represents the parameter of a CTRNN. The system is fast, simple, modular, and a novel approach. High recognition rate of 94% is achieved from testing the dataset. This system is limited by the person's movements and activities which caused a higher noise that has significant effect on the results. The dependency of segmentation operation on the predictor proved to greatly affect the segmentation results. Stergiopoulou and Papamarkos [16] presented static hand-gesture-recognition-based Self-Growing and Self-Organized Neural Gas (SGONG) Network. Digital camera was used for input image. For hand region detection, YCbCr color space was applied and then threshold technique used to detect skin color. SGONG proved to be a fast algorithm that uses competitive Hebbian learning algorithm (the learning starts with two neurons and grows) in which a grid of neurons would detect the exact shape of the hand. However, the recognition rate was as low as 90.45%.

In all the previous works mentioned, they are purely software-based approaches. In order to increase the speed of execution while maintaining the flexibility, we propose an HW/SW approach. General software that can perform gesture recognition is MATLAB, Microsoft Visual C#, Microsoft Visual C++, and Microsoft Visual Basic. The most common software used are MATLAB and Microsoft Visual C# both of which are very powerful tools. MATLAB is chosen over others because it is a high-performance language for technical computing, perfect for speeding up development process as it allows the user to work faster with the features of toolboxes and provides ease and flexibility to the design of the programming. It is a tool of choice for high-productivity research, analysis, and development. It is preferred based on the previous research developed by Maung [13], Maraqa and Abu-Zaiter, [14] and on the review in [25, 26]. It integrates programming, computation, and visualization in user-friendly environment where problems and solutions are presented in common mathematical notation [27]. Also in industry, MATLAB's tool is widely used for high-productivity research, development, and analysis [27].

FPGAs have the advantage of hardware parallelism performing more like concurrent execution of functions. Thus by breaking the paradigm of sequential execution and accomplishing more per clock cycle, the FPGAs exceed the computing power of digital signal processors (DSPs) [28]. Mostly, inputs and outputs of the applications are controlled at the hardware level in order to provide faster response times, and dedicated parts of hardware reduce the power consumption for few complex operations. The programmable silicon indicates no fabrication costs or long lead times for assembly [28]. In general, the software tools provide the programming environment whereas the FPGA-based design is a completely hardware implementation of the application. Processor-based systems often involve several layers of abstraction to help schedule tasks and share resources between multiple processes [28]. The driver layer controls hardware resources and the operating system manages memory and processor bandwidth. FPGAs minimize the reliability concerns with true concurrent execution and dedicated hardware since they do not use the operating system [28].

3. Gesture Recognition Algorithm

The gesture recognition system is mostly classified into three major steps after acquiring the input image from camera(s), videos or even data glove instrumented device. These steps are preprocessing, features estimation and extraction, and classification or recognition [25, 26]. The pre-processing

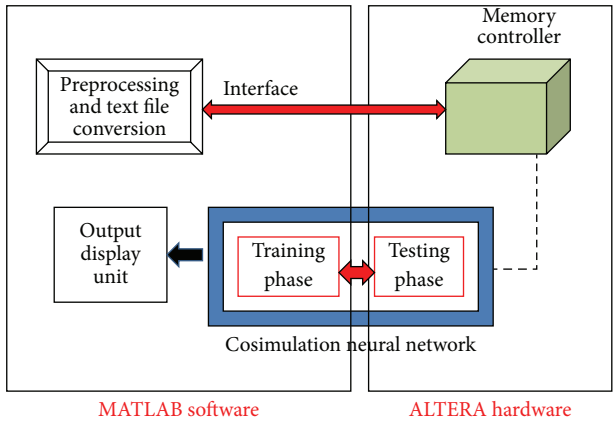

Figure 1: System architecture of gesture recognition on hardware/software cosimulation platform.

module of the system developed involves the application of morphological operations and edge detection in order to segment the image into regions of interest. The feature extraction is a very crucial step in the gesture recognition algorithm. Feature vector of the segmented image can be extracted in different ways depending on the application. Different methods are presented that use the skin color, hand shape, hand contour, fingertips position, palm center, and so forth. In our design, the hand shape and magnitude orientation is being considered in order to obtain the feature vector. Thus, the results are not dependent on the skin color or the background of the image gestures obtained.

In our approach, the back propagation is used to train and test the neural network on the cosimulation platform. The interface among the modules of the neural network eases data transfer and controls signals communication. The training phase of the network is done on software platform and the testing phase is done on hardware platform. The data is stored in the memory controller module created using VERILOG HDL and has bus interface for reading and writing back the data to the memory.

Figure 1 shows the system architecture being developed in this paper. MATLAB is used as the software platform and ALTERA-ModelSim 6.3 g_p1 is used as the hardware platform. QUARTUS II (8.1 web edition) design flow is used to simulate and verify the functionality of HDL code. Xilinx ISE 10.1 is used to understand the device and logic utilization, memory design, and test control of the architecture developed.

3.1. Database Generation. The images for database are captured using a Cannon camera which produces image frames of RGB pixels. The ASL alphabet gestures are used to obtain the hand images. The database contains two different subjects with two different backgrounds. Figure 2 shows few of the database sets used in this particular application. The Cannon camera is actually not stationary since it is not in a fixed position but the background is maintained as either black or white. The images are stored in .jpg format. The temporal resolution requirements of the application have to be considered. The size of the image frame is 640 × 480 pixels. To maintain the resolution and decrease redundancy, the frames are resized to 64 × 64 pixels. Once the input images are obtained, they are converted to grayscale values within a range of 0–255.

The database obtained is read on the computer using MATLAB. The software converts the entire database image frames into text files. These files are stored in the memory of the ALTERA-ModelSim using command "$readmemh." The ALTERA-ModelSim is being called from the MATLAB using HDLDAEMON as shown in Figure 3. HDLDAEMON controls the server that supports interactions with HDL simulators. To communicate with HDL simulators, the server uses one of two interprocess communication methods: shared memory (which is the default) or TCP/IP sockets. Communication through shared memory is usually faster than that of sockets. When using shared memory, the server can communicate with only one HDL simulator at a time, and the HDL simulator must be running on the same host. When using sockets, the server can communicate with multiple HDL simulators simultaneously, running on the same host or other hosts [29].

