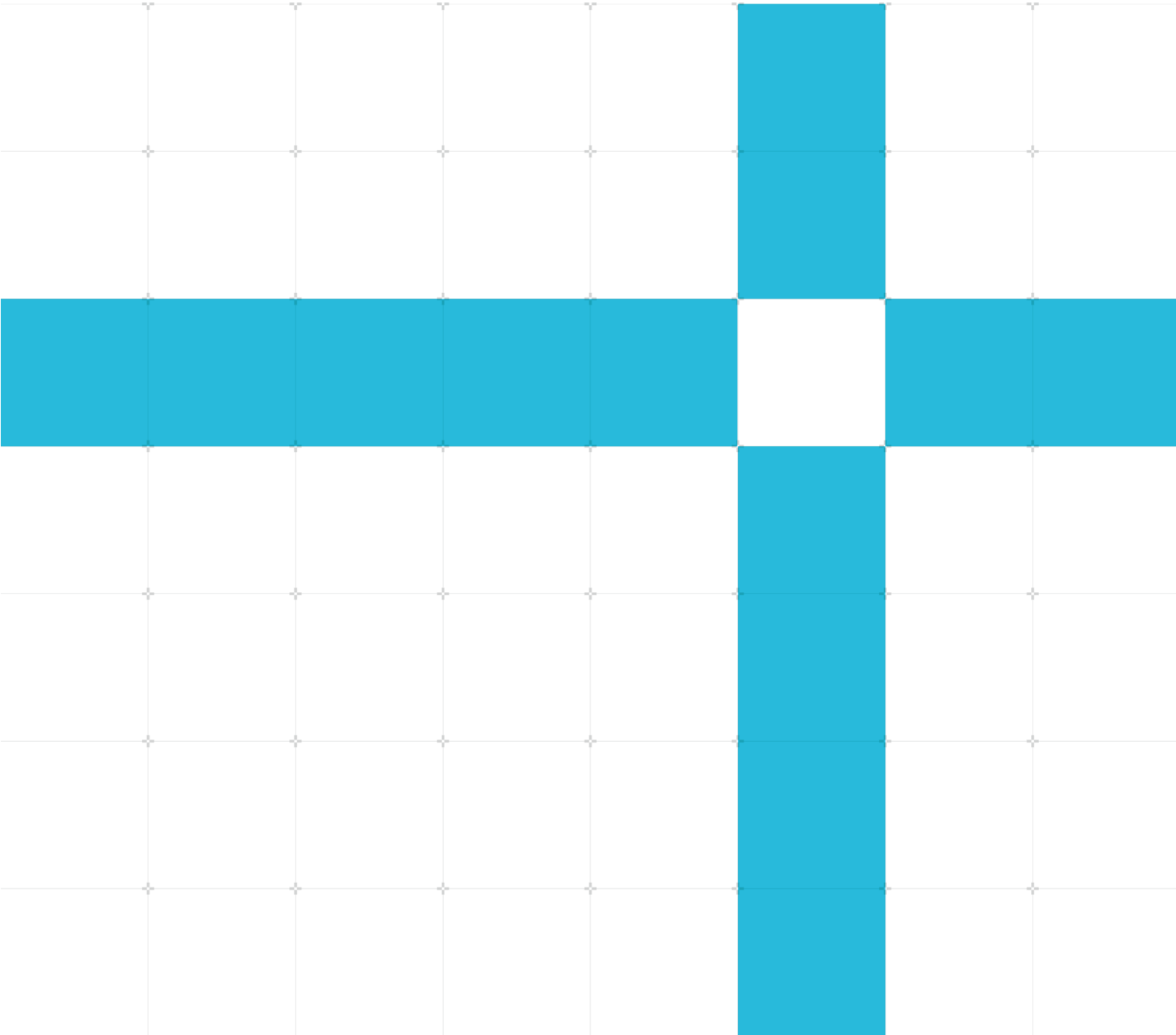
arm

Cortex-M

Maximize energy efficiency on SoC design for endpoint AI

Non-Confidential

Issue 1.0
102723



Cortex-M

Maximize energy efficiency on SoC design for endpoint AI

Release information

Document history

Issue	Date	Confidentiality	Change
01	December 8, 2021	Non-confidential	First version

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

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.

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

www.arm.com

Contents

1 Overview

As modern microcontrollers and SoCs become increasingly complex, designers face the challenge of maximizing energy efficiency while achieving a higher level of integration. To maximize energy efficiency, the use of multiple power domains is widely adopted in the low-power SoC market. At the same time, to address a higher level of integration, many SoCs contain multiple subsystems. Each subsystem can be reused for multiple projects, leading to better time to market and software reusability.

To address the challenges from the combination of these requirements, a standardized low power control interface and a common power control concept is needed in the system architecture.

In this guide, we demonstrate the deployment of these approaches in Arm Cortex-M-based subsystems. In addition, to reduce the software burden of working out power domain dependency, a mechanism called Power Dependency Control Matrix (PDCM) is introduced and its benefits are explained.

There is an increasing need for on-device processing closer to the data generated by Internet of Things (IoT) endpoints, while still operating within low power budgets. These energy efficient and highly compute-capable modern System on Chips (SoC) unlock new use cases called endpoint Artificial Intelligence (AI). Traditionally, a low-power microcontroller can support multiple levels of power states. Based on the sleep status of the processor, some of the functional design elements inside the microcontroller can have its clocks gated off or even powered down. To help system designs, the Arm Cortex-M processors architecturally support two levels of software-controlled sleep modes: sleep and deep sleep. The corresponding status is indicated by the SLEEPING and SLEEPDEEP status output signals:

Processor state	SLEEPING output	SLEEPDEEP output
Running	0	0
Sleeping	1	0
Deep sleep	1	1

Table 1: Processor sleep modes

Different power saving methods can be applied to the system in different sleep modes. For example, a simple microcontroller can use these sleep output signals to control the memory macros' power modes and to control each peripheral. A peripheral could be running (ON with toggling clock), clock gated (ON with static clock), in state retention (RET) or power down (OFF). To customize power control of each peripheral based on application requirements, the following programmable registers are added to the system to configure the power mode options of the peripherals:

Processor and system power state	Embedded flash	SRAM	Peripheral 1	Peripheral 2	Peripheral 3	Real Time Clock
Running	Running	Running	Configurable	Configurable	Configurable	Configurable
Sleeping	Clock gated	Clock gated	Configurable	Configurable	Configurable	Configurable

