ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.10, No.5, 1–7

Multi-level programming of memristor in nanocrossbar

Xuan Zhu[1,2a)], **Chunqing Wu**[2], **Yuhua Tang**[1,2], **Junjie Wu**[1,2], **and Xun Yi**[1,2]

[1] *State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha, Hunan, China*

[2] *School of computer, National University of Defense Technology, Changsha, Hunan, China*

a) *zhuxuan1986@gmail.com*

Abstract: Utilizing memristor to obtain multi-level memory in nano-crossbar is a promising approach to enhance the memory density. In this paper, we proposed a solution for multi-level programming of memristor in nanocrossbar, which can be implemented on nanocrossbar without the need for extra selective devices. Meanwhile, using a general device model, this solution is demonstrated to be adaptive to a wide range of memristors that have been experimentally fabricated through HSPICE simulation.

Keywords: memristor, multi-level, nanocrossbar, memory, HSPICE

Classification: Storage technology

References

[1] L. O. Chua and S. M. Kang, "Memristive devices and systems," *Proc. IEEE*, vol. 64, no. 2, pp. 209–223, 1976.

[2] H. Kim, M. P. Sah, C. Yang, and L. O. Chua, "Memristor-based multilevel memory," *Proc. 12th. Int. Workshop on Cellular Nanoscale Networks and Their Applications*, pp. 1–6, 2010.

[3] I. E. Ebong and P. Mazumder, "Self-controlled writing and erasing in a memristor crossbar memory," *IEEE Trans. Nanotechnol.*, vol. 10, no. 6, pp. 1454–1463, 2011.

[4] K. H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications," *Nano Lett.*, vol. 12, no. 1, pp. 389–395, 2011.

[5] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Roger, "A memristor device model," *IEEE Electron Device Lett.*, vol. 32, no. 10, pp. 1436–1438, 2011.

[6] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, vol. 3, no. 7, pp. 429–433, 2008.

[7] G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, pp.035204, 2007.

[8] J. Mustafa, "Design and analysis of future memories based on switchable resistive elements," Ph.D. dissertation, Dept. Elect. Eng., RWTH Aachen Univ., Aachen, Germany, 2006.

DOI: 10.1587/elex.10.20130013
Received January 08, 2013
Accepted January 12, 2013
Published March 05, 2013

1 Introduction

Multi-level memory is an effective approach to enhance memory density. As a novel type of nanoelectronic device that has attracted great attention recently, memristor intrinsically owns the multi-level resistance characteristic [1]. Meanwhile, memristors that are integrated in nanocrossbar structure can achieve a unit cell area of as small as $4F^2$(F is the feature size). Combining these advantages together, it is distinct that multi-level memristor integrated in nanocrossbar has a significant potential in achieving ultra-high memory density. Besides, it also shows important application in other fields, such as hardware neural networks.

However, implementing multi-level memristor memory on nanocrossbar faces many challenges. Multi-level programming of memristor is such a problem that need to be properly solved. [2] connects each memristor cell with a selective transistor to achieve selection, which incurs higher cost, and makes it unsuitable for nanocrossbar structure. Self-controlled write of memristor on nanocrossbar implemented by [3] achieved only binary memory and also need the help of additional selective diode for each memiristor cell. [4] is by far the closest experimental demonstration of multi-level memristor memory using nanocrossbar. However, programming method in this work is closely related to the intrinsic characteristic of the device, which might be unadaptive to other types of memristors.

Based on a general device model, we proposed an multi-level programming approach for memristor in nanocrossbar. By generating voltage pulses of multiple width, we achieved precise programming of memristors' resistance with small error distribution. This approach can be implemented on nanocrossbar without the need for extra selective devices, and is suitable for a wide range of memristors that have been experimentally demonstrated.

2 Modeling of nanocrossbar and memristor

The nanowire is simulated using a distributed pi model. According to [7], the resistivity can be approximately expressed by:

$$\frac{\rho}{\rho_0} = 1 + 0.75 \times (1-p)(\frac{\lambda}{d}) \tag{1}$$

where ρ_0, p and λ are constant parameters, and d is the width of nanowire which is estimated to be 10 nm. Thereof, a 4 Ω interconnection resistance between two adjacent cells can be obtained. Following a conservative estimation, we set the interconnection resistance to 10 Ω. The parasitical capacitance of nanowire is set to $2pF \cdot cm^{-1}$.

