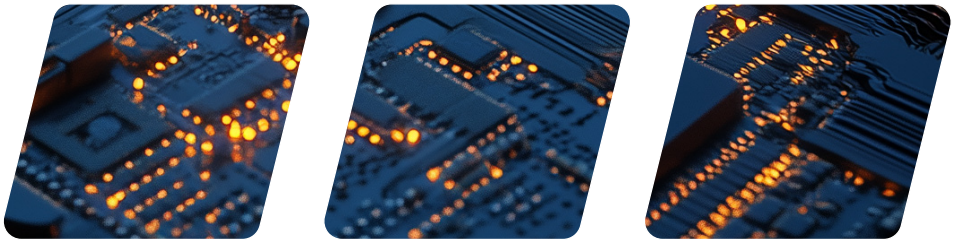


Lattice FPGAs Offer Flexible Hardware Virtualization for Server Management



White Paper

Author:
Munir Ahmad, Distinguished Engineer, Data Center Architecture, Lattice Semiconductor

DISCLAIMERS

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. Lattice products and services are not designed, manufactured, or tested for use in life or safety critical systems, hazardous environments, or any other environments requiring fail-safe performance, including any application in which the failure of the product or service could lead to death, personal injury, severe property damage or environmental harm (collectively, "high-risk uses"). Further, Buyer must take prudent steps to protect against product and service failures, including providing appropriate redundancies, fail-safe features, and/or shut-down mechanisms. Lattice expressly disclaims any express or implied warranty of fitness of the products or services for high-risk uses. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

INCLUSIVE LANGUAGE

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language FAQ 6878 for a cross-reference of terms. Note that in some cases such as register names and state names, it has been necessary to continue to use older terminology for compatibility.

ABSTRACT

The fragmented server ecosystem, with its multiple System on Chip (SoC) architectures and compute platforms, creates a dependency in which the system management software package is locked to specific hardware. Lattice FPGAs remove this dependency by providing a vendor-neutral hardware abstraction layer that virtualizes the platform interface, enabling any BMC software to interoperate seamlessly with any compute vendor's hardware, and giving designers the flexibility to scale one architecture across their entire product portfolio.

Modern server platforms spanning enterprise, AI, and modular computing depend on Baseboard Management Controllers (BMCs) for out-of-band monitoring, firmware management, and security enforcement. As hardware configurations diversify to serve evolving workloads, the traditional approach of tightly coupling BMC software to specific endpoints creates lengthy validation cycles and delayed product launches. This white paper introduces a flexible BMC architecture that uses an FPGA-based hardware abstraction layer from Lattice Semiconductor to decouple hardware variation from the Baseboard Software Package (BSP). The result is a single, unified BSP that works across multiple server configurations, dramatically reducing development effort, accelerating time to market, and lowering total cost of ownership.

