AN98548

Double Programming for High Reliability Systems

Author: Ken Perdue

AN98548 provides additional information on Cypress's approach to mitigating Pseudo Single Bit Charge Loss (pSBCL).

1 Abstract

Cypress ships Floating Gate Flash Memory to many market segments through out the world. Automotive, Aerospace, and other Markets are typically willing to make appropriate changes and investments to improve system reliability.

This document is intended to be used as an internal application note which supports an earlier Cypress Application Note: "Mitigating Pseudo Single Bit Charge Loss". The supplemental information includes real implementation experiences at multiple Automotive Customers, relevant data from Cypress's CCARs Group, Marketing's long term proposed support for our customers, and real customer expectation and questions.

The goal of this application note is to provide the user information that supports the following items:

- Cypress sees pSBCL as a customer application induced failure and not a device related failure.
- The occurrence of Pseudo Signal Bit Charge Loss (pSBCL) is a rare phenomenon resulting in a very low ppm failure rate which typically means only customers with strong quality goals to eliminate re-occurrence of all identified failures will be interested in implementing the recommended corrective action.

2 Pseudo_SBCL Definition and Prevention

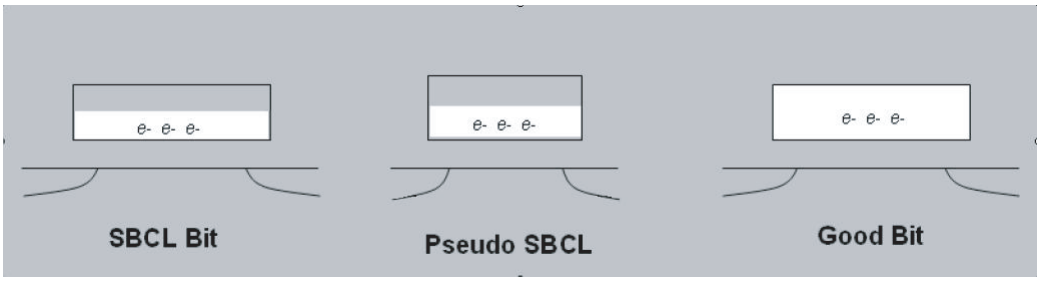
2.1 Defining pSBCL

A Single Bit Charge Loss (SBCL) cell is a defective memory cell which exhibits electron leakage. The electron leakage from the floating gate can be accelerated with voltage or high temperature stress. The subject cell does not have good data retention which is equal to defective bit. The Cypress CCAR has standard processes to characterize and identify defective SBCL cells.

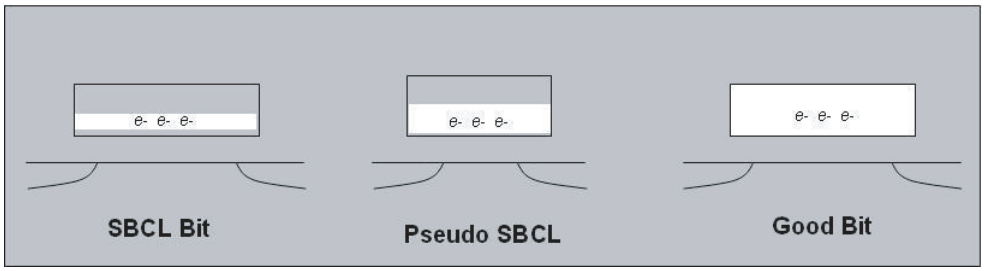
A pSBCL cell is a physically good memory bit which had only been marginally programmed. The cell has no physical anomalies and does not exhibit electron leakage from the floating gate with voltage or high temperature stress. The subject cell has good data retention which is equal to good bit.

The follow characterizes the differences in the behavior and identification of SBCL & pSBCL cells.

- Initial Analysis of true SBCL & pSBCL appear very similar; the subject SBCL & pSBCL are depleted compared to a known good programmed cell.

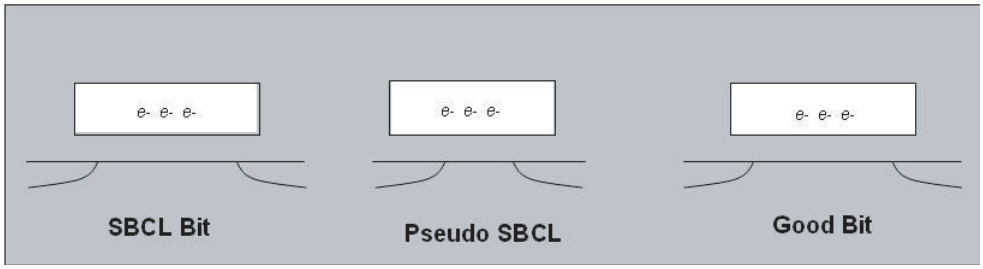


- Note after standard Electrical and Thermal Stress Testing there are differences in the cell charge levels.