At present, many circuit models for memristor have been proposed. Here, we adopt the model reported in [5] as the base of our approach, because this model depicts several details of the actual devices' characteristics, and can be applied to a wide range of memristors. In this model, the I-V characteristic of memristor is expressed as follows:

$$I(t) = \begin{cases} a_1 x(t) \sinh(bV(t)), V(t) \geq 0 \\ a_2 x(t) \sinh(bV(t)), V(t) < 0 \end{cases} \tag{2}$$

DOI: 10.1587/elex.10.20130013
Received January 08, 2013
Accepted January 12, 2013
Published March 05, 2013

where $x(t)$ is the state variable which defines the resistive state of the memristor. $x(t)$ is composed of two factors as expressed by:

$$\frac{dx}{dt} = g(V(t))f(x(t)) \tag{3}$$

Here, $g(V(t))$ describes the threshold effect of some of the memristor, which holds the memristor's state unchanged when a voltage smaller than a certain threshold is applied:

$$g(V(\mathrm{t})) = \begin{cases} A_p(e^{V(t)} - e^{V_p}), & V(t) > V_p \\ -A_n(e^{-V(t)} - e^{V_n}), & V(t) < -V_n \\ 0, & -V_n \le V(t) \le V_p \end{cases} \tag{4}$$

$f(x(t))$ reflects the relationship between the changing rate of $x(t)$ and its current state:

$$f(x) = \begin{cases} e^{-\alpha_p(x-x_p)}\frac{1-x}{1-x_p}, & x \ge x_p \\ 1, & x < x_p \end{cases}, \mathrm{when V(t)} > 0 \tag{5}$$

$$f(x) = \begin{cases} e^{\alpha_n(x+x_n-1)}\frac{x}{1-x_n}, & x \le 1-x_n \\ 1, & x > 1-x_n \end{cases}, \mathrm{when V(t)} < 0 \tag{6}$$

Here, V_p,V_n,A_p,A_n,x_p,x_n,α_p,α_n,a_1,a_2,b are all parameters that are used to fit the characteristic curve of specific devices. It is clear that when applying a constant voltage that exceeds the threshold to a memristor, the time needed to program the memristor from an initial state to a target state is numerically solvable, given the fitting parameter set.

3 Multi-level programming circuit

The block diagram of the proposed programming circuit is shown in figure 1. The word line controller(WLC) and the bit line controller(BLC) generate control signals according to the data that is being written, and output to the word line drivers(WLD) and bit line drivers(BLD), respectively. Each word(bit) line is driven by a WLD(BLD) which outputs the driving voltage needed to program or hold a memristor's resistance.

The selecting scheme in this paper is similar to the published $V_{dd}/2$ scheme which makes use of the threshold effect [8]. Specifically, the word line and the bit line pair that is selected is applied by either a write voltage pair of $(V_{wp},-V_{wp})$ or a erase voltage pair of $(-V_{wn},V_{wn})$, according to the process of the programming which will be introduced later. Here, choice of V_{wp} and V_{wn} should meet the following requirements:

$$\begin{cases} V_p/2 < V_{wp} < V_p \\ V_n/2 < V_{wn} < V_n \end{cases} \tag{7}$$

The unselected word lines and bit lines are all grounded. Using such a configuration, only the voltage across the selected memristor will exceed the threshold, and thus only the resistance of the selected memristor will be changed.

DOI: 10.1587/elex.10.20130013
Received January 08, 2013
Accepted January 12, 2013
Published March 05, 2013

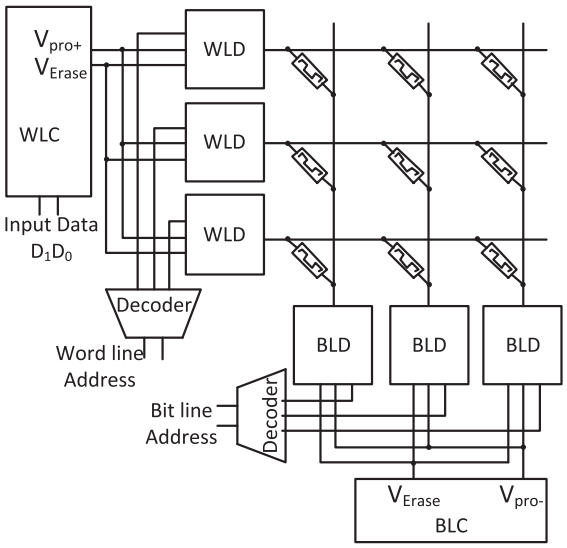

Fig. 1. Block diagram of the proposed circuit.

