**ThreshPush: A Fast, Accurate, Energy-Efficient Programming Algorithm for**

**3-bit-per-cell RRAM Array**

B.Q. Le1,2,\*, A. Levy1,\*, E.R. Hsieh1,3, H.-S.P. Wong1, S. Wong1, S. Mitra1

1Dept. of Electrical Engineering, 5Dept. of Computer Science, Stanford University, Stanford, CA, USA

2Dept. of Electrical Engineering, San Jose State University, San Jose, CA, USA,

3Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan

\*These authors contributed equally to this work. Corresponding author e-mail: binh.q.le@sjsu.edu

**Abstract** ~~–~~ 3-bit-per-cell HfO2-based RRAM memory has been recently demonstrated at the array level for the first time. Prior RRAM programming methods perform sub-optimally for array-level RRAM with 3 bits per cell. Here, we present methods for identifying coarse-to-fine-grained controls on RRAM resistance level for a given RRAM technology process. These controls are found for a 1Mb 1T1R HfO2-based RRAM array, using 130nm CMOS process. With these controls, we propose ThreshPush, a novel algorithm for fast, accurate, energy-efficient programming of 3-bit-per-cell RRAM at the array level. We achieve \_\_\_\_\_\_% improvement over prior method in our apples-to-apples comparison (using equivalent resistance ranges/test patterns).

**Introduction** ~~–~~ Multi-level non-volatile memories are usually based on different voltage thresholds or resistance ranges. For 3-bit-per-cell RRAM memory, cell resistance values are distributed into 8 distinct ranges (states) using *sigma-based allocation (SBA)* [CITE], with corresponding distributions as shown in Fig. 1. Compared to 1-bit-per-cell, programming is more difficult, since the width and the gaps between 8 resistance states are much smaller. The conventional *incremental step pulse programming (ISPP)* method [CITES] (Fig. 2a) is slow and ineffective for 3 bits per cell, because to control 8 distributions accurately, the step size must be small. Another method called *fixed pulse program-verify (FPPV)* algorithm discussed in [1] and [2] (Fig. 2b) can achieve excellent speed, but the accuracy is inadequate for 3-bit-per-cell RRAM programming. In this work, we propose an algorithm that can achieve tuning accuracy sufficient for 3-bit-per-cell RRAM without sacrificing programming speed, using a well-designed combination of 3 programming methods: state-dependent retry method (coarse resistance control, fast speed), incremental compliance current method (moderate resistance control, moderate speed) and fine control method (fine resistance control, slow speed). The algorithm is compared to prior methods—all methods are tested on the same 1Mb 1T1R HfO2-based RRAM array fabricated in a 130nm CMOS process, shown in Fig. 3 [CIT].

**Resistance Control Identification** ~~–~~ The identification procedure for coarse-to-fine RRAM resistance control is presented in Fig. \_\_\_\_. We first pick several representative RRAM cells, and FORM them by incrementing the bitline (BL) and wordline (WL) voltages until the cells transition from pristine state to low-resistance state (LRS). We then RESET all cells to the high-resistance state (HRS) to prepare for a sweep of different write conditions. 200ns SET pulses are performed with BL and WL voltages increasing from 1V-5V in steps of 0.1V. The resistance is measured after each pulse—if the cell is successfully SET (Rf < 100kΩ), the final resistance Rf is recorded and the cell is RESET to HRS (distribution 8) with a single strong pulse. To ensure that the cell is never “over-SET” (i.e. the RRAM can no longer RESET to HRS), we check whether the final resistance has decayed below a pre-defined threshold (Rf < 3.2kΩ). If so, we break out of the loop to continue testing the cell without destroying its resistive switching capability. This method allows us to determine the empirically optimal pulse conditions for producing the desired resistance with minimal variability, as shown in Fig. \_\_\_\_\_. Additionally, the empirical data is used to define the variability when determining the resistance ranges to use with SBA.

