Hardware Simulation of Home Automation Using Pattern Matching and Reinforcement Learning for Disabled People

M. B. I. Reaz[1], A. Assim[1], M. I. Ibrahimy[1], and F. Choong[2], F. Mohd-Yasin[2]
[1]Faculty of Engineering, International Islamic University Malaysia, Gombak, Kuala Lumpur, Malaysia
[2]Faculty of Engineering, Multimedia University, Cyberjaya, Selangor, Malaysia

Abstract - *Future Smart-Home device usage prediction is a very important module in artificial intelligence. The technique involves analyzing the user performed actions history and apply mathematical methods to predict the most feasible next user action. Unfortunately most of the techniques tend to ignore the adaptation to the user preferred actions and the relation between the actions and the state of the environment which is not practical for Smart-Home systems. This paper present a new algorithm of user action prediction based on pattern matching and techniques of reinforcement learning. The algorithm is modeled using hardware description language VHDL. Synthetic data had been used to test the algorithm and the result shows that the algorithm performs realistically better than the current available techniques*

Keywords: Smart home, multi-agent system, action prediction, reinforcement learning.

1 Introduction

Smart-Home system is an intelligent system that needs to adapt to the inhabitant's lifestyle, predict their future actions and minimize the user device interaction. Such requirements will never be achieved without proper analysis of inhabitant's device interaction history for every particular state of the home environment. The basic idea behind prediction is given a sequence of user device interaction events, how can the next user action be predicted?

In the previous years many researchers developed techniques for predicting user actions. Some of these methods were used for intelligent Human Computer Interaction (HMI) [1]; others were used for smart-home environment [2]. Although the domain were prediction techniques is different, the idea behind the techniques is the same and it is possible to apply a method of a particular domain to another by modifying the representation of the environment and action sequence.

Some of the techniques that were previously used employ Markov models to optimally predict the next user action in any stochastic sequence[3],[4], on the other hand some techniques simply used sequence matching to predict the next user action [1]. While the techniques that is currently being used passively predicts the user actions they do not take into consideration the user most preferred action for a particular environment state.

To solve this problem a new approach that uses a combination of pattern matching and reinforcement learning techniques to predict the user next action. By applying reinforcement learning the intelligent home system can receive positive reward for each action that is correctly predicted, on the other hand negative reward is given for each wrong action. By using this method the system can adapt the user ideal actions. On the other hand pattern matching will be used to match the most recent user event sequence with the history of the recorded sequences, by combining the results obtained from both methods a sufficient information is obtained to calculate the probability of the most likely to be the next user action.

Very high speed integrated circuit hardware description language (VHDL) is commonly used as a design-entry language for field-programmable gate arrays and application-specific integrated circuits in electronic design automation of digital circuits. This paper will present a VHDL modeling of a multi-agent system that is used to automate the smart-home appliances usage. The system utilizes Artificial Intelligence (AI) techniques together with the prediction algorithm illustrated in this paper of the smart-home inhabitant's appliances interaction in order to perform the task of smart-home devices automation. Details about the modeling of the system along with experimental results obtained from the testing of the algorithm and compare it to other prediction algorithms will be discussed.

2 Design methodology

The goal of the algorithm is to predict the user future actions and to base the prediction decision on the user's preferences. The first thing that has to be taken into consideration in designing any decision algorithm is how the state of the environment will be represented. When designing the algorithm the aim was to employ the algorithm for a smart-home system, therefore the environment representation is based on smart-home environment.

The environment of the home can include massive amount of information such as room temperature, time (time of the day and the date of the month), devices states, user location and many other parameters that can be sensed by the inhabitant. For the purpose of simplification, the time and the devices state as the state of the environment are going to be considered, and the action will be simply a device (On, Off) actions.

