IOP Publishing Semiconductor Science and Technology
Semicond. Sci. Technol. **37** (2022) 075002 (9pp) https://doi.org/10.1088/1361-6641/ac6ae0

Analog synaptic devices applied to spiking neural networks for reinforcement learning applications

Jangsaeng Kim[1,*], Soochang Lee[1], Chul-Heung Kim[1], Byung-Gook Park[1] and Jong-Ho Lee[1,2,*]

[1] Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Republic of Korea
[2] Inter-University Semiconductor Research Center (ISRC), Seoul National University, Seoul 08826, Republic of Korea

E-mail: jhl@snu.ac.kr

Received 11 February 2022, revised 25 April 2022
Accepted for publication 27 April 2022
Published 12 May 2022

Abstract

In this work, we implement hardware-based spiking neural network (SNN) using the thin-film transistor (TFT)-type flash synaptic devices. A hardware-based SNN architecture with synapse arrays and integrate-and-fire (I&F) neuron circuits is presented for executing reinforcement learning (RL). Two problems were used to evaluate the applicability of the proposed hardware-based SNNs to off-chip RL: the *Cart Pole* balancing problem and the *Rush Hour* problem. The neural network was trained using a deep Q-learning algorithm. The proposed hardware-based SNNs using the synapse model with measured characteristics successfully solve the two problems and show high performance, implying that the networks are suitable for executing RL. Furthermore, the effect of variations in non-ideal synaptic devices and neurons on the performance was investigated.

Keywords: hardware-based spiking neural networks (SNNs), reinforcement learning (RL), deep Q-learning algorithm, neuromorphic, TFT-type flash synaptic device

(Some figures may appear in colour only in the online journal)

1. Introduction

Recently, biologically inspired neuromorphic systems have been actively studied due to their potential to overcome the constraints of conventional computing systems based on von Neumann architecture [1–5]. In particular, hardware-based spiking neural networks (SNNs) utilizing electronic synaptic devices have received a lot of attention for their low power consumption and high speed [6–10]. The hardware-based SNNs are capable of massively parallel computation while consuming low power and can imitate human brain behavior. Among the various training schemes, off-chip training schemes have been used in many hardware-based neural networks due to their ability to achieve almost similar performance to software networks [11–13]. In off-chip training schemes, the weights are trained in software and transferred to the conductance of the synaptic devices for the inference process. While there have been many studies conducted on hardware-based SNNs targeting various networks, studies targeting reinforcement learning (RL) have been rarely reported.

RL has shown the potential to solve various problems that are difficult to solve with conventional deep neural networks with superior performance than humans [14, 15]. Particularly, a deep Q-learning algorithm is often preferred in RL dealing with relatively complicated problems. The neural networks using this algorithm showed better performance beyond the human levels in various areas [14–18].

* Authors to whom any correspondence should be addressed.

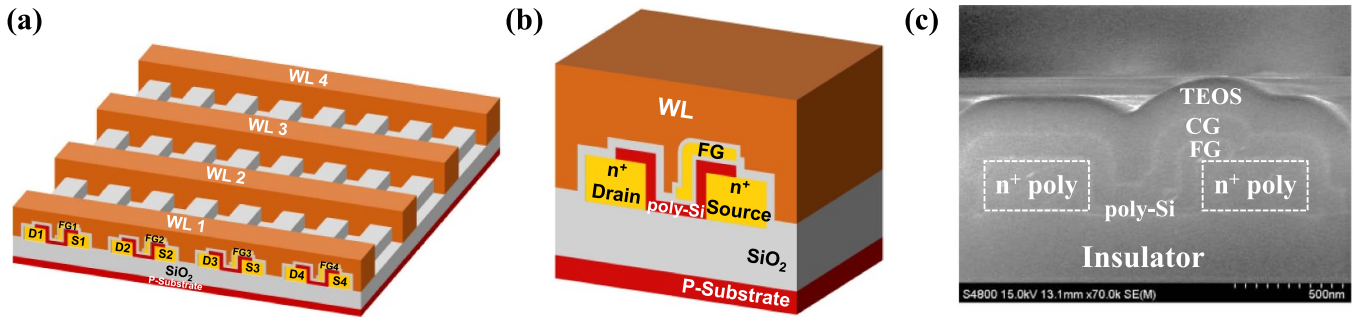

Figure 1. (a) Bird's eye view of a TFT-type flash memory array. (b) Schematic of a TFT-type flash synaptic device. (c) SEM cross-sectional image of fabricated TFT-type flash synaptic device cut in the WL direction.

In this work, we propose hardware-based SNNs applicable to off-chip RL using the fabricated thin-film transistor (TFT)-type flash synaptic devices. SNNs using analog synaptic devices have advantages in terms of area and power consumption compared to digital-based networks used in the previous studies [19, 20]. The neural network was trained using the deep Q-learning algorithm, which is relatively simple to implement in hardware compared to the other algorithms used in the RL, such as actor-critic algorithm [20]. The deep Q-learning algorithm was implemented in the same way as the method used in the previous study [16]. The performance of the network is evaluated through the two problems commonly used in the RL. The simulation results show that the proposed architecture is suitable for implementing off-chip RL by successfully solving the given problems.

This paper is organized as follows: section 2 details the TFT-type flash memory cell used as a synaptic device in this study. Section 3 presents the scenario of the two problems used to evaluate the performance of the neural network and provides the hardware-based SNN architecture. Section 4 presents the system-level simulation results of the proposed hardware-based SNN. Lastly, section 5 provides a summary of the overall paper.