Processor and system power state	Embedded flash	SRAM	Peripheral 1	Peripheral 2	Peripheral 3	Real Time Clock
Deep sleep	Power down	State retention	Configurable	Configurable	Configurable	Configurable

Table 2: System power states

In modern SoC design, this technique is becoming inadequate under certain circumstances:

- There can be more than one processor in the system. There is no longer a single source of sleep mode indication.
- Memories and peripherals can also be accessed by bus transaction initiators other than the processor. For example, a Direct Memory Access (DMA) controller or external debugger. This access occurs whether the processor is sleeping or not.
- Many new SoCs are composed of multiple reusable subsystems. When these subsystems are designed, the details of the SoC power control scheme can be unknown. For example, the designers do not know how many levels of power states are available and whether the SoC fully utilizes all the power states of the subsystem.
- Controlling the system's overall power state and the power mode of each shared resource requires system-specific runtime software intervention. This intervention is time consuming and expensive in terms of processor runtime and resources. The runtime software might also impact the response to power mode changes due to interrupt latency and to handle conflicting processing tasks.

Consider a peripheral that is always on and receives data to its internal First In First Out (FIFO) buffer. The processor is powered down and the DMA and the system Static Random Access Memory (SRAM) are in retention. When the FIFO reaches the occupancy threshold, the DMA flushes it to SRAM. This requires both the SRAM and the DMA to wake, but the processor could remain powered down until a larger amount of data is collected in SRAM that requires processing. This arrangement still requires software intervention to wake the DMA and SRAM, which therefore requires the processor to wake up to handle this.

2 Power dependency

To address these challenges, instead of looking at the power control arrangement using a processor-centric view of sleep modes, we look at power control based on the power dependency relationship between different domains. Consider a design element that is a shared system resource that has its own power domain, like an interconnect or an SRAM. The power mode of this shared resource can be determined by the power mode of one or more other functional design elements that utilizes this resource. For example, if the processor or a DMA is in a functional power mode, the bus interconnect and the SRAM must be in a functional power mode. We expect that the processor and DMA need access to the system SRAM through the interconnect. Both the SRAM and interconnect power mode requirements are dependent on the processor or the DMA, as shown in the following diagram:

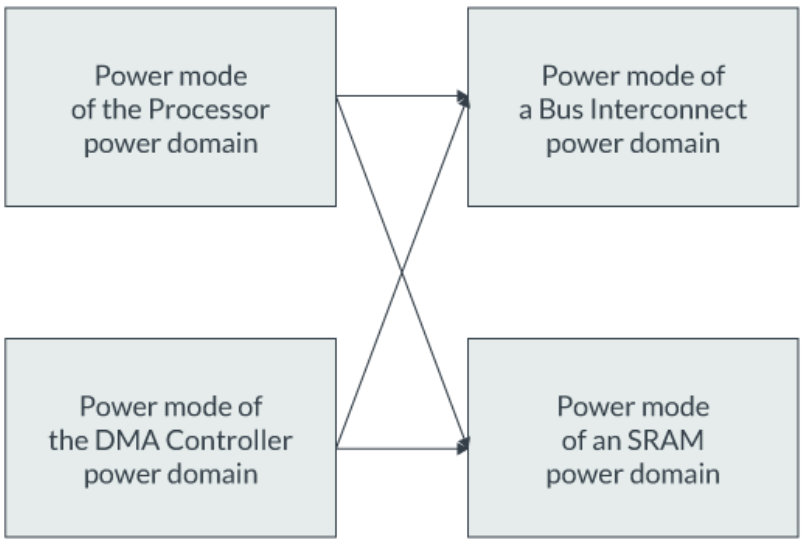


Figure 1: Example power dependency diagram

2.1 Power Dependency Control Matrix

A power dependency diagram can represent the relationship of a small number of power domains, however real systems can have many more power dependencies. These power dependencies are due to more shared resources, peripherals, processing elements, and power domains. Based on the simple dependency concept, we can represent the relationships between these domains using a simple two-dimensional table of dependency called a Power Dependency Control Matrix. The following table shows how we represent the previous power dependency diagram as a control matrix:

	Power mode of precedent power domains	
Power mode of dependent power domains (shared resources)	PD_CPU	PD_DMA
PD_INTERCONNECT	Y	Y

	Power mode of precedent power domains	
PD_SRAM	Y	Y

Table 3: Power Dependency Control Matrix example

The table rows show power domains that are being controlled. The table columns show controller power domains that can act as dependencies. A Y in the table indicates a dependency between a controller power domain and a controlled power domain. This dependency means that when the power mode of PD_CPU is ON, the minimum power mode of PD_INTERCONNECT is also ON. This arrangement does not necessarily guarantee that the PD_INTERCONNECT wakes up simultaneously with the PD_ CPU. This arrangement guarantees if both PD_INTERCONNECT and PD_CPU is ON, the dependency with the PD_CPU maintains the ON power mode of PD_INTERCONNECT. The PD_INTERCONNECT wake behavior depends on the system needs, and either wakes on detecting an access from the PD_CPU or PD_DMA, or when the PD_CPU is switched to ON.

To make the power control scalable and configurable for different use cases, some of the entries in the matrix are controlled by software programmable registers. By having this configurability, the power switching of power domains can be hardware autonomous in runtime after a boot time configuration, or even after a runtime reconfiguration for a long period of use. The following table shows an example of a more complex system where software configurability (cfg), is provided for some entries in the matrix:

	Power mode of precedent power domains			
Power mode of dependent power domains	PD_CPU0	PD_CPU1	PD_DMA0	PD_DMA1
PD_INTERCONNECT	Y	Y	Y	Y
PD_FLASH	Y	Y	cfg	cfg
PD_SRAM0	cfg	cfg	cfg	cfg
PD_SRAM1	cfg	cfg	cfg	cfg

Table 4: Configurable power dependency control matrix

For example, we can configure the PD_SRAM0 to be only dependent on PD_CPU0 and PD_DMA0, and PD_ SRAM1 to be only dependent on PD_CPU1 and PD_DMA1.