To visualize the representation of the environment states and the actions, the graph theory will be used [5]. According to graph theory, a directed graph G is an ordered pair of nodes N and vertices or arrows V as shown in equation 1.

$$G := (N, V) \qquad (1)$$

According to the representation each state will be represented as a node in the graph and the action that is performed in that particular state will yield an edge that will produce another state in the environment as shown in Fig.1.

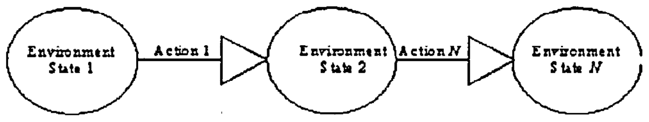


Fig. 1 Environment state and action representation using graph theory

It is worth mentioning that although the representation of the environment state is simplified it can be easily extended to include more information. But it must be taken into consideration that by including more environment variables will require more storage and processing time however prediction accuracy will be improved since the inhabitant interaction history will be significantly more comprehensive.

3 Algorithm design and pseudo code

After illustrating how the environment state and the actions will be represented, the clarification of the internal construction of the algorithm is outlined in this section. The algorithm has two main components and they are:

- The reinforcement learning component.
- The pattern matching component.

The reinforcement learning component is modeled using Q-learning algorithm. Q-learning algorithm is a recent form of reinforcement learning technique that is online and does not need a model of the environment, which makes it a great choice for the system since the prediction for smart-home has to be online also.

Q-Learning is a reinforcement learning technique that works by learning an action-value function that gives the expected utility of taking a given action in a given state. Given the environment devices states as [S], and the device actions that can be taken on a given environment state as [A], the Q value array of reinforcement learning can be formed as shown in equation 2.

$$Q = S \times A \qquad (2)$$

According to Q-Learning algorithm the Q value array is used to store the rewards the agent has received by performing a particular action at a given environment state. Each time the agent makes a correct decision, the agent is given a positive reward or a negative reward. The reward is calculated based on the user feedback to the agents performed action, which can be sensed by the system through monitoring the devices state constantly. The q value function will be calculated as shown in equation 3.

$$Q^*(x,a) = (1-\alpha)Q^*(x,a) + \alpha(r + \gamma V^*(y)) \qquad (3)$$

Where Q^* is the Q-learning value function, x is the environment states, a is the action that can be taken, α is the learning rate, γ is the value of future reinforcement and V^* is the future Q-learning value function.

The uniqueness of is that Q-learning does not specify which action to be taken, it also allow us to perform experimental trials while preserving the current optimal state. Furthermore, since this function is updated according to the ostensibly optimal choice of action at the following state, it does not matter what action is actually followed at that state. For this reason, the estimated returns in Q-learning are not contaminated by "experimental" actions so Q-learning is not experimentation-sensitive [6].

3.1 Pattern matching and algorithm structure

The pattern matching of the component of the algorithm is inspired by method used in IPAM algorithm [1]. The algorithm maintain two main directed graphs data structure, multiple graphs of device usage history sequence and a graph of the most recent device usage sequence. Consider the history sequence as an ordered set of device usage history that has occurred at a particular environment state. The history sequence is used to train the algorithm, during the training state of the algorithm the Q value array of the reinforcement learning algorithm is initialized based on the user selected action in the history. This will make the algorithm learn the optimal user actions for a given state.

The other graph of the algorithm is the recent device usage patterns that is suppose to be sensed by the smart-home system. Each graph will represent the device usage for one particular day, so the recent device usage graph will store the usage patterns for the current day that the smart-home system is running at.

The pattern matching operation starts from the most recent state recorded in the recent graph and going backward to match with each history graph. This operation will continue until the history graphs are completely searched for pattern matching. After the last iteration of this process is reached, the longest pattern match is calculated. This operation can be seen in Fig. 2.

As shown in Fig. 2, the yellow nodes in the graph represent the pattern matched in the history graphs based on the recent graph starting at node with state *S4*. The pattern match will be taken into account only if the successor state, the predecessor state and the action between them is similar to the most recent graph states actions pairs.

