Meeting Diagnostic Coverage Requirements with Mali-G78AE Graphics

INTRODUCTION

This white paper describes methods for achieving diagnostic coverage on the Mali-G78AE GPU when used for graphics applications in Avionics and Automotive. While the discussion is specific to these industries, it addresses functional safety and applies to other applications requiring graphics with functional safety.

Both avionics and automotive safety processes take a similar top-down development approach with software and hardware; the automotive guidance is more prescriptive in guidance than avionics guidance. While both require diagnostic coverage, the automotive standard, ISO 26262, is very specific.

For ISO 26262, ASIL B recommends a Single Point Failure Metric (SPFM) of >90% and for ASIL D that increases to a requirement for >99% coverage. The SPFM is determined by weighting the failure mode diagnostic coverage with the failure rate, thus giving importance to failure modes with higher failure rates.

While ISO 26262 looks at all the failure modes, ARP 4761 PSSA/SSA may focus on the failure modes that can result in a frozen display or active, but misleading display (i.e. Hazardous Misleading Information, HMI). Potential contributors that lead to these hazards include:

1. Hardware failures within the GPU.
2. Design errors within the GPU or software design errors.
3. Failure or inappropriate responses to external events such as EMI, lightning, high operating temperature, or “out of nominal” input power specifications.

This paper focuses on hardware failures, although some techniques also address design errors. Additionally, Latent Failure Metric (LFM) is discussed, specifically for the dual failure case of a hardware failure and a failure in the associate safety net.

A related property to be considered is timing. Detecting failures is valuable if done in a reasonable time, such that the risk of a hazardous outcome is reduced through transitioning to a safe execution state. Otherwise, it can become a system-level hazard. The timing metrics from a fault through to possible hazards are shown in Figure 1.

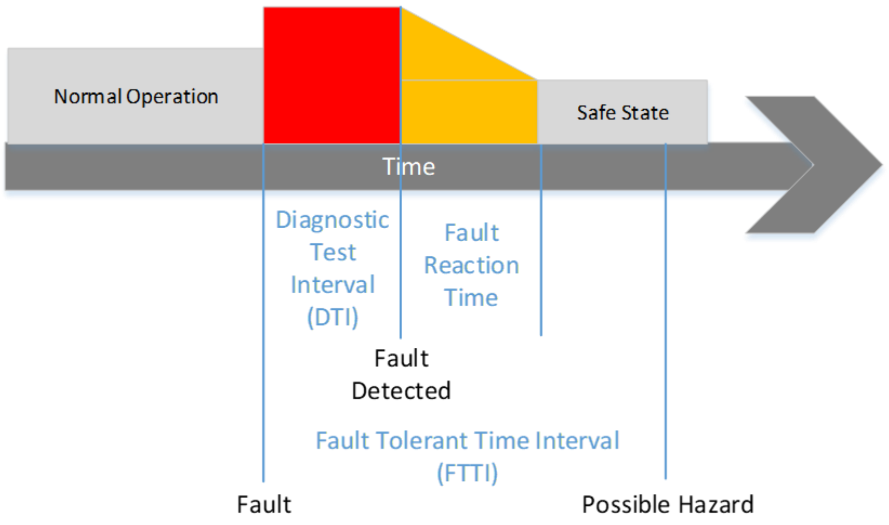

Figure 1: Timing Relationships

The timing parameters in Figure 1 do not have a quantified standard maximum under Functional Safety and are dependent on the application and acceptable risk of a possible hazard. For avionics, AC 25-11B section 3.4.2 states that the electronic attitude display should not be unusable or unstable for more than one second after an electrical transient due to engine failure. From this, we extrapolate that the time from a fault that may lead from HMI to safe mode (e.g. display function restored or crew alerted)—the sum of diagnostic Test Interval and Fault Reaction Time—is less than one second. Since the fault reaction time depends heavily on the application and system architecture, only the Diagnostic Test Interval (DTI) is considered in this paper, as it is the key timing property of diagnostic coverage.

The Mali-G78AE has built-in safety mechanisms around the AXI interfaces (parity) and partition management (Dual-Core Lock Step and Isolation Check), leaving the slices assigned to the safety application cluster partitions.