**ThreshPush Algorithm Description** ~~–~~ First, all cells are brought to level 7 by performing a blanket RESET operation, then they are sequentially programmed to 7 other levels (0, 1, 2, 3, 4, 5, 6) or they may stay at level 7 (Fig. 3a). The steps can be partially overlapped like programming in NAND flash memory. Alternatively, the starting point can be state 0 by performing a blanket SET at the beginning (Fig. 3b). In this work, single RESET operations consume more energy than single SET operations, so blanket RESET is chosen over blanket SET. To bring cells from state 7 to other target states quickly and accurately, we propose close-to-target detection levels called Rmm[n] and RMM[n], which respectively define upper and lower boundaries for switching from coarse to fine-grained resistance tuning (Fig. 4). Leveraging the resistance tuning controls that we previously developed, we can provide fast, moderate, or slow programming speed with coarse, moderate or fine resistance control, respectively. The coarse resistance control (fast speed) can be realized by using the FPPV method [1], in which if a SET pulse fails to bring the cell resistance to the LRS, it will be RESET, and another SET pulse is applied again (uncontrolled retry). For 3-bit-per-cell, beside state 0, the target can be other states, therefore we propose a slightly different method called *state-dependent retry*, in which the SET retry pulse will be set at different BL and WL voltages as defined by the empirical controls established earlier. There is no concept of state-dependent retry in NAND flash or similar technologies since bit-level erase cannot be used during programming. Figure 6 shows that ICC can be used to control coarsely the target state. The moderate resistance control (moderate speed) can be realized by incremental compliance current control method, in which the compliance current (controlled by WL voltage) is stepped up (maxed out at a predetermined value) for each programming pulse. Data in Fig. 7a and 7b shows that the change in the cell resistance is moderately sensitive to the compliance current. To realize this method, we can step up the bitline voltage (with large steps), instead of wordline voltage. Using this method, if the cell resistance does not reach the target value after a SET pulse, unlike the controlled retry method, we will not RESET and SET again. Instead, we continue with SET pulses with increasing compliance current (or BL voltage with coarse step), until the cell resistance falls into the (Rmm[n], RMM[n]) range. Finally, the fine resistance control (slow speed) can be realized by stepping up with small steps, the BL voltage (for SET) or the SL voltage (for RESET) or pulse width, because based on the data in Fig. 7c, 7d, 7e, 7f, the cell resistance change is less sensitive to the BL voltage, or SL voltage, or pulse width, compared to the compliance current (WL voltage). The complete algorithm is shown in Fig 8.

**Results** ~~–~~ Figure 9, 10, 11, … shows the programming speed improvement of this work’s algorithm over the conventional ISPP algorithm … By starting the program operation from an appropriate initial level and carefully combining coarse, medium and fine program methods with close-to-target verify detection levels, our proposed program algorithm can achieve excellent performance and accuracy for 3-bit-per-cell RRAM memory with optimum energy efficiency.

**Acknowledgements** ~~–~~ This work was supported in part by the DARPA ERI 3DSoC program. E.R. Hsieh is supported in part by the Research of Excellence program MOST107-2633-E-009-003, and NCTU-Stanford Dragon-gate program, MOST 107-2911-I- 009-514, Ministry of Science and Technology, Taiwan. A. Levy is supported in part by the National Science Foundation Graduate Research Fellowship program.

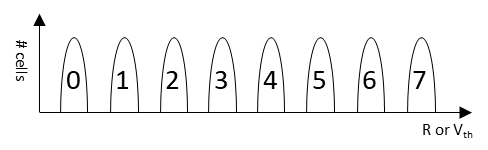
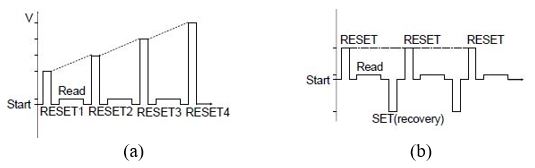


Fig. 1 [Placeholder] Resistance or threshold voltage states for 3-bit-per-cell memory (should include sigma-based allocation markings)



E

D

C

B

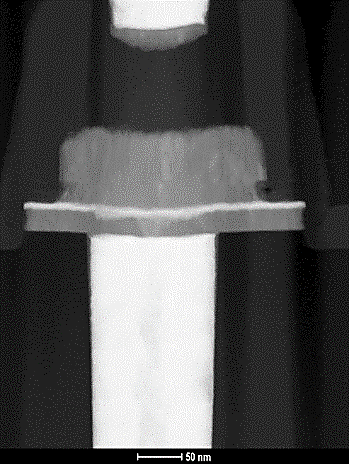
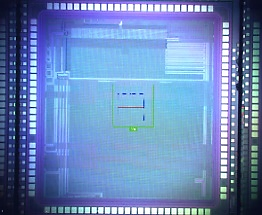
A

# cells

Fig. 2 (a) Conventional ISPP method, (b) Fixed pulse program-verify

method (adapted from [1]).

Fig. 3 (a) Die image (optical) of array, (b) X-sectional TEM of HfO2 RRAM.



(a)

(b)

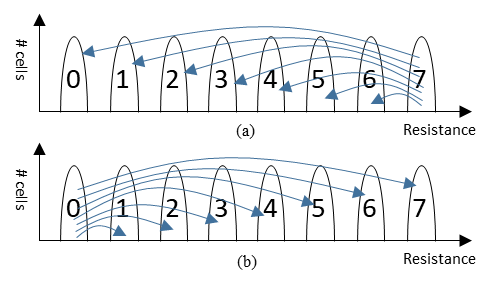


Fig. 3 (a) Starting from H state after a blanket RESET, (b) Starting from A

State after a blanket SET

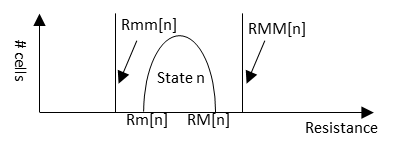


Fig. 4 Proposed detection levels Rmm[n] and RMM[n], n = 0, 1, …, 7.

