RISC-V Server SoC Specification

Server SoC Task Group

Version v, 2023-08-14:

Table of Contents

Preamble

This document is in the Development state

Assume everything can change. This draft specification will change before being accepted as standard, so implementations made to this draft specification will likely not conform to the future standard.

Copyright and license information

Contributors

This RISC-V specification has been contributed to directly or indirectly by (in alphabetical order):

Aaron Durbin, Andrei Warkentin, Beeman Strong, Cameron McNairy, Greg Favor, Heinrich Schuchardt, Isaac Chute, Jon Masters, Ken Dockser, Krste Asanovic, Manu Gulati, Mark Hayter, Michael Klinglesmith, Paul Walmsley, Shubu Mukherjee, Sibaranjan Pattnayak, Ved Shanbhogue

Chapter 1. Introduction

The RISC-V server SoC specification defines a standardized set of capabilities that portable system software such as operating systems and hypervisors can rely on being present in a RISC-V server SoC.

A server is a computing system that is designed to manage and distribute resources, services, and data to other computers or devices on a network. It is often referred to as a "server" because it serves or provides information or resources upon request. Such computing systems are designed to operate continually and have a higher degree of requirements on capabilities such as RAS, security, performance, and quality of service. Examples of servers include web servers, file servers, database servers, mail servers, game servers, etc. The focus of this specification is to define requirements for general purpose server computing systems that may be used for one or more of these purposes.

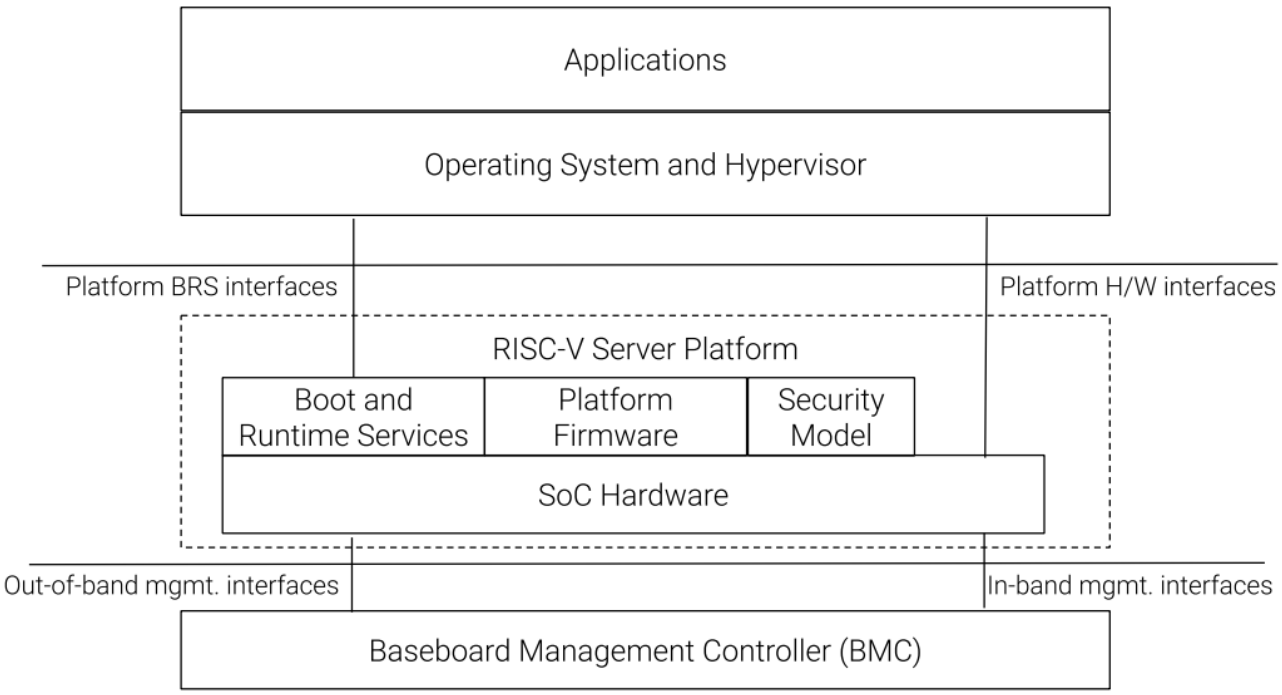

Figure 1. Components of a RISC-V Server Platform

The RISC-V server platform is defined as the collection of SoC hardware, platform firmware, and the boot/runtime services, and security services. The platform provides the hardware interfaces (e.g., harts, timers, interrupt controllers, PCIe root ports, etc.) to portable system software. The platform provides a set of standardized boot and runtime services based on the UEFI and ACPI standards. To support provisioning and management of the platform, it interfaces with a baseboard management controller (BMC) through in-band and out-of-band (OOB) management interfaces. The in-band management interfaces support use of standard manageability specifications like DMTF SPDM and DMTF Redfish for provisioning and management of the operating system executing on the platform. The OOB interface supports use of standard manageability specifications like DMTF Redfish and IPMI for functions such as power/management, telemetry, debug, and provisioning. The security model includes guidelines and requirements such as debug authorization, secure/measured boot, firmware updates, firmware resilience, and confidential computing among others.

The platform firmware, typically operating at privilege level M, is considered part of the platform and is typically expected to be customized and tailored to cater to the requirements of the SoC hardware (e.g., initialization of address decoders, memory controllers, RAS, etc.).

This specification standardizes the requirements for the hardware interfaces and capabilities (e.g., harts, timers, interrupt controllers, PCIe root complexes, RAS, QoS, in-band management, out-of-band (OOB) management, etc.) provided by the SoC to software executing on the application processor harts at privilege level less than M, and enables OS and hypervisor vendors to support such SoCs with a single binary OS image distribution model. The requirements posed by this specification are a standard set of infrastructural capabilities i.e., areas where it is usually not required to be divergent and are not otherwise novel from one implementation to another.

To be compliant with this specification, the SoC MUST support all mandatory requirements and MUST support the listed versions of the specifications. This standard set of capabilities MAY be extended by a specific implementation with additional standard or custom capabilities; including compatible later versions of listed standard specifications. Portable system software MUST support the specified mandatory capabilities to be compliant with this specification.

The requirements in this specification use the following format:

ID#	**Requirement**
CAT_NNN	The `CAT` is a category prefix that logically groups the requirements and is followed by 3 digits - `NNN` - assigning a numeric ID to the requirement. The requirements use the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" that are to be interpreted as described in RFC 2119 [1].
A requirement or a group of requirements may be followed with non-normative text providing context or justification for the requirement. The non-normative text may also be used to point to references that are the source of the requirement.	

This specification groups the requirements in the following broad categories:

- Harts
- Timers
- Interrupts
- IOMMU
- SoC integrated devices
- PCIe subsystem
- RAS
- Quality of Service
- Debug
- Trace
- Performance monitoring

- Security

1.1. Glossary

Most terminology has the standard RISC-V meaning. This table captures other terms used in the document. Terms in the document prefixed by “PCIe” have the meaning defined in the PCI Express (PCIe) Base Specification cite:[PCI} (even if they are not in this table).

Table 1. Terms and definitions

Term	Definition
ACPI	Advanced Configuration and Power Interface [2].
ACS	Follows PCI Express. Access Control Services. A set of capabilities used to provide controls over routing of PCIe transactions.
AER	Advanced Error Reporting. Follows PCI Express. A PCIe defined error reporting paradigm.
AIA	RISC-V Advanced Interrupt Architecture.
ATS	Follows PCI Express. Address Translation Services.
BAR or Base Address Register	Follows PCI Express. A register that is used by hardware to show the amount of system memory needed by a PCIe function and used by system software to set the base address of the allocated space.
BMC	Baseboard Management Controller. A motherboard resident management controller that provides functions for platform management.
CXL	Compute Express Link bus standard.
DMA	Direct Memory Access.
DMTF	Distributed Management Task Force. Industry association for promoting systems management and interoperability.
ECAM	Follows PCI Express. Enhanced Configuration Access Method. A mechanism to allow addressing of Configuration Registers for PCIe functions. In addition to the PCI Express Base Specification, see the detailed requirements in this document.
EP, EP=1	Follows PCI Express. Also called Data Poisoning. EP is an error flag that accompanies data in some PCIe transactions to indicate the data is known to contain an error. Defined in PCI Express Base Specification 6.0 section 2.7.2. Unless otherwise blocked, the poison associated with the data must continue to propagate in the SoC internal interconnect.
GPA	Guest Physical Address: An address in the virtualized physical memory space of a virtual machine.
Guest	Software in a virtual machine.
Hierarchy ID or Segment ID	Follows PCI Express. An identifier of a PCIe Hierarchy within which the Requester IDs are unique.

