AMBA® Low Power Interface

Specification

arm

ARM IHI 0068D (ID102021)

AMBA Low Power Interface
Specification

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
30 July 2013	A	Confidential	Limited initial release
10 February 2014	B	Non-Confidential	First release
13 September 2016	C	Non-Confidential	Second release
29 September 2021	D	Non-Confidential	Added information on Interface Parity Protection

Proprietary Notice

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
LES-PRE-21451 version 2.2

AMBA SPECIFICATION LICENCE

THIS END USER LICENCE AGREEMENT ("LICENCE") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER A SINGLE INDIVIDUAL, OR SINGLE LEGAL ENTITY) AND ARM LIMITED ("ARM") FOR THE USE OF ARM'S INTELLECTUAL PROPERTY (INCLUDING, WITHOUT LIMITATION, ANY COPYRIGHT) IN THE RELEVANT AMBA SPECIFICATION ACCOMPANYING THIS LICENCE. ARM LICENSES THE RELEVANT AMBA SPECIFICATION TO YOU ON CONDITION THAT YOU ACCEPT ALL OF THE TERMS IN THIS LICENCE. BY CLICKING "I AGREE" OR OTHERWISE USING OR COPYING THE RELEVANT AMBA SPECIFICATION YOU INDICATE THAT YOU AGREE TO BE BOUND BY ALL THE TERMS OF THIS LICENCE.

"LICENSEE" means You and your Subsidiaries.

"Subsidiary" means, if You are a single entity, any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by You. A company shall be a Subsidiary only for the period during which such control exists.

1. Subject to the provisions of Clauses 2, 3 and 4, Arm hereby grants to LICENSEE a perpetual, non-exclusive, non-transferable, royalty free, worldwide licence to:

   (i) use and copy the relevant AMBA Specification for the purpose of developing and having developed products that comply with the relevant AMBA Specification;

   (ii) manufacture and have manufactured products which either: (a) have been created by or for LICENSEE under the licence granted in Clause 1(i); or (b) incorporate a product(s) which has been created by a third party(s) under a licence granted by Arm in Clause 1(i) of such third party's AMBA Specification Licence; and

   (iii) offer to sell, sell, supply or otherwise distribute products which have either been (a) created by or for LICENSEE under the licence granted in Clause 1(i); or (b) manufactured by or for LICENSEE under the licence granted in Clause 1(ii).

2. LICENSEE hereby agrees that the licence granted in Clause 1 is subject to the following restrictions:

   (i) where a product created under Clause 1(i) is an integrated circuit which includes a CPU then either: (a) such CPU shall only be manufactured under licence from Arm; or (b) such CPU is neither substantially compliant with nor marketed as being compliant with the Arm instruction sets licensed by Arm from time to time;

   (ii) the licences granted in Clause 1(iii) shall not extend to any portion or function of a product that is not itself compliant with part of the relevant AMBA Specification; and

   (iii) no right is granted to LICENSEE to sublicense the rights granted to LICENSEE under this Agreement.

3. Except as specifically licensed in accordance with Clause 1, LICENSEE acquires no right, title or interest in any Arm technology or any intellectual property embodied therein. In no event shall the licences granted in accordance with Clause 1 be construed as granting LICENSEE, expressly or by implication, estoppel or otherwise, a licence to use any Arm technology except the relevant AMBA Specification.

4. THE RELEVANT AMBA SPECIFICATION IS PROVIDED "AS IS" WITH NO REPRESENTATION OR WARRANTIES EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF SATISFACTORY QUALITY, MERCHANTABILITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE, OR THAT ANY USE OR IMPLEMENTATION OF SUCH ARM TECHNOLOGY WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.

5. NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS AGREEMENT, TO THE FULLEST EXTENT PETMITTED BY LAW, THE MAXIMUM LIABILITY OF ARM IN AGGREGATE FOR ALL CLAIMS MADE AGAINST ARM, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS AGREEMENT (INCLUDING WITHOUT LIMITATION (I) LICENSEE'S USE OF THE ARM TECHNOLOGY; AND (II) THE IMPLEMENTATION OF THE ARM TECHNOLOGY IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS AGREEMENT) SHALL NOT EXCEED THE FEES PAID (IF ANY) BY LICENSEE TO ARM UNDER THIS AGREEMENT. THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

6. No licence, express, implied or otherwise, is granted to LICENSEE, under the provisions of Clause 1, to use the Arm tradename, or AMBA trademark in connection with the relevant AMBA Specification or any products based thereon. Nothing in Clause 1 shall be construed as authority for LICENSEE to make any representations on behalf of Arm in respect of the relevant AMBA Specification.

7. This Licence shall remain in force until terminated by you or by Arm. Without prejudice to any of its other rights if LICENSEE is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to You. You may terminate this Licence at any time. Upon expiry or termination of this Licence by You or by Arm LICENSEE shall stop using the relevant AMBA Specification and destroy all copies of the relevant AMBA Specification in your possession together with all documentation and related materials. Upon expiry or termination of this Licence, the provisions of clauses 6 and 7 shall survive.

8. The validity, construction and performance of this Agreement shall be governed by English Law.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

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

Web Address

`http://www.arm.com`

Contents
AMBA Low Power Interface Specification

Preface

This preface introduces the *AMBA Low Power Interface Specification*. It contains the following sections:

About this specification

This specification describes the interfaces that can be used to control the clock and power states of devices. The supported interfaces are Q-Channel and P-Channel.

Intended audience

This specification is written for hardware and software engineers who want to become familiar with the *Advanced Microcontroller Bus Architecture (AMBA)* and engineers who design systems and modules that are compatible with the *AMBA Low Power Interface*.

Using this specification

This specification is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the AMBA low-power interfaces.

Chapter 2 *Q-Channel Interface Specification*

Describes the Q-Channel low-power interface. Q-Channel is an evolution of the AXI low-power interface and is backward-compatible in most situations.

Chapter 3 *P-Channel Interface Specification*

Describes the P-Channel low-power interface.

Appendix A *Revisions*

Read this for a description of the revisions of this specification.

Conventions

The following sections describe conventions that this specification can use:

- *Typographic conventions* on page ix
- *Timing diagrams* on page ix
- *Signals* on page ix
- *Numbers* on page ix

Typographic conventions

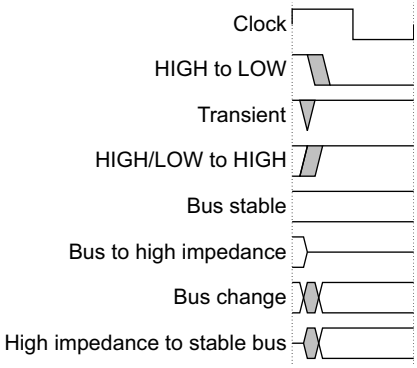
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, and denotes internal cross-references and citations.
bold	Denotes signal names, and is used for terms in descriptive lists, where appropriate.
`monospace`	Used for assembler syntax descriptions, pseudocode, and source code examples. Also used in the main text for instruction mnemonics and for references to other items appearing in assembler syntax descriptions, pseudocode, and source code examples.
SMALL CAPITALS	Used for a few terms that have specific technical meanings.

Timing diagrams

The components used in timing diagrams are explained in the figure *Key to timing diagram conventions*. Variations have clear labels, when they occur. Do not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change shown in *Key to timing diagram conventions*. If a timing diagram shows a single-bit signal in this way, then its value does not affect the accompanying description.

Signals

The signal conventions are:

Signal level	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means: • HIGH for active-HIGH signals • LOW for active-LOW signals
Lowercase n	The start or end of a signal name denotes an active-LOW signal.

Numbers

Numbers are normally written in decimal. Binary numbers are preceded by `0b`, and hexadecimal numbers by `0x`. Both are written in a `monospace` font.

Additional reading

This section lists relevant publications from Arm.

See Arm Developer, https://developer.arm.com/documentation for access to Arm documentation.

Arm publications

- *AMBA AXI Protocol Specification* (ARM IHI 0022)

Feedback

Arm welcomes feedback on this product and its documentation.

Feedback on content

If you have comments on the content of this specification, send an email to `errata@arm.com`. Give:

- The title, *AMBA Low Power Interface Specification*.
- The number, ARM IHI 0068D.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests PDFs only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the appearance or behavior of any document when viewed with any other PDF reader.

Progressive terminology 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. Previous issues of this document included terms that can be offensive. We have replaced these terms.

If you find offensive terms in this document, please contact `terms@arm.com`.

Chapter 1
Introduction

The following sections introduce the low-power interfaces, and describe the differences between them.

1.1 About the low-power interfaces

This document describes interfaces that can be used to control the clock and power states of devices. The interfaces are:

Q-Channel Intended for use where simple run-stop quiescence semantics are suitable. Q-Channel is an evolution of the AXI low-power interface, and its naming reflects the fact that the purpose of the interface is the control of device quiescence.

P-Channel Intended for management of complex power scenarios with multiple transitions. These transitions can be made without returning the device to a common operable state. Its naming reflects the fact that the purpose of the interface is the control of power state transitions.

Note

Each interface has optional Interface Parity Protection. The Interface Parity Protection provides the ability to detect, diagnose, and safely mitigate potential faults on the interfaces.

1.1.1 Common characteristics

The Q-Channel and P-Channel interfaces have the following common characteristics:

- Controller-managed transitions between device states.
- A device can:
  - Indicate that it must exit a lower-power state and enter a higher-functioning state.
  - Hint that it might accept a request to enter a lower-power state.
- Optionally, a device can deny a state change request.
- Clock domain crossing semantics are robust to support asynchronous interfacing.

Although the interfaces are intended for clock and power state control, there are no restrictions on the specific characteristics of the states that the interfaces manage.

1.2 Interface differences and selection

The two interfaces have been designed to complement each other. The selection of interface is dependent on the needs of the implementation.

1.2.1 Q-Channel

The simple run-stop quiescence semantics of Q-Channel are ideal for clock control or simple power control scenarios. The quiescent state entered can vary, but only according to the common configuration of both the device and controller before entering the quiescent state. The quiescent state exit transition is always to a common operable running state that must be entered before the device can enter a different quiescent state.

1.2.2 P-Channel

P-Channel can handle more complex scenarios. Multiple transitions can be made without returning the device, or device partition, to a common operable state between successive power-saving states. The controller provides an explicit state request at each transition, so that common configuration of the controller and device is not needed. The P-Channel interface should be used only if Q-Channel functionality is insufficient.

Chapter 2
Q-Channel Interface Specification

This chapter describes the Q-Channel low-power interface. It includes the following sections:

2.1 Q-Channel interface specification

Q-Channel is an evolution of the AXI low-power interface and is backward-compatible in most situations. For more information, see *Q-Channel backwards compatibility* on page 2-37.

Q-Channel simplifies clock domain crossing by making the handshake mechanism independent of the device activity indication.

2.1.1 Q-Channel signals

Figure 2-1 shows the signals between a device and a clock or power controller:

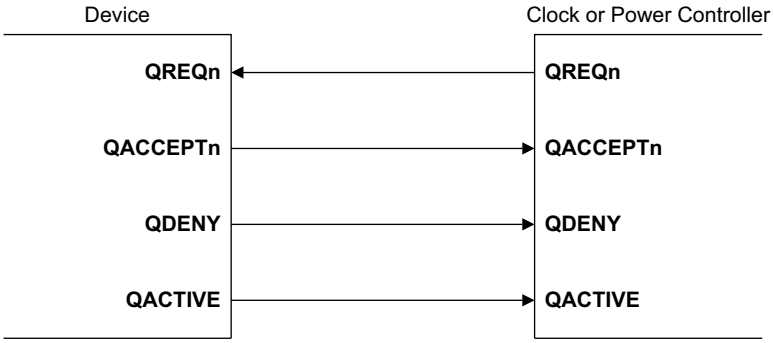


Figure 2-1 Q-Channel device and controller signaling

The Q-Channel interface has the following independent signal groups.

Device activity indication

This group comprises a single signal, **QACTIVE** that can be driven HIGH by a device in any state to indicate that it has operations to perform. When **QACTIVE** is driven LOW by a device, then it is a hint, not a guarantee that the device might accept a quiescence request.

The **QACTIVE** signal from a device can be composed from a number of source signals. To provide wakeup capabilities, these source signals can include device input signals. The final **QACTIVE** signal is driven either directly by a register or by one or more registers and input signals whose outputs are logically combined. *Q-Channel implementation* on page 2-31 gives implementation recommendations for **QACTIVE** combining logic.

If the device itself is unable to assert **QACTIVE** HIGH in the absence of the clock or power supply managed according to the interface, then there must be a system-dependent method to facilitate wakeup outside of the device. This could be a **QACTIVE** from another device combined at the controller with the device **QACTIVE**.

Handshake mechanism

This group manages device quiescence and guarantees safe state transitions. The handshake interface has:

- A quiescence request signal, **QREQn** driven by the controller.
- An acknowledgment signal pair, **QACCEPTn** and **QDENY**, which is driven back to the controller by the device to indicate acceptance or denial of a request. The acknowledgment signals are organized such that only one of them changes per handshake transition. This ensures that the interface can be implemented safely across asynchronous boundaries. **QACCEPTn** is used to accept a request. **QDENY** is used to deny a request.

The **QACCEPTn** and **QDENY** signals from a device and the **QREQn** signal from a controller must all be driven by registers.

The purpose of the denial mechanism is to enable a device to maintain an operational state while also having a mechanism by which it can promptly complete the handshake.

The polarities of the handshake signals are organized to provide a quiescent state where all interface signals are LOW. This facilitates simple default isolation rules.

The handshake signal states are independent of the state of **QACTIVE**. Therefore, transitions on **QACTIVE** are not restricted by the values on **QREQn** or on the **QACCEPTn** and **QDENY** output pair.

The controller can guarantee clock supply or power availability according to the handshake interface state. *Q-Channel handshake* describes these guarantees.

All signals are assumed to be asynchronous.

2.1.2 Q-Channel handshake

This section describes the permitted variations of the Q-Channel interface handshake.

Accepted quiescence request

Figure 2-2 shows a handshake sequence for an accepted quiescence request. It includes the guaranteed activity of an optional controller-supplied clock that is managed according to the interface semantics.

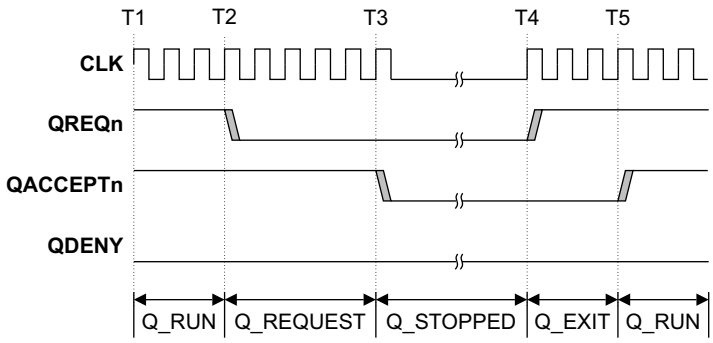


Figure 2-2 Q-Channel accepted quiescence request sequence

Figure 2-2 omits **QACTIVE** because although **QACTIVE** can act as a stimulus for the controller to change interface state, it is independent of the handshake. All interface state changes can be initiated by the controller alone. *Controller policy and QACTIVE* on page 2-23 describes the use of **QACTIVE**.

The transitions shown in Figure 2-2 are:

- At T1, **QREQn** and **QACCEPTn** are both HIGH. This state is referred to as Q_RUN and the device is operational. **QDENY** is LOW in Q_RUN. In this state, the controller guarantees the availability of any clock or power supply that is managed by the interface.
- At T2, **QREQn** is driven LOW by the controller, requesting entry to a quiescent state. This state is referred to as Q_REQUEST. The device remains operational. In this state, the controller guarantees the availability of any clock or power supply that is managed by the interface.
- At T3, the device responds to the quiescence request by driving **QACCEPTn** LOW. **QDENY** remains LOW. This state is referred to as Q_STOPPED. The device is not operational. In this state, the controller does not guarantee the availability of any clock or power supply that is managed by the interface.
- At T4, the controller drives **QREQn** HIGH. Both acknowledgment signals remain LOW. This state is referred to as Q_EXIT. Any clock or power supply managed by the interface is guaranteed after an implementation-dependent delay, but once supplied, is guaranteed to be provided.
- At T5, the device responds to the controller with **QACCEPTn** HIGH, and **QDENY** remains LOW. The interface has returned to the state Q_RUN as at T1.

Denied quiescence request

Figure 2-3 shows a handshake sequence for a denied quiescence request. It includes the guaranteed activity of an optional controller-supplied clock that is managed according to the interface semantics.

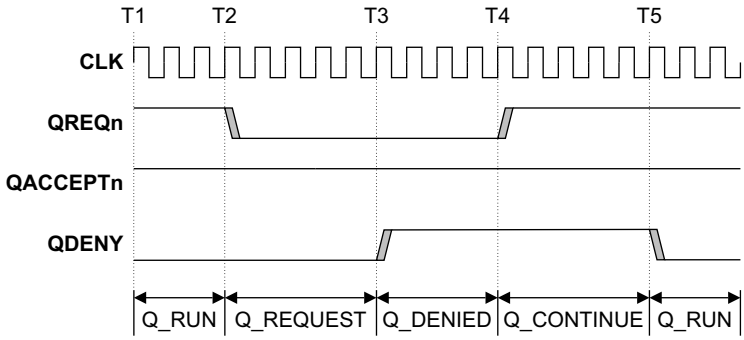


Figure 2-3 Q-Channel denied quiescence request sequence

The transitions shown in Figure 2-3 are:

- The sequence from T1 to T2 is identical to that described in *Accepted quiescence request* on page 2-19.
- At T3, the device drives **QDENY** HIGH while **QACCEPTn** remains HIGH. This state is referred to as Q_DENIED. The device remains operational and the controller must guarantee any clock or power supply managed by the interface.
- At T4, the controller drives **QREQn** HIGH. This state is referred to as Q_CONTINUE and is in response to the quiescence request denial at T3. The device remains operational and the controller must guarantee any clock or power supply managed by the interface.
- At T5, the device drives **QDENY** LOW. The interface has returned to the state Q_RUN, as at T1.