A given power domain can have multiple power modes presented as additional rows and columns in the matrix to provide finer grain dependency control between the power modes. In the following table, unavailable power modes are indicated by a dash (-):

		Precedent power domains							
		PD_CPU0		PD_CPU1		PD_DMA0		PD_DMA1	
Dependent power domains (shared resources)	Power mode	ON	RET	ON	RET	ON	RET	ON	RET
PD_INTERCONNECT	ON	Y	cfg	Y	cfg	Y	cfg	Y	-
	RET	cfg		cfg		cfg		-	
PD_FLASH	ON	Y	cfg	Y	cfg	cfg	-	cfg	-
	RET	-	-	-	-	-	-	-	-
PD_SRAM0	ON	Y	-	cfg	-	cfg	-	cfg	-
	RET	Y	Y	cfg	cfg	cfg	cfg	cfg	-
PD_SRAM1	ON	cfg	-	Y	-	cfg	-	cfg	-
	RET	cfg	cfg	Y	Y	cfg	cfg	cfg	-

Table 5: Power dependency control matrix with fine grain power mode control

The actual number of power modes of the dependent power domain shown in the table can be less than the total number of power modes supported. In addition, the implementation or other runtime conditions such as pending transactions can result in a higher selected power mode used. This mode differs from the power mode defined by the Power Dependency Control Matrix because the Power Dependency Control Matrix defines the minimum power mode requirements.

Some of the power domains in the system can have no internal power control interface to define a minimum power mode. These power domains only maintain a single power mode independently of other conditions typically in the OFF power mode. For example, a RAM macro has no dedicated power control interface. To overcome this limitation, software can define the minimum power mode by programming the Minimum Allowed Power Mode entry for the dependent power domains. This definition allows the power domain to maintain a minimum power mode when no other power domains are requesting the resource.

For example, in the following table PD_INTERCONNECT, PD_SRAM0, and PD_SRAM1 support a configurable minimum power mode that can be ON or RET to preserve content. Subsystem designs are required to be extended with further external functional design elements and interface with other subsystems. To fulfil these requirements, further precedent domains can be added to the Power Dependency Control Matrix as extension power domains. In the following table, PD_EXTn represents extension power domains:

Precedent power domains												
		PD_CPU0		PD_CPU1		PD_DMA0		PD_DMA1		PD_EXTn		Minimum Allowed Power Mode
Dependent power domains (shared resources)	Power mode	ON	RET	ON	RET	ON	RET	ON	RET	ON	RET	
PD_INTERCONNECT	ON	Y	cfg	Y	cfg	Y	cfg	Y	-	cfg	-	cfg: ON/RET/OFF
	RET	cfg	cfg	cfg	cfg	cfg	cfg	-	-	cfg	cfg	
PD_FLASH	ON	Y	cfg	Y	cfg	cfg	-	cfg	-	cfg	-	OFF
	RET	-	-	-	-	-	-	-	-	-	-	
PD_SRAM0	ON	Y	-	cfg	-	cfg	-	cfg	-	cfg	-	cfg: ON/RET/OFF
	RET	Y	Y	cfg	cfg	cfg	cfg	cfg	-	cfg	-	
PD_SRAM1	ON	cfg	-	Y	-	cfg	-	cfg	-	cfg	-	cfg: ON/RET/OFF
	RET	cfg	cfg	Y	Y	cfg	cfg	cfg	-	cfg	cfg	

Table 6: Power Dependency Control Matrix with minimum power mode

The Power Dependency Control Matrix concept can easily scale to accommodate multiple processors if there is no single definition of sleep mode. This concept can effectively cope with a scalable number of functional design elements and shared resources in the system and can be extended to span over multiple subsystems.

The control matrix also allows any connected power domain in the system to request other shared resources and combine these resource requests and manage the availability and context retention of shared resources according to the constraints.

3 Power Control Framework

The Power Dependency Control Matrix and the sensitivity settings defined for each power domain allow, as much as possible, system power control using hardware autonomous dynamic power transitions. These transitions reduce the software interactions needed for system management and therefore improves its responsiveness and power reduction.

These benefits of the Power Dependency Control Matrix require further infrastructure. The provision of the Power Dependency Control Matrix and power efficient components is insufficient. The functional design elements must participate in a coordinated system level power management infrastructure. It is also important that integration of functional design elements is achievable in a timely manner.

This is possible if there is a standardized way to produce, transfer, distribute, and apply power mode information across subsystems. This standardized framework is called the Power Control Framework.

The key parts of this framework include the following:

- Standard Low Power Interfaces
- Low Power Interface infrastructure elements
- Power domain controllers
- Access control bridge components
- Power Dependency Control Matrix component

4 Standard low power interfaces

The Arm Advanced Microcontroller Bus Architecture (AMBA) Low Power Interface (LPI) specification defines the interfaces that allow power mode information to traverse the system. The AMBA LPI specification is data agnostic, therefore it can transfer any type of information. In this guide, the AMBA LPI is used to represent power mode information. The details of the AMBA LPI are described in the AMBA Low Power Interface Specification Arm Q-Channel and P-Channel Interfaces.

For simple use cases where there are only two power modes like ON and OFF, the AMBA Q-Channel interface is sufficient. This interface allows the device to indicate to the power controller that the Quiescent state can be entered or exited using QACTIVE. The interface also allows the power controller to request the device to enter or leave Quiescent state. These features enable the power controller to prepare a power domain for entering or leaving a lower power mode. The interface is shown in the following diagram:

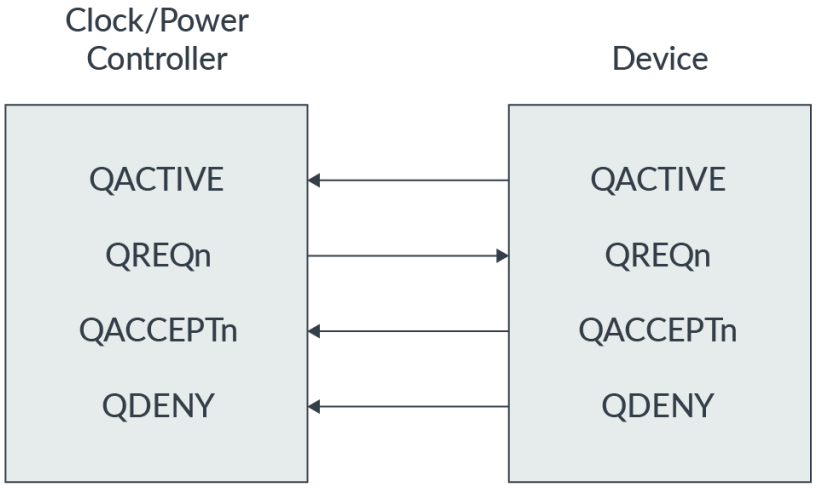


Figure 2: Q-Channel Low Power Interface

For complex cases where there are more than two power modes, AMBA P-Channel interface is required. This interface allows the device to indicate to the power controller the power mode the device needs using PACTIVE. The power controller requests the device to transition to a particular power mode indicted by PSTATE, as shown in the following diagram:

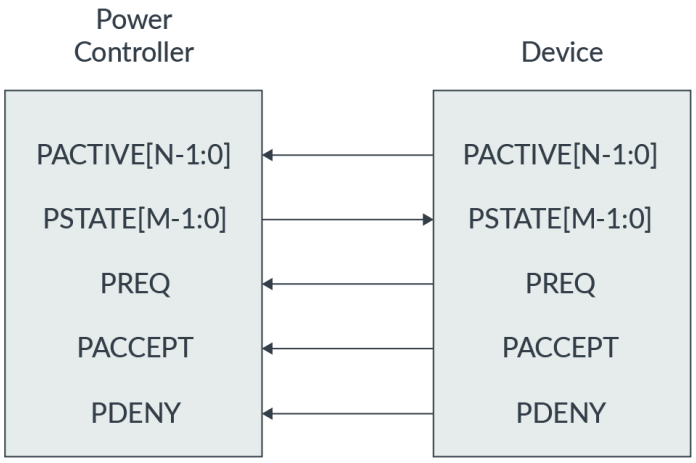


Figure 3: P-Channel Low Power Interface

In the power domain, each functional design element states the power mode requirements and constraints using the individual LPI interface. These requirements are then collated using a power LPI network. Every design element in the power domain is responsible for driving its own power mode constraints. For example, if a hardware accelerator is waiting for an event or processing data, its individual LPI interface uses the power LPI network to tell the power domain controller to stay ON.

Functional design elements without intrinsic support for LPI must implement an integration layer to provide this support. If a component power control interface does not directly match the AMBA LPI interfaces, an integration layer approach can also be used to convert the interfaces into LPI. The LPI integration layer is shown in the following diagram:

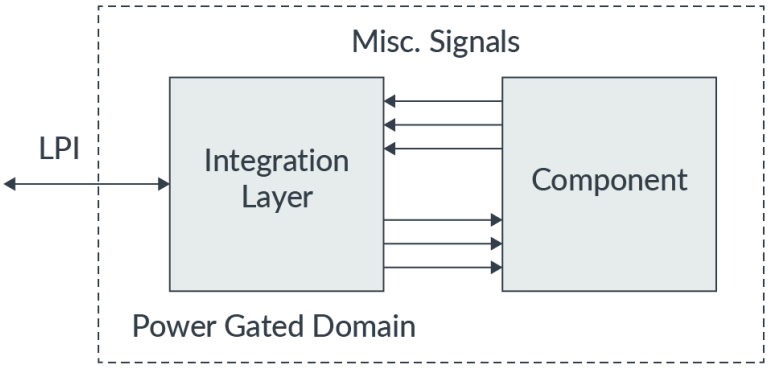


Figure 4: Low Power Interface integration layer

5 Low power interface infrastructure

The power LPI network gathers and merges power mode requirements from the power domain functional design elements. The power LPI network then gives the requirements to the power domain controller using a single power LPI. To help provide these requirements, Arm defined and implemented the following reusable standard LPI infrastructure components:

- Low Power Interface Distributor
- Low Power Interface Combiner
- Low Power Interface Converter, from P-Channel to Q-Channel

The Arm CoreLink Power Control Kit PCK-600 is a complete solution based on AMBA LPI standard. This section provides an overview of this kit. For more details, see the Arm CoreLink PCK-600 Power Control Kit Technical Reference Manual.

5.1 LPI distributor

The LPI distributor shown in the following diagram can split a single LPI into multiple LPIs and distribute the LPIs in parallel or in sequence to devices:

Figure 5: LPI distributor

The LPI distributor is typically used to expand the number of LPIs of the power controller to match the number of LPIs on all functional design elements. Sequencing can be used to manage race conditions in the controlled functional design elements.

When distributing in parallel, the request from the control LPI is sent to all device LPIs and all device LPIs must accept the request before the request on the control LPI is accepted. If any device LPI denies the request, all device LPIs are returned to the previous state and the originating request on the control LPI is denied.

When distributing in sequence, the request from the control LPI is sent to each device LPI and accepted in sequence. When all device LPIs have accepted the request, the request of the control LPI is accepted. If any device LPI denies the request, this process is reversed for all device LPIs that have previously accepted the request to return them to the earlier state before the originating control LPI request is denied.

Both Q-Channel and P-Channel versions of the distributors are defined.

5.2 LPI combiner

The LPI combiner shown in the following diagram merges the LPI requests from multiple power controllers to a single LPI:

Figure 6: LPI combiner

If any upstream power controllers on the control LPI request a lower power mode, the device LPI also requests a lower power mode. Also, if all upstream power controllers on the control LPI request a higher power mode, the device LPI then requests a higher power mode downstream. This request is typically used to manage the boundary of power domains that can operate independently.

5.3 LPI converter

The LPI converter shown in the following diagram can, for example, translate between the LPI protocols from AMBA P-Channel to AMBA Q-Channel:

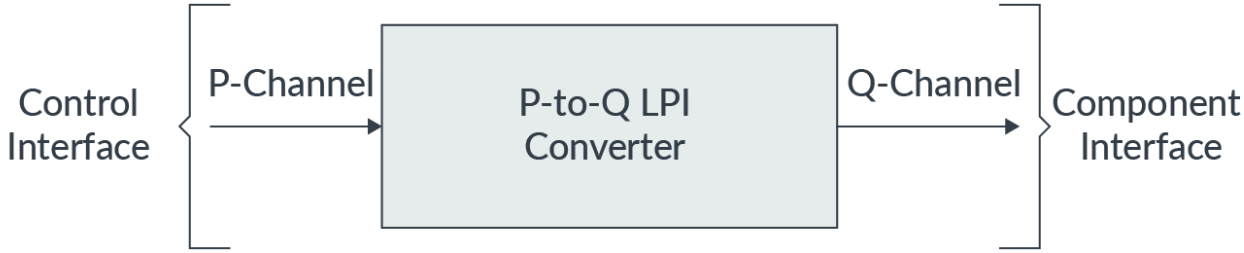


Figure 7: P-to-Q LPI converter

This unit compresses the multiple P-Channel states to two states based on the system integrator configuration. By utilizing these LPI infrastructure components, you can create structures to manage and sequence the state of each functional design element in the power domain so that they can enter a power mode safely. For example, in a power domain with three functional design elements, it may be necessary for one of the functional design elements to enter a specific individual power mode before allowing the other two functional design elements to specify a power mode to avoid a system

deadlock. An LPI Distributor working as a sequencer can be deployed to transition the critical functional design element into the requested power mode before transitioning the remaining functional design elements through another LPI Distributor in parallel.