Term	Definition
Host bridge	Part of a SoC that connects host CPUs and memory to PCIe root ports, RCiEP, and non-PCIe devices integrated in the SoC. The host Bridge is placed between the device(s) and the platform interconnect to process DMA transactions. IO Devices may perform DMA transactions using IO Virtual Addresses (VA, GVA or GPA). The host bridge invokes the associated IOMMU to translate the IOVA to Supervisor Physical Addresses (SPA).
HPM	Hardware Performance Monitor.
Hypervisor	Software entity that controls virtualization.
ID	Identifier.
IMSIC	Incoming Message-signaled Interrupt Controller.
IO Bridge	See host bridge.
IOVA	I/O Virtual Address: Virtual address for DMA by devices.
MCTP	Follows DMTF Standard. Management Component Transport Protocol used for communication between components of a platform management system.
MSI	Message Signaled Interrupts.
NUMA	Non-uniform memory access.
OS	Operating System.
PASID	Follows PCI Express. Process Address Space Identifier: It identifies the address space of a process. The PASID value is provided in the PASID TLP prefix of the request.
PBMT	Page-Based Memory Types.
PRI	Page Request Interface. Follows PCI Express. A PCIe protocol that enables devices to request OS memory manager services to make pages resident.
RCiEP	Root Complex Integrated Endpoint. Follows PCI Express. An internal peripheral that enumerates and behaves as specified in the PCIe standard.
RCEC	Follows PCI Express. Root Complex Event Collector. A block for collecting errors and PME messages in a standard way from various internal peripherals.
RID or Requester ID	Follows PCI Express. An identifier that uniquely identifies the requester within a PCIe Hierarchy. Needs to be extended with a Hierarchy ID to ensure it is unique across the platform.
Root Complex, RC	Follows PCI Express. Part of the SoC that includes the Host Bridge, Root Port, and RCiEP.
Root Port, RP	Follows PCI Express. A PCIe port in a Root Complex used to map a Hierarchy Domain using a PCI-PCI bridge.

Term	Definition
P2P or peer-to-peer	Follows PCI Express. Transfer of data directly from one device to another. If the devices are under different PCIe Root Ports or are internal to the SoC this may involve data movement across the SoC internal interconnect.
PLDM	Follows DMTF standard. Platform Level Data Model.
PMA	Physical Memory Attributes.
PMP	Physical Memory Protection.
Prefetchable Non-prefetchable	Follows PCI Express. Defines the property of the memory space used by a device. For details see the PCIe Base Specification. Broadly, non-prefetchable space covers any locations where reads have side effects or where writes cannot be merged.
SMBIOS	System Management BIOS.
SoC	System on a chip, also referred as system-on-a-chip and system-on-chip.
SPA	Supervisor Physical Address: Physical address used to to access memory and memory-mapped resources.
SPDM	Follows DMTF Standard. Security Protocols and Data Models. A standard for authentication, attestation and key exchange to assist in providing infrastructure security enablement.
SR-IOV	Follows PCI Express. Single-Root I/O Virtualization.
TLP	Follows PCI Express. Transaction Layer Packet. Defined by Chapter 2 of the PCI Express Base Specification.
QoS	Quality of Service. Quality of Service (QoS) is defined as the minimal end-to-end performance that is guaranteed in advance by a service level agreement (SLA) to a workload.
UEFI	Unified Extensible Firmware Interface. [3]
UR, CA	Follows PCI Express. Error returns to an access made to a PCIe hierarchy.
VM	Virtual Machine.

Chapter 2. Server SoC Hardware Requirements

2.1. RISC-V Harts

A RISC-V SoC includes an RISC-V application processor and may include one or more service processors. The service processors may provide services such as security and power management to software executing on the application processors and may themself implement the RISC-V ISA. The requirements in this section only apply to harts in the application processors of the SoC.

ID#	Requirement
RVA_010	The RISC-V application processor harts in the SoC MUST support the RVA23 ISA profile [4].
The next major release of the profiles is expected to be RVA24 which is still under construction. This specification should be updated to comply with the RVA24 profiles as the profile definition firms up.	
RVA_020	The RISC-V application processor harts in the SoC MUST support the following extensions: • Sv48 • Sv48x4 • Svadu • Sdtrig • Sdext • H • Sscofpmf • Zkr • Ssecorrupt • Ssccfg • Ssctr • Sscrind
Ssccfg, Sscind, and Ssctr are under construction. *Many of these mandated extensions are optional in the RVA23 ISA profile. This requirement is placed here as a placeholder. These mandates may be moved downstream into a platform spec (as a chapter) or as a new ISA profile spec.*	
RVA_030	The ISA extensions and associated CSR field widths implemented by any of the RISC-V application processor harts in the SoC MUST be identical.

ID#	Requirement
The RVA23 profile supports a set of optional extensions. The set of optional extensions implemented by the harts must be identical. Where the extension supports optionality in the form of width of fields (e.g., ASIDLEN, VLEN, allowed vstart values, physical address width, debug triggers, cache-block size, etc.), the implementation of these must also be identical. Identical ISA on all harts allows system software to migrate tasks among the harts without constraints.	
RVA_040	The RISC-V application processor harts in the SoC MAY support different power and performance characteristics but must be otherwise indistinguishable from each other from a software execution viewpoint.
All harts in the SoC being indistinguishable from a software execution viewpoint allows system software to migrate tasks among the harts without constraints.	
RVA_050	The RISC-V application processor hart MUST support: • Single stepping using the step bit in `dcsr` • Debug scratch register 0 (`dscratch0`)
RVA_060	The RISC-V application processor hart MUST support: • At least 6 instruction address match triggers. • At least 6 load/store address match triggers. • At least one icount trigger to support single stepping. • At least one interrupt trigger. • At least one exception trigger. • Trigger filtering using `hcontext`. • Trigger filtering using all VMID encodings supported by the hart. • Trigger filtering using `scontext`. • Trigger filtering using all ASID encodings supported by the hart.
RVA_070	The RISC-V application process MUST support at least 6 hardware performance counters defined by the Zihpm extension in addition to the three counters defined by Zicntr extension.

2.2. Clocks and Timers

ID#	Requirement
CTI_010	The `time` CSR MUST increment at a constant frequency and the count MUST be in units of 1 ns. The frequency at which the CSR provides an updated time value must be at least 100 MHz.
CTI_020	The `time` counter MUST appear to be always on and MUST appear to not lose its count across power states including when the hart is powered off.

2.3. External Interrupt Controllers

This section specifies the requirements on the interrupt controllers used to deliver external interrupts to the RISC-V application processor harts.

ID#	Requirement
IIC_010	The RISC-V Advanced Interrupt Architecture [5] MUST be supported.
IIC_020	External interrupts MUST be signaled to a hart as message-signaled interrupts (MSI).
MSI being memory writes allow simplified implementation of rules such as interrupts following a write from a device must only be observed after the writes have been observed and the read completion generated by a device following an interrupt must only be observed after the interrupt has been observed. *Message Signaled interrupts (MSI) is the preferred interrupt signaling mechanism in PCIe. Supporting MSI to harts allows allows low latency interrupt signaling.*	
IIC_030	The IMSIC MUST implement an interrupt file for S-mode.
IIC_040	The IMSIC MUST support at least 5 VS-mode interrupt files.
Supporting 5 VS-mode interrupt files for a hart allows context switching between up to 5 virtual CPUs (vCPU) on a hart without needing to swap the contents of the interrupt file out to memory. Especially when devices are directly assigned to VMs, swapping out the context of an IMSIC interrupt file may incur longer latencies due to the need to redirect device interrupts to a memory resident interrupt file.	
IIC_050	The S-mode interrupt files MUST support at least 255 interrupt identities.
IIC_060	The VS-mode interrupt files MUST support at least 63 interrupt identities.
IIC_070	The memory regions for IMSIC interrupt files MUST have the following PMAs: • Not cacheable, coherent, I/O region. • Supports 4 byte aligned reads and writes.
IIC_080	If the SoC implements devices that use wire-signaled interrupts then the SoC MUST implement an APLIC as specified by the RISC-V AIA specification and MUST use the APLIC to convert the wire-signaled interrupts into MSIs. If implemented, the APLIC MUST support: • Supervisor interrupt domain. • `GEILEN` equal to that implemented by the harts. • MSI delivery mode. • Extempore MSI generation using the `genmsi` register.
_SoC devices using wire-signaled interrupts must implement the rules related to ordering of interrupts vs. older read/writes from devices as specified by the device and/or bus interface specifications that such devices conform to. See also SID_006.	