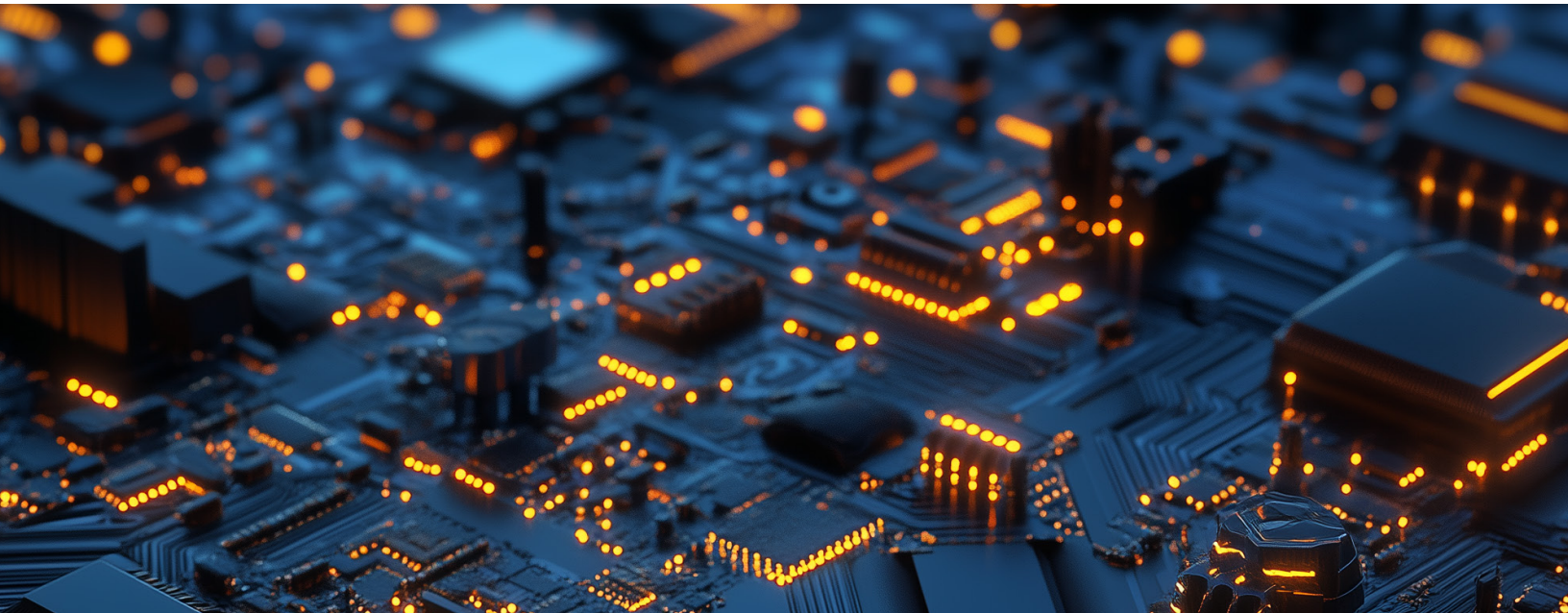


TABLE OF CONTENTS

Disclaimers 2
Inclusive Language 2
Abstract 2
Introduction 4
The Role of BMC in AI Systems 4
BMC Complexity and Baseboard Software Package 5
The Lifecycle Tradeoffs of Fixed BMC ASICs 5
FPGA-Based Hardware Abstraction: A New Approach 6
Architecture Overview 6
Layered BMC and System Management Architecture 8
Benefits of the FPGA-Based Flexible BMC Architecture 8
Conclusion 9

Introduction

Modern server platforms are evolving at an unprecedented pace, driven by the rise of AI workloads, heterogeneous accelerators, disaggregated architectures, and modular system designs. As compute density increases and platform configurations diversify, system management infrastructure must scale in parallel, both in capability and flexibility. At the center of this infrastructure is the Baseboard Management Controller (BMC), a critical subsystem responsible for out-of-band management, platform security, telemetry, and lifecycle control.

Historically, BMC implementations have relied on fixed-function ASICs paired with tightly coupled Baseboard Software Packages (BSPs). While this model has provided a stable and well-integrated foundation for traditional enterprise platforms, increasing hardware diversity and platform variation introduce additional software adaptation and validation considerations. Even modest hardware variations, such as changes in sensor topology, power delivery, storage devices, or accelerator configurations, often require BSP updates, revalidation, and extended integration cycles. For AI and modular platforms with hundreds of endpoints and frequent hardware updates, this drives higher engineering effort and longer integration timelines.

At the same time, security expectations for platform management have grown significantly. Features such as Platform Firmware Resilience (PFR), hardware Root of Trust, cryptographic attestation, and future post-quantum readiness are no longer optional. Fixed BMC ASIC architectures, with security models frozen at silicon tape-out, struggle to keep pace with emerging threats and evolving compliance requirements.

This white paper explores a fundamentally different approach: a flexible BMC architecture built on FPGA-based hardware abstraction. By introducing a Lattice FPGA between the BMC processing element and platform endpoints, hardware variability is absorbed into programmable logic rather than software. The BMC processor interacts with a consistent, standardized interface, enabling a single unified BSP to operate across multiple boards, SKUs, and generations.

