REGULAR PAPER

Dual integration design of approximate random weight generator and computation-in-memory for event-based neuromorphic computing

Japanese Journal of Applied Physics **63**, 03SP83 (2024) **REGULAR PAPER**
https://doi.org/**10.35848/1347-4065/ad2415**

Dual integration design of approximate random weight generator and computation-in-memory for event-based neuromorphic computing

Naoko Misawa*, Shunsuke Koshino, Ruhui Liu, Chihiro Matsui, and Ken Takeuchi

Department of Electrical Engineering and Information Systems, The University of Tokyo, Bunkyo, Tokyo, 113-8656, Japan

*E-mail: misawa@co-design.t.u-tokyo.ac.jp

Received October 20, 2023; revised January 3, 2024; accepted January 29, 2024; published online March 11, 2024

This paper comprehensively analyses dual integration of approximate random weight generator (ARWG) and computation-in-memory for event-based neuromorphic computing. ARWG can generate approximate random weights and perform multiply-accumulate (MAC) operation for reservoir computing (RC) and random weight spiking neural network (SNN). Because of using device variation to generate random weights, ARWG does not require any random number generators (RNGs). Because RC and random weight SNN allow approximate randomness, ARWG only needs to generate approximate random weights, which does not require error-correcting code to correct weights to make the randomness accurate. Moreover, ARWG has a read port for MAC operation. In this paper, the randomness of random weights generated by the proposed ARWG is evaluated by Hamming distance and Hamming weight. As a result, this paper reveals that the randomness required for ARWG is much lower than that for physically unclonable functions and RNGs, and thus the proposed ARWG achieves high recognition accuracy.

1. Introduction

Event-based neuromorphic computing,[1–3] such as reservoir computing (RC)[4–7] and random weight spiking neural network (SNN),[8–11] is superior in processing event data because RC and random weight SNN process time-series data. Figure 1 shows event-based RC and random weight SNN using random weights. Because event data are recorded by an event-based vision sensor[12–16] that detects only brightness changes of pixels, the volume of event data is small. In RC, input weights W_{IN} and recurrent weights W_{REC} are initially set to random weights and are not trained. Only output weights W_{OUT} in the final fully connected layer are trained by linear regression that is used as a classifier. RC has the advantage of easy training due to random weights, which remove frequent weight updates. Random weight SNN is inspired by RC and employs random weights in a convolution layer. By applying random weights to the convolution layer, feature extraction is focused on temporal direction rather than spatial direction. Although random weights remove frequent weight updates and make training easy, to implement RC and random weight SNN on edge devices, generating random weights can be problematic in terms of the quality of random weights and the circuit area for random number generators (RNGs).[17–20] Therefore, approximate random weight generator (ARWG)[21] is proposed. The proposed ARWG can generate approximate random weights and perform multiply-accumulate (MAC) operation for RC and random weight SNN.[22] If RC or random weight SNN is implemented in edge devices, the conventional circuit area requires RNGs and error-correcting code (ECC) for random weights and computation-in-memory (CiM) for MAC operation. However, because the proposed ARWG uses device variation to generate random weights, the proposed ARWG does not require any RNGs. Because RC and random weight SNN allow approximate randomness, the proposed ARWG only needs to generate approximate random weight, which does not require ECC to correct weights to make the randomness accurate. Moreover, the proposed ARWG has a read port for MAC operation.

In this paper, dual integration of ARWG and CiM is comprehensively analyzed. The proposed ARWG is discussed and compared with physically unclonable functions (PUFs) and RNGs in terms of their use, function, and randomness.[23–30] The randomness of random weights generated by the proposed ARWG is analyzed with device and circuit variations. Furthermore, the requirements of random weights from RC and random weight SNN are evaluated by Hamming distance and Hamming weight, which are indicators of repeatability and uniqueness of the randomness. The evaluation of the randomness from the network side can be used as the criteria for the requirements of randomness when fabricating chips. As a result, this paper reveals that the proposed ARWG for RC and random weight SNN can allow a lot lower repeatability and uniqueness of the randomness than the requirement from PUFs and RNGs.

The remainder of this paper is organized as follows. Section 2 describes the use, function and circuit operation of ARWG. Section 3 discusses the randomness of random weights that are generated by ARWG with device and circuit variation. Section 4 evaluates the requirements of random weights from RC and random weight SNN. Finally, Sect. 5 summarizes this study.

2. Proposed ARWG

Figure 2 shows the proposed ARWG. ARWG integrates generating random weights by utilizing device variation and CiM for MAC operation. The proposed ARWG is composed of 8T-SRAM. Unlike conventional PUFs, the proposed ARWG has a read port. As a result, the proposed ARWG not only generates random weights, but also performs MAC operation. For RC, random weights are used for W_{IN} and W_{REC}, while for random weight SNN random weights are used in the convolution layer. Since random weights are not trained, the proposed ARWG only needs to generate random weights once for 8T-SRAM array (Function 1: ARWG) and uses the generated random weights for MAC operation in both training and inference (Function 2: CiM). As shown in Fig. 3, the conventional circuit configuration consists of RNG for generating random weights and CiM for MAC operation for RC and random weight SNN. In addition, to realize a perfect match, ECC is employed in a controller. However, the proposed ARWG does not require ECC since it does not need to correct weights to make the randomness accurate because