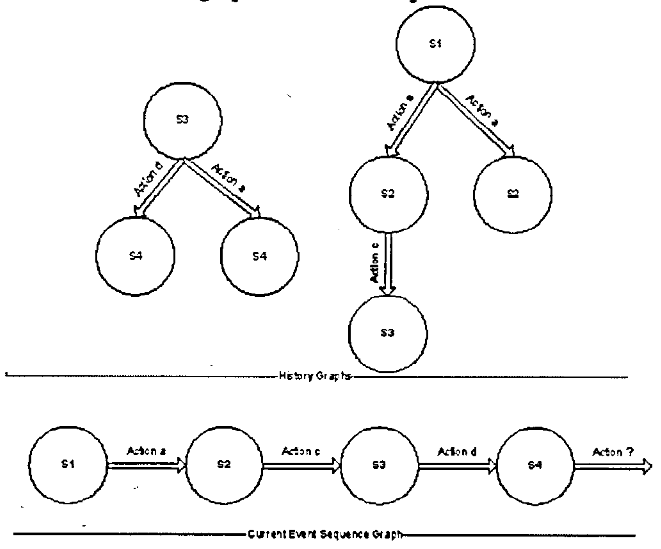


Fig. 2 Illustrating pattern matching operation

The pseudo code of the algorithm is shown in the following steps:

```
1- Let $\alpha$ is the threshold value $0 \le \alpha \le 1$, $f(s,a)$ is the action under consideration under the latest current state in the recent graph of event sequence.
2- Calculate the longest pattern match starting from the current state in the recent graph of event sequence and store it in variable $L$.
3- Calculate the Q-value function for the current action under consideration based on the previous history of events using and store it in variable $R$.
4- $h(s,a)$ is the total number of occurrences of action $a$ under the current state $s$ and $\sum_i h(s,a_i)$ is the calculated total number of occurrences of other actions $a_i$ at the same state $s$
5- Calculate the probability for the current action $a$ under consideration using the formula $f(s,a) = (1-\alpha)L + \alpha\left\{\frac{h(s,a)}{\sum_i h(s,a_i)} + R\right\}$.
6- Repeat steps 2 to 5 for the subsequent actions for the same state $s$ and select the action with the highest ranking.
```

After all the actions have been taken under consideration receive feedback form the home inhabitant. If the action was satisfying give the action a positive reward otherwise give the action a negative reward while taken the user chosen action under consideration. Calculate the Q-value for the selected action by the algorithm using Q-learning formula $Q^*(x,a) = (1-\alpha)Q^*(x,a) + \alpha(r + \gamma V^*(y))$.

4 VHDL modeling of the multi-agent system

The construction of the multi-agent system for smart-home automation is based on VHDL. VHDL is a standard text-based expression of the temporal behavior and/or (spatial) circuit structure of an electronic system. In contrast to a software programming language, an HDL's syntax and semantics include explicit notations for expressing time and concurrencies which are the primary attributes of hardware [8].

The system uses technique of multi-agent systems combined with prediction of smart-home inhabitant's interaction with home appliances to develop a system for home automation; Fig. 3 shows the schematic diagram of the system.

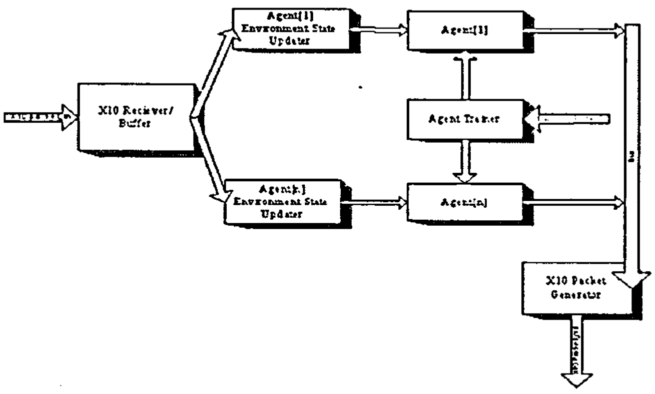
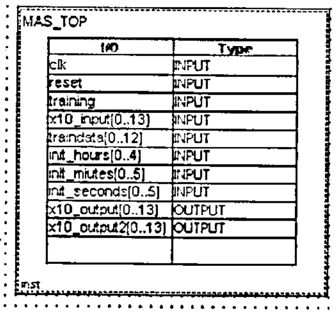