3.2. Text File Conversion. The grayscale images are converted into text files which contain the hexadecimal value of the pixels. Each image is configured and stored with the appropriate configuration ID. The text file contains 4096 values to be read and stored onto the memory of the ALTERA-ModelSim.

3.3. Memory Controller on ALTERA-ModelSim. The text files obtained from the image files are stored onto another file in the memory locations of the ALTERA-ModelSim. The files contain data represented by hexadecimal values and hence contain 16 digits of length. For every negative edge of the clock cycle, the data is read into the memory location. Figure 4 displays the simulation waveform of the memory controller for the database sets. The value *test_design/k* shows the database set images ranging from 1 to N (the last image depending on the number of database sets considered).

4. Cosimulation Neural Network

Neural networks are based on the parallel architecture of neurons present in human brain. It can be defined as a multiprocessor system with very high degree of interconnections and adaptive interaction between the elements. The choice of neural networks to recognize the gestures automatically is due to the following aspects like adaptive learning (using a set of predefined database sets), self-organization from the training module, real-time operation with parallel computations, and high fault tolerance capability [30, 31].

This paper focuses on recognizing static hand gesture images. Since static hand postures not only can express some concepts, but also can act as special transition states in temporal gestures recognition, thus estimating that static hand postures play an important role in gesture recognition applications. A gesture recognition system takes an image as an input, processes it using a transform that converts the image into a feature vector, which will then be compared with the

Figure 2: American sign language Database.

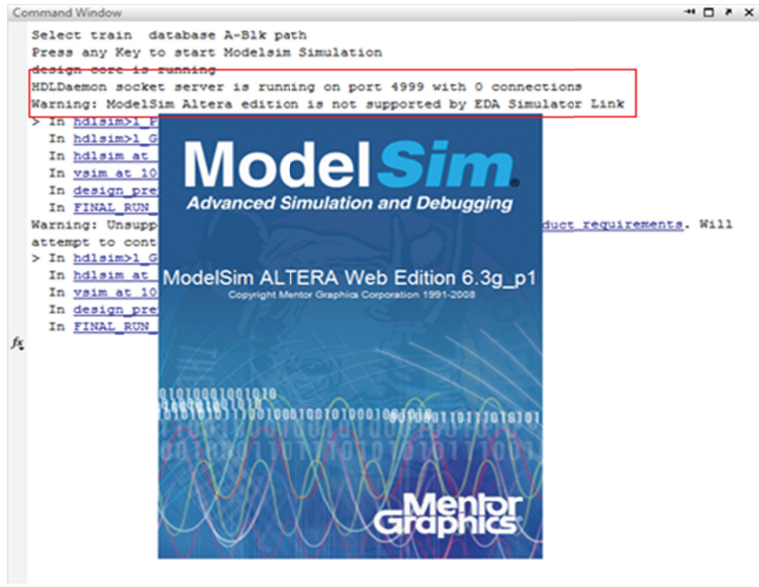

Figure 3: HDLDAEMON as the interface between MATLAB and ALTERA-ModelSim.

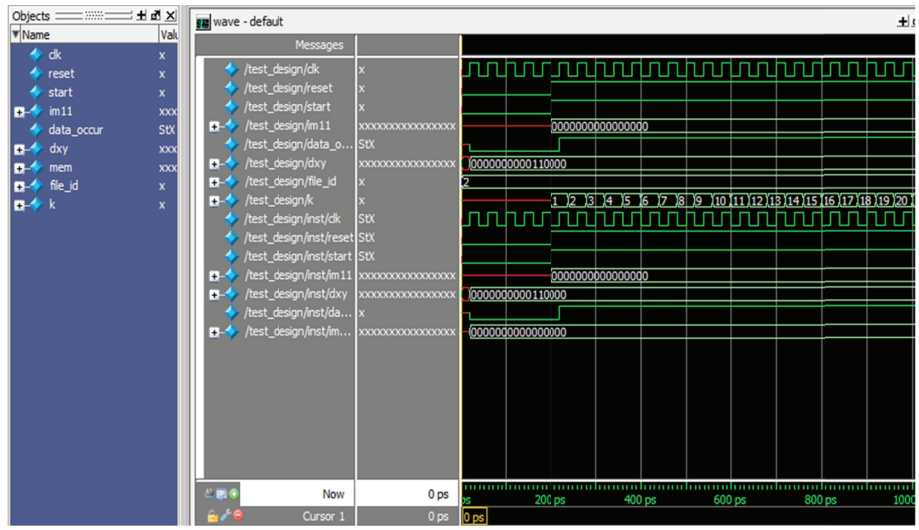

Figure 4: Simulation output waveform of memory design controller.

feature vectors of a training set of gestures. A new technique called cosimulation neural network is being adopted. In this method, a part of the neural network is designed on the hardware with dedicated ports. An interface is being introduced among different levels of the neural network to communicate with one another on two different platforms. A simple neural network model is shown in Figure 5 which consists of input layer, hidden layers, and the output layer with different number of neurons (R, S) in each [32].

Our network is built of 16 input neurons in the input layer, 50 neurons in the first hidden layer, 50 neurons in second hidden layer, and 35 neurons in the output layer. The number of neurons selected resulted from the analysis of the features. Since the features extracted from the image to be used for recognition was 16 as discussed later in the Section 4.1, the input layer has 16 neurons. The output is displayed as a visual

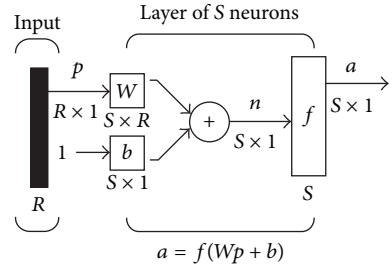

Figure 5: Neural network model [32, 37].

representation of the gesture image and hence is a 7 × 5 (35 neurons) grid display of rows and columns. Since the images are not linearly separable, the hidden layers are necessary

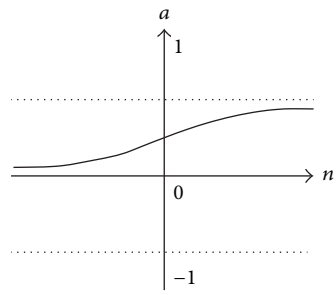

Figure 6: Log-sigmoid transfer function [32].