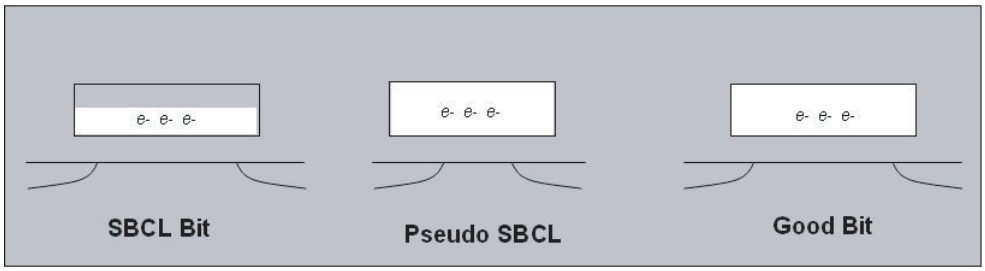
- SBCL Cell, pSBCL Cell and Good Cell all reprogram the same.

.



After standard Electrical and Thermal Stress Testing the true SBCL cell loses charge while a pSBCL & a known good cell have not lost charge.

.



2.2 Cause and Prevention of pSBCL

2.2.1 Background Information: Flash Programming Operation Details

During a Flash programming operation the voltage threshold (Vt) used in Embedded Programming is set at a higher level over the Vt used in Read operation. The Programming Vt is set in this manner to build in additional read margin and guarantee long term data retention. The tighter Program Verify Vt is used to detect any marginally program bits and the Flash State machine will automatically apply the required program pulse(s) to fully charge the subject bit.

2.2.2 Flip Bit Theory

The following highlights the Flip Bit Theory as a possible cause of marginally programmed bit, also known as 'soft-programmed cell'. The Presence of system noise or transients during programming operation can lead to 'under-programming' or uncharged cell. High noise levels could corrupt or limit the programming Vt from reaching the required levels to fully program a cell. When the user performs a “Read Verify” a marginally programmed bit can be read as '0' or a '1'.

In the case where a marginally programmed bit is read as a “0” passing the customer verification testing the units become a probably candidate for a later pSBCL failure.

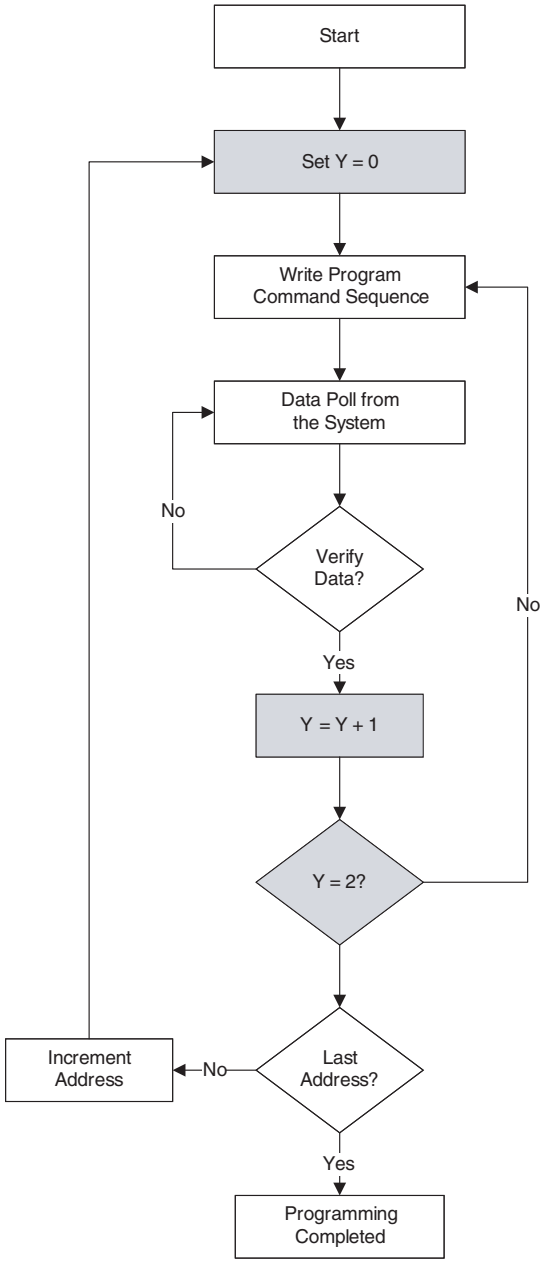
Investigations were made to re-create soft programmed cells in the test lab. Analysis of the Cypress CCAR showed the occurrence of real pSBCL was very low ppm value. Given the rare occurrence of soft programming in real application it was understood the probability to re-create soft programming in the lab would be very low. The lab results were as expected that no soft program cells were created.

2.2.3 Preventing pSBCL Flip Bits

Instituting an additional Program Verify routine immediately after the programming operation can insure the all bits were programmed to the targeted threshold voltage.

The second Program Verify function can be accomplished by modifying the Flash Software drivers to replace typical “Read Verify” after “Programming Completed” with more stringent “Program Verify” as shown in Figure 1. By re-executing the “Write Program Command”, “Program Verify” is leveraged which provides insurance against marginal programmed bits that could lead to a flipped bit or Pseudo-SBCL event. This change results in minimal overhead for the Flash programming process with any marginally programmed bit(s) being detected and programmed automatically. This process ensures quality of programmed bits in a very efficient manner.

