Automotive Functional Safety Using LBIST and Other Detection Methods

Alessandra Nardi, Software Engineering Group Director, Cadence; Antonino Armato, Principal Product Engineer, Cadence; Francesco Lertora, Sr. Principal Solutions Engineer, Cadence

Functional safety requirements for safety-critical applications are addressed with the insertion of safety mechanisms to detect and/or correct potential failures: their effectiveness is measured by diagnostic coverage (DC). Built-in-self-test, or BIST, originally developed for manufacturing test, can be used as a detection mechanism for functional safety. However, it requires original values to be restored and execution time to fit within the required diagnostic time. Here, we introduce how functional safety analysis can be used to achieve a careful tradeoff of several metrics for efficient use of safety mechanisms. Online/offline logic BIST (LBIST) solutions on single- and multicore architectures based on a vision-processing core are analyzed in the context of functional safety and compared to other safety mechanisms to address permanent and transient faults.

Contents

Introduction ........................................ 1
Functional Safety Metrics and Requirements .................................... 2
Functional safety ........................................ 2
Safety mechanisms ........................................ 3
FMEDA and safety requirements ........................................ 3
BIST ....................................................... 5
Overview of basic architectures ........................................ 5
Test and diagnostic coverage ........................................ 6
LBIST Architecture/Configuration and Safety Mechanisms Tradeoffs .... 7
LBIST applied to a single core ........................................ 7
Multicores—LBIST top-level application .................................... 8
Multicores—local application on the configuration and status registers .... 8
Discussion .............................................. 9
Conclusions ............................................. 11
References ............................................ 12

Introduction

In the last decade, the amount of electronic content in passenger cars continues to grow rapidly bringing the automobile from a simple means of transportation to a mobile electronic hub. Recent reports indicate that electronics now contribute about 40% of the total costs of a traditional, internal combustion engine car, and this jumps as high as 75% for the growing number of electric and hybrid-electric vehicles. Moreover, the automobile manufacturers continue to add new advanced safety features, such as collision avoidance, lane change assistance, and automatic parking, so the amount of electronics in the automobile continues to increase. The requirements for functional safety in automotive are becoming mandatory given the increasing system complexity handling safety-critical functions. The safety requirements are driven by the ISO 26262 standard, in which “functional safety” is defined as the absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electrical/Electronic systems. The malfunction is classified into two types of failures (see Figure 1):

- **Systematic failures**: Representing the failures in an item/function that are induced in a deterministic way during development, manufacturing or maintenance (process issues)

- **Random failures**: Appearing during the lifetime of a hardware element and emanating from random defects innate to the process or usage conditions and are further classified in permanent faults (e.g., stuck-at faults) and transient faults (e.g., single-event-upsets or soft errors)
Hardware safety mechanisms have been proposed to detect permanent and transient faults. BIST methods, originally developed for manufacturing testing, can be used in the context of functional safety to detect permanent faults on safety-critical and automotive systems. BIST, in fact, can benefit from automatic insertion and have good diagnostic coverage.

This white paper describes the architectures and tradeoffs to deploy LBIST as a safety mechanism driven by functional safety analysis, and, mainly based on DC, performance test time and area overhead. Advantages and limitations are discussed with respect to other detection options (e.g., redundancy or parity). Solutions to address specific safety metrics goals, by using LBIST as a safety mechanism, are investigated on a single and multiple core architecture based on a vision DSP processor.

This paper also provides a short review of a few functional safety basics, including comments on the test case and on how FMEDA drives safety decisions. Basics of BIST is shown, including BIST architectures.

Functional Safety Metrics and Requirements

Functional safety

Functional safety analysis is used to evaluate the safety level achieved by the product (e.g., an IP or an SoC). It comprises quantitative evaluations such as failure mode effect and diagnostic analysis (FMEDA), timing analysis, and qualitative assessments, such as dependent failure analysis (DFA).

FMEDA is a structured approach to define failure modes, failure rate, and diagnostic capabilities of a hardware component. Each row of an FMEDA table is organized in parts/subparts/failure mode to capture the component functionality. For each defined failure mode (FM) and affecting safety goals, basic needed inputs for the analysis include:

- **Failure rate (FR):** Rate at which the component experiences faults; i.e., the reliability
- **Safety mechanism (SM) and diagnostic coverage (DC):** Presence of an SM and its effectiveness at detecting faults

The outputs of the analysis are the hardware architectural metrics:

- **Single-point fault metric (SPFM):** Reflects the robustness of a function to the single-point faults either by design or by coverage from safety procedures
- **Latent fault metric (LFM):** Reflects the robustness of a function against latent faults either by design, fault coverage via safety procedures, or by the driver’s recognition of a fault’s existence before the violation of a safety objective
- **Probabilistic metric of hardware failures (PMHF):** Provides rationale that the residual risk of a safety goal violation due to random hardware failures is sufficiently low.

