Extended Abstracts of the 1999 International Conference on Solid State Devices and Materials, Tokyo, 1999, pp. 540-541

Comprehensive Study of a New Self-Convergent Programming Scheme for Split Gate Flash Memory

Wei-Zhe Wong, Evans Ching-Song Yang, Amy Hsiu-Fen Chou, Yu-Yuan Yao, Yen-Sen Wang and Charles Ching-Hsiang Hsu

Microelectronic Laboratory, Semiconductor Technology Application Research (STAR) Group,
Department of Electrical Engineering, National Tsing-Hua University, Hsin-Chu 300, Taiwan, R.O.C.
Phone/Fax: +886-5-5715131-4036/+886-5-5718300, E-mail: chhsu@ee.nthu.edu.tw

1. Introduction

In the past years, applying ramp-up voltage to improve the programming speed and reliability of Flash memories had drawn many researchers' attention. The innate constant gate current characteristic during gate voltage ramping in stacked gate Flash memories makes it easy to accurately control the programmed threshold voltages and implicitly alleviate the dielectric stress [1, 2].

However, for split gate Flash memories, the ramp-up control gate voltage operation is no longer helpful to fulfill constant gate current programming, due to the non-stacked gate structure and the large coupling ratio at source side [3]. Instead, a raw ramp-up source voltage programming method is proposed by Brennan et al. [4] to achieve analog storage in split gate Flash memories. Nevertheless, physical mechanisms and detailed operating characteristics had not been investigated yet.

In this paper, a comprehensive understanding of the ramp-up source voltage programming scheme for split gate Flash memories is first delineated.

2. Device Technology

The split gate Flash memory devices (shown in Fig. 1), and the corresponding dummy memory devices are fabricated with 0.35μm CMOS technology. The oxide thickness under the control gate and the floating gate are 32nm and 9nm, respectively. Major devices used in this study have $W/L_{CG}/L_{FG}$ of 0.75/0.5/0.6μm and deep graded source junction.

3. Ramp-Up Source Voltage Programming

For constant voltage programming, the floating gate potential gradually decreases due to electron injection, while the gate current exponentially decreases as shown in Fig. 2. The programming speed is thus tremendously reduced. Constant floating gate potential is thus desired to maintain constant gate current, which used to be achieved by the word-line voltage ramping programming in stacked-gate devices. However, it fails in split gate devices, since the gate current decreases as control gate voltage increases (shown in Fig. 3). To realize constant gate current programming, we turn around to the large coupling ratio source terminal. Both analytical deductions and simulated results [6] prove that constant gate current can be achieved by applying the ramp-up source voltage programming and the gate current is uniquely determined by the ramp rate. The concept is briefly illustrated in Fig. 4. During programming operation, the floating gate potential is decreased by injected electrons (the amount is proportional to the gate current), or it can be increased via ramped-up source voltage coupling. For the potential increment (from source coupling) smaller than potential reduction (induced by the injected electrons), the floating gate potential will be shifted toward negative (path A). On the contrary, the floating gate potential goes toward positive (path B). There exists an exquisite balance point, where the floating gate potential is latched at certain value, so does the gate current. The converged gate current is uniquely determined by the ramp-up rate of source voltage. The programming speed, which is proportional to the gate current, is thus decided by the ramp-up rate of source voltage. In addition, the programmed states are self-converged.

4. Results and Discussions

Flash memory cells in this study are programmed by source-side hot electron injection and erased by poly-to-poly FN ejection. Note that the threshold voltage of split gate Flash memory can not be monitored by sweeping the control gate voltage as that done in stacked gate structure. The floating gate potential is determined by comparing the conduction current with that of the corresponding dummy device under the same bias conditions ($V_{CG}/V_D/V_S/V_B$= 4/2/0/0V).

Two examples shown in Fig. 5 demonstrate the concepts illustrated in Fig. 4. In path A, the higher initial gate current induced by higher starting source voltage leads to faster convergence than in path B, though the extracted gate currents are converged to the same current level due to the same ramp rate. The programming speed is expected linearly dependent on the ramp rate of source voltage as shown in Fig. 6. The relationship between converging speed and the starting source voltage is further clarified in Fig. 7. Notice that more than four programmed states can be distinguished within 20μs. However, higher converging speed arouses more power consumption due to larger drain current induced by higher initial source voltage. To select optimal ramp rate and ramp range is therefore a trade-off between programming speed and power consumption.