Device reset

At reset assertion, a device must drive both **QACCEPTn** and **QDENY** LOW. **QACTIVE** can reset LOW or HIGH. If the device must perform start-up operations on exit from reset, then it can reset **QACTIVE** HIGH, otherwise it is recommended that **QACTIVE** is reset LOW.

A controller can release a device from reset with either:

- **QREQn** LOW with the interface in Q_STOPPED state.
- **QREQn** HIGH with the interface in Q_EXIT state, provided any clock or power supply guarantee is met.

A controller must only assert a device reset when the interface is in the Q_STOPPED state or when the controller and device are reset simultaneously. This is consistent with the recommendation to isolate all signals LOW at power boundaries.

Figure 2-4 shows a reset exit sequence into the Q_STOPPED state with **QREQn** LOW. At some time after reset deassertion, the interface progresses to Q_RUN, possibly in response to a **QACTIVE** assertion. It then stays active for a time before reentering the quiescent Q_STOPPED state after which reset is asserted.

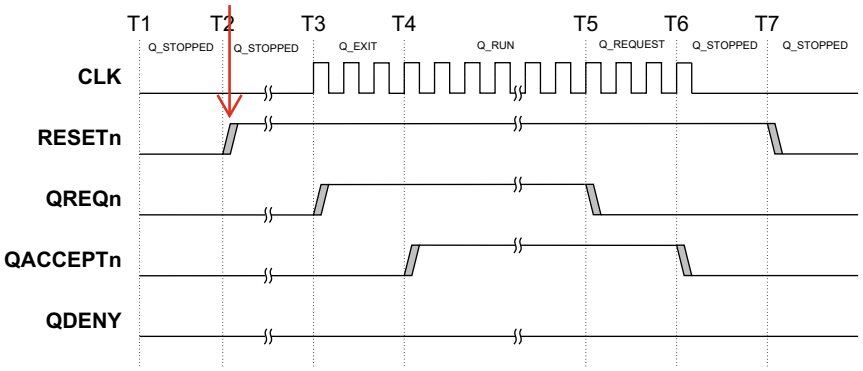


Figure 2-4 QREQn LOW at reset deassertion

Figure 2-5 shows a reset exit sequence into the Q_EXIT state with **QREQn** HIGH. Once the reset is released, the interface responds to the **QREQn** HIGH signal and progresses to Q_RUN. It then stays active for a time before reentering the quiescent Q_STOPPED state after which reset is asserted.

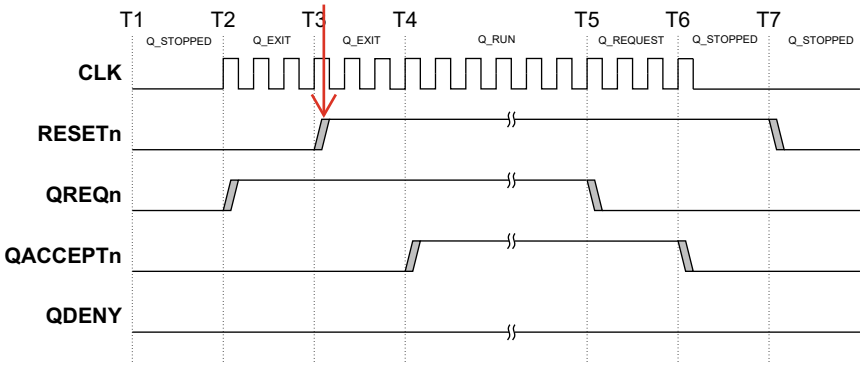


Figure 2-5 QREQn HIGH at reset deassertion

Interface States

Table 2-1 summarizes the interface states and device availability. If a device does not implement a denial mechanism, then **QDENY** is tied LOW or absent, and the first four states represent the complete set.

Table 2-1 Q-Channel interface states

QREQn	QACCEPTn	QDENY	Interface state	Description
1	1	0	Q_RUN	Device is operational.
0	1	0	Q_REQUEST	Device is operational, but requested to become quiescent.
0	0	0	Q_STOPPED	Device has entered the quiescent state. In this state, the controller does not guarantee the availability of any clock or power supply managed by the interface.
1	0	0	Q_EXIT	Supply of clock or power managed by the interface is guaranteed after an implementation-dependent delay. When the device acknowledges by deasserting **QACCEPTn** HIGH, the state moves to Q_RUN.
0	1	1	Q_DENIED	Device denies the request to become quiescent and remains operational. Controller must deassert **QREQn**.
1	1	1	Q_CONTINUE	Controller deasserts **QREQn** HIGH after Q_DENIED. Device is operational.
x	0	1	Unused (illegal)	-

Figure 2-6 is a state diagram showing the possible handshake sequences in terms of the signal states and interface states.

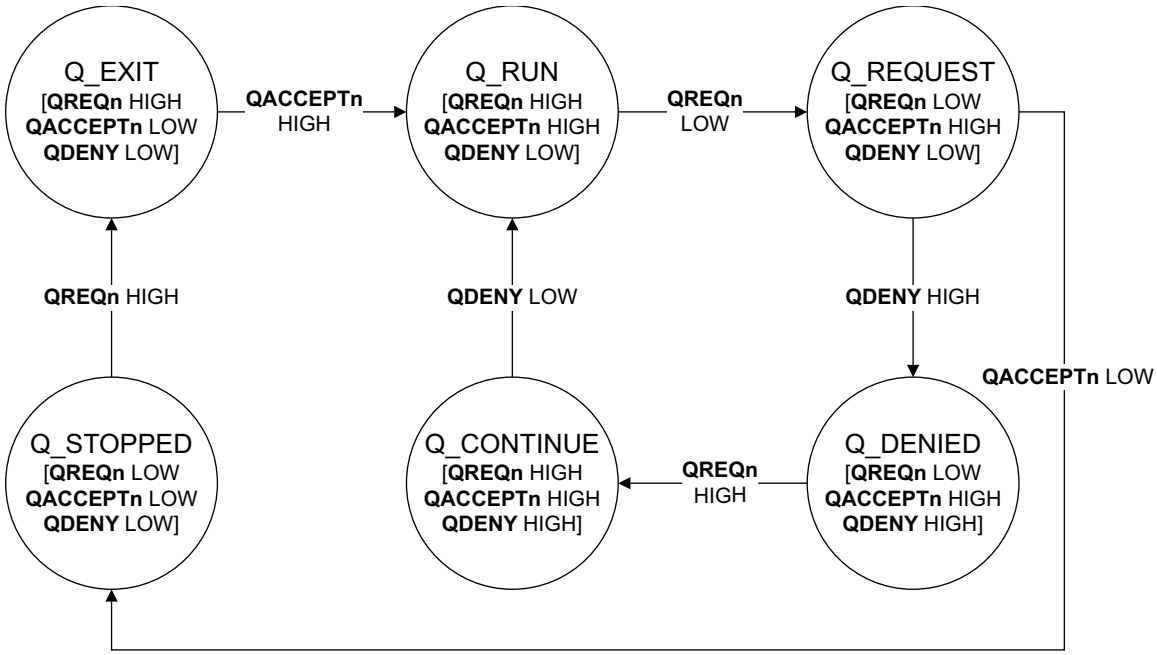


Figure 2-6 Q-Channel states

Handshake rules

The handshake signaling rules are:

- **QREQn** can only transition from HIGH to LOW when **QACCEPTn** is HIGH and **QDENY** is LOW.
- **QREQn** can only transition from LOW to HIGH when either:
  - **QACCEPTn** and **QDENY** are both LOW.
  - **QACCEPTn** and **QDENY** are both HIGH.
- **QACCEPTn** can only transition from HIGH to LOW when **QREQn** is LOW and **QDENY** is LOW.
- **QACCEPTn** can only transition from LOW to HIGH when **QREQn** is HIGH and **QDENY** is LOW.
- **QDENY** can only transition from HIGH to LOW when **QREQn** is HIGH and **QACCEPTn** is HIGH.
- **QDENY** can only transition from LOW to HIGH when **QREQn** is LOW and **QACCEPTn** is HIGH.

2.1.3 Controller policy and QACTIVE

A controller can make any policy decision concerning its management of **QREQn** irrespective of any activity on **QACTIVE**. However, this section describes Q-Channel policies that provide useful solutions.

Asserting **QACTIVE** HIGH can be used as a stimulus for the controller to exit the Q_STOPPED state. The controller responds by driving **QREQn** HIGH exiting the quiescent state.

Note

In normal operation, this is required to allow the device to perform operations. Failure to implement the policy might lead to a system deadlock.

Detecting **QACTIVE** LOW can be used by a controller in the Q_RUN state as a criterion for initiating a quiescence request. However, the controller can change the state of **QREQn** from HIGH to LOW at any time while it is in the Q_RUN state.

Once **QREQn** is driven LOW, the controller does not have to consider the state of **QACTIVE** because **QREQn** cannot be driven HIGH until the handshake is completed by the device with either an acceptance or denial response.

QACTIVE policy example

Figure 2-7 shows a controller policy led by device transitions on **QACTIVE**. When the interface is in Q_STOPPED state, a HIGH level on **QACTIVE** stimulates an exit from the quiescent state. When the interface is in a Q_RUN state, a LOW level on **QACTIVE** causes the controller to make a quiescence request.

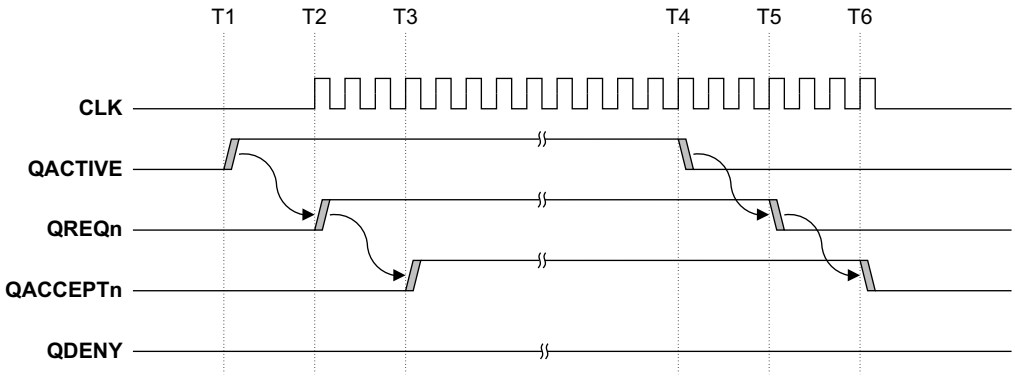


Figure 2-7 Device-led Q_EXIT and Q_REQUEST

2.1.4 Signal subsets

This section describes the permitted signal subsets.

Unused interface

An unused interface must have the **QREQn** input tied HIGH if the device is to be operational. The system has full responsibility for managing the availability of the device by means outside the interface control.

Omission of QDENY

A device that has no requirement to deny a quiescence request can omit **QDENY**. This subset also offers backward compatibility with the AXI low-power interface specification for devices that have no requirement to deny a quiescence request. *Q-Channel backwards compatibility* on page 2-37 describes this backwards compatibility. In this case, **QDENY** must be tied LOW at the controller.

Omission of QACTIVE

In some applications, the initiation of or exit from device quiescence might not require any information from the device. In this case, the device can omit **QACTIVE**. **QACTIVE** must be tied LOW at the controller.

QACTIVE-only interface

A device can present a minimum interface comprising only **QACTIVE** to indicate a requirement to perform operations without any associated handshake. Typically this minimum interface might be used to provide an initial wakeup indication. However, it does not provide any means to guarantee any clock or power availability.

The indication provided by **QACTIVE** alone must be combined with other arrangements of either hardware, software, or both, to provide working solutions. One arrangement might be to permit a device attached to a controller through a **QACTIVE**-only interface to wake a second device, which itself has a Q-Channel interface to the same controller used to guarantee clock or power availability.

2.2 Parity extended Q-Channel interface

The parity extended Q-Channel interface is an optional addition to the Q-Channel interface that allows single bit faults on the interface to be detected and mitigated.

2.2.1 Parity extended Q-Channel signals

In a parity extended Q-Channel, each signal in the standard Q-Channel interface has an associated check signal. Table 2-2 shows the check signal for each standard Q-Channel signal.

Table 2-2 Parity extended Q-Channel check signal relationships

Standard Q-Channel signal	Associated check signal
QACTIVE	QACTIVECHK
QREQn	QREQCHK
QACCEPTn	QACCEPTCHK
QDENY	QDENYCHK

Each check signal provides odd parity checking for its associated signal. Figure 2-8 shows the signal direction for the parity extended Q-Channel.

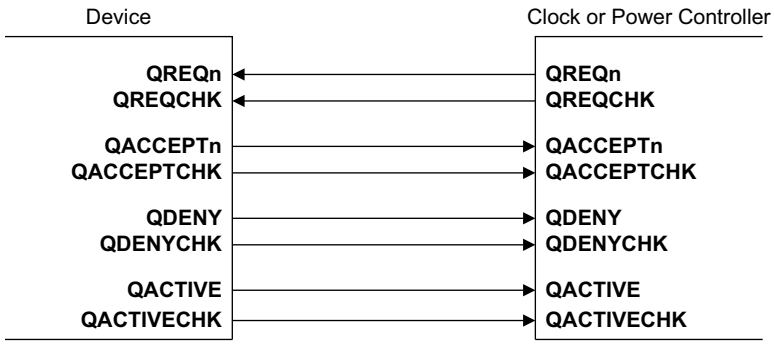


Figure 2-8 Parity extended Q-Channel signaling

There is no timing requirement between the arrival of a Q-Channel signal and its associated check signal. However, the arrival times should be bounded by an implementation to provide a window for fault checking of the signals. The size of this window is IMPLEMENTATION DEFINED.

2.2.2 Parity extended Q-Channel handshake

This section describes the parity extended Q-Channel interface handshake.

The diagrams in this section show each Q-Channel signal and the associated check signal changing simultaneously. However, this is not a requirement and shown only for the convenience of the schematics. Each Q-Channel signal and its associated check signal can change at different times. The Q-Channel state will not progress until both the Q-Channel signal and the associated check signal are in their required state.

Accepted quiescence request

Figure 2-9 on page 2-26 shows a handshake sequence for an accepted quiescence request followed by an exit from quiescence.

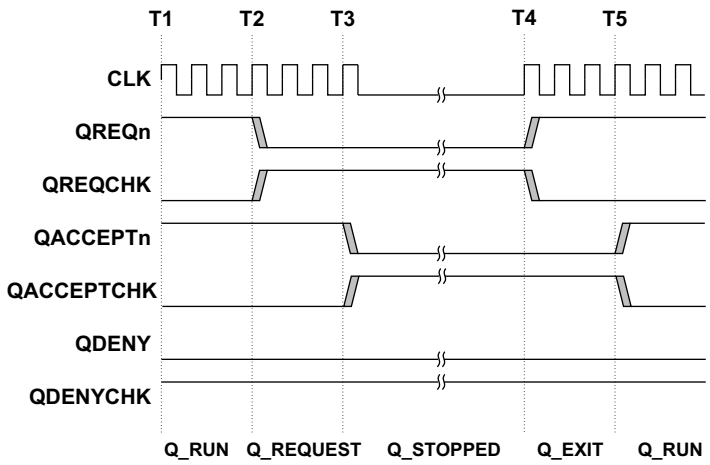


Figure 2-9 Parity extended Q-Channel accepted quiescence request

Figure 2-9 omits **QACTIVE** and **QACTIVECHK** because they are independent of the handshake. These signals can act as a stimulus for the controller to change the interface state. However, all interface state changes can be initiated by the controller alone. For more information, see *Controller policy and QACTIVE* on page 2-23.

The transitions shown in Figure 2-9 are:

T1 The Q-Channel is in Q_RUN and the device is operational.

T2 The controller drives **QREQn** LOW and **QREQCHK** HIGH to request entry to a quiescent state, the Q-Channel progresses to Q_REQUEST. The device remains operational.

T3 The device responds to the quiescence request by driving **QACCEPTn** LOW and **QACCEPTCHK** HIGH, the Q-Channel progresses to Q_STOPPED. The device is not operational. In this state, the controller does not guarantee the availability of any resource that is managed by the interface.

T4 The controller drives **QREQn** HIGH and **QREQCHK** LOW to request exit from the quiescent state, the Q-Channel enters Q_EXIT. Any resource managed by the interface is guaranteed after an implementation-dependent delay.

T5 The device responds to the quiescent exit request by driving **QACCEPTn** HIGH and **QACCEPTCHK** LOW, the Q-Channel enters Q_RUN. The interface has returned to the state it was in at T1.

Device reset

At reset assertion a device must drive:

- **QACCEPTn** LOW and **QACCEPTCHK** HIGH.
- **QDENY** LOW and **QDENYCHK** HIGH.

It is recommended to reset **QACTIVE** LOW and **QACTIVECHK** HIGH. If the device must perform start-up operations on exit from reset, then it can reset **QACTIVE** HIGH and **QACTIVECHK** LOW. A controller can release a device from reset with either:

- **QREQn** LOW and **QREQCHK** HIGH, with the interface in Q_STOPPED state.
- **QREQn** HIGH and **QREQCHK** LOW, with the interface in Q_EXIT state provided any resource guarantee is met.

A controller must only assert a device reset when the interface is in the Q_STOPPED state or when both controller and device are reset simultaneously.

Interface states

Table 2-3 summarizes the interface states and device availability. If a device does not implement a denial mechanism, then **QDENY** and **QDENYCHK** can be present or omitted.

If present and denial is not implemented, then **QDENY** must be tied LOW and **QDENYCHK** must be tied HIGH.

If denial is not implemented, then rows in Table 2-3 with either **QDENY** HIGH or **QDENYCHK** LOW are illegal.