In an intuitive way, a single-point fault can lead directly to the violation of a safety goal, while a latent fault is an undetected fault that allows another fault to cause a hazard.

These three metrics are essentially the measurement of functional safety for hardware components: they capture how likely the component is to fail, how reliable the safety mechanisms are at detecting the failures, and how/whether they bring the system to a safe state.

For a given functionality, a hazard and risk analysis determine the automotive safety integrity level (ASIL); i.e., the level of risk reduction needed to achieve a tolerable risk.
Safety mechanisms

According to the ISO 26262-Part 1 definition, a safety mechanism is a technical solution implemented by E/E functions or elements, or by other technologies, to detect faults or control failures to achieve or maintain a safe state. Examples of safety mechanisms include the following, each with different characteristics in terms of safety metrics, area and performance overhead, detection time, etc.:

- Error correction code (ECC)
- Cyclic redundancy check (CRC)
- Hardware redundancy
- Built-in-self-test (BIST)

The safety mechanisms are introduced during the design and verification of the hardware/software system to handle random failures by making the architecture capable of detecting and/or correcting malfunctions.

The effectiveness of the solution to detect these random failures is measured by the SPFM, LFM and PMHF metrics calculated in the FMEDA. Based on these metrics, the FMEDA drives the design efforts to meet the functional safety targets. The selection of the best safety mechanism for a specific building block needs careful analysis of the tradeoffs between effectiveness and cost, such as power consumption, area, safety metrics, and timing performance.

FMEDA and safety requirements

In this section, we report a sample FMEDA analysis applied to a vision-processing core. The safety analysis is performed to evaluate the safety targets and generate the safety requirements that guide the SM selection.

Tensilica DSP for vision and imaging

The test case we used is a Cadence® Tensilica® Vision DSP for general-purpose imaging with integrated multiply-accumulate (MAC) that increases the computational performance of neural network applications. One of the applications of the DSP is automotive image processing, and in this paper, we refer to it as simply the Vision DSP. Figure 2 shows the basic blocks of the DSP and lists some of its features.

![Figure 2: Tensilica Vision DSP](https://www.cadence.com)

FMEDA

The FMEDA is performed on the Vision DSP to assess the overall hardware metrics and whether additional or better SMs are needed to meet the safety requirements. The FMEDA also guides the insertion of LBIST (or another SM) to reach the ASIL target.
Table 1 reports a simplified FMEDA where some examples of failure modes are taken from the Vision DSP. This sample FMEDA shows how different failure modes have a different weight on the final metrics, and therefore, how the FMEDA can be used to guide architectural decisions on where to insert SMs and which ones can be more effective. Note that this example is focused only on the configuration/status registers, register file and data RAM of the Vision DSP; status registers are used to report information related to exceptions or internal safety mechanisms like ECC/parity.

<table>
<thead>
<tr>
<th>Instance Target</th>
<th>Failure Modes</th>
<th>Failure Rate [FIT]</th>
<th>Safety Mechanisms</th>
<th>DC</th>
<th>SPFM</th>
<th>LFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Register SET1</td>
<td>Fault in core config. registers related to loops</td>
<td>2.90E-03 [*]</td>
<td>LBIST [periodic]</td>
<td>90%</td>
<td>90%</td>
<td>100%</td>
</tr>
<tr>
<td>Configuration Register SET2</td>
<td>Fault in core config. registers related to ALU</td>
<td>2.90E-03 [*]</td>
<td>LBIST [periodic]</td>
<td>90%</td>
<td>90%</td>
<td>100%</td>
</tr>
<tr>
<td>Configuration Register SET3</td>
<td>Fault in core config. registers related to processor configuration</td>
<td>2.90E-03 [*]</td>
<td>LBIST [periodic]</td>
<td>90%</td>
<td>90%</td>
<td>100%</td>
</tr>
<tr>
<td>Status Registers (e.g. ECC Error Status Register GSERR)</td>
<td>Fault in core status registers related to exception/error logging or processor context information</td>
<td>3.10E-03 [*]</td>
<td>LBIST [start-up]</td>
<td>97%</td>
<td>–</td>
<td>97%</td>
</tr>
<tr>
<td>Register file</td>
<td>Wrong data due a fault in the register file</td>
<td>2.10E-02 [*]</td>
<td>Software Test</td>
<td>98%</td>
<td>98%</td>
<td>100%</td>
</tr>
<tr>
<td>Data RAM</td>
<td>Wrong data due a fault in memory cells or memory interface</td>
<td>9.20E-01 [*]</td>
<td>ECC</td>
<td>95%</td>
<td>95%</td>
<td>91.5%</td>
</tr>
<tr>
<td><strong>TOTAL</strong> (SPFM, LFM)</td>
<td>****</td>
<td>****</td>
<td>****</td>
<td>****</td>
<td>****</td>
<td>****</td>
</tr>
</tbody>
</table>