2.4. Input-Output Memory Management Unit (IOMMU)

ID#	Requirement
IOM_010	All IOMMUs in the SoC MUST support the RISC-V IOMMU specification [6].
The number of IOMMUs implemented in the SoC is UNSPECIFIED.	
IOM_020	All DMA capable peripherals (RCiEP and non-PCIe devices) and all PCIe root ports that are made available to software on the RISC-V application processor harts MUST be governed by an IOMMU. This requirement does not apply to platform devices such as the APLIC or the IOMMU itself. This requirement does not apply to memory accesses originated by a debug module using a System Bus Access block.
DMA capable peripherals being governed by an IOMMU allows OS/hypervisors to restrict DMA originating from such devices to a subset of memory to enhance security and software fault tolerance. The address translation capability provided by the IOMMU enables usages such as passthrough of such devices to virtual machines, shared virtual addressing, etc.	
IOM_030	The IOMMU governing a PCIe root port MUST support at least 16-bit wide device IDs.
IOM_040	An IOMMU that does not govern a PCIe root port MUST support a device ID width required to support all requester IDs originated by the devices governed by that IOMMU.
IOM_050	The IOMMU MUST implement all the page based virtual memory system modes and extensions that are supported by the IOMMU and are also implemented by the RISC-V application processor harts in the SoC.
The page based virtual memory system modes that may be optionally supported by the IOMMU are defined in the IOMMU capabilities register.	
IOM_060	The IOMMU SHOULD support the following virtual memory extensions: • Svadu (enumerated by 1 setting of `capabilities.AMO_HWAD`)
Hardware A/D bit updates capability enables efficient support for use models such as VM migration, shared virtual addressing, and user space work submission.	
IOM_070	The IOMMU SHOULD support pass-through mode and MRIF mode MSI address translation.
IOM_080	When MRIF mode MSI address translation is supported, the IOMMU MUST support atomic updates to the MRIF (enumerated by 1 setting of `capabilities.AMO_MRIF`).
IOM_090	IOMMU governing PCIe root ports SHOULD support PCIe address translation services (ATS).
High performance devices such as DPU/SmartNICs, GPUs, and FPGAs used in server platforms rely on ATS and Page Request services to deliver high throughput and low latency I/O. *Supporting ATS is also required for efficiently supporting usage models such as Shared Virtual Addressing and direct work submission from user mode.*	

ID#	Requirement
IOM_100	IOMMU governing PCIe root ports SHOULD support the T2GPA mode of operation with ATS if ATS is supported.
IOM_110	IOMMU governing RCiEP MUST support PCIe address translation services (ATS) if any of the RCiEP governed by the IOMMU support the ATS capability.
The T2GPA control enables a hypervisor to contain DMA from a device, even if the device misuses the ATS capability and attempts to access memory that is not explicitly authorized by the page tables governing that devices memory accesses. The threat model may also include a man-in-the-middle on the PCIe link inserting ATS translated requests to access memory that was not previously authorized. As an alternative to setting T2GPA to 1, the hypervisor may establish a trust relationship with the device if authentication protocols are supported by the device. For PCIe, for example, the PCIe component measurement and authentication (CMA) capability provides a mechanism to verify the device's configuration and firmware/executable (Measurement) and hardware identities (Authentication) to establish such a trust relationship and the PCIe link may be integrity protected using PCIe integrity and data encryption (IDE).	
IOM_120	IOMMU governing RCiEP MAY support the T2GPA mode of operation with ATS if ATS is supported.
The threats associated with misuse of ATS or malicious insertion of ATS translated requests by a man-in-the-middle may not be present with RCiEP being integrated in the SoC.	
IOM_130	IOMMU MUST support MSI and MAY support wire-signaled interrupts for external interrupts originated by the IOMMU itself.
IOM_140	IOMMU MUST support little-endian memory access to its in-memory data structures.
IOM_150	IOMMU MAY support big-endian mode memory access to its in-memory data structures.
The IOMMU memory-mapped registers always have a little-endian byte order.	
IOM_160	IOMMU MAY support the PASID capability.
IOM_170	IOMMU that supports PASID capability MUST support 20-bit PASID width and MAY support 8 and 17 bit PASID width.
PCIe specification strongly recommends that hardware implement the maximum width of 20 bits to ensure interoperability with system software. See also the implementation note on PASID width homogeneity in the PCIe specification 6.0 section 6.20.2.2.	
IOM_180	IOMMU SHOULD support a hardware performance monitor (HPM).
The HPM is a valuable tool for system integrators for performance monitoring and optimizations. An IOMMU is highly recommended to provide a HPM.	
IOM_190	IOMMU that supports a HPM, MUST support the cycles counter and at least 4 event counters.
IOM_200	The cycles counter and the event counters MUST be at least 40-bit wide.
IOM_210	The IOMMU SHOULD support the software debug capabilities enumerated by `capabilities.DBG`.

ID#	Requirement
IOM_220	The physical address width supported by the IOMMU MUST be greater than or equal to the physical address width supported by the RISC-V application processor harts in the SoC.
The physical address width being greater than or equal to the width supported the harts in the SoC enables use of all addressable memory for I/O and enables sharing of page tables between the hart MMU and the IOMMU.	
IOM_230	The reset default of the `iommu_mode` MUST be `Off`.
The IOMMU disallowing DMA unconditionally following reset due to the mode being Off allows the SoC firmware and software to enable DMA when suitable security protections as required have been established. The IOMMU mode being Off at reset does not pose a significant issue to SoC firmware that needs to employ DMA (e.g., for firmware loading) as that firmware may program the mode in the appropriate IOMMU prior to programming the peripheral governed by that IOMMU to perform a DMA.	
IOM_240	The IOMMU SHOULD be implemented as a RCiEP with base class 08H and subclass 06H [7].
The base class 08H and sub-class 06H are designated by PCIe for use by an IOMMU. Implementing the IOMMU as a PCIe device allows an operating system to determine a driver for the IOMMU and to assign resources such as interrupt vectors to the IOMMU in a PCIe compatible manner.	
IOM_250	The host bridge MUST enforce the physical memory attribute checks and physical memory protection checks on memory accesses originated by the IOMMU and signal detected access violations to the IOMMU.
These checks are analogous to the PMA and PMP checks performed by the RISC-V hart. The host bridge (also known as IO bridge) invokes the IOMMU for address translations. To perform the operations requested by the host bridge the IOMMU may need to access in-memory data structures such as the device directory table and page tables.	
IOM_260	An IOMMU MUST support 24-bit device IDs if the IOMMU governs multiple PCIe root ports that may be part of different PCIe hierarchies.
An IOMMU governing PCIe root ports uses requester ID (RID) - the tuple of bus/device/function numbers (or just bus/function numbers, if the PCIe ARI option is used) - to locate a device context to use for address translation and protection. The 16-bit RID uniquely identifies a requester within a hierarchy. This RID needs to be augmented with the Hierarchy ID (also known as segment ID) - a 8-bit number - to uniquely identify a requester across PCIe hierarchies.	
IOM_270	The host bridge MUST provide the PCIe RID as the bits 15:0 of the device_id to the IOMMU for requests from PCIe EPs and RCiEP.
IOM_280	When the IOMMU supports 24-bit device IDs, the host bridge MUST specify the segment number associated with the PCIe hierarchy from which requests were received as the bits 23:16 of the device_id to the IOMMU.

ID#	Requirement
IOM_290	The determination of device_id input to an IOMMU for requests originating from non-PCIe devices is `UNSPECIFIED`. If PCIe and non-PCIe endpoints/RCiEP are governed by the same IOMMU, the SoC MUST ensure that there is no overlap between device_id associated with non-PCIe devices with the device_id formed using the PCIe RID (and if applicable the segment ID).

2.5. PCIe Subsystem Integration

A PCIe subsystem consists of a root complex with a collection of root ports, root complex event collectors (RCECs), root complex register blocks (RCRBs), and root complex integrated end points (RCiEPs). The root complex implements a host bridge to connect the PCIe root ports, RCECs, RCRBs, and RCiEP, to the CPU and system memory in the SoC through an interconnect.

