Arm® BSA Architecture Compliance

Revision: r1p1

Test Scenario

Non-Confidential

Issue 0101-08
ARM040-1254092399-18807

Arm BSA Test Scenario Document

Release information

Document history

Issue	Date	Confidentiality	Change
01	26 July 2021	Non-Confidential	Beta release 0.9
02	13 Sep 2021	Non-Confidential	REL 1.0
03	14 Oct 2022	Non-Confidential	REL 1.1
0101-04	29 June 2023	Non-Confidential	REL v1.0.5
0101-05	29 Sep 2023	Non-Confidential	REL 1.0.6
0101-06	20 October 2024	Non-Confidential	REL 1.0.9
0101-07	31 March 2025	Non-Confidential	REL 1.1.0

Non-Confidential Proprietary Notice

Confidentiality Status

Product Status

The information in this document is for a final product, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on [Product Name], create a ticket on **https://support.developer.arm.com**.

To provide feedback on the document, fill the following survey: **https://developer.arm.com**/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change.

This document includes terms that can be offensive. We will replace these terms in a future issue of this document. If you find offensive terms in this document, please email **terms@arm.com**.

Web Address

www.arm.com

Contents

1 Introduction

1.1 Product revision status

The *rmpn* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

rm	Identifies the major revision of the product, for example, r*1*.
pn	Identifies the minor revision or modification status of the product, for example, p*2*.

1.2 Intended audience

This document is for engineers who are verifying an implementation of Arm® Base System Architecture 1.0.

1.3 Conventions

The following subsections describe conventions used in Arm documents.

1.3.1 Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: **https://developer.arm.com/glossary**.

1.3.2 Typographical Conventions

Convention	Use
italic	Introduces citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` **`bold`**	Denotes language keywords when used outside example code.
`monospace` `underline`	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

1.4 Useful resources

This document contains information that is specific to this product. See the following resources for other relevant information.

- Arm Non-Confidential documents are available on **developer.arm.com/documentation**. Each document link in the tables below provides direct access to the online version of the document.
- Arm Confidential documents are available to licensees only through the product package.

Arm products	Document ID	Confidentiality
Arm® Base System Architecture 1.0	DEN0094D	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
Arm® Architecture Reference Manual for A-profile architecture	DDI0487F	Non-Confidential

Non-Arm resources	Document ID	Organization
PCI Express Base Specification Revision 6.0, Version 1.0	NA	PCI-SIG
PCI-To-PCI Bridge Architecture Specification 1.2	NA	PCI-SIG

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at **http://www.adobe.com**.

1.5 Feedback

Arm welcomes feedback on this product and its documentation.

1.5.1 Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

1.5.2 Feedback on content

If you have comments on content, send an email to **support-systemready-acs@arm.com** and give:

- The title Arm Base System Architecture Scenario.
- The number ARM040-1254092399-18807.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.
- Arm also welcomes general suggestions for additions and improvements.

2 Base System Architecture

Base System Architecture (BSA) specifies a hardware system architecture based on Arm 64-bit architecture, that system software such as operating systems, hypervisors, and firmware can rely on. The document addresses PE features and key aspects of system architecture. The primary goal of the document is to ensure sufficient standard system architecture to enable a suitably built single OS image to run on all the hardware compliant with the specifications.

Arm does not mandate compliance with this specification. However, Arm anticipates that OEMs, ODMs, cloud service providers, and software providers will require compliance to maximize Out-of-Box software compatibility and reliability.

2.1 BSA ACS

BSA ACS provides tests for validating the compliance of a platform with BSA specifications. The tests are divided into a hierarchy of subcategories depending on the runtime environment and the component submodules that are required for achieving the verification. The top level of a hierarchy is consistent with the target hardware subsystem which is validated by a test. A test may check for different parameters of the hardware subsystem.

The tests are classified as:

- PE
- Memory Map
- GIC
- SMMU
- Clock and Timer
- Wakeup Semantics
- Power State Semantics
- Watchdog
- Peripherals
- PCIe
- DeviceID Generation and ITS Groups

2.2 PE

Test number	Rule ID	Level	Scenario	Algorithm
1	B_PE_01	L1	All PEs are architecturally symmetric except for the permitted exceptions listed in Section A of BSA specification.	CPU System register read. Read all the processor ID registers from all PEs and then compare values with the main PE.
2	B_PE_02	L1	The number of PEs must not exceed: • Eight, when the interrupt controller is compliant to GICv2. • 2^28 when the interrupt controller is compliant to GICv3 or higher.	Read from ACPI MADT table.
3	B_PE_03	L1	PEs must implement the Advanced SIMD and FEAT_FP extensions.	CPU System register read: ID_AA64PFR0_EL1 must indicate support bits [23:20].
4	B_PE_04	L1	PEs must support 4KB translation granules at stage 1.	ID_AA64MMFR0_EL1 must indicate support for 4KB granules for all cores.
5	B_PE_05	L1	All PEs are coherent and in the same Inner Shareable domain.	Not implemented There is no architected way to know if all PE's are in same inner shareable domain.
6	B_PE_06	L1	Where export restrictions allow, PEs must implement cryptography extension support for FEAT_AES, FEAT_SHA1 and FEAT_SHA256.	CPU System register read: Bits [4:15] of ID_AA64ISAR0_EL1 must be supported
7	B_PE_07	L1	PEs must implement little-endian support.	CPU System register read: SCTLR.EE should be supported.
8	B_PE_08	L1	PEs must implement EL1 and EL0 in the AArch64 Execution state.	ID_AA64PFR0_EL1.EL0 and ID_AA64PFR0_EL1.EL1 should be supported.
9	B_PE_09	L1	PEs must implement the FEAT_PMU extension, and the base system must expose a minimum of four programmable PMU counters to the operating system.	If ID_AA64DFR0_EL1.PMUVer is supported, then PMCR_EL0.N should be greater than three.
10	B_PE_10	L1	The PMU overflow signal from each PE must be wired to a unique PPI interrupt with no intervening logic.	If ID_AA64DFR0_EL1.PMUVer is supported, install ISR and verify PMU overflow interrupt by programming System register.
11	B_PE_11	L1	Each PE must implement a minimum of six breakpoints, two of which must match virtual address, contextID or VMID.	1. Read ID_AA64DFR0_EL1 and check number of breakpoints. 2. Read DBGBCR<n>_EL1 & check type of breakpoint.

Test number	Rule ID	Level	Scenario	Algorithm
12	B_PE_12	L1	Each PE implements a minimum of four synchronous watchpoints.	ID_AA64DFR0_EL1.WRPs must be greater than 2.
13	B_PE_13	L1	PEs must implement the FEAT_CRC32 instructions.	ID_AA64ISAR0_EL1.CRC32 should be supported.
14	B_PE_15	L1	If FEAT_PAuth (Pointer Authentication), mandatory from Armv8.3, is implemented, Arm recommends that the standard algorithm defined by the Arm architecture is implemented for address and generic authentication. • If an alternative algorithm is used, it must be at least as cryptographically strong as the Arm recommended algorithm.	If pointer signing is implemented, check ID_AA64ISAR1_EL1[11:4] for address authentication and ID_AA64ISAR1_EL1[31:24] for generic authentication.
16	B_PE_14	L1	Implementation of SVE is optional. Implementation of SVE2 (FEAT_SVE2) is required for PEs that are based on the Armv9 architecture (see FEAT_SVE2 in [3]).	If the Processor Series is identified as Armv9 from the SMBIOS table, then ID_AA64ZFR0_EL1.SVEver must indicate support for SVE.
17	B_PE_18	L1	PEs must implement Non-secure EL2 in AArch64.	ID_AA64PFR0_EL1.EL2 must be supported.
18	B_PE_19	L1	PEs must support 4KB translation granules at stage 2.	ID_AA64MMFR0_EL1. TGran4_2 must be supported.
19	B_PE_20	L1	The translation granules supported at stage 2 must match those supported at stage 1.	Check AA64MMFR0_EL1 register.
20	B_PE_21	L1	The base system must expose a minimum of two programmable PMU counters to a hypervisor.	Check whether minimum two programmable PMU counters are exposed.
21	B_PE_22	L1	Two of the implemented breakpoints in each PE must be able to match on VMID. See B_PE_11.	Read DBGBCR<n>_EL1 & check type of breakpoint.
22	B_PE_23	L1	PEs must implement EL3 in the AArch64 Execution state.	ID_AA64PFR0_EL1.EL3 must be supported.

