AMBA® Low Power Interface Specification

ARM® Q-Channel and P-Channel Interfaces

ARM IHI 0068C (ID091216)

AMBA Low Power Interface Specification
ARM Q-Channel and P-Channel Interfaces

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

Proprietary Notice

ARM LOW POWER INTERFACE 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 THE RELEVANT LOW POWER INTERFACE SPECIFICATION ACCOMPANYING THIS LICENCE. ARM IS ONLY WILLING TO LICENSE THE RELEVANT LOW POWER INTERFACE 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 LOW POWER INTERFACE SPECIFICATION YOU INDICATE THAT YOU AGREE TO BE BOUND BY ALL THE TERMS OF THIS LICENCE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENCE, ARM IS UNWILLING TO LICENSE THE RELEVANT LOW POWER INTERFACE SPECIFICATION TO YOU AND YOU MAY NOT USE OR COPY THE RELEVANT LOW POWER INTERFACE SPECIFICATION AND YOU SHOULD PROMPTLY RETURN THE RELEVANT LOW POWER INTERFACE SPECIFICATION TO ARM.

“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 Low Power Interface Specification for the purpose of developing and having developed products that comply with the relevant Low Power Interface 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 ARM Low Power Interface 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 is created under Clause 1(i) or manufactured under Clause 1(ii) it must contain at least one processor core which has either been (a) developed by or for ARM; or (b) developed under licence from ARM;

   (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 Low Power Interface 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 Low Power Interface Specification.

4. THE RELEVANT LOW POWER INTERFACE 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 PERMITTED 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 RELEVANT LOW POWER INTERFACE SPECIFICATION; AND (II) THE IMPLEMENTATION OF THE RELEVANT LOW POWER INTERFACE SPECIFICATION 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 in connection with the relevant Low Power Interface 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 Low Power Interface 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 Low Power Interface Specification and destroy all copies of the relevant Low Power Interface 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.

ARM contract references: LES-PRE-20414 ARM LOW POWER INTERFACE SPECIFICATION LICENCE

In this document, where the term ARM is used to refer to the company it means "ARM or any of its subsidiaries as appropriate".

Web Address

`http://www.arm.com`

Contents
AMBA Low Power Interface Specification ARM Q-Channel and P-Channel Interfaces

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.

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.

There are no restrictions on the specific characteristics of the states that the interfaces manage.

1.2 Interface differences and selection

ARM intends the two interfaces to complement each other, and that the interfaces are used as circumstances dictate.

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. This must be entered before the device can enter a different quiescent state.

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 common configuration of the controller and device is not needed. Typically, each P-Channel maps directly from a controller to the management of a power domain partition of a single device. The P-Channel interface is used only if Q-Channel functionality is insufficient.

Chapter 2
Interface Specification

This chapter describes the Q-Channel and P-Channel low-power interfaces. 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. 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:

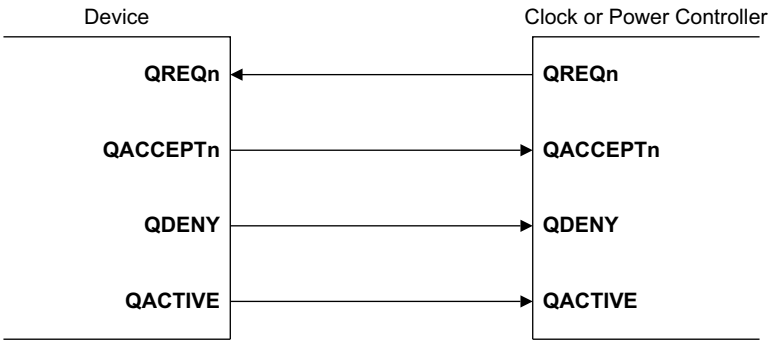

Figure 2-1 Q-Channel device and controller signaling

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 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 can include device input signals. The final **QACTIVE** signal is driven either directly by a register or by a number of registers whose outputs are logically combined. *Q-Channel implementation* on page 3-30 gives implementation recommendations for **QACTIVE** combining logic.

Note

If the device is itself unable to assert **QACTIVE** HIGH, in the absence of the resource managed according to the interface, 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 acknowledgement signal pair, **QACCEPTn** and **QDENY**, which are driven back to the controller by the device to indicate acceptance or denial of a request. The acknowledgement 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 denial mechanism means a device can maintain an operational state while having a mechanism by which it can promptly complete the handshake of a quiescence request.

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:

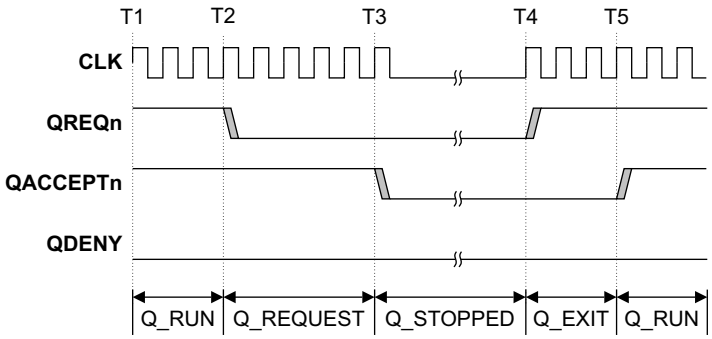

Figure 2-2 Q-Channel accepted quiescence request sequence

Figure 2-2 omits **QACTIVE** because, although **QACTIVE** can act as a stimulus for the controller to change handshake state, it is independent of the handshake. All handshake state changes can be initiated by the controller alone. *Controller policy and QACTIVE* on page 2-17 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.
- 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.
- 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. This is the only state where 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 acknowledgement 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.
- 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:

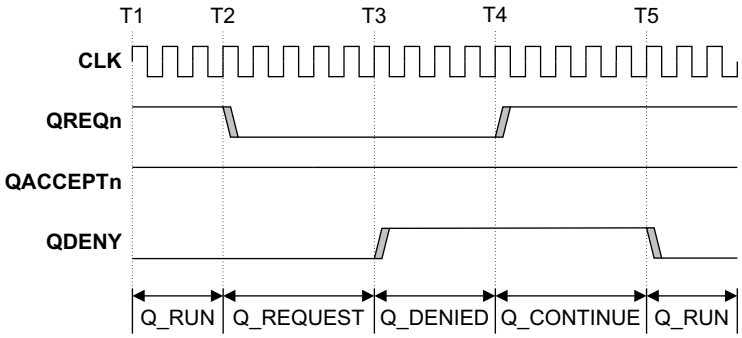

Figure 2-3 Q-Channel denied quiescence request sequence

- The sequence from T1 to T2 is identical to that described in *Accepted quiescence request* on page 2-13.
- At T3 the device drives **QDENY** HIGH whilst **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.
- 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 ARM recommends **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. 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 re-entering the quiescent Q_STOPPED state, after which reset is asserted.

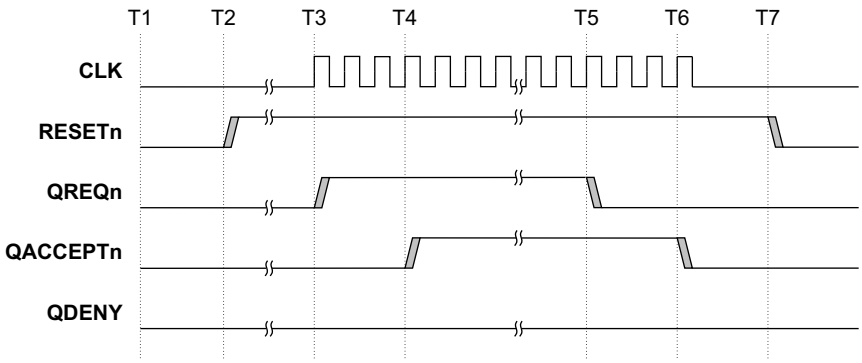

Figure 2-4 QREQn LOW at reset deassertion

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 re-entering the quiescent Q_STOPPED state, after which reset is asserted.