2. Analog synaptic device characteristics

The synaptic devices used to express weight values in a hardware-based neural network are one of the most important elements. A TFT-type flash synaptic device is designed and fabricated in our previous work [21]. Figures 1(a) and (b) depict a schematic cross-sectional view of the proposed synaptic device. The devices are fabricated on a six-inch Si wafer with conventional CMOS process technology. The detailed process flow is described in the previous literature [21]. Below the word line (WL), an n^+ poly-Si floating gate (FG) that only covers the source line (SL) side serves as a charge storage layer. In the full erase condition, the threshold voltage does not fall below 0 V since the FG only covers half of the poly-Si channel. This reduces leakage current and significantly decreases standby power consumption. Figure 1(c) shows a scanning electron microscope (SEM) cross-sectional image of the fabricated device. The poly-Si active layer, blocking SiO_2 layer, and tunneling SiO_2 layer have thicknesses of 20 nm, 15 nm, and 7 nm, respectively. The control gate (CG) has a width of 2 μm, and the source and drain are separated by 0.5 μm. A single synaptic device can achieve a size of $8F^2$ by scaling the width of the CG to the minimum feature size (F). The long-term potentiation (LTP) and long-term depression (LTD) properties of the synaptic device are measured and illustrated in figure 2(a). Fifty repeated erase (ERS) pulses ($V_{WL} = -3$ V, $V_{SL} = 5$ V) and 300 repeated program (PGM) pulses ($V_{WL} = 0$ V, $V_{SL} = -4.8$ V) are applied to obtain LTP and LTD characteristics. The transfer characteristics with successive PGM and ERS pulses are shown in figure 2(b).

The nonlinear characteristics of synaptic devices are usually expressed through the equations from the literature [22–24] as follows:

$$G(n+1) = G(n) + \alpha_p \exp\left(-\beta_p \frac{G(n) - G_{\min}}{G_{\max} - G_{\min}}\right) \text{ for LTP} \tag{1}$$

$$G(n+1) = G(n) - \alpha_d \exp\left(-\beta_d \frac{G_{\max} - G(n)}{G_{\max} - G_{\min}}\right) \text{ for LTD,} \tag{2}$$

where $G(n)$ is the conductance of the synaptic device, n denotes the total time the pulse is applied. $G_{\max}$ and $G_{\min}$ represent the maximum and minimum conductance, respectively. α_p and α_d are the fitting parameters. β_p and β_d indicate the nonlinearity factor, one of the critical factors determining the characteristics of the synaptic devices. The LTP and LTD characteristics of the proposed TFT-type synaptic device were fitted to the models mentioned above for subsequent simulations, as shown in table 1.

3. Hardware implementation

The *Cart Pole* balancing problem and the *Rush Hour* problem commonly solved through RL were used to evaluate the performance of the hardware-based SNNs.

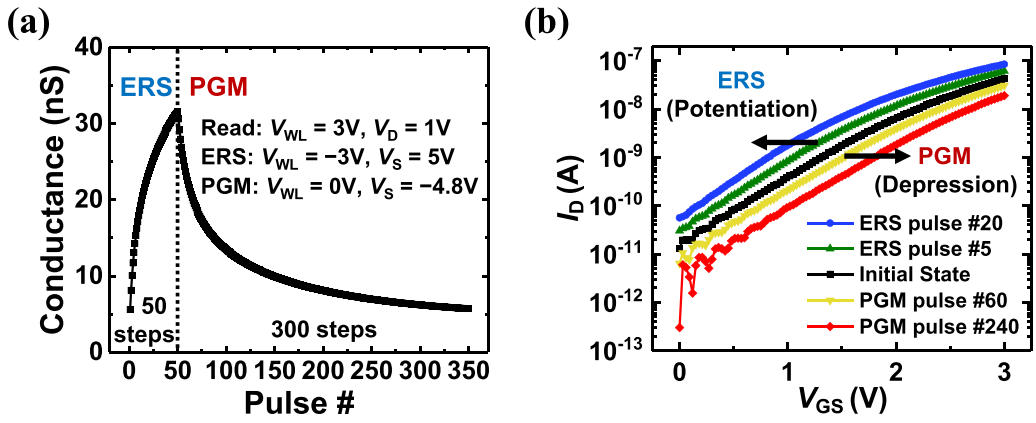

Figure 2. (a) LTP and LTD characteristics of the synaptic device with the number of applied pulses. (b) Transfer characteristics (I_D–V_{GS}) with consecutive PGM and ERS pulses.

Table 1. Fitting parameters for LTP and LTD characteristics.

Parameter	Value	Parameter	Value
α_p	2.47 (nS)	α_d	3.48 (nS)
β_p	2.59	β_d	5.26

3.1. Cart pole balancing problem

Figure 3(a) represents an example of how the *Cart Pole* balancing problem proceeds. The system consists of two elements: a cart and a pole. The pole is attached to the cart, which moves along a horizontal track. The new problem begins ($t_{\text{game}} = 1$) with the pole standing upright, and the goal is to prevent it from falling over by moving the cart position left or right. For each time step, the agent can take two actions: applying left or right directional force to the cart. A reward of 1 is provided for every time step that the pole remains upright. One episode ends when the pole falls more than a certain angle from the vertical, or the cart moves more than a certain distance from the center. Likewise, when the agent gets a total reward of 500, one episode ends and the new problem begins again.

The bottom of figure 3(a) presents an example of output spikes generated in the output neurons when $t_{\text{game}} = t_1$. In this case, because the firing frequency of the output neuron representing the action that applies left directional force is the highest, the agent takes an action that pushes the cart to the left.

