FPGA Hardware Implementation of Smart Home Autonomous System Based on Deep Learning

Basman M. Hasan Alhafidh[1,3(✉)], Amar I. Daood[1,3], Mohammed M. Alawad[4], and William Allen[2]

[1] Department of Electrical and Computer Engineering, Florida Institute of Technology, Melbourne, FL, USA
bhasan2012@my.fit.edu
[2] Department of Computer Sciences and Cybersecurity, Florida Institute of Technology, Melbourne, FL, USA
[3] Department of Electrical and Computer Engineering, University of Mosul, Mosul, Iraq
[4] Biomedical Sciences, Engineering, and Computing Group, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract. The use of deep learning algorithms, as a core element of artificial intelligence, has attracted increased attention from industrial and academic institutes recently. One important use of deep learning is to predict the next user action inside an intelligent home environment that is based on Internet of Things (IoT). Recent researcher discusses the benefit of using deep learning based on different datasets to assist their result. However, assuring the best performance to satisfy real-time applications leads us to use a real-world dataset to make sure that the designed system meets the requirements of real-time applications. This paper uses the MavPad dataset which was gathered from distributed sensors and actuators in a real-world environment. The authors use simulation to investigate the performance of a multilayer neural network that predicts future human actions. The authors also present a hardware implementation of the deep learning model on an FPGA. The results showed that the hardware implementation demonstrated similar accuracy with significantly improved performance compared to the software-based implementation due to the exploitation of parallel computing and using optimization techniques to map the designed system into the target device. Additionally, our implementation of FPGA-based neural network system supports its future utilization for other applications.

Keywords: IoT · Smart environment · Middleware
Machine Learning Algorithm · Neural network
Hardware implementation · FPGA · Embedded systems

D. Georgakopoulos and L.-J. Zhang (Eds.): ICIOT 2018, LNCS 10972, pp. 121–133, 2018.
https://doi.org/10.1007/978-3-319-94370-1_9

1 Introduction

Recently, design and implementation of a smart home environment as fundamental elements of smart cities under the principles of IoT network have been obtained a lot of attention for both academic circles and industry world. L.C.D. Silva et al. [3] define smart homes as a "home-like environment that possesses ambient intelligence and automatic control". The definition indicates that the automated design has a complex and intelligent operating system which may use to do on behalf of its users [4].

In addition to the ability to manage the environment using an intelligent system approach, such system should accurately predict the needs of the human occupants, since the people in their nature try to delegate most of their needs to an automation system. The prediction process produced after in-depth studying of the sequence of interaction events between a user and the surrounded environment. The data provided from distributed sensors and actuators in the environment must be collected, filtered, managed to construct the hypotheses and finally generate the model of prediction. Generating the prediction model implies the use of an Artificial Intelligence (AI) technique or Machine Learning Algorithms (MLAs) such as Neural Network, Deep Learning, Support Vector Machine, etc.

It is important to mention that in such applications, the generated model to predict the next user action inside a smart home is entirely different from other models regarding the limited time constraint in a real-time application. In some applications, a 1 s delay could produce a severe problem [8] or even 500 ms as a maximum delay [10]. In other words, the prediction system of a smart home system should be interactive, robust, dynamic, and fast enough to predict the next user action in real-time domain. Therefore, the authors, as shown in Sect. 4, discuss the implementation not only by using Matlab simulation software but also by implementing the experiment on an embedded FPGA hardware kit to assure that our designed system will operate actively under the constraint of a real-time response in a real-world environment. The following paragraph presents different approaches that have been represented by various researchers.

Some of the recent research describes an environment as a smart due to using a smartphone or some remote controlling and monitoring system as in [2,9,12,13]. Other researchers try to enhance such a system design through the use of intelligence exhibited by machine. In other words, they use a more complex architectural model that has a business platform [5]. Other researchers use a cloud-based architecture design for IoT framework which makes the system incapable of remote access, dynamic monitoring, and real-time management with dynamic response [7,11].

This paper presents a new architecture design for a smart home system that tries to connect all the nodes (sensors & actuators) inside home environment through the use of an intelligent agent (IA). The IA locally manage all the events and status of the entire environment based on real-time application principles. Besides, the paper presents a performance analysis for the prediction model via the comparison between Software and Hardware Implementation. The

comparison is an essential process to assure a real-time response not only in S.W. implementation experiment but also in a real hardware implementation experiment when the configured system predicts the needs of human occupants in a real-time basis.

