Application Note AN472

Integrating GICv2 Interrupt Controllers with ARM® Cortex®-A5x and Cortex®-A72 processors

Non-Confidential

Application Note on Integrating GICv2 Interrupt Controllers with ARM® Cortex®-A5x and Cortex®-A72 processors

Release Information

The following changes have been made to this Application Note.

Change History

Date	Issue	Confidentiality	Change
13 March 2015	A	Non-confidential	First release

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

http://www.arm.com

Contents

Integrating GICv2 Interrupt Controllers with ARM® Cortex®-A5x and Cortex®-A72 processors

1 Conventions and Feedback

The following describes the typographical conventions and how to give feedback:

Typographical conventions

The following typographical conventions are used:

`monospace` denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.

`mono`space denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.

`monospace italic`
denotes arguments to commands and functions where the argument is to be replaced by a specific value.

`monospace bold`
denotes language keywords when used outside example code.

italic highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold highlights interface elements, such as menu names. Also used for emphasis in descriptive lists, where appropriate, and for ARM® processor signal names.

Feedback on this product

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

- Your name and company.
- The serial number of the product.
- Details of the release you are using.
- Details of the platform you are using, such as the hardware platform, operating system type and version.
- A small standalone sample of code that reproduces the problem.
- A clear explanation of what you expected to happen, and what actually happened.
- The commands you used, including any command-line options.
- Sample output illustrating the problem.
- The version string of the tools, including the version number and build numbers.

Feedback on documentation

If you have comments on the documentation, e-mail `errata@arm.com`. Give:

- The title.
- The number, DAI0472A.
- If viewing online, the topic names to which your comments apply.
- If viewing a PDF version of a document, the page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

ARM periodically provides updates and corrections to its documentation on the ARM Information Center, together with knowledge articles and *Frequently Asked Questions* (FAQs).

Other information

- ARM Information Center, http://infocenter.arm.com/help/index.jsp.
- ARM Technical Support Knowledge Articles, http://infocenter.arm.com/help/topic/com.arm.doc.faqs/index.html.
- ARM Support and Maintenance, http://www.arm.com/support/services/support-maintenance.php.
- ARM Glossary, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

2 Preface

This Application Note is intended for system designers who want to integrate GICv2 Interrupt Controllers with ARM® Cortex®-A5x or Cortex-A72 processors. Throughout this document, “Cortex-A5x” refers to the Cortex-A53 and Cortex-A57 processors.

These topics support the following chapters:

- *References* on page 3.

2.1 References

ARM® Cortex®-A53 MPCore Processor Technical Reference Manual (ARM DDI 0501E)

ARM® Cortex®-A53 MPCore Processor Integration Manual (ARM DIT 0036)

ARM® Cortex®-A57 MPCore Processor Technical Reference Manual (ARM DDI 0488G)

ARM® Cortex®-A57 MPCore Processor Integration Manual (ARM DII 0280F)

ARM® Cortex®-A72 MPCore Processor Technical Reference Manual (ARM 100095_0001_02_en)

ARM® Cortex®-A72 MPCore Processor Integration Manual (ARM 100096_0001_02_en)

ARM® CoreLink® GIC-400 Technical Reference Manual (ARM DDI 0471B)

ARM® Generic Interrupt Controller Architecture Specification (ARM IHI 0048)

3 Introduction

The ARM® Cortex®-A5x MPCore and Cortex-A72 MPCore processors do not include an integrated interrupt controller unlike some previous ARM processors such as the Cortex-A9 MPCore. These cores implement GIC CPU interfaces to connect to an external interrupt distributor component through an AXI4 Stream interface. Such interrupt distributor components are based on the ARM Generic Interrupt Controller Architecture Specification version 3.0 (GICv3).

The GICv3 Architecture Specification addresses some of the intrinsic limitations of the older GIC Architecture Specification version 2.0 (GICv2), such as limited scalability as system size increases. It also adds several new features. For example, the GICv2 Architecture Specification restricts the number of processors supported to a maximum of 8, whereas interrupt controllers based on the GICv3 Architecture Specification can support up to 128 cores.

Cortex-A5x and Cortex-A72 systems with up to 8 cores can be configured in a legacy mode to support external interrupt controllers implementing the GICv2 Architecture Specification. In this functional mode the GICv3 functionality and interfaces are disabled.

This document describes how to integrate Cortex-A5x or Cortex-A72 processors with an interrupt controller based on the GICv2 Architecture Specification, such as the ARM CoreLink® GIC-400.

4 Cortex-A5x/A72 processors and GIC Overview

New ARM® Generic Interrupt Controller Architecture (GICv3/v4)

Compared to GICv2, the GICv3 architecture adds new interrupt types and handling for ARMv8 processors, and has support for more than 8 cores using a single GIC. Cortex®-A5x and Cortex-A72 processors implement the GIC CPU interface, as described by the GICv3/v4 architecture, for interfacing with an external GICv3/v4 distributor component such as the GIC-500.