In this work, the SNN used to solve the *Cart Pole* balancing problem consists of a fully connected two-layer SNN with 4 input neurons, 16 hidden neurons, and 2 output neurons. Figure 3(b) shows a schematic illustration of the proposed SNN. The four input neurons indicate the four observed data of the cart: position of the cart, velocity of the cart, angle of the pole, and angular velocity at the tip of the pole. The two output neurons represent the two actions the agent can take, as described above: applying left or right directional force to the cart.

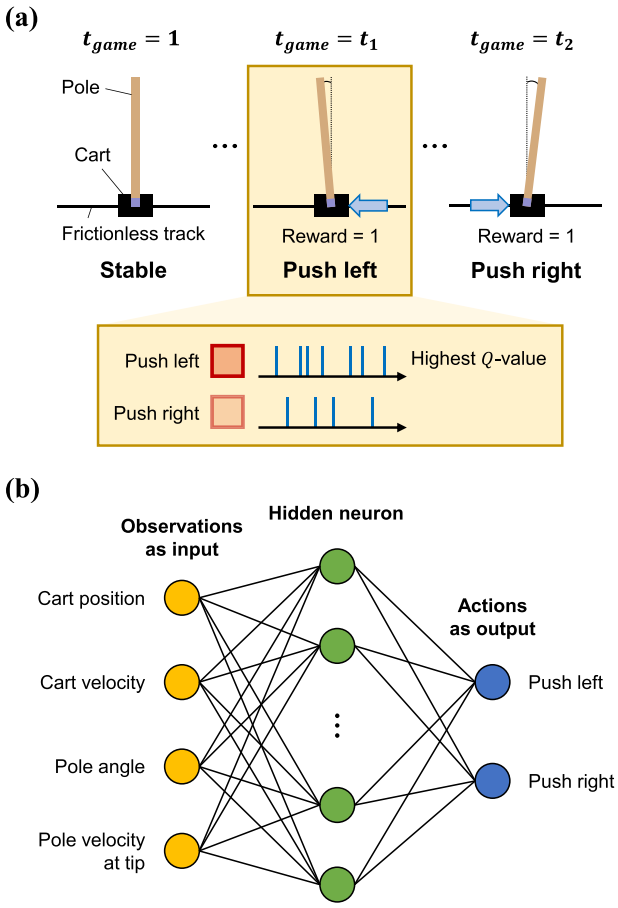

Figure 3. (a) The process of one episode of the *Cart Pole* balancing problem. The agent takes action with the highest firing frequency among two actions. (b) Schematic illustration of the SNN (4-16-2) used to solve the *Cart Pole* balancing problem.

3.2. Rush hour problem

The second problem used to evaluate the performance of the hardware-based SNNs is the *Rush Hour* problem. Figure 4(a) represents an example of the simple *Rush Hour* problem.

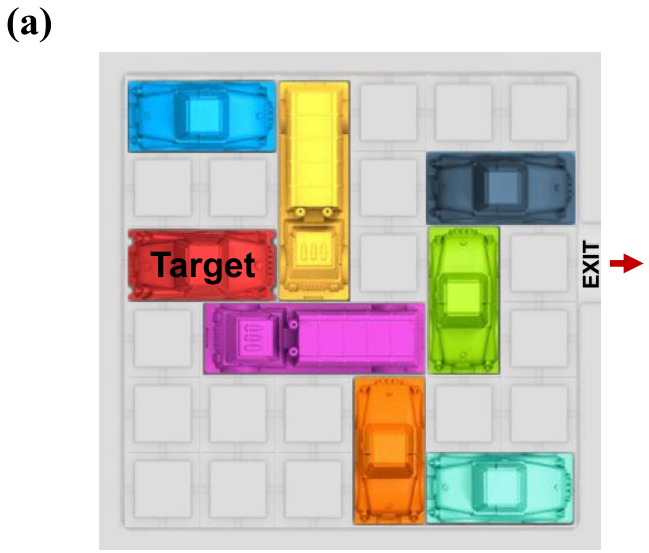

Figure 4. (a) Example of the simple *Rush Hour* problem. (b) Schematic illustration of the SNN (288-16) used to solve the *Rush Hour* problem.

Several cars with a length of 2 or 3 are arranged horizontally or vertically in a 6 × 6 position. The agent can take two actions for each car: move the car upward/left or downward/right. Each action can only move one car by one position in the direction the car is placed. Each car can only move into an empty space that is not blocked by other cars. The goal is to bring the target car (red car) to the exit with minimal movement. When there are no obstructions between the target car and the exit, the reward given to the agent is 1 and the episode concludes. In all other cases, the agent receives a reward of 0.

In this work, the SNN used to solve the *Rush Hour* problem consists of a fully connected one-layer SNN with 288 input neurons and 16 output neurons with no hidden layers. The proposed SNN is schematically illustrated in figure 4(b). Here, the 288 (6 × 6 × 8) input neurons represent the 36 (6 × 6) positions of each car. The 16 (2 × 8) output neurons represent the two directions the agent can move each car: move up/left or down/right.

