Supplementary information

Cellular automata imbedded memristor-based recirculated logic in-memory computing

Yanming Liu[1, 2, †], *He Tian*[1, 2, †,*], *Fan Wu*[1,2, †], *Anhan Liu*[1,2], *Yihao Li*[3], *Hao Sun*[1,2], *Mario Lanza*[4], *Tian-Ling Ren**[1,2]

[1] School of Integrated Circuits, Tsinghua University, Beijing 100084, China.

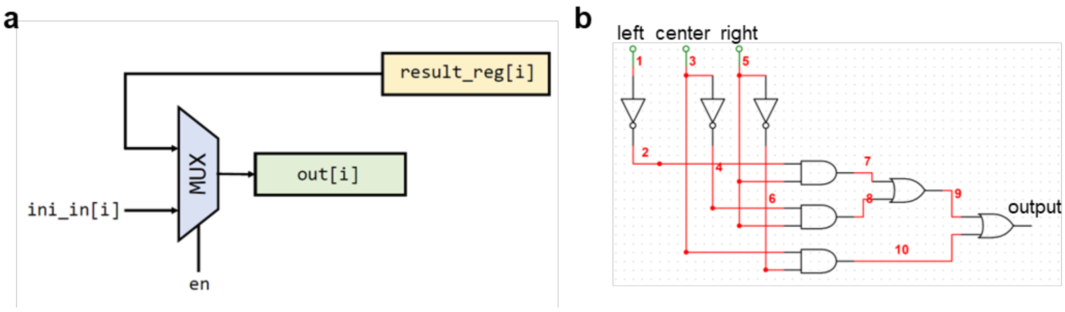
[2] Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China.

[3] Weiyang College, Tsinghua University, Beijing 100084, China.

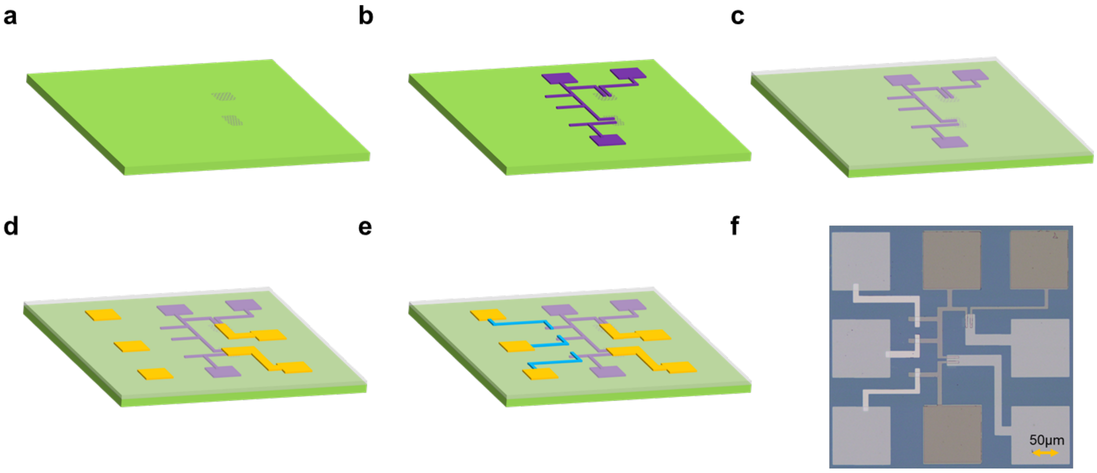
[4] Physical Science and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

†These authors contributed equally to this work.

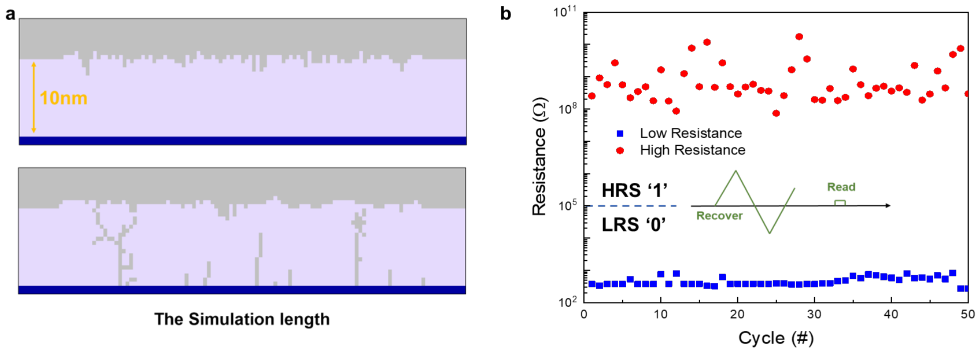
Corresponding to: RenTL@tsinghua.edu.cn, tianhe88@tsinghua.edu.cn