State 0 doesn’t need to have Rmm[0] and state 7 doesn’t need to

have RMM[7].

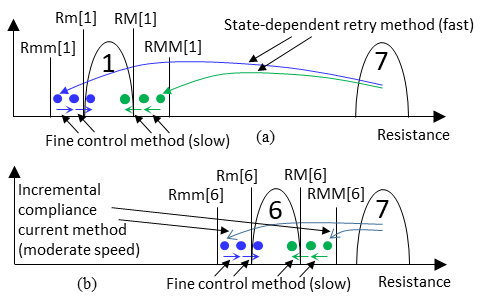


Fig. 5 (a) For far target states like 0, …, 3, use combination of State-dependent

retry method and Fine control method, (b) For near target states like 4,

5, 6, use combination of Incremental compliance current method and

Fine control method

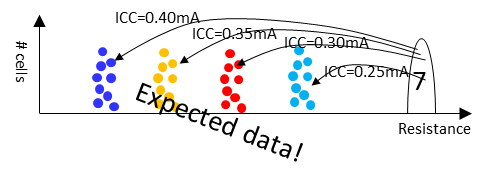


Fig. 6 State-dependent retry.

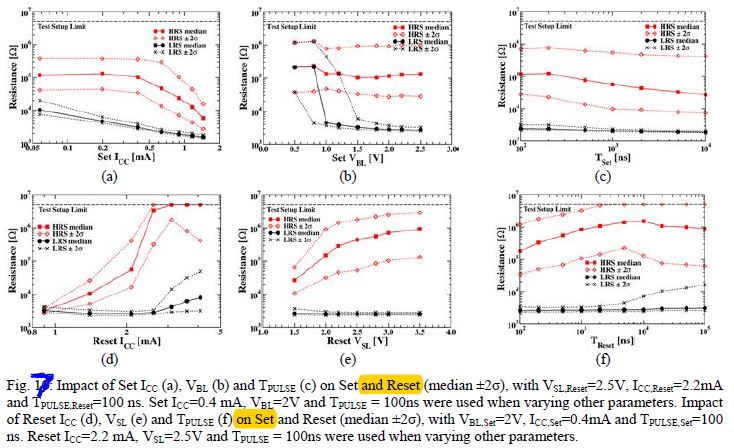
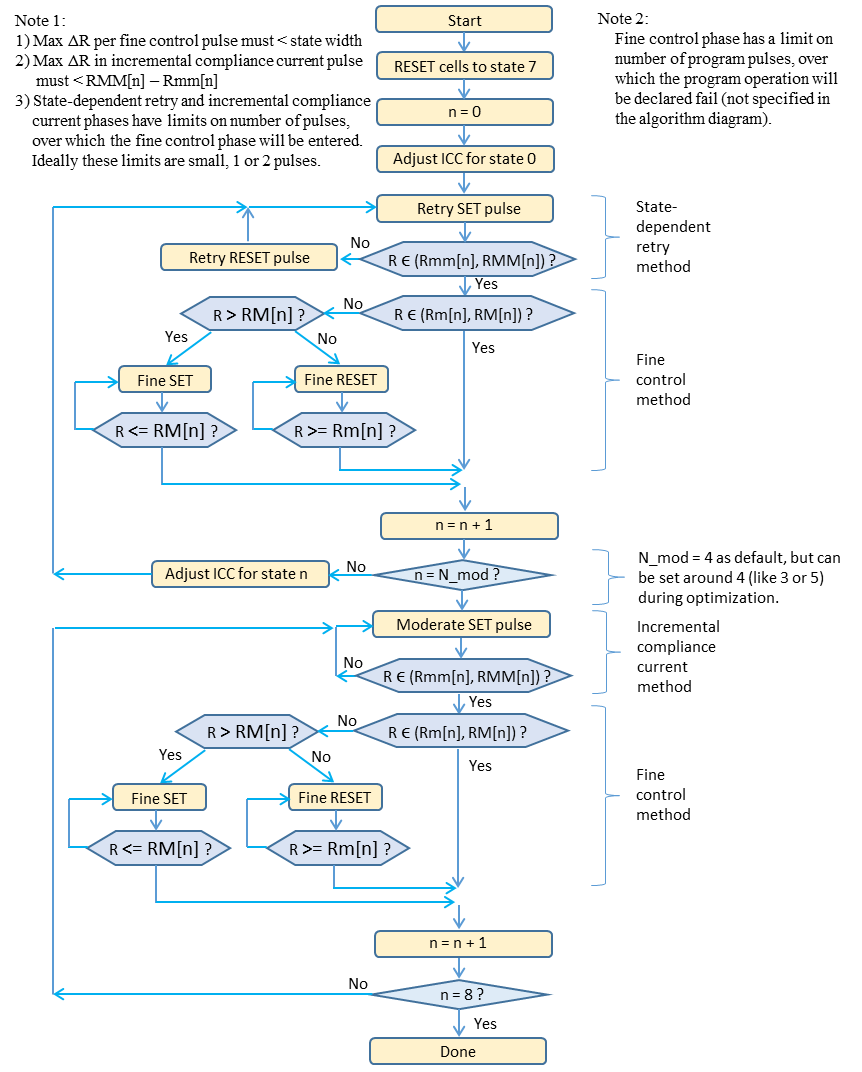