The remainder of this paper is organized as follows. Section 2 lists the proposed architectural design of the smart home environment and explain the prediction process using artificial neural network approach that is used in our experiments. Section 4 presents our experiments results using the MavPad dataset. Section 5 discusses the results and highlights the differences due to using software and hardware implementation approaches. In Sect. 6, we sum up our conclusions.

2 Our Proposed Design Approach

Our proposed system as shown in Fig. 1, tries to enhance the design of the smart home domain by introducing a new method which implies the use of a smart agent for each subsystem. Also, it uses a fog computing agent (Storage Agent) which synchronized to cloud computing environment. The cloud system adds some extra services represented by an essential backup database system for the row of data, information, rules, and hypotheses which generated by the designed system. Also, the cloud enriches the BUTLER with a variety of advanced business analysis and insight services capabilities, ending with the central Intelligent Agent (IA), which connects all the components in our architectural design in one unique design approach, that we call the "BUTLER". The idea behind this design is to present a synergistic approach of a personal assistant in the home.

2.1 Architecture Design

Beginning from the lower layers in our design, we can find that there are eight subsystems which interface the same kind of sensors/Actuators (nodes) type. Each subsystem has been connected the central IA via a dedicated agent. The agent plays an essential role in communication with subsystem and IA, filtering of input data and event, and security for communication. The most important part of the presented design is the local IA which responsible for several things as follows:

- Stakeholder interface
- Communication with IA/subsystems
- Learning stakeholder's behavior patterns
- Implementing the predicted actions
- Security and monitoring subsystem
- Storage of data from long-term and short-term activities

In this design, we would like to concentrate on presenting prediction system that depends on local computations which manipulated by the local IA. The key point here is to show that, such a system should locally monitored, controlled,

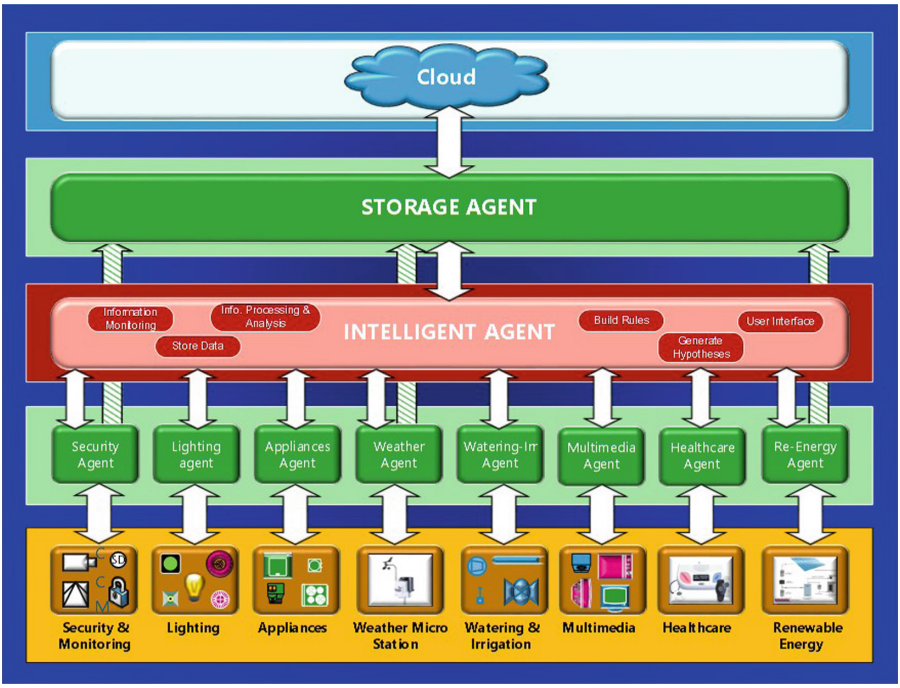

Fig. 1. The proposed architecture design of smart home system

and managed all nodes inside a home environment to predict the next stakeholder actions without the need of an external cloud-based computing system. Cloud system, as presented in recent researches, has many concerns related to its using in real-time application systems. Besides that, we would like to show where the middle-ware is located inside the architecture and how the prediction will take place locally to overcome the mentioned issue as discussed in the next paragraph.