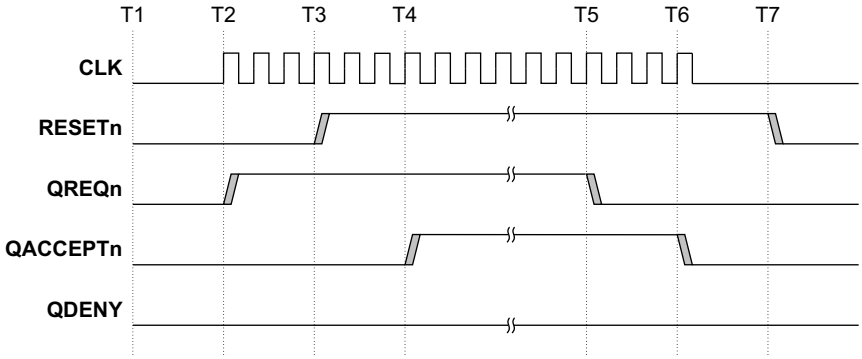

Figure 2-5 QREQn HIGH at reset deassertion

Handshake 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 handshake 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 when idle.
0	0	0	Q_STOPPED	Device has entered the quiescent state. This is the only state where the controller does not guarantee the availability of any resource managed by the interface.
1	0	0	Q_EXIT	Supply of clock or power is guaranteed. When the device acknowledges, by deasserting **QACCEPTn** HIGH, the state moves to Q_RUN.
0	1	1	Q_DENIED	Device denies 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.

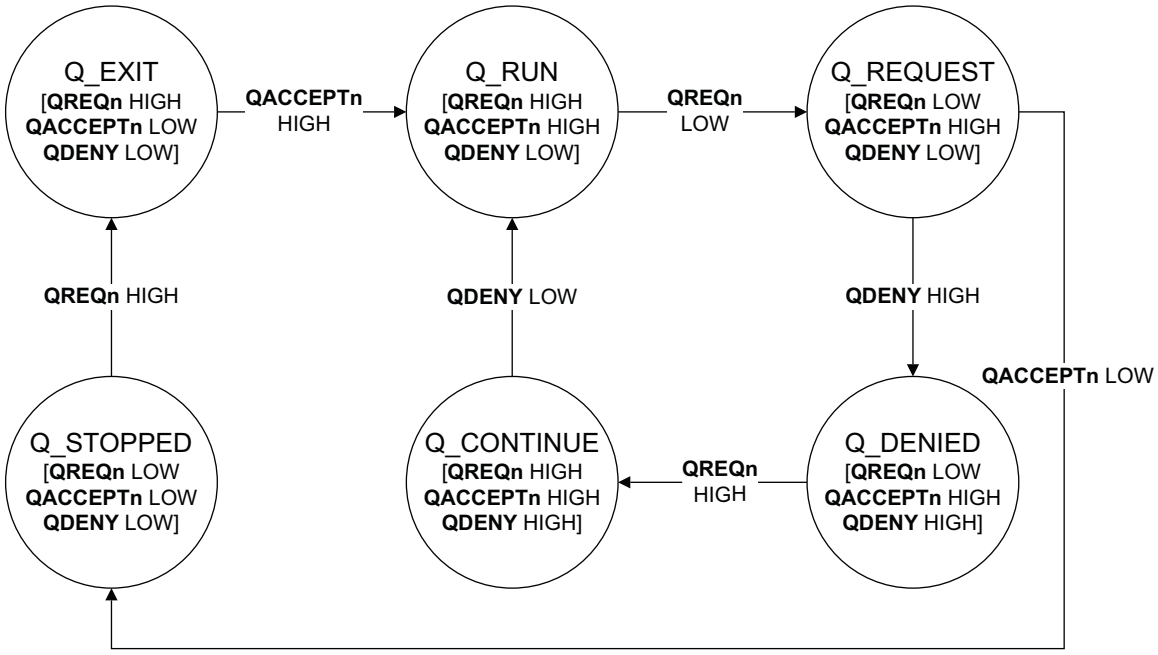

Figure 2-6 Q-Channel states

Handshake rules

The handshake signalling 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 examples

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.

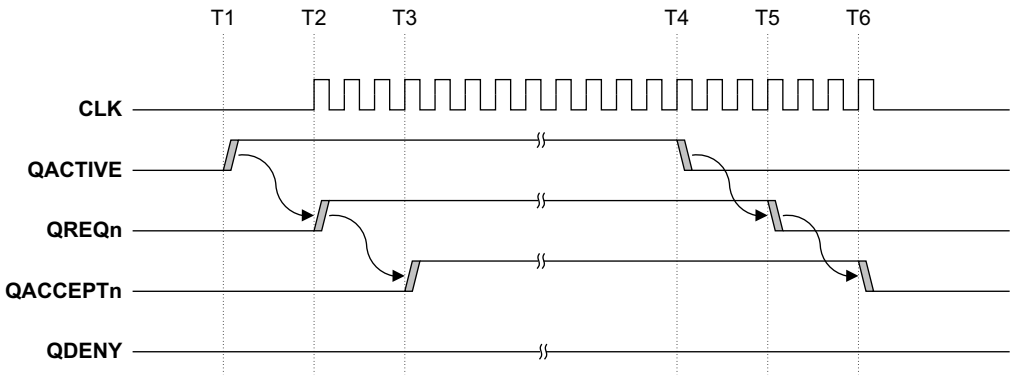

Figure 2-7 Device-led Q_EXIT and Q_REQUEST

Figure 2-8 shows a controller policy where the exit from quiescence is led by the device but the entry is initiated by the controller. This might be a request for the device to complete its current actions, and not accept more, before becoming quiescent.

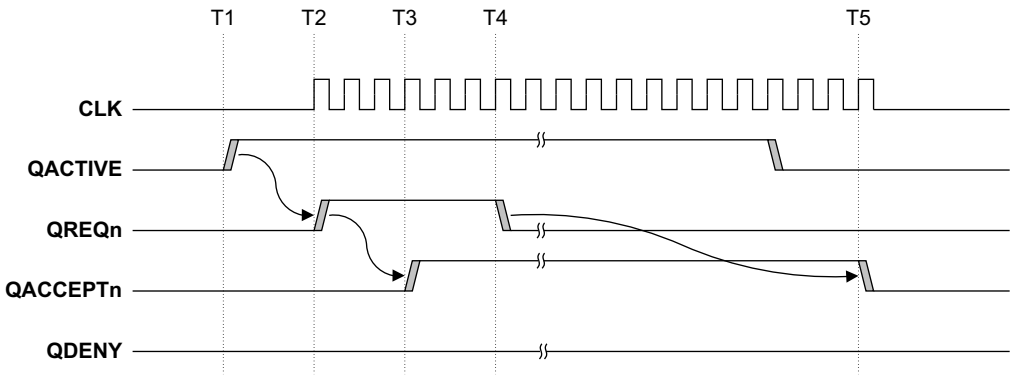

Figure 2-8 Device-led Q_EXIT and controller-led Q_REQUEST

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** with an implicit tie LOW. This subset also offers backward compatibility with the AXI low-power interface specification for devices that have no requirement to deny a quiescence request. Chapter 5 *Q-Channel Backward Compatibility* on page 5-41 describes this backwards compatibility.

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 that is used to guarantee clock or power availability.

2.2 P-Channel interface specification

This section describes the P-Channel interface, that controls power state transitions.