6 Power domain controller

For power domain control, a component called the Power Policy Unit (PPU) is used. The PPU is a versatile technology-independent power infrastructure element that interfaces with functional design elements of a power domain, using the power LPI network for controlling domain power modes in coordination with device quiescence to ensure safe power mode transitions. It also performs the task of sequencing the controls of reset, isolation, clock, supply, and retention. This sequencing removes the need for software to perform low level control of these functionalities.

The PPU also introduces a level of autonomy in which the power domain can switch power modes without using software. This autonomous operation is important for power modes that require fast response times like logic or RAM retention states. No context is lost and the operation can be transparent to software, but state transitions must have minimal latency to preserve system performance. The PPU in dynamic operation can enter and exit allowed power modes based on the power mode requirements from the power LPI network without software intervention. If a dynamic operation is not required, the PPU can operate in static mode using software to request domains to enter a power mode.

For more details about the PPU, see the Arm Power Policy Unit Architecture Specification. The PPU structure is shown in the following diagram:

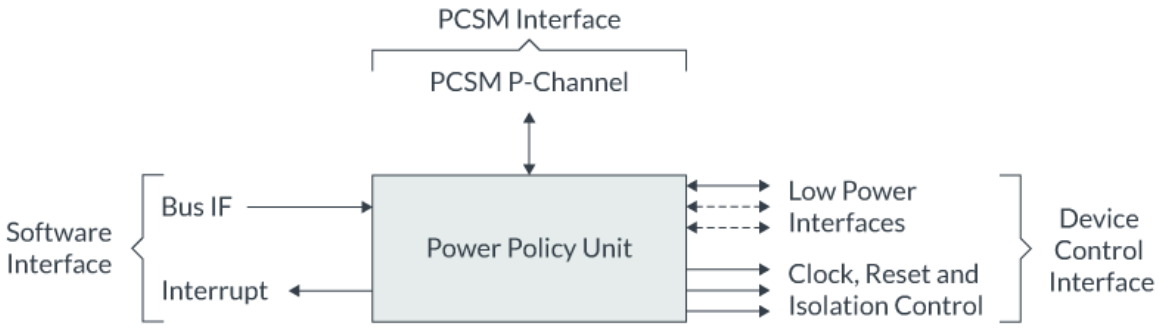


Figure 8: Power Policy Unit structure

The PPU has the following interfaces:

- Device control interface. This interface is for low level device control and to ensure device quiescence and functional control. This includes either a P-Channel or multiple Q-Channel LPI, and additional device controls that include clock enables, resets, and isolation control.
- Power Control State Machine (PCSM) interface. For controlling low-level technology-specific power switch and retention controls. The Power Control State Machine is a technology dependent state machine for the sequencing of power switch chains and retention controls that can include RAM and register retention. The PCSM executes power mode changes under PPU direction. The interface between the PPU and the PCSM is a P-Channel LPI.
- Software interface. Status information, configuration settings and static control.

The following shows a high-level diagram of how the PPU and PCSM connect to each other and control a power gated domain. The dotted lines around components and signal connections indicate these are implementation dependent. The PPU must be placed in a power domain that is always-on relative to the power domain that the PPU is controlling.

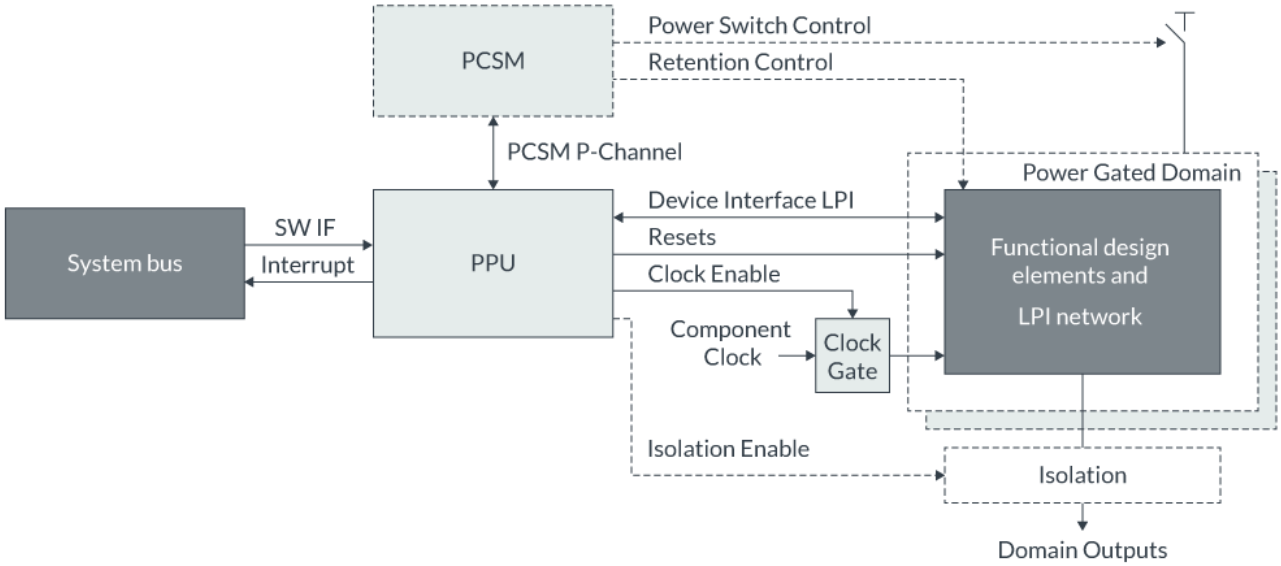


Figure 9: Power Policy Unit high level diagram

7 Access control bridge components

In some cases, you want to allow part of the system to be in a non-functional powered off mode or retention power mode, even though a bus transaction might arrive from another part of the system. This type of capability is referred to in this guide as access control.