Fig. 7 [Placeholder] Redo this plot with Skywater data



Fig. 12 Final resistance mean vs. standard deviation across various pulse conditions. Red dots display conditions used for coarse resistance control based on the principle of minimizing variability.



(a)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sigma multiple  **m = 1.088** | VBL (V) | Resistance range (n) (Ω) | Rmin(n) (Ω) | Rmax(n) (Ω) | # Programming (Pgrm.) Iterations (Fresh) |
| n = 0 | N/A | N/A | N/A | **3200** | 1 |
| n = 1 | 0.20 | 532 | 3306 | 3838 | 7 |
| n = 2 | 782 | 3991 | 4773 | 7 |
| n = 3 | 1215 | 5012 | 6227 | 12 |
| n = 4 | 2034 | 6640 | 8675 | 13 |
| n = 5 | 3793 | 9498 | 13292 | 9 |
| n = 6 | 13192 | 15329 | 28521 | 8 |
| n = 7 | N/A | **39902** | N/A | N/A |
|  |  |  |  | Mean: | 9 |

(b)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sigma multiple  **m = 1.205** | VBL (V) | Resistance range (n) (Ω ) | Rmin(n) (Ω ) | Rmax(n) (Ω ) | # Pgrm. Iterations (Fresh) | # Prgm. Iterations (4k cycles) |
| n = 0 | N/A | N/A | N/A | **3200** | 1 | 1 |
| n = 1 | 0.20 | 597 | 3306 | 3903 | 6 | 10 |
| n = 2 | 0.22 | 903 | 4045 | 4949 | 6 | 11 |
| n = 3 | 0.23 | 1454 | 5174 | 6627 | 8 | 9 |
| n = 4 | 0.24 | 2538 | 7020 | 9559 | 8 | 15 |
| n = 5 | 0.25 | 4968 | 10341 | 15310 | 9 | 15 |
| n = 6 | 0.28 | 16164 | 17154 | 33318 | 8 | 12 |
| n = 7 | 0.40 | N/A | **39978** | N/A | N/A | N/A |
|  |  |  |  | Mean: | 7 | 11 |

(c)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sigma multiple  **m = 4.18** | VBL (V) | Resistance range (n) (Ω) | Rmin(n) (Ω) | Rmax(n) (Ω) | # Pgrm. Iterations (Fresh) | # Pgrm. Iterations (4k cycles) |
| n = 0 | N/A | N/A | N/A | **3500** | 1 | 1 |
| n = 1 | 0.20 | 3790 | 3763 | 7553 | 3 | 5 |
| n = 2 | 0.25 | 19974 | 8583 | 28556 | 4 | 3 |
| n = 3 | 0.40 | N/A | **39970** | N/A | N/A | N/A |
|  |  |  |  | Mean: | 3 | 3 |

Table I: 3 bits-per-cell allocation based on SBA alone (part a, with VBL = 0.2V) and SBA + BVA combined (part b). Current margin ΔI = 1μA. Resistance Range 6 is slightly modified for lognormal distribution since its median is > 12.9kΩ [7]. (c) 2 bits-per-cell allocation with SBA + BVA with current margin ΔI = 3μA for higher sensing speed and Rmax(0)=3.5kΩ to reduce sensing current. N/A: Not applicable.

Fig. 8 Complete algorithm diagram

**References**

[1] MH Lee et al., ICSICT (2016)

[2] F. M. Puglisi et al., IEEE Electron Device Letters, 2015.

[3] B. Le, A. Grossi et al., IEDM (2018) (expected)

[4] A. Grossi et al., IEDM 2016

[5] SS Sheu et al., VLSI Circuits Symposium, pp. …, 2009.

[6] Stathopolous et al., Nature scientific reports, 2017.

Fig. 7: Resistance range, resistance gap, resistance distribution and resistance window for 3 bits-per-cell RRAM. 7 reference resistors (discussed in Fig. 4) are needed for read operation.