Fig. 3 Schematic diagram of multi-agent system for smart-home automation

The system is composed of three main modules; the communication module, the agent module and the trainer module. The also system consists of more than one agent module; each agent module is broken into two sub-modules which are the decision module and the prediction module.

Fig. 4 shows the top entity schematic diagram and illustrates its input/output ports.


MAS_TOP

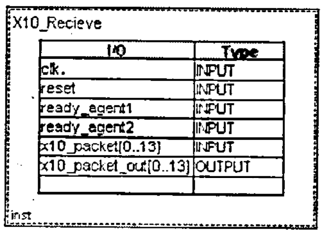
I/O	Type
clk	INPUT
reset	INPUT
training	INPUT
x10_input[0..13]	INPUT
traindata[0..12]	INPUT
init_hours[0..4]	INPUT
init_miutes[0..5]	INPUT
init_seconds[0..5]	INPUT
x10_output[0..13]	OUTPUT
x10_output2[0..13]	OUTPUT

inst

Fig. 4 The system top entity input/output ports

4.1 The communication module

The communication module is the interface of the system to the external environment. The system communicates with the home devices appliances using X10 protocol. The protocol is based on a technology known as Powerline Carrier Systems (PCS a standard that has been implemented by the CEBus and became available by the Electronic Industries Alliance ETA in 1988 [9]). The system constantly monitors the state of the devices and performs a decision, when a decision is performed an X10 packet is sent through the power lines to control the home appliances. Based on the theory of (divide an conquer) Each agent in this multi-agent system is responsible to control a specific area of the smart-home such as living room, kitchen etc, This approach guarantee a fast and accurate decision making process. The agents communicate with each other to exchange information about the devices states in each area. Fig. 5 illustrates the communication module of the system.


X10_Recieve

I/O	Type
clk.	INPUT
reset	INPUT
ready_agent1	INPUT
ready_agent2	INPUT
x10_packet[0..13]	INPUT
x10_packet_out[0..13]	OUTPUT

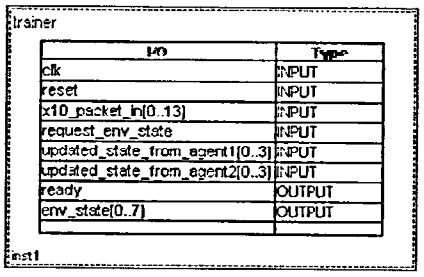
inst

Fig. 5 Illustration of communication entity module input/output ports

4.2 The training module

Before the system is online, it is need to be trained in offline mode. During the training phase a sample devices usage data of the home inhabitant is collected and converted into an appropriate X10 packets. The packets are fed to the system through the trainer to simulate a real device usage patterns, thus this will train the system on how to accurately predict and automate the devices usage of the smart-home where the system is installed.

Fig. 6 shows the schematic diagram of the training module entity and its input and output ports.