Applications for access control include the following:

- To ensure a controlled power cycle for a resource, by ensuring all accesses to it are complete before entering non-functional power mode and providing responses during this period
- To preserve the availability of a resource while allowing it to enter a power saving mode. The resource is woken automatically when access is attempted.
- Preventing access to a resource from selected interfaces during post reset or runtime configuration

Access control support can be integrated into an initiator or into system components such as network interconnects and domain bridges. For example, the Arm CoreLink SIE-300 Access Control Gate, Sync-Down Bridge, and Sync-Up Bridge incorporates access control capabilities. For more information, see the Arm CoreLink SIE-300 AXI5 System IP for Embedded Technical Reference Manual.

The following diagram shows an example integration of the standalone Access Control Gate (ACG) on an interconnect path between two design elements:

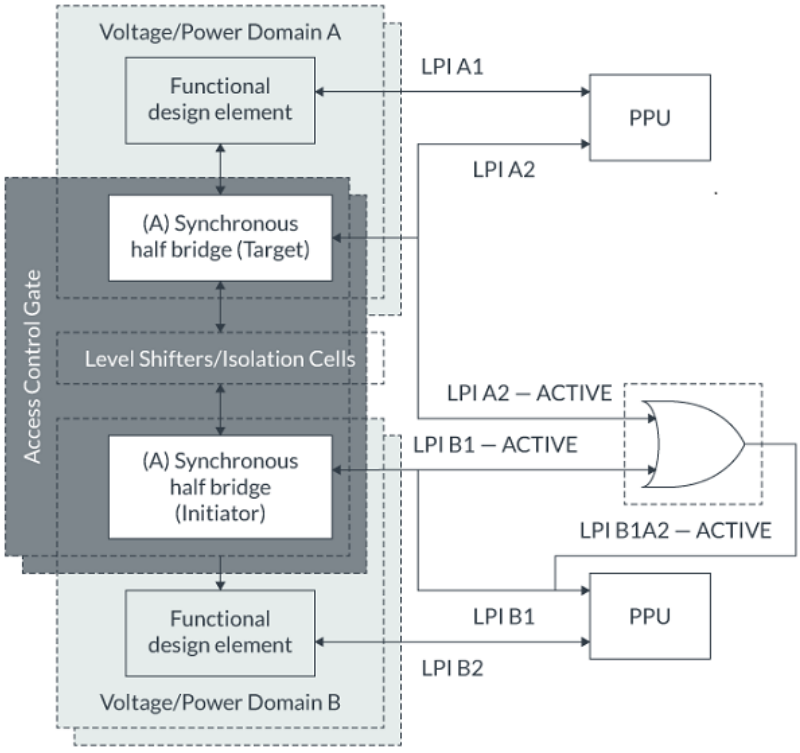


Figure 10: Access Control Gate between two domains

In this diagram, before Power Domain B enters a non-functional power mode, the PPU of Power Domain B requests that the Access Control Gate is closed through LPI B1. In preparation to close, all in-flight transactions must be completed. In-flight transactions can cause denial of the LPI request depending on the ACG configuration. When there are no more in-flight transactions and any new transactions are blocked by either stalling or terminating them with an error response, the ACG breaks the interconnect path between the two power domains. This break allows the initiator half bridge and any functional design element in Power Domain B to be placed into an inaccessible, quiescent state. Then, the LPI B1 accepts the request and indicates to PPU of Power Domain B that the power domain can enter a non-functional power mode that can save power.

When a transaction arrives at the ACG in Power Domain A when Power Domain B is in a non-functional power mode, the ACG can be configured to respond to any new transactions with an error. Alternatively, the transaction can be configured to stall access and send a hint using LPI A2 – ACTIVE to the PPU of Power Domain B to request Power Domain B to enter a functional power mode. When the PPU of Power Domain B receives this hint, the PPU then dynamically requests Power Domain B to enter a functional power mode. This hint is also called a wakeup request in this guide.

8 Power Dependency Control Matrix

The following diagram shows a high-level implementation of the Power Dependency Control Matrix (PDCM):

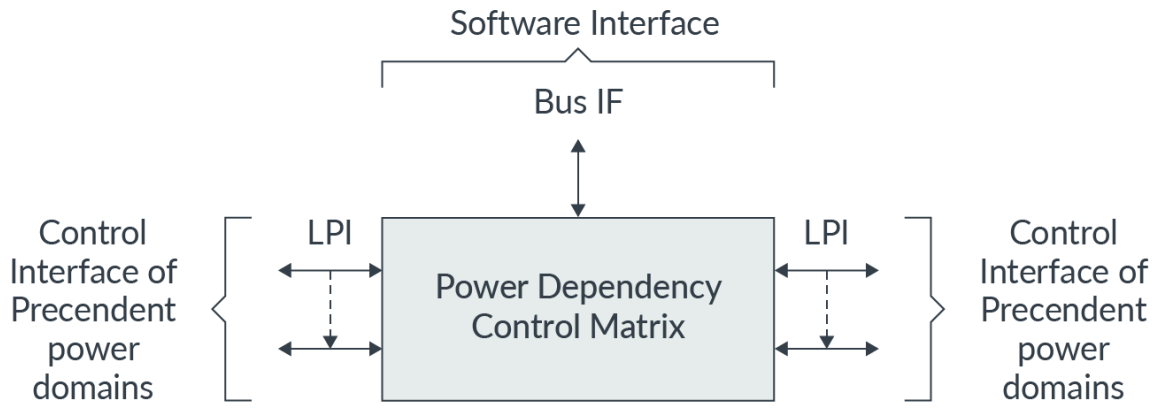


Figure 11: Power Dependency Control Matrix component

PDCM incorporates two sets of LPIs, one to interact with the PPUs of the dependent domains and one to interact with the PPUs of the precedent domains. The software interface provides access to memory mapped registers to configure the required power dependencies.

PDCM provides one-way dependencies, which means the precedent domains are not influenced by the dependent domains through PDCM. Therefore, the control LPI of precedent domains do not provide hints to the precedent PPUs and do not deny requests from the precedent PPUs. These LPIs follow the power modes of precedent power domains.

9 Putting it all together

The power control framework enables a system to use all the benefits of the Power Dependency Control Matrix. Using these components, you can compose a system with multiple power domains and hardware that can autonomously move between power modes, depending on the activity and the preconfigured dependencies.

