High-level Considerations for Power Management of a big.LITTLE™ System

Version: 1.0

Application Note 424

ARM®

High-level Considerations for Power Management of a big.LITTLE System
Application Note 424

Copyright © 2016 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

<div align="right">Change history</div>

Date	Issue	Confidentiality	Change
20 July 2016	A	Non-Confidential	First release

Proprietary Notice

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

Preface

This preface introduces the *High-level Considerations for Power Management of a big.LITTLE™ System*. It contains the following sections:

- *About this application note* on page vii.
- *Feedback* on page ix.

About this application note

This application note is organized into the following chapters:

- Chapter 1 *Introduction*

  Read this chapter for information about the purpose of the application note.

- Chapter 2 *Power-down and power-up considerations*

  Read this chapter for high-level considerations for powerdown and powerup.

- Chapter 3 *Potential SoC integration issues*

  Read this chapter for potential issues when implementing power management for processor cores or clusters on a typical SoC.

- Chapter 4 *Hardware considerations*

  Read this chapter for general advice from the hardware perspective when implementing power-down and power-up sequences for big.LITTLE systems.

Intended audience

This application note is written for hardware *System on Chip* (SoC) designers implementing power-down and power-up sequences for ARM processors.

This document assumes that you have SoC design experience and are familiar with ARM products.

Glossary

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

See *ARM® Glossary, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html*.

Conventions

This book uses the conventions that are described in:
- *Typographical conventions*.

Typographical conventions

The following table describes the typographical conventions:

Typographical conventions

Style	Purpose
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

Additional reading

A *Technical Reference Manual* (TRM) describes the power-down and power-up sequences of a particular core. You can download the latest TRMs directly from the ARM infocenter at `http://infocenter.arm.com`.

- *ARM® Cortex®-A7 MPCore Technical Reference Manual* (ARM DDI 0464F).
- *ARM® Cortex®-A15 MPCore Processor Technical Reference Manual* (ARM DDI 0438I).
- *ARM® Cortex®-A17 MPCore Processor Technical Reference Manual* (ARM DDI 0535C).
- *ARM® Cortex®-A53 MPCore Processor Technical Reference Manual* (ARM DDI 0500F).
- *ARM® Cortex®-A57 MPCore Processor Technical Reference Manual* (ARM DDI0488G).
- *ARM® Cortex®-A72 MPCore Processor Technical Reference Manual* (ARM 100095_0002_03_en).
- *ARM® Cortex®-A73 MPCore Processor Technical Reference Manual* (ARM cortex_a73_trm_100048_0002_04_en).

For access to the following documentation, please contact ARM:

- *ARM® Power Control System Architecture Specification Version 1.0* (ARM DEN 0050).

Feedback

ARM welcomes feedback on this documentation.

If you have comments on content, send an e-mail to `errata@arm.com`. Give:
* The title.
* The number, ARM DAI 0424.
* 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 the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

Read this chapter for information about the purpose and scope of this application note.

It contains the following sections:
- *Document purpose* on page 1-2.
- *Document scope* on page 1-3.

1.1　Document purpose

Power management is important, and has become increasingly complex. The application note provides high-level considerations for power management of a big.LITTLE system, and is intended to help you avoid some potential issues in your big.LITTLE design.

For more comprehensive implementation considerations about power management, see the *ARM® Power Control System Architecture Specification Version 1.0*. You can contact ARM to get access to this document.

1.2 Document scope

This application note focuses on the following processors and highlights important issues when powering up or powering down processor cores and clusters on an SoC.

- Cortex®-A7.
- Cortex®-A15.
- Cortex®-A17.
- Cortex®-A53.
- Cortex®-A57.
- Cortex®-A72.
- Cortex®-A73.

The application note also uses a typical SoC design example to show how to implement correct power-down and power-up sequences for these processors.

Chapter 2
Power-down and power-up considerations

Read this chapter for high-level considerations for powerdown and powerup.

It contains the following sections:

2.1 ARM® big.LITTLE™ system example

A big.LITTLE system uses two different types of cores that are combined in a coherent system. big cores are designed for high performance while LITTLE cores are designed for high energy efficiency. The big cores are used for resource-intensive software threads, and energy-efficient LITTLE cores handle low-intensity software threads that use fewer compute resources. The result is high energy efficiency and high performance.

Idle management techniques include dynamically switching cores on or off, or putting the cores in a low-power state, such as Standby or Dynamic Retention.

——— **Note** ———

Dynamic Retention is not supported on the Cortex-A7 and Cortex-A15 (revisions prior to r3p0) processors.