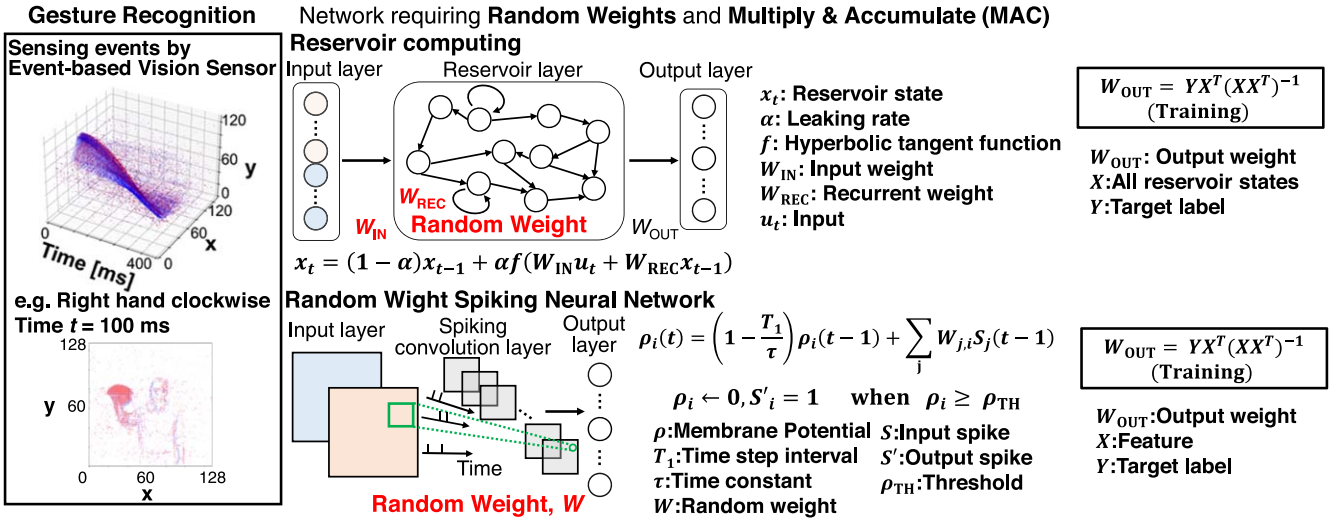

Fig. 1. Event-based RC and random weight SNN.

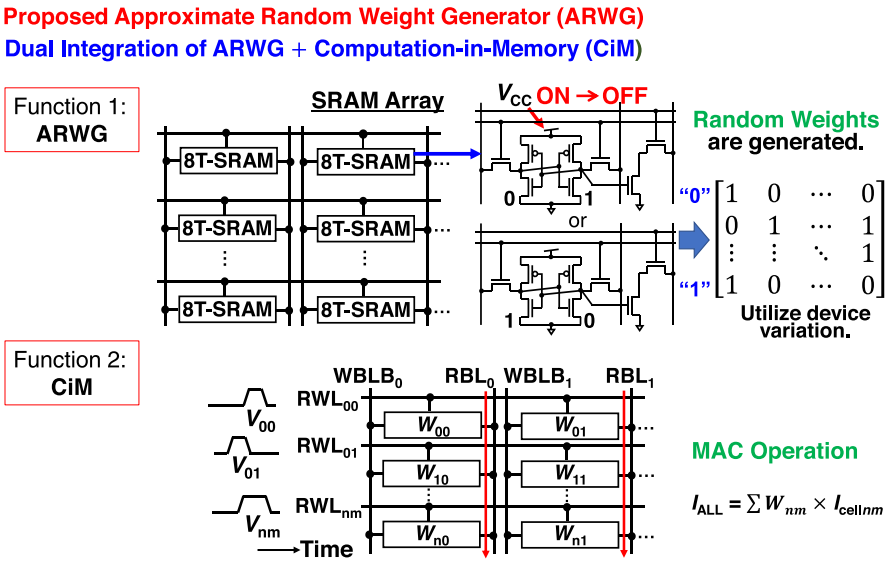

Fig. 2. Proposed ARWG.

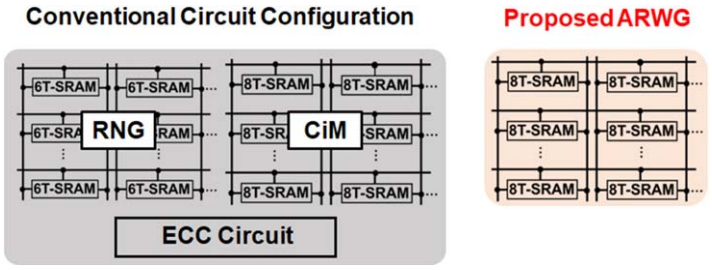

Fig. 3. Circuit area comparison between the conventional circuit configuration and the proposed ARWG for RC and random weight SNN.

RC and random weight SNN allow approximate randomness. Thus, the conventional circuit area is more than twice as large as the proposed ARWG.

Figure 4 describes the operation of the proposed ARWG with (a) input events and V_{CC} relative to time and (b) circuit operation in the 8T-SRAM array. There are four phases of the operation. Phase 1 is random weight setting. When V_{CC} starts up, the proposed ARWG generates random weights based on the uniqueness of SRAM device variation. If power is supplied to V_{CC}, an 8T-SRAM generates either "1" or "0" depending on device variation. SRAM device variation makes randomness in the weights of RC or random weight SNN. Phase 2 is MAC operation. Since V_{CC} remains supplied, the random weights generated in Phase 1 are kept stored in the 8T-SRAM array. Therefore, the proposed ARWG can use the random weights for MAC operation. If input events occur, read word-lines become high. The results of MAC operation are obtained by sensing read bit-lines.