2.2 Prediction Using Automatic Mode of Operation and Machine Learning Algorithms

Beginning with predefined variables and rules which exist in the first phase of the automatic mode of operation as shown in Fig. 2, there are learning phase and action phase. The learning phase starts with collecting the status of the nodes, analyzing and aggregating the features and contexts to establish accurate hypotheses. The generated hypotheses are very necessary to deliver the final decision. The final decision formed in IA using MLAs technique and deep learning based artificial neural network in our case. In the last phase, The predicted action should take place on the targeted actuator. Since the human nature has an inconsistent behavior in a real-life, the BUTLER should be designed to have the ability to adapt to any sudden changes that a stakeholder made himself (manual mode) in case of a stakeholder change his mind.

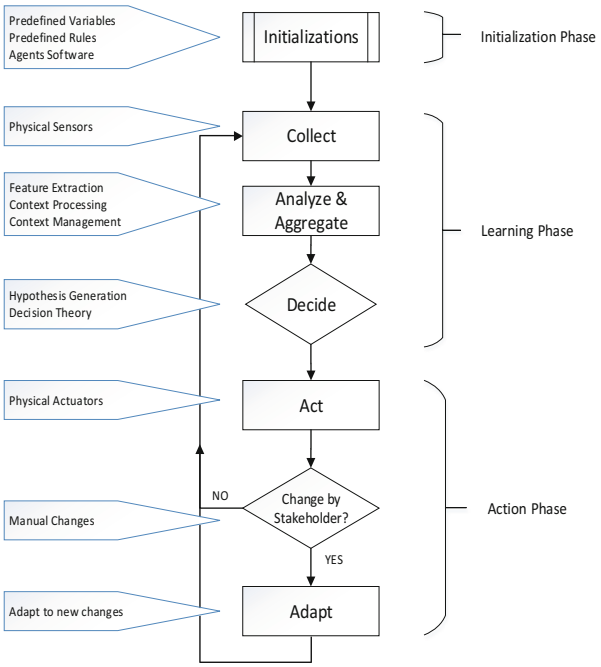

Fig. 2. The development phases for the automatic mode of operation

It is important to mention that a lot of papers present the process to predict a next user action using AI algorithms. The accuracy and time consumption of prediction process is used to choose the best algorithm among MLA techniques. Most of research papers' results describe useable systems. However, these results were collected and aggregated by implementing the MLAs on a software environment using a personal computer, server or even supercomputer. Since such a smart home system is considered to be a real-time operating system, the use of a hardware-based implementation could potentially produce the same accuracy with improved execution time. Thus, this paper attempts to verify whether a hardware implementation will provide the same, or better, results that have been produced by a software implementation. Furthermore, the authors will attempt to demonstrate that the hardware implementation will meet the needs of a real-time applications domain. These experiments use an embedded FPGA hardware development kit for the hardware implementation experiment and make use of deep learning based Neural Network (NN) algorithm, as discussed in Sect. 4.

3 Experimental Design

Before presenting our results in Sect. 4, we describe the dataset that used to implement our experiments. The MavPad dataset contains events that were collected from a real-world interaction between an individual and the home

environment in which he resided [14]. The environment consists of several zones represented by a kitchen, restroom, bedroom, and living room zones. The dataset has 127 nodes which contain 86 sensors (input predictors) and 41 actuators (observers) for seven weeks (49 days). Each day has an independent data file. The data syntax for each file represented by date, time, zone number, state, level, and source information. We used MATLAB to pre-process the row of data by extracting the informative details. After filtering the noise, we converted all 49 data files to one matrix file called (OP.mat). The MAT-file has more than 100 K rows or events for the first day alone and more than 4 million rows of information for a sum of seven weeks. Each raw has a status value for each node type which represents a predictor value or an attribute at a specific time. After data conversion process, we applied the machine learning algorithms, listed in Sect. 4, to predict the next user action in the environment.

We factorize a binary observation vector as Xt = (x1t, x2t, . . ., x86t) for the sensors. Actuators are represented by the status of each actuator at a certain time t, so it is denoted with Yt ∈ 1, ., Q for Q possible states. To investigate the performance of Neural Network algorithm that has been used in our experiments, we chose to study the behavior of a user in one zone (Restroom Zone) in this apartment. Table 1 presents the node's information inside the restroom zone.