The result is a decoupled hardware/software model that aligns BMC evolution with modern server design cycles, enabling platform teams to scale endpoints, customize interfaces, and deploy new security capabilities through FPGA reconfiguration without re-spinning silicon or fragmenting software stacks. Because the Lattice FPGA implements the hardware virtualization for BMC as reconfigurable logic, the resulting management architecture is inherently vendor-agnostic: platform teams are not locked into a feature set or release cadence, and can port, extend, or migrate BMC functionality across product generations. This paper details the architectural principles, hardware and software design, and tangible technical and business benefits of adopting an FPGA-based flexible management strategy for next-generation AI, enterprise, and modular systems.

The Role of BMC in AI Systems

The BMC is a specialized embedded system typically located on the motherboard, Host Processor Module (HPM), or Secure Control Module (SCM) of servers and complex computing platforms. Its primary role is out-of-band management, independent of the host CPU and operating system.

Key BMC Functions

- **Hardware health monitoring**

  Continuous tracking of temperature, voltage, fan speed, power status, and system health indicators. Built-in support for NIST SP 800-193 Platform Firmware Resiliency, Device Identifier Composition Engine (DICE), and Zero Trust architectures.
- **Remote management and control**

  Power on/off, reset, and remote troubleshooting capabilities for data center-scale deployments.
- **Firmware management**

  Secure flashing and updating of BIOS/UEFI and other platform firmware components.
- **Security and attestation**

  Support for Secure Boot, Platform Firmware Resilience, cryptographic attestation, and trust anchoring.
- **Telemetry and logging**

  Collection of logs and operational data for diagnostics, predictive maintenance, and compliance.

Endpoint Management Interfaces

To perform these functions, a BMC communicates with numerous hardware endpoints using protocols such as:

- I2C / SMBus (sensors, VR controllers)
- SPI / QSPI (flash devices)
- PCIe® or FPGA-based bridges
- Platform Environment Control Interface (PECI) for processor telemetry
- Network-based APIs such as Redfish

BMC Complexity and Baseboard Software Package

The BMC is a specialized embedded subsystem on a server motherboard that handles hardware health monitoring, remote management, firmware updates, security attestation, and telemetry. To do this, it communicates with dozens (sometimes hundreds) of hardware endpoints over interfaces such as I2C / SMBus, SPI / QSPI, GPIO, PCIe sideband, and PECI.

As platforms grow in complexity, the number of managed endpoints scales sharply. Variations in endpoint topology and interface composition flow directly into BSP components including drivers, device trees, and initialization logic, making platform evolution a continuous integration challenge.

BMC Complexity

- Standard servers: ~30 endpoints
- AI systems: 100+ endpoints
- High-end modular systems: 200+ endpoints

Baseboard Software Package

The BSP is the software foundation that enables the BMC to interact with platform hardware. A modern BMC BSP can exceed 100 MB in size and represents a significant engineering investment.

Typical BSP components:

- Device drivers
- Board-specific configurations
- Initialization sequences
- Firmware update logic
- Monitoring and telemetry

Hardware changes typically surface at the interface level, and each one can require BSP updates:

- I2C / SMBus topology changes (sensors, VRs, EEPROMs)
- SPI / QSPI device variation (flash layout, CPLDs)
- GPIO / SGPIO configuration differences (reset, power, presence signals)
- PCIe endpoint variation (GPU, NIC, accelerator control paths)
- PECI / CPU telemetry evolution

These variations require updates to BSP drivers, configuration data, and initialization sequences, followed by system-level validation.

The Lifecycle Tradeoffs of Fixed BMC ASICs

There are several architectural considerations associated with ASIC-based BMC implementations:

- The core tension: Server hardware evolves on 12- to 18-month cycles, while BMC ASICs evolve on 3- to 5-year cycles. This difference introduces alignment challenges as platform requirements change.
- BSP dependency on hardware: The BSP is tightly coupled to each platform's endpoint configuration. Changes in interfaces, such as I2C topology, SPI devices, GPIO mappings, or PCIe-attached components, propagate into software and require updates to drivers, device trees, and initialization logic, followed by validation.