Table 2-3 Parity extended Q-Channel interface states

	Interface state						
Interface state	**QREQn**	**QREQCHK**	**QACCEPTn**	**QACCEPTCHK**	**QDENY**	**QDENYCHK**	**Description**
Q_RUN	1	0	1	0	0	1	Device is operational.
	0	0	1	0	0	1	Signal transition towards Q_REQUEST. Excessive time in this interface state can indicate a fault on either of the **QREQn** or **QREQCHK** signals.
	1	1	1	0	0	1	Signal transition towards Q_REQUEST. Excessive time in this interface state can indicate a fault on either of the **QREQn** or **QREQCHK** signals.
Q_REQUEST	0	1	1	0	0	1	Device is operational but requested to become quiescent.
	0	1	0	0	0	1	Signal transition towards Q_STOPPED. Excessive time in this interface state can indicate a fault on either of the **QACCEPTn** or **QACCEPTCHK** signals.
	0	1	1	1	0	1	Signal transition towards Q_STOPPED. Excessive time in this interface state can indicate a fault on either of the **QACCEPTn** or **QACCEPTCHK** signals.
	0	1	1	0	1	1	Signal transition towards Q_DENIED. Excessive time in this interface state can indicate a fault on either of the **QDENY** or **QDENYCHK** signals.
	0	1	1	0	0	0	Signal transition towards Q_DENIED. Excessive time in this interface state can indicate a fault on either of the **QDENY** or **QDENYCHK** signals.

Table 2-3 Parity extended Q-Channel interface states (continued)

Interface state	Interface state						Description
	QREQn	QREQCHK	QACCEPTn	QACCEPTCHK	QDENY	QDENYCHK	
Q_STOPPED	0	1	0	1	0	1	Device has entered the quiescent state. In this state, the controller does not guarantee the availability of any resource that is managed by the interface.
	0	0	0	1	0	1	Signal transition towards Q_EXIT. Excessive time in this interface state can indicate a fault on either of the **QREQn** or **QREQCHK** signals.
	1	1	0	1	0	1	Signal transition towards Q_EXIT. Excessive time in this interface state can indicate a fault on either of the **QREQn** or **QREQCHK** signals.
Q_EXIT	1	0	0	1	0	1	Supply of interface-managed resource is guaranteed after an IMPLEMENTATION DEFINED delay. When the device acknowledges, by setting **QACCEPTn** HIGH and **QACCEPTCHK** LOW, the state moves to Q_RUN.
	1	0	0	0	0	1	Signal transition towards Q_RUN. Excessive time in this interface state can indicate a fault on either of the **QACCEPTn** or **QACCEPTCHK** signals.
	1	0	1	1	0	1	Signal transition towards Q_RUN. Excessive time in this interface state can indicate a fault on either of the **QACCEPTn** or **QACCEPTCHK** signals.
Q_DENIED	0	1	1	0	1	0	Device denies request to become quiescent and remains operational. Controller must set **QREQn** HIGH and **QREQCHK** LOW. Device is operational.
	0	0	1	0	1	0	Signal transition towards Q_CONTINUE. Excessive time in this interface state can indicate a fault on either of the **QREQn** or **QREQCHK** signals.
	1	1	1	0	1	0	Signal transition towards Q_CONTINUE. Excessive time in this interface state can indicate a fault on either of the **QREQn** or **QREQCHK** signals.
Q_CONTINUE	1	0	1	0	1	0	Device is operational. When the device sets **QDENY** LOW and **QDENYCHK** HIGH, the state moves to Q_RUN.
	1	0	1	0	0	0	Signal transition towards Q_RUN. Excessive time in this interface state can indicate a fault on either of the **QDENY** or **QDENYCHK** signals.
	1	0	1	0	1	1	Signal transition towards Q_RUN. Excessive time in this interface state can indicate a fault on either of the **QDENY** or **QDENYCHK** signals.
Unused	Other States						Illegal

Figure 2-10 is a state diagram showing the possible handshake sequences in terms of the signal states and interface states.

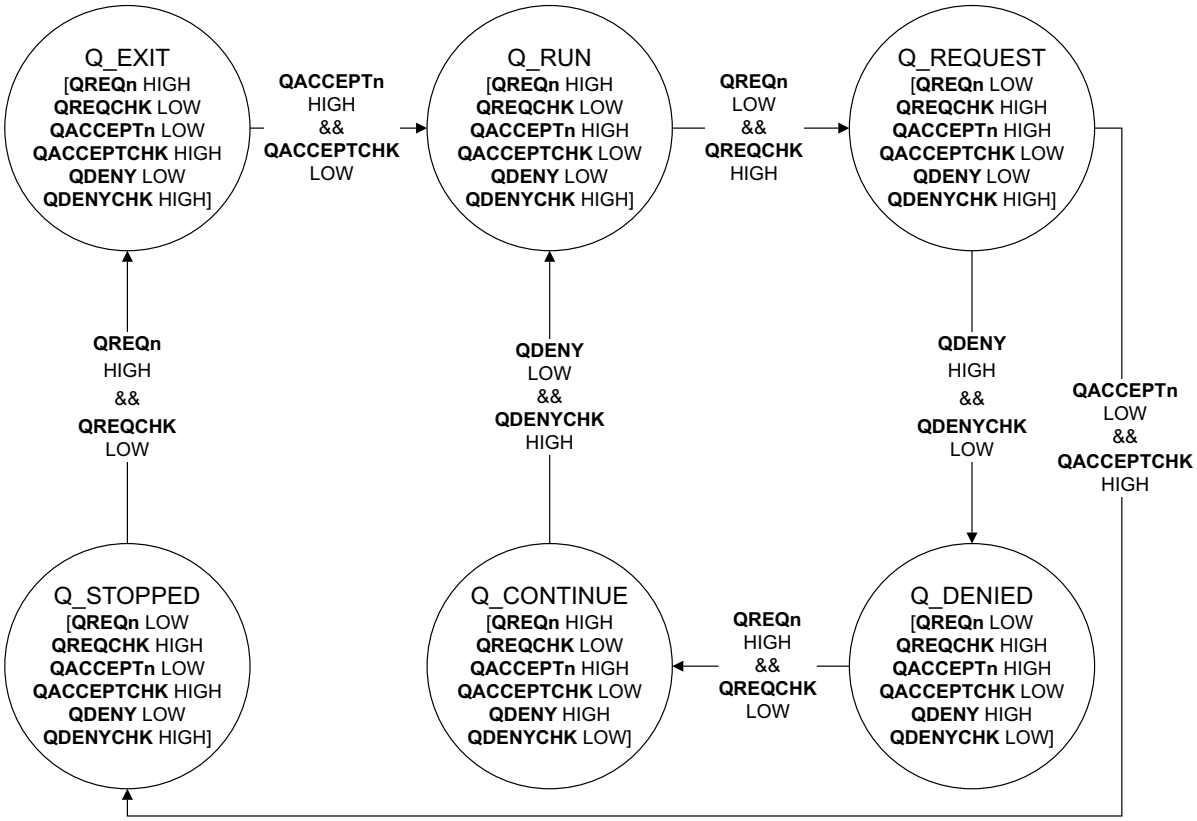


Figure 2-10 Parity extended Q-Channel states

Handshake rules

The handshaking rules are:

- **QREQn** can only transition from HIGH to LOW, and **QREQCHK** can only transition from LOW to HIGH when both:
  - **QACCEPTn** is HIGH and **QACCEPTCHK** is LOW.
  - **QDENY** is LOW and **QDENYCHK** is HIGH.
- **QREQn** can only transition from LOW to HIGH, and **QREQCHK** can only transition from HIGH to LOW when either:

  All the following conditions are true:
  - **QACCEPTn** is LOW and **QACCEPTCHK** is HIGH.
  - **QDENY** is LOW and **QDENYCHK** is HIGH.

  Or all the following conditions are true:
  - **QACCEPTn** is HIGH and **QACCEPTCHK** is LOW.
  - **QDENY** is HIGH and **QDENYCHK** is LOW.

- **QACCEPTn** can only transition from HIGH to LOW, and **QACCEPTCHK** can only transition from LOW to HIGH when both:
  - **QREQn** is LOW and **QREQCHK** is HIGH.
  - **QDENY** is LOW and **QDENYCHK** is HIGH.
- **QACCEPTn** can only transition from LOW to HIGH, and **QACCEPTCHK** can only transition from HIGH to LOW when both:
  - **QREQn** is HIGH and **QREQCHK** is LOW.
  - **QDENY** is LOW and **QDENYCHK** is HIGH.
- **QDENY** can only transition from HIGH to LOW, and **QDENYCHK** can only transition from LOW to HIGH when both:
  - **QREQn** is HIGH and **QREQCHK** is LOW.
  - **QACCEPTn** is HIGH and **QACCEPTCHK** is LOW.
- **QDENY** can only transition from LOW to HIGH, and **QDENYCHK** can only transition from HIGH to LOW when both:
  - **QREQn** is LOW and **QREQCHK** is HIGH.
  - **QACCEPTn** is HIGH and **QACCEPTCHK** is LOW.

Signal subsets

The supported signal subsets are aligned with those of a standard Q-Channel. Where a signal is not present, then its associated check signal is also removed.

An unused device interface must have **QREQn** tied HIGH and **QREQCHK** tied LOW if the device is to be operational. The system has full responsibility for managing the availability of the device by means outside the interface control.

2.3 Q-Channel implementation

Typically a device and a controller are implemented with asynchronous clocks. These might be driven from the same source clock but with a significant phase difference. Figure 2-11 shows the recommended implementation of the resynchronization in this case.

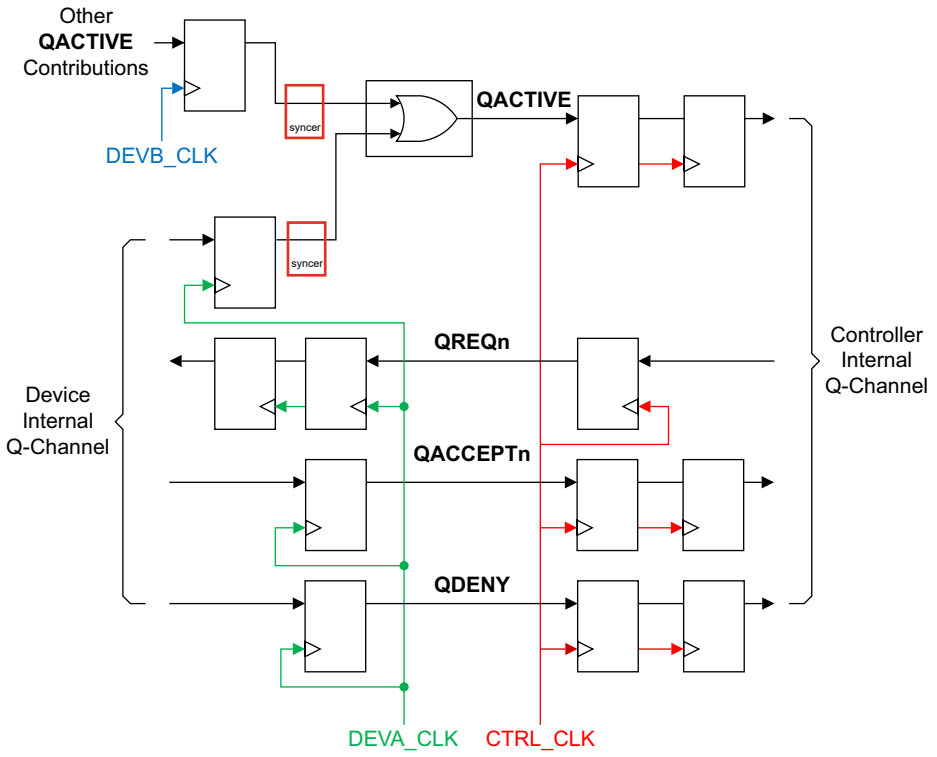


Figure 2-11 Recommended Q-Channel clock domain crossing synchronization

For an asynchronous Q-Channel implementation, it is recommended to provide the following guidance to implementers:

- Resynchronize all signals at the destination before use.
- Only remove clock or power when the interface is in the Q_STOPPED state.
- To ensure operation of the four-phase handshake, register all **QREQn**, **QACCEPTn**, and **QDENY** outputs.
- The **QACTIVE** signal is driven either directly by a register or by a number of registers whose contributions are logically combined. It is recommended that this combining logic is limited to a logical OR where possible, and is implemented using instantiated gates. 直接用cell library
- When other logic is implemented, the implications of **QACTIVE** source register changes on the output **QACTIVE** signal must be carefully considered. Although the handshake protocol guarantees functionally correct behavior regardless of **QACTIVE** activity, it is recommended to implement the simplest possible logic to minimize the likelihood of introduced glitches at the **QACTIVE** output.

Note

Although Figure 2-11 shows a typical two-stage synchronization, resynchronization must be implemented appropriately for the technology libraries used and required frequency targets.

2.4 Connections between parity protected and non-parity protected Q-Channel interfaces

In certain circumstances, it might be required to connect controllers and devices where one has the parity protection and the other does not. The following sections describe the integration requirements for these types of connections. When connecting parity protected and non-parity protected controllers and devices, the interface has no or limited fault protection.

2.4.1 Connecting a non-parity protected Q-Channel controller to a parity protected Q-Channel device

Figure 2-12 shows the integration required between a non-parity protected Q-Channel controller to a parity protected Q-Channel device.

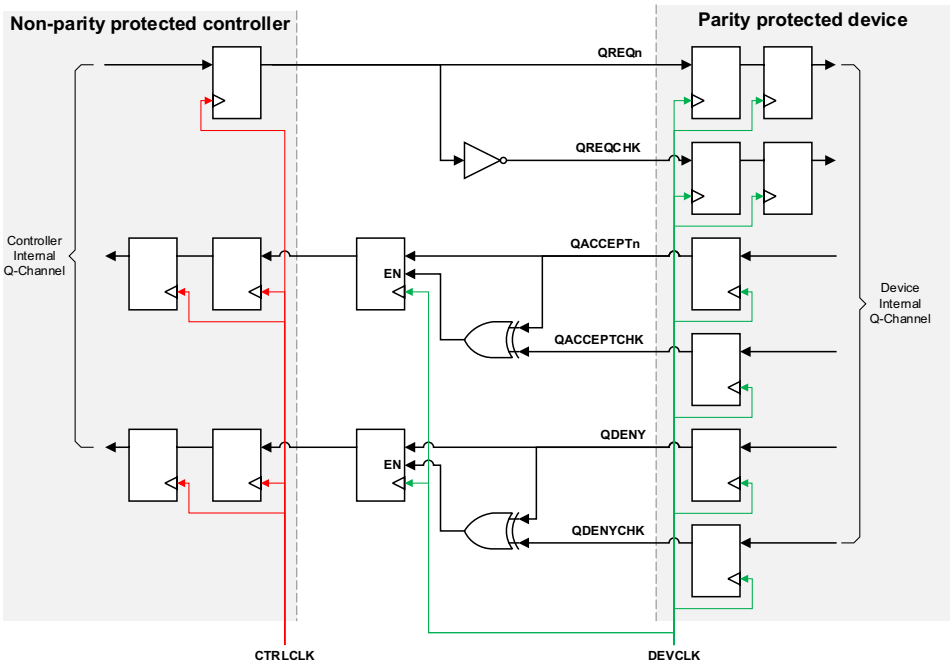


Figure 2-12 Non-parity protected Q-Channel controller to a parity protected Q-Channel device connections

Additional consideration must be taken if the parity protection check signals are generated on a different clock to the standard Q-Channel signals. If so, these signals must be resynchronized using the device clock before being used in the **QACCEPTn** and **QDENY** integration logic shown in Figure 2-12.

QACTIVE

The device **QACTIVE** output can be connected directly to the controller **QACTIVE** input and the device **QACTIVECHK** output left unconnected. This might have the effect of indicating a level, either HIGH or LOW, at the controller before both signals indicate this requirement on the device side.

In this case, the Q-Channel handshake ensures the device maintains a state according to the devices request. However, such behavior might cause unnecessary transitions.

There are alternative integration options that might be used to prevent the propagation of **QACTIVE** until both the standard and check signals have changed. However, the propagation of **QACTIVE** is typically required to be maintained as a combinatorial path to ensure, for example, a request can be made when a local clock is not available.

Adding a non-trivial amount of logic to the **QACTIVE** path might result in glitches. While these glitches will not cause malfunctions due to the handshake guarantees as noted above, they might cause unnecessary transitions.

QREQn

The **QREQCHK** signal is the inversion of the **QREQn** output from the non-parity protected controller.

QACCEPTn and QDENY

The **QACCEPTn** input to the non-parity protected controller is required to be conditioned such that it only updates when both **QACCEPTn** and **QACCEPTCHK** have updated to the new value.

If this conditioning is not present and **QACCEPTn** was connected directly, then a change on **QACCEPTn** when **QACCEPTCHK** has not changed, might result in a change on **QREQn** that might cause a protocol violation at the parity protected device.

This conditioning can be provided by propagating the **QACCEPTn** value only when **QACCEPTn** and **QACCEPTCHK** are different. A register is required to hold the last value because when in transition the **QACCEPTn** and **QACCEPTCHK** signals can be either both HIGH or both LOW regardless of the value to which they are transitioning.

The same provision must also be made for the **QDENY** and **QDENYCHK** signals from the parity protected device to the non-parity protected controller.

2.4.2 Connecting a parity protected Q-Channel controller to a non-parity protected Q-Channel device

Figure 2-13 shows the integration required between a parity protected Q-Channel controller to a non-parity protected Q-Channel device.