Figure 2-1 on page 2-3 shows the structure of a big.LITTLE system, which includes the following basic components:
- Two clusters: A big cluster and a LITTLE cluster.
- The *CoreLink™ Cache Coherent Interconnect* (CCI).
- The *CoreLink™ Generic Interrupt Controller* (GIC).
- The Generic Counter.
- The power controller for power, clock, reset, and other signals.
- Domain Bridges.
- Event Bridges.
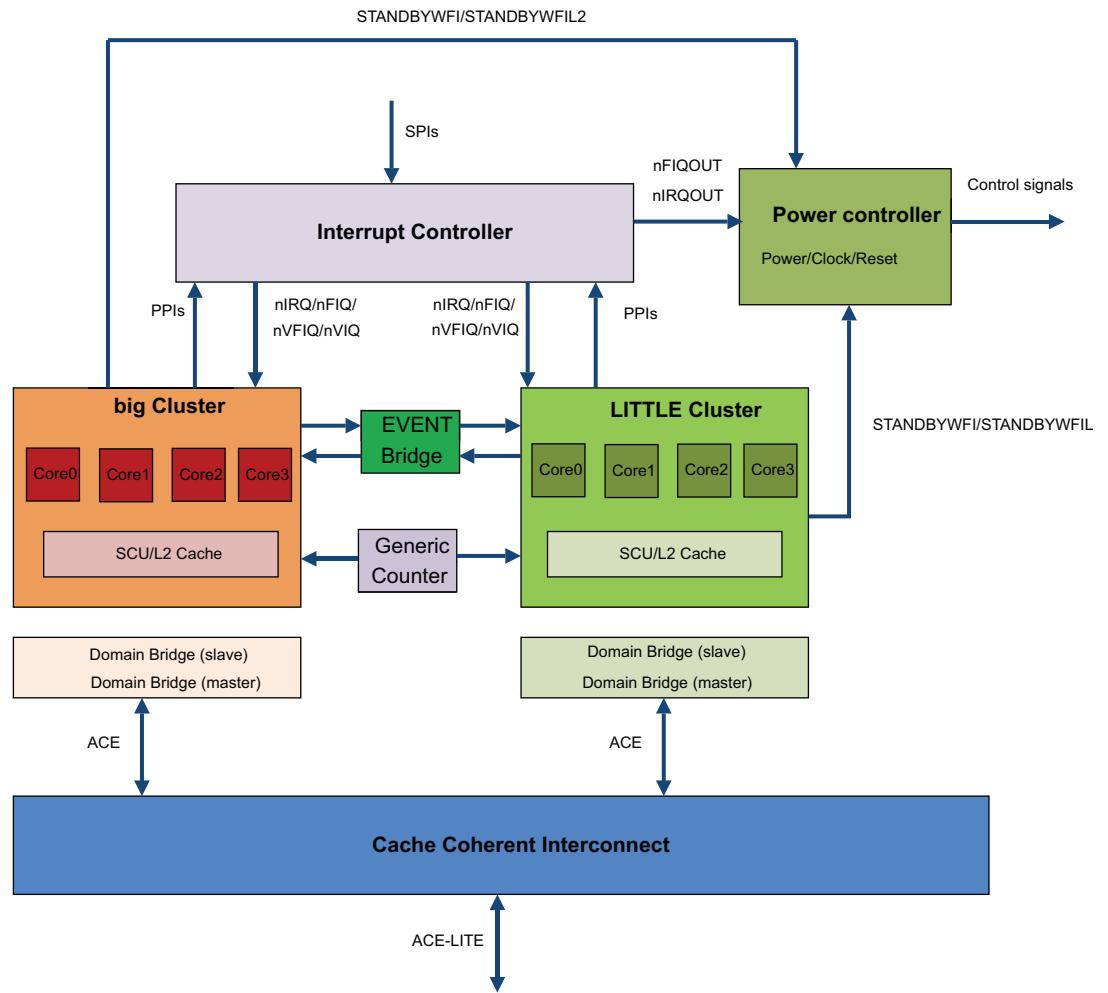
Figure 2-1 A typical big.LITTLE system

2.1.1 big cluster and LITTLE cluster

This example system uses a big cluster and a LITTLE cluster, and each cluster consists of four cores.

There are five power domains for each cluster. Each core is in its own power domain to enable individual powerdown. The rest of the logic within a cluster is in a different power domain. Isolation cells exist between power domains for isolation.

Each cluster is implemented in an independent voltage domain with an independent clock source. Each of the clusters has a unique Dynamic Voltage and Frequency Scaling (DVFS) curve. Therefore, cores in the same cluster must traverse an *Operating Performance Point* (OPP) together. You can change the OPP of cores in a cluster, without affecting the cores in the other cluster.