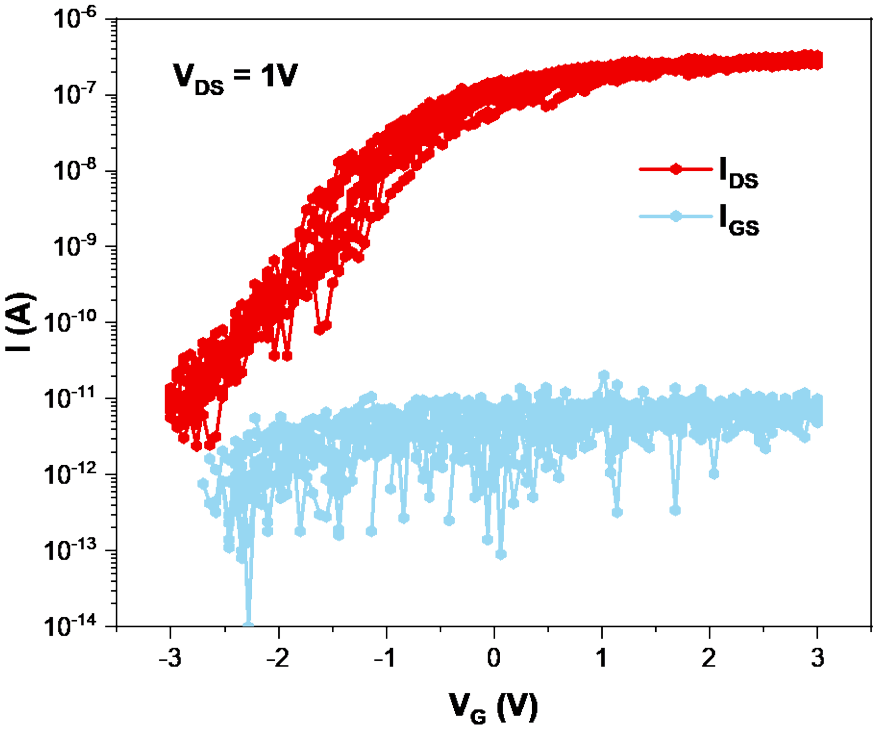
Supplementary Figure 1. Block diagram of the CA cell architecture implemented by FPGA. (a) FPGA implementation scheme of the CA. (b) Rule Switching Logic unit for 1-D elementary CA Rule 110.



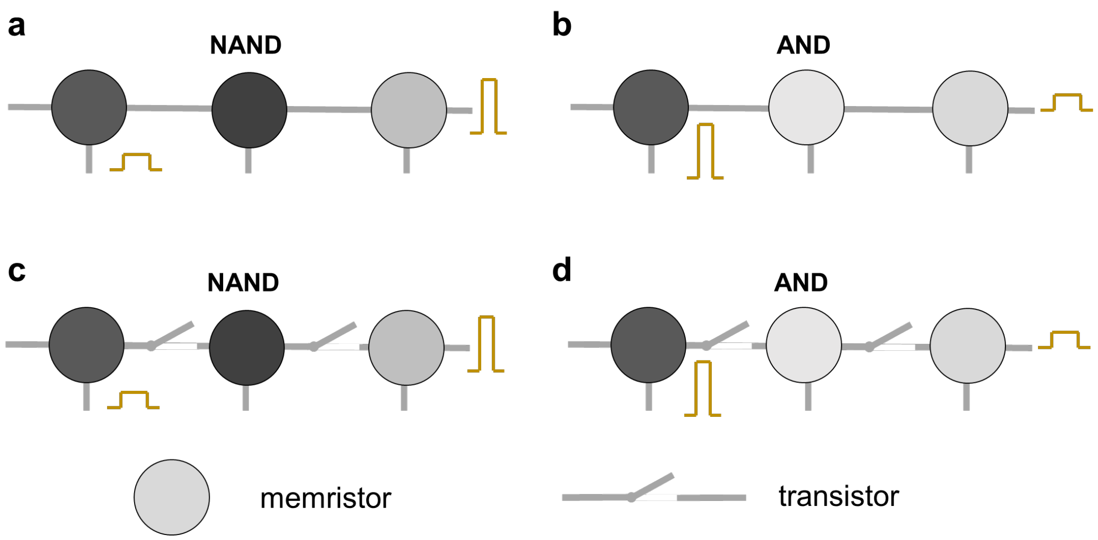
Supplementary Figure 2. RLOS devices manufacturing process. (a) Transfer MoS_2. (b) Using electron-beam lithography (EBL) and electron-beam evaporation (EVP) to fabricate Ti/Pt 2 nm/35 nm thickness electrode. (c) Deposited the 10 nm-thick HfO_2 layer by atomic laye. (d). Carrying EBL and EVP process to fabricate Pt with 40 nm as gate metal of MoS_2 transistor (e) Carrying EBL and EVP process to fabricate Ag with 40 nm as top electrode of memristor. (f) The microscope image for 1 unit structure.



