Application Note

DDR Inline ECC Error Injection and Validation Guide for Jacinto™ 7 and Sitara™ Processors

Linjun Meng *Central FAE*

ABSTRACT

DDR Inline ECC is a key reliability and safety mechanism implemented on Jacinto™ 7 and Sitara™ processors. While ECC functionality is implemented in hardware, enabling ECC alone does not demonstrate that memory faults can be detected, reported, and handled correctly. Error injection testing provides a controlled method for validating ECC detection, correction, error logging, and diagnostic reporting functionality.

This application note describes recommended offline and online DDR Inline ECC error-injection methodologies. This document explains validation procedures for single-bit, double-bit, and multi-phase ECC events, and demonstrates how ECC error logs and ESM notifications can be used to verify correct operation of the complete diagnostic chain. The information is intended for software developers, system architects, and functional safety engineers developing automotive and industrial systems based on Jacinto 7 and Sitara processors.

Table of Contents

List of Figures

List of Tables

Trademarks

Jacinto™ and Sitara™ are trademarks of Texas Instruments.

All trademarks are the property of their respective owners.

1 Introduction

DDR Inline ECC provides hardware-based protection against memory corruption by generating and checking error correction codes (ECC) associated with DDR data.

The ECC mechanism is designed to:

- Correct single-bit errors
- Detect double-bit errors
- Improve robustness against DDR failures
- Increase diagnostic coverage for safety-related systems

Although ECC functionality is implemented in hardware, proper system validation requires more than enabling ECC.

To demonstrate correct operation, controlled ECC fault injection is commonly used to verify:

- ECC detection
- ECC correction
- ECC event classification
- Error-log generation
- ESM interrupt generation
- Software fault handling

This application note describes practical ECC error-injection methodologies and validation procedures for both start-up and runtime environments.

2 Detailed Description

2.1 Why ECC Error Injection Testing is Required

From a diagnostic perspective, enabling ECC only configures the protection mechanism. Enabling ECC does not prove that the diagnostic chain operates correctly.

A complete validation must verify:

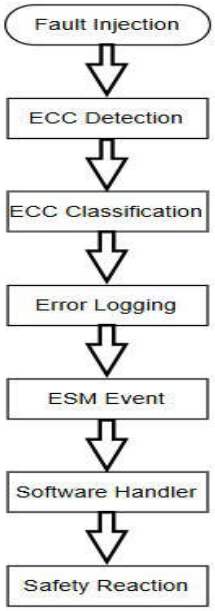

Figure 2-1. Validation Flow

Error injection provides a repeatable mechanism for exercising the entire diagnostic path.

Typical verification objectives include:

- Correctable error detection
- Uncorrectable error detection
- ECC error logging
- ESM event generation
- Software fault handling
- Address-log reporting

2.2 DDR Inline ECC Protection Mechanism

DDR Inline ECC stores ECC syndromes together with DDR data and performs automatic checking during memory reads.

- **SECDED Protection**

The ECC engine provides:

- Single error correction (SEC)
- Double error detection (DED)

For a single-bit corruption event:

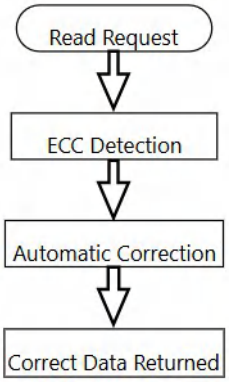

Figure 2-2. Single-Bit Corruption Event

For a double-bit corruption event:

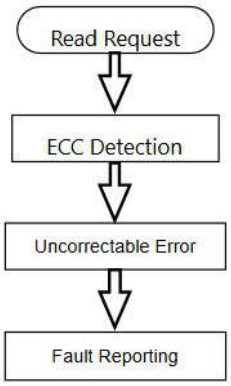

Figure 2-3. Double-Bit Corruption Event

2.2.1 Multi-Phase ECC Protection

DDR accesses occur as burst transactions consisting of multiple phases.