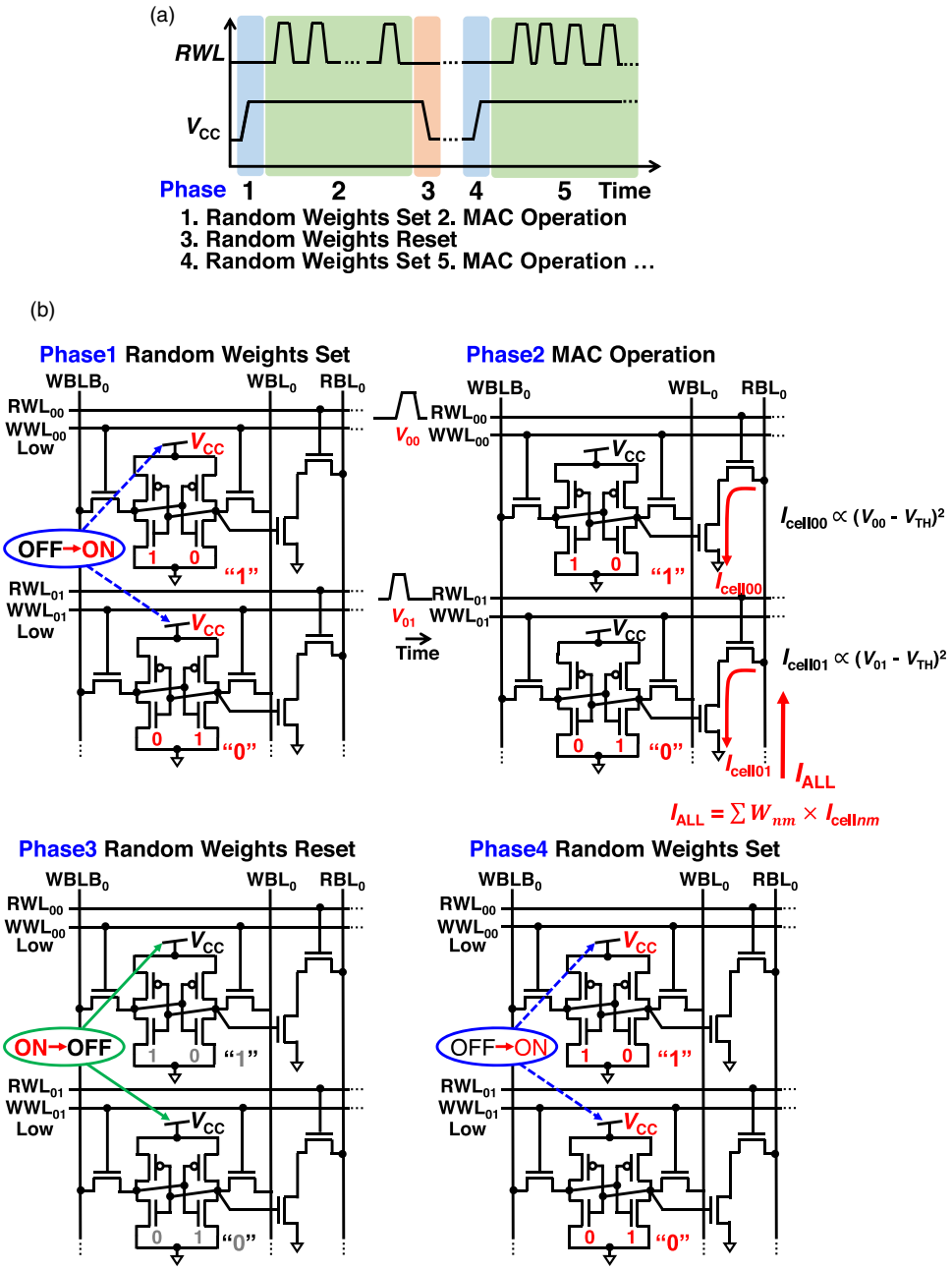

Fig. 4. ARWG operation with (a) input events and V_{CC} and (b) circuit operation in 8T-SRAM.

Phase 3 is random weight resetting. Phase 3 shows when V_{CC} is turned off, such as in cases where users stop using ARWG or change ARWG from training to inference. When power is off in V_{CC}, the random weights used during Phases 1 and 2 cannot be kept stored in the 8T-SRAM array. Phase 4 is random weight setting. Phase 4 indicates how the random weights are restored to perform MAC operation again because ARWG is powered off in Phase 3. Although the random weights are reset in Phase 3, because V_{CC} is turned off the 8T-SRAM requires the random weights to be the same as those during Phases 1 and 2 for MAC operation. In Phase 4, utilizing SRAM device variation, random weights are restored. If power is again supplied to V_{CC}, the 8T-SRAM generates the same weights of "1" or "0" as in Phase 1 because the SRAM device variations are device-specific.

Operation of the proposed ARWG does not imply that device-specific randomness is required for each network but suggests that device-specific randomness can satisfy the random weights in RC and random weight SNN.

The proposed ARWG is used for RC and random weight SNN. On the other hand, PUFs are used for identification, individual authentication, and security technology. RNGs are used for encryption and security technology. Because they have different uses, their randomness requirements are different. To evaluate randomness compared with PUFs and RNGs, repeatability and uniqueness are defined, as shown in Fig. 5(a). Repeatability is defined as a property that reproduces the same output to the same device. Uniqueness is defined as a property that returns a different output for a given input to the different devices. As shown in Fig. 5(b), when the Hamming distance is close to 0.0, repeatability increases, whereas when the Hamming distance is close to 0.5, uniqueness increases. Table I shows the ideal requirements for repeatability and uniqueness from the intended use

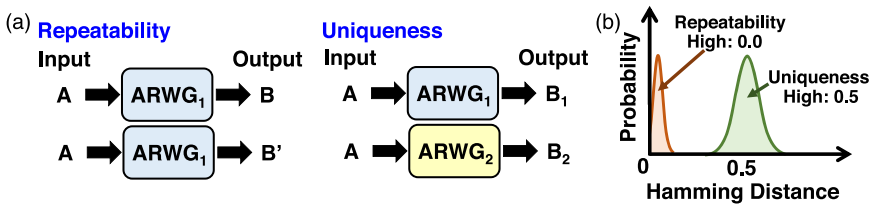

Fig. 5. Definition of (a) repeatability and uniqueness with (b) Hamming distance.

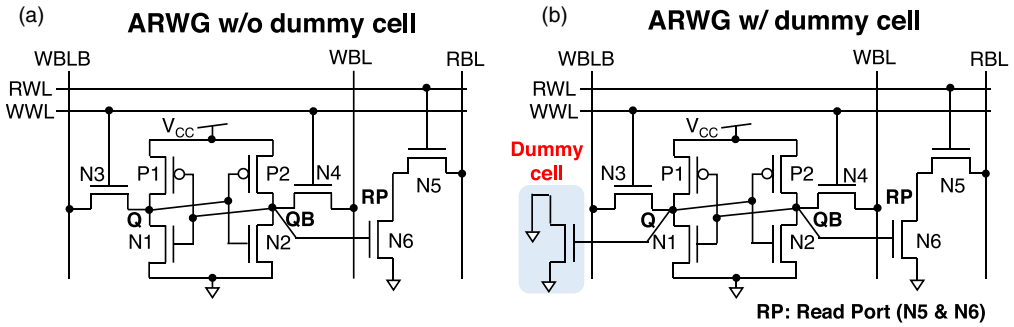

Fig. 6. Circuit configuration of the 8T-SRAM ARWG (a) without a dummy cell and (b) with a dummy cell.

of PUFs, RNGs and ARWG. PUFs require high repeatability and do NOT allow errors by providing ECC in the controller. RNGs must not reproduce the same output. For the proposed ARWG, the requirements from the network side of RC and random weight SNN are shown. The requirements can be used as the criteria that the proposed 8T-SRAM ARWG must satisfy in its fabrication. For repeatability, the proposed ARWG allows a Bit-error rate as high as 0.1%.[4,8] The proposed ARWG generates once random weights for the 8T-SRAM array and uses the random weights for MAC operation. Repeatability for the proposed ARWG is considered as the Bit-error rate when random weights generated once in Phase 1 are restored for MAC operation in Phase 4. In other words, random weights with a Bit-error of less than 0.1%, approximate random weights, are allowed when restoring random weights using SRAM device variations. For uniqueness, PUFs and RNGs require high uniqueness, 0.5 of the Hamming distance. For example, according to a paper,[26] the measured HD of the SRAM array is approximately 0.485–0.497. Although SRAM array is used for PUFs and RNGs, the quality of the randomness requires improvement. However, RC and random weight SNN can allow the value of the measured HD, which is revealed in the following section. In the next section, the randomness of random weights generated by the proposed ARWG is analyzed. Although the randomness is focused on in the following sections, it is important to keep in mind that the proposed ARWG is not just a random weight generator but can also perform CiM function.