Test number	Rule ID	Level	Scenario	Algorithm
63	B_PE_24	L1	PEs must implement Secure state.	1. Read the ID_AA64PFR0_EL1 register. 2. Extract the following fields: RME from bits [55:52] SEL2 from bits [39:36] EL3 from bits [15:12] EL2 from bits [11:8] 3. If RME is zero and EL3 is non-zero and EL2 is non-zero and SEL2 is non-zero, then Pass. 4. If RME is zero and EL3 is non-zero and EL2 is non-zero and SEL2 is zero, then Fail. 5. If RME is zero and EL3 is non-zero and EL2 is zero, then Pass. 6. If RME is non-zero and EL2 is non-zero and SEL2 is non-zero, then Pass. 7. If RME is non-zero and EL2 is non-zero and SEL2 is zero, then Fail. 8. In all other cases, Fail.
15	B_PE_25	FR	All PEs must implement Large System Extensions (LSE) as indicated by ID_AA64ISAR0_EL1.Atomic = 0b0010. See FEAT_LSE in [3].	ID_AA64ISAR0_EL1.Atomic must be supported

2.3 Memory Map

Test number	Rule ID	Level	Scenario	Algorithm
102	B_MEM_01	L1	All memory accesses whether they access memory space that is populated or not, must respond within finite time, to avoid the possibility of system deadlock.	1. Initialize the exception handlers. 2. Get the address range from the system table or memory map. 3. Try to access the memory. 4. Exception may or may not occur, but the test should continue to proceed. 5. Once all memory ranges are accessed, the test is considered as pass.
101	B_MEM_02	L1	Where a memory access is to an unpopulated part of the addressable memory space, accesses must be terminated in a manner that is presented to the PE as either a precise Data Abort, or as a system error interrupt, or an SPI, or LPI interrupt to be delivered to the GIC.	1. Get an unpopulated memory address range 2. Access the unpopulated memory address by performing a write to the address. 3. If an exception is obtained by doing so then the test is considered as PASS.
104	B_MEM_03	L1	All Non-secure on-chip DMA requestors in a base system that are expected to be under the control of the operating system or hypervisor where it addresses all the Non-secure address space.	1. Check if the PCIe device is capable of 64-bit DMA. 2. Else, check if it is present behind an SMMU. 3. Pass if it satisfies one of the above, else fail
106	B_MEM_04	L1	If the Requester goes through an SMMU, then the Requester must be capable of addressing all the Non-secure address space when the SMMU is turned off.	1. Check if the PCIe device is capable of 64-bit DMA if its behind a SMMU.
103	B_MEM_05	L1	All PEs must be able to access all the Non-secure address space.	1. For the current PE, read ID_AA64MMFR0_EL1[3:0] to get max number of bits that PE can access. 2. Check if the maximum accessible memory is accessible by PE.

Test number	Rule ID	Level	Scenario	Algorithm
107	B_MEM_06	L1	Non-secure off-chip devices that cannot directly address all the Non-secure address space must be placed behind a stage 1 SMMU that is compatible with the Arm SMMUv2 or SMMUv3 specification, that has an output address size large enough to address all the Non-secure address space.	Check if non DMA PCIe peripherals devices are behind an SMMU.
-	B_MEM_07	L1	It is possible for the progress of a memory transaction to depend on a second memory access, the system must avoid deadlock if the memory access gets ordered behind the original transaction.	Not implemented. The sequence to generate deadlock is not repeatable in an environment such as UEFI shell.
-	B_MEM_08	L1	The system must provide some memory that is mapped in the Secure address space.	Not implemented. Does not have access to Secure memory map from UEFI shell.

2.4 GIC

Test number	Rule ID	Level	Scenario	Algorithm
201	B_GIC_01	L1	A base system must present operating systems and hypervisors with the interfaces defined by one of the following: • Generic Interrupt Controller (GIC) v2 interrupt controller. • GICv2 interrupt controller with GICv2m extension. • GICv3 interrupt controller.	Check for the version from the platform information. If it is not present in platform information, then read GICD_PIDR2.
202	B_GIC_02	L1	Limitations and valid configurations are allowed in a base system.	Check that PCIe support should not be present if GICv2.
203	B_GIC_03	L1	If the system includes PCI Express and GICv3 interrupt controller is supported, then the GICv3 interrupt controller must implement ITS and LPI.	Check for the GIC version and PCIe presence in the system, and check for ITS or LPI if present.
204	B_GIC_04	L1	If a GICv3 interrupt controller is supported, then the interrupt controller must support two Security states.	Read security implemented from GICD_CTLR.DS bit.

Test number	Rule ID	Level	Scenario	Algorithm
205	B_GIC_05	L1	The system may implement at least eight Non-secure SGIs assigned to interrupt IDs 0-7.	1. Check whether the Distributor forwards Non-secure Group 1 interrupts. 2. If NS Group 1 interrupts are forwarded, try to enable SGI INTID 0-15 by writing to the Distributor or Redistributor enable register. 3. Check which interrupt IDs are enabled afterwards; Secure interrupts return 0, and Non-secure interrupts return 1.
206, 207, 209, 210, 211	B_PPI_00	L1	A base system must map the interrupts shown in Table 3 (B_PPI_01), Table 4(B_PPI_02) and Table 5 (B_PPI_03) to PPI.	1. Detect the interrupt IDs. 2. Install ISRs for those IDs. 3. Trigger the conditions to assert the interrupts. 4. Check whether the interrupt has been received
-	B_PPI_00	L1	A base system must map the interrupts shown in Table 3 (B_PPI_01), Table 4(B_PPI_02) and Table 5 (B_PPI_03) to PPI. Check PPI assignments for Platform Security - B_PPI_03	Not implemented. BSA ACS executes from Non-secure mode. Cannot access Secure PPI interrupts
226	Section I.6	L1	Check SPIs which are assigned to MSIs are edge-triggered.	1. Get the number of MSI frames. 2. Loop through number of SPIs for that frame. 3. Check by reading the value of GICD_ICFGRn register for SPI.
227	Section I.9	L1	Check GICv2m MSI Frame Register Configuration.	1. Get the number of MSI frames. 2. Loop through number of SPIs for that frame. 3. Check MSI_TYPER is RO. 4. Check for SPI_ID is 32-1020. 5. Check MSI_IIDR is RO.
228	Section I.6	L1	Check GICv2m MSI to SPI generation functional test.	1. Get the number of MSI frames. 2. For each frame, get an SPI_ID. 3. Install a handler. 4. Trigger an SPI using MSI_SETSPI_NS register. 5. Check if the interrupt is triggered.

Test number	Rule ID	Level	Scenario	Algorithm
229	Section I.5	L1	SPIs that are allocated to MSIs must only be controllable by the GICv2m MSI registers.	1. Get the number of MSI frames. 2. For each frame, get an SPI_ID. 3. Install a handler. 4. Generate SPI using GICD_ISPENDR register. 5. No interrupt should be generated. 6. Trigger an SPI using the MSI_SETSPI_NS register. 7. Check if the interrupt is triggered as SPIs should be controlled by the MSI frame register instead of other GICD registers.