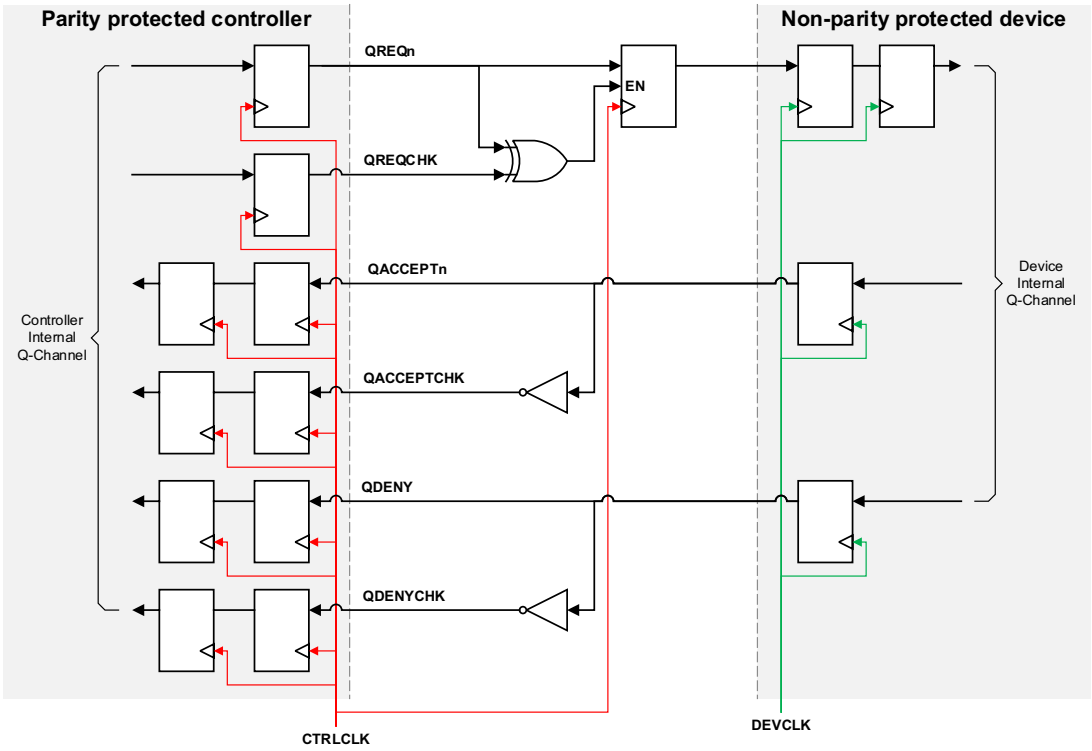


Figure 2-13 Parity protected Q-Channel controller to non-parity protected Q-Channel device connections

Additional consideration must be taken if the parity protection check signals are generated on a different clock to the standard Q-Channel signals. If so, these signals must be resynchronized to the controller clock before being used in the **QREQn** integration logic shown in Figure 2-13.

QREQn

The **QREQn** input to the non-parity protected device is required to be conditioned such that it only updates when both the **QREQn** and **QREQCHK** have updated to the new value.

If this conditioning is not present and the **QREQn** was connected directly, then a change on **QREQn** when **QREQCHK** has not changed, might result in a change on **QACCEPTn** or **QDENY** that might cause a protocol violation at the parity protected controller.

This conditioning can be provided by only propagating the **QREQn** value when **QREQn** and **QREQCHK** are different. A register is required to hold the last value as when in transition the **QREQn** and **QREQCHK** signals can be either both HIGH or both LOW regardless of the value to which they are transitioning.

QACCEPTn, QACTIVE, and QDENY

The **QACTIVECHK**, **QACCEPTCHK**, and **QDENYCHK** inputs for the parity protected controller are generated from an inversion of the **QACTIVE**, **QACCEPTn**, and **QDENY** signals respectively.

2.5 Q-Channel application examples

This section describes a number of applications of the Q-Channel interface.

2.5.1 Clock dependencies in multiple-interface devices

Some devices might require multiple Q-Channel and P-Channel interfaces to manage multiple clock, power, or functional domains of the device. The design and use of devices with multiple interfaces must take into account any dependencies between the states of each interface, and between the guarantees provided by each interface.

The most common dependency when the device is powered on is clock supply to the device because the clock is required to process any operation or state change.

Figure 2-14 shows an example of a device with two interfaces.

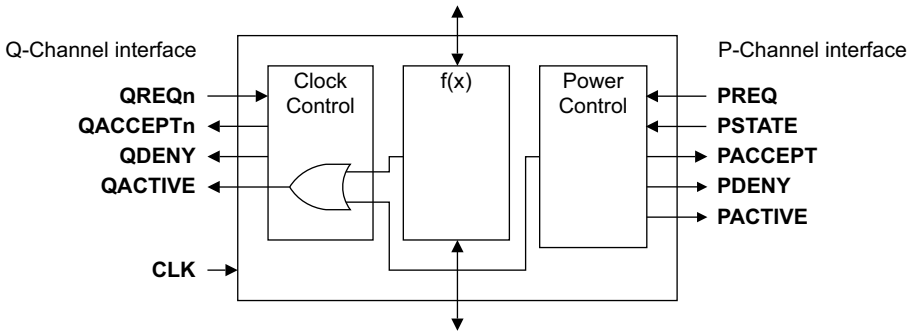


Figure 2-14 Device-level interface dependency example

The device in Figure 2-14 has both a Q-Channel clock control interface and a P-Channel power control interface. The normal function of the device, labeled as the functional block `f(x)`, will place requirements on the activity of **QACTIVE** to ensure availability of **CLK** so that it can process its operations.

When the functional block `f(x)` is not active, with **CLK** inactive and the clock control interface at Q_STOPPED, a power state transition request using **PREQ** and **PSTATE** can still occur. However, **CLK** must be active to perform the management actions in the device that are required before the power state transition and to respond on the power control interface with **PACCEPT** or **PDENY**.

This means there is a dependency between the power and clock control interfaces. In this example, the dependency can be resolved as shown by a **QACTIVE** contribution from the power control interface logic that is active whenever P-Channel transitions require **CLK** to be active. This **QACTIVE** contribution must be a combinatorial combination of the power P-Channel inputs so that **CLK** can be requested when it is not present.

Dependencies between interfaces used for other purposes cannot be rigorously defined and might be resolved according to the application, either by the device design or the usage requirement.

2.5.2 Interface combining for clock control

Combining Q-Channel interfaces from multiple devices within a single clock domain into the ownership of one controller has the following potential advantages:

- A single clock insertion path from a clock controller to multiple synchronous devices eases physical implementation, reducing power consumption.
- When there is a hierarchical relationship between devices, for example in terms of traffic, the handshake latency overhead for devices downstream of the first device can be mitigated.

The Q-Channel interfaces from each device can either be merged at a controller or combined at an intermediate functional block into a single interface, which is then presented to the controller.

In all cases, the solution must retain dedicated and independent handshake interfaces between each device and the controller or combining functional block. This handshake is important as it ensures that performance limitations or potential deadlock scenarios can be avoided in cases where there are interdependencies between devices with a shared clock.

In the example shown in Figure 2-15, the device interfaces are connected to dedicated functional blocks within a controller and the only shared signal is the **CLK** signal supplied by the controller to both devices.

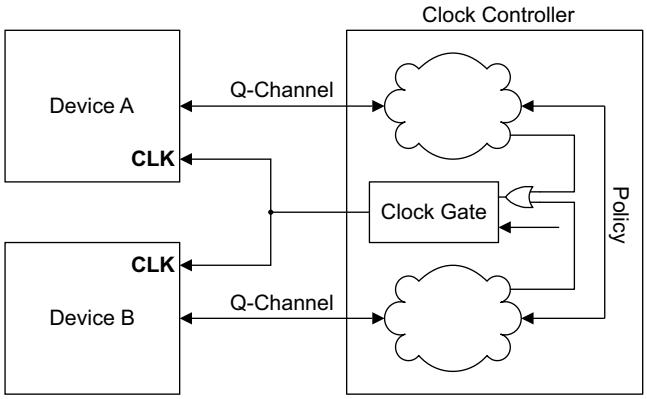


Figure 2-15 Interface combining at clock controller example

In this example, the two functional blocks manage the devices independently with only a simple internal merge of clock enable conditions. This solution can be enhanced by implementing internal policies between the device interface blocks to improve performance, for example by mitigating sequential handshake overhead by combining **QACTIVE** requests from devices or by adding control sequence dependencies.

2.6 Q-Channel backwards compatibility

This section describes connection mappings for ensuring backward compatibility with AXI low-power interface devices and controllers.

Figure 2-16 shows how a device supporting the AXI low-power interface that does not rely on the denial mechanism can be interfaced directly to a Q-Channel controller with an absent or tied LOW QDENY signal.

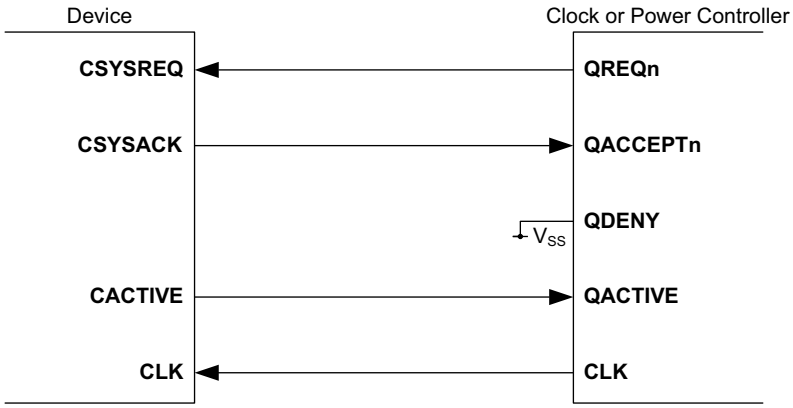


Figure 2-16 Connection of Q-Channel controller to an AXI LPI device without denial support

Figure 2-17 shows how an AXI low-power interface controller component can be connected to a Q-Channel device with the **QDENY** output absent or tied low. In this case, the AXI low-power interface controller must not interpret the denial condition since the activity on **QACTIVE** is not restricted.

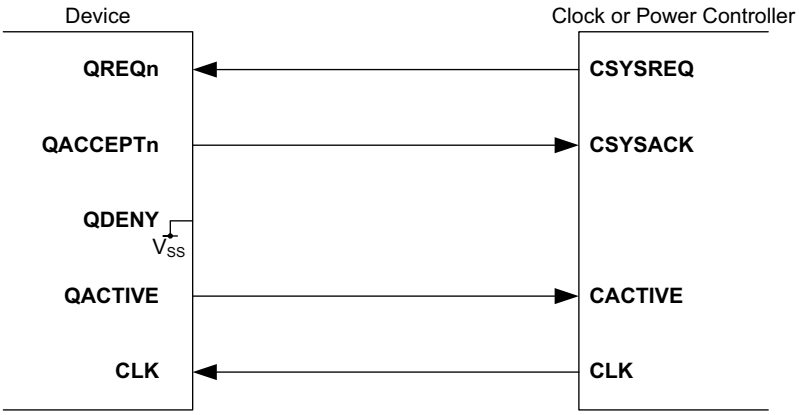


Figure 2-17 Connection of Q-Channel device to an AXI LPI controller, both without denial support

For a device supporting the AXI low-power interface that uses the denial mechanism, solutions are limited to instances where all interface signals are confined to a single synchronous clock domain. Within those restrictions, a simple adapter component to generate the Q-Channel acknowledgment signals can be implemented as shown in Figure 2-18 on page 2-38. This component has a sequential requirement to ensure correct glitch-free handshaking at the controller.

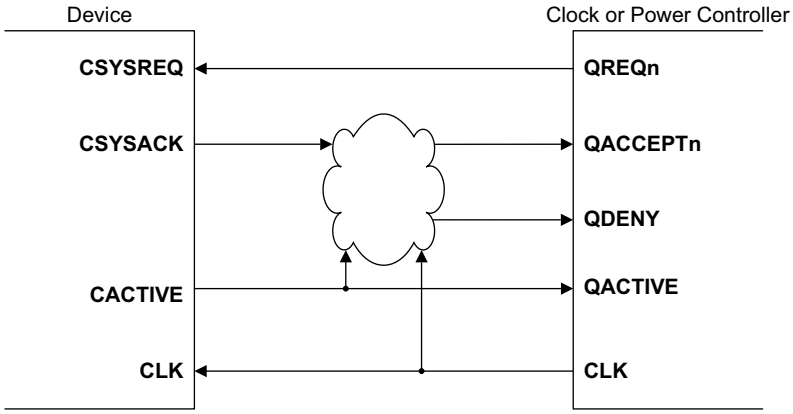


Figure 2-18 Adaptation of AXI LPI device with synchronous denial support to a Q-Channel controller

Successful integration of any other arrangement cannot be guaranteed and would require a case-by-case analysis of the timing relationships, particularly between **CACTIVE** and **CSYSACK**.

Chapter 3
P-Channel Interface Specification

This chapter describes the P-Channel low-power interfaces. It includes the following sections:

3.1 P-Channel interface specification

This section describes the P-Channel interface that controls power state transitions. For the purposes of this specification:

Lower-power state

Typically is a state for which:

- The power consumption is lower.
- The device has less functionality, performance, or state retention than higher states.

Higher-power state

Typically is a state for which:

- The power consumption is higher.
- The device has more functionality, performance, or state retention than lower states.

3.1.1 P-Channel Signals

Figure 3-1 shows the P-Channel signals between a device and a power controller.

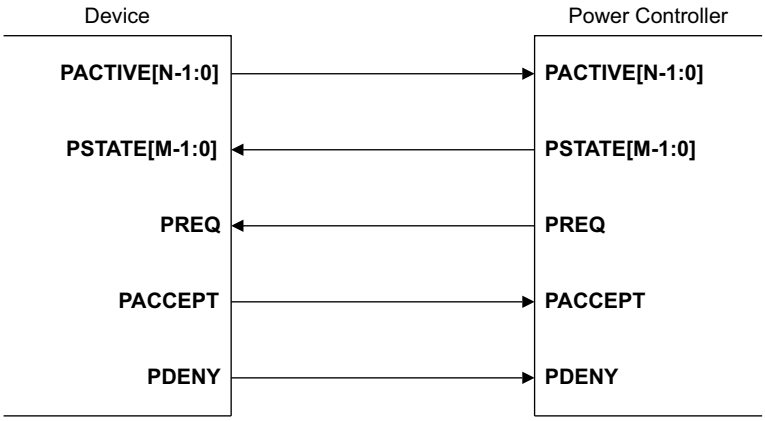


Figure 3-1 P-Channel device and controller signal mappings

A P-Channel interface comprises the following signal groups.

Device activity indication

This group comprises an N-bit signal, **PACTIVE[N-1:0]**:

- Each **PACTIVE** bit presented at a controller can be composed from a number of source signals. To provide wakeup capabilities, these source signals can include device input signals. Each **PACTIVE** bit is driven either directly by a register or by one or more registers and input signals whose outputs are logically combined. *P-Channel implementation* on page 3-57 gives implementation recommendations for **PACTIVE** combining logic.
- All bits of **PACTIVE** are independent of each other and might come from various asynchronous sources. For more information, see *P-Channel implementation* on page 3-57.

Handshake mechanism

This group manages device power state transitions. The group has the following signals:

PSTATE[M-1:0]

The power state to which a transition is requested.

PREQ

Active HIGH request to transition to the power state indicated by **PSTATE**.

PACCEPT

Active HIGH acceptance of the transition to the requested power state.

PDENY

Active HIGH denial of the power state transition request.

PREQ, **PACCEPT**, and **PDENY** form a handshake interface that is used to manage and guarantee safe state transitions. The handshake interface has a transition request signal, **PREQ** driven by a controller and an acknowledgment signal pair, **PACCEPT** and **PDENY**, which are driven by a device to indicate acceptance or denial of a request.

The P-Channel specification requires that only one of **PACCEPT** or **PDENY** changes on a single handshake transition. This ensures that the interface can be implemented safely across asynchronous boundaries. **PACCEPT** is used to accept a request, and **PDENY** is used to deny a request.

The **PACCEPT** and **PDENY** signals from a device and the **PREQ** and **PSTATE** signals from a controller must all be driven directly by registers. PSTATE enumerations are IMPLEMENTATION DEFINED. It is recommended to use the values defined in the Power Policy Unit Specification for compatibility. See *Arm® Power Policy Unit, Architecture Specification, Version 1.1* for more information.

The purpose of the denial mechanism is to enable a device to maintain its current state while also having a mechanism by which it can promptly complete the handshake. This means the controller can make another request, possibly to a different state, if there are changes in the system conditions.

The polarities of the handshake signals are organized so they start and end each transition in a LOW state. This facilitates simple default isolation rules.

The handshake signal states are independent of the **PACTIVE** bits. Transitions on **PACTIVE** bits are not restricted by the values on **PREQ**, **PACCEPT**, and **PDENY**.

The P-Channel interface also has protocol relationships to the device, or device partition, reset. For a device with multiple resets, the reset relationships specified here are for the reset associated with the device P-Channel logic. In this specification, the name **RESETn** is used to represent this reset signal. However, the actual reset naming and polarity is outside the scope of this specification, and not restricted by this specification provided the relationships described are observed. *Device reset and initialization* on page 3-43 describes reset and initialization transitions.

3.1.2 P-Channel handshake

This section describes the permitted variations of the P-Channel interface handshake.

Accepted state transition

Figure 3-2 shows a transition request from State A to State B that is accepted by the device.

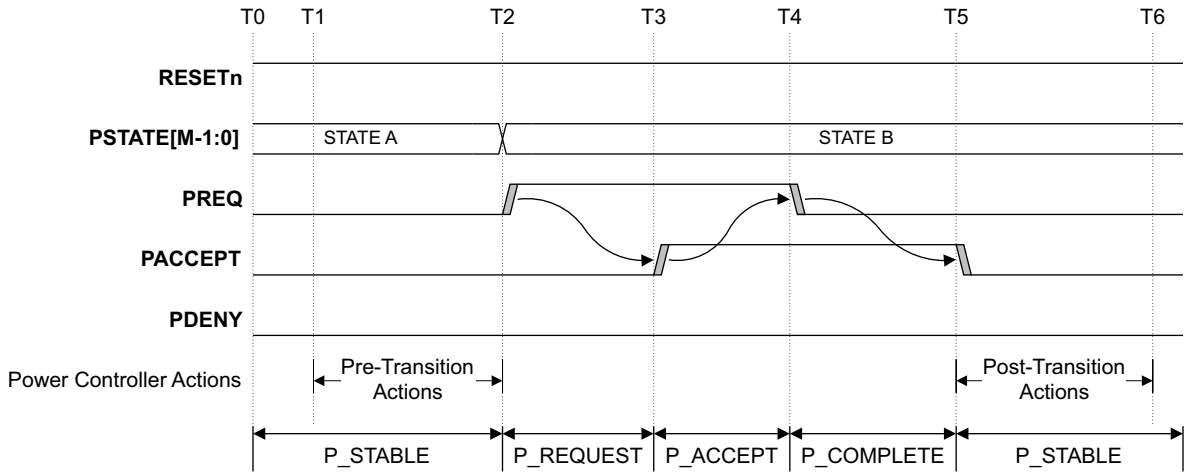


Figure 3-2 P-Channel handshake sequence with accepted request

Figure 3-2 omits **PACTIVE** because although **PACTIVE** can act as a stimulus for the controller to initiate a transition request, it is independent of the handshake. All transition requests can be initiated by the controller alone.