Note

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.

2.2.1 P-Channel Signals

Figure 2-9 shows the P-Channel signals between a device and a power controller.

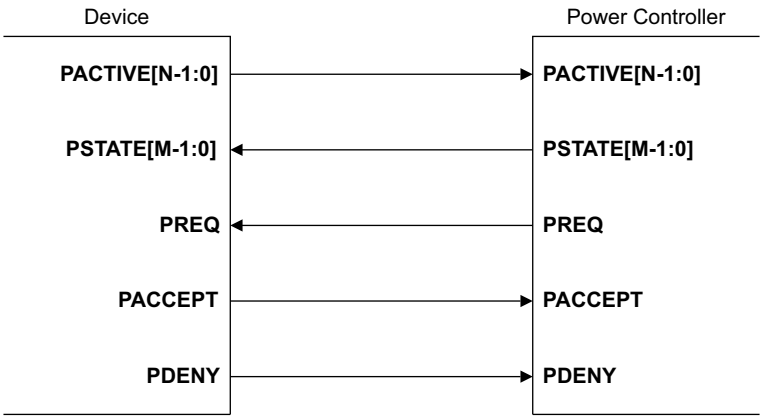

Figure 2-9 P-Channel device and controller signal mappings

A P-Channel interface comprises the following signal groups.

Device activity indication

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

- The **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 a number of registers whose outputs are logically combined. *P-Channel implementation* on page 3-31 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 3-31.

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 acknowledgement 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.

The purpose of the denial mechanism is to enable a device to maintain its current state whilst 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 2-22 describes reset and initialization transitions.

2.2.2 P-Channel handshake

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

Accepted state transition

Figure 2-10 shows a transition request from State A to State B that is accepted by the device.

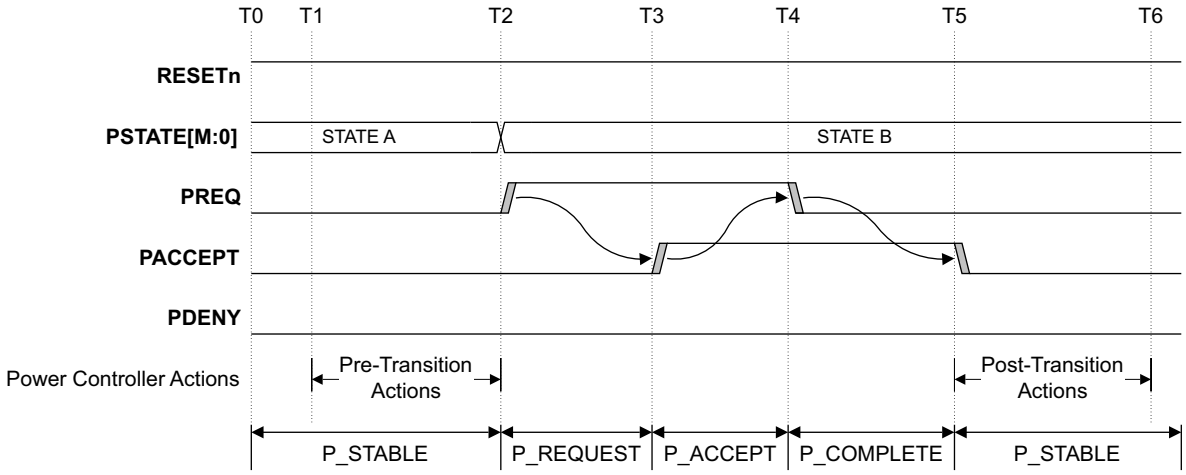

Figure 2-10 P-Channel handshake sequence with accepted request

Figure 2-10 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.

- 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 generally 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** is detected at the device. This is typically achieved by re-synchronizing **PREQ** to the device clock domain and using it as a strobe to capture **PSTATE**.
  - A controller must only present supported **PSTATE** values to a device. See *Device power state and transition support definition* on page 2-28.
- At T3 the device accepts the transition by driving **PACCEPT** HIGH. **PDENY** must be kept LOW. The device can now exploit 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 generally 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 the previous higher-power state.

Denied state transition

Figure 2-11 shows a transition request which is denied by the device.

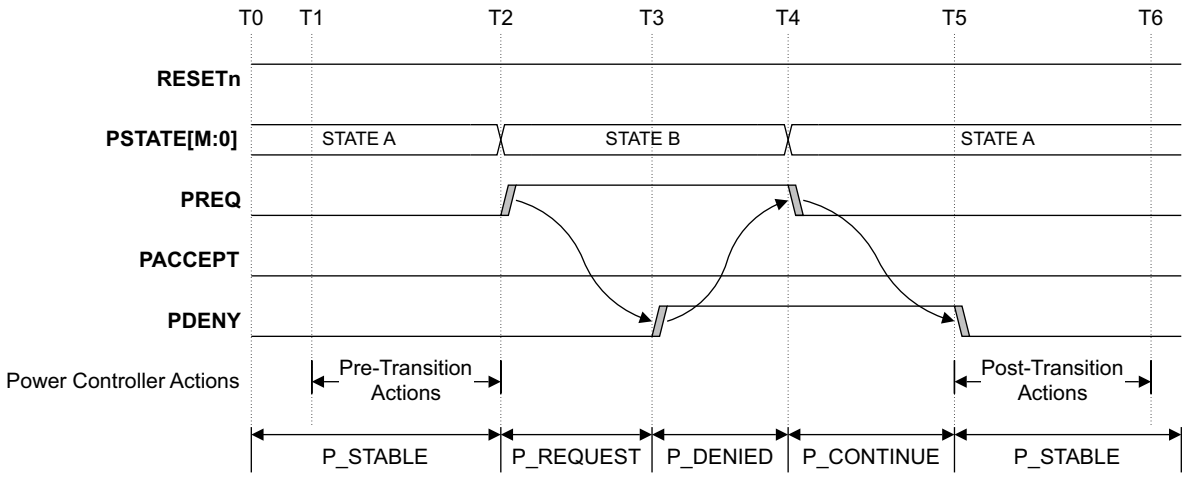

Figure 2-11 P-Channel handshake sequence with denied request

- T0, T1 and T2 follow the same sequence as that in *Accepted state transition* on page 2-21.
- 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 generally require reversing any pre-transition actions which were taken between T1 and T2. The denied transition sequence is complete and the interface state is again P_STABLE.

Note

At all times in the denial sequence the device remains in its pre-transition state and keeps all operational abilities associated with that 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. If no such requirement exists, ARM recommends all **PACTIVE** bits are reset LOW.

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

At reset deassertion, to facilitate device initialization to a desired state, the controller provides a **PSTATE** value which 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. **PSTATE** must be stable throughout this period. *Device power state and transition support definition* on page 2-28 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. This method is always applied to unused interfaces, see *Unused interface* on page 2-28.
- 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. ARM recommends 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 accept another P-Channel request. The device must accept any request made at reset exit or during t_{init}.

Figure 2-12 shows the case where the controller waits until t_{init} has expired before issuing a new transition, followed by assertion of reset.