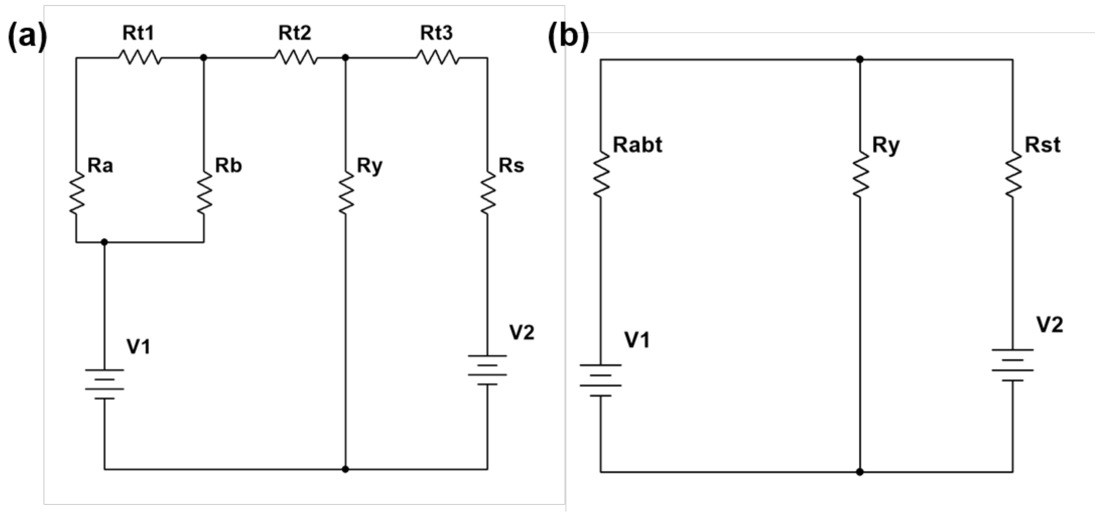
Supplementary Figure 3. Memristor Simulation Model. (a) Particle distribution of memristor. (b) Stability of 50 switching cycles under DC mode. The model can be found in Methods[1].



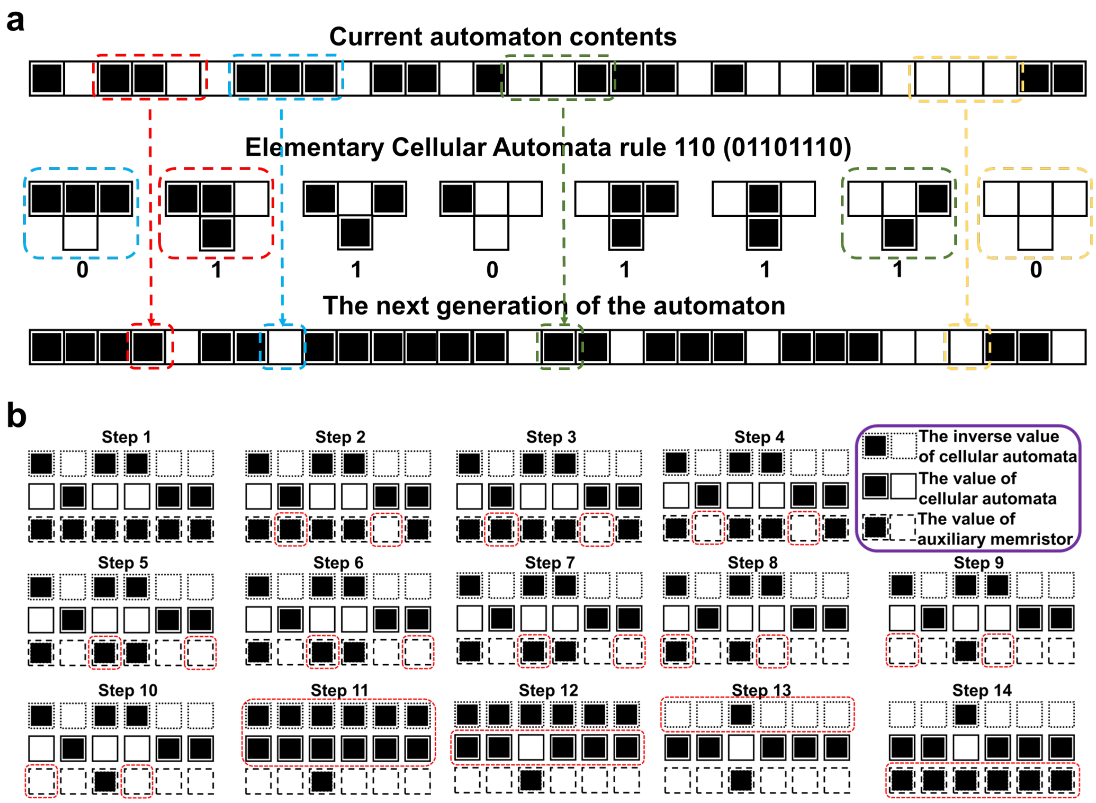
Supplementary Figure 4. The transfer curves of ten typical devices at $V_{DS} = 1$ V.