[*] These values expressed in ‘failures in time’ (FIT) or number of failures that can be expected in 10^9 (one billion) device-hours of operation, are not real or associated to any given technology; the only purpose here is to show how a given failure rate is associated to each failure mode.

Table 1: Simplified FMEDA example for permanent faults focused on status, configuration registers, register file, and data RAM

Safety requirements

The Vision DSP FMEDA identifies the gather/scatter ECC error status register (GSERR) as a significant contributor to the safety requirements. When ECC is configured for any of the processor’s data-RAM, data-cache, instruction-RAM, or instruction-cache interfaces, the “corrected error” output is available to signal when the processor hardware has corrected a memory error. This output pin is asserted when the first instruction commits after the processor hardware corrects a correctable memory error. Figure 3 shows that when an error is detected by the ECC, an exception is generated from the microprocessor and the ECC error detection is logged on the GSERR register.

Figure 3: ECC alarm signal on the GSERR register
The GSERR register has a crucial role to signal detection and correction by the ECC safety mechanisms and faults in the register itself can degrade the DC and affect both SPFM and LFM.

It is important to highlight how the FMEDA drives the safety requirements generation; in fact, the status register does not directly violate the safety goal and therefore it contributes only to LFM. This is an important safety requirement: it has an impact on the selection of the correct SM and the timing constraints. In the case of LBIST, it also helps to determine the level of insertion; i.e., a global SM on the whole core, or a local SM at the register level. In both solutions, it is also needed to evaluate the context of the LBIST application, whether it is a single- or a multiple-core configuration.

In general, starting from the FMEDA in Table 1, for the configuration and status registers the following high-level safety requirements are generated:

- Configuration registers storing safety-related information shall be protected against permanent (single point and multiple points) and transient faults; the registers related to secondary information, such as a context, can be excluded from protection
- Status registers storing safety-related information, especially those related to safety mechanisms alarms, shall be protected against latent and transient faults

Detailed safety requirements can be defined in addition to the high-level requirements, specifying the safety mechanisms to be used to protect from permanent faults and related timing constraints:

- Configuration registers shall be protected by logic BIST and the test shall be periodic, to be scheduled each detection time interval (DTI)
- Status registers shall be also tested at start-up by logic BIST

As already discussed, the second safety requirement is less challenging than the first one in terms of performance impact, since the LBIST is supposed to work only at start-up. The next sections analyze in more detail the DC related to safety mechanisms and review the benefits and drawbacks of different options based on the insertion level.

**BIST**

**Overview of basic architectures**

LBIST is a design for testability (DFT) technique in which a portion of a circuit on a chip, board, or system is used to test the digital logic itself. This typically requires additional circuitry and functionality (DFT) be incorporated into the design of the circuit to support self-testing. LBIST (see Figure 4) is composed of a pseudo-random test pattern generator (PRPG) to feed the circuit under test (CUT). The output responses are compacted and shifted into a signature register (multi-input shift register, or MISR) and compared (COMP) to an embedded golden stored signature. Specific BIST timing control signals, including scan-enabled signals and clocks, are generated by the logic BIST controller for coordinating the PRPG, CUT, and MISR. Figure 4 shows a direct access LBIST architecture where the test can be triggered simply by a “GO” input signal and providing a pass/fail indication once the operation is completed. The automatic LBIST generation can manage also a JTAG LBIST insertion where the test functionalities are fully wrapped by a JTAG controller supporting specific JTAG commands and the signatures are read and compared by the test data out (TDO) signal.
Depending on when the test is performed, the BIST techniques are categorized into online and offline, with different impact on the safety metrics.