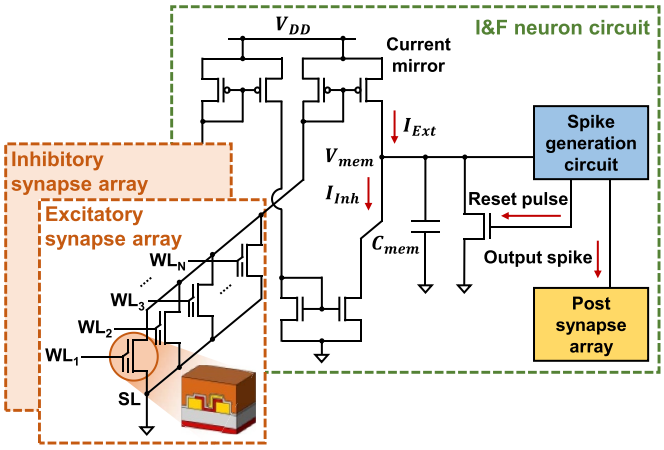

Figure 5. Proposed hardware-based SNN architecture composed of synapse arrays and I&F neuron circuits.

3.3. Hardware-based SNN

In this section, we will explain in detail how we implement hardware-based SNNs. The neural network used to solve the problems mentioned above is deep Q-network, which is commonly used in RL dealing with complex problems. After training the network, the trained weights are converted to the conductance of the synaptic devices as the method proposed in the previous study [25].

There are various methods to encode analog data in SNNs, such as temporal coding, rate coding, etc. In this work, the hardware-based SNN uses the rate coding method, one of the frequently used encoding methods. The rate coding method converts the analog input data into the frequency of the input pulses.

Figure 5 shows a schematic circuit diagram of a proposed hardware-based SNN. The SNN consists of integrate-and-fire (I&F) neuron circuits and excitatory/inhibitory synaptic device arrays ($G^+ - G^-$). The I&F neuron circuit is composed of a current mirror, a membrane capacitor, and a spike generation circuit. To express a negative weight, each weight in the hardware-based SNNs is represented by the difference in the conductance of two synaptic devices: excitatory synaptic device and inhibitory synaptic device. When the conductance of the excitatory synaptic device is larger than that of the inhibitory synaptic device, the weight value indicates a positive value. In the opposite case, the weight value indicates a negative value.

When the synaptic devices receive input pulses, the current flowing through the synaptic devices is summed through the

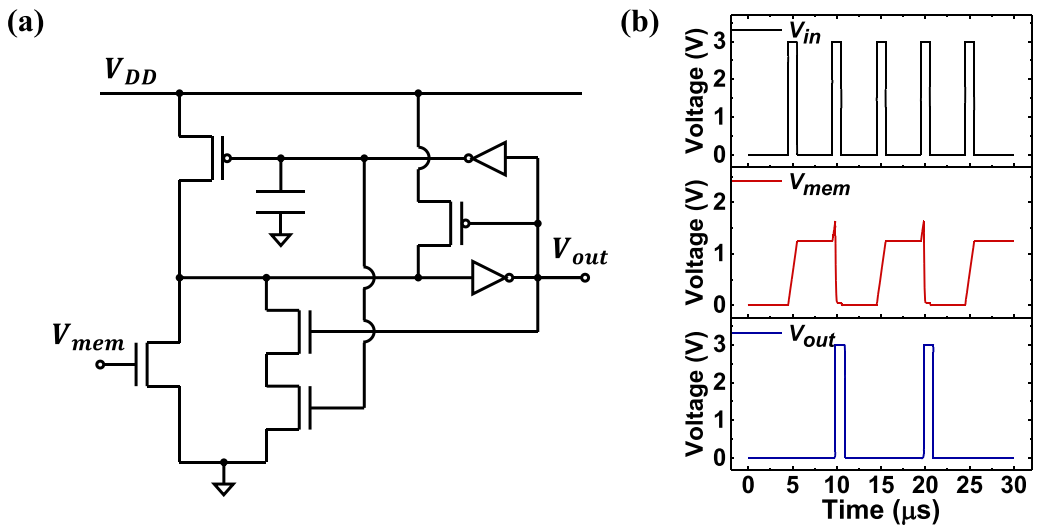

Figure 6. (a) Spike generation circuit. (b) Circuit simulation results of the I&F neuron circuit. The integration operation of the membrane capacitor and the firing operation of the spike generation circuit work properly.

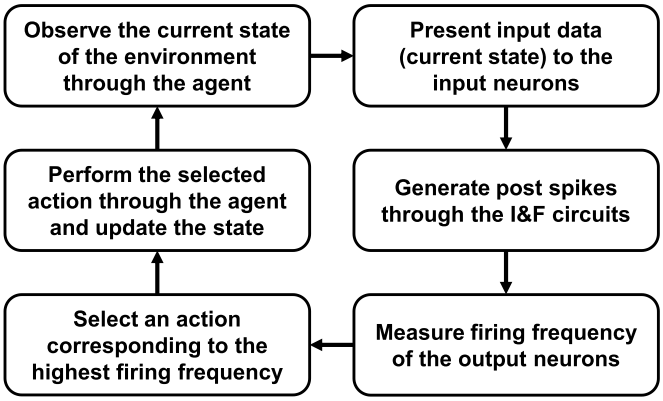

Figure 7. Flowchart of action selection. A series of processes are repeated for the changing environment while solving the given problem.

current mirror. The summed current charges the membrane capacitor to perform a synaptic input integration operation. The spike generation circuit generates a postsynaptic spike when the membrane potential is higher than the threshold voltage. The generated spikes are transmitted to the post synapse array and also function as reset pulses, discharging the charge in the membrane capacitor (resetting the membrane potential). This series of processes enables vector-by-matrix multiplication operation and is performed continuously from the input layer to the output layer. The output spikes are used to determine which action the agent should take.