2.1.2 Power controller

The power controller contains the hardware power control functions of the SoC. The operations of the power controller are both directed by the *Operating System Power Management* (OSPM) software and hardware events in the system. The power controller works closely with relevant signals of ARM IPs and other system components for power saving.

2.1.3 Cache Coherent Interconnect

The *Cache Coherent Interconnect* provides full cache coherency between the two clusters. Typically, the CCI-550 is used in a big.LITTLE system.

2.1.4 Domain Bridges

A Domain Bridge passes a protocol transaction from one domain to another. This includes voltage, power, and clock domains or a combination.

Domain Bridges exist between a cluster and the CCI for the voltage domains to enable DVFS. The slave side of the Domain Bridge shares the same power domain with the L2 and SCU logic of the cluster. Typically, an ADB-400 is used as the Domain Bridge.

2.1.5 Interrupt Controller

The Interrupt Controller comprises a set of hardware resources for managing interrupts in a big.LITTLE system. Typically, the GIC-500 is used in a big.LITTLE system, and is an ARM implementation of the GICv3 Architecture.

2.1.6 Event Bridge

An Event Bridge synchronizes events on a single channel from the master domain to the slave domain. The event acknowledgement from the slave domain is synchronized and presented to the master domain.

2.1.7 Generic Counter

The Generic Counter provides a uniform view of system time.

2.2 High-level considerations

This section describes high-level considerations when implementing the power-down and power-up sequences for an SoC:

- *System partitioning*.
- *Coherency in a big.LITTLE system* on page 2-6.
- *Power ordering* on page 2-6.
- *Interrupts* on page 2-8.
- *Events* on page 2-8.
- *Downstream components* on page 2-8.
- *Power-up supply* on page 2-8.
- *Debug requirements* on page 2-9.

2.2.1 System partitioning

An SoC based on ARM components can be partitioned into voltage, power, and clock domains.

Voltage domains

The voltage supply to a domain might be scaled or removed for power or performance reasons.

Except for low complexity solutions, it is rare to use a single logic voltage supply for the whole SoC.

The primary reason for additional voltage domains is to support DVFS for functional areas of the SoC. The second reason is to enable external supply switch-off, or reduction to non-functional state retention levels, to some logic areas while maintaining an operational level supply to others.

However, the cost for additional voltage domains is significant, because additional voltage regulators, extra effort, and complexity are required in the SoC physical implementation. Therefore, you must carefully assess the value of the addition of each voltage domain against the performance and power requirements for the design.

In a big.LITTLE system, each cluster must have a dedicated voltage supply. This is a critical success factor when combined with big.LITTLE software.

Power domains

A voltage domain can be partitioned into one or more power domains. A power-gated domain is a power domain whose power can be removed by on-chip power switches. A voltage domain can thus be partitioned into power-gated domains and an always-on power domain.

The purpose of the power domain strategy is to minimize the powered-on area in a given scenario.

ARM recommends that you implement core and cluster power domains in each of the big and LITTLE clusters. The power controller is always on. For other components in Figure 2-1 on page 2-3, place them in a system power domain.

Clock domains

Clock domains can interact with each other synchronously or asynchronously. Synchronous clock domains can have independent source activity. Each cluster requires an independent clock, and the CCI requires a clock.

2.2.2 Coherency in a big.LITTLE system

Coherency is a key ingredient that makes big.LITTLE technology possible. big.LITTLE software models require transparent and seamless transfer of data between big and LITTLE processors. Hardware coherency transparently enables seamless transfer of data. Without hardware coherency, the transfer of data between big and LITTLE cores must occur through the main memory. This is slow and not power-efficient, and it requires complex cache management software to enable data coherency between big and LITTLE processors.

Figure 2-1 on page 2-3 is an example of a CPU subsystem consisting of a big cluster, a LITTLE cluster, and a set of system fabric components. The system fabric components enable seamless data transfer between clusters. This fabric is collectively referred to as the CCI.

Core in cluster coherency

The processors described in this document implement write-back caches. When a core executes store instructions to Normal cacheable memory, the data cache of the core might hold data newer than the data in downstream caches or main memory. Therefore, the core must clean its entire data cache before its power is removed.

Cluster in system coherency

When multiple clusters are in the same coherent domain of an SoC, cache access and maintenance operations performed on a core might issue an AXI Coherency Extension (ACE) transaction to a core in another cluster. The CCI transfers the ACE transaction to the ACE AC channel of another processor to snoop the caches of the cores inside that cluster.