The programming of a selected memristor consists of two steps: firstly, erase the memristor from its current state to high resistive state, and then write it from high resistive state to the target state. We use such an erase-and-write scheme rather than program it directly from the current state to target state, because such a scheme avoids the need for reading the current state, and thus simplifies the circuit overhead.

During the erase process, a fixed width(T_{Erase}) erasing voltage pulse is applied to the selected word line and bit line. As the decreasing speed of $x(t)$ decreases exponentially when it is below x_n, final state of the erase process will fall into a small range after a long enough process, even if the initial state is distributed in a wide range. During the write process, according to the data that is being written, WLC outputs a voltage signal V_{PRO+} of a previously calculated width to enable the selected WLD. The WLD then applies the driving voltage V_{wp} to the selected word line. Meanwhile, BLC outputs a voltage signal V_{PRO-} to enable the selected BLD to apply a driving voltage of $-V_{wp}$ to the selected bit line. By doing so, resistance of the selected memristor is modified according to the width of the V_{PRO+} signal.

Take 4-level memory as an example. Figure 2 illustrates the circuit design of the WLC, WLD, and BLD. Assume four target states $x_0(0.125)$, $x_1(0.375)$, $x_2(0.625)$, $x_3(0.875)$ are used to represent the two-bit data 00, 01, 10, 11, respectively. The driving voltage width that is needed to write 01, 10, 11 can be obtained by numerical calculation using the above-mentioned model, which is denoted by T_{P1}, T_{P2}, T_{P3}, respectively. The data decoder selects one of the reference voltage generated by the voltage divider according to the input data and outputs to the positive input end of a comparator. A saw-tooth wave voltage is provided to the negative input end of the comparator with an amplitude of V_{dd} and a rising time of T_{P3}. The reference resistors in

DOI: 10.1587/elex.10.20130013
Received January 08, 2013
Accepted January 12, 2013
Published March 05, 2013

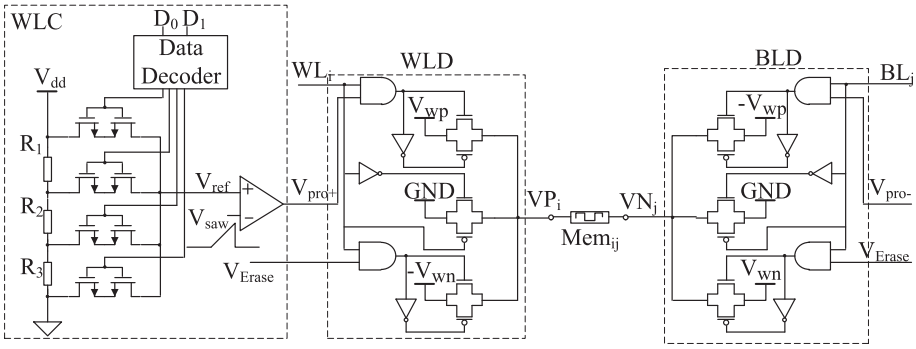

Fig. 2. Details of a 4-level programming circuit.

the divider are set as follows to generate the proper voltage reference:

$$\begin{cases} \dfrac{R_1}{R_1+R_2+R_3} = \dfrac{T_{P1}}{T_{P3}} \\ \dfrac{R_1+R_2}{R_1+R_2+R_3} = \dfrac{T_{P2}}{T_{P3}} \end{cases} \tag{8}$$

Time sequence of all the control signals during the programming process are shown in figure 3, where the specifics of voltage reference, the saw-tooth wave V_{saw} and the output of the WLC V_{PRO+} are provided in the insets.