The spike generation circuit used in this work is presented in figure 6(a). The operation scheme of the spike generation circuit is detailed in the previously conducted research [26]. The simulation result using a circuit simulator (HSPICE) for the I&F neuron circuit operation is shown in figure 6(b). The black, red, and blue lines represent the input spike, the voltage of the membrane capacitor, and the output spike, respectively.

Table 2. Model parameters for hardware-based SNNs.

Parameter	Value	Parameter	Value
G_{max}	31.5 (nS)	V_{th}	1.5 (V)
G_{min}	5.0 (nS)	C_{mem}	30 (fF)

When the input spike train is applied to the I&F circuit, the membrane voltage (V_{mem}) increases. When the V_{mem} exceeds the threshold voltage (V_{th}), it is reset to 0 V and generates an output spike. It is well known that the I&F neuron of the SNNs is capable of approximating rectified linear unit activation function [27–29].

Figure 7 indicates a flowchart showing the process of how the agent selects an action in a given environment. Through the process, the agent can obtain the highest expected value of the reward.

4. Simulation results and discussion

To evaluate the proposed hardware-based SNNs in terms of performance, system-level simulations were conducted for two problems: the *Cart Pole* balancing problem and the *Rush Hour* problem. The programming language Python was used for system-level simulations. Table 2 presents the model parameters used in this simulation. For all simulations, the synaptic weights are initialized following the method suggested by He [30].

4.1. Cart Pole balancing problem

Figure 8(a) represents the obtained reward of the *Cart Pole* balancing problem in the proposed hardware-based SNN with the number of training episodes. The performance of the network was evaluated every 20 training episodes. Thousand test problems were used to obtain the average value of the reward. At the beginning of the training, the agent hardly gets a reward

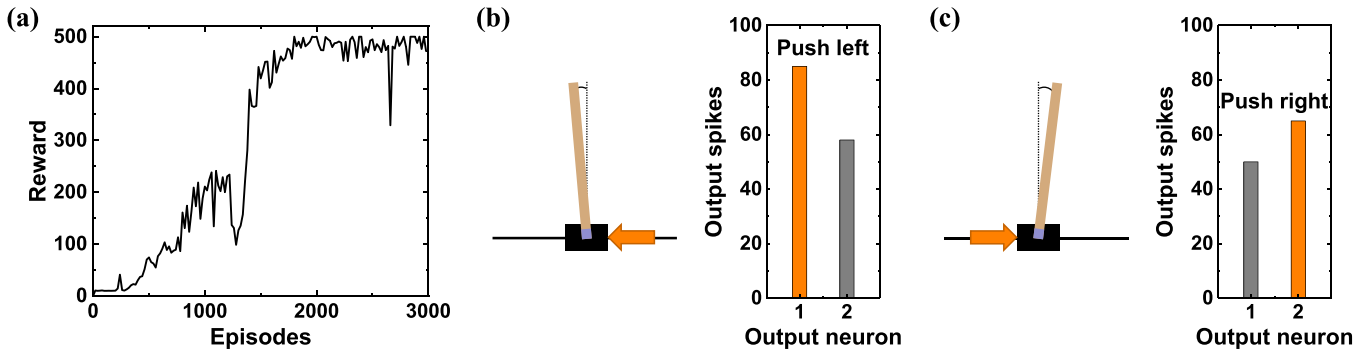

Figure 8. (a) Gained reward of the *Cart Pole* balancing problem with the number of episodes. The number of output spikes at a specific state of the *Cart Pole* balancing problem: pole tilted to the (b) left side and (c) right side. The output neuron representing the proper action fires the most in a given state.

since the agent takes a random action by randomly initialized weights. However, as the training progresses, the reward given to the agent increases and converges to the maximum value of 500.

Figures 8(b) and (c) indicate the number of output spikes obtained through the proposed hardware-based SNN at a randomly selected specific state after the training is finished. When the pole is tilted to the left side, the output neuron corresponding to the action applying the left directional force to the cart fires the most, as shown in figure 8(b). This means that pushing the cart to the left is determined to be optimal action in a given state. This action moves the cart to the left, helping the pole to stand upright. On the other hand, when the pole is tilted to the right side, the output neuron corresponding to the action applying the right directional force to the cart fires the most, as shown in figure 8(c). As the agent takes the appropriate action in a given state, it is considered that the network is trained well.

The effect of variation in the synaptic devices on the reward is also investigated, as indicated in figure 9 (black line). Here, the device-to-device variation is considered [31, 32]. The device-to-device variation is modeled as follows:

$$G_{\text{real}} = G_{\text{expected}} \times N\left(1, \sigma^2\right), \tag{3}$$

where G_{real} and G_{expected} represent the conductance value of a non-ideal real device and the expected conductance value of an ideal device, respectively. σ is the standard deviation of variation in synaptic conductivity. The σ of variations in synaptic conductivity and the threshold voltage of 50 fabricated devices are 0.219 and 0.059, respectively.

In addition, the effect of variation in the neurons on the reward is investigated, as indicated in figure 9 (red line). The variation in the firing threshold voltage of the neurons is considered as the variation in the neurons and modeled as follows:

$$V_{\text{th, real}} = V_{\text{th, expected}} \times N\left(1, \sigma^2\right), \tag{4}$$

where $V_{\text{th, real}}$ and $V_{\text{th, expected}}$ represent the threshold voltage of a non-ideal neuron and the expected threshold voltage of an ideal neuron, respectively.