To power down a multi-core cluster entirely, the cluster L2 cache must be flushed first. In addition, the system must ensure that no ACE snoop transaction is issued to the ACE AC channel of the cluster, by using suitable system control of relevant IPs, such as coherent interconnect. The processor provides an interface for the system to notify the cluster that no further snoops are being sent.

Similarly, if a cluster supports an Accelerator Coherency Port (ACP) slave interface, an ACP access transaction that is issued to its ACP slave interface accesses caches of the cores inside the cluster too.

To power down a cluster with an ACP slave interface, the cluster L2 cache must be flushed first. In addition, the system must ensure that no ACP access transaction is issued to the cluster, by using suitable system control of relevant IPs, such as an ACP master. The processor provides an interface for the system to notify the cluster that no transaction is currently issued to the ACP slave interface.

2.2.3 Power ordering

To provide a functional system that can run a standard OS, you must maintain some fundamental relationships among critical resources in an SoC based on ARM components. There are some high-level considerations for power domain hierarchies to maintain these relationships.

Figure 2-2 on page 2-7 shows the power domains of the example SoC.
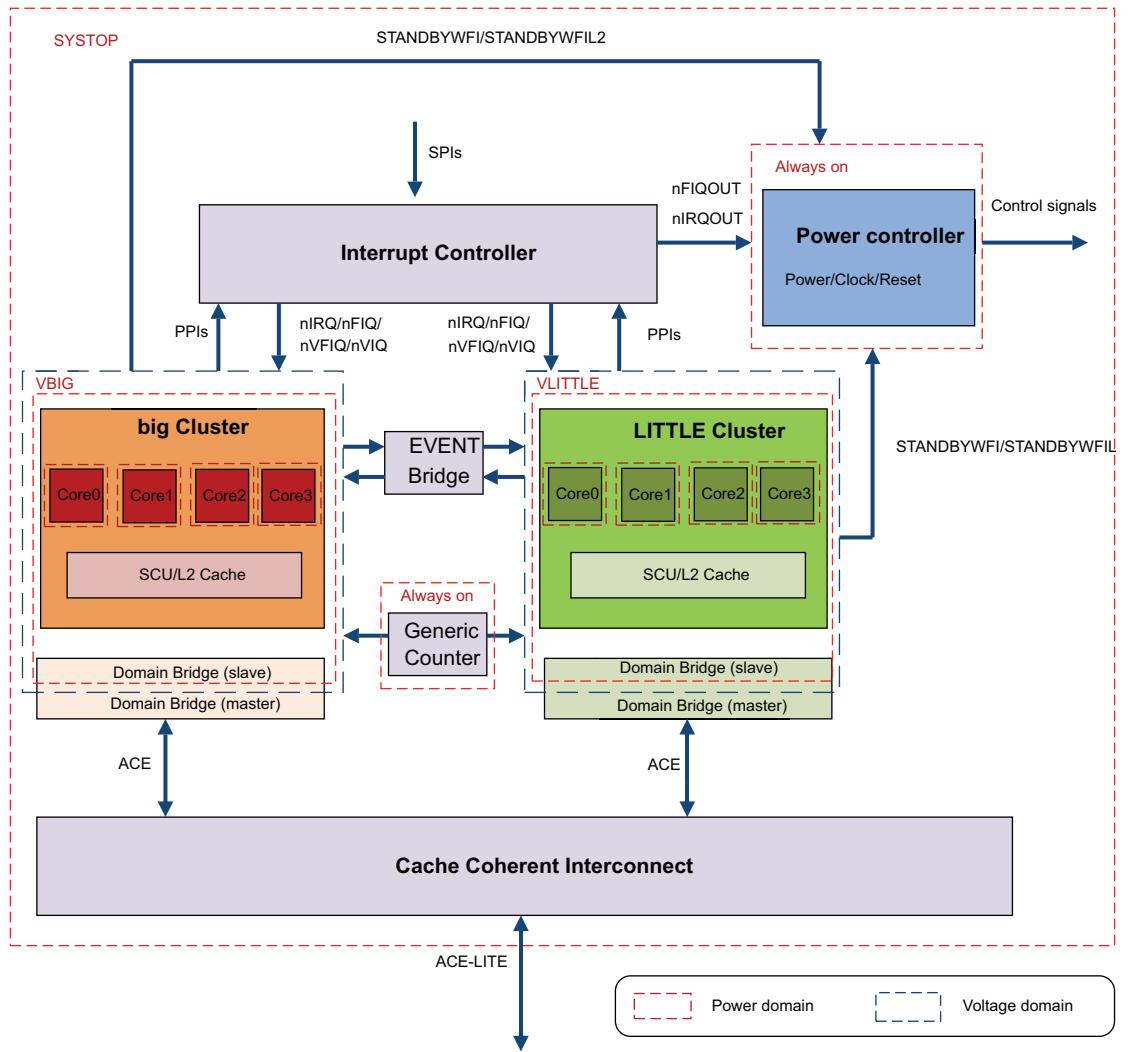
Figure 2-2 Power domain hierarchy

In Figure 2-2, two processor clusters are implemented with per-core and cluster power domains. The big cluster is in the VBIG voltage domain, while the LITTLE cluster is in the VLITTLE voltage domain. The system controller is in an always-on power domain. All the other components are in the system logic power domain (SYSTOP).