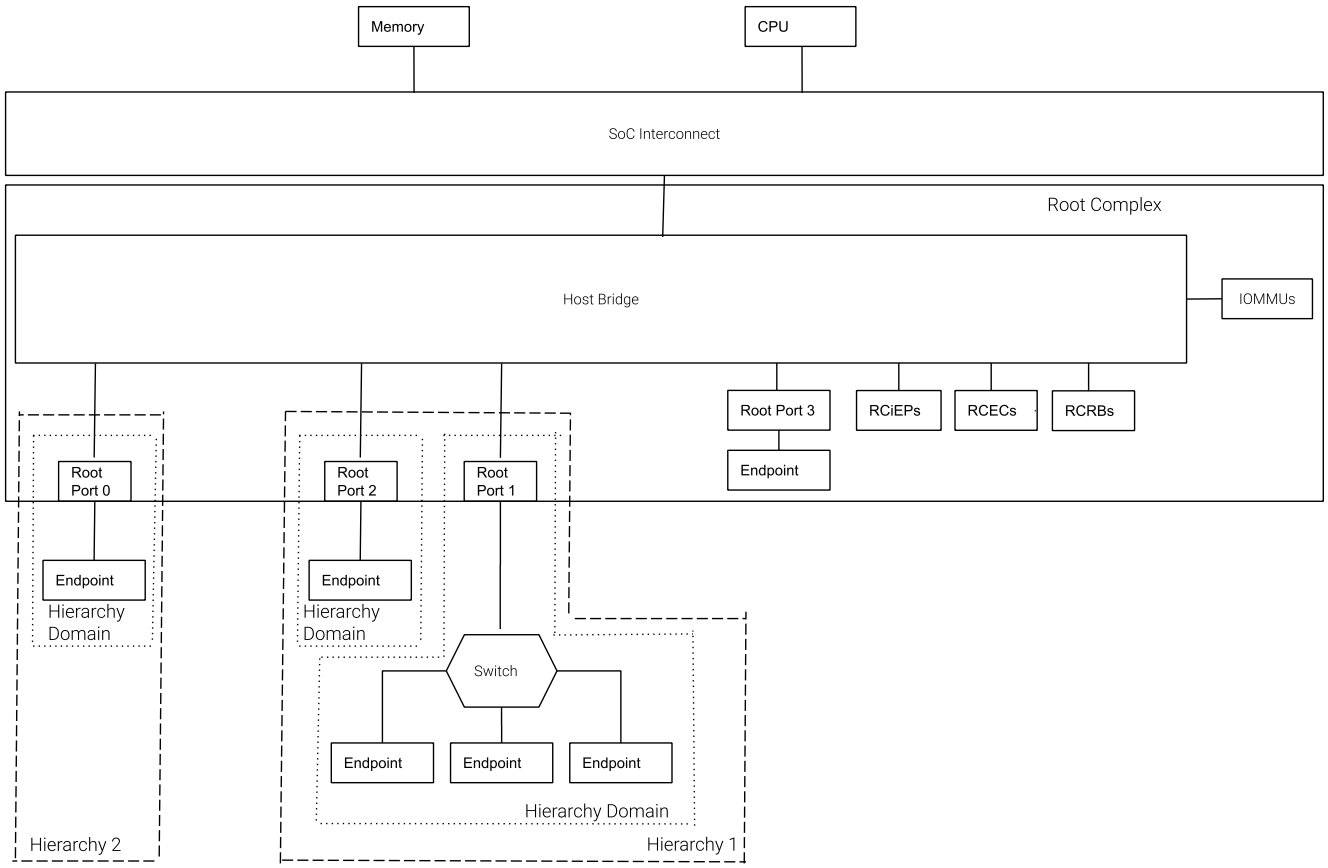

Figure 2. PCIe root complex

One or more root ports in a root complex may be part of a hierarchy where a hierarchy is a PCI Express I/O interconnect topology, wherein the Configuration Space addresses, referred to as the tuple of Bus/Device/Function Numbers (or just Bus/Function Numbers, for PCIe ARI cases), are unique. These addresses are used for Configuration Request routing, Completion routing, some Message routing, and for other purposes. In some contexts a Hierarchy is also called a Segment, and in Flit Mode, the Segment number is sometimes also included in the ID of a Function. Each root port in a hierarchy originates a hierarchy domain i.e. a part of a Hierarchy originating from a single

Root Port. The root ports are PCI-PCI bridges that bridge a primary PCIe bus to a range of secondary and subordinate buses.

In some SoCs, PCIe devices may be integrated in the same package/die as the root complex. Examples of such devices are network controllers, USB host controllers, NVMe controllers, AHCI controllers, etc. Such SoC integrated devices may be presented to software using one of the following options:

1. Presented to software as a PCIe endpoint (EP; See section 1.3.2.2 of the PCIe 6.0 specification) connected to a PCIe root port (See example of such an endpoint connected to root port 3 in [RISC-V-Server-RC]). Such PCIe endpoints must comply with the PCIe specified rules for endpoints.
2. Presented to software as a root complex integrated endpoint (RCiEP; See section 1.3.2.3 of the PCIe 6.0 specification). Such PCIe endpoints must comply with the PCIe specified rules for RCiEP.

Implementing integrated devices that perform as RCiEP or EP allows the use of standardized PCIe frameworks for memory and interrupt resource allocation, virtualization (SR-IOV), ATS/PRI for shared virtual addressing, trusted IO using SPDM/TDISP, RAS frameworks like data poisoning and AER, power management, etc.

The host Bridge is placed between the device(s) and the system interconnect to process DMA transactions. Devices perform DMA transactions using IO Virtual Addresses (VA, GVA or GPA). The host bridge invokes the associated IOMMU to translate the IOVA to Supervisor Physical Addresses (SPA).

2.5.1. Enhanced Configuration Access Method (ECAM)

Each PCIe endpoint and the PCIe root port itself implement a set of memory mapped configuration registers that are accessed using the PCIe enhanced configuration access method (ECAM). The memory mapped ECAM address range for a hierarchy is up to 256 MiB in size and the base address of the range is naturally aligned to the size. Each PCIe function is associated with a 4 KiB page in this range such that the address bits (20+b):20 where b=0 to 7 identify the bus number of that function (see also recommendations in the PCIe specification 6.0 section 7.2.2), the address bits 19:15 identify the device number, and the address bits 14:12 identify the function number. The host bridge in conjunction with the SoC firmware maps the ECAM address range to the hierarchy domain originating at each PCIe root port.

ID#	Requirement
ECM_010	The ECAM address ranges MUST have the following physical memory attributes (PMAs): • Not cacheable, non-idempotent, coherent, strongly-ordered (I/O ordering) I/O region • One, two, and four byte naturally aligned read and write MUST be supported.
ECM_020	Writes to the ECAM address range from a RISC-V hart MUST be non-posted and the write MUST complete at the hart only after a completion is received from the function hosting the accessed configuration register.

ID#	Requirement
Besides performing a write, software executing on a hart must not require any additional actions to achieve this property. *This requirement satisfies the processor and host bridge implementation requirement mentioned in the "Ordering Considerations for the Enhanced Configuration Access Mechanism" implementation note of the PCIe 6.0 specification.*	
ECM_030	The ECAM address range for a hierarchy MUST be contiguous and the base address of the range must be naturally aligned to the size of the ECAM address range associated with the hierarchy.
ECM_040	A SoC MAY support multiple hierarchies. When multiple hierarchies are supported, the ECAM address range of the hierarchies MUST not overlap and but are not required to be contiguous.
ECM_050	The configuration space of the PCIe root ports MUST be associated with the primary bus number of the hierarchy associated with the root port.
PCIe root ports are PCI-PCI bridges that bridge the primary bus to the secondary/subordinate buses. The root port itself enumerates as a PCI-PCI bridge device on the primary bus. The collection of primary, secondary, and subordinate buses are part of a single hierarchy domain that originates at that PCIe root port.	
ECM_060	The configuration space of functions on the primary bus must be accessible irrespective of the state of the corresponding PCIe link.
Discovery and activation of the PCIe link requires accessing the configuration space registers of the PCIe root port itself and the PCIe root port is a PCI-PCI bridge device on the primary bus.	
ECM_070	The PCIe root port MUST enumerate as a PCI-PCI bridge to software and comply with the rules specified for PCIe root ports in PCIe specification 6.0.
ECM_080	The PCIe root port MUST support the PCIe Configuration RRS software (CRS) visibility enable control.
The number of times a configuration request is retried on an RRS response is `UNSPECIFIED`*.*	
ECM_090	Read and/or write to the ECAM range of the hierarchy domain originating at a root port MUST generate PCIe configuration transactions as type 0 or type 1 configuration transactions following the rules specified for ECAM in PCIe specification 6.0.

ID#	Requirement
ECM_100	Read access to ECAM address range from a RISC-V hart MUST be responded with all 1s data if any of the following conditions are TRUE: • Access is to non-existent functions on the primary bus of a hierarchy domain. • Accessed bus is not part of any of the hierarchy domains. • An Unsupported Request or Completer Abort response was received. • A completion timeout occurs. • Access targets a function downstream of a root port whose link is not in DL_Active state. • A PCIe RRS response was received and CRS software visibility is not enabled. • PCIe CRS software visibility is enabled, but the access does not target the vendor ID register, and a RRS response was received on each retry of the configuration read.
The data response to the Vendor ID register on receipt of a RRS response must follow the PCIe defined rules. *See also the recommendations in PCIe specification 6.0 section 2.3.2.*	
ECM_110	Write access from a RISC-V hart to configuration registers of non-existent functions on the primary bus MUST be dropped (silently ignored or discarded) and the write completed. Such accesses MUST not lead to any other behavior (e.g., hangs, deadlocks, etc.).
ECM_120	Poisoned data received from completers (EP=1) MUST be forwarded to the requesting RISC-V hart as poisoned data unless such forwarding is disallowed (e.g., SoC does not support data poisoning or forwarding of poisoned data is disabled though implementation defined means). If forwarding of poisoned data is disallowed then then the poisoned data MUST be replaced with all 1s data.