- **Online BIST** is performed when the functional circuitry is in its normal operational mode; when performed periodically within each DTI, it contributes to the single point fault metric (SPFM).

- **Offline BIST** is performed when the functional circuitry is not in its normal operating mode; e.g., during power-on reset at the engine startup and it addresses latent faults only (LFM).

LBIST offers a general test methodology for any IC typically delivering high-fault coverages and is also supported for automatic insertion. However, typically LBIST **destroys the internal state** of the device under test (DUT), which needs to be restored before resuming normal system operation; i.e., a reboot is required. Therefore, it can be applied only when and if the normal functionality is not required for a given timeframe.

To reduce test time, LBIST can be used to detect faults on smaller parts of the design, often identified with the support of the FMEDA and divided across time slices which occur during allotted functional operational time frames. Following the pseudo-random patterns application, proper isolation is required to avoid that the logic under test corrupts the outer part of the design.

To further improve the test coverage and reducing the LBIST runtime, the test generation can include a configurable test points insertion aimed to improve the CUT controllability and observability.

Some of the parameters that make this solution highly customizable to manage the test time and circuit area overhead are the LBIST compression ratio, the scan chain number, and length, the number of test patterns, the configurable number of test points to be added.

Refer to the “**Unified Compression and LBIST in a Physically Aware Environment**” Cadence white paper (see the References section) as a starting point to be aligned with the latest innovations introduced in automatic LBIST generation. In particular, the physical-aware design for test and the Elastic 2D compression architecture are technologies able to achieve very high compression ratios without degradation of the test coverage and are able to reduce the LBIST channels length by 50% with a related reduction in LBIST runtime.

**Test and diagnostic coverage**

LBIST was originally developed for manufacturing fault detection, having a given test coverage (TC) as a result. It is worth noting that BIST TC is different from BIST DC—though correlated—as defined in a functional safety context. In fact, TC is the percentage of detected over detectable faults, when the circuit logic is under a specific and well-known state (e.g., test mode); this is a deterministic value that does not consider the functional interpretation of the faults.
related to the DC. The DC represents the effectiveness of an SM (working to detect, mitigate, or tolerate faults on an item component that, in general, is not in a well-known state since it is implementing its intended functionality) to detect dangerous faults; i.e., faults that affect safety-critical functionality. DC can be determined as the ratio between the failure rate of detected dangerous faults and the failure rate of total dangerous failures. In many practical cases, TC and DC are correlated and we use them interchangeably in the rest of the paper.

**LBIST Architecture/Configuration and Safety Mechanisms Tradeoffs**

This section provides an overview of different BIST applications and how they affect the safety metrics and the overall system availability, including a comparison with other safety mechanisms.

**LBIST applied to a single core**

LBIST uses scan test techniques and can be deployed on the complete Vision DSP (see Figure 5) to achieve a high DC. However, test time can be significant and limit the use of offline testing, such as during start-up. In this context, LBIST provides high coverage for LFM.

![Figure 5: LBIST applied to a single core](Image)

Other system-level approaches are also important: LBIST can be applied offline at start up or shut down to distinguish between transient and permanent faults. In fact, after fault detection, the core can be put into a safe state and LBIST can be used to investigate the origin of the fault. In the case of LBIST detection, the fault can be classified as a permanent or transient fault, and the core can be reverted to normal use. This approach can be fundamental in an automotive application to meet safety requirements related to the functional continuity of the system in reduced mode.

Further improvement of this approach can be obtained by performing LBIST only on a circuit subset. Figure 6 shows how the safety mechanisms can be used to provide DC for a specific part of the core, then a supervisor can discriminate the origin of the alarm and select only the LBIST patterns to cover the part that is suspected to be faulty. This technique reduces the test time, improving the overall core start-up. The FMEDA can be used to derive the correct mapping between the safety mechanism and the corresponding logic to be covered.
Multicores—LBIST top-level application

As previously noted, LBIST as a safety mechanism can claim two major relevant benefits: high coverage and automatic insertion. However, the test time can limit its applicability to offline testing only. When considering multicore systems, test time issues can be overcome and online LBIST can be used to provide coverage for SFPM as well. Figure 7 shows a multicore solution where LBIST is used for cores diagnostic: during the processing, one of two cores is stopped and tested by LBIST, while the second core can continue processing data. It must be noted that even if the LBIST test time can be reasonably short and meet the DTI requirement, the overall system boot time (likely in the order of many milliseconds) needs to be evaluated and can be relevant to determine the practical applicability of this architecture.