- Fixed peripheral sets: BMC ASICs come with a fixed set of I2C buses, SPI controllers, GPIOs, and ADC channels. When a platform needs additional or different interfaces, designers must add external muxes, expanders, or bridge chips. This adds more components and increases system complexity.
- Architecture boundaries: Core architecture elements are fixed at silicon design time, so software must adapt as platforms evolve.
- Security evolution: ASIC firmware operates on a fixed trust model. Adding features such as a hardware Root of Trust, custom attestation flows, or post-quantum cryptography must be carefully aligned with changing platform requirements over time.

FPGA-Based Hardware Abstraction: A New Approach

To address these challenges, Lattice offers a solution that introduces an FPGA-based hardware abstraction layer between a processing element and a platform endpoint. See Figure 1.

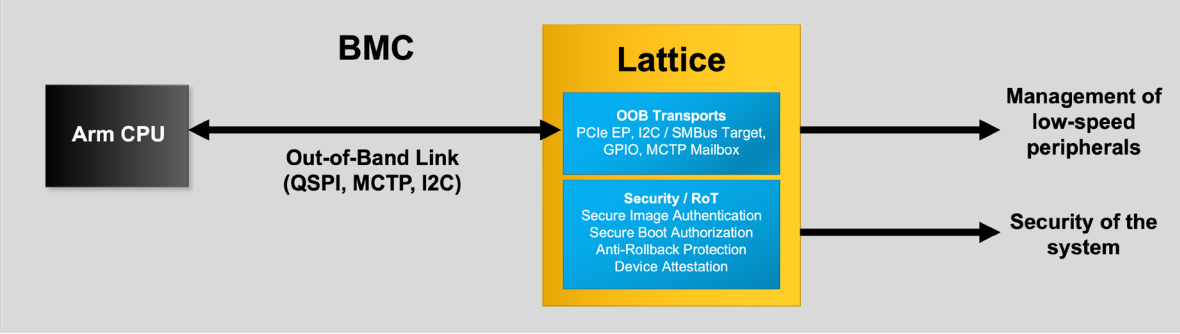
Core Concept

Instead of embedding hardware-specific logic in the BSP:

- The FPGA interfaces directly with hardware endpoints across I2C, SPI, GPIO, PCIe sideband, and PECI.
- The processor communicates with the FPGA through a standardized interface, regardless of physical topology.
- The FPGA absorbs endpoint topology and interface changes, so the BSP does not change.

Architecture Overview

Figure 1: BMC system architecture overview based on an example solution from an Arm CPU and Lattice collaboration



Traditional Model: Hardware variation (I2C / SPI / GPIO / PCIe changes)

- Driver / device tree modification
- BSP rebuild
- Platform validation

FPGA Abstraction Model: Hardware variation

- FPGA interface adaptation (bitstream update)
- Stable BMC interface
- No BSP modification

The Lattice FPGA is programmed to support multiple endpoint interfaces and protocols, including:

- I2C / SMBus
- SPI / QSPI
- GPIO
- PCIe

The FPGA presents a consistent register map or API to the BMC, abstracting away hardware differences. Each management protocol (I2C, SPI, UART, SGPIO, JTAG, and PECI) is instantiated as a soft IP controller in the FPGA. A platform with 12 I2C buses, 4 SPI channels, and 96 GPIOs simply instantiates those exact quantities in the register-transfer level (RTL) design. Adding or removing controllers for a new platform variant is an RTL change and bitstream rebuild, not a silicon re-spin. This eliminates the external mux and expander chips that ASIC-based designs require when they run out of native interfaces.