2.5.2. PCIe Memory Space

ID#	Requirement
MMS_010	The SoC MUST support designating, for each hierarchy domain, one or more ranges of system physical addresses that may be used for mapping memory space of endpoints in that hierarchy domain using 64-bit wide base address registers (BARs) of the endpoint.
MMS_020	SoC MUST support designating, for each hierarchy domain, at least one system physical address range for mapping memory space of endpoints in that hierarchy domain using 32-bit wide BARs of the endpoint.
The ranges suitable for mapping using 32-bit BARs are also sometimes termed as the low MMIO ranges and those suitable for use with 64-bit BARs termed as high MMIO ranges.	

ID#	Requirement
MMS_030	The system physical address ranges designated for mapping endpoint memory spaces MUST be have the following physical memory attributes (PMAs): . Not cacheable, non-idempotent, coherent, strongly-ordered (I/O ordering) I/O region. . MUST support all aligned and unaligned access sizes that can be generated by data requests from any of the RISC-V application processor hart in the SoC or by peer endpoints including RCiEP. . MAY support atomics, instruction fetch, and page walks.
Software may use the Svpbmt extension to override the PMA to NC if such an override is compatible with the restricted programming model of the device.+ *See also the implementation note on optimizations based on restricted programming mode in section 2.3.1 of PCIe specification 6.0.*	
MMS_040	A load from a RISC-V application processor hart to memory ranges designated for mapping memory space of endpoints or RCiEP MUST complete with all 1s response and MUST not lead to any other behavior (e.g., hangs, deadlocks, etc.) if any of the following are TRUE: • Address is not within the any of the following: ◦ Memory base/limit or prefetchable memory base/limit of any root port. ◦ BAR (including when EA capability is used) mapped range of any RCiEP. ◦ BAR (including when EA capability is used) mapped range of any root port. • The PCIe link of the root port to which the access is routed is not active. ◦ Including due to root port entering downstream port containment state. • A UR or a CA response is received from the completer. • A completion timeout occurs.
MMS_050	A store from a RISC-V application processor hart to memory ranges designated for mapping to memory space of endpoints or RCiEP MUST be dropped (silently ignored or discarded) and MUST not lead to any other behavior (e.g., hangs, deadlocks, etc.) if any of the following are TRUE: • Address is not within the any of the following: ◦ Memory base/limit or prefetchable memory base/limit of any root port. ◦ BAR (including when EA capability is used) mapped range of any RCiEP. ◦ BAR (including when EA capability is used) mapped range of any root port. • The PCIe link of the PCie root port to which the access is routed is not active ◦ Including due to root port entering downstream port containment state.
MMS_060	Poisoned data received from completers (EP=1) MUST be forwarded to the requester PCIe device (a RCiEP or an endpoint) as poisoned data unless such forwarding is disallowed (e.g., poisoned TLP egress blocking).

ID#	Requirement
MMS_070	Poisoned data received from completers (EP=1) MUST be forwarded to a requester RISC-V hart as poisoned data unless such forwarding is disallowed through implementation defined means. When such forwarding is disallowed then the poisoned data MUST be replaced with all 1s data.
MMS_080	SoC MUST not use EA capability to indicate memory for allocation to endpoints downstream of a PCIe root port.

2.5.3. Access Control Services (ACS)

The PCIe ACS provides controls on routing of PCIe TLPs. ACS controls may be used to determine whether the TLP should be routed normally, blocked, or redirected. These controls may be applicable to the root complex, switches, multi-function devices, and SR-IOV capable devices.

ID#	Requirement
ACS_010	PCIe root ports and SoC integrated downstream switch ports MUST support the following PCIe access control services (ACS) controls • ACS source validation • ACS translation blocking • ACS I/O request blocking
ACS_020	If a PCIe root port or an SoC integrated downstream switch port implements memory BAR space then it SHOULD support the following PCIe ACS DSP memory target access control.
The ACS DSP memory target access control can be used to prevent unauthorized accesses to protected memory spaces such as the PCIe root ports BAR mapped registers.	
ACS_030	Root ports and any SOC integrated downstream switch ports that support direct routing between root ports or direct routing from ingress to egress port of a root port MUST support the following PCIe ACS controls • ACS P2P request redirect • ACS P2P completion redirect • ACS upstream forwarding • ACS direct translated P2P
ACS_040	Root ports and any SoC integrated downstream switch ports that support direct routing between root ports or direct routing from ingress to egress port of a root port SHOULD support: • ACS P2P Egress control
More commonly P2P routing is accomplished by forwarding the TLP to the host bridge for routing. See also the application note accompanying Fig 2-14 and the section 1.3.1 of PCIe specification 6.0.	
ACS_050	The ACS features including detection, logging, and reporting of ACS violations MUST comply with the rules in the PCIe specifications 6.0.

2.5.4. Address Routed Transactions

The rules in this section apply to treatment in the root complex of TLPs that are routed by address. Addresses carried in such transactions may be the address of a host memory location or the address of a location in the memory space of a peer endpoint or RCiEP.

ID#	Requirement
ADR_010	The host bridge MUST request IOMMU translations for addresses (Translated, Untranslated, or a PCIe ATS address translation request) used in the request by endpoints and RCiEPs.
The IOMMU must be invoked even for Translated requests to allow determination of whether the requester is configured by software to use Translated requests. *When the IOMMU operates in the T2GPA mode, the IOMMU provides GPA as translated addresses in response to Translation requests. In this mode of operation, the IOMMU must be invoked by the host bridge for Translated requests to translate the GPA to a SPA.* *When ACS direct translated P2P controls are enabled the translated requests may not be routed through the host bridge. Software should use the ACS controls to direct such requests to the root complex if direct P2P routing is not desired due to security and/or functional reasons (e.g., when operating in T2GPA mode).*	
ADR_020	The host bridge MUST enforce PMA and PMP checks on the translated address provided by the IOMMU and MUST treat violating requests as Unsupported Requests.
ADR_030	For Translated and Untranslated requests, the host bridge must use the translated addresses provided by the IOMMU to determine whether the transaction is targeting host memory or peer device memory.
ADR_040	The host bridge MAY support access to peer device memory. If peer device memory access is not enabled (by design or by configuration) then such accesses MUST be responded to with an UR/CA response. The host bridge must not cause any other error (e.g., hang, deadlock, etc.) when rejecting access to peer device memory.
_A virtual machine may violate the peer to peer policies and/or configurations established by the hypervisor and/or SoC firmware to disallow peer device memory accesses. Such a VM may attempt to program devices passed through to the virtual machine to perform peer memory accesses. Such attempts to violate the peer to peer policies must not lead to system instabilities (e.g., hangs, deadlocks, etc.) or errors.	
ADR_050	When a posted or non-posted-with-data request is allowed to access peer device memory, then the poisoned data (EP=1) must forwarded as poisoned data unless such forwarding is disallowed (e.g., poisoned TLP egress blocking, data poisoning is not supported by SoC).
ADR_060	Host memory writes caused by posted or non-posted-with-data requests with poisoned data (EP=1) must mark the data in host host memory as poisoned.

ID#	Requirement
ADR_070	Host memory reads that have uncorrectable data errors detected within the SoC must cause a response with poisoned data (EP=1) if transmission of poisoned TLPs is not blocked (see also section 2.7.2.1 of PCIe specification 6.0).

2.5.5. ID Routed Transactions

The rules in this section apply to treatment in the root complex of TLPs that are routed by ID. Such requests may be ID Configuration requests, ID routed messages or completions.

ID#	Requirement
IDR_010	Configuration requests from endpoints and RCiEP MUST be treated as Unsupported Requests.
IDR_020	P2P routing of PCIe VDM between root ports within or across hierarchies SHOULD be supported.
MCTP transport protocols using PCIe VDM are used by the BMC to manage PCIe/CXL devices. These messages are used to support manageability protocols such as PLDM, NVMe-MI, Redfish, etc.	
IDR_030	P2P routing of PCIe VDM to/from RCIeP MAY be supported.

2.5.6. Cacheability and Coherence

ID#	Requirement
CCA_010	The host bridge MUST enforce PCIe memory ordering rules and SHOULD support the relaxed ordering (RO) and ID-based ordering (IDO).
An implementation may sometimes or never allow the relaxations allowed by RO and/or IDO attributes. Such implementations will only lead to a more conservative implementation of the ordering rules but not lead to violation of the ordering rules.	
CCA_020	Writes to host or device memory using the RO attribute set to 0 MUST be observed by other harts and bus mastering devices in the order in which the write was received by the PCIe root port or the host bridge such that all previous writes are globally observed before the RO=0 write is globally observed.
CCS_030	The host bridge MUST enforce the idempotency, coherence, cacheability, and access type PMA of the accessed memory and perform any reordering or combining of PCIe transactions only if the combination of PMA and TLP specified memory ordering attributes allow it.
CCS_040	The host bridge SHOULD implement hardware enforced cache coherency, irrespective of the “No Snoop” attribute in the TLP, unless it has been configured through `UNSPECIFIED` means to not enforce coherency for TLPs with “No Snoop” attribute set to 1.

