Unified Silicon Runtime™ Functional Safety Architecture

From Redundancy-Centric Safety to Runtime Structural Integrity

Technical Foundations and Deployment Path for VegaSafe™

White Paper | April 2026

Ben Zaryouh
Founder & CEO
VegaSemi
ben@vegasemi.com

Table of Contents

1 Executive Summary

Functional safety is becoming harder to scale in modern SoCs as advanced nodes, heterogeneous compute, and tighter efficiency constraints expose the limits of conventional safety architectures.

Existing approaches typically rely on duplication-centric mechanisms such as lockstep for SPFM and separate in-field diagnostics for LFM. While proven, these methods increase area, power, and verification cost, remain exposed to common-cause failure concerns, and create a fragmented safety architecture. Single-core alternatives based on temporal diversity reduce hardware duplication, but they rely on repeated execution on the same fabric, which can limit structural confidence and impose performance overhead.

VegaSafe introduces a unified single-core runtime structural safety architecture that supports both SPFM and LFM through runtime structural diagnostics applied to the live functional fabric. By replacing fragmented and duplication-heavy safety methods with a coordinated runtime integrity model, VegaSafe provides a more scalable and efficient path to functional safety for modern heterogeneous SoCs.

2 Introduction

Functional safety has traditionally been built on architectural techniques that prioritize determinism, fault detection, and bounded fault handling under clearly defined operating assumptions. These methods have served the industry well and remain foundational in automotive, industrial, aerospace, and other mission-critical systems. However, the silicon and system context in which they now operate has changed substantially.

Modern SoCs are no longer composed of a small number of relatively uniform compute elements operating with generous implementation margin. They increasingly integrate CPUs, GPUs, AI accelerators, DSPs, interconnect fabrics, memory subsystems, and domain-specific engines within aggressive area and power budgets. At the same time, advanced semiconductor nodes introduce tighter timing margins, stronger local variation, sharper thermal gradients, more pronounced aging effects, and greater sensitivity to runtime conditions. As a result, maintaining confidence in functional correctness is becoming more difficult not only at design time, but across the full operating life of the product.

This shift places growing pressure on conventional safety architectures. Existing solutions typically separate the safety problem into two categories. Active fault detection and SPFM-oriented objectives are commonly addressed through duplication-centric methods such as lockstep-class architectures, while latent fault coverage and LFM-oriented objectives are often handled through separate in-field diagnostics or periodic test strategies. These methods are proven and widely deployed, but they also create a fragmented safety model in which different fault classes are addressed by different mechanisms, often with substantial cost in area, power, verification effort, and architectural complexity.

Duplication-centric safety remains effective in many contexts, but it scales poorly as compute complexity and heterogeneity increase. It also remains exposed to common-cause failure concerns and imposes a permanent hardware burden even when the safety goal is confidence rather than constant mirrored execution. Single-core alternatives based on temporal diversity attempt to reduce that burden by re-executing instructions on the same fabric, but they introduce their own limitations. Because the same execution hardware performs both the functional computation and its validation, structural independence is weakened and fault escape becomes a concern. These approaches also impose direct performance cost due to repeated execution and still do not naturally unify SPFM and LFM within one coherent runtime framework.

The broader issue is therefore not simply that existing safety methods are inadequate. It is that they were developed for a different balance of silicon cost, runtime observability, and compute composition than what advanced SoCs now require. As systems become more heterogeneous and more dependent on sustained in-field confidence, functional safety can no longer be treated only as a static design-time property reinforced by hardware duplication and occasional diagnostics. It must increasingly be treated as a runtime architectural problem.

VegaSafe addresses this need through a unified runtime structural safety architecture. Rather than relying on one mechanism for active faults and another for latent faults, VegaSafe supports both SPFM and LFM through structurally meaningful runtime diagnostics applied to the live

functional fabric. In doing so, it shifts the safety model from fragmented and duplication-heavy protection toward coordinated runtime structural integrity.

This paper develops that argument in detail. It first examines why conventional functional safety approaches are coming under increasing pressure in modern SoCs, then introduces the VegaSafe architectural model and its system-level implications. The result is a functional safety framework intended to align more naturally with the realities of advanced heterogeneous silicon: lower overhead than duplication-centric designs, stronger lifecycle relevance than isolated periodic tests, and a more unified basis for safety confidence during field operation.

3 Limitations of Conventional Functional Safety Approaches

Key Observations

- Functional safety requirements are rising while silicon efficiency budgets are tightening.
- Duplication-centric techniques provide strong confidence for active fault detection, but often at substantial implementation cost.
- Latent fault coverage is typically addressed through separate in-field diagnostics, creating fragmented safety handling across SPFM and LFM.
- Single-core temporal-diversity approaches reduce hardware duplication, but structural independence remains limited.
- Heterogeneous compute fabrics complicate direct application of classical CPU-style safety assumptions.
- A more unified runtime safety model is needed for modern SoCs.