Dynamic Configuration

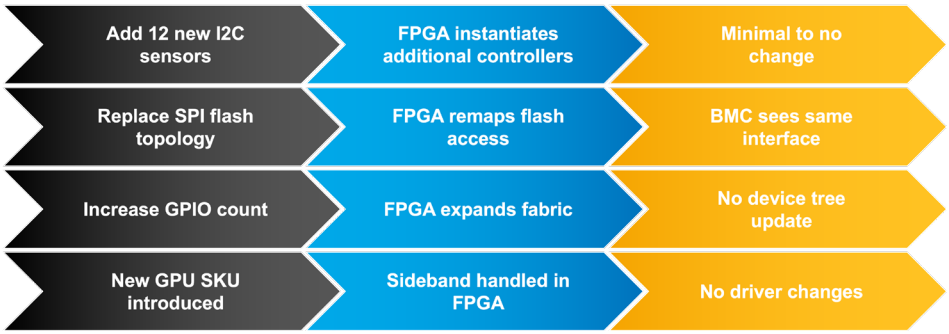
When hardware endpoints change, updates are applied within the FPGA abstraction layer rather than in the BSP.

- A new FPGA bitstream is loaded.
- Minimal or no BSP changes result from abstracting interface variation into FPGA logic.
- The BMC continues to run the same unified software image.

Platform variations at the interface level are handled in programmable logic as shown in Figure 2.

- Addition of I2C devices → controller instantiation + address mapping
- SPI topology changes → FPGA remapping
- GPIO changes → FPGA fabric update
- PCIe variation → sideband handled in FPGA

Figure 2: Platform Variations at the Interface Level



Unified BSP Model

- One BSP supports multiple boards and SKUs.
- Hardware-specific logic resides entirely in FPGA configuration.
- Software maintenance and validation are simplified.

Hardware Security Architecture

Security functions in the FPGA include:

- Secure Boot sequencer validating firmware images before execution.
- PFR state machine per NIST SP 800-193 protecting, detecting, and recovering platform firmware.
- SHA-256/384/512 and AES-256 hardware accelerators for attestation and encrypted communication.
- Post-quantum cryptographic primitives deployable via bitstream update as standards finalize.

Because these functions are implemented in FPGA fabric, rather than as firmware routines on a fixed processor, they provide stronger isolation and tamper resistance than software-only security on an ASIC-based BMC.