The following diagram shows an example subsystem using power control with the Power Dependency Control Matrix:

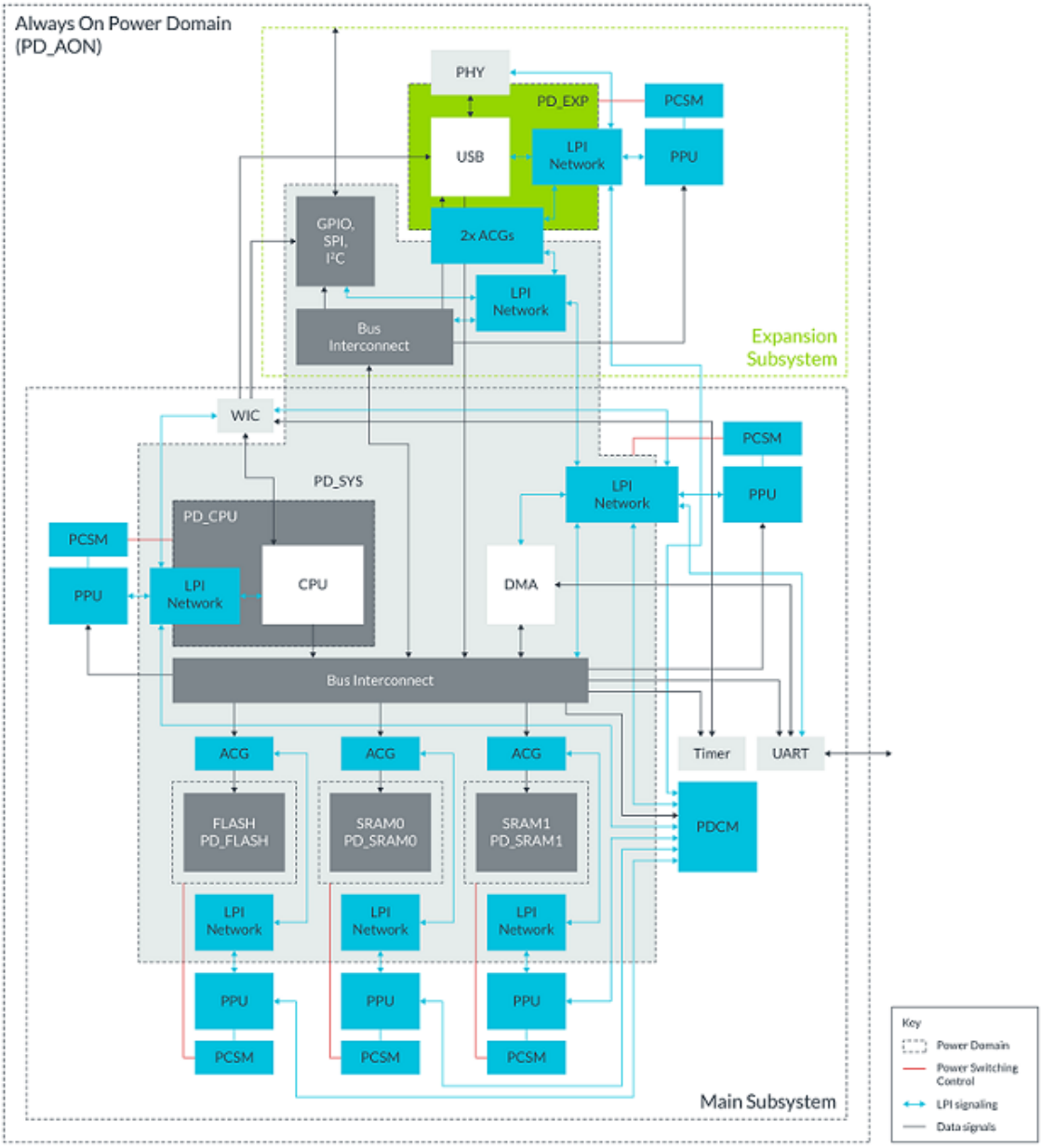


Figure 12: Subsystem example

This system consists of a prebuilt main subsystem that is reusable, and an additional expansion subsystem that adds functionality to form a complete system.

The expansion subsystem contains the following:

- PD_AON is the Always-On power domain and hosts all other power gated domains inside. There is a minimal amount of logic in it, such as functional design elements, that need to operate like Timer and PHY. The domain also contains the Wake-up Interrupt Controller (WIC) needed to wake PD_SYS, PD_CPU, the PPUs, and PCSMs to provide power control-related functionality.
- PD_SYS is the System Power Domain and PD_EXP is an independent Expansion Power Domain, containing most of the functional design elements
- PD_CPU, PD_FLASH, and PD_SRAM0/1 are additional power domains inside PD_SYS for the CPU and for each memory block
- ACGs are present across the system for access control. These ACGs are used in stall and wake configuration. On a transaction arrival, the PPU of the target power domain is requested to move the power domain to a functional power mode through LPI of the ACG.
- PPUs and PCSMs are provided to control each power domain
- LPI networks for each PPU are used to collate all local LPIs together to connect to the PPU
- PDCM is connected to all PPUs which helps to define the relationship between each power domain

Notice how the expansion subsystem uses its own LPI network to support multiple additional LPIs in the expansion subsystem using a single PD_SYS LPI expansion. ACGs then allow the USB controller to be independently power controlled while an LPI to the PDCM allows dependency-based power control between PD_EXP and the rest of the system. In the following table, if a domain is Y, the power mode of the dependent power domain is sensitive to the power mode of the precedent power domain:

		Precedent power domains						
		PD_SYS		PD_CPU		PD_EXP		Minimum Allowed Power Mode
Dependent power domains (shared resources)	Power mode	ON	RET	ON	RET	ON	RET	
PD_SYS	ON	-	-	Y	-	cfg	-	cfg: ON/RET/OFF
	RET	-	-	Y	Y	cfg	cfg	
PD_FLASH	ON	cfg	-	Y	-	cfg	-	OFF
PD_SRAM0	ON	cfg	-	Y	-	cfg	-	cfg: ON/RET/OFF
	RET	cfg	cfg	Y	Y	cfg	cfg	
PD_SRAM1	ON	cfg	-	cfg	-	cfg	-	cfg: ON/RET/OFF
	RET	cfg	cfg	cfg	cfg	cfg	cfg	

Table 7: Power dependency control matrix example

If a dependency is configured, the power mode of the precedent domain determines the minimum power mode for the dependent domain. The PDCM prevents the dependent power domain from entering a lower power mode. When a domain needs to maintain a certain power mode independently

of other conditions, software can control the intended minimum power mode by programming the Minimum Allowed Power Mode for the domain. This power mode can be ON, Retention, or OFF.