3. Randomness of random weight generated by ARWG

In this section, the randomness of random weights generated by the proposed 8T-SRAM ARWG are discussed. Figure 6 shows the circuit configuration of the proposed 8T-SRAM ARWG (a) without a dummy cell and (b) with a dummy cell. While PUFs or RNGs use a 6T-SRAM to generate random numbers, the 6T-SRAM responses are either "1" or "0" due to the difference between the threshold voltages (V_{th}) of both PMOS transistors. The proposed ARWG has extra 2T-SRAM for MAC operation. Figure 7 shows Monte Carlo simulation results from 0–2 ms with (a) V_{CC}, (b) V_Q without a dummy cell and (c) V_Q with a dummy cell. To show further details, Figs. 7(d) and 7(e) describe the results from 0–120 μs with (d) V_Q without a dummy cell and (e) V_Q with a dummy cell. Figure 8 shows the probability of response "1" or "0" from the proposed ARWG with and without a dummy cell. For the proposed ARWG without a dummy cell, the probability of response "1" is higher than response "0". Because QB is connected to N6, QB is subject to N6, and tends to become low. As a result, the probability of response "1" increases. To improve the responses from the proposed ARWG, a dummy cell is applied. Due to the dummy cell, the proposed ARWG improves the probability of the response by 19.2%. Figure 9 shows the probability of responses "1" and "0" from the proposed ARWG without a dummy cell when the width of the read port is changed. The read port consists of N5 and N6 for MAC operation as the function of CiM. If a dummy cell is not used, the capacity of Q is smaller than the capacity of QB.

Table I. Comparison between PUF, RNG and the proposed ARWG.

	Physically unclonable function (PUF)	Random number generator (RNG)	Prop. ARWG
Repeatability	Required w/ECC[a)]	Must not	Acceptable Bit-error rate $< 0.1\%$ [4, 8]
Uniqueness	Required	Required	Approximate

a) For a perfect match, ECC is employed in the controller.

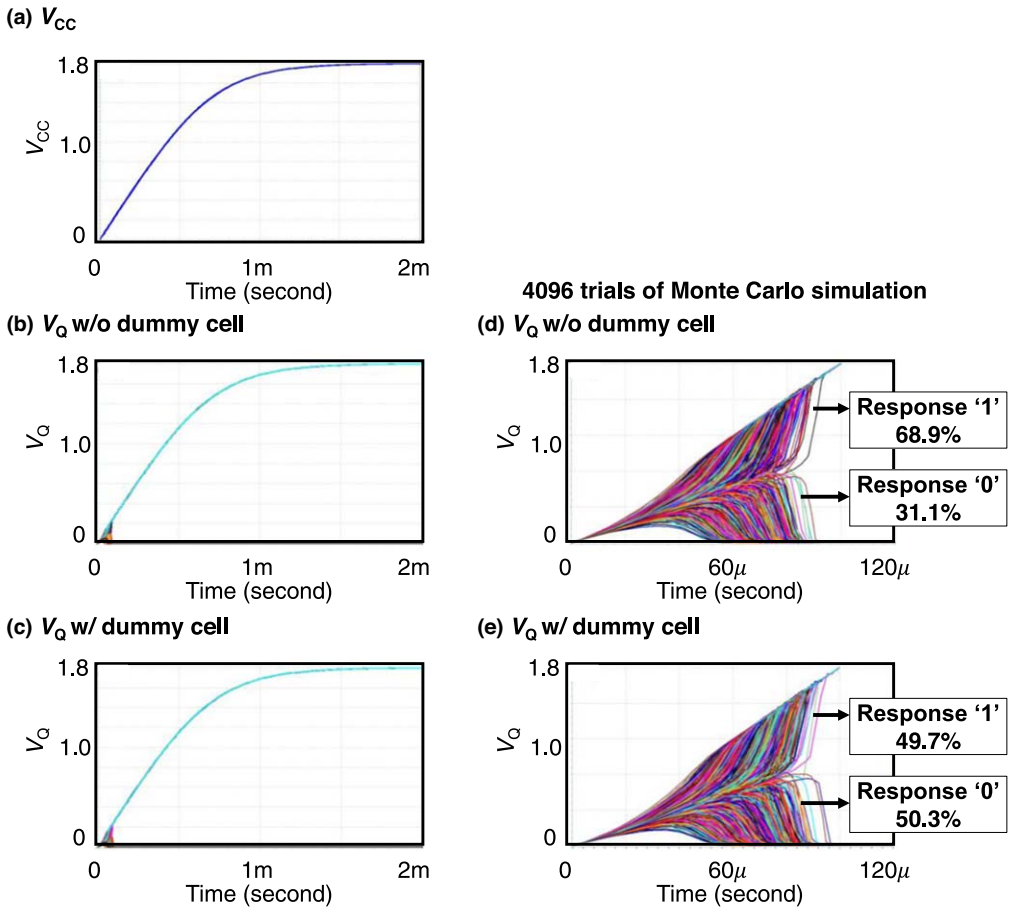

Fig. 7. Monte Carlo simulation results from 0–2 ms with (a) V_{CC} and (b) V_Q without a dummy cell. Monte Carlo simulation results from 0–120 μs with (c) V_Q with a dummy cell, (d) V_Q without a dummy cell and (e) V_Q with a dummy cell.

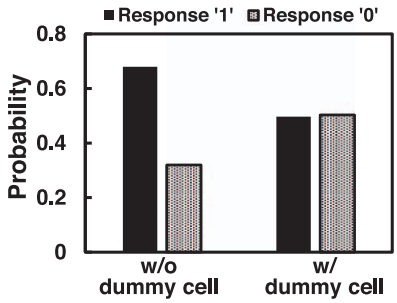

Fig. 8. Probability of response "1" and "0" from the proposed ARWG with and without a dummy cell.

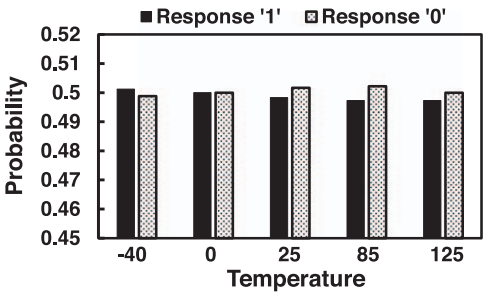

Fig. 10. Probability of response "1" and "0" from the proposed ARWG with a dummy cell with different temperatures.