Conventional functional safety methods remain foundational and widely deployed, but their limitations are becoming more visible in modern SoCs. Existing approaches typically rely on hardware duplication to support active fault detection and SPFM-oriented goals, while separate in-field diagnostics are used to address latent faults and LFM objectives. Although effective, this creates a fragmented safety model in which different fault classes are handled by different mechanisms, each with its own cost and coverage limitations.

As semiconductor scaling, heterogeneous compute, and lifecycle variability continue to increase, these limitations become harder to ignore. Duplication-centric methods carry substantial area, power, and verification cost, remain vulnerable to common-cause failure concerns, and do not scale cleanly across all compute domains. Periodic diagnostics help extend coverage, but they are interval-bound and structurally limited. Even single-core temporal-diversity approaches reduce hardware cost only partially, while introducing fault-escape concerns and direct performance overhead. Taken together, these issues motivate the need for a more unified runtime safety architecture.

3.1 Why Functional Safety Is Becoming More Difficult

Functional safety is becoming more challenging for reasons that extend beyond standards compliance. Advanced semiconductor nodes reduce timing margin and increase sensitivity to local variation, thermal gradients, aging, and intermittent behavior. At the same time, safety-relevant systems increasingly depend on complex SoCs that integrate multiple compute styles, dynamic software stacks, and changing operational conditions.

This changes the practical nature of the problem. The question is no longer only whether faults can theoretically be covered in an architectural model. The question is whether fault detection, detection latency, and confidence mechanisms remain scalable, economically sustainable, and relevant throughout the real operating life of the product.

The challenge is intensified by the need to achieve both high diagnostic coverage and acceptable implementation efficiency. Safety architectures that are straightforward to justify conceptually can become difficult to scale across modern compute fabrics without substantial area, power, verification, and validation cost. Conversely, lightweight approaches may preserve efficiency while leaving gaps in structural confidence, latent fault handling, or lifecycle relevance.

3.2 Duplication-Centric Architectures for Active Fault Detection

Duplication-centric approaches such as dual-core lockstep remain among the most established techniques for functional safety, especially in CPU-centric control systems. Their attraction is clear: parallel execution combined with comparison provides a direct and intuitive mechanism for detecting divergence during active operation, making them well aligned with SPFM-oriented objectives.

However, the cost of this approach can be substantial. Lockstep requires duplicated compute resources, comparison logic, synchronization infrastructure, fault-handling support, and additional verification effort across both nominal and fault conditions. The area and power burden may be acceptable in some systems, but it becomes increasingly difficult to justify as compute complexity, core count, and heterogeneity increase.

More fundamentally, duplication does not eliminate all safety concerns. Common-cause failures, shared resource dependencies, systematic design issues, and correlated physical disturbances can all reduce the practical independence of duplicated paths unless the entire architecture is carefully engineered around them. As a result, duplication-centric safety remains powerful, but it is not a frictionless or universally scalable answer for modern SoCs.

3.3 Single-Core Temporal-Diversity Alternatives

Single-core approaches based on temporal diversity attempt to reduce the hardware overhead of duplication-centric safety architectures by re-executing instructions or instruction sequences on the same compute fabric at different times. Rather than relying on a mirrored core or permanently duplicated hardware path, these methods seek to derive diagnostic confidence from repeated execution and comparison of results within a single-core framework.

The appeal of this approach is clear. Compared with lockstep-class architectures, temporal-diversity methods can reduce area overhead, lower hardware duplication, and simplify some aspects of implementation. They offer an attractive middle ground for applications that seek stronger safety capability than lightweight software diagnostics alone, but with lower structural cost than full redundancy-centric designs.

However, these benefits come with a fundamental limitation. The same execution fabric is responsible for both functional computation and its validation. Because the diagnostic path is not structurally independent, fault-detection confidence is inherently constrained by the condition of the very hardware being evaluated. Faults may therefore escape detection if a defective structure affects both executions similarly or compromises the repeated instruction flow itself.

These approaches also impose direct performance cost because safety confidence depends on repeated execution. The additional cycles required for re-execution reduce useful compute throughput and can make the safety mechanism harder to scale in performance-sensitive systems. More broadly, while temporal-diversity methods reduce the hardware burden of duplication, they do not naturally unify SPFM and LFM within a single runtime safety framework. They are therefore better understood as a partial reduction of duplication cost than as a complete architectural answer to modern functional safety requirements.

3.4 Periodic Diagnostics and In-Field Test for Latent Fault Coverage

Periodic diagnostics, software test libraries, and related in-field test strategies provide another widely used path to functional safety coverage. These approaches are particularly relevant to latent fault management and therefore play an important role in supporting LFM objectives. Their value lies in the fact that they can leverage existing compute resources, target identified fault classes, and fit well into established safety processes.

Yet these methods also have important limitations. Their diagnostic execution is periodic rather than continuous, which means fault detection remains interval-bound. Their structural reach depends heavily on the quality of the diagnostic content and on the assumptions used to map the applied tests to actual fault coverage. They may also consume valuable execution time, require carefully managed scheduling windows, and become more difficult to scale as system complexity increases.

