Finite State Automata Design using 1T1R ReRAM Crossbar

Simranjeet Singh*¶, Omar Ghazal†‖, Chandan Kumar Jha‡, Vikas Rana¶, Rolf Drechsler‡§,
Rishad Shafik†, Alex Yakovlev†, Sachin Patkar*, Farhad Merchant†
*Indian Institute of Technology Bombay, India,‖University of Mosul, Iraq, †Newcastle University, UK,
‡University of Bremen, Germany, §DFKI GmbH, Germany, ¶Forschungszentrum Jülich GmbH, Germany,
{simranjeet, patkar}@ee.iitb.ac.in, {si.singh, v.rana}@fz-juelich.de, chajha@uni-bremen.de, drechsler@uni-bremen.de,
{O.G.G.Awf2, rishad.shafik, alex.yakovlev, farhad.merchant}@newcastle.ac.uk

***Abstract*—Data movement costs constitute a significant bottleneck in modern machine learning (ML) systems. When combined with the computational complexity of algorithms, such as neural networks, designing hardware accelerators with low energy footprint remains challenging. Finite state automata (FSA) constitute a type of computation model used as a low-complexity learning unit in ML systems. The implementation of FSA consists of a number of memory states. However, FSA can be in one of the states at a given time. It switches to another state based on the present state and input to the FSA. Due to its natural synergy with memory, it is a promising candidate for in-memory computing for reduced data movement costs. This work focuses on a novel FSA implementation using resistive RAM (ReRAM) for state storage in series with a CMOS transistor for biasing controls. We propose using multi-level ReRAM technology capable of transitioning between states depending on bias pulse amplitude and duration. We use an asynchronous control circuit for writing each ReRAM-transistor cell for the on-demand switching of the FSA. We investigate the impact of the device-to-device and cycle-to-cycle variations on the cell and show that FSA transitions can be seamlessly achieved without degradation of performance. Through extensive experimental evaluation, we demonstrate the implementation of FSA on 1T1R ReRAM crossbar.**

***Index Terms*—FSA, Machine Learning, ReRAM, Memristors, In-Memory Computing**

I. Introduction

In-memory computing (IMC) using memristive devices has become popular in elevating the von Neumann bottleneck by storing and processing data in memory, especially for machine learning (ML) applications [1]–[5]. Memristive devices connected in a crossbar structure allow the program to run in parallel, making the IMC comparable with conventional computing in terms of energy efficiency and performance [6]. Memristive devices, such as resistive random access memory (ReRAM) [7], can be configured as multi-level cell [8], where the device has multiple intermediate states between low resistive (LRS) and high resistive state (HRS). Still, modern ML workloads require massive storage resources and parallelism to accelerate. However, finite state automata (FSA) capture the real-world constraint of finite memory to implement learning applications [9]. The multi-level behavior of ReRAM can be used to design FSA in memory.

FSA is an abstract machine that can be at exactly one of the finite number of states at any given time. FSA changes its state from one to another, called a transition, when a specific event occurs [10]. FSA has a significant edge on applications that require low-latency or real-time processing, such as automated verification [11], autonomous vehicles [12], and tsetlin machine [13]. The energy efficiency, high density, and IMC properties of ReRAM devices make them suitable for FSA implementation [14] [15]. Multiple states of ReRAM, between LRS and HRS, can be mapped to the states of FSA. An FSA cell can be represented as a single ReRAM device with a CMOS transistor in series (1T1R), as shown in Fig. 1(a). The material stack of the memristive cell is shown in Fig. 1(b), along with the I-V characteristics of the 1T1R cell in Fig. 1(c). Fig. 1(c) shows the multiple states in SET (switching to LRS) and RESET (switching to HRS) states, which have been utilized in this study to implement the FSA on ReRAM devices [16].