ID#	Requirement
A PCIe requester is permitted to set the "No Snoop" in transactions it initiates that do not require hardware enforced cache coherency. Host bridges that do not support isochronous VCs or can meet deadlines with hardware enforced coherency may always enforce coherency. Enforcing cache coherency is always conservative and will not lead to data corruption. *Modern systems with integrated memory controllers and snoop directories may not require the use of "No Snoop" to meet the latency targets as memory regions accessed for isochronous operations would usually be device exclusive. PCIe requires a function to guarantee that addresses accessed using "No Snoop" set to 1 are not cached in any of the caches and software that instructs a device to perform "No Snoop" transactions must only do so when it can provide this guarantee.* *Some caches in a SoC may perform clean evictions to memory. In such SoCs, if the addresses used by the non-snooped transactions may be cached (e.g., due to speculative accesses from a hart) then such clean evictions may cause data corruption even if the caches were explicitly cleaned by software using the cache management operations. Software should use memory that has such non-cacheable PMA or use the Svpbmt extension to override the PMA to NC/IO to implement the guarantee required by the PCIe specification with use of "No Snoop" attribute set to 1. If Svpbmt was used to override the PMA then use of cache management operations defined by Zicbom may be required following to flush data that may be already cached.* *See also section 7.5.3.4 of PCIe specification 6.0.*	
CCS_050	The host bridge MUST not violate the coherence PMA if the "No Snoop" attribute in the TLP is 0.
CCS_060	The interpretation of the TLP processing hints (TPH) by the SoC is `UNSPECIFIED`.
A future extension of the RISC-V IOMMU specification may define standard interpretation of the TPH including the use of ATS memory attributes (AMA) for performing cache management.	

2.5.7. Message signaled interrupts

A message signaled interrupt (MSI or MSI-X) is the preferred interrupt signaling mechanism in PCIe.

ID#	Requirement
MSI_010	Message Signaled Interrupts (MSI/MSI-X) MUST be supported.
MSI_020	SoC MUST not require any further action from system software besides configuring the MSI address register in devices with the address of an IMSIC interrupt register file (or a virtual interrupt file) and the MSI data register in devices with an external interrupt identity to enable use of MSI/MSI-X.
MSI_030	SoC MUST not support INTx virtual wire based interrupt signaling.
PCIe supports INTx emulation to support legacy PCI interrupt mechanisms. Modern SoC and devices should not be limited by this emulation mode being not supported.	
MSI_040	PCIe RCiEP, host bridge, RCRB, and root ports MUST support MSI/MSI-X and MUST not use INTx virtual wire interrupt signaling.

2.5.8. Precision Time Measurement (PTM)

ID#	Requirement
PTM_010	PCIe root ports SHOULD support PCIe PTM capability.
Several applications such as instrumentation, media servers, telecom servers, etc. require high precision monitoring and tracking of time. The PCIe PTM protocol supports synchronization of multiple devices/functions to a common shared PTM master time provided by the PTM root.	
PTM_020	When PCIe PTM capability is supported, the SoC MUST make the PTM master time available to software.
Making PTM master time available to software enables software to translate timing information between local time and PTM master time and thereby enable coordination of events across multiple PCIe devices.	
PTM_030	When PCIe PTM capability is supported, the PTM master time MUST be 64-bit wide and MUST use the same or higher resolution clock than the clock used to provide `time`.

2.5.9. Error and Event Reporting

ID#	Requirement
AER_010	PCIe root ports MUST support advanced error reporting (AER) capability for reporting errors from connected devices or the errors detected by the root port itself.
AER capability defines more robust error reporting as compared to the baseline error reporting capability.	
AER_020	PCIe root ports MUST support the downstream port containment (DPC) capability.
AER_030	PCIe root ports MUST support the RP-PIO controls.
The root port programmed I/O (PIO) controls enable fine-grained control over handling of non-posted requests that encounter errors and allows handling of such errors as either uncorrectable or advisory based on policies established by system software.	
AER_040	RCiEP in the SoC SHOULD support the AER capability if it detects any of the errors defined by PCIe specification 6.0 (See section 6.2.7).
AER_050	RCiEP in the SoC MUST support the AER capability if the ACS capability is supported.
AER_060	SoC MUST implement one or more PCIe RCEC in the root complex if any of the RCiEP implement the AER capability or implement PME signaling.
AER_070	Each PCIe RCEC implemented in the SoC MUST implement the RCEC endpoint association extended capability.

2.5.10. Vendor Specific Registers

ID#	Requirement
VSR_010	Vendors specific registers in the root ports, host bridge, RCiEP, and RCRB MUST be implemented using one or more the following capabilities: • Vendor specific capability. • Vendor specific extended capability. • Designated Vendor Specific extended capability.
VSR_020	SoC must not require hypervisors and/or operating system interaction with vendor specific registers not defined by industry standards. Non-standard vendor specific registers, if supported, must only be used by the SoC firmware.
Some Industry standards such a CXL may define standard DVSEC structures in the configuration space. *The preferred way to implement device/SoC vendor specific registers that need to be used by drivers in the run-time environment is to implement them in the memory space of the device. Certain operating systems and hypervisors may disallow and/or require mediating access to the configuration space of devices. See also the implementation note in the PCIe specification 6.0 section 7.2.2.2.*	

2.5.11. SoC Integrated PCIe Devices

ID#	Requirement
SID_010	SoC integrated PCIe devices MUST implement all software visible rules defined by the PCIe specification 6.0 for an EP or RCiEP as applicable.
Implementing integrated devices as RCiEP or EP allows the use of standardized frameworks for memory and interrupt resource allocation, virtualization (SR-IOV), ATS/PRI, shared virtual addressing, trusted IO using SPDM/TDISP, participate in RAS frameworks like data poisoning and AER, power management, etc.	
SID_020	SoC integrated PCIe devices MUST not be legacy devices and MUST not require use of I/O space, I/O transactions, and INTx virtual wire interrupt signaling mechanism.
SID_030	SoC integrated PCIe devices that cache address translations MUST implement the PCIe ATS capability if the address translation cache needs management by the operating system or hypervisors.
SID_040	SoC integrated PCIe devices that support PCIe SR-IOV capability SHOULD support MSI-X capability.
MSI-X capability enables virtual machines to assign interrupt resources to virtual functions without needing access to the configuration space of the function. Access to the configuration space of the virtual function is usually mediated by the hypervisor.	
SID_050	SoC integrated PCIe devices MAY support the PASID capability. When PASID capability is supported, the devices SHOULD support a 20-bit wide PASID.

ID#	Requirement
Endpoints are recommended to support 20-bit wide PASID to ensure interoperability with system software. See also the implementation note on PASID width homogeneity in the PCIe specification 6.0 section 6.20.2.2.	
SID_060	SoC integrated PCIe devices (a multi-function device or a SR-IOV capable device) that support P2P traffic among functions (including among SR-IOV virtual functions) of the device MUST support the following PCIe ACS controls: • ACS P2P request redirect. • ACS P2P completion redirect. • ACS direct translated P2P.
SID_070	SoC integrated PCIe devices MAY support programmable BAR registers. When programmable BAR registers are supported, the Memory Space Indicator (bit 0) of such BAR MUST be 1 and they SHOULD support being mapped anywhere in the 64 bit memory space.
SID_080	RCiEP MAY support the PCIe enhanced allocation (EA) capability for fixed allocation of memory resources.
SID_090	SoC integrated PCIe devices MUST support the PCIe defined baseline error reporting capability and MAY support PCIe Advanced Error Reporting capability. If PCIe ACS controls are supported then the PCIe Advanced Error Reporting capability MUST be supported.
See PCIe specification 6.0 section 7.5.1.1.14.	
SID_100	A RCiEP that supports PCIe Advanced Error Reporting must be associated with a Root Complex Event Collector.

2.6. Reliability, Availability, and Serviceability (RAS)

ID#	Requirement
RAS_010	The level of RAS implemented by the SoC is `UNSPECIFIED`.