Most importantly, periodic diagnostics are often deployed as a separate complement to active fault-detection mechanisms rather than as part of a unified safety architecture. In practice, this means one mechanism is used primarily for SPFM-oriented active fault detection, while another is used for LFM-oriented latent fault coverage. The result is a fragmented safety model rather than a single coherent runtime framework.

3.5 Simplified Runtime Safety Models in Heterogeneous Compute

In throughput-oriented compute domains such as GPUs, AI accelerators, and other specialized engines, the industry often adopts lighter or more distributed safety models that combine hardware checks, watchdogs, ECC, software diagnostics, partitioning, runtime monitoring, and workload management. These approaches can be practical and may align better with massively parallel or loosely ordered execution styles than classical CPU lockstep.

However, they also reveal a broader limitation in conventional safety thinking: not all compute fabrics map cleanly to the same architectural safety model. As SoCs become more heterogeneous, applying one CPU-oriented mechanism everywhere becomes unrealistic. The result is often a patchwork of safety strategies with different assumptions, different visibility limits, and different confidence levels across the platform.

This fragmentation becomes increasingly problematic when safety must be argued at the system level. If CPUs, AI engines, DSPs, and accelerators all rely on different safety mechanisms with

different strengths and limitations, the SoC lacks a common runtime safety foundation. What emerges is not a unified safety architecture, but an accumulation of domain-specific methods stitched together at integration time.

3.6 Structural Coverage, Lifecycle Confidence, and Common Architectural Gaps

Functional safety is not only about immediate fault detection. It also depends on maintaining confidence against latent faults, evolving degradation, intermittent behavior, and workload-dependent activation conditions over time. This becomes especially important in advanced silicon, where aging, localized stress, and shifting operational profiles can change which failure mechanisms are most relevant during field life.

Conventional approaches often struggle to maintain this type of lifecycle-relevant structural confidence efficiently. Duplication-centric methods impose permanent hardware cost regardless of actual runtime need. Periodic diagnostics remain bounded by when they are scheduled and by how effectively they activate structurally meaningful fault conditions. Temporal-diversity approaches reduce hardware cost, but their structural independence is limited by reuse of the same fabric. Simplified heterogeneous safety models may preserve practicality, but often at the cost of uneven confidence across domains.

The common gap is therefore not that any one technique is useless. It is that existing techniques tend to optimize one part of the problem while leaving another part unresolved. Some provide strong active fault detection at high structural cost. Some help with latent fault coverage but only periodically. Some reduce duplication cost but weaken independence. Some scale across accelerators only by relaxing the safety model. The result is a set of partial solutions rather than a unified safety architecture.

3.7 Transition to a Unified Runtime Safety Paradigm

These limitations reveal a consistent pattern. Existing safety approaches remain important, but they either carry high permanent hardware burden, provide bounded rather than continuous confidence, or become increasingly fragmented as heterogeneous compute grows.

The central need is therefore for a safety architecture that can strengthen active and latent fault coverage within one framework, operate during normal field life with minimal disruption, align more directly with structural fault objectives, scale across CPU and non-CPU domains, and remain relevant as silicon and workloads evolve over time.

VegaSafe addresses this need by shifting the safety model from fragmented and duplication-heavy protection toward unified runtime structural integrity. Rather than treating SPFM and LFM as separate architectural problems requiring separate classes of solution, it provides a path toward coordinated runtime safety confidence built around the live functional fabric itself. That shift forms the basis for the architectural model introduced in the next section.

4 VegaSafe Architectural Solution

The limitations discussed in the previous section reveal a broader architectural gap in conventional functional safety practice. Existing solutions remain valuable, but they often rely on separate mechanisms for active fault detection and latent fault coverage, carry substantial area and power cost when based on hardware duplication, and remain exposed to common-cause failure concerns. These limitations become harder to manage as SoCs grow more heterogeneous, more performance-sensitive, and more dependent on sustained in-field confidence over product life.

VegaSafe addresses these challenges through two complementary architectural paths. The first strengthens conventional lockstep-class safety by improving observability, coordination, and resilience against limitations that comparison-based duplication alone does not fully resolve. The second provides a single-core lockstep-equivalent safety architecture that supports both SPFM and LFM through unified runtime structural safety mechanisms, without requiring full-time duplicated execution or full reliance on conventional scan-based DFT infrastructure during field operation.

Together, these two paths support both evolutionary and transformational deployment. VegaSafe can enhance incumbent safety architectures already familiar to the industry, while also providing a more efficient single-core path for future systems that require lower overhead, broader scalability, and a more unified safety framework.

4.1 Enhancing Conventional Lockstep Safety

Lockstep remains one of the most established architectural approaches for functional safety, particularly in CPU-centric systems where duplicated execution and comparison provide a direct path to active fault detection. However, conventional lockstep does not fully eliminate all relevant safety limitations. Common-cause failure exposure, limited structural observability beyond comparison mismatch, and fragmented handling of active and latent fault objectives can all reduce the efficiency and completeness of the overall safety architecture.

VegaSafe improves conventional lockstep by extending safety beyond duplicated execution alone. Rather than treating the lockstep pair and comparator as the complete safety solution, VegaSafe adds runtime safety telemetry, monitoring, and control. This broader layer improves system-level visibility, coordination, and response.