Table 1. Sensors and actuators details for restroom zone

No.	Node type	Node label	Node name	Status value
1	Sensor	V21	Motion sensor on ceiling over toilet	0/1
2	Sensor	V22	Motion sensor on ceiling over shower	0/1
3	Sensor	V23	Motion sensor on ceiling over bathroom door	0/1
4	Sensor	S137	Light, east wall. Facing into room	DDI
5	Sensor	S138	Heat, east wall. Facing into room	DDI
6	Sensor	S139	Humidity, east wall. Facing into room	DDI
7	Sensor	S140	Reed switch over door	0/1
8	Actuator	B5	Light over mirror	0/1
9	Actuator	B6	Fan on ceiling	0/1
10	Actuator	B7	Shower light over shower	0/1

Two different combinations of sensors were used to predict the next stakeholder's action. The first case applies all the seven sensors that existed inside the restroom as shown in Table 1. The second case uses all the 86 sensors inside all the zones of the environment. The predicted action represents an actuator with binary outputs (0/1) inside the restroom labeled in (B5). The actuator B5 is used to turn the light over the mirror ON or shut it OFF.

In our experiment, we decide to take the first four weeks (28 days) for training dataset and the fifth week as the test dataset.

4 Experiments Results

4.1 Why We Use a Neural Network (NN)

The Neural Network is a commonly used tool in the machine learning field. The strength of the neural network comes from the mathematical model representation. NN is inspired by the biological nervous system of the human brain. It tries to mimic the way of the human brain processes and learns patterns. A Neural Network consists of interconnected nodes (neurons) that process the input data in a certain way to perform a specific task. Theoretically, the NN can represent many different kinds of complex function. However, a neural network has two major issues. First, the training time which requires high resources availability especially when deep learning is adopted in the network (for more than two hidden layers network). Second, the over-fitting problem due to lack of data which makes the network less generalized to unexpected patterns. Recently, data revolution, parallel architectures, and GPU designs have been developed drastically. Therefore, the neural network has become an efficient tool in the machine learning discipline. Neural network's nature offers very beneficial characteristics such as learning adaptation, self-organization, real-time output, and implementation ease.

4.2 NN Software Implementation Results

In this section, we describe the software implementation of our network. Network configuration, such as network depth (i.e., number of layers) and the number of neurons of each layer, determines computational speed. Although increasing the depth of NN improves the recognition rate (in case of having enough data), it consumes more CPU and memory resources. In this work, network depth, the number of neurons for each layer, and the training window size (i.e., the number of samples used in the training process of each step to update network's parameters) were determined experimentally by maximizing the classification performance using the available resources.

First, we started our implementation with one hidden layer. Then, we increased the number of neurons in the hidden layer to find the best representation experimentally. After that, we increased the depth of the network by adding a second layer. By fixing the number of neurons for the first layer, we increased the number of neurons for the second layer to come up with the best representation. We repeated this approach for the third layer in our network to optimize the number of nodes. Additionally, we used drop-out layers to reduce the effect of the over-fitting problem. We added a drop-out layer between every tow fully connected layers by a factor of 0.5. Removing some units of a network during training prevented excessive parameter updating. This drop-out technique may help reduce overfitting effect. We used two types of zones in our experiments, local zone, and global zone. The local zone uses seven input sensors, as shown in Table 1, to predict the B5 actuator., while the global zone uses all

sensors in the environment (86 sensors) to predict the B5 actuator. The results of these experiments are shown in Tables 2 and 3 concurrently.

A Deep Learning technique was used to facilitate the power of neural network via the implementation of the multi-layer approach. As shown in Table 2, the use of the local sensors with only one layer will produce an accuracy of %95.85 with 245 ms needed for the prediction process. Supporting the first layer with a second one can significantly enhance the accuracy performance to be 99.11% with an acceptable time of 387 ms in the scope of real-time application. While adding a third layer to the designed model, doesn't enhance the accuracy. Also, the response time of prediction process using three layers is about the double of what we possess using two layers. The mentioned results using local sensors proves that adding more layers in deep learning model doesn't always assure the best performance: Therefore, layers optimization process needs to be considered when adding further layers to the system architecture.