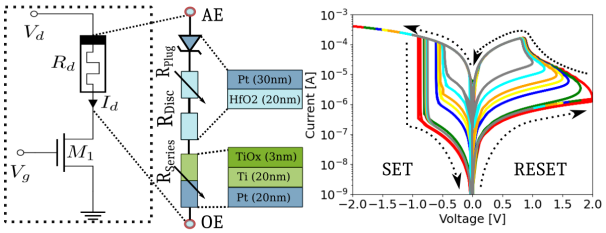

Fig. 1: 1T1R cell for FSA (a) cell structure, (b) device material stack, and (c) multi-level characteristic of a device.

However, transitions of FSA from one state to another with accurate detection of the current FSA state under device variations remains challenging. In this paper, we propose an architecture to implement the FSA on the ReRAM crossbar for IMC. We show in the proposed architectures how a single 1T1R cell can be used to design a six-state FSA. Next, We evaluate the proposed architecture in terms of energy efficiency and performance. Moreover, we assess the architecture under device variations such as device-to-device (D2D) and cycle-to-cycle (C2C) [17]. To summarize the main contributions:

- The integration of 1T1R ReRAM technology into FSA design by utilizing the multi-level behavior of ReRAM. The gradual RESET method has been utilized to achieve the multi-level behavior.
- Investigation of the impact of D2D and C2C variations on state transitions and detection.
- Extensive evaluation of the efficiency of the proposed architecture in terms of energy efficiency and latency.

The remainder of the paper is organized as follows: Section II presents the proposed architecture to design FSA on ReRAM. Section III presents experimental results and validation for the design. Finally, Section IV concludes the paper.

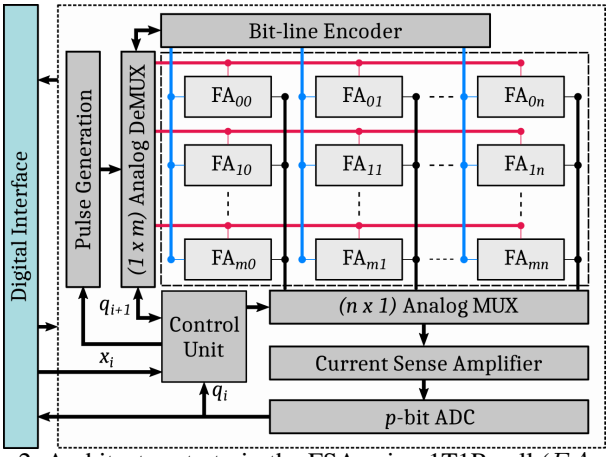

Fig. 2: Architecture to train the FSA using 1T1R cell (FA_{mn}), where '*m*' and '*n*' represent rows and columns in a crossbar, respectively, and '*p*' is the number of ADC bits, which is given as $\lceil \log_2(s) \rceil$, '*s*' is the number of states in a FA cell.

II. Proposed Architecture

This section discusses the architecture to implement the FSA on the 1T1R ReRAM crossbar, depicted in Fig. 2. At the core of the proposed architecture, 1T1R cells connected in a crossbar structure called finite automaton (FA) are used.

A. FA using 1T1R cell

The FA cell is constructed using one memristor and one NMOS transistor in series. The structure of a single FA cell is shown in Fig. 1. The I-V characteristics shown in Fig. 1(c) show the multi-level characteristics of FA, which have been mapped to different states in FA. In this study, the FA has seven states ('*s*') from S_0 to S_6, starting from LRS to HRS. S_0 has a minimum resistance of around 7.8KΩ and S_6 has maximum resistance of around 1.5MΩ. All other states are mapped into the intermediate values between S_0 to S_6. Each FA in the crossbar represents six states (excluding S_0), and they can be independently programmed. However, multiple FAs can be combined to run a complex application that needs more than six states. For this work, we will limit our study to the working of independent FA in the crossbar. The state transitions of FA are examined next.