The transitions shown in Figure 3-2 are:

- At T0, the interface is idle and all signals are LOW. The interface state is P_STABLE and the device remains in its current state.
- At T1, recognizing that a handshake request is required, the power controller must take any necessary pre-transition actions prior to requesting a new state. These actions are typically associated with a transition to a higher-power state. This might include powering up a domain or bringing RAM out of the retention state. The interface state remains P_STABLE.
- At T2, the power controller puts the required power state value on **PSTATE** and sets **PREQ** HIGH. The interface state is now P_REQUEST. The protocol requires that **PSTATE** is stable when **PREQ** going HIGH is detected at the device. This is typically achieved by resynchronizing **PREQ** to the device clock domain and using it to capture **PSTATE**.
  - A controller must only present supported **PSTATE** values and transitions to a device. See *Device power state and transition support definition* on page 3-49.
- At T3, the device accepts the transition by driving **PACCEPT** HIGH. **PDENY** must be kept LOW. The device can now use the capabilities of any higher state. The interface state is now P_ACCEPT.
- At T4, the power controller samples **PACCEPT** HIGH and sets **PREQ** LOW. The interface state is P_COMPLETE.
- At T5, the device samples **PREQ** LOW and sets **PACCEPT** LOW. Once the controller samples **PACCEPT** LOW, it can take any post-transition actions required. These actions are typically associated with transitioning to a lower-power state. This might include removing power from a power domain or placing a RAM into a retention state. The transition is now complete and the interface state returns to P_STABLE. When moving to a lower-power state after setting **PACCEPT** LOW, the device cannot assume the availability of any properties of a previous higher-power state.

Note

If the P-Channel transitions to a lower-power state, the availability of any properties of the previous higher-power state can be removed. However, removal is not guaranteed so the device operation should not depend on it, or other associated actions, for example, device reset.

Denied state transition

Figure 3-3 shows a transition request that is denied by the device.

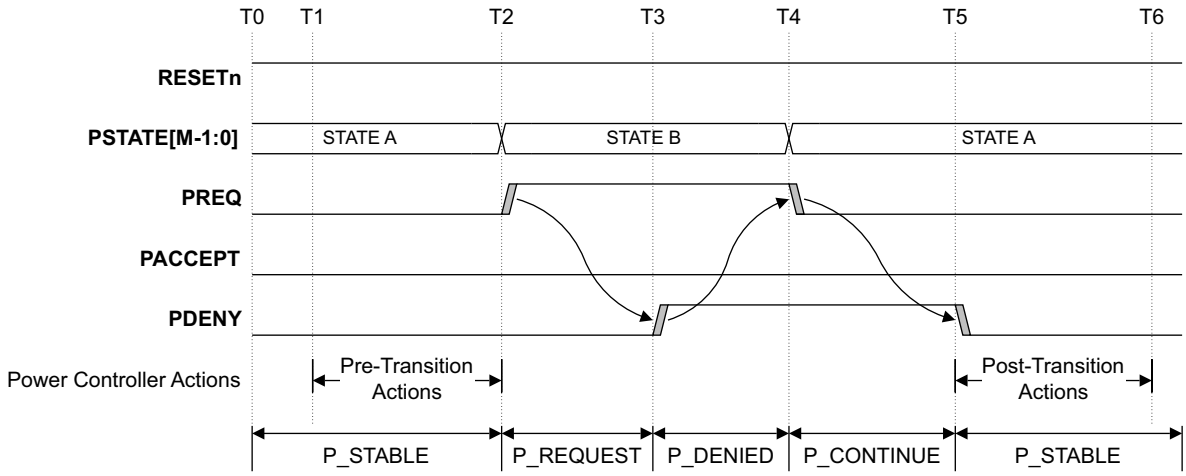


Figure 3-3 P-Channel handshake sequence with denied request

The transitions shown in Figure 3-3 are:

- T0, T1, and T2 follow the same sequence as described in *Accepted state transition* on page 3-42.
- At T3, the device denies the request by setting **PDENY** HIGH. **PACCEPT** must remain LOW. The interface state is P_DENIED.
- At T4, the power controller samples **PDENY** HIGH and sets **PREQ** LOW and returns **PSTATE** to the value for the current state, state A. The protocol requires that **PSTATE** is stable and has the value of the original state when **PREQ** going low is detected at the device. The interface state is P_CONTINUE.
- At T5, the device samples **PREQ** LOW and sets **PDENY** LOW. Once the power controller samples **PDENY** LOW, it can take any post-transition actions required. In the case of a denial, this will typically require reversing any pre-transition actions that were taken between T1 and T2. The denied transition sequence is complete and the interface state is again P_STABLE.

Note

In the denial sequence, the device retains all operational abilities associated with its current state. The operation of the device is therefore not interrupted by the transition request.

Device reset and initialization

At reset assertion, a device must set both **PACCEPT** and **PDENY** LOW. **PACTIVE** bits can be reset LOW or HIGH. If the device must enter a specific power state to process start-up operations, then the relevant **PACTIVE** bit can be reset HIGH. It is recommended that all **PACTIVE** bits are reset LOW.

Device reset assertion by a controller must occur only when the interface is in the P_STABLE state or when the controller and device are being reset simultaneously. This is consistent with the recommendation to isolate all signals LOW at power boundaries.

To facilitate device initialization to a desired state at reset deassertion, the controller provides a **PSTATE** value that the device must sample before taking appropriate initialization actions. The protocol requires that **PSTATE** is stable when the reset deassertion is detected at the device.

The device must specify an initialization period t_{init}, which is the number of device clock cycles required after reset deassertion before the **PSTATE** value is guaranteed to be captured for all possible reset states. *Device power state and transition support definition* on page 3-49 specifies the requirements for this initialization period.

A device must support all of the following valid controller behaviors during the initialization period.

- **PREQ** is LOW at reset deassertion. The controller waits until t_{init} has expired before requesting a transition to a new state. The controller must ensure an appropriate clock is running for the t_{init} period to allow **PSTATE** to be sampled. This method is always applied to unused interfaces, see *Unused interface* on page 3-49.
- Before reset deassertion, the controller asserts **PREQ** HIGH, then waits until the P-Channel transition has completed before requesting a transition to a new state. It is recommended to use this method in preference to the **PREQ** LOW behavior for actively managed interfaces.
- **PREQ** is LOW at reset deassertion. After reset deassertion but before t_{init} has elapsed, the controller maintains the value of **PSTATE** and asserts **PREQ** HIGH. It is IMPLEMENTATION DEFINED whether the device interprets this as a second transition. The controller waits until the P-Channel transition has completed before requesting a transition to a new state.

On sampling **PREQ** HIGH at reset exit or during t_{init}, a device must only complete the P-Channel transition when it is ready to receive another P-Channel request. The device must accept the first request made at reset exit or during t_{init}.

Figure 3-4 shows the case where the controller waits until t_{init} has expired before issuing a new transition, followed by assertion of reset.

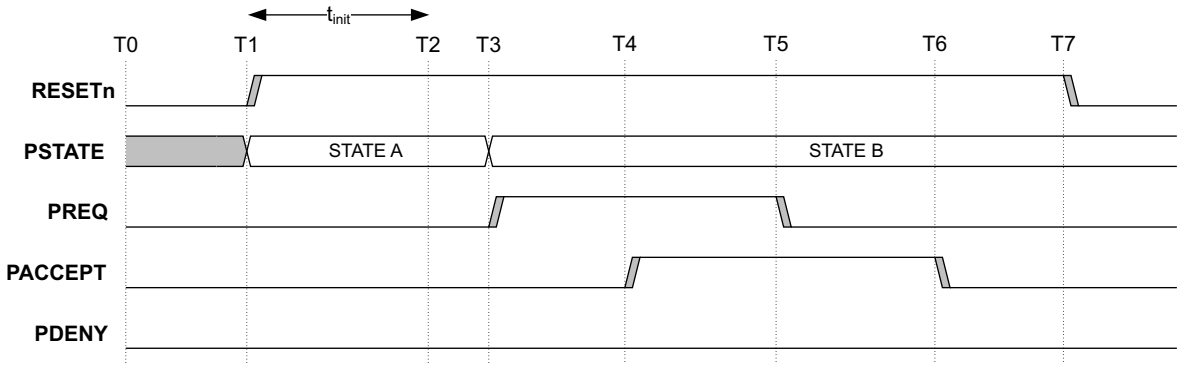


Figure 3-4 P-Channel initialization with controller-timed t_{init}, followed by a reset assertion

The transitions shown in Figure 3-4 are:

- At T0, the interface is idle and the device is in the reset state.
- Between T0 and T1, the power controller must take any pre-transition actions required before deasserting the reset, for example powering-on the domain.
- At T1, the power controller puts the current power state on **PSTATE** and releases the reset. The device must capture the **PSTATE** value within t_{init}. The controller must ensure that **PSTATE** is stable when the reset deassertion is detected by the device. A device sampling an initial **PSTATE** that is different from its default state, will cause the device to perform an internal transition to that state. It should delay responding to subsequent P-Channel requests until that internal transition is completed.
- At T2, t_{init} is complete and transitions to new states can be made.
- At T3, the controller requests that the device be placed in a state, STATE B, from which it can be safely returned to reset.

- Between T3 and T6, the device completes the transition.
- At T7, the reset is asserted.

Figure 3-5 shows the case where the controller sets **PREQ** HIGH before reset deassertion and then waits until the P-Channel transition is completed before issuing a further request. The device is guaranteed to be ready to receive another P-Channel request when this first transition completes, even if it completes within t_{init}.

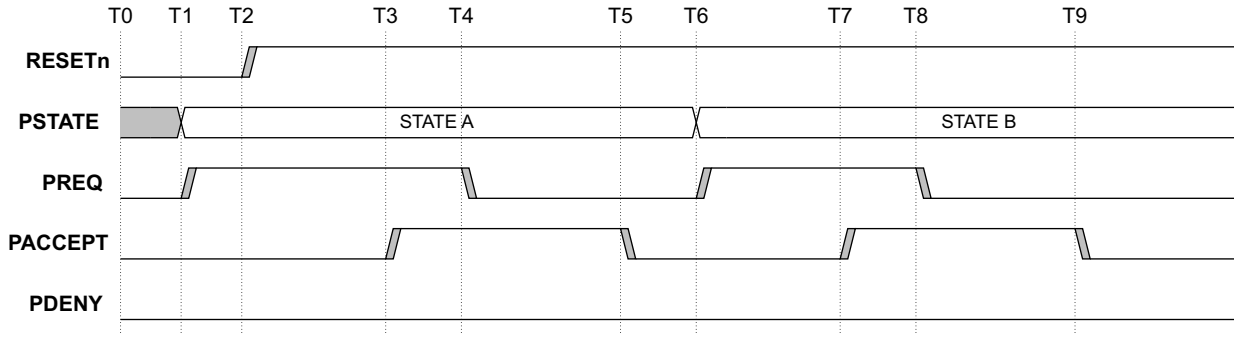


Figure 3-5 P-Channel initialization with PREQ HIGH at reset deassertion

Figure 3-6 shows the case where the controller sets **PREQ** HIGH after reset deassertion with the same **PSTATE** value as at reset deassertion. The controller waits until the P-Channel transition is complete before issuing a further request. The device is guaranteed to be ready to receive another P-Channel request when this first transition completes, even if it completes within t_{init}.

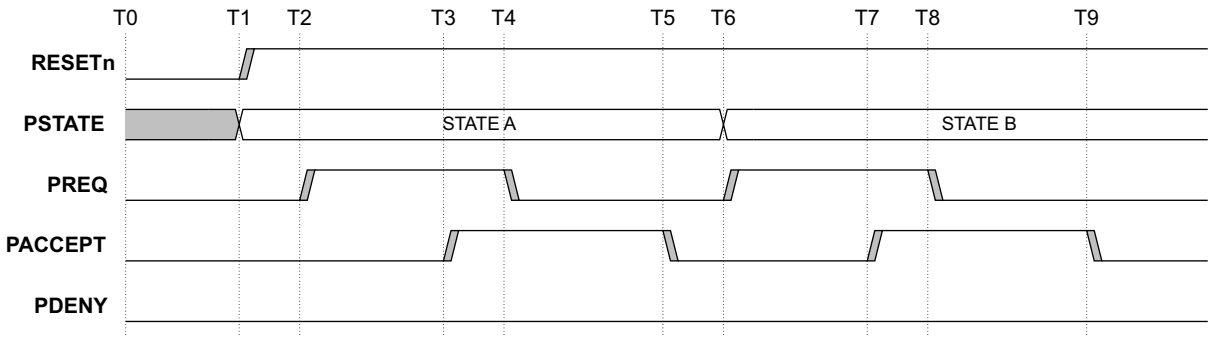


Figure 3-6 P-Channel initialization with PREQ HIGH during t_{init}

Multiple power state transitions

The P-Channel interface can be used to transition between multiple power states without having to pass through a common operational state.

Figure 3-7 shows how the P-Channel interface transitions from State A to State B and then on to State C without having to return to State A.

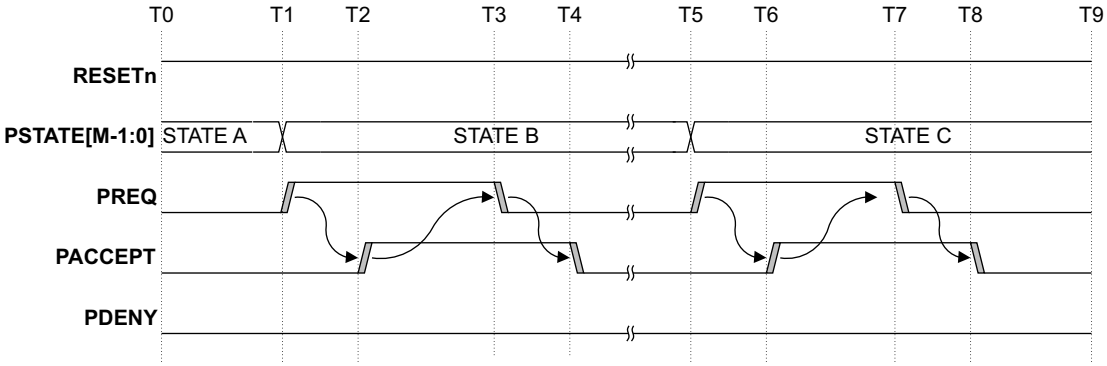


Figure 3-7 Multiple state transitions

The transitions shown in Figure 3-7 are:

- At T0, the interface is in the P_STABLE state.
- Between T1 and T4, the interface transitions from State A to State B, returning to the P_STABLE interface state at T4.
- Between T5 and T8, the interface transitions from State B to State C without having to first return to State A.

Interface states

Table 3-1 shows the interface states. If the device does not implement a denial mechanism, **PDENY** is tied LOW or absent, and the first five states constitute the complete set.

Table 3-1 P-Channel interface states

RESETn	PREQ	PACCEPT	PDENY	Interface state	Description
0	x	0	0	P_RESET	Device is in reset.
1	0	0	0	P_STABLE	Device state is stable.
1	1	0	0	P_REQUEST	Device is requested to transition to the state indicated by **PSTATE**.
1	1	1	0	P_ACCEPT	Device has accepted the request. Controller must set **PREQ** LOW.
1	0	1	0	P_COMPLETE	Controller has set **PREQ** LOW after P_ACCEPT. Device must set **PACCEPT** LOW.
1	1	0	1	P_DENIED	Device denies request and remains in the current state. Controller must set **PREQ** LOW and set **PSTATE** to the current state value.
1	0	0	1	P_CONTINUE	Controller has set **PREQ** LOW after P_DENIED. Device must set **PDENY** LOW.
x	x	1	1	Unused	Illegal signal combination.

Figure 3-8 is a state diagram showing the possible handshake sequences in terms of the signal states and interface states of each state change.