A central benefit of this approach is stronger handling of common-cause failure conditions that conventional compare-based lockstep does not fully address on its own. These conditions can include:

- supply droop and voltage-noise events,
- shared thermal stress and correlated thermal gradients,
- aging-related correlated degradation,
- shared clock-source or clock-distribution disturbances,

- shared memory, interconnect, or infrastructure-related fault effects.

With this added runtime layer, safety confidence is no longer limited to divergence detection alone. The architecture can observe and respond to correlated conditions that may affect both duplicated paths simultaneously, strengthening the overall safety posture beyond comparator-based mismatch detection.

VegaSafe also extends the value of lockstep by enabling runtime latent fault management with minimal interruption to normal functionality. In this model, lockstep is no longer treated only as an SPFM-oriented active-fault mechanism that must be complemented separately by coarse in-field diagnostics. Instead, it participates in a broader runtime safety framework that can support LFM objectives during field operation without requiring full dependence on conventional scan-based test structures or pre-generated scan vectors to be available and applied in-field.

Figure 1 illustrates this enhancement path. Conventional dual-core lockstep derives confidence primarily from duplicated compute paths and comparison logic. VegaSafe preserves the familiarity of this model while adding coordinated runtime safety control, common-cause monitoring, and broader system-level safety visibility to strengthen robustness and extend runtime safety coverage.[1]

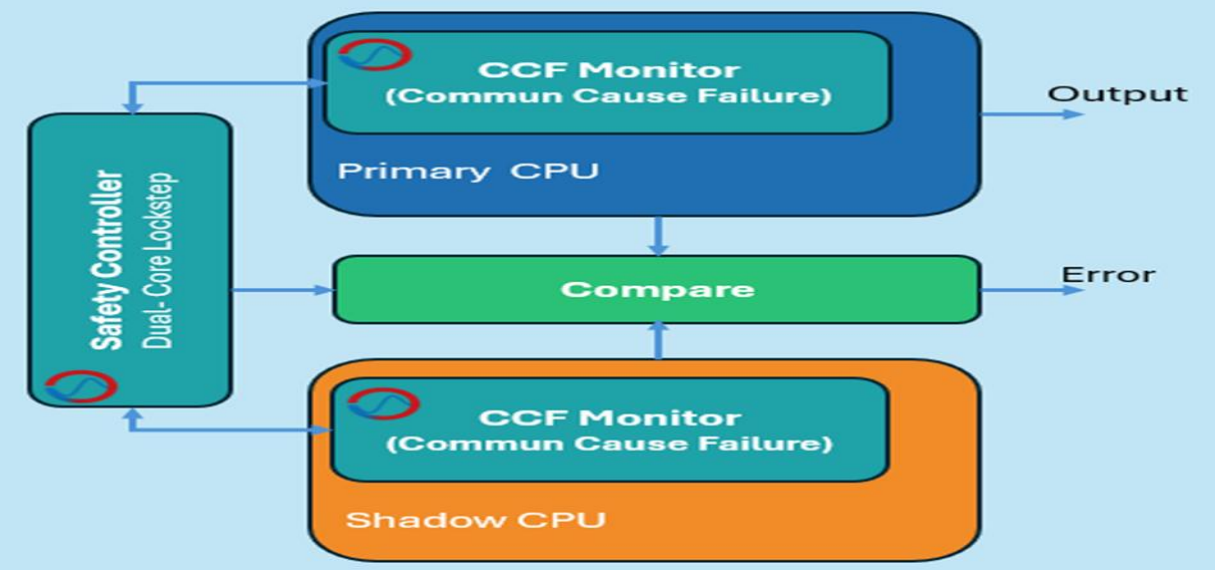

Figure 1. VegaSafe Enhancement of Conventional Dual-Core Lockstep Safety

[1] *The architecture is organized around coordinated runtime safety telemetry, common-cause condition monitoring, and safety management layered on top of conventional duplicated execution and comparison. This broader runtime layer strengthens observability and response to correlated conditions such as supply droop, thermal stress, aging-related degradation, and shared infrastructure effects, while also enabling runtime latent fault support with reduced dependence on conventional in-field scan structures or scan vectors.*

This path matters for both architecture and deployment. Many organizations already rely on lockstep and have safety cases, certification strategies, and implementation flows built around it. VegaSafe therefore provides a practical improvement path that preserves the familiarity of lockstep while elevating it into a more observant, more coordinated, and more complete runtime safety architecture.

4.2 Single-Core Lockstep-Equivalent Safety

VegaSafe also enables a more advanced architectural path: single-core lockstep-equivalent safety. This model is intended to provide strong runtime functional safety confidence without requiring permanently duplicated compute resources as the primary mechanism for protection.

The objective is not merely to reduce hardware duplication. It is to replace the conventional split between active-fault and latent-fault handling with a more unified safety model. Rather than relying on one mechanism for active faults and another for latent faults, VegaSafe supports both SPFM and LFM through coordinated runtime structural safety mechanisms applied during field operation.