B. State transitions in FA

An FA has a finite number of states, seven in this study, and it changes the state from one to another or the next state (q_{i+1}) based on the input (i_i) and current state (q_i) similar to a mealy machine. A pulse generation module in Fig. 2 can generate the different pulse widths of a fixed voltage amplitude. The control circuit selects the appropriate signal for the next state based on the present state and input. The parameter to switch the state from S_0 to any possible state is given in Table I. Next, the analog demultiplexer (DeMUX) and bit-line encoder select a FA in the crossbar by applying an ON voltage to the NMOS transistor and the required pulse signal at the row of the crossbar.

For the functional correctness of the FSA transition, it is important to identify the present state correctly. FSA can

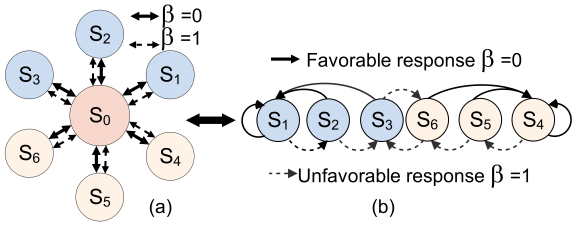

Fig. 3: State transitions in a FA cell (a) and (b) shows the use of FSA as Krinsky learning automaton [18].

jump from the present state to any other possible state in FA. So, it is expected from the FSA that it should give the same current value for the same state transition. However, the gradual RESET method limits the transitions of the states only in the forward direction ($S_1 \rightarrow S_6$). In order to change the state which is less than the current state (backward direction), FA needs to switch to S_0 (intermediate state) before switching to the next desired state. Also, it is expected that states after switching can correctly be identified during forward or backward direction switching. Therefore, an intermediate state is added in every state transition in FA, which provides three main advantages; (a) switching to any state in FA, (b) state retention while looping in the same state, and (c) reducing the complexity of control circuitry.

The state transition graph of single FA is shown in Fig. 3(a). It can be used for learning applications such as Krinsky automaton [18]. The mapping of a Krinsky learning automaton has been shown in Fig. 3(b). The unfavorable response ($\beta = 1$) gradually moves toward the boundary states separating the two actions, behaving as binary states. In another response ($\beta = 0$), S_i switches to S_1 and S_4 for $(1 \leq i \leq 3)$ and $(4 \leq i \leq 6)$, respectively. The proposed architecture is highly flexible regarding its control circuit and switching characteristics. It provides the facility to transit from the current state to any next state via S_0 with an adaptive STG control unit. The proposed approach can accommodate FSM with more than 6 states by utilizing multiple 1T1R cells arranged to represent different states and encoded into binary form, offering flexibility for varying numbers of states.

C. Peripherals to control FSA

Various peripherals around the crossbar are required to implement FSA on the ReRAM crossbar, as shown in Fig. 2. The control unit decided the transitions in FA, which are functions of x_i and q_i.

Pulse generation module generates the voltage pulses with a required duration for transitioning the state from one to another according to Table I. Since state S_0 is an intermediate state,

TABLE I: State transition of 1T1R cell

V_{fixed} = 1.8V, V_{SET}= $-2V$, V_{READ}= $0.1V$			
State	P_{width} (ns)	I_d (μA)	Resistance (KΩ)
S0	10ns at -2V	12.8	7.8
S1	5ns	12.6	8.0
S2	10ns	1.6	95.2
S3	15ns	0.56	196.1
S4	30ns	0.3	342.5
S5	60ns	0.2	588.2
S6	150ns	0.07	1492.5

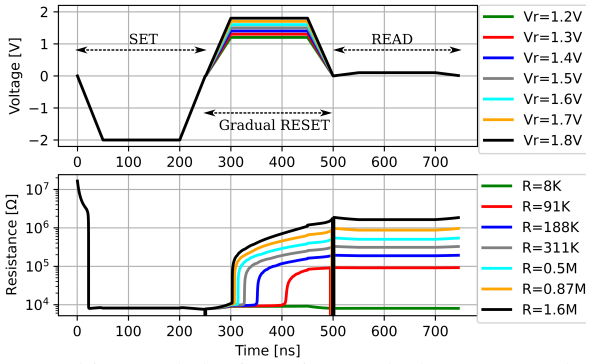

Fig. 4: Multi-states behavior using gradual RESET method.

the small gap between S_0 and S_1 will not pose an issue in state estimation.