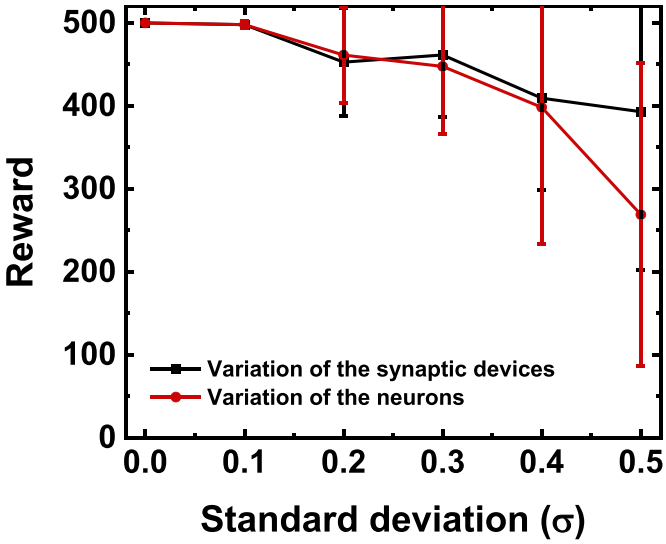

Figure 9. Gained reward of the *Cart Pole* balancing problem with variations in synaptic conductivity and neuron threshold voltage.

The simulation was conducted under the same conditions as in figure 8 except for the variations in synaptic devices and neurons. The simulation result shows the average value of the reward obtained over ten iterations versus the variations in synaptic devices and neurons. The simulation was conducted by varying the standard deviation from 0 to 0.5, which is a considerably large value.

The reward decreases as the standard deviation increases, showing vulnerability to variations in synaptic devices and neurons. However, up to the standard deviation of 0.3, the reward decreases slightly and maintains decent performance (about 90% of maximum performance when $\sigma = 0.3$). The vulnerability to variation in the synaptic devices is a common problem in hardware-based neural networks using off-chip training schemes and can be compensated by processing technologies and software approaches [23, 25, 33–35].

4.2. *Rush Hour problem*

Figure 10(a) shows the number of moves required to solve the *Rush Hour* problem with the number of training episodes.

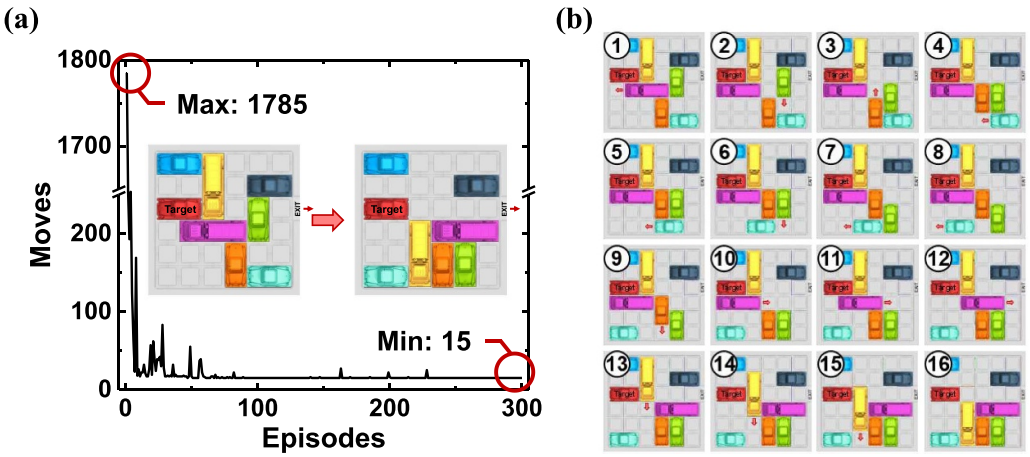

Figure 10. (a) The number of moves required to solve the *Rush Hour* problem with the number of episodes. (b) Optimal solution to the given *Rush Hour* problem.

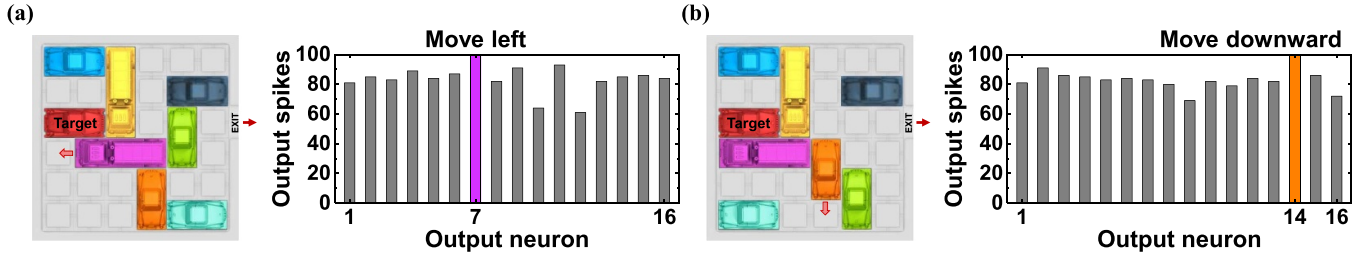

Figure 11. The number of output spikes at a specific state of the *Rush Hour* problem: (a) the initial state and (b) the state after seven steps. The output neuron representing the proper action fires the most in a given state.

The performance of the network was evaluated every training episode. Ten test problems were used to obtain the average value of the required number of moves. At the beginning of the training, a lot of moves are required to solve the problem since the agent takes a random action by randomly initialized weights. However, the number of moves required to solve the problem decreases to the optimal value of 15 as the training progresses. The optimal solution for moving the target car to the exit with the least movement in the given problem is shown in figure 10(b).