In order to illustrate the insignificant impact of process misalignment on convergent properties, Fig. 8 shows convergent behaviors of different devices, and only slight deviation is observed. Figure 9 also shows the convergent behavior of cells with dispersed initial threshold voltages caused by FN erasure.

From above results, the ramp-up source voltage programming scheme has the following advantages: (1) the programmed states can be accurately controlled; (2) insensitive to process misalignment; (3) threshold voltage dispersion caused by erasing operation can be corrected. It implies the suitability for multi-level charge storage. Typical operating conditions for 4-level charge storage are listed in Table I. The program/erase endurance characteristics are shown in Fig. 10, and no degradation of tunnel oxide is observed for all programmed states during 10^4 cycles. The closure phenomenon of erased state indicates possibly trapped electrons in inter-poly dielectric, which can be opened up by increasing the erase voltage.

5. Conclusions

A ramp-up source voltage programming method for split gate Flash memory is comprehensively studied in this paper. The effects of ramp rate, starting voltage and process deviations on the programming speed and convergent behaviors are presented. Experimental results show the self-convergent behavior, and the programming speed is uniquely determined by the ramp-up rate of source voltage. The fulfilled constant gate current facilitates the accurate control of programmed states, and this technique is demonstrated superior for multi-level charge storage applications.

References

[1] D. Esseni, et al., IEEE IEDM, p. 991, 1998.
[2] T. Kawahara, et al., IEEE J. Solid State Circuits, p. 1554, 1995.
[3] S. Kianiam, et al., Proc. Symp. on VLSI Tech., p. 71, 1994.
[4] J. Brennan, et al., IEEE NVSWM, p. 41, 1998.
[5] J. V. Houdt, et al., IEEE Tran. on ED, p. 1150, 1992.
[6] W-Z. Wong, Master Thesis, 1999.

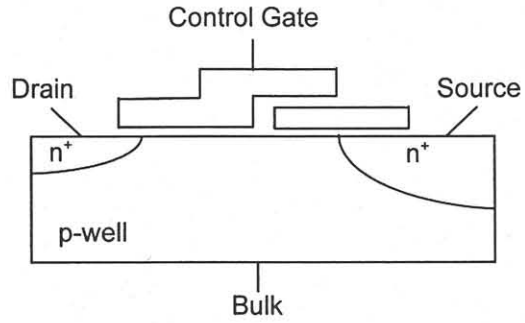

Fig. 1 Cross-section view of split gate Flash cell

Table I. Typical operating conditions of the multi-level programming.

	V_{CG}	V_D	$V_{S.START}$	V_B
ERS "11"	12V	0V	W/O	0V
PGM "10"	2V	0V	3V	0V
PGM "01"	2V	0V	4.5V	0V
PGM "00"	2V	0V	6V	0V

* The ramp-up rate of source voltage is 80mV/μsec and programming time is 50μsec

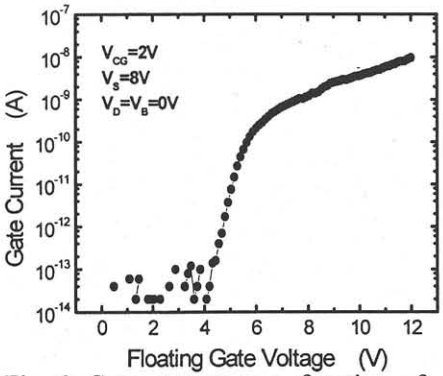

Fig. 2 Gate current as a function of floating gate voltage.

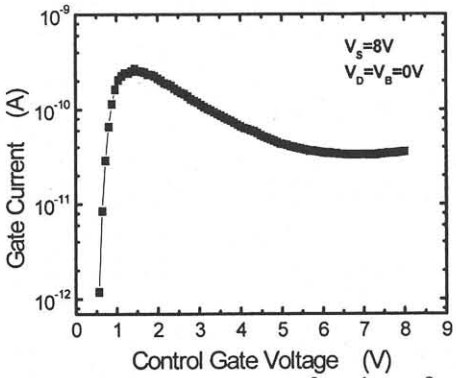

Fig. 3 Gate current as a function of control gate voltage.