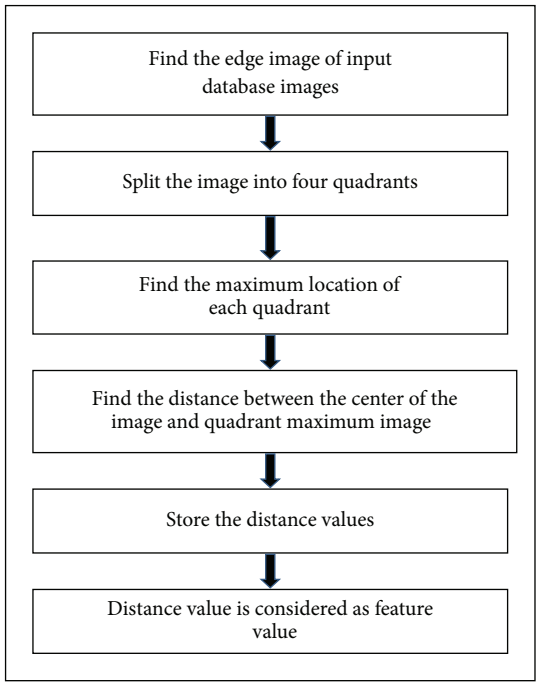

Figure 7: Feature Extraction Algorithm for distance values.

for accurate recognition of the gestures. The neural networks with two hidden layers can represent functions with any kind of shapes. It can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy [33]. Regarding the neurons present in the hidden layers, Geman et al. [34] discuss that there are methods for determining the correct number of neurons to use, such as the following: The number of hidden neurons should be between the size of the input layer and the size of the output layer. The number of hidden neurons should be 2/3 the size of the input layer plus the size of the output layer. The number of hidden neurons should be less than twice the size of the input layer [33]. Hence with some analysis of these methods, the number of hidden neurons is being estimated to be taken as the 2/3 the size of the input layer plus the size of the output layer (i.e., 2/3 times of 16 + 35) which results in 34 neurons and less than twice the output size, that is, 70 neurons. Lawrence et al. [35] have shown that increasing the number of weights makes it easier for standard back propagation to find a good optimum using oversize networks as it can reduce both training error and generalization error. Hence, the choice of the number of hidden layer neurons was made to be 50 neurons with the above-mentioned criteria and some trial and error computations.

The transfer function (f) may be a linear or a nonlinear function. A particular transfer function is chosen based on the specific requirement of the application. A log-sigmoid function, also known as a logistic function, is used as the transfer function in the network designed. The relationship is as follows:

$$\sigma(t) = \frac{1}{1 + e^{-\beta t}}. \quad (1)$$

In the equation above, β is a slope parameter. This is called the log-sigmoid transfer function depicted in Figure 6. The log-sigmoid transfer function is commonly used in multilayer neural networks that are trained using the back propagation algorithm, since the function is differentiable. The log sigmoid has the similar property to that of the step function, with the addition of a region of uncertainty. A log-sigmoid function in this respect is very similar to that of the input-output relationships of biological neurons, although not exactly the same. Figure 6 is the graph of a log-sigmoid function.

Sigmoid functions are also preferred because it is easy to calculate the derivatives, which is helpful for calculating the weight updates in certain training algorithms.

4.1. Training and Testing. We have used four different sets of data for training and testing of the cosimulation neural network designed. The memory module of the design and the testing of the network are being shifted onto the hardware level to speed up the performance. The neural network contains 16 input neurons; each database image is processed and stored as a feature vector of 16 values. The image is being compressed from 4096 pixel values into 16 feature vector values. The feature extraction algorithm is shown in the flowchart in Figure 7. Once the input preprocessed image is obtained, the edge image is calculated. The edge image is split into four quadrants and the maximum location of each quadrant is calculated. The distance value to be stored as the feature

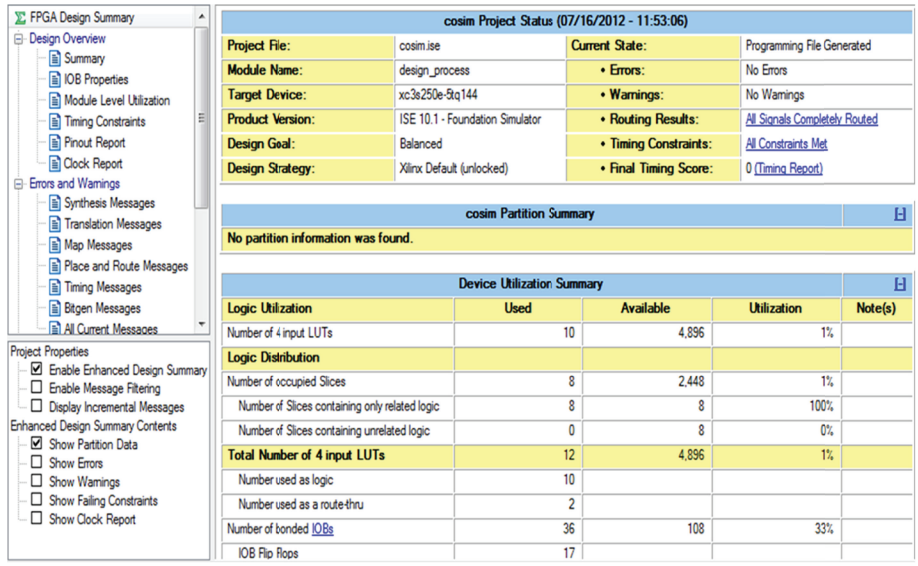

Figure 8: Device utilization Summary Report for memory storage.

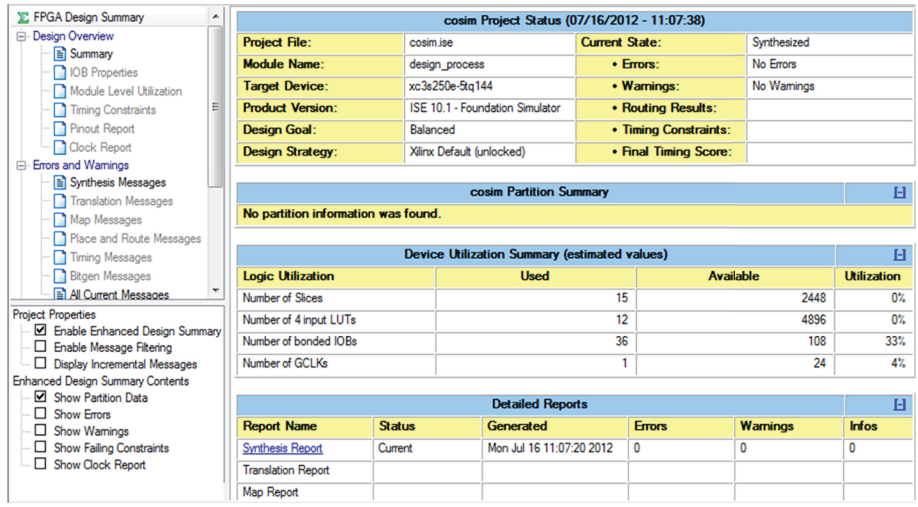

Figure 9: Device utilization Summary Report for gesture recognition system test design module.

value is the distance between the center of the image and each quadrant maximum image.