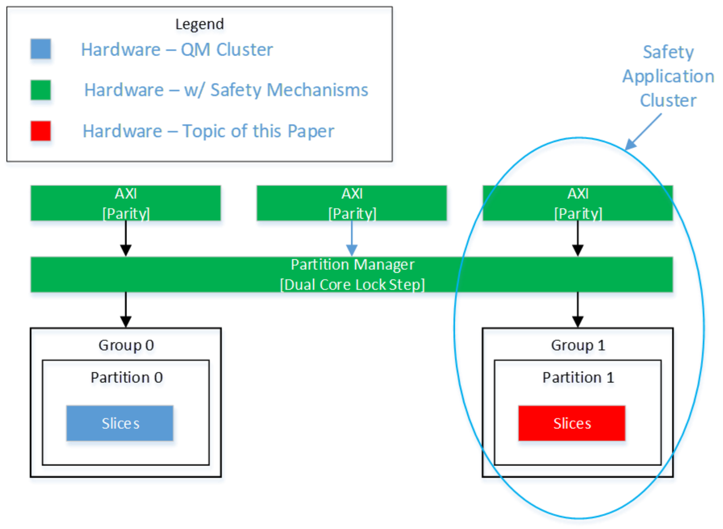
Legend
Hardware – QM Cluster
Hardware – w/ Safety Mechanisms
Hardware – Topic of this Paper
Safety Application Cluster
AXI [Parity]
AXI [Parity]
AXI [Parity]
Partition Manager [Dual Core Lock Step]
Group 0
Partition 0
Slices
Group 1
Partition 1
Slices

Figure 2: Simple View of Mali-G78AE Showing Focus of Paper (red)

This paper presents safety mechanism options for providing the required diagnostic coverage of the safety application slices. Figure 2 provides a simple picture of these elements.

The challenge in meeting the diagnostic coverage on GPUs is that there is a hardware function in each partition that assigns shader core resources to the tasks—the Job Manager in the case of the Mali-G78AE. That is, when there are more than a few shader units, one can no longer feasibly guarantee that all instances of the hardware can be functionally tested. By feasibly, we mean that one could devise a significantly wide enough data set to force the use of all the instances. Completing this action and then checking the result is an unwarranted performance hit.

This paper will describe the primary safety concept and how to address ASIL D and HMI for applications requiring a high level of diagnostic coverage (or when BIST is not supported in silicon).

BUILT-IN-SELF-TEST (BIST)

The primary Mali-G78AE functional safety concept for detecting the accumulation of faults is through an on-chip hardware mechanism called BIST. BIST would be run periodically with the results used to handle the faults to help ensure that the device remains in a safe execution state. BIST is comprised of two types of built-in-self-test: Memory Built-In-Self-Test (MBIST) for memory and Logic Built-In-Self-Test (LBIST) for digital logic.

BIST functionality implemented in the silicon must include a Self-Test Control Unit (STCU). Software interfaces with the STCU to configure and trigger the execution of MBIST and LBIST. Figure 3 shows how a silicon integrator may include LBIST.