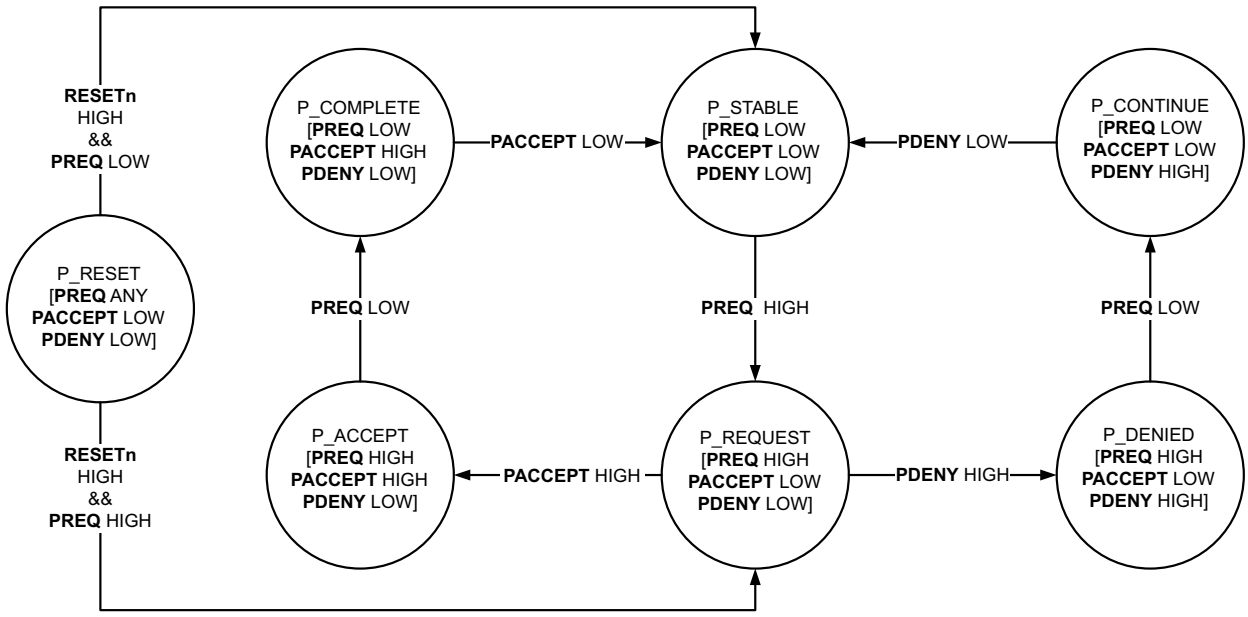


Figure 3-8 P-Channel state transitions

Handshake rules

The handshake signaling rules are:

- **PREQ** can only transition from LOW to HIGH when **PACCEPT** and **PDENY** are both LOW.
- **PREQ** can only transition from HIGH to LOW when either:
  - **PACCEPT** is HIGH and **PDENY** is LOW.
  - **PACCEPT** is LOW and **PDENY** is HIGH.
- **PSTATE** can only transition when either:
  - **PREQ**, **PACCEPT**, and **PDENY** are all LOW.
  - **PREQ** and **PDENY** are both HIGH and **PACCEPT** is LOW.
- **PACCEPT** can only transition from LOW to HIGH when **PREQ** is HIGH and **PDENY** is LOW.
- **PACCEPT** can only transition from HIGH to LOW when **PREQ** is LOW and **PDENY** is LOW.
- **PDENY** can only transition from LOW to HIGH when **PREQ** is HIGH and **PACCEPT** is LOW.
- **PDENY** can only transition from HIGH to LOW when **PREQ** is LOW and **PACCEPT** is LOW.

3.1.3 PACTIVE operation

PACTIVE output bits are used to indicate device requirements to the power controller with each bit representing a different requirement. A **PACTIVE** bit being HIGH indicates that the requirement should be provided to allow operations to progress. A **PACTIVE** bit being LOW is a hint that the requirement is no longer needed.

The P-Channel handshake is independent of **PACTIVE** and the controller can make any policy decision irrespective of any transitions on **PACTIVE**. However, the device can deny any request that is not appropriate.

The arrangement of **PACTIVE** bits is not restricted. However, a typical **PACTIVE** arrangement for each power state supported by the device is to have an associated **PACTIVE** output bit, ordered from the lowest state on the least significant bit (LSB) to the highest state on the most significant bit (MSB). The highest **PACTIVE** bit that is asserted then indicates the minimum power state required by the device to progress operations. In this example, the P-Channel controller transition requests behave as follows:

- If the power controller is at a lower state than that corresponding to the highest **PACTIVE** bit set HIGH, the controller requests a transition to that state or any higher supported state to allow operations to progress. Failure by the power controller to respond to a request for a higher state might lead to system deadlock.
- If the power controller is at a higher state than that corresponding to the highest **PACTIVE** bit set HIGH, the controller can request a transition to any state at or above the requested minimum.
- A device might not make use of all **PACTIVE** bits to explicitly request the corresponding state. In this case, the following rules apply:
  - Unused **PACTIVE** bits must be omitted or tied LOW.
  - Entry to a state corresponding to an unused **PACTIVE** bit when it is above the minimum requested by driven **PACTIVE** bits, is an IMPLEMENTATION DEFINED system decision.

The ordering of the states represented by the **PACTIVE** bits from least significant to most significant is IMPLEMENTATION DEFINED, but typically corresponds to increasing functionality and power consumption.

See also *Example PACTIVE usage* on page 3-63.

Controller policy and PACTIVE

Figure 3-9 shows an example of a device-led power transition based on changing **PACTIVE** values.

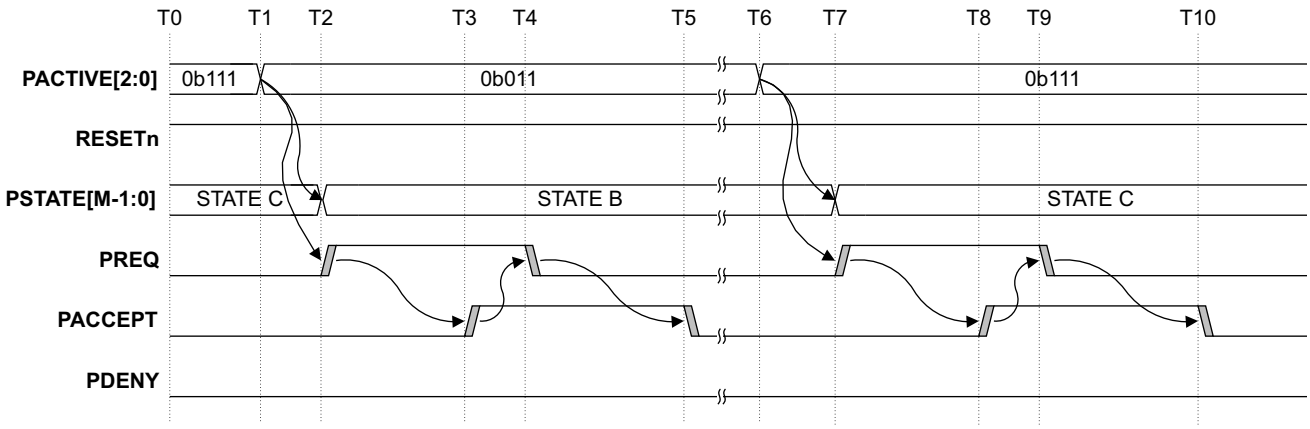


Figure 3-9 Device PACTIVE-led state transitions example

In this example, the **PACTIVE** bits are mapped to power states as follows:

- **PACTIVE[2]**: State C
- **PACTIVE[1]**: State B
- **PACTIVE[0]**: State A

The transitions shown in Figure 3-9 are:

- At T1, **PACTIVE[2]** goes LOW making State B the new minimum as **PACTIVE[1:0]** are still HIGH.
- Between T2 and T5, the interface transitions to State B.
- At T6, **PACTIVE[2]** goes HIGH indicating that State C is required.
- Between T7 and T10, the interface transitions to State C.

3.1.4 Device power state and transition support definition

To allow for correct controller design, device documentation must include a full enumeration of its power states and the allowed transitions between them specifying the following:

- All supported device power states including:
  - **PSTATE** definitions and encodings.
  - **PACTIVE** bit assignments to device requirements.
  - Recommended use of the **PACTIVE** outputs to initiate state transitions.
  - **PACTIVE** bits that are tied LOW or omitted.
- Supported transitions between device power states including:
  - Any actions taken by the device if the controller requests a transition to the current device state.
  - Which supported device power state transitions can be conditionally denied.
- **PSTATE** values supported at reset deassertion for device state initialization.
- The period t_{init}, in device clock cycles after reset deassertion, after which the device is guaranteed to have sampled **PSTATE** for all possible valid reset states. If a device has multiple clocks, it must also specify the clock to which this period relates.
  - The defined period is typically dependent on any delay in availability of the clock from reset exit. If the clock availability timing is system-dependent, this must be taken into account.

3.1.5 Signal subsets

This section describes the requirements for signal subsets that allow some interface signals to be omitted.

Unused interface

An unused interface must have the **PREQ** input signal tied LOW and the **PSTATE** inputs tied to a value corresponding to the state the device is required to enter when reset is deasserted.

Additionally, this **PSTATE** value must be tied to a functional state and the device must support entry into this state directly from reset to be operable.

Omission of PDENY

A device that has no requirement to deny a transition request can omit **PDENY**.

In this case, **PDENY** must be tied LOW at the controller.

3.2 Parity extended P-Channel interface

The parity extended P-Channel interface is an addition to the P-Channel interface that allows single bit faults on the interface to be detected and mitigated.

3.2.1 Parity extended P-Channel signals

In the parity extended P-Channel, each signal in the standard P-Channel interface has an associated check signal. Table 3-2 shows the check signal for each standard P-Channel signal.

Table 3-2 Parity extended P-Channel check signal relationships

Standard P-Channel signal	Associated check signal
PACTIVE[N-1:0]	**PACTIVECHK[N-1:0]**
PSTATE[M-1:0]	**PSTATECHK**
PREQ	**PREQCHK**
PACCEPT	**PACCEPTCHK**
PDENY	**PDENYCHK**

Each check signal provides odd parity checking for its associated signal. **PSTATECHK** is one bit irrespective of the width of **PSTATE**. There is one bit of **PACTIVECHK** per bit of **PACTIVE**. Figure 3-10 shows the signal direction for the parity extended P-Channel.

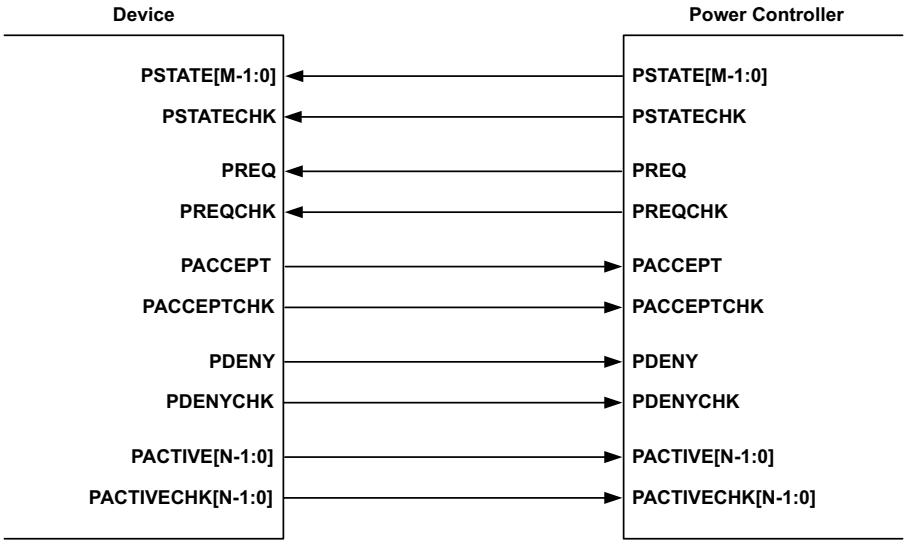


Figure 3-10 Parity extended P-Channel signaling

The protocol requires that **PSTATE** and **PSTATECHK** are both stable when a change on **PREQ** or **PREQCHK** is detected at the device.

There is no timing requirement between the arrival of a P-Channel signal and its associated check signal. However, the arrival times should be bounded by an implementation to provide a window for fault checking of the signals. The size of this window is IMPLEMENTATION DEFINED.

3.2.2 Parity extended P-Channel handshake

This section describes the parity extended P-Channel interface handshake.

The diagrams in this section show each P-Channel signal and the associated check signal changing simultaneously. However, this is not a requirement and shown only for the convenience of the schematics. The P-Channel state will not progress until both the P-Channel signal and the associated check signal are in their required state.

Accepted state transition

Figure 3-11 shows a handshake sequence for an accepted transition request.

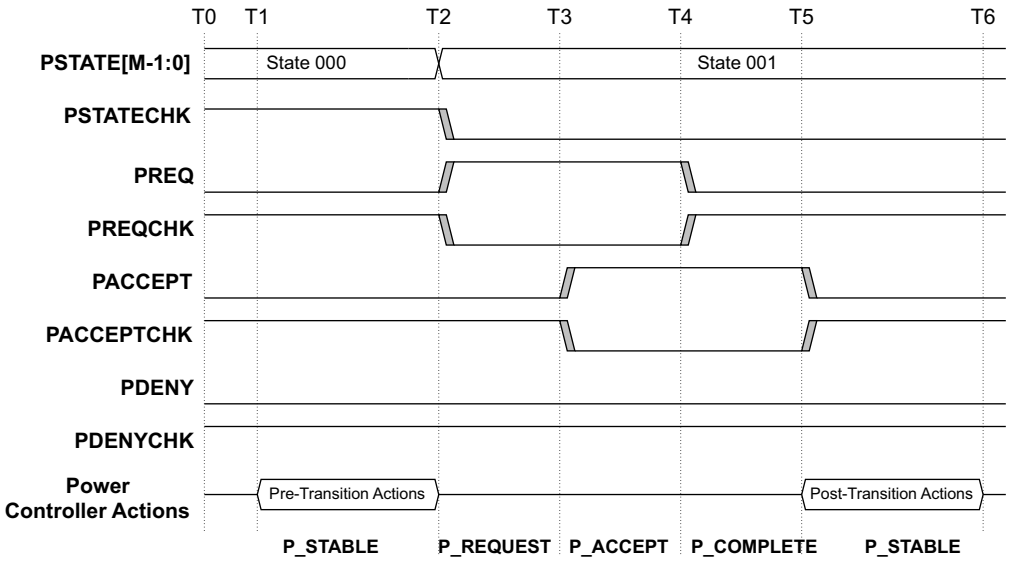


Figure 3-11 Parity extended P-Channel accepted transition request sequence

Figure 3-11 omits **PACTIVE** and **PACTIVECHK** because they are independent of the handshake. These signals can act as a stimulus for the controller to change the interface state. However, all interface state changes can be initiated by the controller alone. For more information, see *PACTIVE operation* on page 3-47.

The **PSTATE** encodings shown here are examples to illustrate the relationship between **PSTATE** and **PSTATECHK**.

The transitions shown in Figure 3-11 are:

T0 The P-Channel is in P_STABLE in the state `0b000`. The device remains in its current state.

T1 Recognizing that a transition request is required, the power controller must take any necessary pre-transition actions prior to requesting a new state. These actions are typically associated with a transition to a higher-power state. This might include powering on a domain or bringing a RAM out of the retention state. The interface state remains in P_STABLE.

T2 The power controller puts the required power state value on **PSTATE** and **PSTATECHK**, and sets **PREQ** HIGH and **PREQCHK** LOW. The interface progresses to P_REQUEST. The protocol requires that **PSTATE** and **PSTATECHK** are stable when the changes on **PREQ** and **PREQCHK** are detected at the device. This is typically achieved by resynchronizing **PREQ** and **PREQCHK** to the device clock domain and using the changes on both to generate a strobe to capture **PSTATE**.

T3 The device accepts the transition by setting **PACCEPT** HIGH and **PACCEPTCHK** LOW. **PDENY** must be kept LOW and **PDENYCHK** must be kept HIGH. The device can now exploit the capabilities of any higher state. The interface state is now P_ACCEPT.

T4 The power controller has sampled **PACCEPT** HIGH and **PACCEPTCHK** LOW, and sets **PREQ** LOW and **PREQCHK** HIGH. The interface state is P_COMPLETE.

T5 The device has sampled **PREQ** LOW and **PREQCHK** HIGH and sets **PACCEPT** LOW and **PACCEPTCHK** HIGH. Once the controller samples **PACCEPT** LOW and **PACCEPTCHK** HIGH, it can take any post-transition actions required. These actions are typically associated with transitioning to a lower-power state. This might include removing power from a power domain or placing a RAM into a retention state. The transition is now complete, and the interface state returns to P_STABLE. When moving to a lower-power state after setting **PACCEPT** LOW and **PACCEPTCHK** HIGH, the device cannot assume the availability of any properties of the previous higher-power state.

Device reset and initialization

At reset assertion a device must drive:

- **PACCEPT** LOW and **PACCEPTCHK** HIGH.
- **PDENY** LOW and **PDENYCHK** HIGH.

It is strongly recommended to reset all **PACTIVE** bits LOW and all **PACTIVECHK** bits HIGH. A controller must only assert a device reset when the interface is in the P_STABLE state or when the controller and device are being reset simultaneously. The remainder of the initialization procedure is the same as described in *Device reset and initialization* on page 3-43, with the addition of the relevant levels on the check signals.

Interface states

Table 3-3 summarizes the interface states and device availability.

If a device does not implement a denial mechanism, then the **PDENY** and **PDENYCHK** signals can be present or omitted. If present and denial is not implemented, then **PDENY** must be tied LOW and **PDENYCHK** must be tied HIGH.

If denial is not implemented, then rows in Table 3-3 with either **PDENY** HIGH or **PDENYCHK** LOW are illegal.

Table 3-3 Parity extended P-Channel interface states

	Interface state							
Interface state	**RESETn**	**PREQ**	**PREQCHK**	**PACCEPT**	**PACCEPTCHK**	**PDENY**	**PDENYCHK**	**Description**
P_RESET	0	X	X	0	1	0	1	Device is in reset.
P_STABLE	1	0	1	0	1	0	1	Device state is stable.
	1	0	0	0	1	0	1	Signal transition towards P_REQUEST. Excessive time in this interface state can indicate a fault on either of the **PREQ** or **PREQCHK** signals.
	1	1	1	0	1	0	1	Signal transition towards P_REQUEST. Excessive time in this interface state can indicate a fault on either of the **PREQ** or **PREQCHK** signals.