The compression ratio is 256 times that of the input values. Hence, only 0.39% of the image is being used as the feature vector to train and test the neural network designed. As the database set gestures involved in the application may vary very rapidly, it is highly essential to keep the feature vector as low as possible with no tradeoff with respect to accuracy and performance. In this particular application, the feature vector is maintained as 16 bit vector which makes the system memory efficient and also highly redundant.

4.2. Gesture Recognition. Gesture Recognition is widely used in applications like sign language, human-computer interfaces, gaming and animation technology, and so forth. Once a test image is selected, the neural network weights matrix is used to display the recognized gesture on a 7×5 display grid.

5. Device Utilization Factor

To understand the resource constraints, Xilinx ISE simulations are performed. The synthesis report provides the information about the device utilization in detail. Figures 8 and 9 show the report with the representation as defined below [36].

Device Utilization indicates the FPGA elements, such as flip-flops, LUTs, block RAM, and DSP48s.
Estimated indicates the number of FPGA elements that the algorithm might use based on the current directive configurations.

Total indicates the total number of FPGA elements available on the FPGA target.

Percent indicates the percentage of the FPGA elements that the algorithm might use.

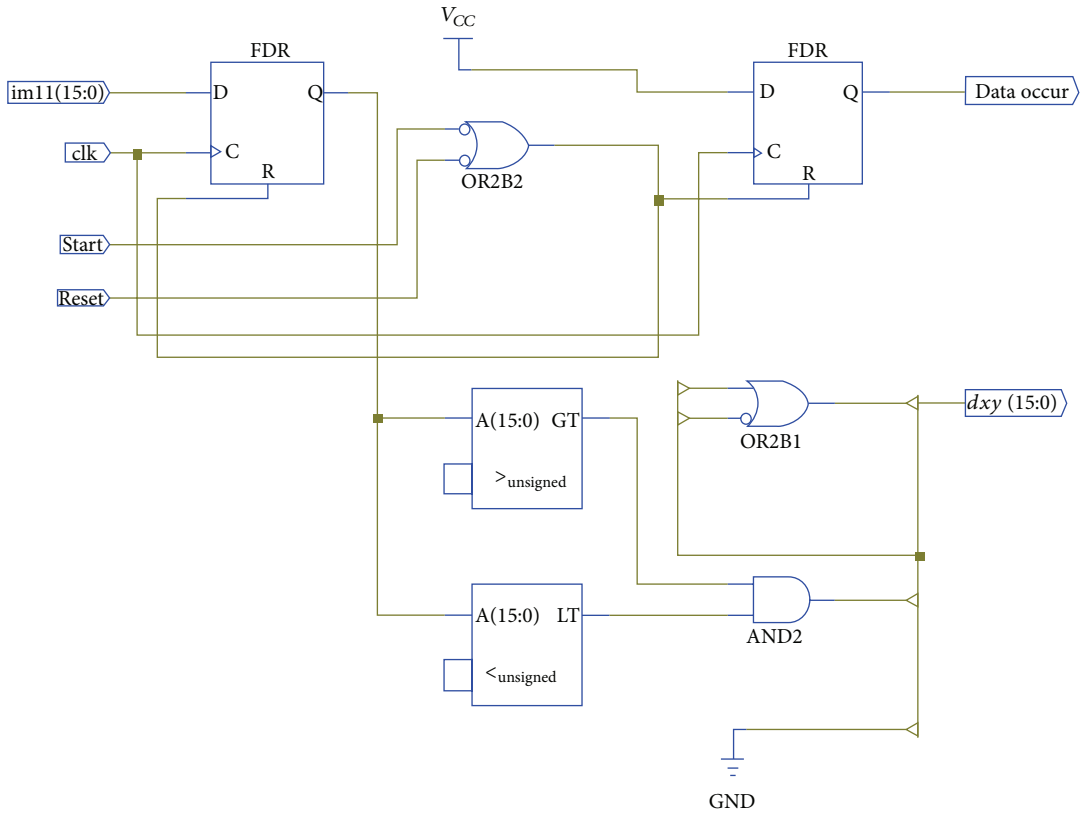

Figure 10: RTL schematic of design process flow (screenshot).

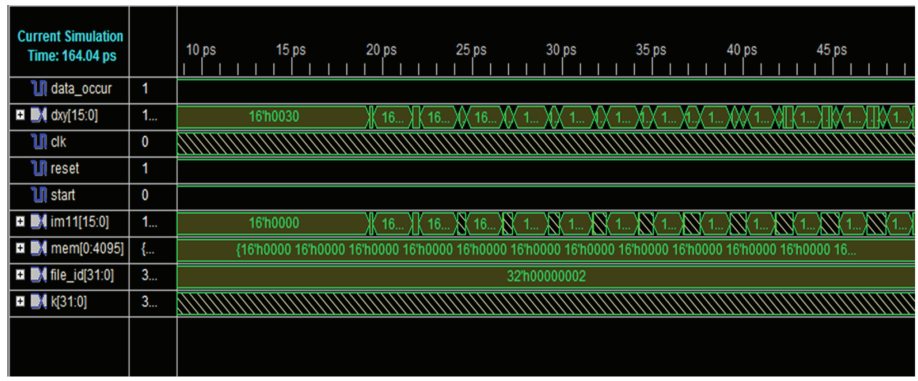

Figure 11: Simulation result in Xilinx ISE indicating the memory stored.

From the summary reports developed, it is observed that only 33% of the IOBs are being used on the hardware platform. The report is being developed on a Xilinx FPGA SPARTAN 3E with target device XC3S250e-5tq144.