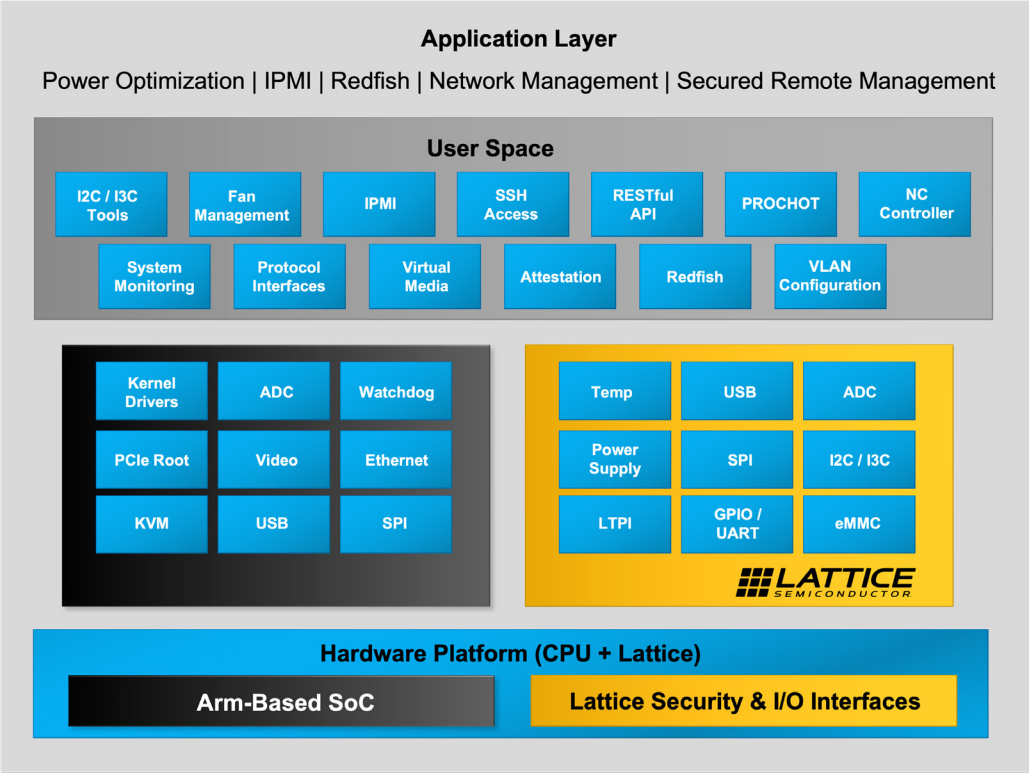
Platform Firmware Resilience

The FPGA acts as the platform's hardware Root of Trust and sits in the boot path of all critical firmware components, including BIOS/UEFI and the OpenBMC firmware itself. The PFR engine within the FPGA verifies cryptographic signatures before allowing firmware to execute, monitors firmware regions at runtime for unexpected modification, and can recover from corruption by restoring known-good images from protected storage.

Layered BMC and System Management Architecture

Figure 3 illustrates a layered BMC and system management architecture based on an example solution from an Arm CPU and Lattice collaboration. The user space hosts management applications and services such as remote management, thermal and power monitoring, NIC and GPU control, and Redfish and Intelligent Platform Management Interface (IPMI) implementations accessed via Representational State Transfer (REST) and Secure Shell (SSH). The kernel/driver layer provides hardware abstraction for interfaces such as PCIe, I2C / I3C, UART, Ethernet, and watchdogs, while the real-time domain aggregates sensor data for temperature, power supplies, fans, LEDs, and low-level I/O. At the bottom, the hardware layer comprises an Arm SoC with security and I/O expander FPGAs, supported by a Secure Boot chain through boot read-only memory (ROM) and secure download mechanisms.

Figure 3: Layered BMC and System Management Architecture



Benefits of the FPGA-Based Flexible BMC Architecture

Technical Benefits

- A single BSP across multiple platforms with reduced dependency on hardware interface topology
- Reduced BSP size and complexity
- Simplified endpoint integration through FPGA-based abstraction of I2C, SPI, GPIO, and PCIe
- Improved isolation between hardware and software
- Hardware changes are isolated within the FPGA configuration, reducing the scope of system-level validation

Business Benefits

- Faster time to market
- Hardware changes require minimal or no BSP revalidation cycles
- Reduced validation effort by focusing on FPGA configuration rather than full BSP regression
- Fewer BSP variants to develop, maintain, and support
- Greater platform scalability across AI, enterprise, and modular platforms

Conclusion

The era of one-size-fits-all BMC ASICs is giving way to more flexible approaches. By introducing FPGA-based abstraction, hardware variation is managed in programmable logic rather than through repeated software modification. This improves BSP reuse, reduces system-level validation overhead, and enables faster platform iteration.

This approach delivers advantages across design flexibility, time to market, security posture, and total platform cost. Lattice FPGAs can work with any BMC processing element. For organizations building next-generation AI, hyperscale, enterprise, or edge server platforms, FPGA-based BMC is more than an architectural choice. It is a strategic shift from consuming management silicon to owning it.



READY TO LEARN MORE?

To learn more about Lattice low power FPGA-based solutions for compute, communications, industrial, and embedded applications, visit www.latticesemi.com or contact us at www.latticesemi.com/contact or www.latticesemi.com/buy.

TECHNICAL SUPPORT ASSISTANCE

Submit a technical support case through **www.latticesemi.com/techsupport**.
For frequently asked questions, please refer to the Lattice Answer Database at **www.latticesemi.com/Support/AnswerDatabase.**

© 2026 Lattice Semiconductor Corporation and affiliates. All rights reserved. Lattice Semiconductor, the Lattice Semiconductor logo, Lattice Nexus, and Lattice Avant are trademarks and/or registered trademarks of Lattice Semiconductor and affiliates in the U.S. and other countries. Other company and product names may be trademarks of the respective owners with which they are associated.

V.2 WP0054