Table 3-3 Parity extended P-Channel interface states (continued)

Interface state	Interface state							Description
	RESETn	PREQ	PREQCHK	PACCEPT	PACCEPTCHK	PDENY	PDENYCHK	
P_REQUEST	1	1	0	0	1	0	1	Device is requested to transition to the state indicated by **PSTATE**.
	1	1	0	0	0	0	1	Signal transition towards P_ACCEPT. Excessive time in this interface state can indicate a fault on either of the **PACCEPT** or **PACCEPTCHK** signals.
	1	1	0	1	1	0	1	Signal transition towards P_ACCEPT. Excessive time in this interface state can indicate a fault on either of the **PACCEPT** or **PACCEPTCHK** signals.
	1	1	0	0	1	0	0	Signal transition towards P_DENIED. Excessive time in this interface state can indicate a fault on either of the **PDENY** or **PDENYCHK** signals.
	1	1	0	0	1	1	1	Signal transition towards P_DENIED. Excessive time in this interface state can indicate a fault on either of the **PDENY** or **PDENYCHK** signals.
P_ACCEPT	1	1	0	1	0	0	1	Device has accepted the request. Controller must set **PREQ** LOW and **PREQCHK** HIGH.
	1	0	0	1	0	0	1	Signal transition towards P_COMPLETE. Excessive time in this interface state can indicate a fault on either of the **PREQ** or **PREQCHK** signals.
	1	1	1	1	0	0	1	Signal transition towards P_COMPLETE. Excessive time in this interface state can indicate a fault on either of the **PREQ** or **PREQCHK** signals.
P_COMPLETE	1	0	1	1	0	0	1	Controller has set **PREQ** LOW and **PREQCHK** HIGH after P_ACCEPT. The device must set **PACCEPT** LOW and **PACCEPTCHK** HIGH.
	1	0	1	0	0	0	1	Signal transition towards P_STABLE. Excessive time in this interface state can indicate a fault on either of the **PACCEPT** or **PACCEPTCHK** signals.
	1	0	1	1	1	0	1	Signal transition towards P_STABLE. Excessive time in this interface state can indicate a fault on either of the **PACCEPT** or **PACCEPTCHK** signals.

Table 3-3 Parity extended P-Channel interface states (continued)

Interface state	Interface state							Description
	RESETn	PREQ	PREQCHK	PACCEPT	PACCEPTCHK	PDENY	PDENYCHK	
P_DENIED	1	1	0	0	1	1	0	Device denies request and remains in current state. Controller must set **PREQ** LOW and **PREQCHK** HIGH and set **PSTATE** to the current device state.
	1	0	0	0	1	1	0	Signal transition towards P_CONTINUE. Excessive time in this interface state can indicate a fault on either of the **PREQ** or **PREQCHK** signals.
	1	1	1	0	1	1	0	Signal transition towards P_CONTINUE. Excessive time in this interface state can indicate a fault on either of the **PREQ** or **PREQCHK** signals.
P_CONTINUE	1	0	1	0	1	1	0	Controller has set **PREQ** LOW and **PREQCHK** HIGH after P_DENIED. Device must set **PDENY** LOW and **PDENYCHK** HIGH.
	1	0	1	0	1	0	0	Signal transition towards P_STABLE. Excessive time in this interface state can indicate a fault on either of the **PDENY** or **PDENYCHK** signals.
	1	0	1	0	1	1	1	Signal transition towards P_STABLE. Excessive time in this interface state can indicate a fault on either of the **PDENY** or **PDENYCHK** signals.
Unused	Other States							Illegal states

Figure 3-12 on page 3-55 is a state diagram showing the possible handshake sequences in terms of the signal states and interface states.

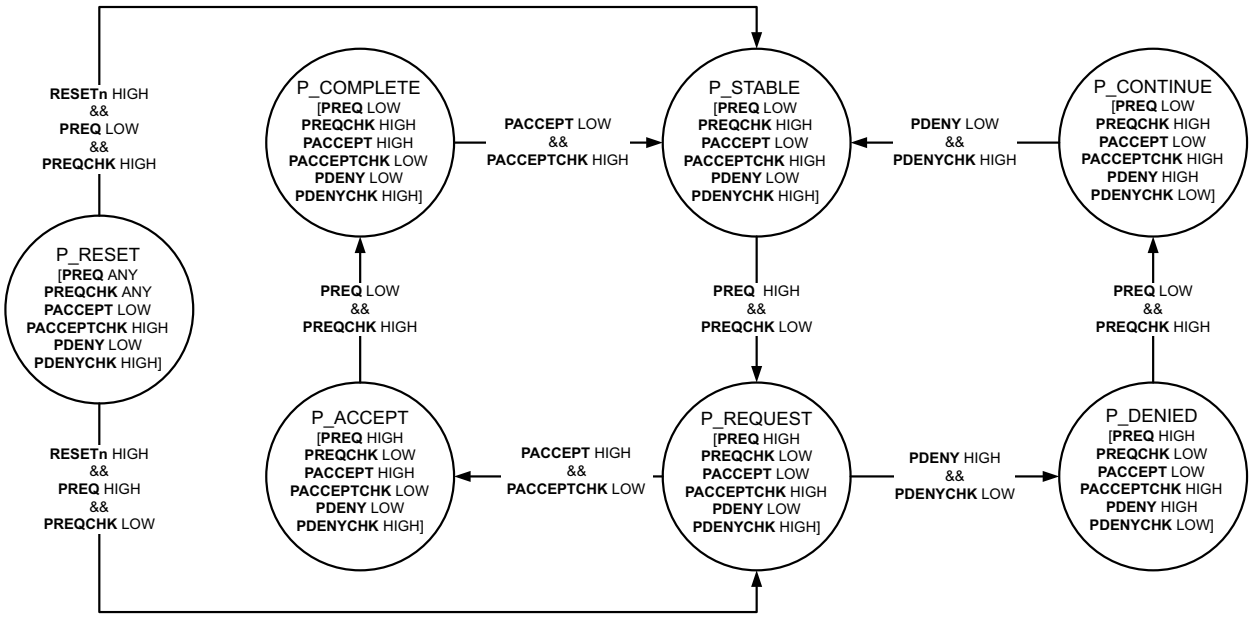


Figure 3-12 Parity extended P-Channel states

Handshake rules

The handshaking rules are:

- **PREQ** can only transition from LOW to HIGH and **PREQCHK** can transition from HIGH to LOW when all the following conditions are true:
  - **PACCEPT** is LOW and **PACCEPTCHK** is HIGH.
  - **PDENY** is LOW and **PDENYCHK** is HIGH.
- **PREQ** can only transition from HIGH to LOW and **PREQCHK** can transition from LOW to HIGH when either:

  All the following conditions are true:
  - **PACCEPT** is HIGH and **PACCEPTCHK** is LOW.
  - **PDENY** is LOW and **PDENYCHK** is HIGH.

  Or all the following conditions are true:
  - **PACCEPT** is LOW and **PACCEPTCHK** is HIGH.
  - **PDENY** is HIGH and **PDENYCHK** is LOW.
- **PSTATE** and **PSTATECHK** can only transition when either:

  All the following conditions are true:
  - **PREQ** is LOW and **PREQCHK** is HIGH.
  - **PACCEPT** is LOW and **PACCEPTCHK** is HIGH.
  - **PDENY** is LOW and **PDENYCHK** is HIGH.

  Or all the following conditions are true:
  - **PREQ** is HIGH and **PREQCHK** is LOW.
  - **PACCEPT** is LOW and **PACCEPTCHK** is HIGH.
  - **PDENY** is HIGH and **PDENYCHK** is LOW.

- **PACCEPT** can only transition from LOW to HIGH and **PACCEPTCHK** can transition from HIGH to LOW when all the following conditions are true:
  - **PREQ** is HIGH and **PREQCHK** is LOW.
  - **PDENY** is LOW and **PDENYCHK** is HIGH.
- **PACCEPT** can only transition from HIGH to LOW and **PACCEPTCHK** can transition from LOW to HIGH when all the following conditions are true:
  - **PREQ** is LOW and **PREQCHK** is HIGH.
  - **PDENY** is LOW and **PDENYCHK** is HIGH.
- **PDENY** can only transition from LOW to HIGH and **PDENYCHK** can transition from HIGH to LOW when all the following conditions are true:
  - **PREQ** is HIGH and **PREQCHK** is LOW.
  - **PACCEPT** is LOW and **PACCEPTCHK** is LOW.
- **PDENY** can only transition from HIGH to LOW and **PDENYCHK** can transition from LOW to HIGH when all the following conditions are true:
  - **PREQ** is LOW and **PREQCHK** is HIGH.
  - **PACCEPT** is LOW and **PACCEPTCHK** is HIGH.

PACTIVE operation

Any unused **PACTIVE** bit and its associated **PACTIVECHK** bit should either:

- Have both signals omitted.
- Have the **PACTIVE** bit tied LOW and the **PACTIVECHK** bit tied HIGH

For more information, see *PACTIVE operation* on page 3-47.

Signal subsets

The supported signal subsets are aligned with those of a standard P-Channel. Where a signal is not present, then its associated check signal is also removed. An unused interface must have the **PREQ** input signal tied LOW and the **PREQCHK** input signal tied HIGH.

The **PSTATE** and **PSTATECHK** input values must be tied to a functional state and the device must support entry into this state directly from reset to be operable.

Omission of PDENY

A device that has no requirement to deny a transition request can omit **PDENY** and **PDENYCHK**. In this case, **PDENY** must be tied LOW and **PDENYCHK** must be tied HIGH at the controller.

3.3 P-Channel implementation

Typically a device and controller are implemented with asynchronous clocks. Figure 3-13 shows the recommended implementation of resynchronization in this case. This figure does not show the mechanism for capturing **PSTATE** at reset deassertion.

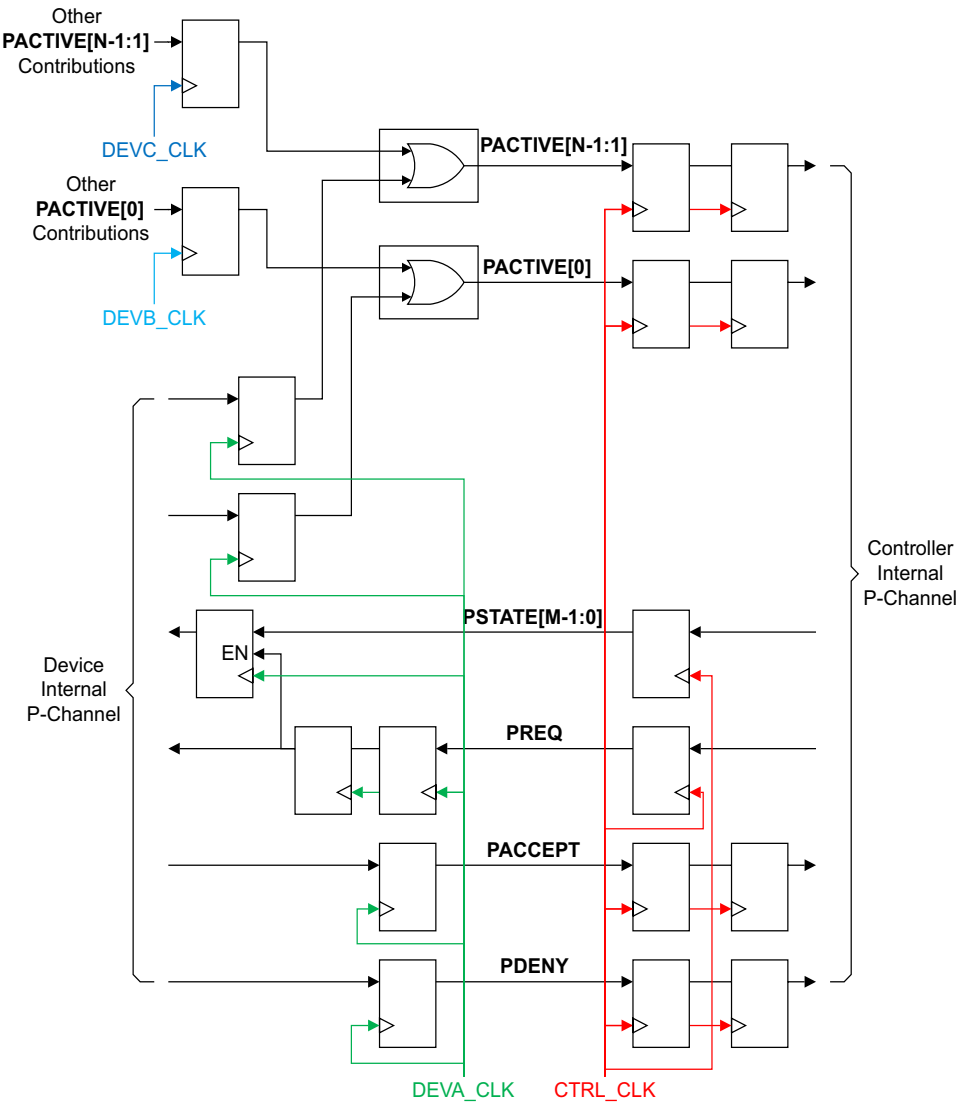


Figure 3-13 Recommended P-Channel clock domain crossing synchronization

For an asynchronous P-Channel implementation, it is recommended to provide the following guidance to implementers:

- Resynchronize all signals, with the exception of **PSTATE** at their destination before use.
- Register the **PSTATE** output to ensure it does not glitch.
- To ensure operation of the four-phase handshake, register all **PREQ**, **PACCEPT**, and **PDENY** outputs.
- Each bit of **PACTIVE** is driven either directly by a register or by a number of registers whose contributions are logically combined. It is recommended that this combining logic is limited to a logical OR where possible, and is implemented using instantiated gates.

- When other logic is implemented, the implications of **PACTIVE** source register changes on a **PACTIVE** output bit must be carefully considered. Although the handshake protocol guarantees functionally correct behavior regardless of **PACTIVE** activity, it is recommended to implement the simplest possible logic to minimize the likelihood of introduced glitches at the **PACTIVE** output.

Note

Although Figure 3-13 on page 3-57 shows a typical two-stage synchronization, resynchronization must be implemented appropriately for the technology libraries used and required frequency targets.

3.3.1 Capturing PSTATE

The **PSTATE** value is captured in several different scenarios:

- **PSTATE** must be captured at reset deassertion. Therefore, a device must have a mechanism for capturing **PSTATE** at this time.
- **PSTATE** must be captured when the device samples **PREQ** HIGH. This can be achieved by resynchronizing **PREQ** to the device clock domain and using it to capture **PSTATE**, as shown in Figure 3-13 on page 3-57.
- **PSTATE** can be captured when a request is denied. The device can either retain knowledge of its current state during a P-Channel request or sample the **PSTATE** value again after a denial when **PREQ** goes LOW. Therefore, the **PSTATE** value can be captured when the device samples **PREQ** LOW with **PDENY** HIGH.

Implementers can place a maximum delay timing constraint on the **PSTATE** signal to ensure it reaches its capture register before **PREQ** is synchronized.

If additional time margin is required on **PSTATE**, add clock cycles at the controller between setting the required **PSTATE** value and setting **PREQ** HIGH.

Figure 3-14 shows an example where **PSTATE** is set before **PREQ** is driven HIGH.

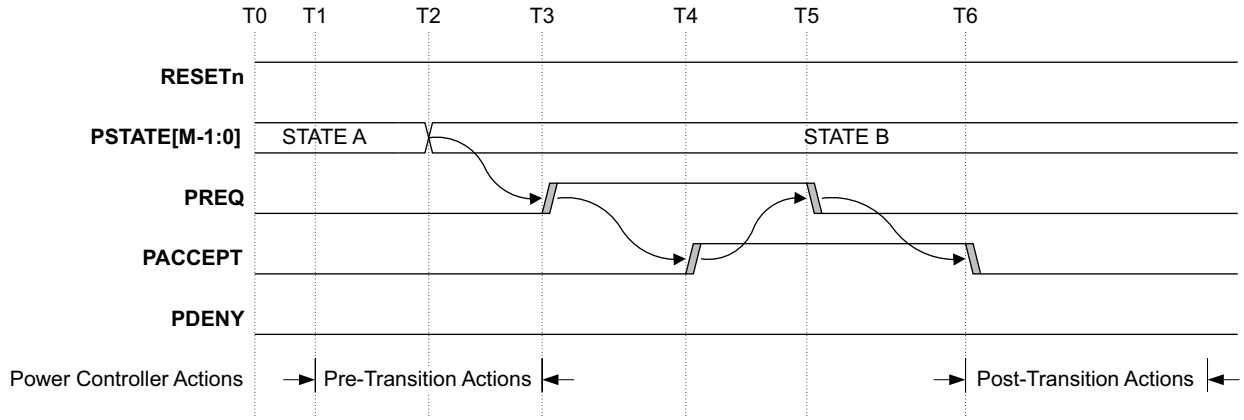


Figure 3-14 Additional timing for PSTATE by asserting PSTATE before PREQ

At T2, **PSTATE** is set to the value required for the next state. There is then a delay until T3 when **PREQ** is asserted.

Note

The power controller can carry out pre-transition actions until the point that **PREQ** is set HIGH as the device must not take any action before it samples **PREQ** HIGH.

3.3.2 Supported PSTATE initialization values

It is strongly recommended that the device supports at least one **PSTATE** initialization value that facilitates exit from reset directly into a functional state.

3.4 Connections between parity protected and non-parity protected P-Channel interfaces

In certain circumstances, it might be required to connect controllers and devices where one has the parity protection and the other does not. The following sections describe the integration requirements for these types of connections. When connecting parity protected and non-parity protected controllers and devices, the interface has no or limited fault protection.

3.4.1 Connecting a non-parity protected P-Channel controller to a parity protected P-Channel device

Figure 3-15 shows the integration required between a non-parity protected P-Channel controller to a parity protected P-Channel device.