2.5 SMMU

Test number	Rule ID	Level	Scenario	Algorithm
301	B_SMMU_01	L1	All the System MMUs presented to an OS or hypervisor must be compliant with the same architecture version.	For all SMMU controllers in the system, check if they are based on either the v2 or v3 architecture.
302	B_SMMU_02	L1	The SMMU must support the translation granule sizes that are supported by the PEs.	1. Read out the granule size support for the PE (AA64MMFR0_EL1.TGran$_x$). 2. Verify whether this is congruent to the granule sizes supported by the SMMU.
303	B_SMMU_06	L1	This means that if PEs implement FEAT_LPA (ID_AA64MMFR0_EL1.PARange = 0b0110), then the SMMU must support a 52-bit output size (SMMU_IDR5.OAS = 0b0110).	1. Read out: ID_AA64MMFR0_EL1.PARange. 2. If enabled, check if SMMU_IDR5.OAS equals "0b0110".
-	B_SMMU_07	L1	Devices that operate across non-contiguously allocated memory require stage 1 System MMU functionality.	Not implemented. This requires device-specific drivers to be available and device-specific knowledge. This is not available from UEFI shell application.

Test number	Rule ID	Level	Scenario	Algorithm
304	B_SMMU_08	L1	If Secure-EL2 is not implemented, stage 1 System MMU functionality that is made visible to an operating system must present the interface of a System MMU compatible with one of the following: • The SMMUv2 specification, where each context bank must present a unique physical interrupt to the GIC. • The Arm SMMUv3 specification or higher, where the integration of the System MMUs is compliant with the requirements in Section D.	1. Read out: ID_AA64PFR0_EL1.SEL2. 2. If enabled, and SMMU handles stage 1 policing, check whether SMMU_AIDR.ArchMajorRev ≥ 3 and SMMU_AIDR.ArchMinorRev ≥ 2. Else, check if SMMU major revision is at least 2.
-	B_SMMU_12	L1	All addresses output from a device to an SMMU must lie in a continuous space with no holes. All addresses in this space will be treated equally by the SMMU. There should be no areas within the address space that receive exceptional treatment like bypassing the SMMU.	Not implemented. We require device drivers to generate DMA and a configurable memory map to access the whole address space. Not feasible to generate these transactions from UEFI shell.
305	B_SMMU_16	L1	If a device is assigned and passed through to an operating system under a hypervisor, then the memory transactions of the device must be subject to stage 2 translation, allocation of memory attributes, and application of permission checks, under the control of the hypervisor.	Depending on the architecture revision of the SMMU, check whether SMMU_IDR0.S2TS (v2) or SMMU_IDR0.S2P (v3) is enabled.
-	B_SMMU_17	L1	From a hardware perspective, this means that a base system supporting a protection hypervisor requires all Non-secure DMA capable devices that will be assigned to a Non-secure VM for direct control to be policed by stage 2 System MMU functionality.	Not covered by ACS. Unable to accurately discover Non-Secure DMA capable devices that will be assigned to VM.

Test number	Rule ID	Level	Scenario	Algorithm
329	B_SMMU_18	L1	If Secure-EL2 is not implemented, stage 2 System MMU functionality must be provided by a System MMU compatible with the Arm SMMUv2 specification or Arm SMMUv3 specification.	Check if SEL2 unimplemented, if true continue else SKIP. Check all SMMUs in the system support stage 2 translation. PASS if all supports else FAIL.
306	B_SMMU_19	L1	When stage 2 System MMU functionality is provided by a System MMU compatible with the Arm SMMUv2 specification: • Each context bank must present a unique physical interrupt to the GIC.	1. Create a list of all physical interrupts per context bank with their corresponding IDs. 2. For each interrupt present in the list, check whether the assigned ID appears elsewhere in the list.
-	B_SMMU_24	L1	If Secure-EL2 is implemented, all Secure DMA capable devices that can be assigned to a Secure VM must be policed by stage 2 Secure SMMU functionality.	Not implemented. It is not feasible to check if Secure-EL2 is implemented and get Secure devices configuration from ACS framework in UEFI shell.
-	B_SMMU_25	L1	If Secure-EL2 is implemented, stage 2 Secure MMU functionality must be provided by a system MMU compatible with the Arm SMMUv3.2, or higher, architecture revision where: • The integration of the system MMUs is compliant with the rules in Section D. • SMMU implementations must provide level 1 or level 2 support for page table resizing.	Not implemented. It is not feasible to check if Secure-EL2 is implemented from ACS framework in UEFI shell.

2.6 Clock and Timer

Test number	Rule ID	Level	Scenario	Algorithm
401	B_TIME_01	L1	The base system must include the system counter of the generic Timer.	If the test can access the timer, the test is passed.

Test number	Rule ID	Level	Scenario	Algorithm
407	B_TIME_02	L1	The system counter of the generic timer must run at a minimum frequency of 10MHz.	1. Read the CNTFREQ register. 2. If the read value is greater than 10MHz, test passed. 3. Else test failed
-	B_TIME_03	L1	The counter must not roll over inside a 10-year period.	Not implemented. RW access to CNTControlBase is allowed only through EL3. The BSA ACS is run on Non-secure mode. Cannot change the register value from the tests.
-	B_TIME_04	L1	The architecture of the counter mandates that the counter must be at least 56 bits, and at most 64 bits. From Armv8.4, for systems that implement counter scaling, the minimum becomes 64 bits.	Not implemented. RW access to CNTControlBase is allowed only through EL3. The BSA ACS is run on Non-secure mode. Cannot change the register value from the tests.
-	B_TIME_05	L1	This count must be available to the PE timers when they are active. When the PEs are in power state, the PE timer must be on.	Covered by wakeup and power tests.
402	B_TIME_06	L1	Unless all the local PE timers are always on, the base system must implement a system wakeup timer that can be used when PE timers are powered down.	1. If any platform timer is available, then the test passes. 2. Else, if all PE timers are always on, then the test passes. 3. Otherwise, the test fails.
403	B_TIME_07	L1	The system wakeup timer must in the form of the memory-mapped timer.	1. Check timer is secured or not. 2. For NS timer, check CNTLBASE is accessible or not 3. If CNTLBase is accessible, then timer is in the form of memory mapped timer.
404	B_TIME_08	L1	On timer expiry, the system wakeup timer must generate an interrupt that must be wired to the GIC as an SPI or LPI. Also, the system wakeup timer can be used to wake up PEs.	1. Generate the timer interrupt. 2. Check if the interrupt is reaching GIC. 3. Check the system-specific interrupt controller with interrupt ID.
405	B_TIME_09	L1	The platform either implements hardware always-on PE timers or uses the platform firmware to save and restore the PE timers in a performance-scalable fashion.	1. Program PE timer with a large value X. 2. Program system timer with a small value Y. 3. Put PE in to Power down mode with PSCI call CPU_SUSPEND. 4. PE should wake up due to sys timer interrupt. 5. Read PE timer current count value, should be less than ((X – Y) + 1% of Y) & count should not be zero.

Test number	Rule ID	Level	Scenario	Algorithm
-	B_TIME_10	L1	If the system includes a system wakeup timer, this memory-mapped timer must be mapped on to Non-secure address space.	Not covered by ACS, would be implemented in future.

2.7 Wakeup Semantics

Tests 501-505 are checking the BSA Table 8: Power state semantics.

Test number	Rule ID	Level	Scenario	Algorithm
-	B_WAK_01	L1	A PE must wake in response to a wakeup interrupt, independent of the state of its PSTATE interrupt mask bits, which are the A, I, and F bits, and of the wakeup interrupt priority.	Not implemented in the current release.
-	B_WAK_02	L1	If the system supports a low-power state where the GIC is powered down, then there must be an IMPLEMENTATION DEFINED way to program the power controller to wake a PE on expiry of the system wakeup timer or the generic watchdog. In this scenario, the system wakeup timer or generic watchdog is still required to send its interrupt.	Not implemented in the current release.