ID#	Requirement
The level of RAS implemented by a SoC depends on the reliability goals of the SoC measured using metrics such as failure-in-time (FIT) and defects-per-million (DPM). *_A combination of fault prevention, error detection and correction techniques may be employed. Fault prevention involves use of techniques that reduce or prevent errors, measured through metrics such as defects-per-million (DPM), that may occur after the product has been shipped. These may be accomplished through the use of high quality in product design, technology selection, materials selection, and manufacturing time screening for defects. Through the use of systematic design, technology selection, and manufacturing tests many errors such as those induced by electric fields, temperature stress, switching/coupling noise (e.g. DRAM RowHammer effect), incorrect V/F operating points, insufficient guard bands, metastability, etc. can be prevented.* *The FIT goals may be further expressed in terms of failures attributed to silent data errors (SDE) and those attributed to detected errors that could not be corrected. A SoC compliant with this specification is expected to implement RAS methods to minimize SDE. A reliable SoC should fail instead of silently producing an incorrect result. Insufficient error correction capabilities, while minimizing SDE, contribute to lowered availability as measured by metrics like mean-time-to-failure (MTTF). A SoC should define its FIT targets to minimize SDE and maximize availability to meet its targets.* *This specification recommends implementation of error detection and correction codes for storage elements such as large caches and memories. This specification recommends implementing mechanisms such as periodic scrubbing (also known as patrol scrubbing) to proactively detect and correct errors before such errors accumulate beyond the capability of the implemented error correction codes (e.g., become double bit errors when the code can only correct a single bit error).*	
RAS_020	SoC SHOULD support generating, storing, and forwarding poisoned data. The granularity at which data is poisoned is `UNSPECIFIED`

ID#	Requirement
A component that detects an uncorrected data error may allow possibly corrupted data to propagate to the requester of the data but associate a poison indicator with the data. Such errors are said to be uncorrected deferred errors (UDE) as they allow the component to continue operation and defer dealing with the error to a later point in time if the data corrupted by the error is consumed. If the poisoned data is consumed by a component (e.g. a hart, an IOMMU, a device, etc.) then an uncorrected urgent error (UUE) occurs and a recovery handler is invoked as immediate remedial actions are required and further deferring of the error is not possible. *Data poisoning allows deferring the handling of uncorrected errors to the point of consumption of the corrupted data. Data poisoning allows a more precise determination of the software and/or hardware components affected by the data corruption and thereby allows a recovery action that impacts the smallest possible components. The poisoned data indicator must itself be protected using error detection and correction codes when stored to avoid subsequent errors from causing silent consumption of the corrupted data.* *Support for data poisoning enables support for error containment features implemented by other standards such as PCIe and CXL.* *See also the RISC-V RERI specification for more detailed discussions on treatment of faults and errors.*	
RAS_030	If poisoned data needs to be forwarded from a first component to a second component that is not capable of handling poison then the first component MUST report an uncorrected urgent error instead of propagating the poisoned data.
Some components act as an intermediary through which the data passes through. For example, a PCIe/CXL port is an intermediary component that by itself does not consume the data it receives from memory but forwards the data to the endpoint. In such cases the component may receive the data with a deferred error. Such a component may propagate the error and not log an error by itself. However, if the component to which the data is being propagated (e.g. a PCIe endpoint) is not capable of handling poison then the former component must signal an urgent error instead of propagating the corrupted data, as the act of propagation breaks containment of the error.	
RAS_040	The SoC SHOULD support the RISC-V RAS error record register interface (RERI) [8] for error logging and signaling.
Note RERI is still under construction.	
RAS_050	If RERI is supported then the RAS error records MUST support individually enabling error signals for each error severity (UUE, UDE, or CE) of errors that may be logged in the error record.
Disabling error signaling enables software flexibility of using an event based or polling based error logging for corrected errors and deferred errors. Software may typically operate in event based mode for urgent errors as these require urgent remedial action when they occur.	

ID#	Requirement
RAS_060	If RERI is supported then the RAS error records MUST retain their state of logged error information (status, address, information, supplemental information, and timestamp) across a RAS initiated reset. The RAS error records MAY also retain their state across other types of implementation defined reset. The state of the control register across reset, including RAS initiated reset, is `UNSPECIFIED`.
Some errors may lead a hardware component to enter a failure mode where it cannot service additional requests (colloquially termed as jammed or wedged). In such cases the SoC may need to be reset to restore it to an operational state (a RAS initiated reset). Preserving the RAS error records across such resets allows the SoC firmware and system software to read out these error records at boot following such a reset for logging and analysis. *Some types of reset, such as those initiated with some debug controls, may also retain state of the RAS error records.*	
RAS_070	If RERI is supported then the RAS error records MAY support error record injection to support RAS handler verification.
Verifying correct implementation of RAS handlers is a challenging task as it is not possible to deterministically cause all possible errors in the SoC to verify that the RAS handler implements the desired recovery actions. An unverified RAS handler may fail to perform as desired when the error happens and reduce availability or impact the serviceability of the SoC. Error record injection provides a convenient method for carrying out such verification by allowing a variety of error signatures to be injected and signaled. While such verification may also be performed by using hardware error injection techniques (e.g., providing methods to corrupt an data location protected by an error detection code), it may not be desirable from a security and stability perspective to open these capabilities generally for use by software.	
RAS_080	If RERI is supported then the hardware components in the SoC that support error correction MUST implement a corrected error counter in their error records and MUST support signaling counter overflows.
Signaling on counter overflow allows software to establish a convenient threshold for when such corrected errors are signaled for logging and analysis. *Some hardware units may maintain a history of corrected errors and may report a corrected error and may increment the corrected error counter only if the error is not identical to a previously reported corrected error. Some hardware units may implement low pass filters (e.g., leaky buckets) that throttle the rate at which corrected errors are reported and counted. This requirement applies to the corrected errors that are counted by the error record after the hardware component has made the determination to report and count the error based on such implementation defined filtering rules.*	

2.7. Quality of Service

Quality of Service (QoS) is the minimal end-to-end performance that is guaranteed in advance by a service level agreement (SLA) to an application. QoS capabilities in the SoC provide mechanisms that may be used by system software to control interference to an application and thereby reduce the variability in performance experienced by one application due to other application's cache capacity usage, memory bandwidth usage, interconnect bandwidth usage, power usage, etc.

ID#	Requirement
QOS_010	SoC SHOULD support QoS mechanisms to address undue performance interference caused by one workload to another when they both access a shared resource such as caches and system memory.
QOS_020	SoC SHOULD support the RISC-V capacity and bandwidth controller register interface (CBQRI) [9] for capacity and bandwidth allocation in significant shared caches and memory controllers.
QOS_030	If CBQRI is supported, RISC-V harts of application processors in the SoC MUST support the `sqoscfg` CSR and the CSR must support at least 16 RCIDs and at least 32 MCIDs
The number of RCID and MCID should scale with the number of RISC-V harts in the SoC.	
QOS_040	If CBQRI is supported, the IOMMU in the SoC MUST support the CBQRI defined extension for associating RCID and MCID with device and IOMMU initiated requests.
QOS_050	If CBQRI is supported, the last level cache in the SoC MUST support cache capacity allocation.
QOS_060	If CBQRI is supported, the last level cache in the SoC MUST support the cache capacity usage monitoring.
QOS_070	If CBQRI is supported, the memory controllers in the SoC SHOULD support bandwidth allocation.
QOS_080	If CBQRI is supported, the memory controllers in the SoC SHOULD support bandwidth usage monitoring.
The method implemented by the SoC for bandwidth throttling and control is implementation specific. The implementation is recommended to implement a scheme that leads to no more than +/- 10 % deviation from the target established by system software through the CBQRI interface.	
QOS_090	If CBQRI is supported, the number of RCID and MCID supported by capacity controllers, bandwidth controllers, and all RISC-V application processor harts in the SoC MUST be identical.
Portable system software may restrict itself to supporting the smallest number of RCID and MCID implemented by any of the controllers when the number of RCID and MCID supported by the controllers is not identical to avoid complex allocations and restrictions on workload placement that would be otherwise required to support such asymmetry.	
QOS_100	If CBQRI is supported, the monitoring counters in the capacity and bandwidth controllers MUST be wide enough to not overflow when sampling at a rate of 1 Hz.
For example, a HBM3 memory interface may support data transfers at rate up to 1 TB/s and thus require a 34-bit counter to not cause an overflow when sampled at 1 Hz.	

