AN5173
Application note

Notes on EDC after ECC functionality

Introduction

This application note describes what the possible causes of the fault channel 64 of the FCCU (EDC after ECC for FLASH Array) are, and how the application can identify them.

This document applies to SPC58NN84x, but concepts are valid for all microcontrollers belonging to the SPC58 family.

The details described in this document are valid for the ECC logic that protects the integrity of the Flash memories and does not apply to the volatile memories.

Contents

List of tables

List of figures

1 ECC logic behavior

The Code and Data flash uses two different instances of the ECC logic. The algorithm is the same, but the Code flash uses 128bit of data and 17bit of parity; the Data flash 64bit of data and 15bit of parity.

If the ECC logic recognizes a fault[a], the hardware takes different reactions depending on whether the source of the error lies in either the Data or Code flash.

If the error occurs within the Code flash, the hardware:

- asserts the respective flag in the MCR register of the Flash, and
- reports to the MEMU the details of the fault including the address of the faulty location.[b]

On the contrary, if the error occurs within the Data flash the hardware:

- asserts the respective flag in the MCR register of the Flash,
- suppresses the report to the MEMU and the FCCU (refer to section *e2eECC on data Flash accesses* of the Reference Manual for all details on this topic).

a. This event can be a SEC, DEC or TED. *Section 3: Additional details of the ECC logic* describes the behavior in the case of multibit error.

b. Depending on its configuration, the MEMU reports the event to the FCCU. For example, FCCU[28] indicates a uncorrectable error within the flash.

2 EDC after ECC behavior

The FCCU[64] indicates that the *EDC after ECC* has detected a random failure occurring in the ECC logic during a read request to Flash (either Data or Code flash).

Although the above analysis is correct, there is another condition that triggers the FCCU[64]. I.e., if a multi-bit error occurs in the flash array, the EDC after ECC can assert the FCCU[64]. This assertion is a fake EDC after ECC fault because the cause is not a failure of the ECC logic.The reason for this behavior lies in the hardware implementation of the ECC logic and the EDC after ECC mechanism (see next section).

The actual behavior in case of multi-bit error depends on the combination of data and parity bits. The most common behavior is that the hardware indicates an "EDC after ECC event[(c)]" together with the indication of a correction/detection[(d)].

However, there are few cases in which the hardware doesn't signal any fault, or triggers only the "EDC after ECC event."

c. Assertion of the MCR.EEE flag.

d. Assertion of either MCR.SBC, MCR.SBC1 or MCR.EER flag.

3 Additional details of the ECC logic

The Code and Data flash uses two different instances of the ECC logic. All concepts described in this section - however - apply on both of them.

Figure 1 shows the high-level schema of the ECC architecture[e]r:

- the blue block represents the ECC logic, and
- the green blocks represent the EDC after ECC that monitors the integrity of the ECC.

The ECC logic assesses the presence of single, double, or triple bit error in a word, including data (D2) and parity (P2). If it evaluates the presence of one of them, it asserts either the SBC, SBC1 or EER flag.

The "DEC-ECC encoder" is a sub-block of the EDC after ECC. It recreates the parity (P3) starting from D2c. Then the comparators compare the physical Parity and the corrected one. In case of a mismatch, it asserts the EEE event.

In case of no error in the data and parity, and in the ECC logic, the ECC logic and the EDC after ECC don't signal an error.

In case of multi-bit errors, the couple D2/P2 can be either a legal or illegal couple. If it's a legal couple, the hardware behaves as "no error." I.e., no error indication from the ECC logic and EDC after ECC.

If the couple is illegal - in most cases - the ECC logic triggers a wrong correction/detection. For example, it may correct the data and parity in such a way that:

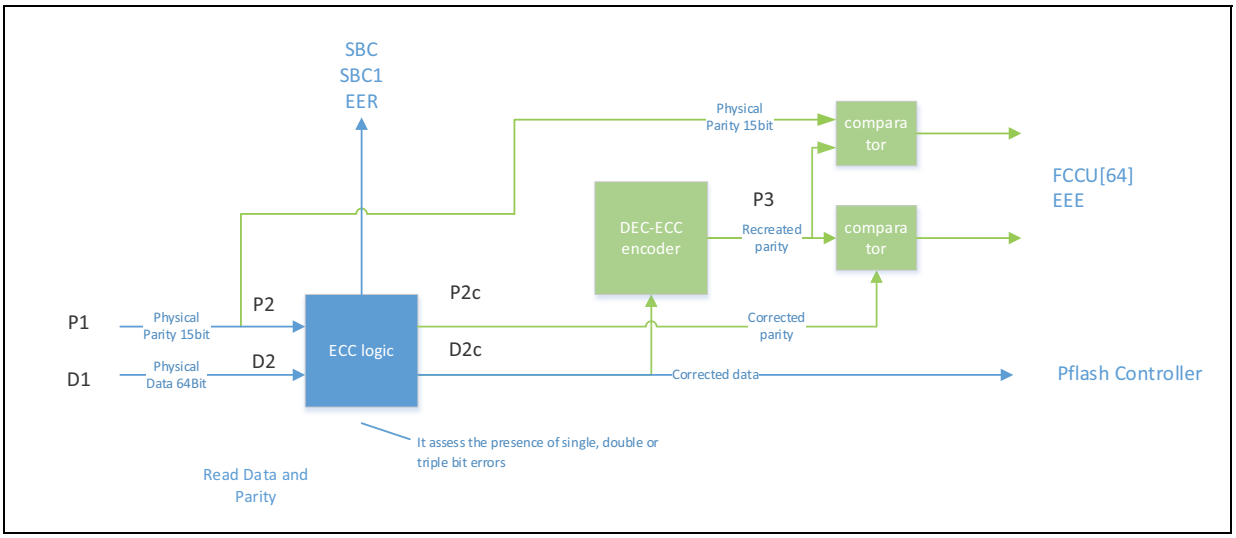
- P2=P2c, and
- D2 <> D2c.

As a consequence, the DEC-ECC encoder recreates the parity starting from D2c, and it gets P3 (the legal parity bits corresponding to D2c). Since P3 is different from P2, the comparator triggers an EEE event.

As a result, there is the assertion of the EEE flag together with one of the SBC, SBC1, or ERR flag.

e. This schema is a simplification of the actual implementation. It helps to identify the conditions of the assertion of the EEE flag in case of no correction/detection.

Figure 1. High-level schema of the ECC logic of the Flash (Dx is the data and Px is the parity)



There are few cases - however - in which ECC logic doesn't detect the presence of a single, double, or triple bit error, even though the couple P2/D2 is not legal. Consequently:

1. P2=P2c, and
2. D2=D2c.

In this case, the DEC-ECC encoder recreates the parity P3 that is different than P2c[f]. As a result, The comparators assert the EEE event, even if there is not the assertion of SBC, SBC1, or EER flag.

In the data flash, there are about 32768 combinations[g] of data and parity bits, around 120[h] of them causes this situation.

It is important to notice that the purpose of the ECC logic is to assess the presence of single, double and triple bit errors. It can't assess the presence of multi-bit errors. Other mechanisms - e.g., Array Integrity check and application level checksum - are in place to detect multi-bit errors.

f. P2c is different respect P3 because the couple P3/D2c is a valid couple and P2C/D2C is not.

g. Around 131072 for the code flash.

h. Around 250 for the code flash.

4 Flash error reporting

This section describes the reporting of the MCR flags in case of different types of error events.

1 bit error

The SBC1 flag indicates this event

2 bit error

The SBC flag indicates this event

3 bit error

The EER flag indicates this event

Error in the ECC logic (wrong correction or wrong detection)

The EEE flag indicates this event when data is legal or containing up to 3 bit errors.

More than 3 bit error (i.e., >=4)

In this case, we can't predict the behavior without knowing the original data and the position of the bit flips. The hardware can react in multiple ways:

1. do nothing (i.e., not signaling of any flag)
2. assert either SBC, SBC1, or EER without EEE
3. assert either SBC, SBC1, or EER with EEE
4. assert EEE only.

It is worth highlighting that by design the purpose of the ECC logic is not to detect multi-bit errors but cope with errors in the data and parity containing maximum 3 bit flips.In this condition, the EDC after ECC monitors the integrity of the ECC logic.

Other mechanisms must detect multi-bit errors or avoid their occurrence.
For example running the Array Integrity Check once after the boot before the safety application starts to protect the Data with an application level checksum

An example of a mechanism that decreases the possibility of multi-bit errors in the array is the multiplexing. The bits of the same word are not one next to each other, but they are spaced of a fixed number of bits.

5 Summary

The hardware asserts the FCCU[64] if:

1. a random failure affects the functionality of the ECC logic, or
2. a multi-bit error[(i)] occurs either within the Code or Data sector.

The application can identify the cause by application level mechanisms that assess the presence of a multi-bit error.

Depending on the cause and location of the error, the application can jump to the safe state, or try recovering to a valid system state.

i. It means more than 3 bit in the same word (i.e starting from 4 bits).

Appendix A Document reference

- *SPC58xNx 32-bit Power Architecture® microcontroller for automotive ASILD applications* (RM0421, DocID028528).

Revision history

Table 1. Document revision history

Date	Revision	Changes
24-May-2018	1	Initial release.