trainer

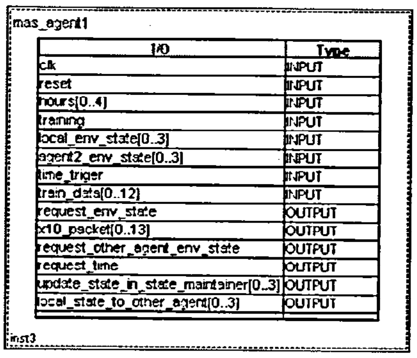
I/O	Type
clk	INPUT
reset	INPUT
x10_packet_in[0..13]	INPUT
request_env_state	INPUT
updated_state_from_agent1[0..3]	INPUT
updated_state_from_agent2[0..3]	INPUT
ready	OUTPUT
env_state[0..7]	OUTPUT

inst1

Fig. 6 The training entity module input/output ports

4.3 The agent module

The multi-agent system consist of more than one agent, each agent controls a sub-area of the smart-home like kitchen, living room, bedroom etc. The agents are homogeneous since they share the same implementation and goal. Each agent communicate with the out side environment by generating and receiving X10 packets. The agents also communicate with each other to exchange environment state on other areas of the smart-home. The agent also keeps record of the time using an internal clock. Fig. 7 shows the agent entity module and its corresponding input/output ports.

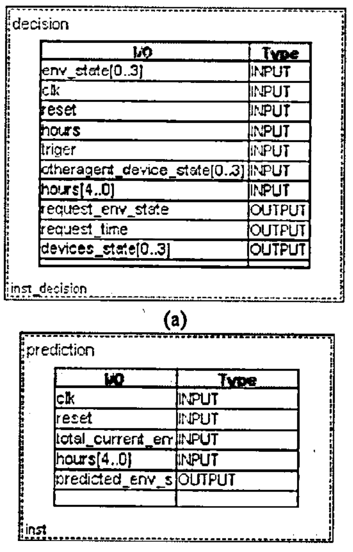

mas_agent1

I/O	Type
clk	INPUT
reset	INPUT
hours[0..4]	INPUT
training	INPUT
local_env_state[0..3]	INPUT
agent2_env_state[0..3]	INPUT
time_triger	INPUT
train_data[0..12]	INPUT
request_env_state	OUTPUT
x10_packet[0..13]	OUTPUT
request_other_agent_env_state	OUTPUT
request_time	OUTPUT
update_state_in_state_maintainer[0..3]	OUTPUT
local_state_to_other_agent[0..3]	OUTPUT

inst3

Fig. 7 The agent entity module input/output

Each agent is further broken down into two sub-modules, the decision module and the prediction module. The prediction module is architecture is modeled after the prediction algorithm illustrated in this paper. On the other hand the decision module is architected using the Q-Learning algorithm. The prediction unit predicts a set of possible next action to be performed; the decision unit will select one of those predicted actions to be performed. Thus this approach will minimize the prediction errors and make the decision process faster by limiting the number of possible actions to be performed. Fig. 8 illustrates the decision module and the prediction module input/output ports.



I/O	Type
env_state[0..3]	INPUT
clk	INPUT
reset	INPUT
hours	INPUT
triger	INPUT
otheragent_device_state[0..3]	INPUT
hours[4..0]	INPUT
request_env_state	OUTPUT
request_time	OUTPUT
devices_state[0..3]	OUTPUT

I/O	Type
clk	INPUT
reset	INPUT
total_current_en	INPUT
hours[4..0]	INPUT
predicted_env_s	OUTPUT

(b)
Fig. 8 Illustration of (a) the decision module and (b) prediction module input/output ports

5 Testing results and comparison

To test the algorithm JAVA programming language was used to implement the testing module for the algorithm. The data used for testing the algorithm has been generated synthetically to simulate device usage patterns of home inhabitant. The data then has been formulated and stored in a database file then fed to the testing module for training the algorithm and the Q-learning module. After the training phase testing patterns are performed while carefully analyzing the result and the synthetic generated patterns. In the testing we assumed that the home consist of five devices and the actions that can be taken on these devices is simply device (On, Off) actions sample data format can is shown in table 1.