A conventional SECDED implementation protects individual data words. DDR Inline ECC implements Multi-Phase ECC protection that extends coverage beyond individual phases.

This mechanism provides additional protection against:

- Address failures
- Command failures
- Address aliasing
- Burst corruption
- Large-scale memory corruption

As a result, DDR Inline ECC increases diagnostic coverage compared with conventional SECDED-only implementations.

2.3 Offline Error Injection

Offline error injection is typically used in automotive validation flows.

A common sequence is:

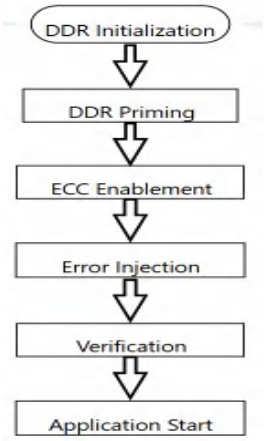

Figure 2-4. Offline Error Injection Flow

Since the application is not yet running, arbitrary DDR locations can be modified safely without impacting software execution.

Typical Procedure

1. Complete DDR initialization.
2. Complete DDR priming.
3. Enable ECC generation.
4. Disable ECC checking.
5. Read data from a target location.
6. Modify one or more bits.
7. Write modified data back.
8. Re-enable ECC checking.
9. Read the modified location.

At this point, the ECC engine detects the injected corruption.

2.4 Online Error Injection

Industrial systems often operate continuously and cannot easily be stopped for start-up-style validation.

Examples include:

- PLCs
- Robotics controllers
- Industrial gateways
- Edge-AI systems

In these systems, online diagnostics are commonly used.

2.4.1 Method 1: ESM Fault Injection

The ESM module can provide fault-injection capabilities.

This method validates:

- ESM infrastructure
- Interrupt routing
- Software fault handling

However, the ESM module does not verify actual DDR ECC functionality.

2.4.2 Method 2: Reserved DDR Validation Region

A dedicated ECC validation region can be reserved within DDR.

Requirements:

- No executable code
- No stack allocation
- No heap allocation
- No application data

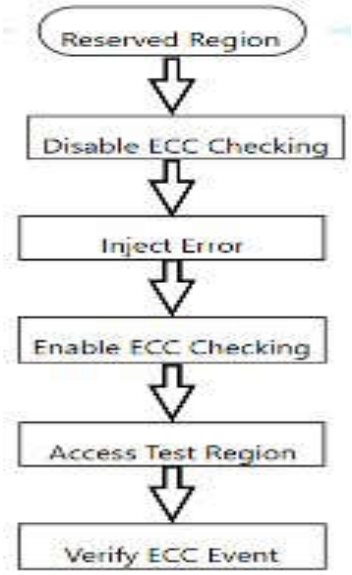

Figure 2-5. Dedicated ECC Validation Region Online Error Injection

This method validates the complete ECC data path while allowing normal application execution.

2.5 Single-Bit Error Validation

Injection Method

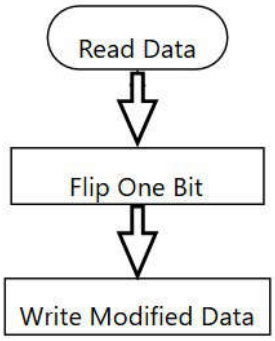

Figure 2-6. Single-Bit Injection Method

Expected Behavior

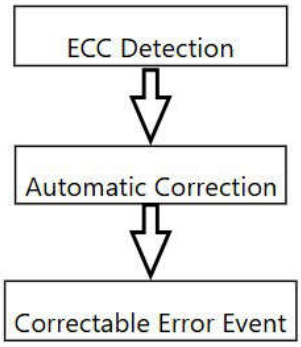

Figure 2-7. Single-Bit Injection Expected Behavior

Expected results:

- Data returned correctly
- Correctable error counter incremented
- ECC address logged
- ESM notification generated