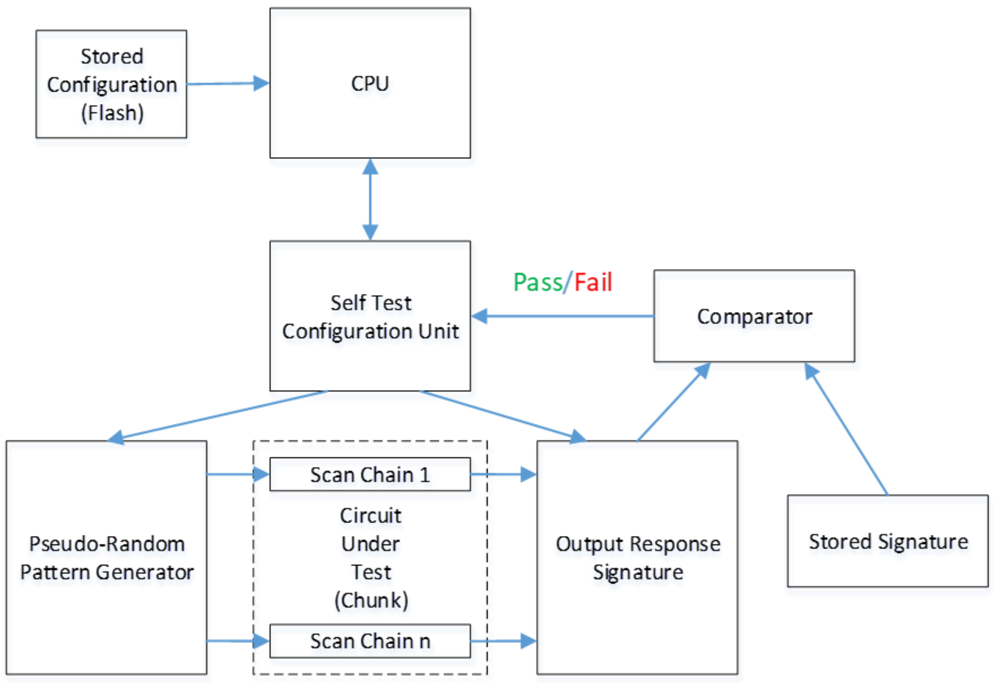
Stored Configuration (Flash)
CPU
Self Test Configuration Unit
Pass/Fail
Comparator
Pseudo-Random Pattern Generator
Scan Chain 1
Circuit Under Test (Chunk)
Scan Chain n
Output Response Signature
Stored Signature

Figure 3: Example Structure LBIST

While it is possible to generate a complete set of test patterns, this is normally avoided due to the cost of generating and the amount of on-silicon memory required to store for complex designs like a GPU. To provide a level of benefit associated with BIST without the overheads, pseudo-random test patterns are applied and checked (simple, fast, and easy to implement). This approach can achieve ISO 26262 ASIL B >90% fault coverage in a reasonable amount of time, as shown in Figure 4.

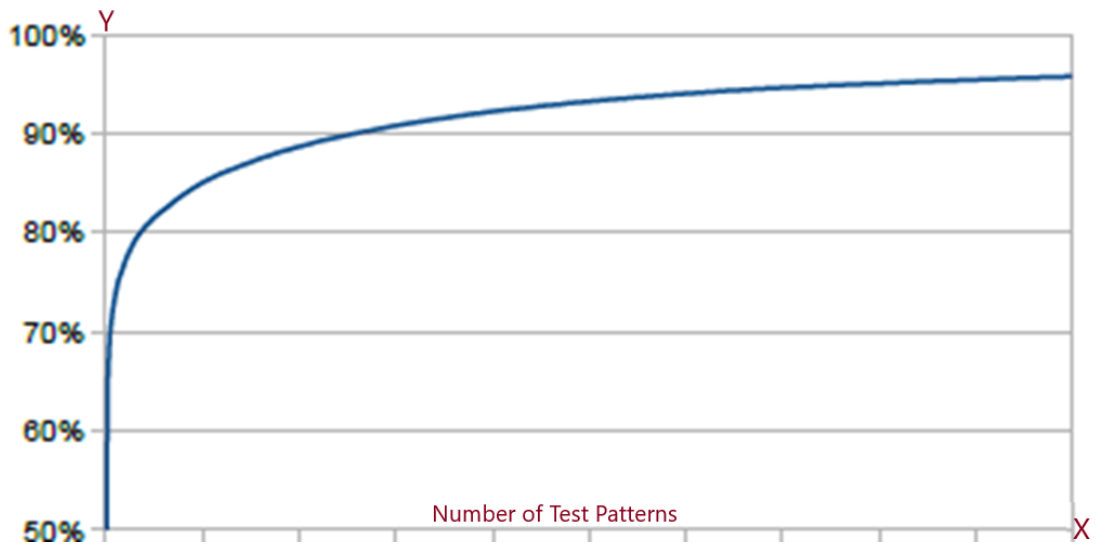

Figure 4: Example Stuck-At Test Coverage as a Function of Numbers of Patterns

The test coverage for the Mali-G78AE is >90%, satisfying the ISO 26262 ASIL B metrics. To claim test coverage, testing of a chunk should be completed once started; however, the health monitor or application can distribute the execution of BIST by choosing when to exercise the individual chunks of gates (i.e. it does not need to exercise all chunks at once). A recommended approach for the application is shown in Figure 5a with a further description of BIST execution provided in Figure 5b.