Multicores—local application on the configuration and status registers

A strategy to extend the applicability of LBIST is to focus its deployment only on a specific part of the core logic. The FMEDA is used to identify the failure modes requiring good DC not achievable by traditional safety mechanisms (Table 1). As an example, consider applicability to status/configuration registers identified by the FMEDA as critical to be protected and potential other safety mechanisms that we compare to LBIST for effectiveness (e.g., area overhead, test coverage, test time).
An online diagnostic measure often used on configuration registers is, for example, the “Read Back Periodic by Software of Configuration Registers”. This software safety mechanism is typically implemented by executing a periodical check of the configuration registers of each used core configuration register with its expected value that was previously stored in RAM and adequately updated after each configuration change. It mainly addresses transient faults affecting the configuration registers, while its DC is limited for permanent faults (Figure 8). The defined values that the configuration registers can assume, reduce the possibility to fully stress the register’s fan-in logic cones. Figure 8 shows two patterns in red that can achieve high DC for the logic cones; however, these patterns don’t represent a correct configuration of the circuit and therefore cannot be used. In green is an example of a unique pattern that stimulates the logic cones since it corresponds to a unique register configuration. Additionally, status registers that are mainly controlled by hardware logic are not addressed by this SM.

An alternative and popular safety mechanism for online testing is the triple modular redundancy (TMR). In this case, the logic (memory cell) sensitive to single-event-upsets is tripled and voters are placed at the outputs to identify the correct value. This technique covers only SPFM permanent and transient for the sequential elements that are triplicated. This SM does not raise an alarm when a fault is detected and therefore it does not cover latent faults. It also does not cover the sequential elements fan-in logic: this aspect can be addressed by triplicating the logic as well (called global triple modular redundancy, or GTMR), but that can clearly increase the area overhead significantly.

Another common approach to cover status and configuration registers is the parity. With this safety mechanism, every register is extended by one parity bit (or two, in a double parity bit), which completes each word to an even or odd number of logical 1s. The parity of the data word is checked each time it is read. If the wrong number of 1s is found, a failure message generates. Like TMR, parity effectiveness is negligible on the register interface logic, only provides an effective DC for permanent and transient on FFs.

**Figure 8: Configuration registers and effectiveness of readback solution**

Based on these considerations and depending on the application, LBIST can be a very effective alternative to detect faults on configuration and status register.

### Discussion

Considering the parameters involved in the automatic LBIST generation, the user can evaluate the many trade-offs for what concerns, for example, the size of the logic covered by the test, the area overhead (total cell-area after and before LBIST insertion) and test time for a given clock frequency. Further, the complexity (defining complexity as an aggregate factor of the resulting number of test patterns, area overhead and test time) is to be considered exponential with the test coverage. If the requested test coverage is kept low so it will be the resulting complexity, even with large portions of circuit covered by LBIST. At the same time, even for small percentages of the covered circuit, if the requested test coverage is very high (targeting 99%), the complexity will be asymptotically high, and this behavior will be further exponential rising the amount of covered logic by LBIST. Figure 9 shows an empirical representation of this concept.
Figure 9: Empirical representation of LBIST trade-offs

Table 2 summarizes different safety mechanisms to protect the status and configuration registers and their effectiveness for different metrics. The area overhead is relative to the failure mode to address; for example, in absolute terms, the area overhead is higher for GTMR than for TMR, as TMR only covers the registers, while GTMR also covers the register fan-in logic cones.

<table>
<thead>
<tr>
<th>Safety Mechanism</th>
<th>SPFM</th>
<th>LFM</th>
<th>Note</th>
<th>Test Time</th>
<th>Area Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parity</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>FF Only</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Read-back</td>
<td>~30%</td>
<td>~99%</td>
<td>~30%</td>
<td>Covers Transient Only on FF readable from SW</td>
<td>Depends on register interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>TMR [*]</td>
<td>99.9%</td>
<td>99.9%</td>
<td>–</td>
<td>FF Only</td>
<td>1 Voter Delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Very High</td>
</tr>
<tr>
<td>GTMR [**]</td>
<td>99.9%</td>
<td>99.9%</td>
<td>–</td>
<td>1 Voter Delay</td>
<td>Very High</td>
</tr>
<tr>
<td>LBIST</td>
<td>~97%</td>
<td>–</td>
<td>~97%</td>
<td>Configurable, but possible exponential overheads toward 99% coverage</td>
<td>2.4msec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
</tr>
</tbody>
</table>