The power ordering requirements for the example in Figure 2-2 are as follows:

• The always-on domain must be available whenever the SoC is powered up.

• The SYSTOP domain must be available whenever any processor cluster is powered up.

  This means that the GIC, system memory, time distribution, and Event Bridge resources are available whenever any processor is powered up.

• The VBIG or VLITTLE domain must be available whenever any of its cores is powered up.

2.2.4 Interrupts

An interrupt that is routed to a core might be triggered at any time. For example, the interrupt might occur during the power-down process or after the power is removed. The system must handle the interrupt appropriately.

The interrupt can be serviced, rerouted to another core, or made pending in the GIC. If the core is already powered down and a pending interrupt requires this particular core to service it, the interrupt must remain asserted, not serviced, until the system powers up the core. When the core is restored, the core services the interrupt. In this case, the interrupt is routed as a wake request to the power controller.

2.2.5 Events

The Wait-For-Event (WFE) mechanism allows a core in a multiprocessor system to request entry to a lower-power state. If the request succeeds, the core remains in the low-power state until it receives an event generated by a Send Event operation on another core in the system.

The clusters in a big.LITTLE system provide signals to communicate Send Event Operations between clusters. Each cluster has an **EVENTI** input to receive an event from the other cluster. In addition, each cluster has an **EVENTO** output to send an event to the other cluster.

When integrating each cluster, you must ensure that the EVENTO output from one cluster is considered as a factor for the **EVENTI** input to the other cluster. You can also consider other system events as factors for the **EVENTI** input to each cluster.

Similarly, you can also consider the EVENTO output from each cluster as a factor for a WFE mechanism that is part of a different system component. In doing so, you must consider the clock and power domains of the clusters. ARM recommends that you build a clock domain crossing circuit that handshakes each event between the clusters.

It is acceptable for two or more events that are sent in close succession to be combined into a single event. The resulting event must be received after all of the events that are combined have been sent. Clusters that are powered off are not required to receive events.

2.2.6 Downstream components

There will be some downstream components connected to a processor. If the processor and its downstream components share a power domain, the system must ensure that operations on the downstream components are completed before the power supply is removed from the shared power domain for all relevant components.

2.2.7 Power-up supply

The internal state of the core is unpredictable when power is reapplied to it. This initial unstable phase must not affect the rest of the SoC. You can avoid this problem by following the TRM requirement that all relevant reset signals are asserted for the required number of clock cycles before releasing the output signal clamps.

When powering up a core, the system design must ensure that the reapplication of power does not have adverse effects on other logic or cores in the system. For example, power might be reapplied gradually to different components in the system to ensure that there are no sudden spikes in current draw.

2.2.8   Debug requirements

When designing an SoC, ARM recommends that you consider the following issues, and define a power controller to support specific debug requirements of the SoC, based on core power-up and power-down scenarios:

*   *Debug No Powerdown*.
*   *Debug Over Powerdown*.
*   *Debug power-up request*.

Debug No Powerdown

Power-up and power-down sequences are often problematic areas in an SoC design. When designing an SoC, ARM recommends that you implement the *Debug No Powerdown* mode, which allows the power-down sequence to be executed in an emulation mode. In this emulation mode, every step of the power-down sequence is conducted, except for removal of the power supply to the core.

Debug Over Powerdown

Debug Over Powerdown mode enables the debugger to get information about a processor even when the processor is already powered down.

Most ARM multi-core processors support Debug Over Powerdown for the individual core if the non-CPU logic, often called *L2* logic, is powered up. If all cores inside the multi-core processor are powered down, the system might power down non-CPU logic within the multi-core processor to enter a lower power state. As a system designer, you must consider whether and how to maintain power to non-CPU logic to keep Debug Over Powerdown capabilities of the powered-down cores still available.

Debug power-up request

The SoC can define whether to use a debugger power-up request or specific debug events, or both, as the processor wakeup condition.

These wakeup conditions must be detectable in the system, for example as interrupt inputs, or in software, by power control logic.