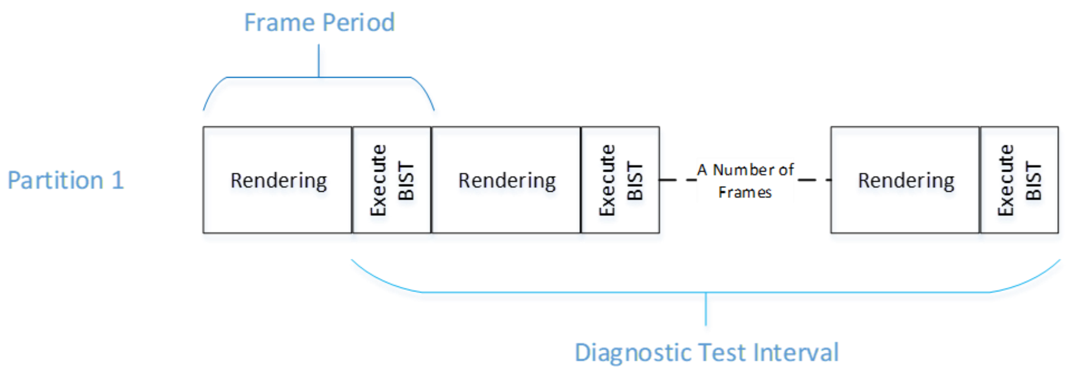

Figure 5a: Example Application Execution Model with BIST

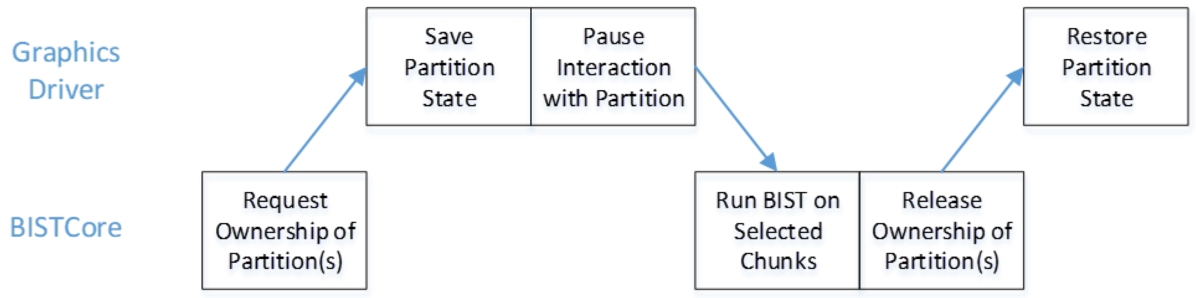
Graphics Driver
BISTCore
Request Ownership of Partition(s)
Save Partition State
Pause Interaction with Partition
Run BIST on Selected Chunks
Release Ownership of Partition(s)
Restore Partition State

Figure 5b: Example Execution Model for BIST

Any transient faults are addressed by frequent execution of BIST, which resets the GPU, thus removing any soft errors in the GPU. As the application renders frames multiple times within the Fault Tolerant Time Interval (FTTI), the GPU will have self corrected any error in the rendered output automatically by rendering the next frame (after reset).

If the silicon integrator connects BIST to the Mali-G78AE, the Mali-G78AE partition arbiter is triggered by BIST management driver software, such as CoreAVI's BISTCore, to request ownership of partition(s) to test in parallel (if more than one). Otherwise, if BIST is not connected to the Mali-G78AE, BISTCore will use a graphics driver API to request ownership of the partition and subsequent release of the partition.

The decision to implement BIST or not, along with determining how to segment the hardware into 'chunks' is done by the silicon integrator. Therefore, details of the segmentation of chunks (size and quantity) and fault isolation is a topic for discussion with respect to specific silicon integrations. The number of chunks and decisions on which ones to run in each timeframe drives the Diagnostic Test Interval (when all chunks have been tested).

With the low-level gate coverage rather than functional coverage per an Failure Modes Effects Diagnostic Analysis (FMEDA), the partition is considered to have failed and the mechanism for a safe state may include reverting to a back-up, using an alternate display, identifying to operator, and so on.