TABLE 1
SAMPLE SYNTHETIC DATA USED TO TRAIN THE MULTI-AGENT SYSTEM

Date and Time	Action	Device	Location
2006-03-03 / 09:21	On	Lamp1	Living Room
2006-03-03 / 10:26	Off	Fan1	Bedroom
2006-03-03 / 10:29	On	Tv1	Living Room
2006-03-03 / 18:21	Off	Lamp1	Living Room
2006-03-03 / 20:22	Off	Tv2	Bedroom

When testing the algorithm, many trials had to be performed to set the most suitable threshold value a. After many trials we received the optimal results while setting the threshold value to 6.0. Fig. 9 shows the testing results and comparing it to other methods of [1] and [7].

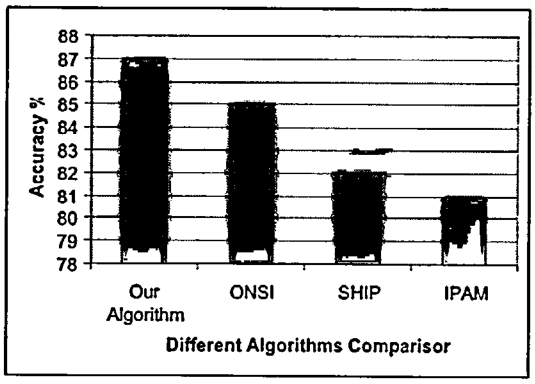


Fig. 9 Result comparison with different prediction algorithms

After verifying the simulation results from the software implementation of the prediction and the decision making algorithm. The VHDL modeling of the system was done using Altera Quartus-II software. Then a test-bench was designed to simulate the system using ModelSim software. First the training data was converted to binary format X10 packets. The data was stored in a file on the hard disk, and then during the simulation the test-bench reads the training data from the file and trains the system during the training phase. After the training phase is completed the test-bench performs synthetic simulated real-time testing of the system by issuing X10 packets as input to the system. The simulation result was recorded, analyzed and compared to the expected output pattern for results evaluation.

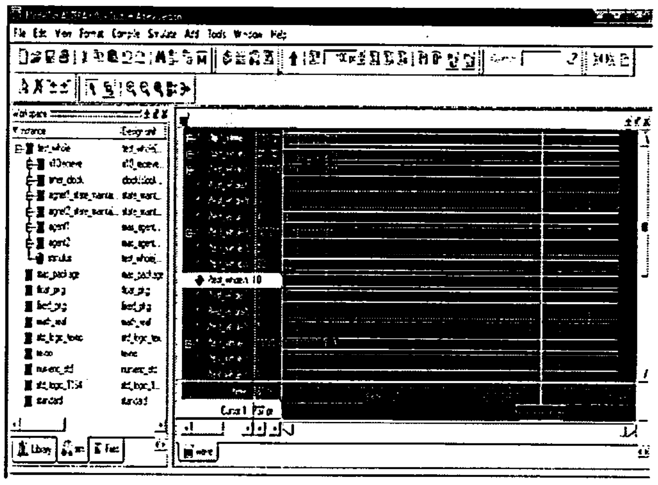
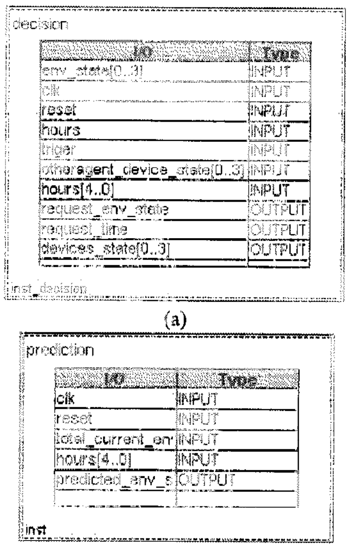


Fig. 10 Snapshot of the System Simulation Using ModelSim

Table 2 shows a summary of one of the trials that was summarized from the simulation results.