Chapter 3
Potential SoC integration issues

Read this chapter to see potential issues when implementing power management for processor cores or clusters on a typical SoC. This chapter is intended to help you avoid these issues in your design.

It contains the following sections:

3.1 Interrupts

Potential interrupt issues are as follows:
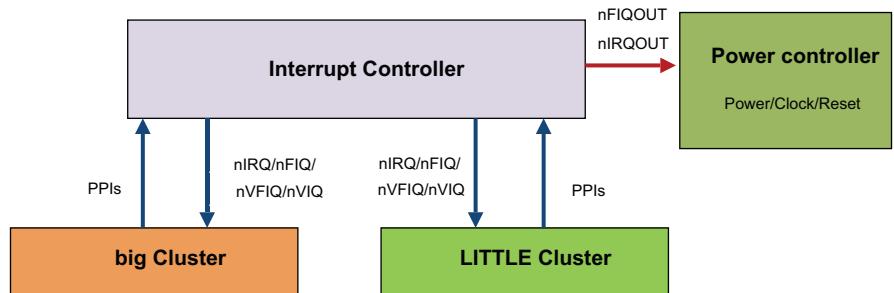
- Wake request signals are not used, as described in *Must use wake request signals*.

- Wake request signals are ORed, as described in *Must not OR wake request signals* on page 3-3.

3.1.1 Must use wake request signals

In addition to normal interrupt outputs, the GIC architecture defines wake request signals. These are connected to the power controller to indicate that the associated core must be woken to process an interrupt. When the core is powered up, the interrupt can be sent to the core.
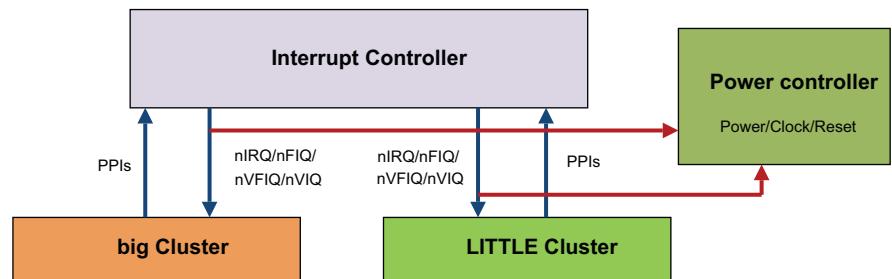
GICv2

The GICv2 architecture defines two wake request signals for each core, **nFIQOUT** and **nIRQOUT**. As shown in Figure 3-1, you must connect **nFIQOUT** and **nIRQOUT** to the power controller so that when an interrupt is triggered, the power controller is notified that the core is required to be powered up.


Figure 3-1 Proper nFIQOUT and nIRQOUT integration

A potential mistake is that the wake request signals, **nFIQOUT** and **nIRQOUT**, are not connected to the power controller. Instead, **FIQ** and **IRQ** are connected to the power controller, as shown in Figure 3-2. This mistake would cause system failure.
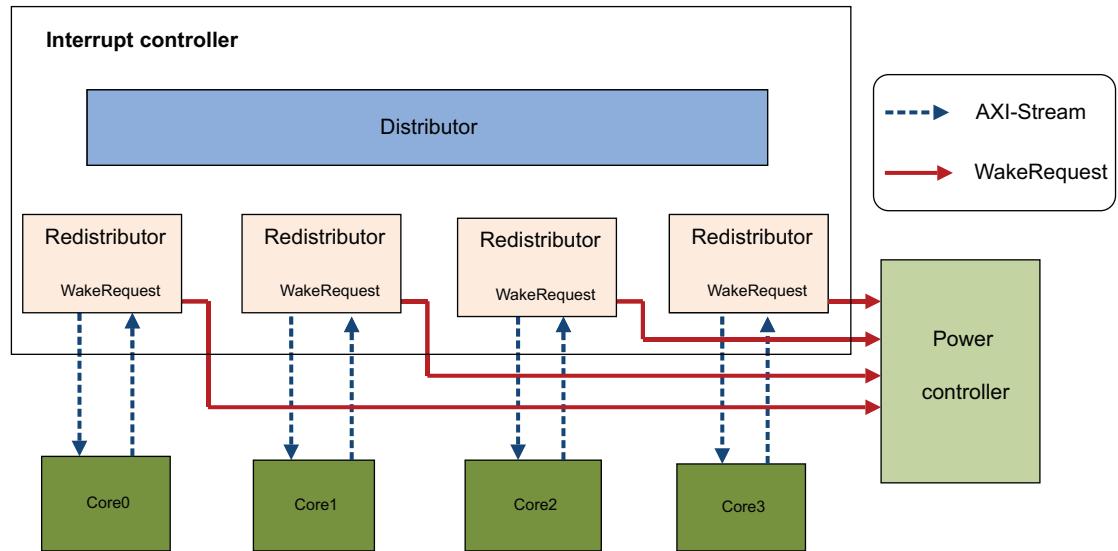

Figure 3-2 Improper nFIQOUT and nIRQOUT integration

GICv3

When a core powers up, it must connect the GIC CPU interface to the GIC redistributor. Before powerdown, it must disconnect this interface. When the redistributor to the GIC CPU interface is disconnected and there is an interrupt pending for that core, the redistributor indicates the core wake requirement by using the **WakeRequest** signal.