Multiplexer (MUX), demultiplexer (DeMUX), and bit-line encoder are the selection peripherals. For a given $(m \times n)$ size crossbar, $(1 \times m)$ sized DeMUX are attached to select a row of the crossbar to apply a pulse for transition. Bit-line encoder enables the transistor of selected FA. At column, $(n \times 1)$ MUX is connected to read the state of a FA.

Current sense amplifier (CSA) and Analog-to-digital converter (ADC) are sensing peripherals. The CSA converts the current to an amplified voltage, which is further used by the ADC to detect the current state of FA. A common CSA and ADC have been used in the proposed architecture, where a FA can be read in each cycle. A 'p' bit ADC is required to correctly detect the 's' number of states in FA.

Control unit contains the algorithms to switch the states of FA in sequence. The control unit manipulates the select lines of MUX and DeMUX to select an FA and generate the required pulses for transitions via the pulse generation module. It takes input from the digital interface (x_i) and ADC (q_i) and calculates the required control signal to switch the state to q_{i+1}; alternatively, $q_{i+1}(q_i, x_i)$, where $0 <= |x_i| <= 1$ and $S_1 <= q_i <= S_6$.

III. Experimental Results

This section evaluates the proposed methodology in terms of energy efficiency and area. First, we study the switching characteristics of a FA and the state transitions from one to another. Next, we look at the impact of D2D and C2C variations on state transitions.

A. 1T1R cell characterization

1T1R cell used for this study is designed using a Pt/Ti/TiO_x/HfO_2/Pt material stack memristive devices in series with a 45nm transistor. The material stack used in the memristive device adheres to the characteristics of the experimental devices. Fig. 1 shows the configuration of the 1T1R cell along with the material stack of the device and I-V characteristics. Table II shows the parameters used for the ReRAM device model. The device has multiple-state characteristics, and the gradual RESET method has been used to achieve this behavior. In the gradual RESET method, the device is initialized to an LRS state by applying a positive voltage of a specific duration

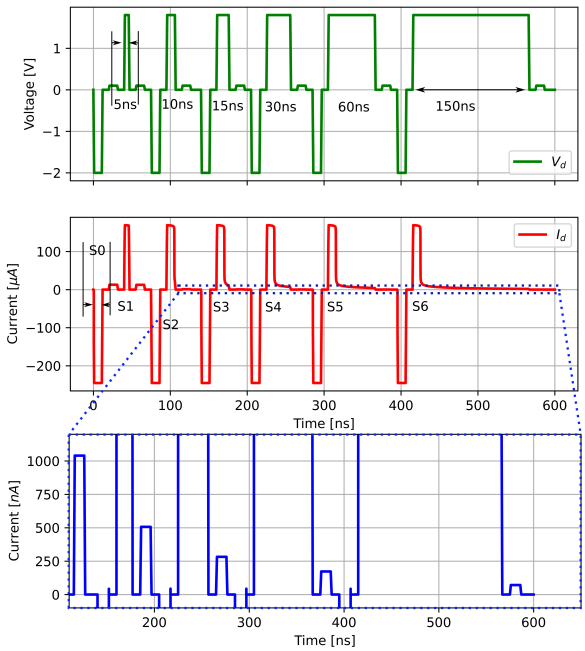

Fig. 5: State switching from S_0 to S_6. For the desired state, the device is first switched to S_0 and then directly switched to the required state by applying the appropriate pulse.

on the ohmic electrode (OE). Next, the device is switched into HRS by applying RESET pulses gradually. Fig. 4 shows the gradual RESET method, where the voltages across the device and its corresponding resistance states have been plotted. The gradual RESET method results in multi-level behavior of 1T1R cells. Additionally, a transistor in series helps to control the current precisely.