501-505	B_WAK_03	L1	Whenever a PE is woken from sleep or off state the OS or hypervisor must be presented with an interrupt so that the PE software can determine the device that requested the wakeup.	1. Wake from EL1 physical timer a. Configure EL1 physical timer interrupt. b. Program timer to expire after a fixed timeout. c. Enter low-power/WFI state. d. After wakeup, check if the EL1 physical timer caused it; pass if yes, otherwise treat as skipped. 2. Wake from EL1 virtual timer (with failsafe) a. Configure EL1 virtual timer interrupt and a separate failsafe timer interrupt. b. Program virtual timer for the target timeout and failsafe to expire slightly later. c. Enter low-power/WFI state. d. After wakeup, pass if the virtual timer fired first, fail if the failsafe fired first, otherwise skip (another wake source). 3. Wake from EL2 physical timer (with failsafe) a. Only run when executing at EL2. b. Configure EL2 physical timer interrupt and an EL1 physical timer as failsafe. c. Program EL2 timer for the target timeout and failsafe to expire slightly later. d. Enter low-power/WFI state and, on wakeup, interpret EL2 timer first as pass, failsafe first as fail, other wake sources as skip. 4. Wake from watchdog WS0 interrupt (with failsafe) a. Enumerate all non-secure watchdogs. b. For each, configure its interrupt and a separate failsafe timer interrupt. c. Program the watchdog timeout and a slightly later failsafe timeout, then enter low-power/WFI. d. On wakeup, treat watchdog first as pass, failsafe first as fail, other wake sources as skip; disable the watchdog afterward. e. If no non-secure watchdogs exist, skip the test. 5. Wake from system timer interrupt (with failsafe)

Test number	Rule ID	Level	Scenario	Algorithm
				a. Program the system timer for the target timeout and a slightly later failsafe timeout, then enter low-power/WFI. b. On wakeup, treat system timer first as pass, failsafe first as fail, other wake sources as skip. c. If no suitable non-secure system timer exists, skip the test.
-	B_WAK_04	L1	The interrupt must be pending in GIC at the point that control is handed back to the OS or hypervisor from the system-specific software performing the state restore.	Not implemented in the current release.
-	B_WAK_05	L1	This interrupt must behave like any other, where a device sends an interrupt to the GIC, and the GIC sends the interrupt to the OS or hypervisor. The OS or hypervisor must not communicate with a system-specific interrupt controller.	Not implemented in the current release.
-	B_WAK_06	L1	If the wakeup event is an edge, then the system must ensure that this edge is not lost. The system must ensure that the edge wakes the system and is then delivered to the GIC without losing the edge.	Not implemented in the current release.

2.8 Power State Semantics

Tests 501-505 are checking the BSA Table 8: Power state semantics.

Test number	Rule ID	Level	Scenario	Algorithm
-	B_WAK_07	L1	OS or hypervisor, or both, can be the reason for wakeup events and to know which timers are available to wake up the PE. All PEs must be in a state that is consistent with one of the semantics.	Covered by Test 501-505.

Test number	Rule ID	Level	Scenario	Algorithm
-	B_WAK_08	L1	System MMUs and GICv3 uses tables in memory in the power states where GIC is on. For this type of state, system memory must be available and responds to requests without requiring intervention from software running on the PEs.	Covered in SMMU and GIC tests.
-	B_WAK_09	L1	When the system is in a state where the GIC is powered down, devices must not send messaged interrupts to the GIC. See Table 8 BSA.	Not implemented in the current release.
-	B_WAK_10	L1	Schematic check.	Not implemented in the current release.
-	B_WAK_11	L1	Component check.	Not implemented in the current release.

2.9 Watchdog

Test number	Rule ID	Level	Scenario	Algorithm
701	B_WD_01	L1	The generic watchdog must be implemented as specified in Section C of BSA specification 1.0.	Read and verify watchdog refresh and control registers.
-	B_WD_02	L1	The watchdog must have both its register frames mapped on to Non-secure address space. This watchdog is referred to as the Non-secure watchdog.	Covered by test 701.
702	B_WD_03	L1	Watchdog signal 0 is routed as an SPI or an LPI to the GIC and it is expected that this is configured as a Non-secure EL2 interrupt, targeting a single PE.	1. Generate the watchdog interrupt. 2. Check if the interrupt is reaching GIC. 3. Check the system-specific interrupt controller with interrupt ID.
-	B_WD_04	L1	Watchdog signal 1 must be routed to the platform.	Not implemented. WS1 signal is routed to a higher privilege entity to perform IMPDEF behavior.

Test number	Rule ID	Level	Scenario	Algorithm
-	B_WD_05	L1	The action taken on the raising of watchdog signal 1 is IMPLEMENTATION SPECIFIC.	Not implemented. WS1 signal performs platform-specific action. This is IMPDEF behavior.

2.10 Peripherals

Test number	Rule ID	Level	Scenario	Algorithm
601	B_PER_01	L1	If the system has a USB2.0 host controller peripheral, it must conform to EHCI v1.1 or later.	For systems with ACPI 1. Get the BDF of USB. 2. Read the class code of the BDF from the config space and check if it confirms to the specification. 3. If not a PCIe device, then read the class code through firmware and check if it conforms to the specification. For systems with DeviceTree All the available UART in the Device Tree are parsed and their type is obtained using compatible string, if the type does not match EHCI/XCHI test is failed. If EHCI passed If XHCI test is skipped.
608	B_PER_02	L1	If the system has a USB3.0 host controller peripheral, it must conform to XHCI v1.0 or later.	For systems with ACPI 1. Get the BDF of USB. 2. Read the class code of the BDF from the config space and check if it confirms to the specification. 3. If not a PCIe device, then read the class code through firmware and check if it conforms to the specification. For systems with DeviceTree All the available UART in the Device Tree are parsed and their type is obtained using compatible string, if the type does not match EHCI/XCHI test is failed, If XHCI passed If EHCI test is skipped.

Test number	Rule ID	Level	Scenario	Algorithm
602	B_PER_03	L1	If the system has a SATA host controller peripheral, it must conform to AHCI v1.3 or later.	For systems with ACPI 1. Get the BDF of SATA device. 2. Read the class code of the BDF from the config space and check if it confirms to the specification. 3. If not a PCIe device, then read the class code through firmware and check if it confirms to the specification. For systems with DeviceTree All available SATA in the Device Tree is parsed and their type is obtained using compatible string, if the type does not match AHCI test is failed.
-	B_PER_04	L1	Peripheral subsystems which do not conform to rues B_PER_01, B_PER_02 or B_PER_03 are permitted if those peripherals are not required to boot and install an OS.	Manual verification required by user.

Test number	Rule ID	Level	Scenario	Algorithm
603	B_PER_05	L1	For system development and bring up, the base system must include a UART. The UART must be one of: • The generic UART as specified in Section B. A fully 16550 compatible UART [9].	For generic UART algorithm steps are 1. Install handlers to catch unexpected exceptions. 2. Get the base address for each UART present. 3. Validate the read-only registers UARTFR, UARTRIS, UARTMIS, UARTDR if they conform to BSA specification. For 16550 UART, algorithm steps are: 1. Get 16550 UART details from system hardware table (ACPI or DT). 2. Check if the I/O base address is present. 3. Check the Baud rate from the hardware table. 4. Check the Baud rate in the UART register. Obtain the divisor by enabling the divisor latch access and reading the divisor latch byte1 and byte2. Divisor = system clock speed / (16 * Baud rate). 5. Check the read and write property of Line Control Register. 6. Check the read and write property of Interrupt Enable Register. 7. Check if UART is present using loopback test mode
606	B_PER_06	L1	The UART interrupt output is connected to the GIC as an SPI or an LPI.	1. Get the interrupt ID of the UART peripheral. 2. Install the ISR for the interrupt ID. 3. Generate an UART transaction. 4. Check if the interrupt is received in the ISR.
-	B_PER_07	L1	UART must be mapped on to Non-secure address space. This is called the Non-secure UART.	Not covered by ACS, would be implemented in future.
-	B_PER_08	L1	If the system has a PCI Express root complex, then it must comply with the rules in Section E.	Covered by PCIe tests.