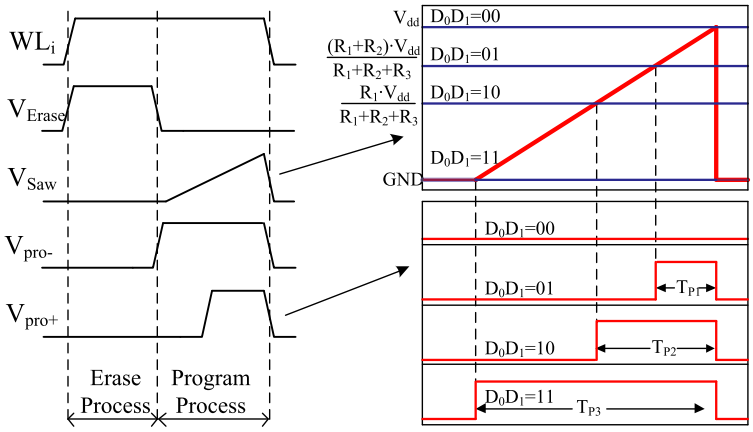

Fig. 3. Sequence of all the control signals.

4 Simulation results

We take the device reported in [6] as an example to carry out the simulation. The parameters are set to:$V_p = 1.2\,V$, $V_n = 0.6\,V$, $A_p = 5$, $A_n = 30$, $x_p = 0.7$, $x_n = 0.8$, $\alpha_p = 4$, $\alpha_n = 24$, $a_1 = 2.3e^{-4}$, $a_2 = 3.8e^{-4}$, $b = 0.04$, according to [5]. It should be noted that as the nanowire width is scaled from 50 nm as reported in [6] to 10 nm, parameter b is correspondingly adjusted to 1/25 of the original value. According to the numerical solving of a single memristor model, the length of T_{Erase}, T_{P1}, T_{P2} and T_{P3} is 35.77 ms, 34.17 ms, 17.54 ms, 8.78 ms, respectively. The divider resistor R_1, R_2, R_3 are correspondingly set to 25.7 $K\Omega$, 25.6 $K\Omega$ and 48.7 $K\Omega$, respectively. However, the numerical

DOI: 10.1587/elex.10.20130013
Received January 08, 2013
Accepted January 12, 2013
Published March 05, 2013

solving omits some of the circuit's error, thus, the actual programming results may vary from the numerical method.

We set up the circuit simulation of a 16×16 nanocrossbar, and consider the memristor cell under the worst case scenario which locates at the furthest corner to the driving circuits. The dash dot lines in figure 4 show the changing of state variable $x(t)$ when programming a memristor from an initial state of x_3 to each of the four target states using numerical method. The dash line show the actual situation obtained by circuit simulation. It can be seen that all the memristor's final states are lower than the desired level. This can be explained that due to the voltage drop on transistors and nanowires, the actual voltage difference applied to the selected memristor decreases as its resistance decreases.

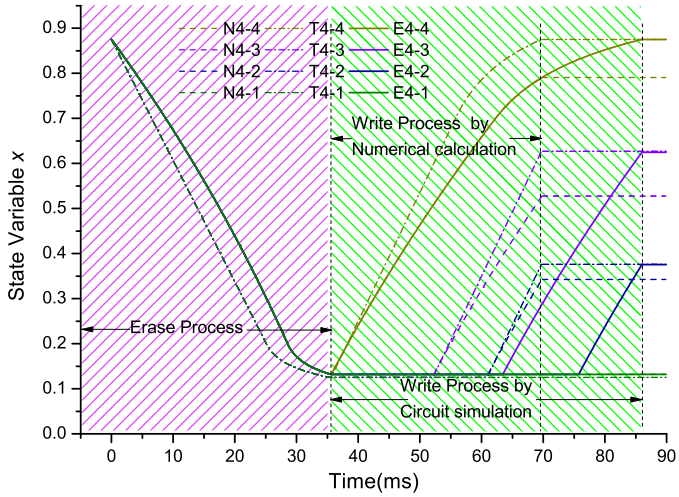

Fig. 4. Changing of the state variable $x(t)$ from an initial state of x_3. Erasing processes of the four different target states overlap with each other.