[*] TMR: Triple Modular Redundancy
– Triplicated FF + Voter

[**] GTMR: Global Triple Modular Redundancy
Full Logic Triplication, Including Voters

Table 2: Comparison of safety mechanisms for configuration and status registers of a Vision DSP

TMR shows a higher area overhead compared to LBIST but with better timing performances. The LBIST test time can, however, be considered high for online testing in most of the cases, even if LBIST overcomes the main limitation of the TMR about the LFM coverage. Simple parity shows a very low area overhead, but it is only suited to covers the register’s FFs and not the related fan-in logic cones.

An interesting additional consideration can be done about the indirect effect of the register’s safety metrics on the LFM DC of the memories protected by ECC. As a starting point, ECC can offer a 95% DC coverage for LFM (that is degraded, starting from 99%, due to the ECC correction logic). However, the detection/correction signal is not read directly out of the ECC—but, in this design example, the signal goes through the GSERR status register (Figure 3), so a fault in the GSERR affects the overall available DC for the memory. Assuming, for example, that the ErrorCorrected signal in Figure 3 is set to report a corrected error in a memory cell, a stuck-at-1 in the GSERR-related bit has no effect; but a stuck-at-0 “overwrites” the correction signal, and the system will not know there is a fault that is latent over time. Therefore, if the GSERR register is not protected at all (see the first row in Table 3), the overall LFM for the memory will be half of the original achievable (95%) with ECC. Table 3 also shows how the “native” LFM for ECC is affected by different SMs applied on the GSERR register. Note that TMR and parity DC values are scaled compared to Table 2 to consider the interface logic (the logic cones “driving” the GSERR).
It can be seen how LBIST can be very effective to maintain good LFM; Table 3 reports the combined effect of LBIST on status registers and ECC on memory. The final SM architecture selection depends on the application and ASIL requirements, but Table 2 gives a clear guideline example on how to trade off the metrics and to select one or more SM to meet the safety requirements.

<table>
<thead>
<tr>
<th>Technique To Protect The ECC Status Registers</th>
<th>DC on ECC Alarms</th>
<th>LFM on memories protected by ECC</th>
<th>Comments</th>
</tr>
</thead>
</table>
| No Protection                                 | 0.00%            | 47.50%                          | • ECC LFM 99% → 95% due to the ECC correction logic  
• The ECC Alarm is managed through a Status Register (e.g. GSERR).  
  - No → Half of the stuck@0/1 will not be detected → 95%/2 |
| TMR (only FFs)                                | 66.67% [*]       | 63.33%                          | 95% * DC |
| GTMR                                          | 99.90%           | 94.91%                          | 95% * DC |
| Parity (only FFs)                             | 90.00% [*]       | 85.50%                          | 95% * DC |
| LBIST                                         | 96.68%           | 91.85%                          | 95% * DC |

[*] Scaled to take into account the register’s interface logic

Table 3: Diagnostic techniques on ECC status register (GSERR)

Conclusions

This white paper introduced functional safety concepts and reviewed how FMEDA is used to identify areas to focus on, to meet the safety metrics and to derive safety requirements. An example of FMEDA of a Vision DSP has been reviewed to illustrate some of the basic tradeoffs when implementing safety mechanisms. Based on these tradeoffs, several architectures and configurations have been reviewed to evaluate the use of LBIST as a safety mechanism in the context of safety-critical applications and compared to other safety mechanisms such as redundancy, parity, and registers read-back. Guidelines are provided to navigate the many metrics to be met; e.g., SPFM, LFM, PMHF, test time, area overhead. LBIST is particularly appealing since its insertion can be highly automated and LFM DC coverage is high with little area overhead and good test time. Additionally, LBIST applied at the top level of the entire core proves to be a good solution to cover latent faults at the start-up. The indirect effect on cascaded failure modes has been also illustrated where ECC alarm propagation must be protected to maintain good ECC diagnostic coverage.
References

- ISO 26262:2018 Road vehicles - Functional Safety
- IEC 61508:2007, ed.1.0 Functional Safety – Standards