decision

I/O	Type
env_state[0..3]	INPUT
clk	INPUT
reset	INPUT
hours	INPUT
triger	INPUT
otheragent_device_state[0..3]	INPUT
hours[4..0]	INPUT
request_env_state	OUTPUT
request_time	OUTPUT
devices_state[0..3]	OUTPUT

inst_decision

(a)

prediction

I/O	Type
clk	INPUT
reset	INPUT
total_current_en	INPUT
hours[4..0]	INPUT
predicted_env_s	OUTPUT

inst

(b)

Fig. 8 Illustration of (a) the decision module and (b) prediction module input/output ports

5 Testing results and comparison

To test the algorithm JAVA programming language was used to implement the testing module for the algorithm. The data used for testing the algorithm has been generated synthetically to simulate device usage patterns of home inhabitant. The data then has been formulated and stored in a database file then fed to the testing module for training the algorithm and the Q-learning module. After the training phase testing patterns are performed while carefully analyzing the result and the synthetic generated patterns. In the testing we assumed that the home consist of five devices and the actions that can be taken on these devices is simply device (On, Off) actions sample data format can is shown in table 1.

TABLE 1
SAMPLE SYNTHETIC DATA USED TO TRAIN THE MULTI-AGENT SYSTEM

Date and Time	Action	Device	Location
2006-03-03 / 09:21	On	Lamp1	Living Room
2006-03-03 / 10:26	Off	Fan1	Bedroom
2006-03-03 / 10:29	On	Tv1	Living Room
2006-03-03 / 18:21	Off	Lamp1	Living Room
2006-03-03 / 20:22	Off	Tv2	Bedroom

When testing the algorithm, many trials had to be performed to set the most suitable threshold value α. After many trials we received the optimal results while setting the threshold value to 6.0. Fig. 9 shows the testing results and comparing it to other methods of [1] and [7].

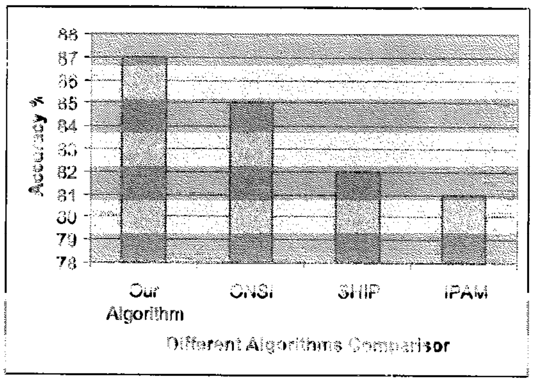


Fig. 9 Result comparison with different prediction algorithms

After verifying the simulation results from the software implementation of the prediction and the decision making algorithm. The VHDL modeling of the system was done using Altera Quartus-II software. Then a test-bench was designed to simulate the system using ModelSim software. First the training data was converted to binary format X10 packets. The data was stored in a file on the hard disk, and then during the simulation the test-bench reads the training data from the file and trains the system during the training phase. After the training phase is completed the test-bench performs synthetic simulated real-time testing of the system by issuing X10 packets as input to the system. The simulation result was recorded, analyzed and compared to the expected output pattern for results evaluation.

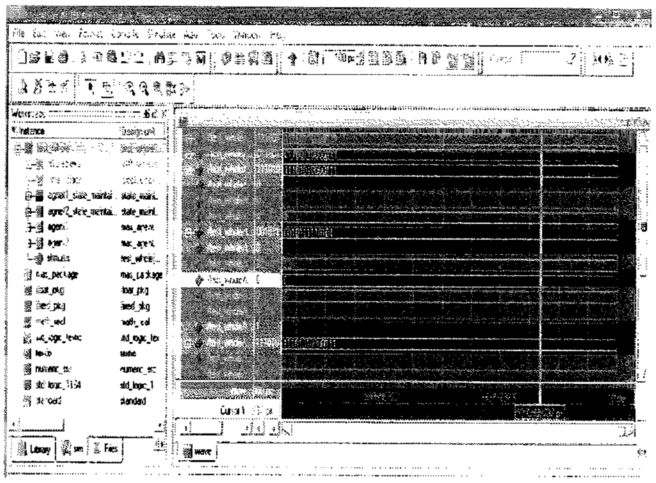


Fig. 10 Snapshot of the System Simulation Using ModelSim

Table 2 shows a summary of one of the trials that was summarized from the simulation results.