Test number	Rule ID	Level	Scenario	Algorithm
604	B_PER_09	L1	The memory attributes of DMA traffic must be one of the following: • Inner Write-Back, outer Write-Back, Inner Shareable. • Inner non-cacheable, outer Non-cacheable. • A device type.	1. Get the DMA controllers present in the system. 2. Allocate a memory for a DMA transaction. 3. Read the memory attributes of the allocated memory. 4. Check if traffic attributes comply with attributes mentioned in rule. If complies PASS the test.
607	B_PER_10	L1	I/O coherent DMA traffic must have the attribute - Inner Write-Back, Outer Write-Back, Inner Shareable.	1. Get the I/O coherent DMA controllers present in the system. 2. Allocate a memory for a DMA transaction. 3. Read the memory attributes of the allocated memory. 4. Check if traffic attributes comply with attributes mentioned in rule. If complies PASS the test.
-	B_PER_11	L1	If a TCG TPM-based security model is supported, the base system must provide a TPM implementation that is compliant to TPM library specification, Family 2.0.	Covered by SIE ACS
-	B_PER_12	L1	To ensure standard software support, a device claiming to follow the PCI Express specification must follow all the rules in PCIe specification which are software-visible	Covered by PCIe tests

2.11 PCIe

Test number	Rule ID	Level	Scenario	Algorithm
801	PCI_IN_01	L1	Systems must map memory space to PCI Express Configuration Space, using the PCI Express Enhanced Configuration Access Mechanism (ECAM).	Read out the number of detected ECAM regions.
802	PCI_IN_02	L1	Once the boot firmware hands control over to the operating system, application processor accesses to ECAM regions must work with no additional programming. The accesses must not require any OS visible programming.	For each ECAM region, access the PCI header space and extended PCIe configuration space.
-	PCI_IN_03	L1	The configuration space of all the devices, Root Ports, Root Complex Integrated Endpoints, and switches behind a PHB must be in a single ECAM region.	Part of device print information. Manual verification required by user by analyzing prints in the logs.
803	PCI_IN_04	L1	The configuration space of all the endpoints and switches in a Root port's hierarchy must be in the same ECAM space as the root port.	For all PCIe devices and switches in the system: 1. Find the Root Port they are under. 2. Check if the RP and the device are in the same ECAM region.
820, 822, 824, 825, 826, 833, 1517	PCI_IN_05	L1	Root Port must appear as a PCI-PCI bridge to software (See Section 7.1). This implies that a Root Port: • Must have all registers that are part of the type 1 header, as specified in PCIe specification (See Section 7.5.1.3). • Must have all the capabilities required by PCIe specification for a Root Port. This includes the PCI Express capability structure (See Section 7.5.3). • Registers must follow the access attributes (RW/RO, and so on) specified in the PCIe specification.	Register checks. Covered by Test 820, 822,824,825,826,833

Test number	Rule ID	Level	Scenario	Algorithm
-	PCI_IN_06	L1	PHB with Root Port, must recognize transactions that are coming in from application PEs as PCIe configuration transactions if the transaction address is within the ECAM range mapped to the Root Port, or the hierarchy that originates at that Root Port. This must be done by mapping the address of the incoming memory transaction to the PCIe configuration address space, as described in Table 22 (See the Section 7.2.2).	Covered by PCIe table creation.
-	PCI_IN_07	L1	PHB with Root Port must return all 1s as read response data for configuration read requests to nonexistent functions and devices on the root bus, that is the primary bus of the Root Port. No error must be reported to the software by the Root Port unless explicitly enabled to do so.	Covered by PCIe Table creation.
-	PCI_IN_08	L1	PHB with Root Port, must return all 1s as read response data for configuration read requests that get an unsupported request response from downstream endpoints or switches. No error must be reported to software by the Root Port unless explicitly enabled to do so.	Covered implicitly as part of multiple PCIe tests Requires manual verification of ACS bdf entries with system pci/lspci output

Test number	Rule ID	Level	Scenario	Algorithm
-	PCI_IN_09	L1	PHB with Root Port must return all 1s as read response data for configuration read requests that arrive at the Root Port when the Root port link is in DL_Down state (See Section 2.9.1 [1]). Note that this includes the case when the link is in L3 and the downstream device is in D3cold. No error must be reported to software by the Root Port unless explicitly enabled to do so.	Not implemented. Controlling link state and device state through test sequence is not feasible without device-specific drivers.
-	PCI_IN_10	L1	PHB with Root Port must send out configuration transactions that are intended for the subordinate bus range of the Root Port as type 1 configuration transactions to downstream devices and switches. Subordinate bus range is between secondary bus number, exclusive, and the subordinate bus number, inclusive.	Increment the Root Port with exerciser endpoint connected subordinate bus by 1. Generate transaction by reading endpoint vendor id register. The endpoint must receive type 1 transactions. In case RP has a right sibling, set its secondary and subordinate bus number to invalid bus number so that transaction is not forwarded to it. Bus number restored at the end of test.
1510	PCI_IN_11	L1	PHB with Root Port must send out configuration transactions that are intended for the secondary bus of the Root Port as type 0 configuration transactions to devices and switches downstream (See the Section 3.2.2.3.1).	Initiate transaction for exerciser endpoint which is connected to the Root Port by reading its vendor ID. The exerciser must get transaction as type 0.
837	PCI_IN_12	L1	PHB with Root Port must recognize and consume configuration transactions intended for the Root Port configuration space and, read or write the appropriate Root Port configuration register (See the Section 3.2.2.3.1).	For all Root Ports in the system: 1. Read the Class Code through the ECAM Method and Pciio Protocol method 2. The value obtained from both the methods should match

Test number	Rule ID	Level	Scenario	Algorithm
804 , 805	PCI_IN_13	L1	PHB with Root Port must recognize transactions received on the primary side of the Root Port PCI-PCI bridge, targeting prefetchable or non-prefetchable memory spaces of devices and switches that are on the secondary side of the bridge.	For all Root Ports in the system: 1. Detect the Non-Prefetchable address range. 2. Read/write memory from/to this memory range.
--	PCI_IN_14	L1	PHB with Root Port must return all 1s data to the requestor PE as the response for a configuration read if the following are true: • CRS software visibility is disabled or not present. • CRS response was received for the request the first time it was issued by the Root Complex. The Root Complex then tried to make the request return valid data by re-issuing the request an IMPLEMENTATION DEFINED number of times, but CRS was the response received for all such re-issues.	Not implemented. Generating CRS requests as part of test sequence is not possible on silicon tests.
--	PCI_IN_15	L1	PHB with Root Port must return all 1s data to the requestor PE as the response for a configuration read if all the following are true: • CRS software visibility is enabled. • The configuration read is not targeting the Vendor ID register. • CRS response was received for the request the first time it was issued by the Root Complex. The Root Complex then tried to make the request return valid data by re-issuing the request an IMPLEMENTATION DEFINED number of times, but CRS was the response received for all such re-issues.	Not implemented. Generating CRS requests as part of test sequence is not possible on Silicon tests.