Figures 11(a) and (b) indicate the number of output spikes obtained through the proposed hardware-based SNN at a specific state after the training is finished. At the initial state of the problem, the output neuron corresponding to the action moving the purple car to the left fires the most, as shown in figure 11(a). This means that moving the purple car to the left is determined to be optimal action in a given state. Likewise, at the state after seven steps, the output neuron corresponding to the action moving the orange car downward fires the most, as shown in figure 11(b). This means that moving the orange car downward is determined to be optimal action in a given state. Through this process, the agent can move the target car to the exit with a total of 15 actions. The network is regarded to be well-trained as the agent takes the appropriate action in a given state.

The effect of variations in synaptic devices and neurons on the number of moves required to solve the *Rush Hour* problem is also investigated, as indicated in figure 12. Variations in

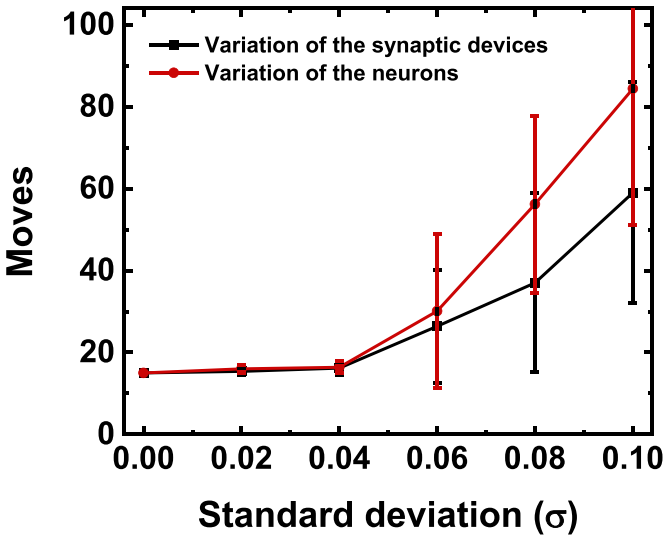

Figure 12. The number of moves required to solve the *Rush Hour* problem with variations in synaptic conductivity and neuron threshold voltage.

synaptic devices and neurons were modeled as used in the *Cart Pole* balancing problem. The simulation was conducted under the same conditions as in figure 10 except for the variations in synaptic devices and neurons. The simulation result shows the average value of the required number of moves obtained over 10 iterations versus the variations in synaptic devices and

neurons. The simulation was conducted by varying the standard deviation from 0 to 0.1.

The required number of moves increases as the standard deviation increases, showing vulnerability to variations in synaptic devices and neurons. However, up to the standard deviation of 0.06, the number of moves required is slightly increased, and decent performance is maintained (about 90% of maximum performance when $\sigma = 0.06$).

5. Conclusion

In this study, we have designed hardware-based SNNs to evaluate the performance using the system-level simulations. In the *Cart Pole* balancing problem, the proposed network shows that the reward increases to the maximum value of 500 as the training progresses. Likewise, the proposed network shows that the number of moves required to solve the *Rush Hour* problem decreases to the optimal value of 15 as the training progresses. For a specific environment, the proposed network allows the agent to take appropriate actions. Simulation results show that the proposed hardware-based SNNs are well applicable to off-chip RL with a deep Q-learning algorithm and show high performance. In addition, the effect of variations in synaptic devices and neurons was investigated. With increasing characteristic variation in non-ideal synaptic device arrays and neuron arrays, the performance of the network decreases since the off-chip training scheme was implemented. This vulnerability to variation should be carefully compensated, for example, by employing processing technologies and software approaches [23, 25, 33–35]. This can also be compensated through an on-chip training scheme, which is a challenging future study. It might require complex architecture and additional circuits. This issue will be addressed more thoroughly in the future work.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

This work was supported by the BK21 FOUR program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2021, and National R&D Program through the National Research Foundation of Korea (NRF) funded by Ministry of Science and ICT (2021M3F3A2A02037889).

ORCID iDs

Jangsaeng Kim https://orcid.org/0000-0003-4519-135X
Soochang Lee https://orcid.org/0000-0002-7554-143X
Byung-Gook Park https://orcid.org/0000-0002-2617-7627
Jong-Ho Lee https://orcid.org/0000-0003-3559-9802