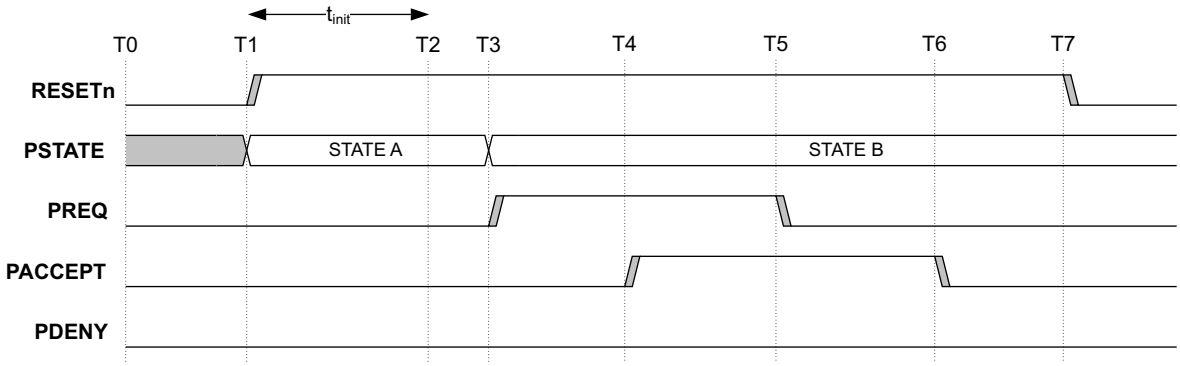

Figure 2-12 P-Channel initialization with controller-timed t_{init}, followed by a reset assertion

- 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.
- 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 2-13 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 accept another P-Channel request when this first transition completes. The device can complete this transition within t_{init}, but must be ready to accept another P-Channel request immediately after completing the first transition.

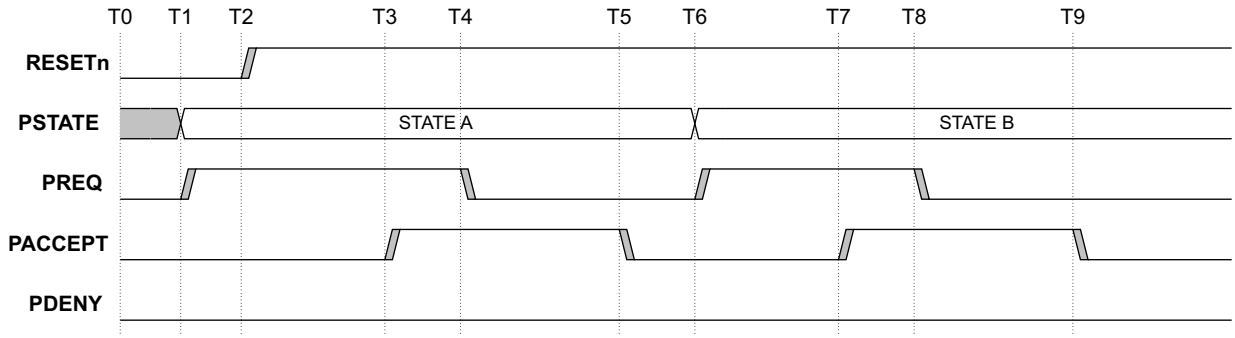

Figure 2-13 P-Channel initialization with PREQ HIGH at reset deassertion

Figure 2-14 shows the case where the controller sets **PREQ** HIGH after reset deassertion with the same **PSTATE** value as at reset deassertion. The controller then waits until the P-Channel transition is complete before issuing a further request. The device is guaranteed to be ready to accept another P-Channel request when the first transition completes. The device can complete this transition within t_{init}, but must be ready to accept another P-Channel request immediately after completing the first transition.

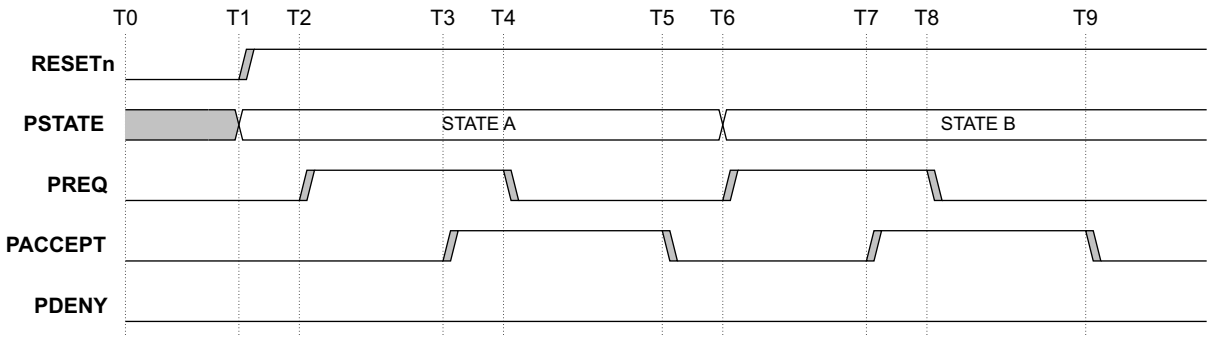

Figure 2-14 P-Channel initialization with PREQ HIGH during t_{init}

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 2-15 depicts 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.

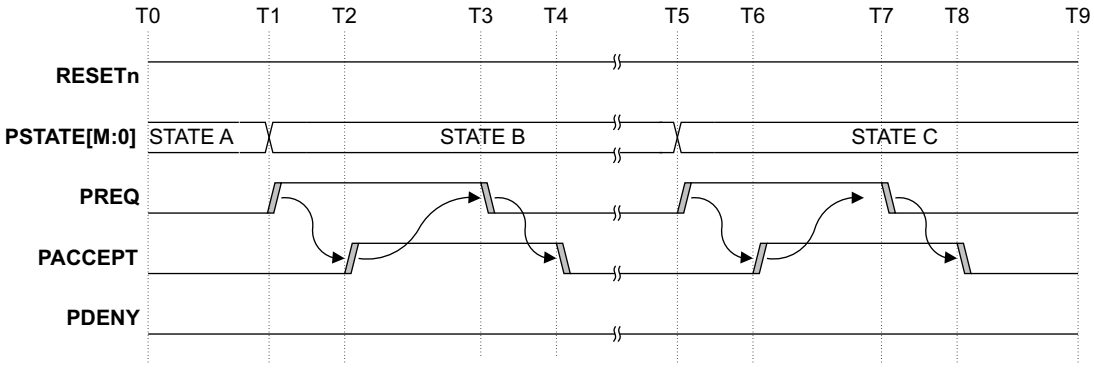

Figure 2-15 Multiple state transitions

- 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 state at T4.
- Between T5 and T8 the interface transitions from State B to State C without having to first return to State A.

Handshake states

Table 2-2 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 2-2 P-Channel handshake 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 sets **PREQn** LOW after P_DENIED. Device must set **PDENY** LOW.
x	x	1	1	Unused	Illegal signal combination.

Figure 2-16 is a state diagram showing the possible handshake sequences in terms of the signal states and interface states of each state change.