Table 2. The accuracy and prediction time results using local zone's sensors for multilayer neural network

No. of hidden layers	Accuracy	T (Sec)
One hidden layer	0.9585	0.245
Two hidden layers	0.9911	0.387
Three hidden layers	0.9917	0.768

Table 3. The accuracy and prediction time results using global zone sensors for multilayer neural network

No. of hidden layers	Accuracy	T (Sec)
One hidden layer	0.9017	6.1023
Two hidden layers	0.9536	9.108
Three hidden layers	0.9611	13.706

Similarly, Table 3 discusses the performance of adding a second and a third layer to the prediction system. Since this experiments use 86 sensors, which distributed in the entire environment, we can see that the response time is much higher than what we have in the first experiment that uses only seven sensors. A significant enhancement in accuracy can be noticed when adding a second layer to the deep learning model with 9.1 s of prediction time. A similar result of accuracy values has been seen when adding a third layer. In other words, support the model with a third layer doesn't facilitate better performance in the model; Therefore, the authors decided to consider designing the deep learning model to have the first two layers only which possess a maximum average accuracy and minimum average prediction time.

4.3 NN Hardware Implementation Result Using FPGA

The mathematical operations of neural network models are simple. However, the massive number of operations needs intensive computing resources. Therefore, a fast and efficient realization is required to achieve the benefits of neural models. The FPGA based system allows designers to create digital designs, test them, make modification very quickly, and reduce development time significantly [1]. Also, the Hardware implementation of nonlinear activations, e.g., the sigmoid function, is one of the challenges due to the complexity of implementing division and exponential regarding time and hardware resources. Therefore, the approximate-based approach has been presented to realize sigmoid function efficiently and maintain an acceptable level of accuracy, such as using Lookup Table LUT [15].

In this paper, we utilize an FPGA platform to realize reconfigurable hardware-based neural networks for smart home systems. The proposed architecture is shown in Fig. 3, which can be configured based on the number of hidden layers in a neural network. To achieve high performance, each neuron has one processing unit to make them work in parallel. The configuration as shown in Fig. 3 is a realization of three hidden layers neural network by assigning each set of processing units to a hidden layer. For one hidden layer configuration, all processing units are directly connected to the input buffers and the output layer. The reconfigurable switches in the figure are used for reconfiguration purpose by activating the necessary connections.

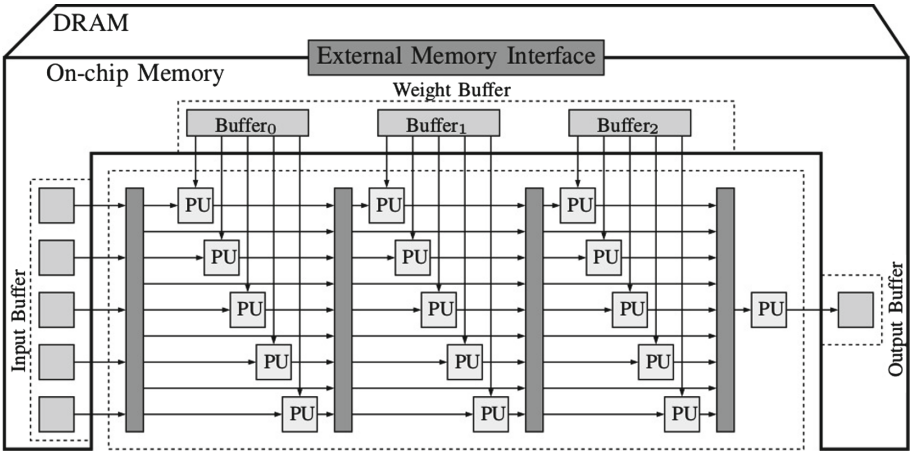

Fig. 3. Reconfigurable neural network architecture, where PUs are processing units and each set of PUs are separated by a reconfigurable switch.

Digital designs are usually modeled using hardware description languages like Very high speed integrated circuit Hardware Description Language (VHDL) and verified by simulation. In this paper, instead of using low-level coding,

such as VHDL, we used a high-level programming language, LabVIEW, to realize the neural network. National Instruments LabVIEW FPGA module uses LabVIEW embedded technology to extend LabVIEW graphical development to target FPGAs on NI reconfigurable I/O (RIO) hardware or some Xilinx boards. This module enables users to create custom hardware without low-level hardware description language coding or board design experience. Moreover, it allows a user to executes multiple tasks simultaneously and deterministically, and also expands the functionality of LabVIEW solutions, including unique timing and triggering routines, ultrahigh-speed control, interfacing to digital protocols, digital signal processing (DSP) virtual instruments (VIs).