Figure 2 illustrates this transformational path. The VegaSafe single-core architecture coordinates SoC safety management, runtime safety control, and compute safety monitoring to support unified SPFM and LFM handling within a single-core framework.[2]

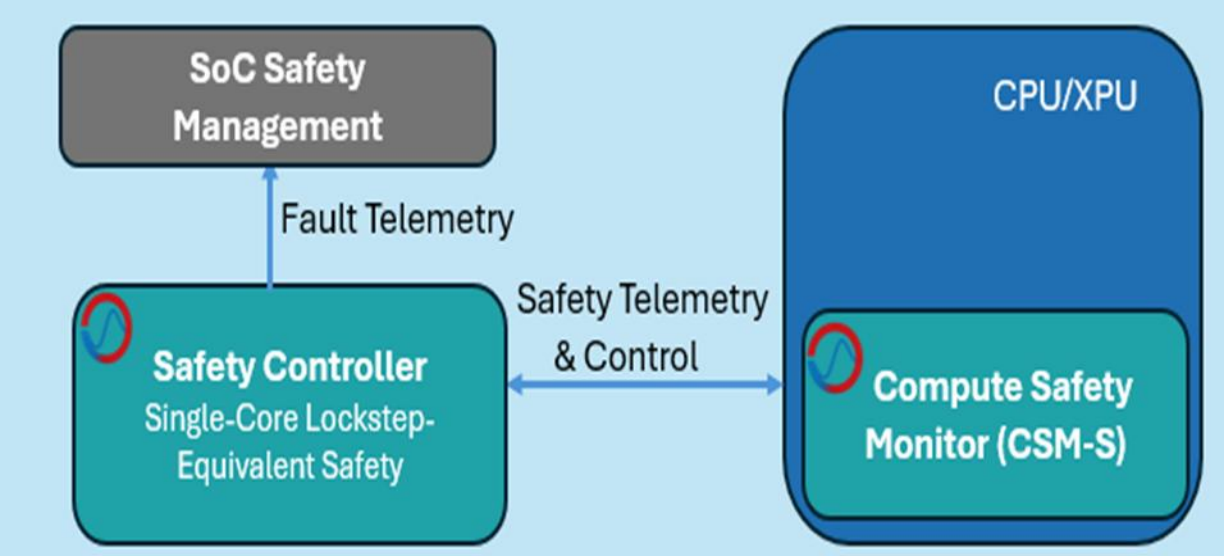

Figure 2. VegaSafe Single-Core Lockstep-Equivalent Safety Architecture

[2] *The architecture is organized around runtime structural safety assessment of the live functional fabric, coordinated through safety telemetry, control, and compute monitoring. It supports both SPFM and LFM within a unified single-core framework, with ATPG-grade diagnostic coverage intent during field operation, minimal functional interruption, and reduced dependence on conventional scan-based DFT infrastructure or scan vectors.*

At the public architectural level, VegaSafe can be understood as enabling runtime structural safety coverage with ATPG-grade diagnostic intent during system operation. This allows the architecture to support active and latent fault objectives within one coordinated safety framework, while avoiding full-time duplicated execution and reducing dependence on conventional scan-based DFT infrastructure during normal field use.

This is a significant shift from conventional single-core alternatives based only on temporal diversity or repeated instruction execution. VegaSafe single-core safety is not positioned as merely a lower-cost replay model. It is positioned as a structurally stronger runtime safety architecture that supports meaningful safety confidence while preserving the efficiency advantages of a single-core implementation.

Another important advantage is that this architecture can operate with minimal disruption to functional behavior. Rather than requiring heavy interruption of mission-mode operation, VegaSafe is designed to support runtime safety coverage in a manner compatible with practical field deployment. This makes the approach more relevant to modern SoCs, where safety must coexist with demanding performance, power, and availability requirements.

The result is a lower-overhead, unified single-core safety architecture. It reduces area and power burden relative to duplication-centric solutions, supports SPFM and LFM within one framework, reduces fragmentation between active and latent fault handling, and aligns more naturally with heterogeneous SoCs where full duplication is increasingly difficult to justify across all compute domains.

This single-core lockstep-equivalent path is particularly important for future CPU, GPU, XPU, and mixed-compute systems, where the cost of conventional duplication continues to rise and where safety must scale across different execution models. By establishing runtime structural confidence directly in the live functional fabric, VegaSafe provides a path toward high-confidence single-core safety that is more efficient, more unified, and better aligned with modern SoC realities than conventional approaches built around permanent duplication or repeated self-check alone.

4.3 Architectural Significance

Taken together, these two VegaSafe paths define a broader safety strategy than either conventional lockstep or lightweight single-core alternatives alone. VegaSafe can improve incumbent dual-core safety deployments by strengthening runtime visibility, safety coordination, and common-cause mitigation, while also enabling a more efficient single-core lockstep-equivalent architecture for systems that require lower overhead and stronger architectural unification.

This dual-path model is strategically important. It gives the industry both a near-term path and a long-term path. The near-term path improves existing lockstep-based systems without requiring immediate architectural disruption. The long-term path introduces a single-core runtime structural safety framework capable of supporting both SPFM and LFM in a more unified and scalable manner.