Test number	Rule ID	Level	Scenario	Algorithm
808	PCI_IN_16	L1	PHB in conjunction with the Root Port must return all 1s data to the requestor PE as the response for a configuration read if the following are true: • Target bus number of the request is not within the secondary bus to subordinate bus range of any of the Root Ports. • Target Bus, Device, and Function (BDF) of the request does not match BDF of any on-chip functions. • Target BDF of the request does not match the BDF of any of the Root Ports.	1. Get the maximum bus value from the PCIe info table. 2. Get highest BDF of the segment. 3. Get least high of max bus number. 4. Form BDF using the segment, bus, device, and function numbers. 5. Read should return all 1s.
836,1515	PCI_IN_17	L1	The Root port must comply with the following (as per section 6.13 of PCI Express Base Specification Revision 5.0, version 1.0. PCI-SIG).	Partially covered. If ARI forwarding is disabled and target device number of the request > 0, then the access is terminated and all 1s data is returned to the requestor PE. If ARI forwarding is enabled, then the access is to the secondary bus of the Root Port, and the access is forwarded downstream as type 0 configuration request regardless of the target device number of the request.
811	PCI_IN_18	L1	The Root Port must comply with the byte enable rules that are specified in the PCIe specification (See Section 2.2.5) and must support 1 byte, 2 byte and 4-byte configuration read and write requests.	For every Root Port in the system: 1. Read the command register of the RP with 8-bit (1 byte), 16-bit (2 byte), 32-bit (4 byte), and compare. 2. Verify the read and write behavior for each of 8-bit (1 byte), 16-bit (2 byte), and 32-bit (4 byte).
-	PCI_IN_19	L1	All registers present in the Root Port PCIe configuration space must follow the rules as defined in section 7.2 of the PCIe specification.	Covered by other PCIe test 820, 822, 830, 831, 832

Test number	Rule ID	Level	Scenario	Algorithm
809	PCI_IN_20	L1	Any vendor-specific data in the PCIe configuration space must be presented by one of the following capabilities, as defined in the PCIe specification: • Vendor-specific Capability • Vendor Specific Extended Capability (VSEC) • Designated Vendor-Specific Extended Capability (DVSEC).	For all the Root Ports in the system, search through the base and extended PCIe configuration spaces for non-PCIe compliant capabilities.
845, 1516	PCI_MM_01	L1	All systems must support mapping PCI Express memory space as device memory.	Map the BARs with device memory attributes and verify that transactions complete without triggering any exceptions.
-	PCI_MM_02	L1	All systems must support mapping PCI Express memory space as non-cacheable memory.	Not covered
894, 1539	PCI_MM_03	L1	When PCI Express memory space is mapped as Normal memory, the system must support unaligned accesses to that region.	Map the BARs to normal memory attribute and check unaligned access is complete without triggering any exceptions

Test number	Rule ID	Level	Scenario	Algorithm
847	PCI_MM_04	L1	Systems compliant to this specification must support 32-bit programming of NP BARs on such endpoints. This can be achieved in two ways: Method 1: PE physical address space can be reserved below 4GB, while maintaining a one-to-one mapping between PE physical address space and NP memory address space. Method 2: It is also possible to use a fixed offset translation scheme that creates a fixed offset in direction between PE physical address space, and PCI memory. This allows a window in PE physical address space that is above 4G to be mirrored in PCI memory space below 4G. This requires support in the PHB. Furthermore, firmware must program the PHB with the fixed offset, and to supply this information to the OS.	For the host bridge in the system, check the pre-fetchable type, if 0 then read the memory type. Scan all the bridge devices and check the memory type.
895	PCI_MM_05	L1	For accesses from a PCIe endpoint to the host memory system, the address sent by PCI Express devices must be presented to the memory system or SMMU unmodified.	For all DMA Requesters populated in the info table which are behind an SMMU, verify there are no additional translations before address is given to SMMU. Check if IOMMU operations are properly integrated for this device by making the standard OS DMA API call and verifying the DMA address is part of the IOVA translation table.
-	PCI_MM_06	L1	For accesses from a PCIe endpoint to the host memory system, in a system where the PCI Express does not use an SMMU, the PCI Express devices have the same view of physical memory as the PEs.	Not covered

Test number	Rule ID	Level	Scenario	Algorithm
905	PCI_MM_07	L1	For accesses from a PCIe endpoint to the host memory system, in a system with an SMMU for PCI Express there are no transformations to addresses being sent by PCI Express devices before they are presented as an input address to the SMMU.	For all DMA Requesters populated in the info table which are behind an SMMU, verify there are no additional translations before address is given to SMMU.
839	PCI_MSI_1	L1	Support for Message Signaled Interrupts (MSI/MSI-X) is required for PCI Express devices.	Check if PCI device is PCI Express capable and MSI is supported
1533 ,897	PCI_MSI_2	L1	The intended use model is that each unique MSI(-X) must trigger an interrupt with a unique ID and the MSI(-X) must target GIC registers requiring no hardware-specific software to service the interrupt.	For all exercisers in the system, 1. Disable all SMMU and check if MSI-X capability is supported. If not supported, skip. Else, create MSI mapping and configure the MSI table. 2. Install ISR and trigger the interrupt. 3. Check completion of ISR. Clear the interrupt and the MSI mappings.
806	PCI_LI_01	L1	PCI Express legacy Interrupt messages must be converted to an SPI.	For all PCIe devices in the system: 1. Detect whether the legacy IRQ map exists. 2. If it exists, check whether the interrupt falls in SPI range.
1506, 896	PCI_LI_02	L1	A unique SPI ID must be allocated to each of the legacy interrupt lines of a PHB. It is permissible to share SPIIDs across PCI host bridges.	1. Allocate memory for interrupt mappings. 2. Get the exerciser BDF. 3. Register an interrupt handler to verify legacy interrupt functionality. 4. Trigger the legacy interrupt. 5. Check the completion of interrupt service routine. 6. Return the interrupt.

Test number	Rule ID	Level	Scenario	Algorithm
823	PCI_LI_03	L1	Each legacy interrupt SPI must be programmed as level-sensitive in the appropriate GIC_ICFGR.	For all PCIe devices in the system: 1. Detect whether the legacy IRQ map exists. 2. If it exists, read GICD_ICFGR to check whether the interrupts in this map are level or edge sensitive.
--	PCI_LI_04	L1	IMPLEMENTATION DEFINED registers must not be used to deliver these messages, only registers defined in the PCI Express specification and the Arm GIC specification.	OS boot covers this.
--	PCI_SM_01	L1	Hardware support for function or virtual function assignment to a VM or user-space driver is optional, but if required must use a System MMU compliant with the Arm System MMU specification.	OS boot covers this.
835	PCI_SM_02	L1	Functions intended for VM assignment, or assignment to a user space driver must implement function level reset.	For all Endpoints in the system if FLR is supported, check: 1. Initiate FLR by setting the FLR bit 2. Wait for max FLR period 3. The device must be recognized after max FLR period

Test number	Rule ID	Level	Scenario	Algorithm
1507	PCI_IC_11	L1	If an I/O subsystem supports I/O coherency, it must be in the same Inner Shareability domain as the PEs.	For all exercisers in the system, 1. Find the SMMU node and disable it globally so that the transaction passes through the SMMU without address modification. 2. Get a WB, outer shareable DDR buffer. 3. Program exerciser hierarchy to start sending or receiving TLPs with no snoop attribute header. 4. Initialize main memory region marked as WB, Outer Shareable by the PE page tables. CPU Write and read to this region with new data and ensure that the new data is cached in the CPU caches. Read and write the same data locations from the Exerciser with NS=0. The exerciser and CPU should get the latest data (hardware enforced cache coherency expected).
-	PCI_IC_12	L1	Until final memory type and attributes are known, the transaction must follow PCI Express transaction rules and must not be collapsed/combined/merged or otherwise altered based on an intermediate memory type or attribute.	Not implemented
-	PCI_IC_13	L1	PCI Express Translated transactions may contain ATS Memory Attributes (AMA). An implementation must provide a mode of operation where AMA values, if presented in the PCI Express transaction, are ignored and do not alter the memory type, memory attributes, or cache allocation hints of a transaction. Any other interpretation of AMA is IMPLEMENTATION DEFINED. Arm may provide guidance in the future with respect to AMA.	Not implemented