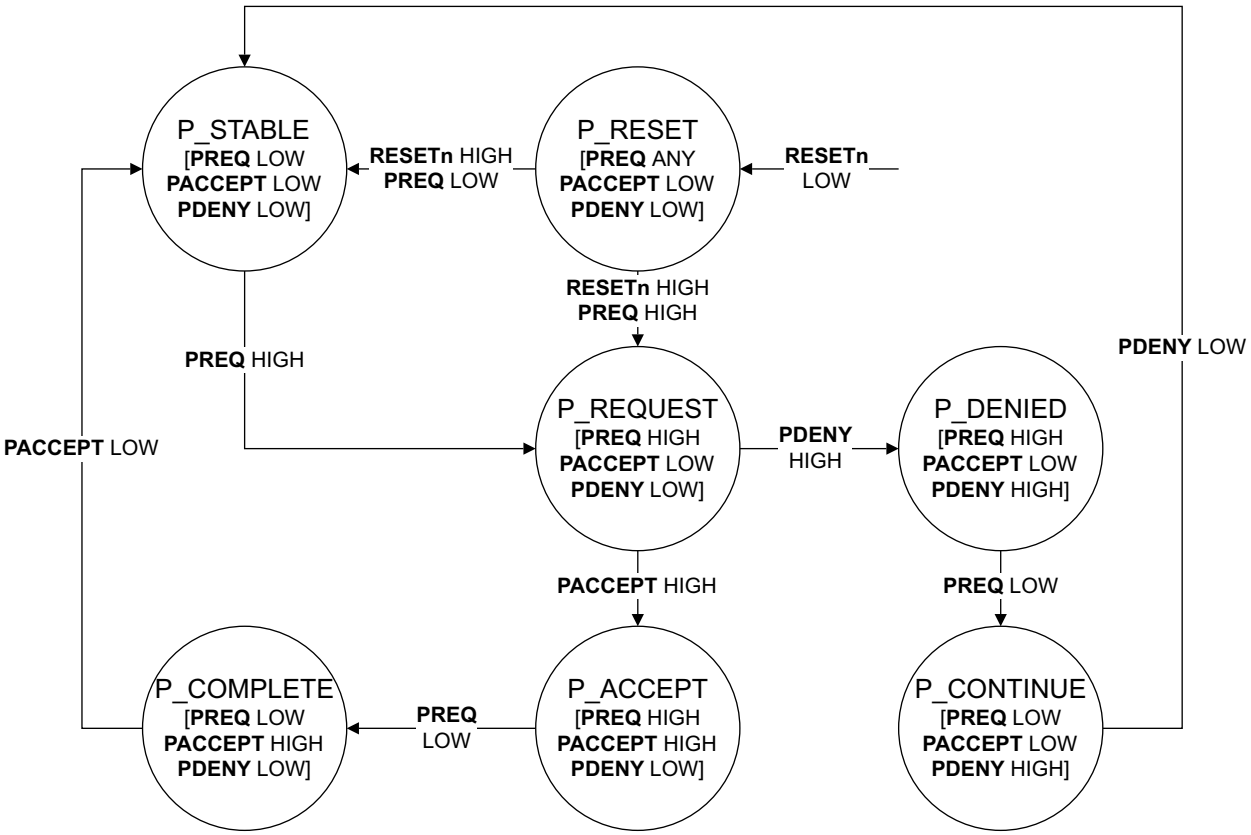

Figure 2-16 P-Channel state transitions

Handshake rules

The handshake signalling 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.

2.2.3 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 required

The P-Channel handshake is independent of **PACTIVE** and the controller can make any policy decision irrespective of any activity on **PACTIVE**. However, the device can deny any inappropriate request.

The arrangement of **PACTIVE** bits is not restricted. However, a typical **PACTIVE** arrangement is for each power state supported by the device to have an associated **PACTIVE** output bit, ordered from the lowest state on the LSB to the highest state on the MSB. The most significant **PACTIVE** bit set HIGH then indicates the minimum power state required by the device to process 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 most significant 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 most significant 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 4-38.

Controller policy and PACTIVE

Figure 2-17 shows an example of a device-led power transition based on changing **PACTIVE** values.

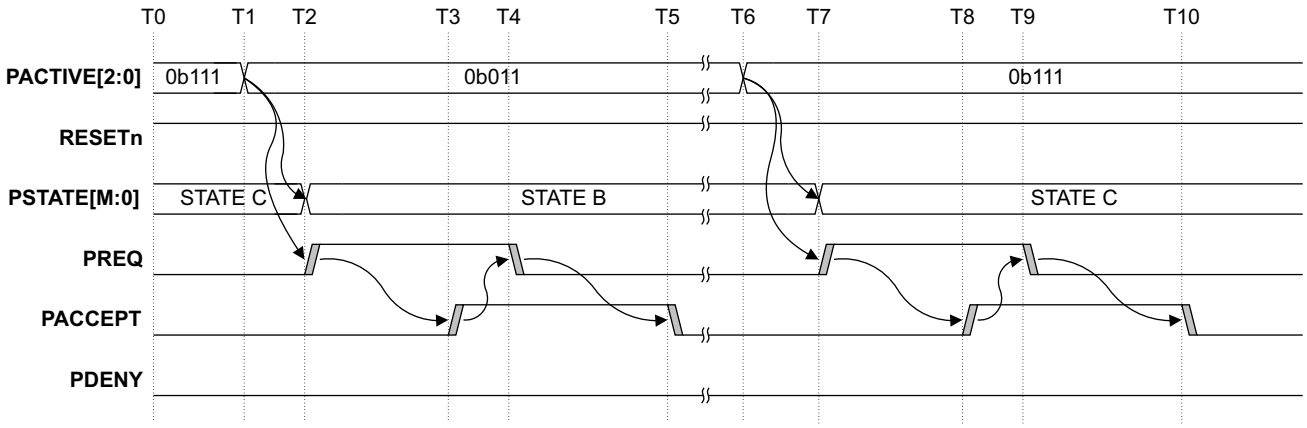

Figure 2-17 Device PACTIVE-led state transitions example

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 sequence is as follows:

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

2.2.4 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 which 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.
  - *Clock dependencies in multiple-interface devices* on page 4-34 outlines the resolution of dependencies between clock and power control interfaces in device design.

2.2.5 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.

This **PSTATE** value must correspond 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.

Chapter 3
Implementation Guidelines

The following sections describe implementation considerations that are important for the successful operation of the device interface. These considerations relate primarily to crossing clock domains.

3.1 Q-Channel implementation

Typically a device and a controller are implemented with asynchronous clocks, that might be driven from the same source clock but with a significant phase difference. Figure 3-1 shows the recommended implementation of the re-synchronization in this case.

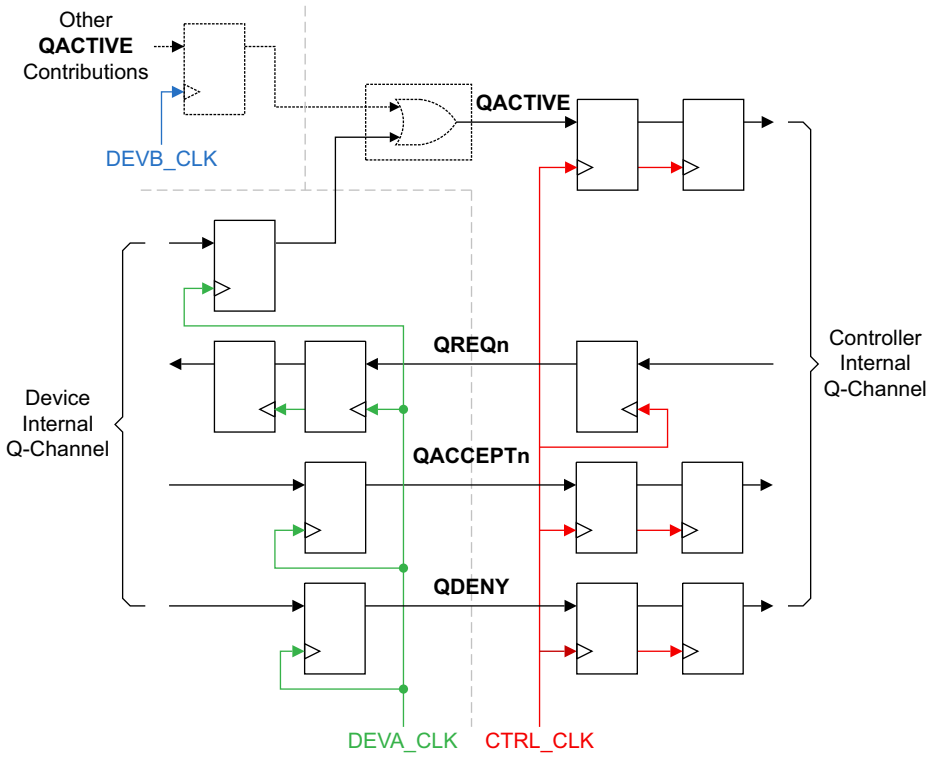

Figure 3-1 Recommended Q-Channel clock domain crossing synchronization

For an asynchronous Q-Channel implementation ARM provides the following guidance to implementers:

- Re-synchronize 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. ARM strongly recommends 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 **QACTIVE** source register changes on the output **QACTIVE** signal must be carefully considered. Although the handshake protocol guarantees functionally correct behavior regardless of **QACTIVE** behavior, ARM recommends implementing the simplest possible logic to minimize the likelihood of introduced glitches at the **QACTIVE** output.