In order to deal with this error, we could adjust the programming condition empirically by extending the write process. Specifically, we reconfigured the resistance of R_1, R_2 and R_3 to: $20.4\,K\Omega$, $24.4\,K\Omega$, $55.2\,K\Omega$, respectively. Simulation results after the adjustment are shown in figure 4 by the solid lines, which match the final states obtained by numerical simulation closely.

Although empirical adjustment makes the simulation results match with the numerical method at the initial state of x_3, error might exist when the programming starts from the other states. In order to evaluate the error and exclude the possibility that it might be accumulated to disturb the next round programming, we record the distribution of the programming's final states from different initial states, as shown by figure 5. 999 samples of the initial state uniformly distributed from 0.001 to 0.999 are used for simulation, with four target states of x_0, x_1, x_2, x_3. It can be seen that after programming, distribution of the final states can be distinctly distinguished. The minimum noise margin locates between the upper limit of state 00(0.197) and the lower limit of state 01(0.266), which is about 0.069.

DOI: 10.1587/elex.10.20130013
Received January 08, 2013
Accepted January 12, 2013
Published March 05, 2013

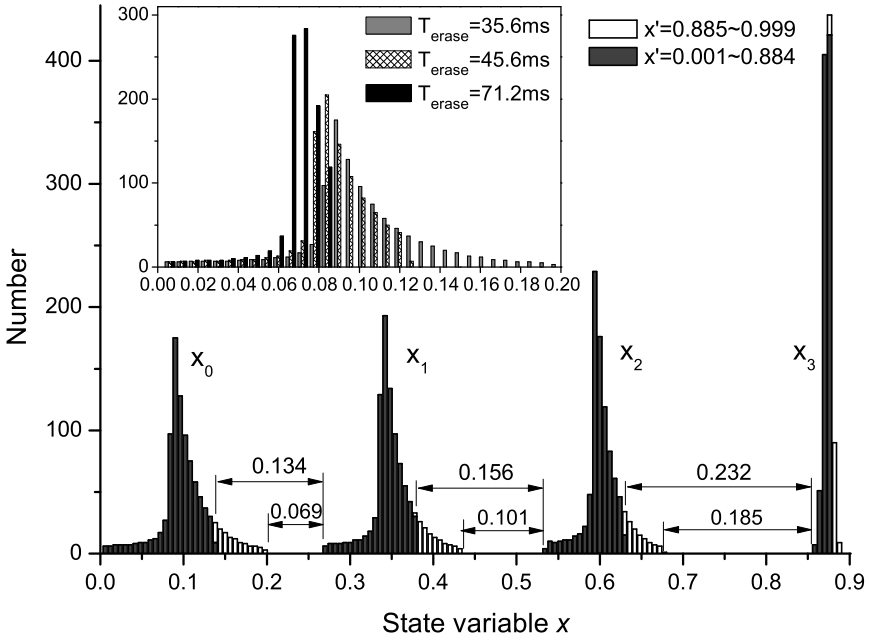

Fig. 5. Distribution of the final states with a initial state normally distributed from 0.001∼0.999.

However, this noise margin could be easily improved. After one programming process, distribution of the final state will be compressed from 0.999 to 0.884. For the second time programming with a smaller range of initial state, distribution of the final state will be more convergent, as shown by figure 5. The noise margin between the upper limit of the state 00(0.132) and the lower limit of the state 01(0.266) will be increased to 0.134. Inset of figure 6 shows the distribution of x_0 after different time length of erasing process. As the erasing process extends, the distribution becomes narrower, which also means the error distribution can be further reduced by increasing the erasing time. Altogether, final state error incurred by circuit's parasitical effect will not effect the precision of the programming.

5 Conclusion

In conclusion, we proposed a circuit for multi-level programming of memristor in nanocrossbar memory. Using a general device model, we verify the circuit's function by HSPICE simulation. The simulation results show that after the programming, memristors with randomized initial states are configured to different discrete states with sufficient noise margin.

Acknowledgments

This work was supported by NSFC project (Grant No. 61003082).

DOI: 10.1587/elex.10.20130013
Received January 08, 2013
Accepted January 12, 2013
Published March 05, 2013