To wake a core, a GICv3 implementation provides a **WakeRequest** output signal from each Redistributor for the associated core, as shown in Figure 3-3.


Figure 3-3 Correct GICv3 Integration

When **GICR_WAKER.ProcessorSleep** is set HIGH, the **WakeRequest** signal is asserted when an interrupt specifically targeting that core is received. Each **WakeRequest** signal must be connected to the power controller.

3.1.2 Must not OR wake request signals

The wake request signals for a specific core can be combined to produce a single wake signal to the power controller. However, the wake request signals for multiple cores must not be combined.

A separate wake requirement for each core must be separately visible to the power controller, so it can determine which core to power up. That is, you must keep wake request signals as per-core signals, so the power controller knows which core it needs to wake.

If you combine wake request signals for multiple cores and OR the wake request signals, the power controller cannot determine which core to wake up.

3.2 Core power sequencing assumptions

A common misconception is that a specific application processor core is always the first to power up and the last to power down. For example, you cannot assume that CPU0 is always first-on and last-off. SoCs with that assumption can cause many problems for software.

At cold boot time, it is possible to force an ordering, so that a specific core is always the first to power up. At run time, however, the operating system scheduling can idle any core at any time. When a core enters idle state, the core is a candidate for power-down selection.

Furthermore, the operating system can target interrupts to any core in the system. The targeted core must power up when the interrupt becomes pending and the corresponding wake request is asserted. In this expected method of run-time operation, any core can be the last to power down and any core can be the first to power up.

Implementing a power control system where a specific core must be the first to power up and the last to power down has the following potential consequences:

- Additional software stack complexity

  To allow this specific core to remain powered, when not required by the OS, requires platform-specific adaptations to power management software. This affects time-to-market, because of additional effort and error-prone modifications to standard software.

- Increased power consumption

  This specific core remains powered up when not required. Moreover, this can force a cluster to be powered up unnecessarily in a multi-cluster system.

- Increased latency

  The increased complexity in control sequences might increase the latency of powerdown and wakeup, and affect system responsiveness and power consumption.

3.3 Missing Event Bridge

The big.LITTLE system provides signals that allow for the communication of Send Event operations between clusters, as described in *Events* on page 2-8.

A potential mistake is to directly cross-connect the event signals between big and LITTLE. As a result, the events are lost. These events are single-cycle pulses and originate in different clock (and voltage) domains. Therefore, an Event Bridge is needed to ensure that the event is captured and retransmitted. This requires more than adding synchronizers.

To successfully implement an Event Bridge, you must consider the clock and power domains of the clusters. ARM recommends that you build a clock domain crossing circuit that handshakes each event between the clusters. It is acceptable for two or more events that are sent in close succession to be combined into a single event. The resulting event must be received after all of the events which are combined have been sent. Clusters that are powered off are not required to receive events.

3.4 Not waiting for correct signaling
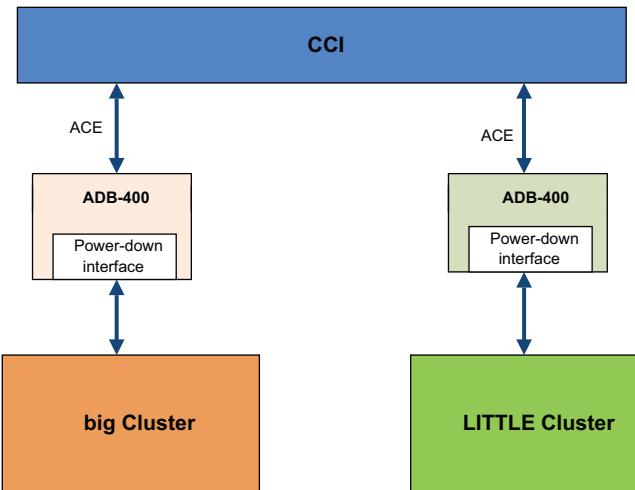
There might be a risk if you do not follow the required hardware sequencing requirements. A typical problem is the failure to wait for correct signalling during the power-down sequence.