Using BIST has implications for board-level design. The pseudo-random pattern testing toggles more logic than functional operation, which increases the dissipated power. While the silicon integrator may include BIST, it may be implemented to run at clocks that are less than a functional operation to limit the dissipated power during the test. This comes at the cost of execution time. It is estimated that the BIST execution time for the Mali-G87AE is in the 10mS range to test all applicable logic with a scan clock of 100MHz.

LFM is not discussed here because it is dependent on the BIST circuitry implemented by the silicon integrator and the level of protection and self-test offered by the implementation, as well as protection against unwanted access to the BIST Self Test Configuration Unit.

HAZARDOUSLY MISLEADING INFORMATION: ASIL D OR THERE IS NO BIST?

The methods for diagnostic coverage discussed so far do not meet avionics needs when it comes to detecting and preventing Hazardously Misleading Information (HMI) related to DAL A and DAL B systems, as well as being short on meeting ISO 26262 ASIL D metrics. In this section, we will discuss an approach that may be used to meet these stringent requirements and may be considered for silicon implementations where BIST is not implemented.

To discuss this method, the example configuration shown in Figure 6 will be used.

In this configuration Group 1 contains all the safety-critical resources. In this example, Group 1 is partitioned into an application partition, Partition 1, and test partition, Partition 2.

