**Fast, Accurate and Energy-Efficient Programming Algorithm for 3-bit per cell RRAM memory**

**Abstract** ~~–~~ HfO2-based RRAM memory with 3-bit per cell at the array level has been demonstrated for the first time recently, but the program performance is not the focus. To have enough sensing margin, all 8 cell resistance distributions need to be controlled accurately, but program methods known so far are inadequate in terms of speed or accuracy. Leveraging fundamental differences between RRAM and other threshold-based non-volatile memories like NAND or NOR flash, we demonstrate a fast, accurate and energy-efficient programming algorithm for 3-bit per cell RRAM at the array level, by combining fast control, moderate control and fine control, which we have identified from the data we took on that array. The algorithm is evaluated based on a 4Kb 1T1R HfO2-based RRAM array, using 130nm silicon CMOS technology.

**Introduction** ~~–~~ Multi-level non-volatile memory are usually based on different cell threshold or resistance values. For 3-bit per cell RRAM memory, cell resistance values are distributed into 8 different ranges (states), forming corresponding distributions shown in Fig. 1. Compared to 1-bit per cell RRAM where there are only 2 resistance distributions, programming is much harder for 3-bit per cell RRAM since the width and the gaps between 8 resistance states are much smaller, and the program operation needs to be able to adjust the resistance of the cells accurately into these states. Conventional incremental step pulse programming (ISPP) method (Fig. 2a) is slow and not effective because to control the 8 distributions accurately, ISPP step size must be small. Another method called the fixed pulse program-verify algorithm discuss in [1] (Fig. 2b) or its modified version discuss in [2] can achieve excellent speed, but the accuracy is inadequate for 3-bit per cell RRAM programming. In this work, we demonstrate a program algorithm that can achieve good accuracy for 3-bit per cell RRAM without sacrificing programming speed by 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). Note that unlike RRAM, for threshold-based memories like NAND and NOR, programming operation has to start from the erase level (distribution 0), and over-program is forbidden since we cannot use erase operation (which is block-based and is very slow) to adjust the over-programmed cells back to the correct levels, and therefore very coarse threshold control cannot be used. We leverage this fundamental difference to improve speed and energy for the program operation by using RRAM very coarse resistance control method (or moderate resistance control method) where over-programmed could occur. This is because in RRAM, correction for over-programed or under-programed is easy, using the fine control method.

**Algorithm Description** ~~–~~ First, all cells are brought to state 7 by doing a blanket RESET operation, then they are sequentially programed to 7 other states (0, 1, 2, 3, 4, 5, 6) or stay at state 7 (Fig. 3a). The steps can be partially overlap like programming in NAND flash memory. If cells were not brought to an initial level, there would be 56 combinations (starting level can be one of the 8 levels, and the target level can be one of 7 other levels, 8x7=56), which can lead to slow and energy inefficient programming. Similarly, the starting point can be state 0 by doing a blanket SET at the beginning, depending on whether SET or RESET operation consume more energy (Fig. 3b). In this work, since RESET takes more energy, a blanket RESET (instead of a blank SET) is chosen, because setting cells to the other 7 levels (using SET operation) takes much more time and pulses than the blanket RESET at the beginning. To bring cells from state 7 to other states quickly and accurately, we propose 2 close-to-target detection levels called Rmm[n] and RMM[n], which are close enough to the borders of a distribution Rm[n] and RM[n], respectively (Fig. 4). The strategy to get both speed and accuracy is to use a coarse resistance control (fast speed) method to bring the cells into the range between Rmm[n] and RMM[n], then use a fine resistance control (slow speed) method to continue to move the cell resistance to the correct range, between Rm[n] and RM[n] (Fig. 5a). Note that for technology like NAND flash, there is no RMM[n] level since over programming is not allowed because there is no bit-level erase to correct over-programming. For closer target states, for example from state 7 to state 4, 5 or 6, a moderate resistance control (moderate speed, no retry) operation can be used in place of the coarse resistance control method before using a fine resistance control method (Fig. 5b). Leveraging intrinsic properties of the RRAM cells, we have identified methods that 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 fixed pulse program-verify method [1], in which if a SET pulse doesn’t 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 the state-dependent retry, in which the SET retry pulse will be set at different compliance current ICC (controlled by wordline voltage), depending on what the target state is. Again, there is no concept as state-dependent retry in NAND flash or similar technologies since block erase (no 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 wordline voltage) is stepped up (and max out at a predetermined value) for each programming pulse. Data in Fig. 7a and 7b shows the change in the cell resistance is moderately sensitive to the compliance current. Note that 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 doesn’t get to 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 …

**Conclusion** ~~–~~ 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. This algorithm can be applied equally well for 2-bit per cell or more than 3-bit per cell RRAM.

**Acknowledgements** ~~–~~ This work was supported in part by ....

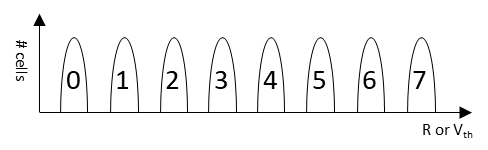
Fig. 1: Resistance or threshold voltage states for 3-bit per cell memory.

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

verify method.



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



Fig. 5 Proposed detection levels Rmm[n] and RMM[n], n = 0, 1, ..., 7.

Range 0 doesn’t need to have Rmm[0] and range 7 doesn’t need to

have RMM[7].



Fig. 4 Starting from range 7 after a blanket RESET.

E

D

C

B

A

# cells



Fig. 6 Use a combination of State-dependent retry method and Fine-control method to achieve good speed and accurate distribution control.

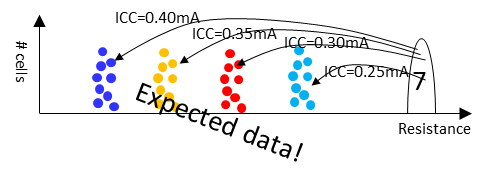




Fig. 8 Algorithm diagram. Note that the Fine-control phase has a limit on

the number of pulses, over which the program operation will be

declared “fail” (not specified in the diagram). Fine SET and Fine

RESET conditions must be optimized such that the resistance

change ΔR per fine-control pulse is smaller than the range width.

**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.