The implementation of the trained weight data, the synaptic coefficients which are determined off-line in a computing environment, is done using signed fixed-point representation 16-bit total length. Fixed point arithmetic is used for NNs realization, which is implemented as one of the available data types in LabVIEW FPGA module. Therefore, NN coefficients are used in the hardware design without additional pre-calculation. Also, LabVIEW FPGA module provides nonlinear functions, which are used to implement the nonlinear activation functions of each neuron. The other important feature in using LabVIEW software is the ability to import weight coefficients of NNs from Matlab to the NN structure implemented in the FPGA, specifically to the block RAMs that store these coefficients.

Three optimization techniques are used in this paper to optimize and improve the performance of the FPGA-based neural network. The first one is loop pipelining to achieve high throughput by organizing the overlap in the sequence of operation of neural network systems. Single Cycle Timed Loop (SCTL) was used to reduce the required hardware resources and improve the execution speed of our proposed neural network circuit. SCTL is an optimization technique available in LabVIEW FPGA module to eliminate the unnecessary resources exploited by the standard while loop function. Due to limited computation resources in FPGA platforms and the massive computation required in realizing neural network models, loop unrolling has been exploited to efficiently utilize the resources and avoid complex hardware connections. We used this strategy to unroll the independent data and avoid complex connection topologies.

5 Discussion

The hardware implementation of the neural network has been done by LabVIEW FPGA module and downloaded to Xilinx XC3S500E FPGA. The whole neural network system fits in a low-cost Xilinx Spartan-3E FPGA platform and uses block RAMs as on-chip buffers and a DRAM as external storage. The Spartan-3E family of field-programmable gate arrays is specifically designed to meet the needs of high volume, and cost-sensitive consumer hardware digital systems, where the cost must be lower than the general purpose processors. The five-member family offers densities ranging from 100,000 to 1.6 million system gates. The XC3S500E FPGA has 4,656 slices, almost 10,476 logic cells, twenty

Table 4. FPGA-based neural network resource utilization and a comparison with other FPGA implementations

Reference	Platform	Slice LUTs	Slice registers	DSP48Es/ Multipliers	CPS	CPPSL
Gomperts et al. 2011	XC5VSX50T	8043	2243	70	536 M	9.6 M
Zhai et al. 2013	Virtex-4 LX40	4346	N/A	8	1.2 M	0.07 M
Zhai et al. 2016	XC7Z010T	4032	2863	28	72.3 M	10.3 M
Proposed	XC3S500E	3938	2862	20	481.3 M	31.2 M

18×18 hardware multipliers, as well as twenty 18 Kbits modules of dedicated dual-port RAM. In this section, we report the performance of our proposed reconfigurable FPGA-based neural network architecture and compare it with the software implementation. Then, we provide the hardware efficiency compared with the existing FPGA implementations.

Table 5. The prediction time results using local and global zone's sensors for hardware-based multilayer neural network

Zone name	No. of hidden layers	T (Sec)
Local zone's sensors	One hidden layer	0.091
	Two hidden layers	0.199
	Three hidden layers	0.485
Global zone's sensors	One hidden layer	0.329
	Two hidden layers	0.658
	Three hidden layers	1.841

Our proposed FPGA realization of neural network has **the same accuracy results** as what we have in Matlab implementation, and the accuracy is not compromised due to the usage of fixed-point computing units. The characteristic feature of using hardware platforms is performing the mathematical calculations of neural networks in parallel. This feature cannot be achieved with the software-based implementation of neural networks because of the sequential execution of the code. Table 5 illustrates the prediction time results for local and global zone's sensors with different network structures. The results show that the hardware implementation significantly improves the prediction time on average by factors of two times and thirteen times compared with simulation results in the local and global zone respectively. The significant improvement in average prediction

time commits substantial evidence that our designed autonomous system applies to be used in such a real-time application paradigm.