Each image is being read and stored onto the memory of the ALTERA-ModelSim. The RTL schematic design is being shown in Figure 10.

6. Simulation Output

The simulation output of the test design is being shown in Figure 11 run on the Xilinx ISE. The memory *mem[0 : 4095]* loads 4096 values of each image and then processes it to extract features of set of 16 values and stores in *dxy[15 : 0]* as shown. Figures 12, 13, 14, and 15 show the simulation output

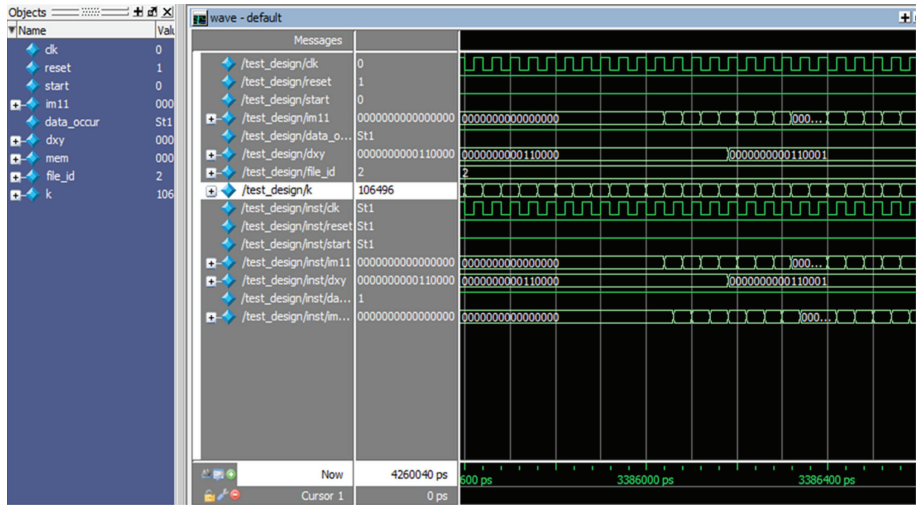

Figure 12: Simulation output for database set 1.

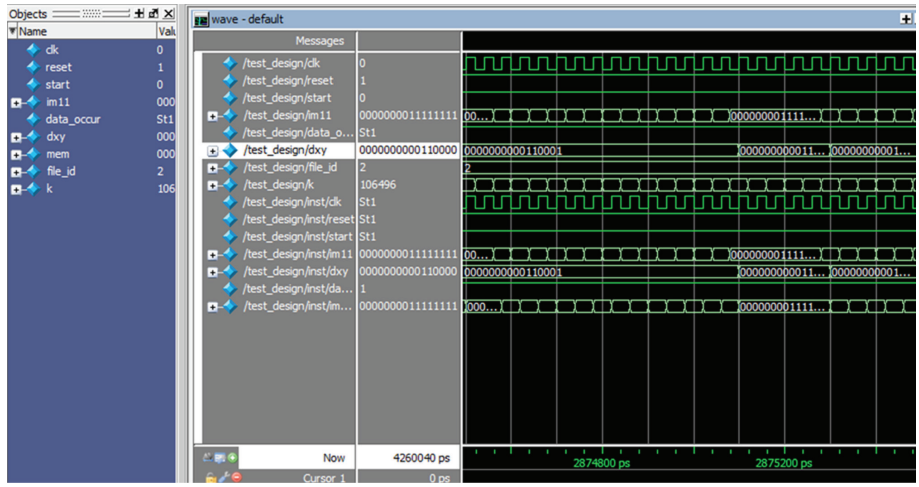

Figure 13: Simulation output for database set 2.

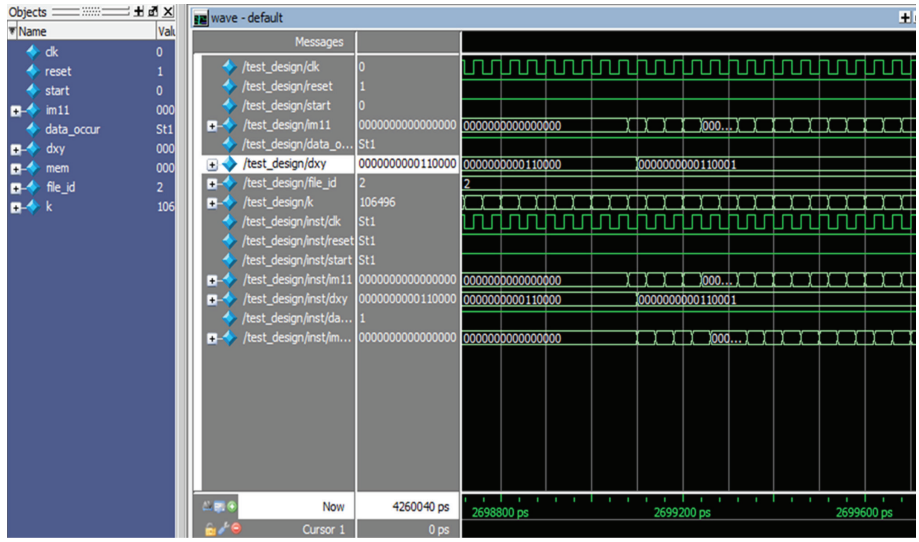

Figure 14: Simulation output for database set 3.

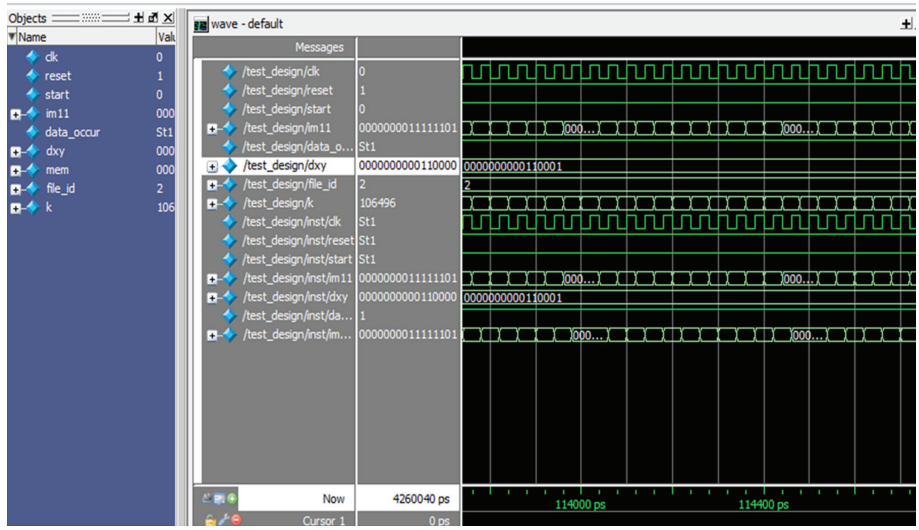

Figure 15: Simulation output for database set 4.