B. Realization of state transitions

As the memristive devices change their states based on the value of integral over time of the applied voltage, the width of the applied pulse can be used for state transitions instead of gradually varying voltage. Table I shows the switching of FA from S_0-S_6 with different pulse widths. The state S_0 is the initial state or LRS state, which can be switched by applying a 10ns pulse of -2V voltage. The states from S_1 to S_6 (6 states) are the actual states used for FA. Generally, the FA switches the state in the forward (S_1-S_6) or backward (S_6-S_1) directions. However, the gradual method only works in forward state switching. The backward state switching is tackled by switching to the intermediate state (S_0) before switching to the desired state. For example, the current state is $S3$, and the next expected state is S_2. In this case, the device first is switched to S_0 and then switched from S_0 to S_2 ($S_3 \rightarrow S_0 \rightarrow S_2$). To

TABLE II: Model parameters

Symbol	Value	Symbol	Value
l_{cell}	3 nm	l_{det}	4 nm
r_{det}	20 nm	N_{plug}	20×10^{26} m^{-3}
a	0.25 nm	μ_n	1×10^{-6} m^2/Vs
ϵ	17 ϵ_0	$N_{disc,min}$	0.008×10^{26} m^{-3}
$\epsilon\phi\beta$	5.5 ϵ_0	$N_{disc,max}$	20×10^{26} m^{-3}
$e\phi_{\beta n0}$	0.3 eV	$e\phi_{\beta n}$	0.1 eV
ΔW_A	0.7 eV	A	0.00392 1/Ω
R_{series}	650 Ω	R_o	719.244 Ω
$R_{th,line}$	90.47 $K\Omega$	R_{th0}	1.572×10^7 Ω

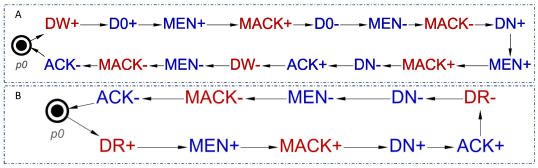

Fig. 6: STG of the control unit, (a) state transition cycle, and (b) reading the present state in FA.

reduce the complexity of the control circuit and accurate state detection, the intermediate state has been used in forward, and backward switching. Fig. 5 shows the switching of the states through the intermediate state. Table I also shows the states' current and resistance, indicating enough margin between the state's current/resistance for accurate state detection.

C. Impact of variations

The D2D and C2C variation in ReRAM devices can affect the switching behavior. To simulate D2D variations, the random set of values for device parameters such as radius, length, and minimum and maximum oxygen vacancy in the disc are drawn from the experiment-verified Gaussian distribution [17]. These variations were then independently applied to the available devices in the crossbar. C2C variations are simulated by changing the variable parameters in a period of a single cycle. It has been observed that states from S_0 to S_3 (low states) have a larger impact of the variations compared to high states (S_4 to S_6), which is around $\pm 50\%$ change in the read current for low states and $\pm 20\%$ for high states. However, for low states, the margin between the state is more than five times which enables the accurate detection of the state even if the variations have a larger impact. Moreover, switching through the intermediate state prevent error accumulation over time and reduces the impact of variations.

D. Control circuitry

An asynchronous digital controller has been designed using Workcraft [19] to coordinate data flow between different components in the architecture. Faster operation and lower power consumption are some of the advantages of asynchronous circuits over global clocked-based circuits. The control unit's signal transition graphs (STG) [20] to perform state detection (Read cycle) and state transitions (write cycle) are shown in Fig. 6. When there is a request to read the state of FA, the control unit receives a data reading strobe (*DR*+) from the digital interface environment, and a reading cycle begins. It activates MUX and bit line encoder to select a device (MEN+). Next, it starts reading the data for the n_{th} FA (DN+) after receiving acknowledgment from MUX (MACK+). Lastly, the data is read, and the next read cycle is prepared by resetting DN-, MEN-, and MACK-. The final ACK signal is sent out as an acknowledgment for the digital interface.