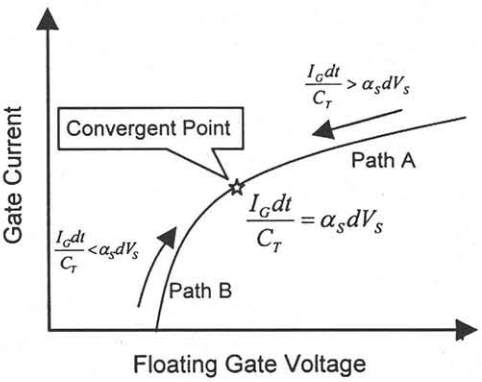

Fig. 4 Concepts of the moving trajectories during ramp-up source voltage programming.

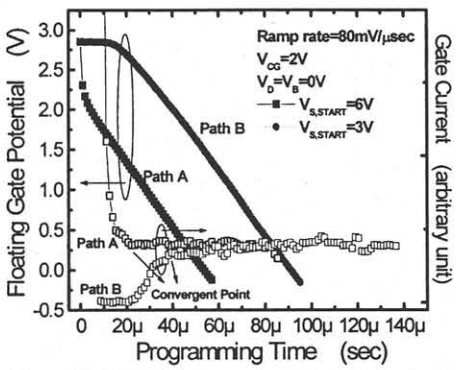

Fig. 5 Typical programming trends and extracted gate current.

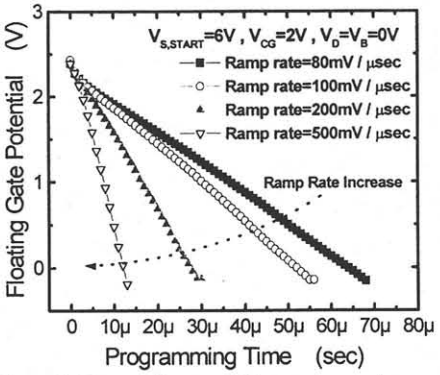

Fig. 6 Dependence of programming speed on ramp rate.

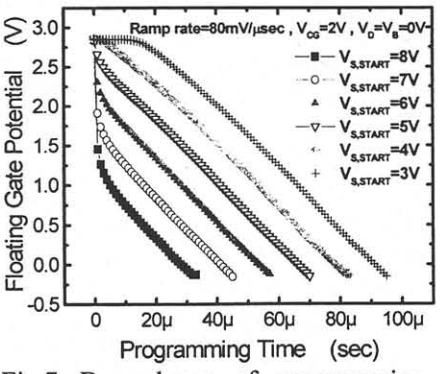

Fig.7 Dependence of programming speed on ramp range.

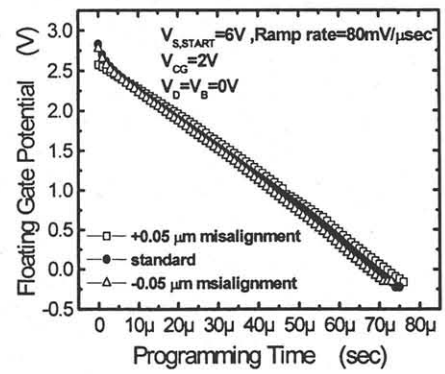

Fig. 8 Sensitivity of process misalignment for ramp-up source voltage programming.

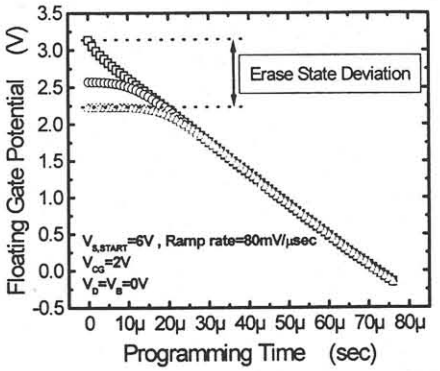

Fig. 9 Convergent behavior of ramp-up source voltage programming.

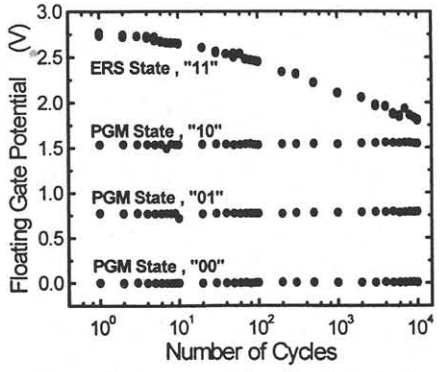

Fig. 10 Endurance characteristics for multi-level programming.