Test number	Rule ID	Level	Scenario	Algorithm
-	PCI_IC_14	L1	The initial memory attributes associated with a PCI Express transaction must be Normal Inner and Outer Write-back Cacheable Outer Shareable.	Not implemented
1503	PCI_IC_15	L1	PCI Express transactions contain a No_snoop attribute. If the No_snoop attribute is set in the PCI Express transaction, the memory attributes associated with the transaction must be replaced with Normal Inner and Outer Non-cacheable.	1. Read and write on config space mapped to device memory. Map config space to Arm device memory in MMU page tables. Perform transactions on incremental aligned address and on the same address. 2. Read and write on BAR space mapped to Device memory. Map MMIO space to Arm device memory in MMU page tables. Perform transactions on incremental aligned address and on the same address.
-	PCI_IC_16	L1	If there is a System MMU in the path of the transaction, the memory attributes associated with the transaction must be provided as input to the System MMU. The memory attributes of the transaction must be replaced with the output of the System MMU. See [15] for System MMU operation	Not implemented

Test number	Rule ID	Level	Scenario	Algorithm
-	PCI_IC_17	L1	For some address targets and platform configurations, an IMPLEMENTATION DEFINED mechanism may need to be provided to override or ignore the memory type and attributes associated with the transaction in order to guarantee the required properties of the target: • Message Signaled Interrupts (MSIs) must be recorded according to PCI Express transaction rules. They must remain ordered behind prior writes, must not merge, and must not allocate in caches. • If PCI Express peer-to-peer functionality is supported, PCI Express transactions that target PCI Express memory-mapped I/O space must not violate PCI Express transaction rules or the properties required by the target PCI Express BAR region. • Other special address regions may be defined by the platform.	Not implemented
-	PCI_IC_18	L1	The shareability domain of an I/O subsystem that supports PCI Express must not contain any caches outside the shareability domain of the PEs.	Not implemented
-	PCI_IO_01	L1	If an implementation supports legacy I/O, it is supported using a one-to-one mapping between legacy I/O space and a window in the host physical address space. However, such schemes must not require a kernel driver to be set up, any necessary initialization must be performed before OS boot.	Not implemented. This rule requires an EP with legacy I/O support and device driver.

Test number	Rule ID	Level	Scenario	Algorithm
-	PCI_IEP_1	L1	Anything claiming to follow the PCI Express specification must follow all the specification that is software-visible to ensure standard, quality software support.	Covered by SBSA PCIe tests
	PCI_PP_01	L1	It is system-specific whether peer-to-peer traffic through the system is supported.	Not implemented. This is IMPDEF rule.
1514	PCI_PP_02	L1	Systems must not deadlock if PCI Express devices attempt peer-to-peer transactions, even if the system does not support peer-to-peer traffic. This rule must uphold the principle that a virtual machine and its assigned devices should not deadlock the system for other virtual machines or the hypervisor.	1. Check if the platform does not support peer-to-peer transaction 2. Initiate a peer-to-peer transaction by generating a DMA from one exerciser to another 3. The transaction should not result in a deadlock and the ACS must continue execution
819	PCI_PP_03	L1	In a system where the PCIe hierarchy allows peer-to-peer transactions, the root ports in an Arm-based SoC must implement PCIe access control service (ACS) features.	It is IMPLEMENTATION DEFINED whether a given platform supports peer-to-peer traffic. If the platform supports this, check in the ACS PCIe capability whether: 1. Source validation is supported. 2. Translation blocking is supported. 3. P2P request redirect is supported. 4. P2P completion redirect is supported. 5. Upstream forwarding is supported

Test number	Rule ID	Level	Scenario	Algorithm
1501, 1502	PCI_PP_04	L1	For Root ports this means that the following must be supported: 1. ACS Source Validation. (V) 2. ACS Translation Blocking. (B) 3. ACS P2P Request Redirect (R). 4. ACS P2P Completion Redirect (C). 5. ACS Upstream Forwarding (U). 6. The Root Port must support redirected request validation by querying an Arm architecture compliant SMMU to get the final target physical address and access permission information. 7. The Root Port must support ACS violation error detection, logging, and reporting. Logging and reporting must be through the usage of AER mechanism.	It is IMPLEMENTATION DEFINED whether a given platform supports peer-to-peer traffic. If the platform supports this, For an exerciser in the system. 1. Get RP of the exerciser. 2. If ACS supported, enable Source Validation & Transaction Blocking. 3. Find another exerciser on other root port, break from the test if no such exerciser is found. 4. If both RPs supports ACS, then check for ACS functionality.

-	PCI_PP_06	L1	Since isolation between IO devices can be broken by the presence of any peer-to-peer capable entity in a PCIe hierarchy, the following is recommended for an Arm based system: • All PCIe switches should support the following features: 1. ACS Source Validation (V). 2. ACS Translation Blocking (B). 3. ACS P2P Request Redirect (R). 4. ACS P2P Completion Redirect (C). 5. ACS Upstream Forwarding (U). 6. ACS Direct Translated P2P (T). 7. ACS violation error detection, logging, and reporting as specified in the PCIe specification for ACS 8. Use of AER capability for logging and reporting ACS violation errors • All multi-function devices, SR-IOV and non-SR-IOV, that are capable of peer-to-peer traffic between different functions should support the following features: 1. ACS P2P Request Redirect (R). 2. ACS P2P Completion Redirect (C). 3. ACS Direct Translated P2P (T) 4. The device must support ACS violation error detection, logging, and reporting as specified in the	It is IMPLEMENTATION DEFINED whether a given platform supports peer-to-peer traffic. If the platform supports this, then for every DP or UP of a switch, check in the ACS PCIe capability whether 1. Source validation is supported. 2. Translation blocking is supported. 3. P2P request redirect is supported. 4. P2P completion redirect is supported. 5. Upstream forwarding is supported 6. Direct Translated P2P is supported

Test number	Rule ID	Level	Scenario	Algorithm
			L1PCIe specification for ACS [1].	
817, 818	PCI_PP_05	L1	If the Root Port supports peer-to-peer traffic with other root ports, then it must support the following: • Validation of the peer-to-peer transactions before sending it to the destination root port using the same mechanism as ACS redirected request validation. Any ACS violation error generated because of the request validation should be reported using the standard ACS violation error detection, logging and reporting mechanism specified in PCIe specification. • If the Root port supports Address Translation services and peer-to-peer traffic with other root ports, then it must support ACS direct translated P2P (T).	It is IMPLEMENTATION DEFINED whether a given platform supports peer-to-peer traffic. If the platform supports this, then check: 1. If ACS is supported, direct translated P2P is supported. 2. AER is supported
842	PCI_PAS_1	L1	If the system supports PCIe PASID, then at least 16 bits of PASID must be supported. This support must be full system support, from the root complex through to the SMMUv3 and any end points for which PASID support is required.	For all devices in the system behind an SMMU: Verify the maximum PASID width supported by the device through the PASID capability structure is at least 16 bits and also ensure that the SMMU supports a minimum of 16 PASID bits.

Test number	Rule ID	Level	Scenario	Algorithm
-	PCI_PTM_1	L1	Any system that implements PCIe Precision Time Measurement (PTM) must use the Arm architecture defined System Counter as PTM Requester time source at the PTM root(s).	Not implemented in current release.

2.12 DeviceID Generation and ITS Groups

Test Number	Rule ID	Level	Scenario	Algorithm
251	ITS_01	L1	An ITS group can contain one or more ITS blocks.	1. Get number of ITS groups. 2. Check the number of ITS blocks in each ITS group. 3. If number of blocks < 1 in any group, the test fails. Otherwise, the test passes.
252	ITS_02	L1	An ITS block is associated with one ITS group.	1. Get ITS id of first block in first group. 2. Check whether ITS id is repeating in other groups, if yes, the test fails. 3. Repeat this for all ITS blocks.
1511	ITS_03	L1	A device that is expected to send an MSI is associated with one ITS group.	Verify whether the device supports MSI or MSI-X capabilities. If supported, ensure it is associated with a valid ITS group.
1535	ITS_04	L1	Devices can be programmed to send MSIs to any ITS block within the group.	1. Get ITS ID, device ID for current instance of exerciser. 2. Get Group Index of this ITS using get_its_info api. 3. Create ITS LPI mappings for this device, and fill MSI-X table. 4. Generate MSI and check if interrupt is raised. 5. Repeat 3-4 for all the ITS Blocks for this ITS Group. 6. Repeat 1-5 for all exerciser instances.