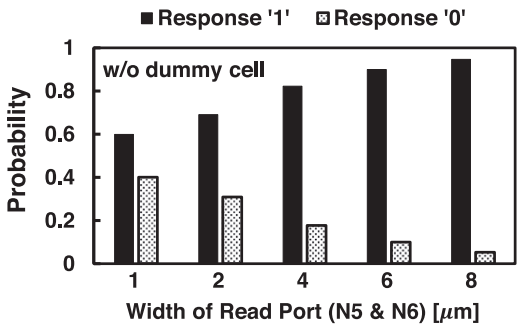

Fig. 9. Probability of response "1" and "0" from the proposed ARWG without a dummy cell when the width of the read port is changed.

Therefore, the probability of response "1" increases. Increasing the width of the read port increases the capacity of QB. As shown in Fig. 9, it can be said that the probability of response "1" increases as the capacity of QB increases. When the read port is wide, the probability of responses between "1" and "0" become extremely unbalanced. To return highly random responses of "1" and "0", where the probability of response is 0.5, a dummy cell is very useful. Although the read port consists of two transistors, one dummy cell is effective, and thus, the circuit area can be reduced.

The randomness of random weights is affected by variation of temperature. Figure 10 shows the probability of responses "1" and "0" from the proposed ARWG with a dummy cell

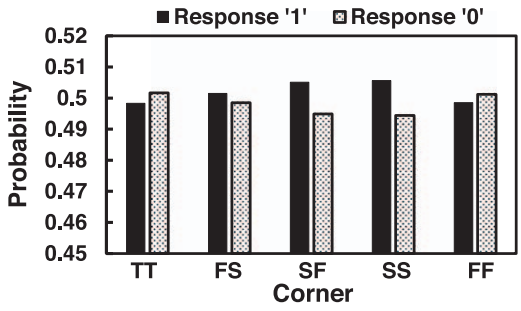

Fig. 11. Probability of response "1" and "0" from the proposed ARWG with a dummy cell with corner variations.

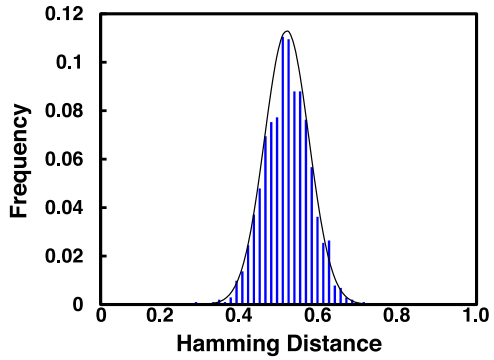

Fig. 12. Hamming distance calculated from 1000 Monte Carlo simulations assuming 8×8 8T-SRAM Array.

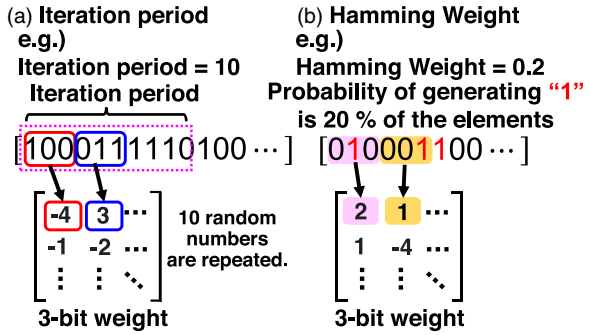

Fig. 13. Random weights generated by specifying (a) the iteration period and (b) Hamming weight.

with different temperatures. Although the temperature increases, the probability of response "1" slightly decreases. The difference between the probability of "1" and "0" is within 1.0% randomness of the proposed ARWG, which can be changed due to process variations. Figure 11 shows the probability of responses from the proposed ARWG with a dummy cell with corner variations. In the case of SS, where both pMOS and nMOS are slow, this results in the most unbalanced probability between responses "1" and "0". However, the difference in probability is less than 1.0%. Finally, Fig. 12 shows the Hamming distance calculated from 1000 Monte Carlo simulations assuming an 8×8 8T-SRAM array. The distribution is centered at 0.5, which can be said to be ideal. As discussed above, the proposed 8T-SRAM ARWG improves the randomness of random weights by applying a dummy cell and generates high quality of random weights within 1.0% error.

4. Evaluation of randomness required from event-based neuromorphic computing

4.1. Iteration period and Hamming weight for RC and random weight SNN

To evaluate the randomness of random weights for RC and random weight SNN, random weights are generated by specifying the iteration period and Hamming weight. Figure 13 shows examples of random weights where each weight has 3-bit precision generated by specifying (a) the iteration period and (b) Hamming weight. The iteration period is defined as the unit of a pattern. Random weights are generated by repeating the iteration period. To correlate the evaluation from the network side with the proposed 8T-SRAM array, the iteration period is converted to Hamming distance, which is generally used for the evaluation of SRAM chips. Hamming weight is defined as the probability of "1" in the element of a vector. Figure 14 shows the relationship between the Hamming distance and iteration period of (a) 3, (b) 10, (c) 100 and (d) 1000. Figure 15 shows the relationship between the Hamming distance and Hamming weight of (a) 0.01, (b) 0.1, (c) 0.3 and (d) 0.5. The Hamming distance is calculated from two W_{REC} in 1000 trials if reservoir size $N = 1000$ and there is 3-bit precision for W_{REC} in RC. As the iteration period becomes longer, variance becomes smaller

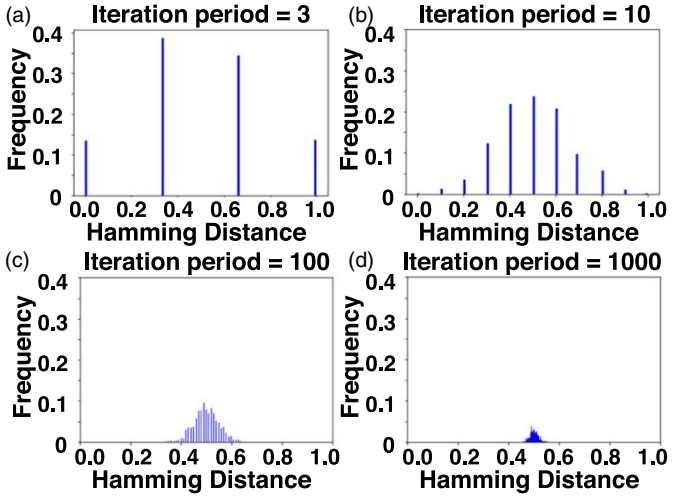

Fig. 14. Relationship between the iteration period and Hamming distance. Frequency at Hamming distance with iteration period of (a) 3, (b) 10, (c) 100 and (d) 1000.