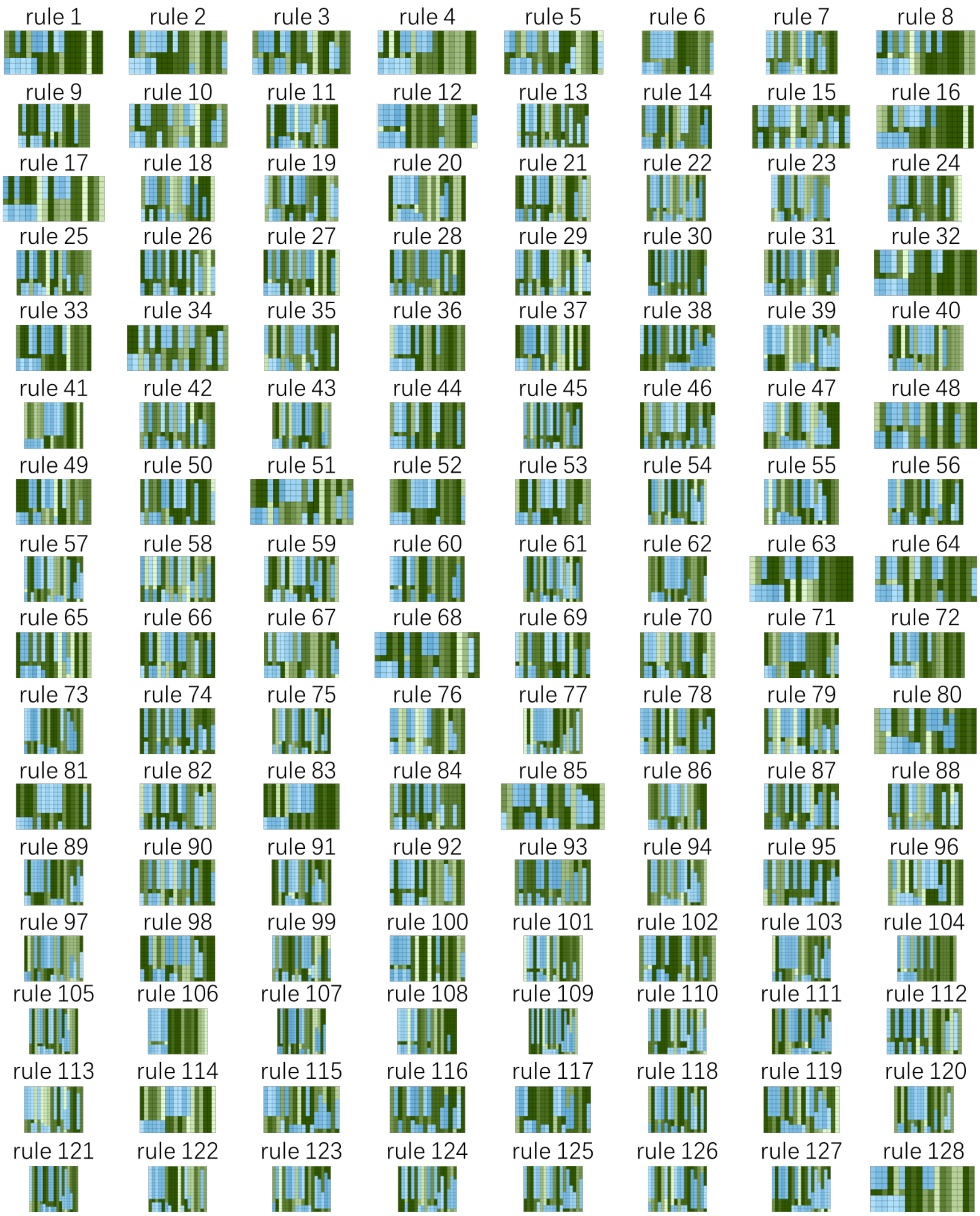
Supplementary Figure 5. The schematic of NAND and AND logic operations. (a) sub-circuit of NAND and (b) AND logic operations executed by memristor[2]. (c) sub-circuit of NAND and (d) AND logic operations executed by memristor, transistor hybrid circuit. The circle presents for memristor and the switch presents for transistor.



Supplementary Figure 6. The simplified circuit of the NAND RRAM-based circuit. The R_a, R_b, R_y are the memristor. R_{t1}, R_{t2}, R_{t3} presents for transistor. R_s presents for R_s. (a) the corresponding circuit for Figure 2c. (b) The equivalent circuit of (a). The calculation of the circuit can be found in Supplementary Note 1.