2.6 Double-Bit Error Validation

Injection Method

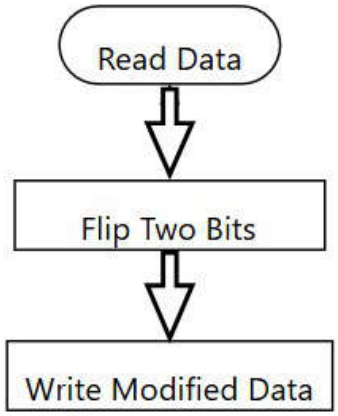

Figure 2-8. Double-Bit Injection Method

Expected Behavior

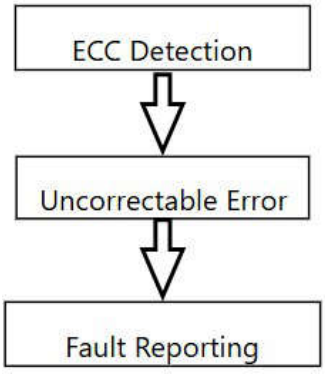

Figure 2-9. Double-Bit Expected Behavior

Expected results:

- UECC event generated
- Error logged
- ESM notification generated
- Software exception handling executed

2.7 Multi-Phase ECC Validation

For safety-related systems, validation must also include Multi-Phase ECC coverage.

A representative validation method injects faults into multiple phases of the same DDR burst.

Example:

Phase 1 → Single-Bit Error

Phase 5 → Single-Bit Error

Expected behavior:

Multi-Phase ECC Detection

↓

Uncorrectable error

This test exercises detection of:

- Address failures
- Command failures
- Burst corruption
- Large-scale memory corruption

2.8 Verification Using ECC Error Logs

After an ECC event occurs, software must verify:

ECC Error Counters

Examples:

- ECC_1B_ERR_CNT
- ECC_2B_ERR_CNT

ECC Error Logs

Examples:

ECC_1B_ERR_ADR_LOG

ECC_1B_ERR_MSK_LOG

ECC_2B_ERR_ADR_LOG

ECC_2B_ERR_MSK_LOG

The logged information must correspond to the injected fault location.

Address Verification

Injected Address == Decoded Error Address

A successful match demonstrates that:

- ECC detection is operational
- Error logging is operational
- Address reporting is operation

2.9 Verification Using ESM

In addition to ECC logs, the system must verify ESM behavior.

Confirm:

- Correct ESM channel triggered
- Correct interrupt generated
- Correct event classification
- Expected software response executed

This provides confirmation that the fault propagates correctly through the diagnostic chain.

2.10 Recommended Validation Strategies

2.10.1 Automotive Systems

Recommended approach:

Offline Validation

Typical sequence:

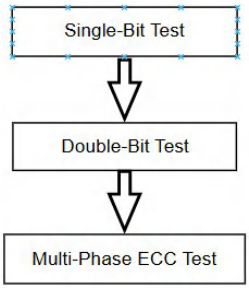

Figure 2-10. Automotive Offline Validation Typical Sequence

Advantages:

- Highest coverage
- Deterministic execution
- Practical for safety validation

2.10.2 Industrial Systems

Recommended approaches:

- Periodic ESM Test
- Reserved DDR Region ECC Test

3 Summary

DDR Inline ECC error injection provides a practical method for validating DDR memory protection mechanisms on Jacinto™ 7 and Sitara™ processors.

By combining controlled fault injection with ECC error-log verification and ESM monitoring, system developers can verify correct operation of ECC detection, correction, Multi-Phase ECC protection, error logging, interrupt routing, and software fault handling. Offline validation is typically preferred for automotive start-up verification, while online validation methods are more practical for continuously operating industrial systems

4 References

1. Texas Instruments, TDA4VE TDA4AL TDA4VL Functional Safety Manual version 0.99, functional safety manual.
2. Texas Instruments, J721S2, TDA4AL, TDA4VL, TDA4VE, and AM68A Technical Reference Manual (TRM), technical reference manual.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025