TABLE III: 1T1R cell energy consumption

State Switching	Intermediate State	Energy (pJ)	State Switching	Intermediate State	Energy (pJ)
$S_0 \rightarrow S_1$	–	1.74	$S_1 \rightarrow S_2$	S_0	8.2
$S_2 \rightarrow S_3$	S_0	8.3	$S_3 \rightarrow S_4$	S_0	8.5
$S_4 \rightarrow S_5$	S_0	8.8	$S_5 \rightarrow S_6$	S_0	9.25
Average energy			**7.5pJ**		

The control circuit for state transitions provides a facility to transition the FA state from any present state to any other possible state. This increases the flexibility of the proposed architecture to run any FSA application. However, every state transition in FA is done via S_0 to maintain functional correctness and reduce controller complexity. The control circuitry handles this situation by generating an acknowledgment signal, which includes the transition of S_0 and desired state (S_n). The transition cycle is initiated whenever there is a request on a DW+ signal from the environment. The first step for state transition includes switching to the S_0 state. The FA is selected by enabling the row multiplexer and bit-line encoder (MEN+). The FA changes the state from the n_{th} state to S_0 and disables the row MUX before sending the acknowledgment for the final transition. At this time of the cycle, FA is in S_0 and ready to be switched to the desired state (S_n). Similar to S_0 switching, the final transition starts by enabling the peripherals. Before the transition cycle is finished, the signal DN- initiates the ACK+ that will be delivered to the digital interface, resetting DW-, MEN-, and MACK-. The final acknowledgment for the digital interface is ACK- signal, which indicates a successful state transition.

E. Energy and latency analysis

Each state transition in FA consumes different energy, which is given in Table III. In every transition in FA, 7.5pJ energy is consumed on average. An FA has to switch to the intermediate state before the desired transition, which increases energy consumption. However, the intermediate state makes the state switching robust against D2D and C2C variations. As the proposed architecture used different pulse duration to state transitions, state S_6 takes 150ns pulse. Hence the pulse generation module generates each pulse for a 150ns period with varying widths. However, latency can be improved further by increasing the voltage amplitude of the applied pulse, which increases energy consumption. So, there is a trade-off between energy consumption and latency.

IV. Conclusions

In this work, for the first time, we proposed the architecture to implement the FSA using a 1T1R ReRAM crossbar. This paper offers insights into the scope of FSA utilizing ReRAM and CMOS technology. We use the multi-level characteristics of ReRAM, achieved using the gradual RESET method, to implement FSA on the crossbar. We studied the impact of variation on state transitions. Finally, we evaluated the proposed framework in terms of latency and energy consumption. The results are encouraging and demonstrate the potential for using ReRAM-based FSA designs. We will explore the prototyping of the proposed designs and test the architecture with learning automaton applications in the future.

Acknowledgments

This work was supported in part by the Federal Ministry of Education and Research (BMBF, Germany) in the project NEUROTEC II under Project 16ME0398K, Project 16ME0399, German Research Foundation (DFG) within the Project PLiM (DR 287/35-1, DR 287/35-2) and through Dr. Suhas Pai Donation Fund at IIT Bombay.

REFERENCES

[1] G. Pedretti and D. Ielmini, "In-memory computing with resistive memory circuits: Status and outlook," *Electronics (Switzerland)*, vol. 10, 05 2021.

[2] S. Yu, A. Soltan, R. Shafik, T. Bunnam, F. Xia, D. Balsamo, and A. Yakovlev, "Current-mode carry-free multiplier design using a memristor-transistor crossbar architecture," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 638–641.