Note

Although Figure 3-1 shows a typical two-stage synchronization, re-synchronization must be implemented appropriately for the technology libraries used and required frequency targets.

3.2 P-Channel implementation

Typically a device and controller are implemented with asynchronous clocks. Figure 3-2 shows the recommended implementation of re-synchronization in this case.

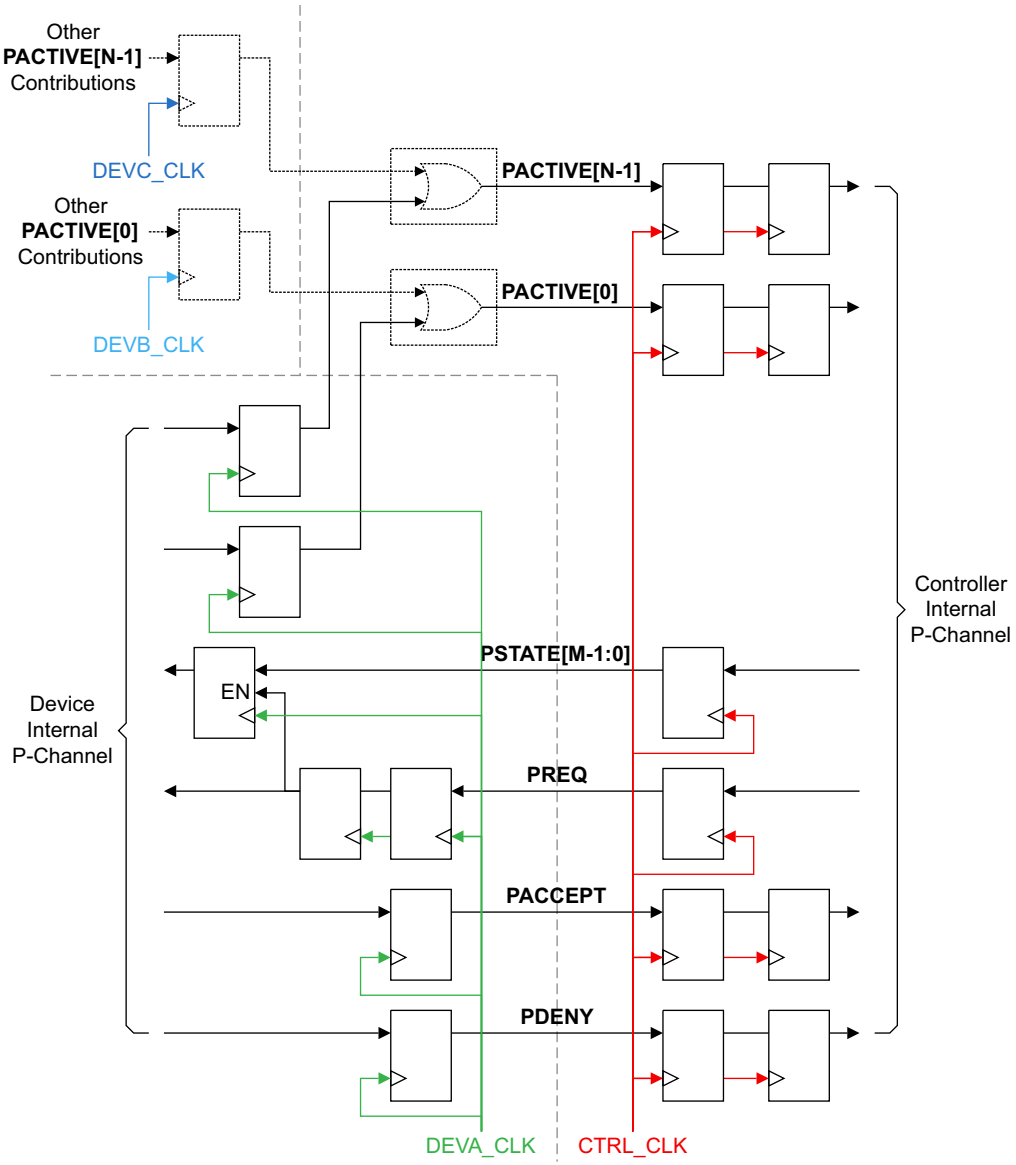

Figure 3-2 Recommended P-Channel clock domain crossing synchronization[1]

For an asynchronous P-Channel implementation ARM provides the following guidance to implementers:

- Re-synchronize 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 the **PACTIVE** signal is driven either directly by a register, or by a number of registers whose contributions are logically combined. ARM strongly recommends that this combining logic is limited to a logical OR, where possible, and is implemented using instantiated gates.

1. This figure does not show the mechanism for capturing **PSTATE** at reset deassertion.

- 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** behavior, ARM recommends implementing the simplest possible logic to minimize the likelihood of introduced glitches at the **PACTIVE** output.

Note

Although Figure 3-2 on page 3-31 shows a typical two-stage synchronization, re-synchronization must be implemented appropriately for the technology libraries used and required frequency targets.

3.2.1 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 as a strobe to capture **PSTATE**, as shown in Figure 3-2 on page 3-31.
- **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 3-3 shows an example where **PSTATE** is set before **PREQ** is driven HIGH.

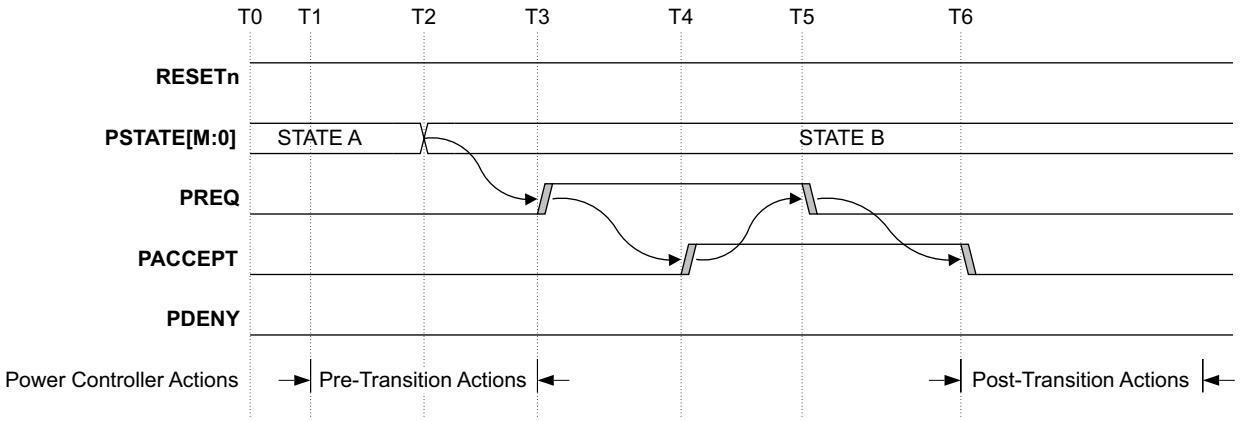

Figure 3-3 Additional timing for PSTATE by asserting PSTATE before PREQ

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.