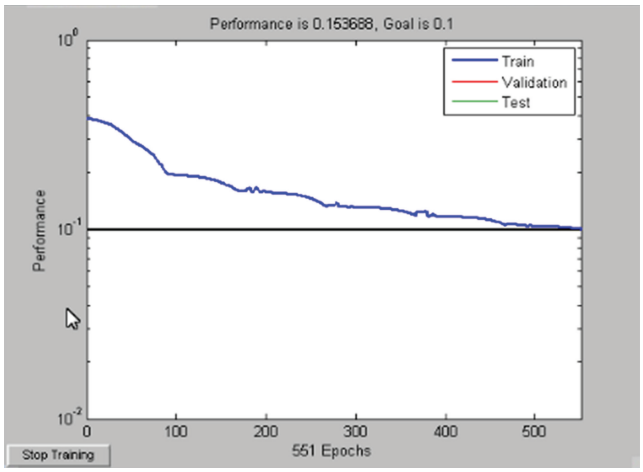

Figure 16: Performance curve for gesture recognition using SW (MATLAB software) analysis with reduced database sets for training (screenshot).

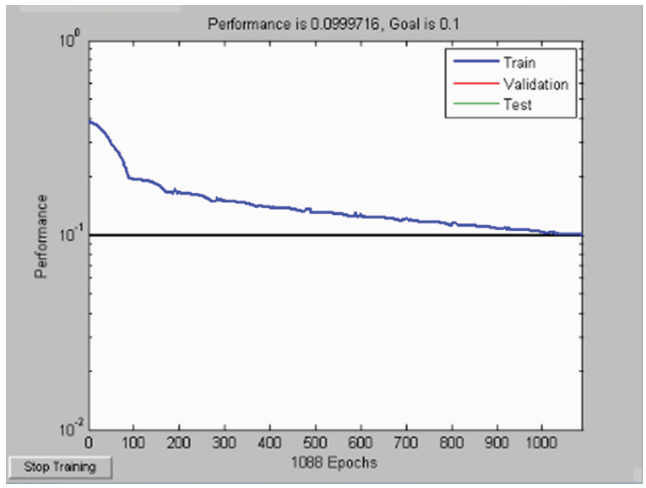

Figure 17: Performance curve for gesture recognition using HW/SW cosimulation analysis (screenshot).

for different database sets used for the gesture recognition application. The simulation time took was 164.04 ps to load the memory into the ModelSim and test the system design. The clock is represented by *clk*, memory represented by *mem*, feature vector represented by *dxy*, and the database image file is represented by *im11*. The test_design/k stores 106496 (=4096 × 26) values, which shows 4096 values of each of the 26 alphabet (A to Z) images considered.

7. Results

The cosimulation platform is designed to improve the speed of the application. Current academic and industrial researchers have recently been focusing on analyzing images of people. While researchers are making progress, the problem is hard and many present-day algorithms are complex, slow, or unreliable. The algorithms that run near real-time require computers that are very expensive relative to the existing hand-held interface devices. The proposed method runs quickly, accurately and fits for the low-power dissipation application. Figure 16 shows the performance curve for gesture recognition using MATLAB software analysis with reduced database sets for training. Figure 17 shows the performance curve for gesture recognition using HW/SW cosimulation analysis (our approach). Figure 18 shows the performance curve for gesture recognition using MATLAB software simulation analysis for complete database sets. To validate the performance of the HW/SW cosimulation network, the mean square error is generated as shown in the performance curves. Epoch-based updating of the weights is performed and mean square error is decreasing at an exponential rate and settling down to an almost constant value as shown in Figure 16. An epoch is the presentation of the entire training set to the neural network once, and for the network to reach the

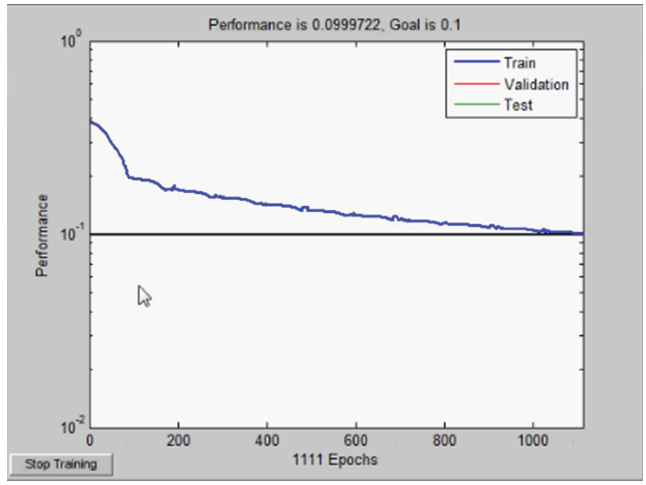

Figure 18: Performance curve for gesture recognition using SW (MATLAB) simulation analysis (screenshot).