6 Discussion

In this paper, we have presented a new method of user action prediction. The algorithm was designed and intended to be implemented in smart-home system. The algorithm introduces a new technique of adapting to the user proffered actions using a combination of pattern matching and reinforcement learning technique. The result shows that the algorithm performs well under synthetic data. Future work would be to test the algorithm in real smart-home environment to get the real life optimal results and to further improve the algorithm prediction accuracy.

Also a suggested future work could be to use a hardware prototype of multi-agent system to manage the smart-home environment. Hardware prototyping using FPGA has been used effectively in many areas and showed a great result. We believe that a multi-agent hardware prototype could be very useful for home automation, as it could provide faster speed and more efficient power consumption than software based.

7 Acknowledgment

This work is supported by eScienceFund, a Ministry of Science, Technology & Innovation, Malaysia sponsored program for the advancement of R&D activities. Project No: 01-01-08-SF0045.

8 References

[1] Gorniak, P. and Poole, D., "Predicting Future User Actions by Observing Unmodified Applications", Proceedings of the 17th National Conference on Artificial Intelligence. AAAI Press, Cambridge, MA, 2000.

[2] Edwin O. Heierman, Diane J. Cook, "Improving Home Automation by Discovering Regularly Occurring Device Usage Patterns", 3rd IEEE International Conference on Data Mining, pp. 537- 540, Washington, DC, USA, 19-22 November 2003.

[3] Sira Panduranga Rao, Diane J. Cook, "Identifying Tasks and Predicting Action in Smart Homes using Unlabeled Data", International Journal on Artificial Intelligence Tools, vol. 13, part 1, pp. 81-100, 2004.

[4] Lucian Vintan, Arpad Gellert, "Person Movement Prediction Using Neural Networks", Proceedings of the KI2004 International Workshop on Modeling and Retrieval of Context, vol. 114, pp. 618-623, Ulm, Germany, September 2004.

[5] Reinhard Diestel, "Graph Theory", Springer, Chapter 1, September 15, 2005.

[6] R. S. Sutton, A.G. Barto, "Reinforcement Learning: An Introduction", MIT Press, Cambridge , Chapter 6 page no. 148, 1998.

[7] Das, S.K. Cook, D.J. Battacharya, A. Heierman, E.O., III Tze-Yun Lin "The Role of Prediction Algorithms in the Mavhome Smarthome Project", IEEE Wireless Communications, vol. 9, issue 6, pp. 77- 84, December 2002.

[8] Ulrich Heinkel, Martin Padeffke, "The VHDL Reference", John Wiley & Sons, Ltd , 2002.

[9] P. Parks, "Home Automation: A View from a Marketer," International journal of Digital and Analog Cabled Systems, March 1988.

TABLE 2
SUMMARY OF ONE SIMULATION TRIAL RESULTS

Trial 1							
Lamp 1	Lamp 2	TV	Fan	Time	Action Generated	Recommended Action	Current State
on	on	off	off	8	Device 3 =on	device 3 = on	11
on	on	on	off	8	Device 4 = on	Device 4 = on	15
on	on	on	on	21	Device 2 = on	all Devices on	36
on	on	on	on	23	Device 4 = on	all Devices on	37
on	on	on	on	7	Device4 = on	Device 1 = off	22
off	on	on	on	15	Device 4 = on	Device 3 = off	30
off	on	off	on	15	Device 4 = on	Device 4 = on	25
off	on	off	on	22	Device 3 = of	Device of = on	32
off	on	off	on	1	Device 3 = on	Device 3 = on	11
off	on	on	on	1	Device 1 = on	Device 1 = on	15
on	on	on	on	1			16