3.2.2 Supported PSTATE initialization values

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

Chapter 4
Application Examples

This chapter describes examples of particular applications in which the low-power interfaces might be used. It contains the following sections:

4.1 Q-Channel application examples

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

4.1.1 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 is clock supply to the device, because the clock is required to process any operation or state change.

Figure 4-1 shows an example of a device with two interfaces.

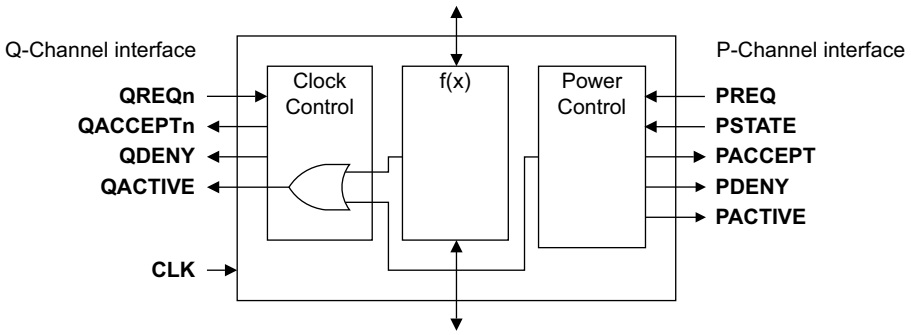

Figure 4-1 Device-level interface dependency example

The device in Figure 4-1 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 **PREQ** 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 simple 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.

A further consideration might be to ensure that, once the P-Channel interface has made a transition to a state where the device can be powered down, the clock control Q-Channel interface will then accept, rather than deny, any quiescence request overriding other conditions that would normally cause a denial response.

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.

4.1.2 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 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 simple example shown in Figure 4-2, 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.

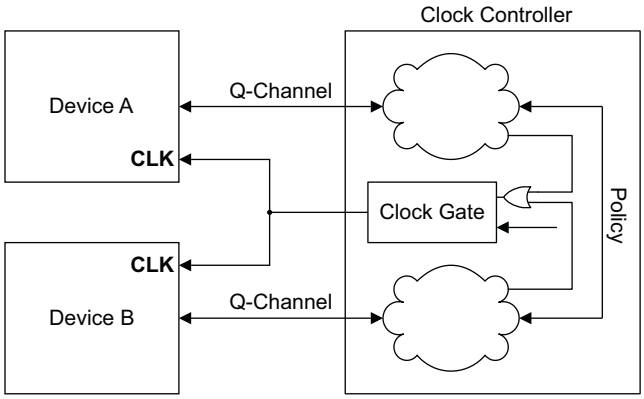

Figure 4-2 Interface combining at clock controller example

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, or by adding control sequence dependencies.

4.1.3 Application of QACTIVE-only interface

One application for a device with only **QACTIVE** is in combination with the **QACTIVE** of another device that supports the handshake signals, to form an interface to a controller. Figure 4-3 shows an example of this application.

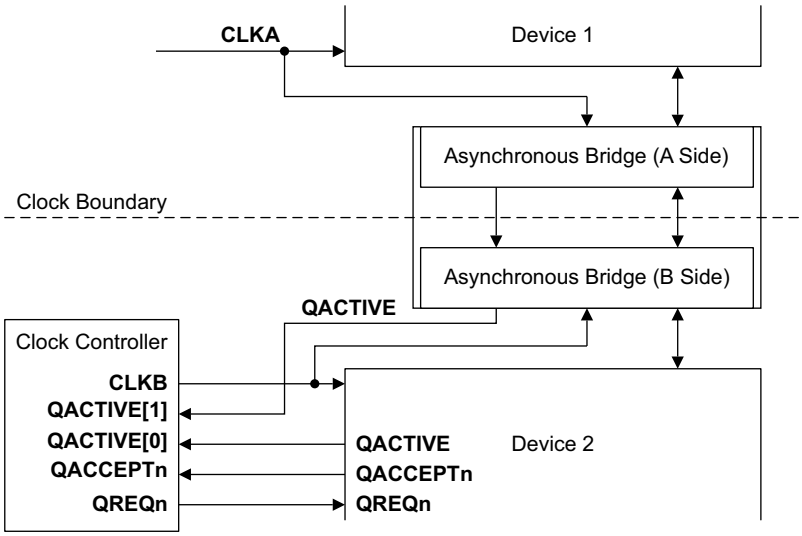

Figure 4-3 Application of device with QACTIVE only

Figure 4-3 shows two devices that are interfaced with an asynchronous bridge component.

When Device 1 sends a transaction towards Device 2, it cannot progress from the A side of the bridge to the B side unless **CLKB** is active. To provide an initial wakeup capability a signal is routed from the A side of the bridge though the B side of the bridge to give a **QACTIVE** output from the bridge. This signal is driven HIGH whenever the A side of the bridge has a transaction pending for the B side of the bridge.

The **QACTIVE** output from the bridge is then combined with the **QACTIVE** output of Device 2 at the clock controller. The clock controller then initiates a handshake with Device 2 and activates **CLKB**. Once **CLKB** is activated, Device 2 can then manage the handshake with the controller, and can also take responsibility for tracking the transaction and the activity of its own **QACTIVE** output.

4.2 P-Channel application examples

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

4.2.1 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 4-4 shows a simple example with three power domains across two devices.

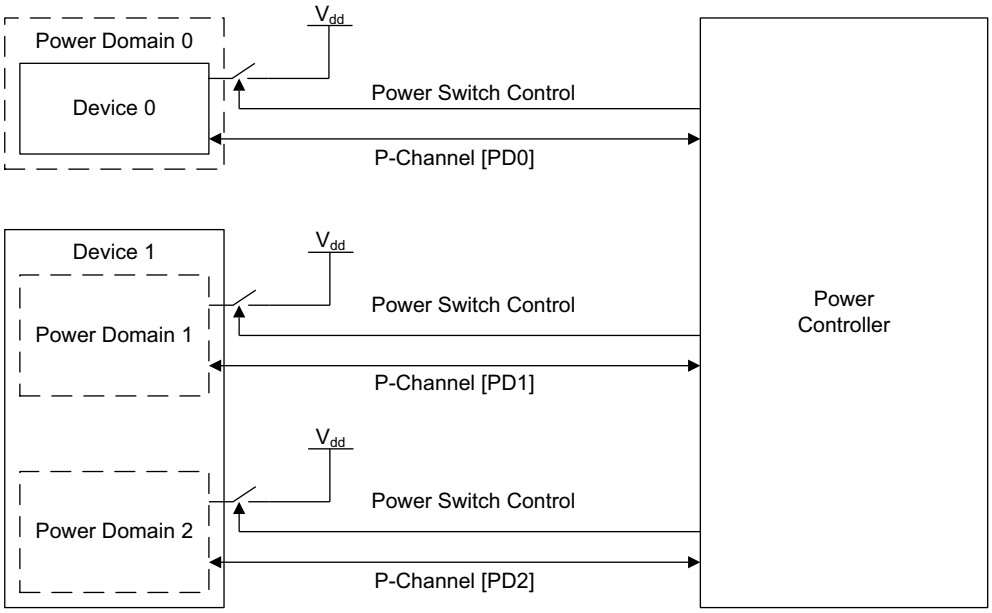

Figure 4-4 Simple power domain example

In Figure 4-4, 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 signalling using **PACTIVE** from states without operable logic. In this state, the system is responsible for wakeup signalling.

Also, transitions between power states without operable logic require intermediate transitions to and from the operable state, and a single transition between successive lower-power states cannot be made. Since some logic functionality is required at minimum to respond to the P-Channel requests, this arrangement cannot support RAM-only power domains.