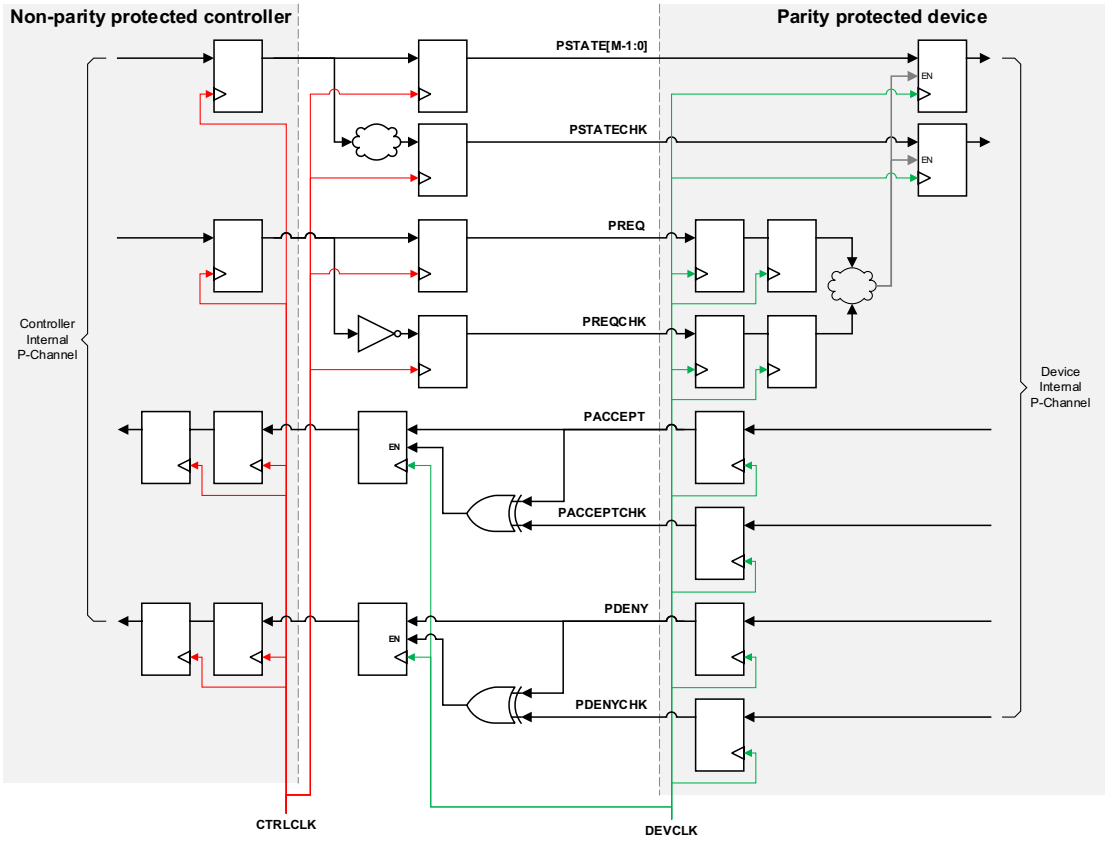


Figure 3-15 Non-parity protected P-Channel controller to a parity protected P-Channel device connections

Additional consideration must be taken if the parity protection check signals are generated on a different clock to the standard P-Channel signals. If so, these signals must be resynchronized to the device clock before being used in the **PACCEPT** and **PDENY** integration logic shown in Figure 3-15.

PACTIVE

Each **PACTIVE** bit must be conditioned separately as each is an independent indication of a power mode requirement. Each device **PACTIVE** output bit can be directly connected to the corresponding controller **PACTIVE** input bit and the corresponding device **PACTIVECHK** output bit is left unconnected.

This might have the effect of indicating a level, either HIGH or LOW, at the controller before both signals indicate this requirement on the device side. In this case, the P-Channel handshake ensures the device maintains a power mode suitable to the device's request. However, such behavior might cause unnecessary P-Channel requests.

There are alternative integration options that might be used to prevent the propagation of a **PACTIVE** bit until both the standard and check signals have changed. However, the propagation of each **PACTIVE** bit is typically required to be maintained as a combinatorial path to ensure, for example, a request can be made when a local clock is not available.

Adding a non-trivial amount of logic to the **PACTIVE** bit path might result in glitches. While these glitches will not cause malfunctions due to the handshake guarantee as noted above, they might cause unnecessary transitions.

PSTATE

The device **PSTATECHK** input must be generated from the **PSTATE** value. The **PSTATECHK** value is calculated as an odd parity of the **PSTATE** value. This signal must be registered with the controller clock after it is calculated to ensure that it is suitable to capture in the device clock domain.

PREQ

The **PREQCHK** signal should be generated from the non-parity protected **PREQ** controller output. This can be done with a simple inversion. The **PREQ** and **PREQCHK** should additionally be registered after generation to ensure their propagation delay is consistent with the **PREQCHK** that has to be registered after its generation.

PACCEPT and PDENY

The **PACCEPT** input to the non-parity protected controller is required to be conditioned such that it only updates when both **PACCEPT** and **PACCEPTCHK** have updated to the new value.

If this conditioning is not present and **PACCEPT** is connected directly, then a change on **PACCEPT** when **PACCEPTCHK** has not changed, might result in a change on **PREQ** or **PREQCHK** that might cause a protocol violation at the parity protected device.

This conditioning can be provided by propagating the **PACCEPT** value only when **PACCEPT** and **PACCEPTCHK** are different. A register is required to hold the last value as when in transition the **PACCEPT** and **PACCEPTCHK** signals can be either both HIGH or both LOW regardless of the value to which they are transitioning.

The same provision must also be made for **PDENY** and **PDENYCHK** signal from the parity protected device to the non-parity protected controller.

3.4.2 Connecting a parity protected P-Channel controller to a non-parity protected P-Channel device

Figure 3-16 on page 3-61 shows the integration required between a parity protected P-Channel controller to a non-parity protected P-Channel device.

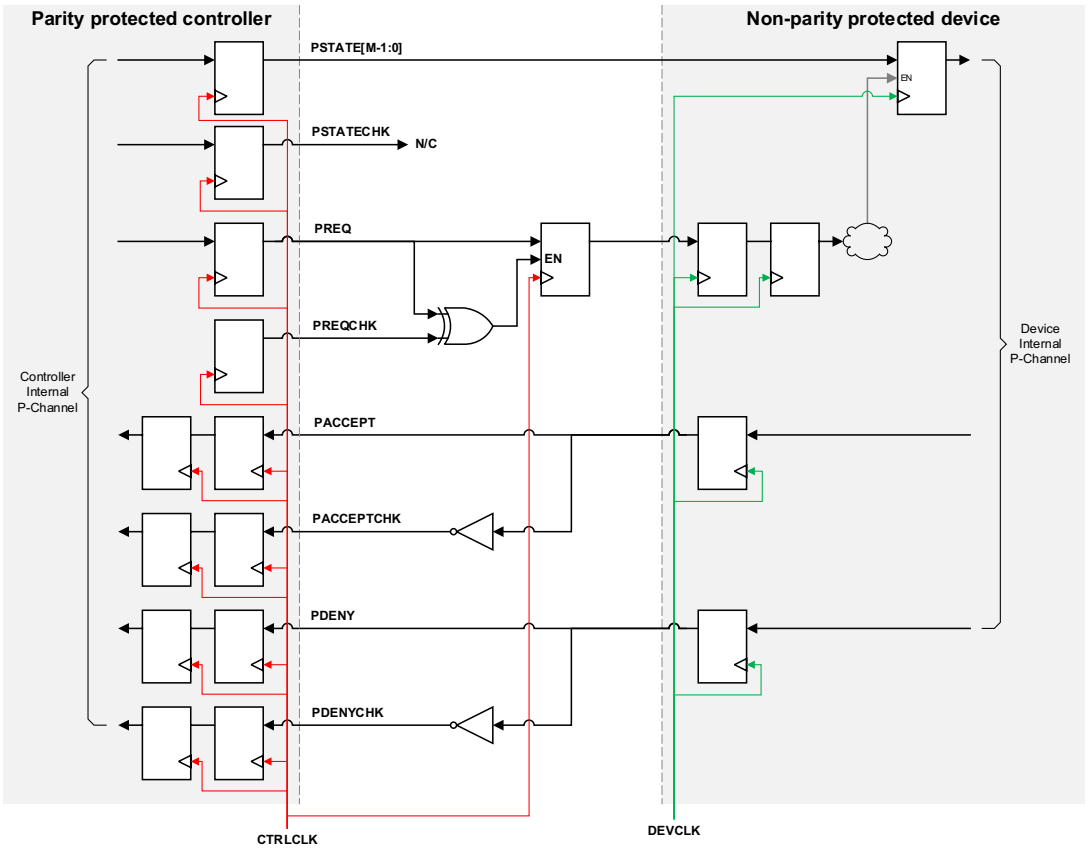


Figure 3-16 Parity protected P-Channel controller to a non-parity protected P-Channel device connections

Additional consideration must be taken if the parity protection check signals are generated on a different clock to the standard P-Channel signals. If so, these signals must be resynchronized using the controller clock before being used in the **PREQ** integration logic shown in Figure 3-16.

PSTATE

The **PSTATE** value should be connected directly. The conditioning on the **PREQ** ensures that the handshaking progresses in the correct order.

PREQ

The **PREQ** input to the non-parity protected device is required to be conditioned such that it only updates when both the **PREQ** and **PREQCHK** have updated to the new value.

If this conditioning is not present and **PREQ** was connected directly, then a change on **PREQ** when **PREQCHK** has not changed, might result in a change on **PACCEPT** or **PDENY**, that might cause a protocol violation at the parity protected controller.

This conditioning can be provided by only propagating the **PREQ** value when **PREQ** and **PREQCHK** are different. A register is required to hold the last value as when in transition the **PREQ** and **PREQCHK** signals can be either both HIGH or both LOW regardless of the value to which they are transitioning.

PACTIVE, PACCEPT, and PDENY

The **PACTIVECHK**, **PACCEPTCHK**, and **PDENYCHK** inputs for the parity protected controller should be generated from an inversion of the **PACTIVE**, **PACCEPT**, and **PDENY** signals respectively.

3.5 P-Channel application examples

This section illustrates a number of applications of the P-Channel interface.

3.5.1 Power domain management

The P-Channel device control logic can be supported:

- Within the managed power domain.
- Within a parent power domain that supports a nested child power domain.

Figure 3-17 shows a simple example with three power domains across two devices.

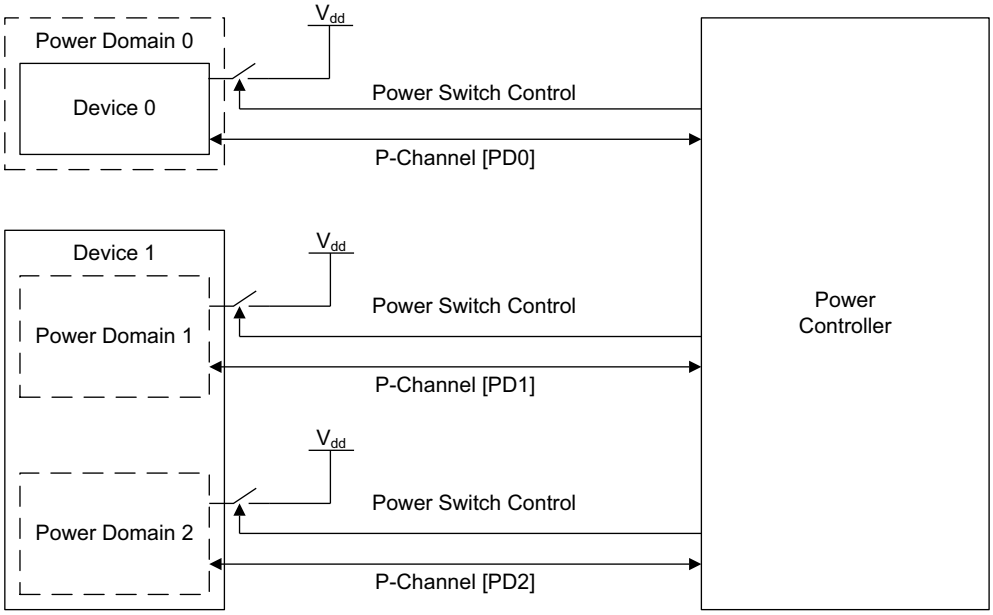


Figure 3-17 Simple power domain example

In Figure 3-17, Device 0 is wholly in one power domain, while Device 1 contains two power domains. There is one P-Channel interface for each power domain and in all cases the P-Channel control logic is contained in that power domain.

In this arrangement, each power domain can only make transitions between states with the corresponding P-Channel responses, if logic functionality in that domain is available at that time. Such a device cannot create wakeup signaling using **PACTIVE** from states without operable logic. In this state, the system is responsible for wakeup signaling.

Since some logic functionality is required at minimum to respond to the P-Channel requests, this arrangement cannot support RAM-only power domains.

Figure 3-18 shows an example of a device with multiple power domains where there is a parent-child relationship between a primary domain and two secondary domains.

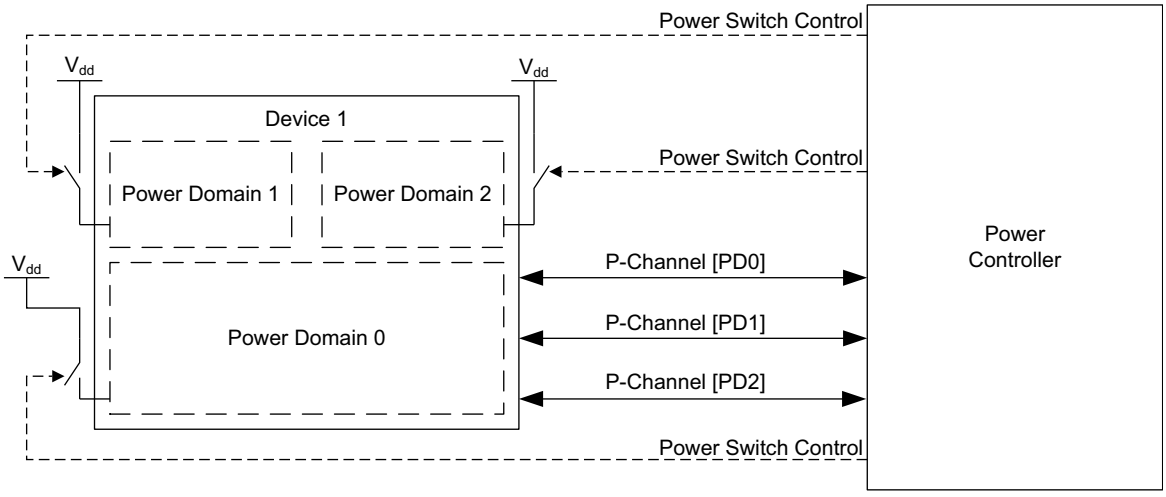


Figure 3-18 Parent-child power domain example

In Figure 3-18, Power Domain 0 is the parent domain, with a first-on, last-off relationship to the child power domains, Power Domain 1, and Power Domain 2. All P-Channel interface logic and control logic are located within Power Domain 0.

In this case, the device capabilities can include parent domain detection and signaling of wakeup conditions using **PACTIVE** on behalf of secondary power domains, to cause a transition to a higher state. This can, for example, enable implementation of opportunist power state control for secondary power domains that might otherwise be complex or impossible to implement at system level.

The secondary power domains can also transition between states without any operable logic on the interface in either state. This example can support both increased state management flexibility between various levels of non-operational power state, and the use of domains that do not contain logic capabilities, such as a RAM-only power domain.

3.5.2 Example PACTIVE usage

Figure 3-19 shows an example of device power states and supported transitions.

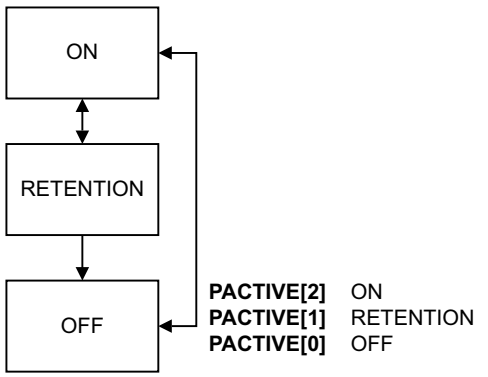


Figure 3-19 Example device power states and PACTIVE enumeration

Table 3-4 shows the device power states and related P-Channel behavior.

Table 3-4 Device power state enumeration

State	PACTIVE bit	PACTIVE status	Supported transitions, to	Transition denial support
ON	2	Driven	RETENTION or OFF	Yes
RETENTION	1	Driven	ON or OFF	Only to OFF
OFF	0	Omitted	ON	No

PACTIVE[0] represents the OFF state. It can be omitted by the device as this is an implicit minimum state requirement when all other bits are LOW. It should be tied LOW at the controller if supported as an input.

If **PACTIVE[2]** is HIGH, then the device is requesting the ON state.

If, while the device in the ON state, **PACTIVE[2]** transitions LOW but **PACTIVE[1]** is HIGH, then the device is requesting the RETENTION state as a minimum state. The controller can choose to remain in the ON state or request a transition to the RETENTION state, depending on its configuration.

If no **PACTIVE** bits are HIGH, then the device can be in any state, but this is an indication that the device might accept a request to enter the OFF state.

If only **PACTIVE[2]** is driven by the device, then when it is LOW the controller might request any of the ON, RETENTION, or OFF states. This could be defined, for example, by software configuration of the power controller policy.

Appendix A
Revisions

This appendix describes changes between released issues of this document.

Table A-1 Issue B

Change	Location
First release	-

Table A-2 Differences between issue C and issue B

Change	Location
Clarified the composition of **QACTIVE** and **PACTIVE** signals.	• *Device activity indication* on page 2-18. • *Device activity indication* on page 3-40. • *Q-Channel implementation* on page 2-31. • *P-Channel implementation* on page 3-57.
Clarified points at which **PSTATE** must be stable.	• *Denied state transition* on page 3-43. • *Device reset and initialization* on page 3-43.
Changed reference **PACCEPT** to **PREQ** in the table entry for interface state P_ACCEPT. Elaborated on the description in the table entry for interface state P_DENIED.	Table 3-1 on page 3-46.

Table A-2 Differences between issue C and issue B (continued)

Change	Location
Changed **QDENYn** to **QDENY** in the figure showing recommended Q-Channel clock domain crossing synchronization. Corrected signal names in the guidance following the figure.	*Recommended Q-Channel clock domain crossing synchronization* on page 2-31.
Added Note about what the figure does not include.	*Recommended P-Channel clock domain crossing synchronization* on page 3-57.
Added information about capturing PSTATE.	*Capturing PSTATE* on page 3-58.

Table A-3 Differences between issue D and issue C

Change	Location
Added *Preface.*	*Preface*.
Content reorganization.	• Chapter 2 *Q-Channel Interface Specification*. • Chapter 3 *P-Channel Interface Specification*.
Added information on Interface parity protection and parity protection implementation guidelines.	• *Parity extended Q-Channel interface* on page 2-25. • *Parity extended P-Channel interface* on page 3-50.