

Design Technology

"Executable Interrupt Specifications: Managing Control-Plane Complexity in Disaggregated Server SoCs"

Modern chiplet-based server SoCs often contain hundreds of interrupt sources distributed across numerous IPs and chiplets. Capturing interrupt source definitions, routing paths, aggregation structures, interrupt-controller mappings, and servicing ownership in a consistent and maintainable form has become increasingly challenging as systems scale in complexity. Maintaining alignment between architecture specifications, firmware implementations, verification environments, and evolving design updates requires significant manual effort and is prone to inconsistencies.

Traditional interrupt specifications are typically represented using spreadsheets, diagrams, and textual documentation. While these artifacts describe interrupt behavior, they often create multiple disconnected representations of the same architectural intent. As designs evolve, maintaining consistency across architecture documents, firmware implementations, verification environments, and review collateral becomes difficult and error prone. Answering questions related to interrupt ownership, routing paths, aggregation behavior, servicing responsibilities, and system-level connectivity frequently requires manual analysis across multiple information sources.

This paper presents an executable interrupt specification methodology that captures interrupt architecture as machine-readable design intent. Hierarchical YAML-based specifications describe interrupt semantics, attributes, ownership, and handling requirements at the IP level, while SoC-level specifications capture interrupt connectivity, routing structures, aggregation hierarchies, interrupt-controller mappings, and servicing ownership across chiplets.

An automated Single Source of Truth (SSOT) framework validates distributed specifications, constructs a systemlevel interrupt model, and generates interrupt maps,

firmware collateral, verification artifacts, and other derived outputs from a common representation of architectural intent. By deriving these artifacts from the same model, the framework reduces specification drift and simplifies maintenance as architectures evolve.

The methodology improves consistency, traceability, and reviewability while reducing manual synchronization across architecture, firmware, and verification workflows. The generated system model enables rapid analysis of interrupt relationships, supports architecture reviews, simplifies impact assessment, and improves visibility into system-level interrupt connectivity.

The framework also enables AI-assisted engineering workflows by providing structured architectural data for specification authoring, design reviews, debugging, and knowledge retrieval. Although demonstrated using interrupt architectures, the methodology readily extends to other machine-readable SoC specifications, including Reliability, Availability, and Serviceability (RAS) definitions and Performance Monitoring Unit (PMU) infrastructure. The approach illustrates how executable specifications can improve the communication, validation, and maintenance of architectural intent in increasingly complex server SoCs.

- Sindhu / Amireddy / **Microsoft**