References

[1] Ielmini D and Wong H-S P 2018 In-memory computing with resistive switching devices *Nat. Electron.* **1** 333–43
[2] Kuzum D, Yu S and Wong H-S 2013 Synaptic electronics: materials, devices and applications *Nanotechnology* **24** 382001
[3] Indiveri G and Liu S-C 2015 Memory and information processing in neuromorphic systems *Proc. IEEE* **103** 1379–97
[4] Ielmini D and Pedretti G 2020 Device and circuit architectures for in-memory computing *Adv. Intell. Syst.* **2** 2000040
[5] McKee S A 2004 Reflections on the memory wall *Proc. Conf. Comput. Front* pp 1–6
[6] Ielmini D and Ambrogio S 2019 Emerging neuromorphic devices *Nanotechnology* **31** 092001
[7] Masquelier T and Thorpe S J 2010 Learning to recognize objects using waves of spikes and spike timing-dependent plasticity *Int. Joint Conf. Neural Networks (IJCNN)* pp 1–8
[8] Suri M, Bichler O, Querlioz D, Cueto O, Perniola L, Sousa V, Vuillaume D, Gamrat C and DeSalvo B 2011 Phase change memory as synapse for ultra-dense neuromorphic systems: application to complex visual pattern extraction *IEDM Tech. Dig.* pp 79–82
[9] Yu S, Gao B, Fang Z, Yu H, Kang J and Wong H-S P 2012 A neuromorphic visual system using RRAM synaptic devices with Sub-pJ energy and tolerance to variability: experimental characterization and large-scale modeling *IEDM Tech. Dig.* pp 239–42
[10] Sidler S, Pantazi A, Wozniak S, Leblebici Y and Eleftheriou E 2017 Unsupervised learning using phase-change synapses and complementary patterns *Int. Conf. Artif. Neural Networks (ICANN)* pp 281–8
[11] Yu S 2018 Neuro-inspired computing with emerging nonvolatile memorys *Proc. IEEE* **106** 260–85
[12] Shafiee A, Nag A, Muralimanohar N, Balasubramonian R, Strachan J P, Hu M, Williams R S and Srikumar V 2016 ISAAC: a convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars *Proc. IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)* pp 14–26
[13] Yu S, Li Z, Chen P-Y, Wu H, Gao B, Wang D, Wu W and Qian H 2017 Binary neural network with 16 Mb RRAM macro chip for classification and online training *IEDM Tech. Dig.* pp 16.2.1–4
[14] Mnih V *et al* 2015 Human-level control through deep reinforcement learning *Nature* **518** 529–33
[15] Silver D *et al* 2017 Mastering the game of Go without human knowledge *Nature* **550** 354–9
[16] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D and Riedmiller M 2013 Playing Atari with deep reinforcement learning (arXiv:1312.5602) pp 1–9
[17] Hasselt H, Guez A and Silver D 2016 Deep reinforcement learning with double Q-learning *Proc. AAAI* vol 30 pp 1–7
[18] Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M and Freitas N 2016 Dueling network architectures for deep reinforcement learning (arXiv:1511.06581) pp 1–15
[19] Spano S, Cardarilli G C, Nunzio L D, Fazzolari R, Giardino D, Matta M, Nannarelli A and Re M 2019 An efficient hardware implementation of reinforcement learning: the Q-learning algorithm *IEEE Access* **7** 186340–51
[20] Wu N, Vincent A, Strukov D and Xie Y 2020 Memristor hardware-friendly reinforcement learning (arXiv:2001.06930) pp 1–10
[21] Kim C-H, Lee S, Woo S Y, Kang W-M, Lim S, Bae J-H, Kim J and Lee J-H 2018 Demonstration of unsupervised learning with spike-timing-dependent plasticity using a TFT-type NOR flash memory array *IEEE Trans. Electron Devices* **65** 1774–80

[22] Querlioz D, Dollfus P, Bichler O and Gamrat C 2011 Learning with memristive devices: how should we model their behavior? *IEEE/ACM Int. Symp. Nanoscale Architectures* pp 150–6
[23] Querlioz D, Bichler O, Dollfus P and Gamrat C 2013 Immunity to device variations in a spiking neural network with memristive nanodevices *IEEE Trans. Nanotechnol.* **12** 288–95
[24] Ernoult M, Grollier J and Querlioz D 2019 Using memristors for robust local learning of hardware restricted Boltzmann machines *Sci. Rep.* **9** 1–15
[25] Kwon D, Lim S, Bae J-H, Lee S-T, Kim H, Kim C-H, Park B-G and Lee J-H 2018 Adaptive weight quantization method for nonlinear synaptic devices *IEEE Trans. Electron Devices* **66** 395–401
[26] Kang W-M, Kim C-H, Lee S, Woo S Y, Bae J-H, Park B-G and Lee J-H 2019 A spiking neural network with a global self-controller for unsupervised learning based on spike-timing-dependent plasticity using flash memory synaptic devices *Int. Joint Conf. Neural Networks (IJCNN)* pp 1–7
[27] Diehl P U, Neil D, Binas J, Cook M, Liu S-C and Pfeiffer M 2015 Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing *Int. Joint Conf. Neural Networks (IJCNN)* pp 1–8
[28] Rueckauer B, Lungu I-A, Hu Y, Pfeiffer M and Liu S-C 2017 Conversion of continuous-valued deep networks to efficient event-driven networks for image classification *Front. Neurosci.* **11** 1–12
[29] Tavanaei A and Maida A 2019 BP-STDP: approximating backpropagation using spike timing dependent plasticity *Neurocomputing* **330** 39–47
[30] He K, Zhang X, Ren S and Sun J 2015 Delving deep into rectifiers: surpassing human-level performance on imagenet classification *IEEE Int. Conf. Computer Vision (ICCV)* pp 1026–34
[31] Gong N, Idé T, Kim S, Boybat I, Sebastian A, Narayanan V and Ando T 2018 Signal and noise extraction from analog memory elements for neuromorphic computing *Nat. Commun.* **9** 1–8
[32] Sun X and Yu S 2019 Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks *IEEE J. Emerg. Sel. Top. Circuits Syst.* **9** 570–9
[33] Ambrogio S *et al* 2018 Equivalent-accuracy accelerated neural-network training using analogue memory *Nature* **558** 60–67
[34] Lee J-H, Lim D-H, Jeong H, Ma H and Shi L 2019 Exploring cycle-to-cycle and device-to-device variation tolerance in MLC storage-based neural network training *IEEE Trans. Electron Devices* **66** 2172–8
[35] Kim S, Lim M, Kim Y, Kim H-D and Choi S-J 2018 Impact of synaptic device variations on pattern recognition accuracy in a hardware neural network *Sci. Rep.* **8** 1–7