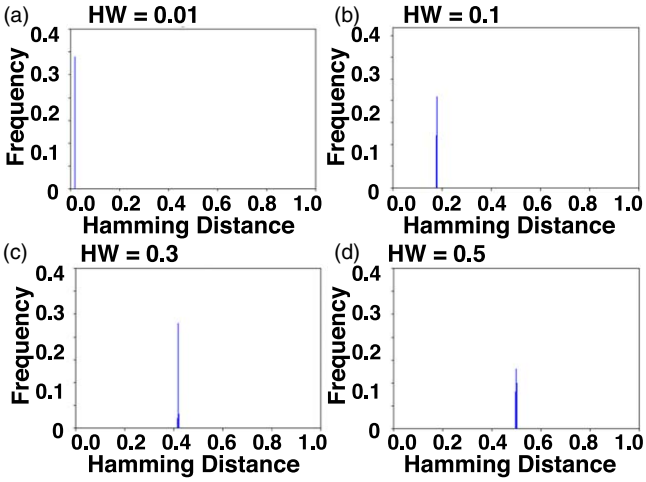

Fig. 15. Relationship between the Hamming weight and Hamming distance. Frequency at Hamming distance with Hamming weight of (a) 0.01, (b) 0.1, (c) 0.3 and (d) 0.5.

and closer to 0.5 of the Hamming distance, resulting in better uniqueness of random weights. Similarly, as the Hamming weight becomes closer to 0.5, the average value of the Hamming distance becomes closer to 0.5, better uniqueness of random weights. In this study, DVS128[31] Gesture Dataset is employed for the gesture recognition task using RC and random weight SNN. In the gesture recognition task, event-based data are classified into 11 categories such as "arm roll" and "right-hand clockwise." The input size of event-based data during training and inference is 128 (height) × 128 (width) for ON and OFF events, respectively. Note that 80% of recognition accuracy is used as the criterion.

4.2. Evaluation of randomness for RC

In RC, the input layer has 162 inputs by leaky integrate-and-fire pooling the event-based data. For the reservoir layer, reservoir sizes of 1000, 500 and 200 are analyzed, respectively. Linear regression is used as the classifier and the output layer has 11 outputs. The reservoir layer and output layer are fully connected and only W_{OUT} between the reservoir layer and output layer is trained.[4] Figure 16 shows the recognition accuracy of RC with different iteration periods (a), (c), (e) in W_{IN}, and (b), (d), (f) in W_{REC}. In evaluating W_{IN} and W_{REC}, the weights of W_{REC} and W_{IN} are set to Gaussian distribution, respectively. Reservoir size N is (a), (b) 1000, (c), (d) 500, and (e), (f) 200. For reservoir size $N = 1000$ and 500, more than 10^2 and 10^3 iteration periods are required for both W_{IN} and W_{REC}, respectively. For reservoir size $N = 200$, although more than 10^3 iteration period is required for W_{IN}, it can be said that recognition accuracy with different iteration periods in W_{REC} is unstable. When the iteration period is less than 10^2 in W_{IN}, recognition accuracy significantly decreases.

Figure 17 shows the recognition accuracy of RC with different Hamming distances (a), (c), (e) in W_{IN}, and (b), (d), (f) in W_{REC}. Reservoir size N is (a), (b) 1000, (c), (d) 500, and (e), (f) 200. For W_{IN}, acceptable Hamming weights are between 0.1 and 0.9 for reservoir size N of 1000 and between 0.1 and 0.8 for reservoir size N of 500. As the reservoir size decreases to 200, the range of acceptable Hamming weights is smaller, from 0.1–0.7. For W_{REC}, Hamming weight from 0.01–0.99 is acceptable regardless of reservoir size. The results indicate that W_{IN} is more sensitive than W_{REC}.

Figure 18 shows recognition accuracy with different Hamming weights in both W_{IN} and W_{REC}. If the Hamming weight is extremely small or large, recognition accuracy decreases significantly. When Hamming weights are 0.1 for W_{IN} and between 0.1 and 0.5 for W_{REC}, recognition accuracy is high.

Figure 19 shows extreme examples with reservoir state transitions. Figure 19(a) shows the reservoir state transitions when (a) the Hamming weights of W_{IN} and W_{REC} are 0.5. The random weights of both W_{IN} and W_{REC} are unique. Figure 19(b) shows the reservoir state transitions when W_{IN} is initialized to a positive value and the Hamming weight of

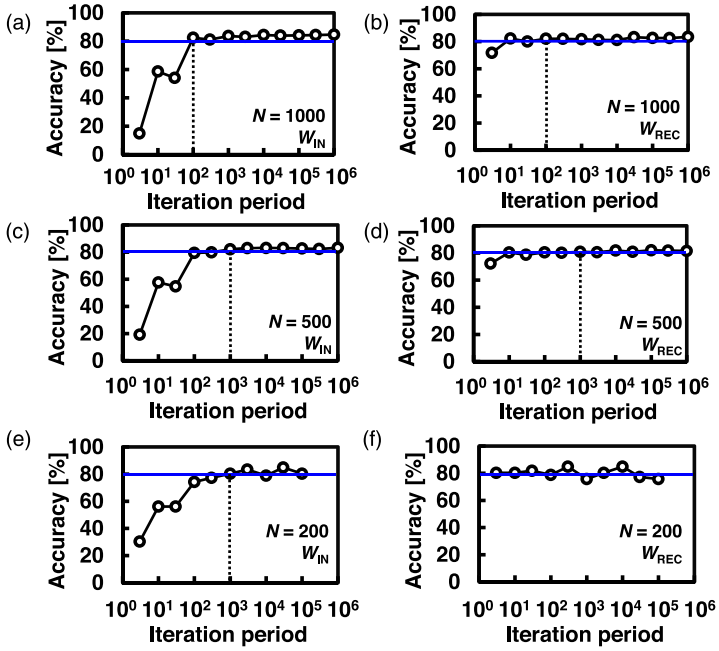

Fig. 16. Recognition accuracy with different Iteration periods of RC in (a), (c), and (e) in W_{IN} and (b), (d), and (f) in W_{REC}. Reservoir size N is (a), (b) 1000, (c), (d) 500, and (e), (f) 200.