2.8. Manageability

This section defines the rules for RISC-V server SoC to support a common set of protocols and standards for server management. The use of industry standards specified in this section enables

server platforms compliant with this specification to be integrated into the server management frameworks and tools used by data centers and enterprises

ID#	Requirement
MNG_010	SoC SHOULD support the datacenter-ready secure control module (DC-SCM) specified by the Open Compute Project for server management, security, and control features.
MNG_020	The SoC SHOULD support a x1 PCIe lane (at least Gen 2; preferably Gen 5) connection to the DC-SCM.
This interface is typically connected to a BMC as a PCIe endpoint for usages including enabling host to BMC communication for functions such as video, MCTP over PCIe VDM, memory mapped PCIe device and/or a USB controller. *The in-band network interfaces are used by system software to communicate with the BMC using protocols such as the Redfish host interface. The PCIe interface to the BMC enables the BMC, through the use of SoC routed PCIe VDMs, to use such VDMs to carry MCTP messages to manage platform devices such as network controllers, NVMe controllers, FPGAs, GPUs, etc.*	
MNG_030	The SoC MUST support one of the following transport bindings for MCTP/PLDM protocol for out of band manageability between a baseboard management controller (BMC) and the SoC manageability controller. • I3C as specified by DSP0233 MCTP I3C transport binding 1.0.0. cite:[DSP0233] • PCIe VDM as specified by DSP0238 MCTP PCIe VDM transport binding 1.2.0. cite:[DSP0238]
PCIe VDM and I3C interface supports high bandwidth communication between the BMC and the management controllers in the SoC. MCTP provides a standard transport protocol for manageability protocols such as PLDM and Redfish. The out-of-band interface enables monitoring of sensors (e.g., temperature, power, etc.), control of parameters (e.g, power limits, etc.), and logging (e.g., RAS error records, etc.) by the BMC.	
MNG_040	The SoC SHOULD support Security protocol and data model (SPDM) attestation and the use of SPDM encrypted secure messages for in-band and out-of-band communication with the BMC as specified in: • DSP0274 Security protocol and data model (SPDM) specification 1.2.1. cite:[DSP0274] • DSP0275 SPDM over MCTP binding specification 1.0.1. cite:[DSP0275] • DSP0276 Secured messages using SPDM over MCTP binding specification 1.1.0. cite:[DSP0276] • DSP0277 Secured messages using SPDM specification 1.1.0. cite:[DSP0277]
SPDM authentication protocols support establishing a trust relationship between the in-band and out-of-band manageability agents in the SoC and the BMC. Use of SPDM secured messages enables preserving the confidentiality and integrity of data exchanged between the BMC and manageability agents in the SoC.	

ID#	Requirement
MNG_050	The SoC MUST support remote debug using a JTAG interface with the BMC.
MNG_060	The SoC MUST provide the following interfaces to firmware/OS on the RISC-V application processor for in-band server management. • Network • UART
MNG_070	The SoC MUST support the following DMTF management protocols for in-band and out-of-band manageability: • DSP0236 MCTP base specification 1.3.1. cite:[DSP0236] • DSP0240 PLDM base specification 1.1.0. cite:[DSP0240] • DSP0266 Redfish specification 1.30. cite:[DSP0266] • Intelligent Platform Management Interface (IPMI) 2.0. cite:[IPMI20]
Use of DMTF specified manageability protocols enables simpler integration of RISC-V based server platforms into cloud and enterprise manageability infrastructure.	

2.9. Debug

ID#	Requirement
DBG_010	The SoC MUST support at least one RISC-V debug modules as specified by the RISC-V debug specification.
DBG_020	The debug modules MUST support the following capabilities: • Program buffer to execute instructions in debug mode. • Support at least one halt group and at least one resume group. besides group 0 • Support debugging harts immediately out of reset. • Always perform program buffer and abstract memory access with exact and full set of permissions (i.e., hardwire `relaxedpriv` to 0). • Freezing hart local counters using `stopcount` control.
Ability to halt harts as a group and ability to halt at reset allows ease of debugging system software. *Program buffer provides more flexibility and ease of use compared to abstract access.* *Enforcing a full and exact set of permissions on abstract and program buffer memory access avoids security concerns.*	
DBG_030	The SoC MUST support a JTAG debug transport module as specified by the RISC-V debug specification.

2.10. Trace

ID#	Requirement
TRC_010	The SoC MUST support instruction trace using one of the following standard extensions: • RISC-V E-trace. cite:[ETRACE] • RISC-V N-trace. cite:[NTRACE]
TRC_020	The SoC MUST support the RISC-V trace control interface specification.
N-trace and the trace control interface specification are still under construction.	
TRC_030	The trace control interface MUST support filtering by privilege levels.
TRC_040	The SoC SHOULD support System Memory trace sink.
System memory sink supports self hosted trace decoders for software debug and profiling. The system memory sink also supports conveniently transmitting the trace off-chip through a PCIe or USB port.	

2.11. Performance Monitoring

ID#	Requirement
SPM_010	Significant caches in the SoC SHOULD support a HPM capable of counting: • Cache lookup • Cache miss If the SoC supports NUMA configurations, then the HPM SHOULD support filtering the counting based on whether the request originated in a local node or in a remote node.
SPM_020	The memory controllers in the SoC SHOULD support a HPM capable of counting: • Read bandwidth • Write bandwidth If the SoC supports NUMA configurations, then the HPM SHOULD support filtering the counting based on whether the request originated in a local node or in a remote node.
SPM_030	The PCIe ports in the SoC SHOULD support a HPM capable of counting: • Read bandwidth (from system memory) • Write bandwidth (to system memory)

ID#	Requirement
SPM_040	The SoC SHOULD support a HPM capable of counting the average latency of a read request from a memory requester (e.g., a hart, a PCIe host bridge, etc.) in the SoC. If the SoC supports NUMA configurations, then the HPM SHOULD support filtering the counting based on whether the request is to local memory or to remote memory.
Bandwidth and latency are the most commonly used performance metrics used to guide workload placement and tuning. Bandwidth may be computed as the number of bytes transferred passed the monitoring point over an interval. *The average latency may be computed by summing the number of transactions in a queue on each cycle and dividing by the total number of requests allocated into the queue.*	
SPM_050	The PCIe ports in the SoC SHOULD support the Flit performance measurement extended capability defined by PCIe specification 6.0.

2.12. Security Requirements

ID#	Requirement
SEC_010	The PCIe root ports in the SoC SHOULD support PCIe Integrity and Data Encryption (IDE) capability.
SEC_020	The SoC SHOULD support encryption of off-chip DRAM using a transient memory encryption key that has at least 192-bit key lengths.
Off-chip memory encryption provides protection to critical assets in memory such as credentials, data encryption keys, and other secrets.	
SEC_030	The cryptographic modules used to implement PCIe and off-chip DRAM encryption SHOULD comply with security requirements specified by standards such as FIPS 140-3.

Bibliography

[1] “Key words for use in RFCs to Indicate Requirement Levels.” [Online]. Available: datatracker.ietf.org/doc/html/rfc2119.

[2] “Advanced Configuration and Power Interface (ACPI) Specification.” [Online]. Available: uefi.org/specifications.

[3] “Unified Extensible Firmware Interface.” [Online]. Available: uefi.org/specifications.

[4] “RVA23 Profiles.” [Online]. Available: github.com/riscv/riscv-profiles/blob/main/rva23-profile.adoc.

[5] “RISC-V Advanced Interrupt Architecture.” [Online]. Available: github.com/riscv/riscv-aia.

[6] “RISC-V IOMMU Architecture Specification.” [Online]. Available: github.com/riscv-non-isa/riscv-iommu.

[7] “PCI Code and ID Assignment Specification Revision 1.1.” [Online]. Available: pcisig.com/sites/default/files/files/PCI_Code-ID_r_1_11__v24_Jan_2019.pdf.

[8] “RISC-V RAS error record register interface.” [Online]. Available: github.com/riscv/riscv-ras-eri.

[9] “RISC-V Capacity and Bandwidth QoS Register Interface.” [Online]. Available: github.com/riscv/riscv-cbqri.

[10] “MCTP I3C transport binding 1.0.0.” [Online]. Available: www.dmtf.org/dsp/DSP0233.

[11] “ MCTP PCIe VDM transport binding 1.2.0.” [Online]. Available: www.dmtf.org/dsp/DSP0238.

[12] “Security protocol and data model (SPDM) specification 1.2.1.” [Online]. Available: www.dmtf.org/dsp/DSP0274.

[13] “SPDM over MCTP binding specification 1.0.1.” [Online]. Available: www.dmtf.org/dsp/DSP0275.

[14] “Secured messages using SPDM over MCTP binding specification 1.1.0.” [Online]. Available: www.dmtf.org/dsp/DSP0276.

[15] “Secured messages using SPDM specification 1.1.0.” [Online]. Available: www.dmtf.org/dsp/DSP0277.

[16] “MCTP base specification 1.3.1.” [Online]. Available: www.dmtf.org/dsp/DSP0236.

[17] “PLDM base specification 1.1.0.” [Online]. Available: www.dmtf.org/dsp/DSP0240.

[18] “Redfish specification 1.18.0.” [Online]. Available: www.dmtf.org/dsp/DSP0266.

[19] “Intelligent Platform Management Interface (IPMI) 2.0.” [Online]. Available: www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ipmi-second-gen-interface-spec-v2-rev1-1.pdf.

[20] “RISC-V Processor Trace Specification.” [Online]. Available: github.com/riscv-non-isa/riscv-

trace-spec.

[21] “RISC-V N-Trace Specification.” [Online]. Available: github.com/riscv-non-isa/tg-nexus-trace.