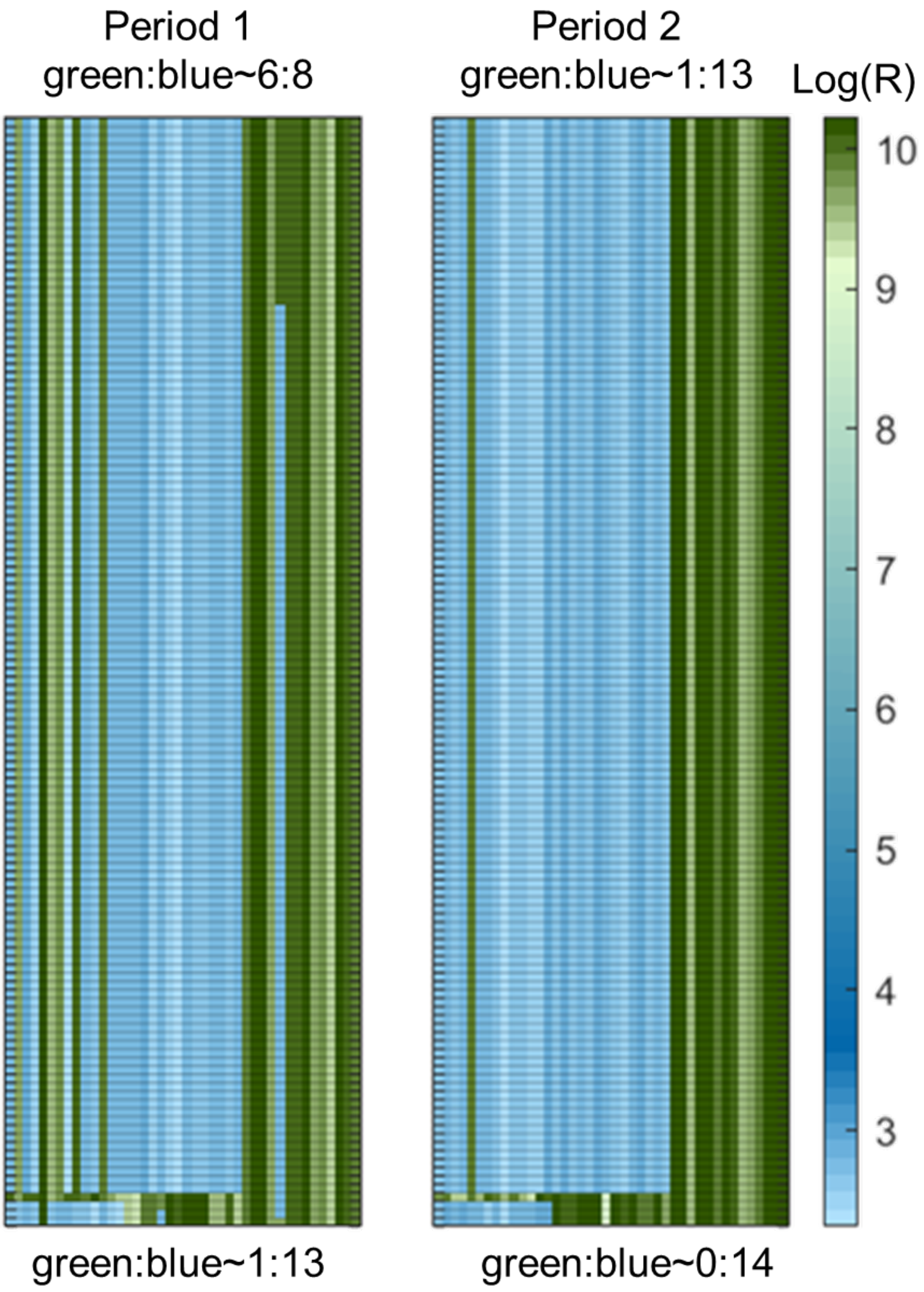


Supplementary Figure 7. Rule Switching Logic unit for Majority classification algorithm based on 1-D CA. (a) the transition rule of elementary cellular automata rule 110. (b) The evolution diagram of the memristor array corresponding the rule 110. The first line presents the inverse value of cellular automata. The second line presents the value of cellular automata. The third line presents the value of auxiliary memristor.