The hardware utilization summary is shown in Table 4 and compared with other FPGA realization approaches [6,15,16]. The proposed architecture consumes 2,828 slices out of the available 4,656 slices, which is about 60% of the total number of slices. Specifically, the reconfigurable hardware realization utilizes 3,938 slice LUTs or 42% and 2,862 slice registers or 30%. Connections per second (CPS) metric, the number of operations to be performed in a second, is used to compare hardware-based neural network architectures due to using different FPGA platforms for realization. The second metric is connection primitives per second per LUT (CPPSL), which takes the hardware resource utilization into account. CPPS can be calculated by multiplying CPS by the bit width of inputs and weights. Table 4 shows the performance comparison between our proposed architecture and other implementations using CPS and CPPSL metrics. Our proposed architecture achieves three times more CPPSL compared to the best of existing FPGA implementations.

6 Conclusion

This paper presents the design and implementation of a reconfigurable hardware-based neural network for smart home systems. The proposed architecture outperformed the performance of software-based implementation regarding speed due to exploiting parallel computing and some optimization techniques. LabVIEW software enables developers to implement digital systems without the need for low-level HDL language knowledge and reduces the usage of FPGA hardware resources. It also eliminates the need for pre-processing the neural network weights and maps them to FPGA's storage units. The implementation of FPGA-based neural network system allows its future utilization for other applications.

References

1. Ali, F.H., Mahmood, H.M., Ismael, S.M.B.: LabVIEW FPGA implementation of a PID controller for D.C. motor speed control. In: 2010 1st International Conference on Energy, Power and Control (EPC-IQ), pp. 139–144, November 2010
2. Bian, J., Fan, D., Zhang, J.: The new intelligent home control system based on the dynamic and intelligent gateway. In: 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), 2011, pp. 526–530. IEEE (2011)
3. De Silva, L.C., Morikawa, C., Petra, I.M.: State of the art of smart homes. Eng. Appl. Artif. Intell. **25**(7), 1313–1321 (2012). https://doi.org/10.1016/j.engappai.2012.05.002
4. Galvin, P.B., Gagne, G., Silberschatz, A.: Operating System Concepts. Wiley, New York (2013)

5. Gao, S., Zhu, H., Zhou, X., Liu, Y., Sun, L.: Design and implementation of smart home linkage system based on OSGI and REST architecture. In: Sun, L., Ma, H., Fang, D., Niu, J., Wang, W. (eds.) CWSN 2014. CCIS, vol. 501, pp. 568–582. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46981-1_54
6. Gomperts, A., Ukil, A., Zurfluh, F.: Development and implementation of parameterized FPGA-based general purpose neural networks for online applications. IEEE Trans. Ind. Inf. **7**(1), 78–89 (2011). https://doi.org/10.1109/TII.2010.2085006
7. Han, J., Choi, C.S., Park, W.K., Lee, I., Kim, S.H.: Smart home energy management system including renewable energy based on ZigBee and PLC. IEEE Trans. Consum. Electron. **60**(2), 198–202 (2014)
8. Hu, F.: Cyber-Physical Systems: Integrated Computing and Engineering Design. CRC Press, Boca Raton (2013)
9. Hwang, S., Yu, D.: Remote monitoring and controlling system based on ZigBee networks. Int. J. Softw. Eng. Appl. **6**(3), 35–42 (2012)
10. Koçak, D.: Thinking embedded, designing cyber-physical: Is it possible? In: Suh, S., Tanik, U., Carbone, J., Eroglu, A. (eds.) Applied Cyber-Physical Systems, pp. 241–253. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-7336-7_18
11. Mendes, T.D., Godina, R., Rodrigues, E.M., Matias, J.C., Catalão, J.P.: Smart home communication technologies and applications: wireless protocol assessment for home area network resources. Energies **8**(7), 7279–7311 (2015)
12. Piyare, R.: Internet of things: ubiquitous home control and monitoring system using android based smart phone. Int. J. Internet Things **2**(1), 5–11 (2013)
13. Riaz, M.T., Ahmed, E.M., Durrani, F., Mond, M.A.: Wireless Android Based Home Automation System
14. Youngblood, G.M., Cook, D.J.: Data mining for hierarchical model creation. IEEE Trans. Syst. Man Cybern. Part C (Applications and Reviews) **37**(4), 561–572 (2007)
15. Zhai, X., Ali, A.A.S., Amira, A., Bensaali, F.: MLP neural network based gas classification system on Zynq SoC. IEEE Access **4**, 8138–8146 (2016)
16. Zhai, X., Bensaali, F., Sotudeh, R.: Real-time optical character recognition on field programmable gate array for automatic number plate recognition system. IET Circuits Devices & Syst. **7**(6), 337–344 (2013)