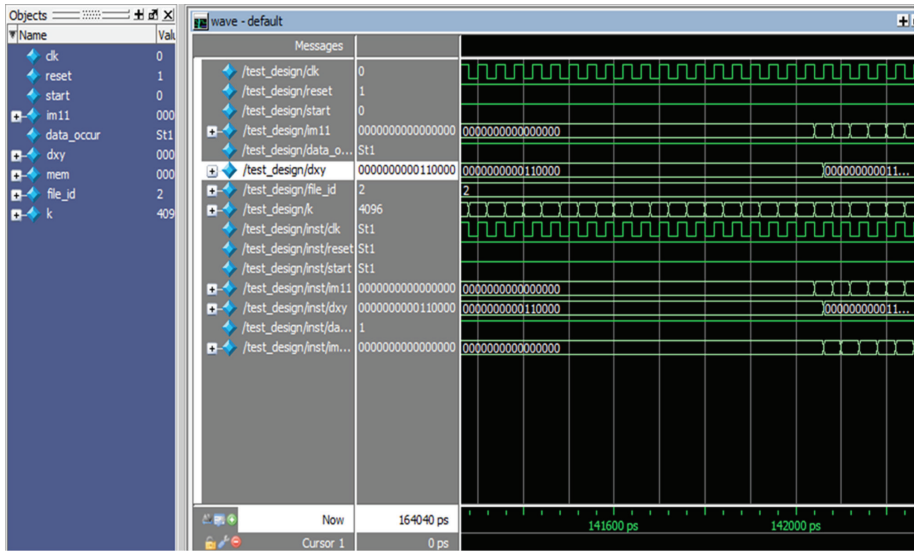

Figure 19: System design test output simulation for image gesture alphabet "P."

minimum threshold error the training is done multiple times counted as number of epochs. The maximal weight change in each epoch is decreasing and finally reaches to the least value possible. The epochs involved to reach the steady state are approximately equal (1088 and 1111, resp.) in both cases of Figures 17 and 18, but the decrease in mean square error is at higher exponential rate when compared to Figure 16 making the HW/SW better choice in terms of speed. The higher the exponential rate at which the MSE decreases, the faster the settling time and the faster training is completed.

The algorithm is able to detect the gesture input test image with 100% success rate for all the sign language alphabets (A to Z). The images with black and white background considered provided the same results as the algorithm is designed for uniform background. For users with different skin colors, this algorithm provides the same results as the features used to classify are skin color independent. Figures 19 and 20 depict the output result for the test letter "P." The influences on major factors like performance, epochs, MSE (mean square error), gradient, and time are being compared to both approaches with various database memories and are tabulated in Table 1.

It is observed from the results tabulated that there is a 10 times decrease in the MSE of the HW/SW cosimulation platform compared to completely SW-based platform. The decrease in MSE is directly related to the increase in the accuracy of the approach adopted. The aim of the back propagation algorithm is to minimize the total error which is the sum of errors (squared difference between the desired and actual outputs) generated by all patterns in the training set. The total error is computed once all patterns in the training set have been presented. One presentation of a training set is known as an epoch. It is normal for many hundreds or thousands of epochs to pass before an acceptable total error is reached. Thus, HW/SW approach reaching a MSE that is 10 times less compared to the SW approach indicates a higher

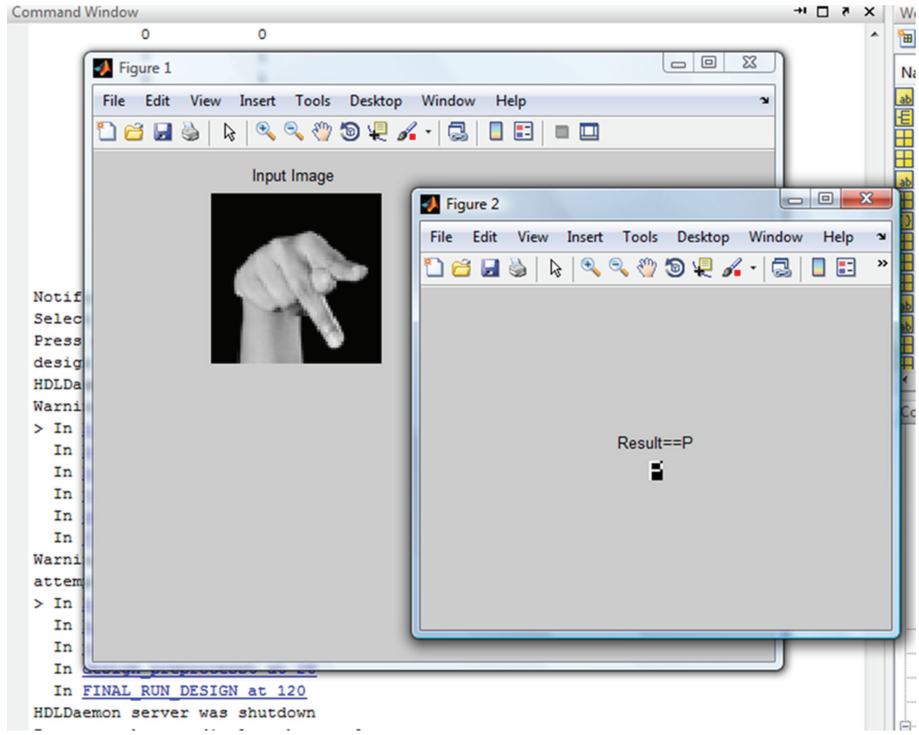

Figure 20: Output sign recognition displayed as a grid matrix.

Table 1: Comparison of SW versus HW/SW simulation platforms.

	Software simulation (MATLAB)	HW/SW cosimulation (MATLAB and ALTERA-ModelSim Quartus II)
Performance	0.999722	0.999716
Epochs	1111/2000	1088/2000
MSE	0.998182/0.1	0.0999716/0.1
Gradient	$0.00269182/1e-006$	$0.00204379/1e-006$
Recognition time	0.0058 secs	$5.7656e-004$ secs

accurate system with faster training time. Hence, the gain in speed of the system is attained.

Given an input frame for testing the time taken by the network architecture to process and recognize the sign is the single pattern recognition time. The recognition time indicated in the table represents the time taken to recognize the test image (time involved in providing the output gesture identified). From the table, it is clear that the recognition time is reduced by 10.059 times by adopting the HW/SW cosimulation platform instead of completely using the SW-based approach. This implies that the cosimulation network designed is 10 times faster than the existing approach.

Also a $640\times480\times3$ image has been compressed into a 64×64 image resulting in a compression ratio of 225 times. Then, the compressed image is again being processed to obtain a 16 feature vector which results in a compression ratio of 256 times that of the 64×64 reduced input images. This implies that only 0.39% of the reduced image is being used for the recognition system with no tradeoff with respect to accuracy and performance. The simulation time was 164.04 ps to load the memory into the hardwired memory controller due to reduced data and the hardware design of memory.

References

[1] T. Suh, H. S. Lee, S. Lu, and J. Shen, "Initial observations of hardware/software co-simulation using FPGA in architecture research," in *Proceedings of the Workshop on Architecture Research Using FPGA Platforms in Conjunction with International Symposium on High-Performance Computer Architecture*, Austin, Tex, USA, February, 2006.