Test Number	Rule ID	Level	Scenario	Algorithm
1512	ITS_05	L1	If a device sends an MSI to an ITS block outside of its assigned group, the MSI write is illegal and does not trigger an interrupt that could appear to originate from a different device. See Section H.2.2 for permitted behavior of illegal MSI writes.	1. Get ITS ID, device ID for current instance of exerciser. 2. Get Group Index of this ITS using get_its_info api. 3. Get ITS ID in an ITS group other than assigned ITS group. 4. Create ITS LPI mappings for this device, and fill MSI-X table. 5. Try to Generate MSI and check interrupt should not be raised. 6. Repeat 1-5 for all exerciser instances.
-	ITS_06	L1	An ITS group represents a DeviceID namespace independent of any other ITS group.	Test 1535, 1512 combined covers this rule. In 1535, we are checking for inside the ITS group we can generate the interrupt. In 1512, we are checking, interrupt should not be raised for other ITS Groups.
-	ITS_07	L1	All ITS blocks within an ITS group support a common DeviceID namespace size, a common input EventID namespace size and can receive an MSI from any device within the group.	Not completely covered
-	ITS_08	L1	All ITS blocks within an ITS group observe the same DeviceID for any given device in the same ITS group.	1535 covers this rule
-	ITS_DEV_1	L1	Every device Requester that is expected to send MSIs has a DeviceID associated with it.	Covered by Test 1535.
253	ITS_DEV_2	L1	The system designer assigns a Requester unique StreamID to device traffic input to the SMMU.	1. Get the StreamID for the device. 2. Check for remaining device of the group. 3. StreamID should be unique in a group. 4. Repeat this for all the groups.
-	ITS_DEV_3	L1	When a device is not behind an SMMU, its DeviceID appears to high-level software as though it is assigned directly by the system designer.	Not implemented in the current release.

Test Number	Rule ID	Level	Scenario	Algorithm
1513	ITS_DEV_4	L1	The system must not allow this behavior to trigger an MSI that masquerades as originating from a different Requester. The system must anticipate that PEs also have the potential to be misused in this manner.	1. Create all the mappings for an Exerciser A. 2. Get the MSI ADDR + DATA values for this mapping. 3. Program MSI Table of Exerciser B (Same ITS group as Exerciser A) with above Addr + Data. 4. Use val_exerciser_ops (Generate_MSI), for Exerciser B in the same group. 5. Since mappings were created for Exerciser A, B should not be able to generate MSI.
-	ITS_DEV_5	L1	Every device that is expected to originate MSIs is associated with a DeviceID.	Covered by Test 1535
1504	ITS_DEV_6	L1	DeviceID arrangement and system design prevents any mechanism that any software that is not the most privileged in the system, for example VM, or application, can exploit to trigger interrupts associated with a different body of software, for example. a different VM, or OS driver.	For all exercisers in the system, 1. Disable all SMMU and check if MSI-X capability is supported. If not supported, skip. Else, create MSI mapping and configure the MSI table. 2. Install ISR. 3. trigger the interrupt by writing to GITS_TRANSLATER from the PE. 4. Check interrupt should not be raised. Clear the MSI mappings.
254	ITS_DEV_7	L1	If a device is a client of an SMMU, the associated DeviceID is derived from the SMMU StreamID with an identity or simple offset function.	1. Get information from the first device in the group. 2. Calculate the Constant offset for the group. 3. Check for remaining device of the group. 4. Repeat this for all the groups.

Test Number	Rule ID	Level	Scenario	Algorithm
255	ITS_DEV_8	L1	DeviceIDs derived from other kinds of system IDs are also created from an identity or simple offset function. For a Root Complex without an SMMU, the relationship is: DeviceID = zero_extend(RequesterID[N-1:0])+ (1<<N)*Constant_C	1. Identify MSI/MSI-X Devices Without SMMU 2. For each ITS group, establish the initial DeviceID - RequesterID offset and verify it remains constant across devices in the same group.
-	ITS_DEV_9	L1	The relationships between a device, its StreamID and its DeviceID are considered static by OS or hypervisorsoftware. If the mapping is not fixed by hardware, the relationship between a StreamID and a DeviceID mustnot change after system initialization, and OS drivers must not be required to set it up.	Not implemented in the current release.

NOTE: ITS rules are available only for systems that present firmware compliant to SBBR.

Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue 01

Change	Location
First release	-

Table A-2 Issue 02

Change	Location
Updated the ITS tests in DeviceID Generation and ITS Groups section.	**2.12 DeviceID Generation and ITS Groups**
Added PCI_IN_08, PCI_PP_02, RE_ORD_4 rule IDs in PCIe section.	**2.11 PCIe**

Table A-3 Issue 03

Change	Location
Deleted B_PE_16, B_PE_17, B_SEC_01, B_SEC_02, B_SEC_03, B_SEC_04, B_SEC_05 rule IDs in PE section.	**2.2 PE**
Deleted B_MEM_09 rule IDs in Memory Map section.	**2.3 Memory Map**
Deleted B_SMMU_03, B_SMMU_04, B_SMMU_05, B_SMMU_09, B_SMMU_11, B_SMMU_13, B_SMMU_14, B_SMMU_20, B_SMMU_21, B_SMMU_22, B_SMMU_23 rule IDs in SMMU section.	**2.5 SMMU**
Deleted RCiEP rule IDs in PCIe section.	**2.11 PCIe**

Table A-4 Issue 03 to Issue 0101-04

Change	Location
Single Test scenario document for BSA ACS	
Updated the exerciser and PCIe tests number as per latest code	
Added note for ITS test NA for IR systems	

Table A-5 Issue 0101-04 to Issue

Change	Location
Updated algorithm for B_PE_04 rule ID in PE section.	**2.2 PE**

Table A-5 Issue 0101-06 to Issue 0101-07

Change	Location
Added BSA Errata Updates in PE and GIC	**PE GIC**

Table A-5 Issue 0101-07 to Issue 0101-08

Change	Location
• Added explicit tests for B_TIME_01 (generic timer presence) and B_TIME_02 (minimum frequency requirement.)	Timer
• Updated algorithm for B_WAK_03 and B_WAK_07. • Coverage for B_WAK_01, B_WAK_02, B_WAK_04, B_WAK_05, B_WAK_06, B_WAK_09 - B_WAK_11 is under analysis and was removed for the release	Power Wakeup
• Added a new test for B_PE_24, which is no longer implicitly covered by B_PE_23	PE
• Split tests for addressability rules B_MEM_03, B_MEM_04, and B_MEM_06 into exclusive tests.	Memory Map
• B_PER_09 and B_PER_10 test was split to target coherent and non-coherent DMA separately. • Consolidated UART tests for B_PER_05 and S_L3PR_01 into a single test.	Peripherals
• PCIe header rules – 820/822 now focus on Type-1 header (PCI_IN_05). Reference to PCI_IN_19 dropped; 1507 keeps only PCI_IC_11. • Bus Master Enable test (Root Port) – 1517 now stands alone to check BME for RP devices (PCI_IN_05). • PASID rule – PCI_PAS_1 removed from the exerciser set; now handled by PCIe tests. • ITS rules – 1511 now covers only ITS_03; new 1535 covers ITS_04. • Memory-access tests – 1516 keeps device-memory check (PCI_MM_01); new 1539 checks normal memory access (PCI_MM_03). • Legacy interrupts – 806 limited to PCI_LI_01; new 823 verifies level-sensitive legacy IRQs. • PCIe view of physical memory – 895 now only covers PCI_MM_05; new 905 covers PCI_MM_07 (no extra SMMU translation). • Removed explicit PCI_PP_06 mention from 893 (rule is informational and implied for switches).	PCIe
• GIC ITS mapping – Split test 254 to target only ITS_DEV_7; new 255 handles ITS_DEV_8 (Root complex without SMMU case).	ITS Groups