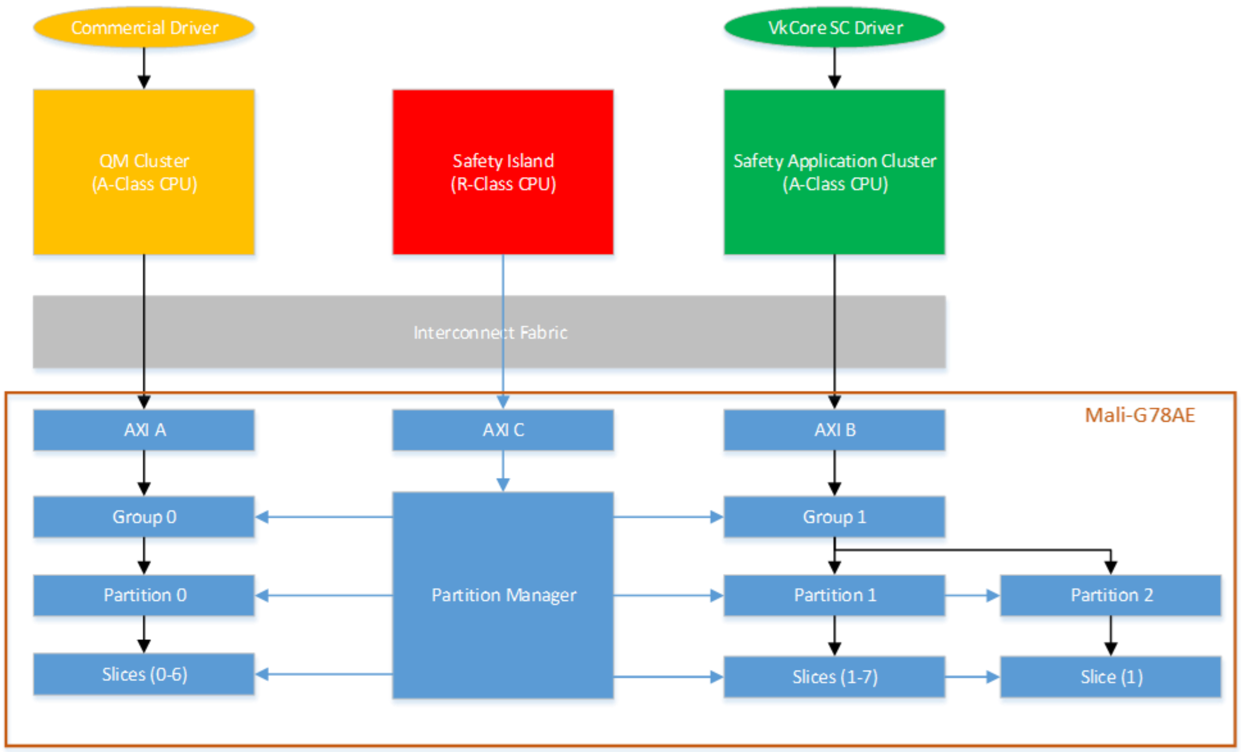

Figure 6: Example Configuration for Functional Safety Coverage

In this scenario, the application creates the final displayed image as layers which are then composited to a frame image for display. The decomposition of the display image into layers enables a smaller GPU (partition 2) to render each of the layers, one at a time with one rendered within the time it takes the application to render all layers (partition 1) and combine them together (compose). Using the GPU-generated Cyclic Redundancy Checks (CRCs,) the CRC for the layer rendered by the small GPU is then compared to the CRC for the same layer rendered by the application. By checking all the layers, the displayed safety-critical portions of the image are known to be correct. The execution model is shown in Figure 7.

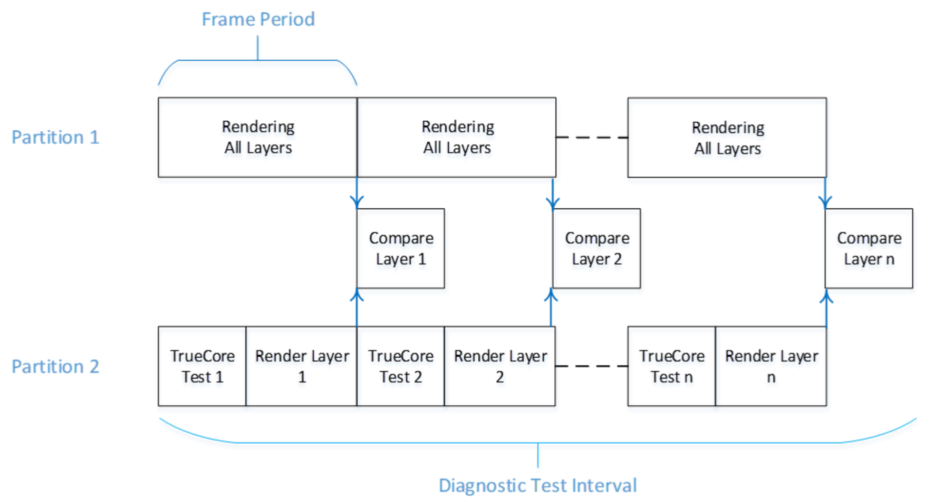

Figure 7: Example Execution Model for Full Frame Compare

To address common-mode failures that may occur, CoreAVI's TrueCore™ testing is performed on the small GPU to check for failures. For efficiency, TrueCore testing involves checking the result of the previous test and then initiating the next test; it does not need to wait for results.

The Diagnostic Test Interval is based on the number of safety-critical layers in the final application image and the number of TrueCore tests, whichever is greater (they do not need to be the same).

Composition could also incorporate images from the Quality Management (QM) ASIL group (partition 0), thus providing a safe mixed-criticality solution.

This solution provides a high SPFM, meeting ASIL D and detection of incorrect display requirements, given that the output frame buffer is compared to a separately generated image. A high LFM is achieved because the rendering is done on independent resources with common mode failure concerns address using TrueCore software GPU safety monitor.

CONCLUSION

Regardless of the required level of Single Point Failure detection for a safety-critical application and no matter how the silicon integrator integrates the Mali-G78AE GPU, there is a method that can be used to achieve required failure coverage.

CoreAVI's Vulkan® SC software driver, VkCore® SC, for the Mali-G78AE is enabled to support hardware arbitration with BIST, as well as a software API for the case in which a silicon integrator does not connect BIST directly to the Mali-G78AE partition arbitration mechanism. CoreAVI supports diagnostic coverage with the TrueCore software library and BISTCore software driver.

All CoreAVI software is designed and developed to meet DO-178C DAL A and ISO 26262 ASIL D.

The information contained in this document is for informational purposes only and is subject to change without notice. CoreAVI, the CoreAVI tracer logo, VkCore®, TrueCore™, and combinations thereof are trademarks of CoreAVI. All other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

© 2020 Core Avionics & Industrial Inc. All rights reserved.