In that sense, VegaSafe is not simply another safety mechanism added to an SoC. It is a broader architectural model for how modern functional safety can evolve: from fragmented, duplication-heavy protection toward coordinated runtime structural safety across the live silicon platform.

5 System-Level Benefits of VegaSafe

Key Observations

- VegaSafe creates value through both enhancement of existing lockstep deployments and a lower-overhead single-core safety path.
- The lockstep enhancement path improves common-cause observability, runtime latent-fault support, and architectural cohesion.
- The single-core lockstep-equivalent path reduces duplication overhead while unifying SPFM and LFM within one framework.
- Both paths strengthen lifecycle confidence and align naturally with broader Silicon Life Management objectives.
- VegaSafe fits more naturally than conventional safety architectures in heterogeneous modern SoCs.

5.1 Benefits of Enhancing Conventional Lockstep

VegaSafe delivers immediate value even when the underlying safety architecture remains based on conventional dual-core lockstep. This is important because many existing products and safety programs already depend on lockstep and are not in a position to replace it abruptly. In this context, VegaSafe acts as an architectural enhancement path that improves the practical effectiveness of lockstep without requiring abandonment of the incumbent model.

Key benefits of this path include:

- **Stronger common-cause visibility and response.**
  VegaSafe extends safety beyond comparator-based mismatch detection and improves observability of correlated conditions such as supply disturbance, thermal stress, aging-related degradation, clock-related effects, and shared infrastructure dependencies.
- **Runtime support for latent fault management.**
  Conventional lockstep is strongest as an active-fault mechanism. VegaSafe broadens its value by enabling stronger runtime LFM support during field operation.
- **Minimal functional interruption.**
  Latent-fault support can be strengthened without requiring heavy disruption of normal mission-mode behavior.
- **Reduced dependence on in-field scan structures and scan vectors.**
  The architecture reduces the need to depend fully on conventional scan-based mechanisms during field use.
- **Better cohesion between SPFM and LFM handling.**
  Instead of treating active and latent fault objectives as largely separate architectural domains, VegaSafe helps bring them into a more coordinated runtime safety model.

The result is a more complete and more deployable lockstep-based safety architecture with stronger runtime visibility and broader practical safety value.

5.2 Benefits of Single-Core Lockstep-Equivalent Safety

The transformational VegaSafe path provides a different class of benefit. By enabling single-core lockstep-equivalent safety, VegaSafe reduces reliance on permanent hardware duplication as the main source of runtime safety confidence. This creates a more efficient path for future systems that require strong safety with tighter area, power, and scalability constraints.

Key benefits of this path include:

- **Lower area overhead.**
  The architecture reduces the need for duplicated compute resources and the additional structural burden that comes with them.
- **Lower persistent power cost.**
  Safety confidence is no longer tied to the continuous power penalty of full duplicated execution.
- **Unified support for SPFM and LFM.**
  VegaSafe supports both active and latent fault objectives within one coordinated runtime framework rather than splitting them across separate mechanisms.
- **Reduced architectural fragmentation.**
  A single-core runtime structural safety model provides a more coherent safety foundation than separate duplication-centric and periodic diagnostic solutions.
- **Better scalability across heterogeneous SoCs.**
  The architecture aligns more naturally with systems that integrate CPUs, GPUs, XPUs, AI engines, DSPs, and other domain-specific compute blocks.
- **Better fit for performance-sensitive systems.**
  VegaSafe is designed for safety coverage with minimal functional interruption, making it more practical for modern high-performance SoCs.

The result is a lower-overhead safety architecture that preserves strong runtime confidence while fitting more naturally into future mixed-compute platforms.

5.3 Lifecycle Confidence and Silicon Life Management

A major system-level benefit of VegaSafe is that it strengthens confidence over the full operating life of the product. Functional safety is not only about detecting immediate active faults. It also depends on maintaining trust as silicon ages, workloads evolve, thermal behavior shifts, and degradation mechanisms accumulate over time.

This lifecycle relevance aligns naturally with broader Silicon Life Management objectives. VegaSafe contributes to a system architecture in which silicon is not only protected, but also observed and managed across field life.

Key benefits in this area include:

- **Renewed confidence during product life.**
  Safety confidence can be refreshed during operation rather than assumed only from initial qualification and isolated periodic testing.
- **Better handling of aging-related effects.**
  The architecture remains relevant as degradation accumulates over time.
- **Improved visibility into intermittent and evolving behavior.**
  Runtime operation provides a better basis for addressing faults and sensitivities that may shift with workload and usage history.
- **Stronger alignment with SLM objectives.**
  VegaSafe supports a broader model in which silicon integrity, safety confidence, and operational longevity are managed together.
- **Improved field trust and operational longevity.**
  The system can maintain a more meaningful confidence model throughout real deployment life.

In this sense, VegaSafe helps extend functional safety from a static design property into a lifecycle-aware runtime discipline.