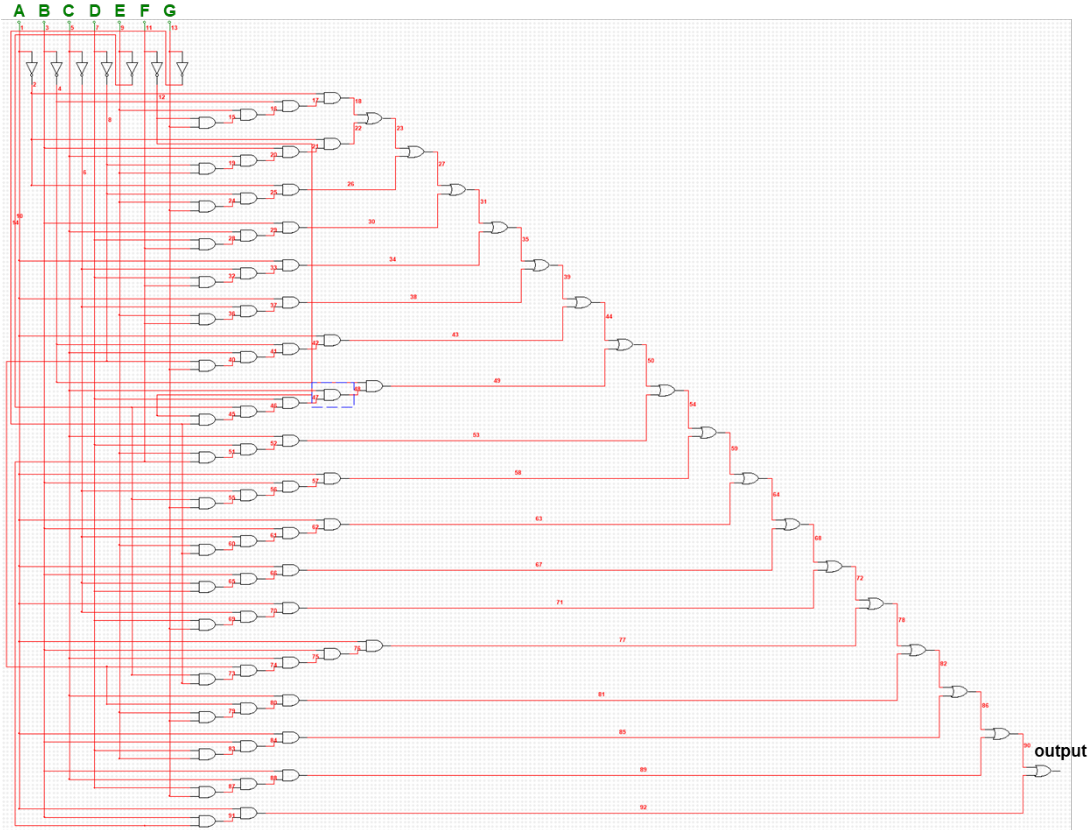


Supplementary Figure 8. Memristors evolution diagram for entire rule of 1D elementary cellular automata (Rule 1-128).

Supplementary Figure 9. Memristors evolution diagram for entire rule of 1D elementary cellular automata (Rule 129-254).



Supplementary Figure 10. Memristors evolution diagram for majority classification algorithm with 14 input. The initial input is 01001110100100. After two periods, all cells have been converted to ‘0’ state, which authenticate the algorithm.



Supplementary Figure 11. Rule Switching Logic unit for Majority classification algorithm based on 1D CA. (Rule: 0504058705000f77037755837bffb77f[3], corresponding logic expression: A’B’EF’G + A’BCD’E + A’D’EG + BCDF + AC’DF + AC’EF + AB’CD’G + B’CDE’F’G’ + CDEF + ABC’E’G + ABC’EG’+ ABC’D + AC’DG + ABCD’E’G’ + CD’EG + ABDE + BCDG + ABF. Components number:86).

Supplementary Table 1. The logic expression of Elementary Cellular automata rules (1-254).

	Logic Expression		Logic Expression
1	AB’C’	2	A’B’C
3	A’B’	4	A’BC’
5	A’C’	6	A’B’C + A’BC’
7	A’B’ + A’C’	8	A’BC
9	A’B’C’ + A’BC	10	A’C
11	A’B’ + A’C	12	A’B

13	A’C’ + A’B	14	A’C + A’B
15	A’	16	AB’C’
17	B’C’	18	A’B’C + AB’C’
19	A’B’ + B’C’	20	A’BC’ + AB’C’
21	A’C’ + B’C’	22	A’B’C + A’BC’ + AB’C’
23	A’B’ + A’C’ + B’C’	24	AB’C’ + A’BC
25	A’BC + B’C’	26	AB’C’ + A’C
27	B’C’ + A’C	28	AB’C’ + A’B
29	B’C’ + A’B	30	AB’C’ + A’C + A’B
31	B’C’ + A’	32	AB’C
33	A’B’C’ + AB’C	34	B’C
35	A’B’ + B’C	36	A’BC’ + AB’C
37	AB’C + A’C’	38	A’BC’ + B’C
39	A’C’ + B’C	40	A’BC + AB’C
41	A’B’C’ + A’BC + AB’C	42	A’C + B’C
43	A’B’ + A’C + B’C	44	AB’C + A’B
45	AB’C + A’C’ + A’B	46	B’C + A’B
47	B’C + A’	48	AB’
49	B’C’ + AB’	50	B’C + AB’
51	B’	52	A’BC’ + AB’
53	A’C’ + AB’	54	A’BC’ + B’C + AB’
55	A’C’ + B’	56	A’BC + AB’
57	A’BC + B’C’ + AB’	58	A’C + AB’
59	A’C + B’	60	A’B + AB’
61	A’C’ + A’B + AB’	62	A’C + A’B + AB’
63	A’ + B’	64	ABC’
65	A’B’C’ + ABC’	66	A’B’C + ABC’
67	ABC’ + A’B’	68	BC’
69	A’C’ + BC’	70	A’B’C + BC’
71	A’B’ + BC’	72	A’BC + ABC’
73	A’B’C’ + A’BC + ABC’	74	ABC’ + A’C
75	ABC’ + A’B’ + A’C	76	A’B + BC’
77	A’C’ + A’B + BC’	78	A’C + BC’

79	BC’ + A’	80	AC’
81	B’C’ + AC’	82	A’B’C + AC’
83	A’B’ + AC’	84	BC’ + AC’
85	C’	86	A’B’C + BC’ + AC’
87	A’B’ + C’	88	A’BC + AC’
89	A’BC + B’C’ + AC’	90	A’C + AC’
91	A’B’ + A’C + AC’	92	A’B + AC’
93	A’B + C’	94	A’C + A’B + AC’
95	A’ + C’	96	AB’C + ABC’
97	A’B’C’ + AB’C + ABC’	98	ABC’ + B’C
99	ABC’ + A’B’ + B’C	100	AB’C + BC’
101	AB’C + A’C’ + BC’	102	B’C + BC’
103	A’B’ + B’C + BC’	104	A’BC + AB’C + ABC’
105	A’B’C’ + A’BC + AB’C + ABC’	106	ABC’ + A’C + B’C
107	ABC’ + A’B’ + A’C + B’C	108	AB’C + A’B + BC’
109	AB’C + A’C’ + A’B + BC’	110	A’C + B’C + BC’
111	B’C + BC’ + A’	112	AB’ + AC’
113	B’C’ + AB’ + AC’	114	B’C + AC’
115	AC’ + B’	116	BC’ + AB’
117	AB’ + C’	118	B’C + BC’ + AB’
119	B’ + C’	120	A’BC + AB’ + AC’
121	A’BC + B’C’ + AB’ + AC’	122	A’C + B’C + AC’
123	A’C + AC’ + B’	124	A’B + BC’ + AB’
125	A’B + AB’ + C’	126	A’C + BC’ + AB’
127	A’ + B’ + C’	128	ABC
129	A’B’C’ + ABC	130	A’B’C + ABC
131	ABC + A’B’	132	A’BC’ + ABC
133	ABC + A’C’	134	A’B’C + A’BC’ + ABC
135	ABC + A’B’ + A’C’	136	BC
137	A’B’C’ + BC	138	A’C + BC
139	A’B’ + BC	140	A’B + BC