[2] X. Ling, Z. Li, J. Hu, and S. Wu, "HW/SW co-simulation platforms for VLSI design," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS '08)*, pp. 578–581, 2008.

[3] M. M. Hasan and P. K. Misra, "Brightness factor matching for gesture recognition system using scaled normalization," *International Journal of Computer Science & Information Technology*, vol. 3, no. 2, pp. 35–46, 2011.

[4] M. P. Paulraj, S. Yaacob, M. S. bin Zanar Azalan, and R. Palaniappan, "A phoneme based sign language recognition

system using skin color segmentation," in *Proceedings of the 6th International Colloquium on Signal Processing and Its Applications (CSPA '10)*, pp. 1–5, May 2010.

[5] P. Mekala, Y. Gao, J. Fan, and A. Davari, "Real-time sign language recognition based on neural network architecture," in *Proceedings of the IEEE International Conference on Industrial Technology & 43rd Southeastern Symposium on System Theory (SSST '11)*, pp. 195–199, Auburn, Ala, USA, March 2011.

[6] S. Mitra and T. Acharya, "Gesture recognition: a survey," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 37, no. 3, pp. 311–324, 2007.

[7] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231–268, 2001.

[8] J. J. LaViola Jr., *A survey of hand posture and gesture recognition techniques and technology [M.S. thesis]*, NSF Science and Technology Center for Computer Graphics and Scientific Visualization, Providence, RI, USA, 1999.

[9] S. Meena, *A study on hand gesture recognition technique [M.S. thesis]*, Department of Electronics and Communication Engineering, National Institute of Technology, Rourkela, India, 2011.

[10] M. M. Hasan and P. K. Mishra, "HSV brightness factor matching for gesture recognition system," *International Journal of Image Processing*, vol. 4, no. 5, pp. 456–467, 2010.

[11] P. Garg, N. Aggarwal, and S. Sofat, "Vision based hand gesture recognition," *World Academy of Science, Engineering and Technology*, vol. 49, pp. 972–977, 2009.

[12] K. Murakami and H. Taguchi, "Gesture recognition using recurrent neural networks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching through Technology*, pp. 237–242, 1999.

[13] T. H. H. Maung, "Real-time hand tracking and gesture recognition system using neural networks," *World Academy of Science, Engineering and Technology*, vol. 50, pp. 466–470, 2009.

[14] M. Maraqa and R. Abu-Zaiter, "Recognition of Arabic Sign Language (ArSL) using recurrent neural networks," in *Proceedings of the 1st IEEE International Conference on the Applications of Digital Information and Web Technologies (ICADIWT '08)*, pp. 478–481, August 2008.

[15] G. Bailador, D. Roggen, and G. Troster, "Real time gesture recognition using continuous time recurrent neural networks," in *Proceedings of the 2nd ICST International Conference on Body Area Networks*, 2007.

[16] E. Stergiopoulou and N. Papamarkos, "Hand gesture recognition using a neural network shape fitting technique," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 8, pp. 1141–1158, 2009.

[17] X. Li, *Gesture Recognition Based on Fuzzy C-Means Clustering Algorithm*, Department of Computer Science, The University of Tennessee Knoxville, 2003.

[18] C. Lien and C. Huang, "The model-based dynamic hand posture identification using genetic algorithm," *Springer Machine Vision and Applications*, vol. 11, no. 3, pp. 107–121, 1999.

[19] R. Yang and S. Sarkar, "Gesture recognition using hidden Markov models from fragmented observations," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 766–773, June 2006.

[20] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis, "A hidden Markov model-based isolated and meaningful hand gesture recognition," *International Journal of Electrical and Electronics Engineering*, vol. 3, no. 3, pp. 156–163, 2009.

[21] R. Verma and A. Dev, "Vision based hand gesture recognition using finite state machines and fuzzy logic," in *Proceedings of the IEEE International Conference on Ultra Modern Telecommunications and Workshops (ICUMT '09)*, pp. 1–6, Petersburg, Russia, October 2009.

[22] B. Krose and P. van der Smagtan, *An Introduction to Neural Networks*, The University of Amsterdam, 8th edition, 1996.

[23] A. Chaudhary, J. L. Raheja, K. Das, and S. Raheja, "Intelligent approaches to interact with machines using hand gesture recognition in natural way: a survey," *International Journal of Computer Science & Engineering Survey*, vol. 2, no. 1, 2011.

[24] J. Wu, *Neural Networks and Simulation Methods*, Marcel Dekker, Inc., New York, NY, USA, 1994.

[25] R. Z. Khan and N. A. Ibraheem, "Hand gesture recognition: a literature review," *International Journal of Artificial Intelligence & Applications*, vol. 3, no. 4, 2012.

[26] R. Z. Khan and N. A. Ibraheem, "Survey on gesture recognition for hand image postures," *International Journal of Computer and Information Science*, vol. 5, no. 3, pp. 110–121, 2012.

[27] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *Proceedings of the International Workshop on Automatic Face and Gesture-Recognition*, pp. 296–301, IEEE Computer Society, Zurich, Switzerland, June 1995.

[28] http://www.ni.com/white-paper/6984/en.

[29] https://www.mathworks.com/accesslogin/login.do?uri=http://www.mathworks.com/help/toolbox/edalink/ref/hdldaemon.html.

[30] G. A. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *Computer*, vol. 21, no. 3, pp. 77–788, 1988.

[31] A. R. Omondi and J. C. Rajapakse, *FPGA Implementations of Neural Networks*, Springer, Dordrecht, The Netherlands, 2006.

[32] L. Fausett, *Fundamentals of Neural Networks—Architecture, Algorithms and Applications*, Prentice Hall Publishers, Upper Saddle River, NJ, USA, 1994.

[33] J. Heaton, *Introduction to Neural Networks for JAVA*, Heaton Research, Inc, 2nd edition, 2008.

[34] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[35] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What size neural network gives optimal generalization? Convergence properties of back propagation," Tech. Rep. UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, 1996.

[36] Xilinx, *XST User Guide*, Xilinx Inc., 2009.

[37] T. Y. Young and K. Fu, *Handbook of Pattern Recognition and Image Processing*, Academic Press, Orlando, Fla, USA, 1986.