5.4 System-Level Impact

Taken together, these benefits make VegaSafe relevant at more than the mechanism level. It improves the practical value of existing lockstep deployments, enables a lower-overhead path for future safety architectures, strengthens lifecycle confidence, and aligns naturally with broader SLM and USRA objectives.

The overall system-level impact includes:

- **stronger practical safety value for existing lockstep-based systems,**
- **a more efficient path to high-confidence safety in future single-core architectures,**
- **unified treatment of SPFM and LFM within one runtime framework,**
- **better lifecycle confidence across real field operation,**
- **stronger alignment with Silicon Life Management,**
- **improved coordination with power, reliability, diagnostics, and security,**
- **and better scalability across heterogeneous modern SoCs.**

The result is a safety architecture that is more efficient, more unified, and more relevant to modern silicon realities than conventional approaches based solely on permanent duplication and separate periodic diagnostics.

6 Integration and Deployment Path

A strong functional safety architecture must provide not only technical value, but also a credible path to adoption. This is especially important in safety-critical systems, where architectural change is constrained by validation effort, certification expectations, installed design flows, and organizational risk tolerance. For that reason, VegaSafe is designed to support both evolutionary and transformational adoption rather than requiring abrupt replacement of incumbent safety practice.

The adoption path is organized around a simple principle: improve what exists first, build confidence under real deployment conditions, and then expand architectural scope where the value is clear.

6.1 Why Adoption Resistance Exists

Adoption resistance in functional safety is not driven only by technical conservatism. It is driven by the practical realities of qualification, certification, implementation cost, and product risk. Established approaches such as lockstep remain attractive not because they are ideal in every respect, but because they are familiar, auditable, and already integrated into existing safety processes.

Key sources of adoption resistance include:

- **certification inertia,** because incumbent safety mechanisms are already embedded in safety cases and assessment flows,
- **validation burden,** because any new safety architecture must be justified across fault coverage, behavior, and operational assumptions,
- **implementation risk,** because teams are reluctant to disrupt proven product schedules with abrupt architectural change,
- **organizational familiarity,** because lockstep and periodic diagnostics are widely understood across design, safety, and certification teams,
- **tool and flow dependence,** because existing development infrastructure is often built around conventional safety assumptions.

For these reasons, a successful deployment model must reduce risk while preserving a clear path to architectural improvement.

6.2 Evolutionary Path: Enhancing Existing Lockstep

The first adoption path is evolutionary. VegaSafe can be introduced as an enhancement layer around existing dual-core lockstep architectures rather than as a replacement for them. This allows organizations to strengthen safety capability while preserving the incumbent compute architecture, compare-based active fault detection model, and broader certification posture.

In this path, VegaSafe is used to extend the practical value of lockstep in several ways:

- **improving visibility into common-cause conditions** such as supply disturbance, thermal stress, aging-related degradation, and shared infrastructure effects,
- **adding coordinated runtime safety telemetry and control** beyond comparator mismatch alone,
- **supporting runtime latent-fault management** with minimal interruption to normal operation,
- **reducing dependence on conventional in-field scan structures and scan vectors** for selected runtime safety objectives,
- **improving cohesion between SPFM and LFM handling** within a broader runtime safety framework.

This is the lowest-risk adoption path because it does not require abandonment of familiar safety architecture. It allows VegaSafe to be introduced where the value is immediately understandable: as a practical way to elevate existing lockstep deployments.

6.3 Evidence-Building Phase

Before a broader architectural transition is attempted, deployment should pass through an evidence-building phase. In this stage, VegaSafe operates in a bounded and well-defined role while the system gathers confidence in runtime behavior, diagnostic value, and architectural fit.

The purpose of this phase is to demonstrate:

- **runtime relevance,** by showing that the architecture provides meaningful safety insight under real workloads and operating conditions,
- **coverage contribution,** by showing how VegaSafe strengthens active and latent fault handling in practice,
- **operational compatibility,** by showing that the architecture can run with minimal disruption to mission-mode behavior,
- **integration feasibility,** by showing compatibility with existing safety, validation, and product flows.

This phase is critical because it turns VegaSafe from a conceptual improvement into a deployment-proven safety capability. It also provides the foundation for expanding architectural authority in later phases.

6.4 Transformational Path: Single-Core Lockstep-Equivalent Deployment

Once confidence has been established, VegaSafe enables a more transformational path: single-core lockstep-equivalent safety. This path is intended for future systems in which permanent duplication becomes increasingly difficult to justify because of area, power, performance, or heterogeneous scaling constraints.

In this path, VegaSafe becomes the primary runtime structural safety framework rather than an enhancement around conventional duplicated execution. The architecture supports both SPFM

and LFM within one coordinated single-core model and provides a more unified basis for runtime safety confidence.

This path is particularly attractive where the system requires:

- **lower area and power overhead** than duplication-centric safety,
- **unified SPFM and LFM support** rather than separate architectural mechanisms,
- **minimal functional interruption** during runtime safety operation,
- **reduced dependence on conventional scan-based field diagnostics,**
- **better scalability across CPUs, GPUs, XPUs, AI engines, and mixed-compute platforms.**

This transformational path should not be presented as an abrupt replacement of existing safety practice. It is better understood as the natural next step once runtime evidence, deployment experience, and architectural confidence have been established.