141	A’C’ + BC	142	A’C + A’B + BC
143	BC + A’	144	AB’C’ + ABC
145	ABC + B’C’	146	A’B’C + AB’C’ + ABC
147	ABC + A’B’ + B’C’	148	A’BC’ + AB’C’ + ABC
149	ABC + A’C’ + B’C’	150	A’B’C + A’BC’ + AB’C’ + ABC
151	ABC + A’B’ + A’C’ + B’C’	152	AB’C’ + BC
153	B’C’ + BC	154	AB’C’ + A’C + BC
155	A’B’ + B’C’ + BC	156	AB’C’ + A’B + BC
157	A’C’ + B’C’ + BC	158	AB’C’ + A’C + A’B + BC
159	B’C’ + BC + A’	160	AC
161	A’B’C’ + AC	162	B’C + AC
163	A’B’ + AC	164	A’BC’ + AC
165	A’C’ + AC	166	A’BC’ + B’C + AC
167	A’B’ + A’C’ + AC	168	BC + AC
169	A’B’C’ + BC + AC	170	C
171	A’B’ + C	172	A’B + AC
173	A’C’ + A’B + AC	174	A’B + C
175	A’ + C	176	AB’ + AC
177	B’C’ + AC	178	B’C + AB’ + AC
179	AC + B’	180	A’BC’ + AB’ + AC
181	A’C’ + B’C’ + AC	182	A’BC’ + B’C + AB’ + AC
183	A’C’ + AC + B’	184	AB’ + BC
185	B’C’ + AB’ + BC	186	AB’ + C
187	B’ + C	188	A’B + AB’ + BC
189	A’C’ + AB’ + BC	190	A’B + AB’ + C
191	A’ + B’ + C	192	AB
193	A’B’C’ + AB	194	A’B’C + AB
195	A’B’ + AB	196	BC’ + AB
197	A’C’ + AB	198	A’B’C + BC’ + AB
199	A’B’ + A’C’ + AB	200	BC + AB
201	A’B’C’ + BC + AB	202	A’C + AB

203	A’B’ + A’C + AB	204	B
205	A’C’ + B	206	A’C + B
207	A’ + B	208	AC’ + AB
209	B’C’ + AB	210	A’B’C + AC’ + AB
211	A’B’ + B’C’ + AB	212	BC’ + AC’ + AB
213	AB + C’	214	A’B’C + BC’ + AC’ + AB
215	A’B’ + AB + C’	216	AC’ + BC
217	B’C’ + AC’ + BC	218	A’C + AC’ + BC
219	A’B’ + AC’ + BC	220	AC’ + B
221	C’ + B	222	A’C + AC’ + B
223	A’ + C’ + B	224	AC + AB
225	A’B’C’ + AC + AB	226	B’C + AB
227	A’B’ + B’C + AB	228	BC’ + AC
229	A’C’ + BC’ + AC	230	B’C + BC’ + AC
231	A’B’ + BC’ + AC	232	BC + AC + AB
233	A’B’C’ + BC + AC + AB	234	AB + C
235	A’B’ + AB + C	236	AC + B
237	A’C’ + AC + B	238	C + B
239	A’ + C + B	240	A
241	B’C’ + A	242	B’C + A
243	B’ + A	244	BC’ + A
245	C’ + A	246	B’C + BC’ + A
247	B’ + C’ + A	248	BC + A
249	B’C’ + BC + A	250	C + A
251	B’ + C + A	252	B + A
253	C’ + B + A	254	C + B + A

Supplementary Table 2. The logic operation of majority classification based on cellular automata.

	Logic Operation		Logic Operation		Logic Operation
1	S = NAND(A’,B’,E,F,G)	2	S = NAND(A’,B,C,D’,E)	3	S = NAND(A’,D’,E,G)

4	S= NAND(B,C,D,F)	5	S = NAND(A,C’,D,F)	6	S = NAND(A,C’,E,F)
7	S =NAND(A,B’,C,D’,G)	8	S = NAND(B’,C,D,E’,F’,G’)	9	S = NAND(C,D,E,F)
10	S = NAND(A,B,C’,E’,G)	11	S = NAND(A,B,C’,E,G’)	12	S = NAND(A,B,C’,D)
13	S = NAND(A,C’,D,G)	14	S = NAND(A,B,C,D’,E’,G’)	15	S = NAND(C,D’,E,G)
16	S = NAND(A,B,D,E)	17	S = NAND(B,C,D,G)	18	S = NAND(A,B,F)

Supplementary Note 1 - Calculation of the R_C

Using the Kirchhoff Laws can calculate voltages across the R_a, R_b, R_y and R_s.