The PD_EXP is not a dependent power domain in the table because the USB controller in this domain can request its power mode to remain active independently once it is in functional power mode. For example, a register write through the ACG to the USB controller can be used to wake the PD_EXP power domain. This first register write sets a mode register inside the controller to cause the USB to request PD_EXP to be kept ON. From this point, the power mode of the USB controller remains ON until it has completed its task. Alternatively, a register write to the USB controller shuts it down, causing it to remove its request to keep PD_EXP ON. The power domain is then powered down.

10 Software and hardware interactions

This section is an example showing a setup of the Power Dependency Control Matrix. The example system acts as a USB device that responds to USB host requests periodically. In between the host requests the system is IDLE and includes a suspended USB link and the processor in Wait For Interrupt (WFI) sleep mode.

10.1 Software setup and partitioning

The following is required for this example:

- SRAM0 contains the stack and heap of the software running on the CPU
- SRAM1 contains the circular buffers of USB
- Flash contains the software that is executed in place

10.2 Power Dependency Control Matrix setup

After power on reset for first time boot, software needs to setup the PDCM. This is done using memory mapped register configuration of the PDCM. The software needs to configure PDCM to represent the power dependencies of the use case so that no data loss can occur.

In this example, the PDCM entries are set up in the following ways:

		Precedent power domains						
		PD_SYS		PD_CPU		PD_EXP		Minimum Allowed Power Mode
Dependent power domains (shared resources)	Power mode	ON	RET	ON	RET	ON	RET	
PD_SYS	ON	-	-	Y	-	cfg: N	-	cfg: OFF
	RET	-	-	Y	Y	cfg: Y	cfg: Y	
PD_FLASH	ON	cfg: N	-	Y	-	cfg: N	-	OFF
PD_SRAM0	ON	cfg: N	-	Y	-	cfg: N	-	cfg: OFF
	RET	cfg: N	cfg: N	Y	Y	cfg: N	cfg: N	
PD_SRAM1	ON	cfg: N	-	cfg: Y	-	cfg: N	-	cfg: OFF
	RET	cfg: N	cfg: N	cfg: Y	cfg: Y	cfg: Y	cfg: Y	

Table 8: Power dependency control matrix setup

10.3 Hardware and software event flow

In each use case, the system transitions through different system states. In this guide, only the important LPI transactions are represented for clarity.

The following describes the sequence of activities in the Power Control Framework elements corresponding to the transitions:

1. System idle state. When all outstanding operations are performed, the processor enters WFI deep sleep with WIC enabled. After the USB link times out, the link is suspended and the system becomes idle. During idle periods, only the USB PHY and the WIC are active.
2. System Power down state. The PD_CPU and PD_EXP power domains target the software-defined minimum power mode RET. As shown in the PDCM settings in Software hardware interactions, the PD_SYS, PD_SRAM0, and PD_SRAM1 target RET power mode. PD_FLASH targets OFF power mode.
3. Wake up PD_EXP. When the USB PHY detects a wake request, the PD_EXP moves to ON power mode by the corresponding PPU. This mode enables the USB controller to restore the link and process the request from the host. The dependent power domains of PD_EXP remain in RET power mode.
4. Wake up PD_SYS and PD_SRAM1. On the first bus access to SRAM1 by the USB controller, the ACGs located on the data path temporarily stall the access while the PD_SYS and the PD_SRAM1 power domains wake up in sequence. At this point, the PD_SYS, PD_EXP, and PD_SRAM1 are in

ON power mode. PD_SRAM0 is still in RET power mode and the PD_FLASH is still in OFF power mode.

5. Wake up PD_CPU. Once the USB descriptor is updated in SRAM1 by the USB controller, an IRQ is generated for the CPU to further process the descriptor. This IRQ through the Wakeup Interrupt Controller of the CPU wakes the PD_CPU and the corresponding PPU moves it to functional power mode. As part of the power mode change, the PDCM is also notified that the PD_CPU is now in ON state.
6. Wake up PD_SRAM0 and PD_FLASH. When the processor wakes up and resumes execution, typically a stacking operation occurs resulting in the RAM access. The ACG stalls the bus transaction until the PD_SRAM0 is moved to a functional power mode by its corresponding PPU.
    a. The same behavior occurs for flash access and PD_FLASH is moved to functional power mode by the corresponding PPU.
    b. At this point, the PD_SYS, PD_EXP, PD_SRAM1, PD_SRAM0, and PD_FLASH are in ON power mode. All the descriptor content can then be processed by the processor.

When the operation is finished, the system goes back to idle state.

The software and hardware event flow are shown in the following diagram:

Figure 13: Software and hardware event flow

11 Related information

The following resources are related to material in this guide:

- AMBA Low Power Interface Specification
- Arm CoreLink PCK-600 Power Control Kit
- Arm CoreLink SIE-300 AXI5 System IP for Embedded Technical Reference Manual
- Arm Corstone-300 Foundation IP
- Arm Power Policy Unit Architecture Specification
- Cortex-M55
- Ethos-U55 and Ethos-U65

12 Next steps

In this guide, you learned that power dependency usage relationships between different power domains enables a more scalable power control scheme than the processor-centric view of sleep modes. This approach addresses the challenges of system integration and expandability of modern SoCs targeting energy-efficient, intelligent IoT endpoints.

The power dependency-centric view with the framework offers hardware autonomous power transitions with very low latency and minimal software intervention, enabling opportunistic power saving. In some application scenarios, full software control could still be required. The introduction of the framework and the PPU allows Arm partners to take full software control over these mechanisms if needed using the memory mapped registers of the PPU.

As a next step, consider the following points when designing a system based on the principles described in this guide:

- The frequency of secure software calls during PDCM programming can have performance implications. A non-secure PDCM interface can be provided to reduce secure service calls.
- Software and hardware race conditions can occur when programming the PDCM. Software that configures the PDCM should consider these conditions.
- When re-programming the PDCM to switch between use cases that remove a particular dependency, care must be taken not to corrupt an actively used dependency.