6.5 Deployment Progression

A practical deployment sequence for VegaSafe can therefore be summarized as follows:

- **Step 1: augment existing lockstep systems** with VegaSafe runtime safety telemetry, common-cause monitoring, and runtime latent-fault support.
- **Step 2: build deployment evidence** under real operating conditions and establish confidence in safety contribution, runtime compatibility, and implementation value.
- **Step 3: broaden runtime authority** by increasing the role of VegaSafe in ongoing safety management and confidence maintenance.
- **Step 4: deploy single-core lockstep-equivalent safety** in systems where lower-overhead unified safety becomes the preferred architectural direction.

This progression allows industry adoption to proceed in a controlled way rather than through disruptive replacement.

6.6 Why This Path De-Risks Adoption

The strength of this deployment model is that it aligns with how safety architectures are actually adopted in practice. It does not require the industry to discard familiar methods before trust has been established. Instead, it starts with enhancement of what is already deployed, proves value in runtime use, and then expands into architectures that are more efficient and more scalable.

This approach de-risks adoption in several important ways:

- **it preserves compatibility with current lockstep-based systems,**
- **it provides a practical near-term benefit before larger architectural transition,**
- **it allows confidence to be built through runtime evidence rather than assumption,**
- **it supports gradual expansion from enhancement to primary safety role,**

- **and it gives product teams both an immediate improvement path and a future strategic direction.**

In that sense, VegaSafe is not only a stronger technical architecture. It is also a deployable one. Its adoption path is designed to match the realities of safety engineering: incremental where necessary, transformational where justified, and always grounded in building confidence through real system use.

7 Why This Matters Now

The need for a new functional safety architecture is no longer theoretical. It is being driven by the practical direction of modern silicon platforms. Advanced nodes continue to tighten timing margins, amplify variation sensitivity, and increase dependence on runtime conditions. At the same time, SoCs are becoming more heterogeneous, more compute-dense, and more constrained by area, power, and thermal limits. Under these conditions, conventional safety approaches become increasingly costly to scale and increasingly fragmented in how they address active and latent fault objectives.

This pressure is especially visible in emerging Physical AI systems. Robotics, autonomous machines, intelligent edge platforms, and other physically interactive compute systems increasingly require three things at once: high compute capability, strong functional safety, and low-power operation. That combination is difficult to satisfy with conventional safety architectures built primarily around permanent duplication and separate periodic diagnostics. As compute intensity rises, the area and power cost of duplication becomes harder to justify. As safety expectations rise, fragmented handling of SPFM and LFM becomes harder to sustain. As efficiency constraints tighten, the architectural penalty of conventional methods becomes more visible.

The issue is therefore not simply that existing safety methods are obsolete. It is that they were developed for a different balance of compute density, architectural heterogeneity, and efficiency pressure than what many emerging systems now demand. Modern platforms increasingly need safety architectures that can scale with mixed compute fabrics, maintain confidence over product life, and do so without imposing excessive structural overhead.

This is why VegaSafe matters now. It provides a path to stronger runtime safety confidence through a more unified architectural model: one that can enhance incumbent lockstep systems where needed, while also enabling a lower-overhead single-core lockstep-equivalent path for future platforms. In doing so, it aligns functional safety more naturally with the realities of advanced SoCs, Physical AI, and power-constrained high-performance silicon.

The broader industry shift is clear. Future systems will not succeed by treating safety, compute scaling, and efficiency as separate design problems. They must be addressed together. VegaSafe is designed for that convergence.

8 Conclusion

Modern functional safety is approaching the limit of what can be efficiently achieved through permanent hardware duplication, fragmented handling of SPFM and LFM, and bounded periodic diagnostics alone. These methods remain important and will continue to play a role, but their structural cost, common-cause exposure, and limited scalability are becoming more visible as SoCs grow in complexity, heterogeneity, and runtime sensitivity.

VegaSafe addresses this challenge through a broader and more flexible architectural model. It strengthens existing dual-core lockstep deployments by improving observability of common-cause conditions, extending runtime latent-fault support, and reducing dependence on conventional in-field scan structures and scan vectors for selected safety objectives. At the same time, it enables a transformational single-core lockstep-equivalent safety path that supports both SPFM and LFM within a unified runtime structural safety framework.

This combination is important because it gives the industry both an evolutionary path and a transformational one. VegaSafe does not require abrupt abandonment of incumbent safety practice. It allows existing systems to be enhanced in practical ways while also establishing a more scalable direction for future safety-critical SoCs that must deliver strong functional safety under tighter area, power, and compute constraints.

In this sense, VegaSafe is not merely another safety mechanism. It represents a broader transition in how modern SoCs can achieve and maintain functional safety: from duplication-heavy and fragmented protection toward coordinated runtime structural safety across the live silicon platform.

As heterogeneous compute, Physical AI, and mission-critical systems continue to expand, that transition will become increasingly important. Future platforms will require high compute capability, strong functional safety, and low-power operation at the same time. VegaSafe provides a foundation for that future.