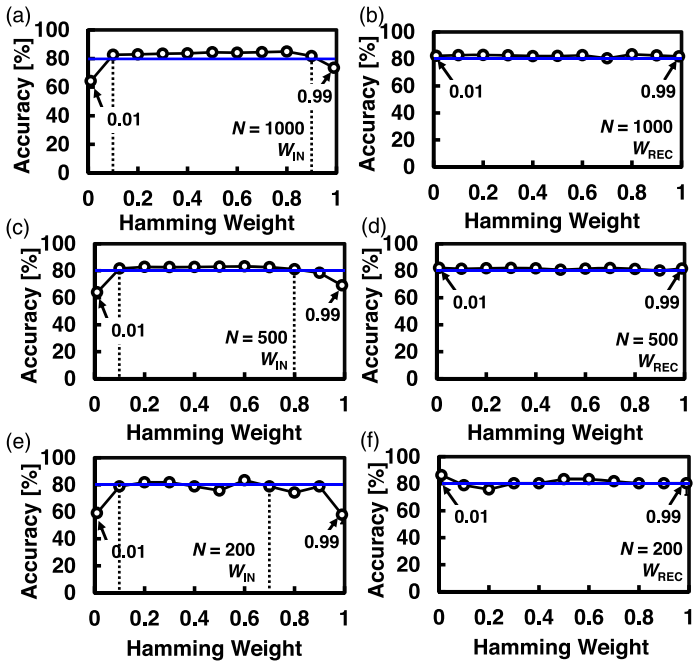

Fig. 17. Recognition accuracy with different Hamming weight of RC in (a), (c), and (e) in W_{IN} and (b), (d), and (f) in W_{REC}. Reservoir size N is (a), (b) 1000, (c), (d) 500, and (e), (f) 200.

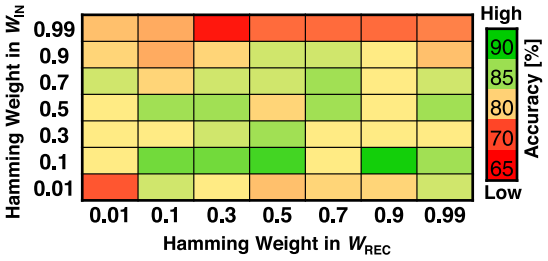

Fig. 18. Recognition accuracy with different Hamming weights in both W_{IN} and W_{REC}.

W_{REC} is 0.5, while Fig. 19(c) shows when the Hamming weight of W_{IN} is 0.5 and W_{REC} is initialized to a positive value. As shown in Fig. 19(b), almost all reservoir states stick to $+1$ and are no longer functional. However, Fig. 19(c) shows that the reservoir state transitions properly despite all positive values of W_{REC}. The results indicate that the function of W_{IN} is to transmit input, while W_{REC} is only intended to assist short-term memory.

4.3. Evaluation of randomness for random weight SNN

In random weight SNN, random weights are used in the convolution layer. The size of the convolution layer is 17×17.[8] Under the same conditions as RC, linear regression for

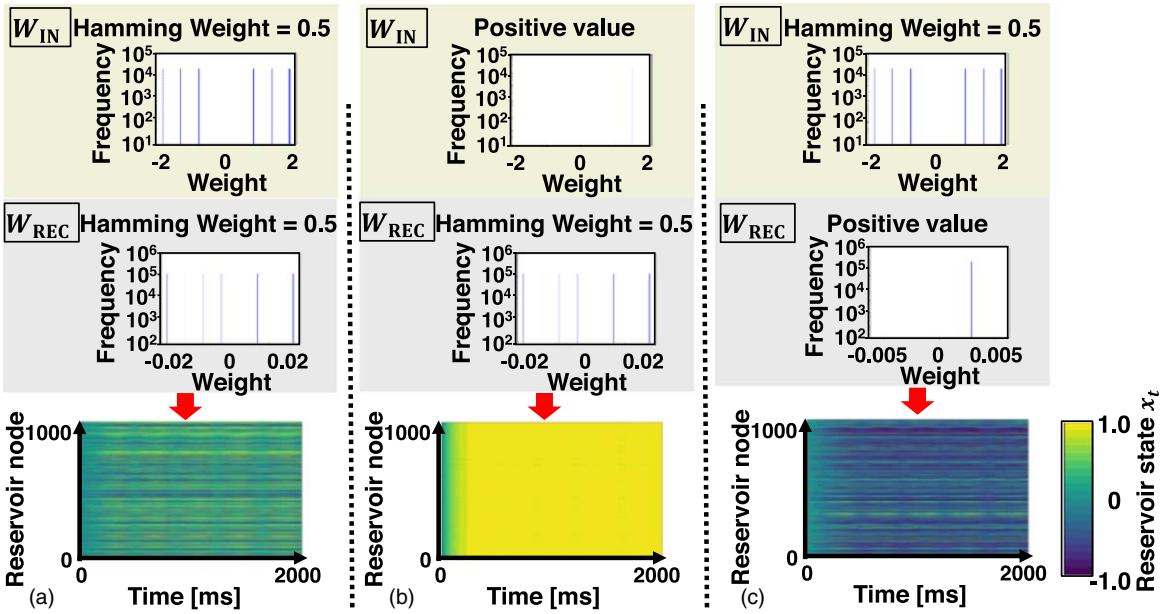

Fig. 19. Extreme examples with reservoir state transitions when (a) the Hamming weights of W_{IN} and W_{REC} are 0.5. (b) W_{IN} is initialized to a positive value while the Hamming weight of W_{REC} is 0.5. (c) Hamming weight of W_{IN} is 0.5 while W_{REC} is initialized to a positive value.

Table II. Summary of this study.