[3] C. K. Jha, P. L. Thangkhiew, K. Datta, and R. Drechsler, "Imagin: Library of imply and magic nor-based approximate adders for in-memory computing," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 8, no. 2, pp. 68–76, 2022.

[4] S. Froehlich and R. Drechsler, "Unlocking approximation for in-memory computing with cartesian genetic programming and computer algebra for arithmetic circuits," *it - Information Technology*, vol. 64, no. 3, pp. 99–107, 2022.

[5] F. Staudigl, F. Merchant, and R. Leupers, "A survey of neuromorphic computing-in-memory: Architectures, simulators, and security," *IEEE Design & Test*, vol. 39, no. 2, pp. 90–99, 2022.

[6] A. Sebastian, M. Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol. 15, 03 2020.

[7] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-based resistive switching memories – nanoionic mechanisms, prospects, and challenges," *Advanced Materials*, vol. 21, pp. 2632–2663, 07 2009.

[8] A. Siemon, D. Wouters, S. Hamdioui, and S. Menzel, "Memristive device modeling and circuit design exploration for computation-in-memory," 05 2019, pp. 1–5.

[9] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 114–125, 1959.

[10] A. Rezvanian, B. Moradabadi, M. Ghavipour, M. M. Daliri Khomami, and M. R. Meybodi, *Introduction to Learning Automata Models*. Cham: Springer International Publishing, 2019, pp. 1–49. [Online]. Available: https://doi.org/10.1007/978-3-030-10767-3_1

[11] K. Luo and Y. Liu, "Automatic verification of fsa strategies via counterexample-guided local search for invariants," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 1814–1821.

[12] M. Zhang, N. Li, A. Girard, and I. Kolmanovsky, "A finite state machine based automated driving controller and its stochastic optimization," 10 2017, p. V002T07A002.

[13] O.-C. Granmo, "The Tsetlin Machine – A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic," 2018. [Online]. Available: https://arxiv.org/abs/1804.01508

[14] A. Wheeldon, R. Shafik, T. Rahman, J. Lei, A. Yakovlev, and O.-C. Granmo, "Learning automata based energy-efficient AI hardware design for IoT applications," *Philosophical Tran of the Royal Society A*, vol. 378, no. 2182, p. 20190593, 2020.

[15] Y. Halawani, B. Mohammad, M. Abu Lebdeh, M. Al-Qutayri, and S. F. Al-Sarawi, "Reram-based in-memory computing for search engine and neural network applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 388–397, 2019.

[16] H. Aziza, S. Hamdioui, M. Fieback, M. Taouil, M. Moreau, P. Girard, A. Virazel, and K. Coulié, "Multi-level control of resistive ram (rram) using a write termination to achieve 4 bits/cell in high resistance state," *Electronics*, vol. 10, no. 18, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/18/2222

[17] C. Bengel, A. Siemon, F. Cüppers, S. Hoffmann-Eifert, A. Hardtdegen, M. von Witzleben, L. Hellmich, R. Waser, and S. Menzel, "Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using SPICE level compact models," *IEEE TSCAS I*, vol. 67, no. 12, pp. 4618–4630, 2020.

[18] R. Iraji, M. T. Manzuri-Shalmani, A. H. Jamalian, and H. Beigy, "Ija automaton: Expediency and e-optimality properties," in *2006 5th IEEE International Conference on Cognitive Informatics*, vol. 1, 2006, pp. 617–622.

[19] I. Poliakov, V. Khomenko, and A. Yakovlev, "Workcraft – a framework for interpreted graph models," in *Applications and Theory of Petri Nets*, G. Franceschinis and K. Wolf, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 333–342, (available from http://workcraft.org).

[20] L. Rosenblum and A. Yakovlev, "Signal Graphs: From Self-Timed to Timed Ones," in *Int. Workshop on Timed Petri Nets*, 1985, pp. 199–206.