Define:

$$R_{abt} = R_t + \frac{R_b(R_t + R_a)}{R_b + R_t + R_a}$$

$$R_{st} = R_t + R_s$$

Based on the circuit in S6:

$$V_y = V_2 \frac{\frac{R_y R_{abt}}{R_y + R_{abt}}}{R_{st} + \frac{R_y R_{abt}}{R_y + R_{abt}}} + V_1 \frac{\frac{R_y R_{st}}{R_y + R_{st}}}{R_{abt} + \frac{R_y R_{st}}{R_y + R_{st}}}$$

$$V_{abt} = V_2 \frac{\frac{R_y R_{abt}}{R_y + R_{abt}}}{R_{st} + \frac{R_y R_{abt}}{R_y + R_{abt}}} - V_1 \frac{R_{abt}}{R_{abt} + \frac{R_y R_{st}}{R_y + R_{st}}}$$

$$V_b = V_{abt} \frac{\frac{R_b(R_t + R_a)}{R_b + R_t + R_a}}{R_t + \frac{R_b(R_t + R_a)}{R_b + R_t + R_a}}$$

$$V_a = V_b \frac{R_a}{R_a + R_t}$$

$$V_s = \frac{R_s}{R_{st}} \left(V_1 \frac{\frac{R_y R_{st}}{R_y + R_{st}}}{R_{abt} + \frac{R_y R_{st}}{R_y + R_{st}}} - V_2 \frac{R_{st}}{R_{st} + \frac{R_y R_{abt}}{R_y + R_{abt}}} \right)$$

Then we get all the voltages.

Supplementary Note 2 – Logic operation of edge detection algorithm

The transition rule of edge detection algorithm based on CA can be defined as the following table[4]:

The sum of the nine cell values	9	8	7	6	5	4	3	2	1	0
The next state of center cell	0	1	1	1	0	0	0	0	0	0

The transition rules can be converted to logic formula as:

Y= ABCDEFG’ + ABCDEF’G + ABCDEF’H + ABCDEF’I + ABCDGH’I + ABCDE’FG + ABCDE’FH + ABCDE’FI + ABCDF’GH + ABCDF’HI + ABCEGH’I + ABCEF’GH + ABCEF’HI + ABCFGH’I + ABCD’EFG + ABCD’EFH + ABCD’EFI + ABCE’FGH + ABCE’FHI + ABCF’GHI + ABDEGH’I + ABDEF’GH + ABDEF’HI + ABDFGH’I + ABDE’FGH + ABDE’FHI + ABDF’GHI + ABEFGH’I + ABEF’GHI + ABC’DEFG + ABC’DEFH + ABC’DEFI + ABD’EFGH + ABD’EFHI + ABE’FGHI + ACDEGH’I + ACDEF’GH + ACDEF’HI + ACDFGH’I + ACDE’FGH + ACDE’FHI + ACDF’GHI + ACEFGH’I + ACEF’GHI + ACD’EFGH + ACD’EFHI + ACE’FGHI + ADEFGH’I + ADEF’GHI + ADE’FGHI + AB’CDEFG + AB’CDEFH + AB’CDEFI + AC’DEFGH + AC’DEFHI + AD’EFGHI + BCDEFGI’ + BCDEFH’I + BCDEF’GH + BCDEF’GI + BCDEF’HI + BCDE’FGH + BCDE’FGI + BCDE’FHI + BCDF’GHI + BCEF’GHI + BCD’EFGH + BCD’EFGI + BCD’EFHI + BCE’FGHI + BDEF’GHI + BDE’FGHI + BC’DEFGH + BC’DEFGI + BC’DEFHI + BD’EFGHI + CDEF’GHI + CDE’FGHI + CD’EFGHI + A’BCDEFH + B’CDEFGH + B’CDEFGI + B’CDEFHI + C’DEFGHI.

Therefore, each logic operation in one step has 84 rounds. And each step has 756 operations.

Supplementary References

1. Liu Y, Yang K, Wang X, Tian H, Ren T. Lower Power, Better Uniformity, and Stability CBRAM Enabled by Graphene Nanohole Interface Engineering. *IEEE Transactions on Electron Devices* **67**, 984-988 (2020).
2. Huang P*, et al.* Reconfigurable Nonvolatile Logic Operations in Resistance Switching Crossbar Array for Large-Scale Circuits. *Advanced Materials* **28**, 9758-9764 (2016).
3. Mitchell M, Crutchfield JP, Das R. Evolving cellular automata with genetic algorithms: A review of recent work. In: *Proceedings of the First international conference on evolutionary computation and its applications (EvCA’96)*). Moscow (1996).
4. Itoh M, Chua LO. DIFFERENCE EQUATIONS FOR CELLULAR AUTOMATA. *International Journal of Bifurcation and Chaos* **19**, 805-830 (2009).