Figure 4-5 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.

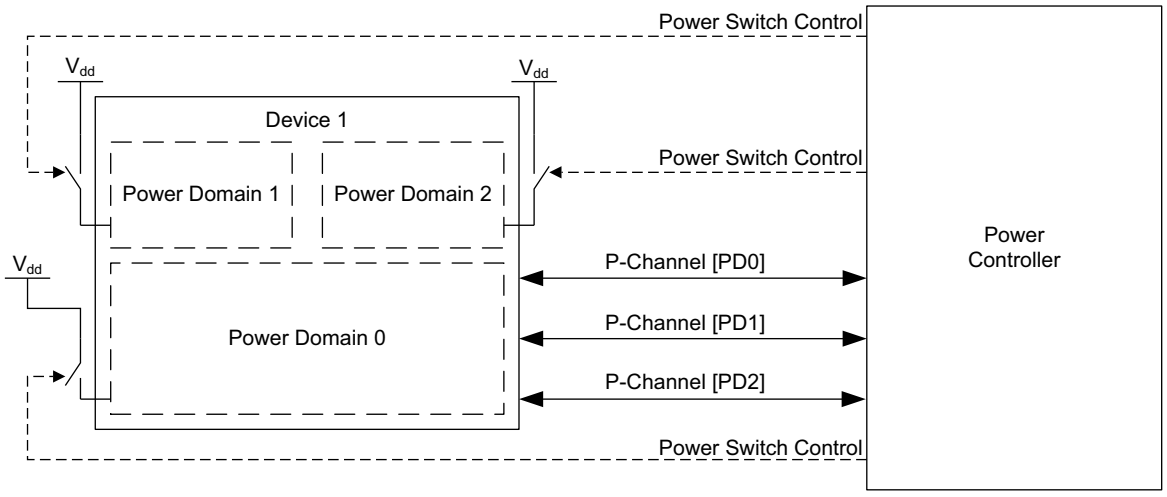

Figure 4-5 Parent-child power domain example

In Figure 4-5, 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 is located within Power Domain 0.

In this case, the device capabilities can include parent domain detection and signalling 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 with absent logic capability, such as a RAM-only power domain.

4.2.2 Example PACTIVE usage

Figure 4-6 shows an example of device power states and supported transitions.

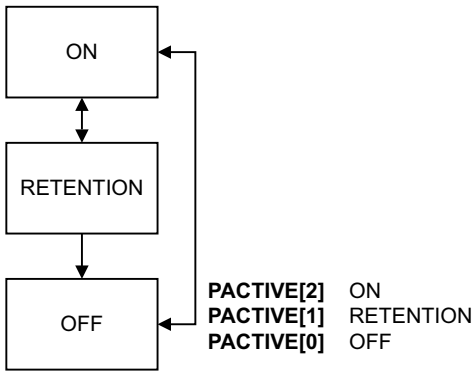

Figure 4-6 Example device power states and PACTIVE enumeration

Table 4-1 shows the device power states and related P-Channel behavior.

Table 4-1 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	Yes
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 must 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.

Chapter 5
Q-Channel Backward Compatibility

This chapter describes connection mappings for ensuring backward compatibility with AXI low power interface devices and controllers. It contains the following section:

5.1 Q-Channel backwards compatibility

Figure 5-1 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.

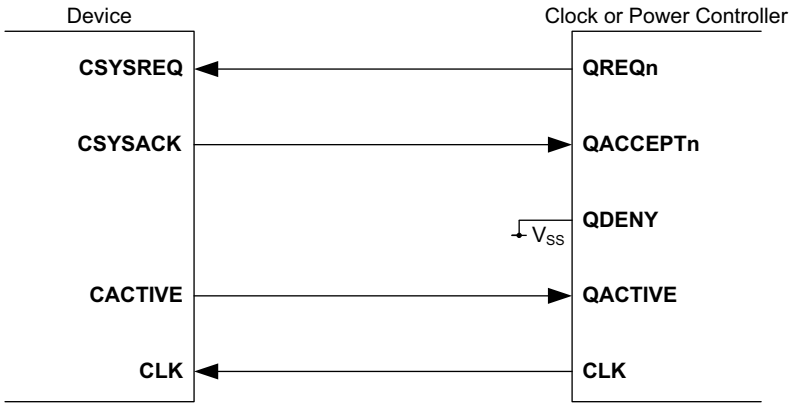

Figure 5-1 Connection of Q-Channel controller to an AXI LP device without denial support

Figure 5-2 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.

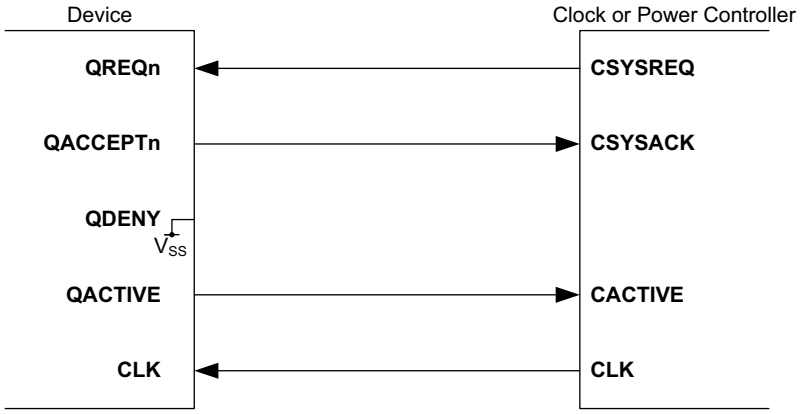

Figure 5-2 Connection of Q-Channel device to an AXI LPI controller, both without denial support

For a device supporting the AXI low-power interface that uses the denial mechanism, solutions are limited to instances where all interface signal are confined to a single synchronous clock domain. Within those restrictions a simple adaptor component, to generate the Q-Channel acknowledgement signals, can be implemented as shown in Figure 5-3. This component has a sequential requirement to ensure correct glitch-free handshaking at the controller.

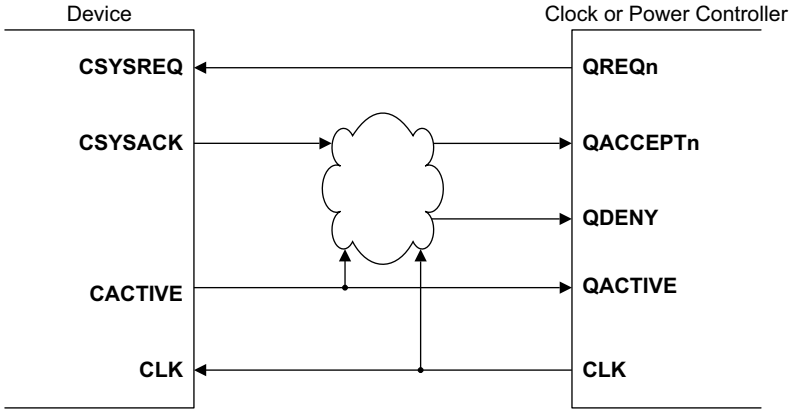

Figure 5-3 Adaptation of AXI LPI device with synchronous denial support to a Q-Channel controller

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**.

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 2-12. • *Device activity indication* on page 2-19. • *Q-Channel implementation* on page 3-30. • *P-Channel implementation* on page 3-31.
Clarified points at which **PSTATE** must be stable.	• *Denied state transition* on page 2-22. • *Device reset and initialization* on page 2-22.
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 2-2 on page 2-25.

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 3-30.
Added Note about what the figure does not include.	*Recommended P-Channel clock domain crossing synchronization* on page 3-31.
Added information about capturing PSTATE.	*Capturing PSTATE* on page 3-32.