Figure 1. Double Programming Algorithm



3 Double Programming implementation

pSBCL is not a device related failure but an application induced failure that is a rare phenomenon resulting in a very low ppm failure rate. Give this low occurrence of pSBCL, there are many customers who are not willing to implementing the recommended corrective action “Double Programming”. The target customers are those who have strong quality goals to eliminate re-occurrence of an identified failure mode.

3.1 Psuedo_SBCL Failure Rates and Corrective Action Effectiveness

At the time of this writing Cypress knows of only two customers in the Automotive Market who have implemented double programming. The first customer has a very demanding application and end customer. The application was a Transmission Control Unit with application temperature requirements ups to 145C. The OEM's end customer place high demands on the OEM to eliminate all identified failures.

Cypress CCAR performed database analysis on the returns from the subject TCU program to understand the effectiveness of implementing “Double Programming” to mitigate pSBCL failures. The CCAR database analysis was partitioned into two segments. Below are the pSBCL failure rates on this particular program prior to implementing “Double Programming” and the analysis showing the new failure rate after implementing “Double Programming”.

- Standard single pass programming: ~6.7ppm (Production: Two Years=> 750K units)
- “Double Programming” Algorithm: 0 ppm (Production Six month => 400K units)

This customer has realized significant reduction in pSBCL failures by implementing Cypress's recommended best practice “Double Programming” corrective actions.

3.2 Typical Customer Expectation and Questions

Each customer and potentially each program at a customer which implements “Double Programming” will have unique requirements. The following is intended to highlight specific tasks that were required prior to or during the implementation of the “Double Programming” algorithm. It is key to clarify for the Customer the differences between pSBCL & SBCL and the new “Double Programming” algorithm. (See *Pseudo_SBCL Definition and Prevention* on page 1.)

Below is a collection of items the customer might request during “Double Programming Implementation.”

1. Double Programming Overhead

   Customer will request Cypress provide information concerning the additional over head associated with Double Programming. There are estimates that the additional over is approximately 20%, obviously the customer must run the new algorithms on their products to obtain a concise answer.

2. Where are Code Updates required

   Once implementation begins the customers program may requires the software algorithms to be updated in any or all of the following areas:

   a. Module Code Updates.

   b. Software Tools used update code.

   c. Flash Programmer Algorithm Updates. Cypress personnel must work with Programmer Mfg to update the subject algorithm(s).

3. Verifying Double Programming Works

   Some customer(s) will require that Cypress provide verification that the new Software Algorithm does fully charge a marginally programmed cell:

Below is one methodology used to generate marginally programmed cells.

1. Identified 'Programmed bits' in each sector of the customer pattern and recorded the BP address locations.
2. Programmed a 'dummy' unit at those BP address locations and read the unit on the Nextest tester.
3. Recorded the Nextest address locations. (Different than BP addresses.)
4. The customer pattern was modified so that the identified bits could be programmed into a 'Weakly' programmed state.
5. Each identified cell was pulsed on the Nextest tester to put the units into a weakly programmed state.

6. The IDS currents at those locations were then recorded for the respective sectors on each device.

3.3 Sample Customer Questions

Question 1

The Cypress applications note “Preventing Pseudo SBCL” shows a flow chart in Figure 1 with added yellow blocks for their proposed containment. The yellow blocks are to add a second “Program Verify”. Why doesn't the first “Program Verify” in the flow catch the weakly charged bit if the threshold is set higher during a “Program Verify”?

Cypress Reply

If there is noise or transient present during the first Program Verify the Program Reference Level can be corrupted or shifted by the subject noise or transient. Under these conditions the subject Cell Charge Level is verified with using an invalid Program Reference Level resulting in a marginally programmed cell.

Reference Section 3.2:

Possible cause for marginal programmed bit, also known as 'soft-programmed cell'

Presence of system noise or transients during programming operation can lead to 'under-programming'

High noise levels could limit the programming voltage from reaching the required levels to strongly program cells

Reading a marginal program bit can be a '0' or a '1'

Follow up to Question 1

Is this is a probabilistic phenomenon?

Cypress Reply

Yes

Question 2

Based on what is being said here; it is theoretically possible that even double programming could result in us having the same issue however the probably is very low (e.g. <<< 1ppm).

Cypress Reply

Yes

Question 3

Is the flow chart in . *Double Programming Algorithm* on page 3 without the yellow blocks the same as the normal programming flow call out in the data sheet?

Cypress Reply

Yes

4 References

Cypress Application Note: “Mitigating Psuedo_SBCL” May 2006

Cypress CCAR Groups: Standard CCAR Flow Process Document

Document History Page

Document Title: AN98548 - Double Programming for High Reliability Systems Document Number: 001-98548				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**	–	–	10/22/2007	Initial version
*A	4978554	MSWI	10/21/2015	Updated in Cypress template
*B	5843240	AESATMP8	08/03/2017	Updated logo and Copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2007-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1s) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.