A core must perform a sequence of steps, such as cleaning and invalidating cache, before powering down the core. After the steps are complete, a single core can be safely powered down only when the **STANDBYWFI** signal corresponding to that core is asserted.

A potential issue is that the power controller powers down a processor without waiting for correct signaling. For example, when software running on an application processor sends a power-down request to the power controller, the power controller acts on the request immediately without waiting for the assertion of **STANDBYWFI**.

3.5 AMBA® Domain Bridge power-down control

The following figure shows an example arrangement of Domain Bridges between CCI and clusters.


Figure 3-4 ADB in a big.LITTLE system

The cluster always powers up first and powers down last, following the rules described in *Power ordering* on page 2-6.

To avoid false transactions, the AMBA Domain Bridge (ADB) power-down interface must be managed as follows:

• The power-down interface must be in the *closed* state before either side powers down.

• The power-down interface can only be returned to the *open* state after both sides are powered on.

For detailed requirements, see the *ARM® CoreLink™ ADB-400 AMBA® Domain Bridge User Guide* (ARM DUI 0615E).

——— **Note** ———

Starting from the revision r3, the ADB power-down interface is replaced with a **Q-Channel**.

For details about the **Q-Channel**, see the *Low Power Interface Specification: ARM® Q-Channel and P-Channel Interfaces*(ARM IHI 0068).

Chapter 4
Hardware considerations

Read this chapter for advice from the hardware perspective when implementing power-down and power-up sequences for ARM processors.

It contains the following section:

4.1 Hardware constraints and advice

This application note applies to a typical SoC based on ARM Cortex-A processors. ARM recommends that you take the following hardware aspects into consideration in your SoC design:

Power domain implementation

Processor TRMs describe the supported power modes and power domains. See TRMs for more details.

In a real system, you must implement appropriate power domains to meet system requirements.

Power domain isolation

With multiple power domains implemented, you must implement clamps with proper functionality clamp value for the power domains. The clamps must be correctly activated and released, based on the power-down and power-up flow.

Level shifters

When crossing voltage domains, you must place level shifters between the two domains to manage the difference in voltage levels between the two supplies.

Power controller

A power controller is implemented, and it is in charge of the power management of the SoC. The power controller works closely with relevant signals of ARM IPs and other system components for power saving.

The components and signals include, but are not limited to, the following:

- Per-core signals:
  — STANDBYWFI.
  — **WARMRSTREQ**.
  — **SMPEN**.
  — **Q-Channel signals**.
- Cluster signals
  — **ACINACTM**.
  — **AINACTS**.
  — **L2FLUSHREQ**.
  — **L2FLUSHDONE**.
  — **L2RSTDISABLE**.
  — **STANDBYWFIL2**.
- ARM system IPs:
  CCI/NIC/DMC support Low Power Interface (LPI), based on the protocols below.
  — **Q-Channel**.
  — **P-Channel**.
  — **AXI LPI**.
- Wake request signals:
  For GICv2:
  — **nFIQOUT**
  — **nIRQOUT**

For GICv3:

— **WakeRequest**

You must consider all the guidelines related to the interrupt controller described in this application note in the system power controller implementation, regardless of the interrupt controller used in your system.

Common system components, such as the clock generator, reset generator, and power supplier, must be controlled by the power controller to provide required clock, reset, and power sources for relevant IPs properly, so that the ARM processor and relevant components can work properly in various system power modes. The processor power-down and power-up sequences, and specific processor *Dynamic voltage and frequency scaling* (DVFS) power-saving modes are typical application cases.

The power controller can be connected to some ARM debug signals for debug through power-down purposes. These debug signals include, but are not limited to, the following signals:

• **DBGRSTREQ**.

• **DBGNOPWRDWN**.

• **DBGPWRDUP**.

• **DBGPWRUPREQ**.

Glossary

Clock Domain A collection of design elements supplied with a common clock source. Other clock domains which interact with the domain might be synchronous, but with independent source activity control, or asynchronous.

Domain Bridge A component which passes a protocol transaction from one domain to another, this includes voltage, power, and clock domains or a combination.

Operating Performance Points The set of voltage and frequency combinations supported for DVFS of a voltage domain.

Power Domain A collection of design elements within a voltage domain that share common power control. A voltage domain can have one or more power domains.

Voltage Domain A collection of design elements supplied by a single voltage source. The voltage supply to a domain might be scaled or removed for power or performance reasons.

Index

T

Typical SoC example   2-2

V

Voltage domains partitioning   2-5
Voltage level translation   4-2

W

Wake request signals   3-2