	ARWG for reservoir computing								
	Reservoir size $N = 200$		Reservoir size $N = 500$		Reservoir size $N = 1000$		ARWG for random weight SNN	PUF	RNG
	W_{IN}	W_{REC}	W_{IN}	W_{REC}	W_{IN}	W_{REC}			
Uniqueness (HD)	$0.45 < HD < 0.55$	—	$0.45 < HD < 0.55$	$0.45 < HD < 0.55$	$0.3 < HD < 0.7$	$0.3 < HD < 0.7$	$0.3 < HD < 0.7$	0.5	—
Hamming weight (HW)	$0.1 < HW < 0.7$	$0.01 < HW < 0.99$	$0.1 < HW < 0.8$	$0.01 < HW < 0.99$	$0.1 < HW < 0.9$	$0.01 < HW < 0.99$	$0.01 < HW < 0.9$	0.5	0.5

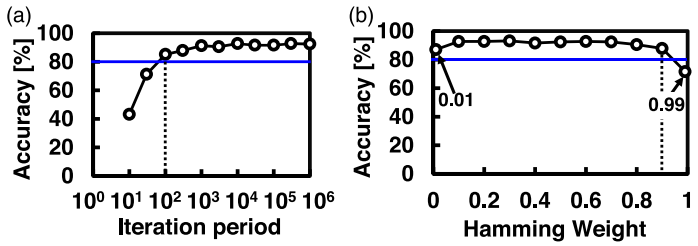

Fig. 20. Recognition accuracy with different (a) iteration periods and (b) Hamming weights.

the classifier and 11 outputs are used. For random weight SNN, 10^2 or more Iteration period [Fig. 20(a)] and Hamming weight between 0.01 to 0.9 [Fig. 20(b)] are required in convolutional weights. Compared with RC, both the iteration period and Hamming weight of random weight SNN have similar trends to W_{IN} of RC.

As a result, the randomness required for RC and random weight SNN is significantly lower than that of PUFs and RNGs. This means that approximate random weights are allowed for RC and random weight SNN.

5. Conclusion

In this paper, dual integration design of ARWG and CiM is proposed for event-based RC and random weight SNN. This paper discusses the functions and circuit operation of the proposed ARWG, and the randomness of the random weights generated by the proposed ARWG. For the circuit design of the proposed ARWG, a dummy cell is proposed in order to improve the probability of response "1" or "0" for generating random weights. With this proposed dummy cell, the probability of response "1" or "0" has an error of less than 1.0% of the ideal value of 0.5. Table II shows a comparison of the randomness with Hamming distance and Hamming weight for ARWG, PUFs and RNGs. For uniqueness, a Hamming distance between 0.3 and 0.7 is accepted for ARWG in the case of RC ($N = 1000$) and random weight SNN, while the Hamming distance for PUFs must be 0.5. For Hamming weight, although 0.5 is required for PUFs and RNGs, ARWG for RC and random weight SNN accepts Hamming weight between 0.01 and 0.99 and Hamming weight between 0.01 and 0.9, respectively. Because low randomness is allowed for RC and random weight SNN, the proposed dual integration of ARWG and CiM can facilitate event-based RC and random weight SNN training by generating approximate random weights and performing CiM operations.

Acknowledgments

This study was supported by JST CREST, Japan (Grant No. JPMJCR20C3).

ORCID iDs

Chihiro Matsui https://orcid.org/0000-0003-4594-6839
Ken Takeuchi https://orcid.org/0000-0002-9345-6503

1) A. Basu et al., IEEE J. Emerg. Sel. Top. Curcuits Syst. **8**, 6 (2018).
2) M. Davies et al., IEEE Micro **38**, 82 (2018).
3) E. Covi, S. Brivio, A. Serb, T. Prodromakis, M. Fanciulli, and S. Spiga, IEEE Int. Symp. on Circuits and Systems, 2016, p. 393, 10.1109/ISCAS.2016.7527253.
4) S. Koshino, C. Matsui, and K. Takeuchi, IEEE Silicon Nanoelectronics Workshop, 2022, p. 25.
5) H. Jaeger, GMD Tech. Rep. **148**, 34 (2001).
6) H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert, Neural Netw. **20**, 225 (2007).
7) N. Schaetti, M. Salomon, and R. Couturier, IEEE Int. Conf. on Computational Science and Engineering and IEEE Int. Conf. on Embedded and Ubiquitous Computing and Int. Symp. on Distributed Computing and Applications for Business Engineering, 2016, p. 484.
8) S. Koshino, N. Misawa, C. Matsui, and K. Takeuchi, IEEE Silicon Nanoelectronics Workshop, 2022, p. 95.
9) S. B. Shrestha and G. Orchard, Conf. Neural Information Processing Systems, 2018.
10) A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, (2019), arXiv1804.08150v4.
11) W. Maass, Neural Netw. **10**, 1659 (1997).
12) C. Posch, D. Matolin, and R. Wohlgenannt, IEEE J. Solid-State Circuits **46**, 259 (2011).
13) C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbrück, IEEE J. Solid-State Circuits **49**, 2333 (2014).
14) B. Son et al., IEEE Int. Solid-State Circuits Conf., 2017, p. 66.
15) T. Finateu et al., IEEE Int. Solid-State Circuits Conf., 2020, p. 112.
16) P. Lichtsteiner, C. Posch, and T. Delbrück, IEEE Int. Solid-State Circuits Conf., 2006, p. 508.
17) R. Zhang et al., Symp. on VLSI Circuits, 2021, p. 1.
18) B. Gao, B. Lin, X. Li, J. Tang, H. Qian, and H. Wu, IEEE Trans. Electron Devices **69**, 536 (2022).
19) W. Y. Yang, B. Y. Chen, C. C. Chuang, E. R. Hsieh, K. S. Li, and S. S. Chung, IEEE Int. Electron Devices Meeting, 2020, p. 39.3.1.
20) S. Ramanujam and W. Burleson, Int. Symp. on Quality Electronic Design, 2021, p. 257.
21) S. Koshino, N. Misawa, C. Matsui, and K. Takeuchi, Ext. Abstr. Int. Conf. Solid-State Devices and Materials, 2023, p. 471.
22) C. Matsui, K. Higuchi, S. Koshino, and K. Takeuchi, Int. Conf. Solid-State Devices and Materials, 2021, p. 676.
23) H. Zhang et al., IEEE Int. Symp. on Circuits and Systems, 2021, p. 1.
24) W. Zhao et al., IEEE Trans. Semicond. Manuf. **22**, 196 (2009).
25) Y. Cui, C. Gu, C. Wang, M. O'Neill, and W. Liu, IEEE Access **6**, 28478 (2018).
26) Z. Su et al., IEEE Trans. Nucl. Sci. **69**, 333 (2022).
27) M. Cortez, A. Dargar, S. Hamdioui, and G.-J. Schrijen, IEEE Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology System, 2012, p. 1.
28) M. Cirtez, S. Hamdioui, and R. Ishihara, Latin-American Test Symp., 2015, p. 1.
29) J. Yang et al., IEEE Int. Electron Devices Meeting, 2020, p. 28.6.1.
30) A. Garg and T. T. Kim, IEEE Int. Symp. on Circuits and Systems, 2014, p. 1941.
31) A. Amir et al., IEEE Conf. on Computer Vision and Pattern Recognition2017p. 7243.