Interrupt controllers based on the GICv3/v4 architecture support the AXI4 Stream protocol interface, interrupt virtualization extensions, interrupt grouping and message-based interrupts, as well as support for new system exceptions.

GIC functionality is split between the processor and the external GIC v3/v4:

- GIC CPU interface is implemented within the processor.
- Distributor functionality is in an external GIC.
- Communication between the two takes place through a dedicated AXI4 stream protocol interface.

New exception handling model in ARMv8:

- Regardless of the GIC architecture version.

ARMv8 exceptions are split between:

- Synchronous.
- Asynchronous: IRQ, FIQ, and SError (System Error).

Synchronous exceptions include:

- Data or instruction aborts from MMU, for example, permission failure or alignment checking.
- SP and PC alignment checking.
- Unallocated (undefined) instructions.
- SVCs, SMCs, and HVCs.
- Synchronous external aborts, for example, abort when reading a translation table.
- Debug exceptions.

SError exceptions:

- Asynchronous data aborts, for example, an abort triggered by write-back of dirty cache line.

When the core takes an exception, the EL can stay the same or go higher:

- Configured in EL control registers.

5 GIC-400 Overview

Compliant with ARM® Generic Interrupt Controller architecture version 2 (GICv2):

- Distributes interrupts to target cores.
- Handles masking, prioritization, status tracking, and software generation of interrupts.
- Implements the GIC Security and the GIC Virtualization Extensions.

GIC-400 Interrupt Types:

- **Shared Peripheral Interrupts (SPI)**:
    - RTL configuration option: 0-480 different SPIs (in steps of 32).
    - Programmable to be edge-triggered or active-HIGH level-sensitive.
    - Can be targeted to any core in one or more clusters driven by the GIC.
- **Private Peripheral Interrupts (PPI):**
    - Generated for one specific core.
    - Timer event outputs must be connected to GIC to enable those PPI.
    - nCNT* signals driven by (generic) timer events.
- **Software Generated Interrupts (SGI):**
    - Generated by writing to the Software Generated Interrupt Register (SGIR).
    - 16 SGIs per core.

Interrupt Identification

Interrupt sources are identified using specific ID numbers.

For GIC-400:

- ID0-ID15 (SGI) Software Generated Interrupts.
- ID25 (PPI-6) Virtual Maintenance Interrupt.
- ID26 (PPI-5) Non-secure EL2 Physical Timer Event.
- ID27 (PPI-4) Virtual Timer Event.
- ID28 (PPI-0) Legacy nFIQ input.
- ID29 (PPI-1) Secure EL1 Physical Timer Event.
- ID30 (PPI-2) Non-secure EL1 Physical Timer Event.
- ID31 (PPI-3) Legacy nIRQ input.
- ID32-ID511 (SPI) IRQS[0] through IRQS[479].

Special IDs:

- ID1022 Non-secure interrupt pending when in secure state.
- ID1023 Spurious Interrupt.

6 Cortex-A5x/A72 and GIC Architecture v2 Support

The GIC CPU interface is always present in Cortex-A5x and Cortex-A72 processors. However, it can be disabled at reset to allow integration with external GICv2-based distributor components. The GIC CPU interface must be disabled when the processor is not integrated with a GICv3/v4 distributor component.

Cortex-A5x/A72 processors support the GICv2 architecture for up to 8 cores:

- Can be integrated with the GIC-400.
- The GIC-400 should be connected to the system bus-matrix as a slave to the master interface of the processor.
- Looks similar to the interrupt interface in older Cortex-A processors.
- The GIC-400 supports Cortex-A5x/A72 systems with up to 8 cores.

Cortex-A5x/A72 processor GIC CPU interfaces must be disabled:

- GICCDISABLE input tied HIGH.
- Removes access to the memory-mapped and system GIC CPU interface registers.
- nVIRQ/nVFIQ can be driven by the external GIC.
- The GIC-400 connects to the Cortex-A5x/A72 processor in the same way as a Cortex-A15 processor or Cortex-A7 processor.
- Interrupts are signaled through nIRQ[n:0], nVIRQ[n:0], nFIQ[n:0], nVFIQ[n:0].
- The Technical Reference Manual (TRM) of the processor provides more details.
- AXI4 Stream protocol interface tied off.

Tying off the AXI4 Stream Protocol Interface when GICCDISABLE is HIGH:

- When GICCDISABLE is tied HIGH, the following input signals must be tied LOW:

  ICDTVALID, ICDTDATA, ICDTLAST, ICDTDEST and ICCTREADY.
- When GICCDISABLE is tied HIGH, leave following output signals unconnected:

  ICCTVALID, ICCTDATA, ICCTLAST, ICCTID, ICDTREADY and nVCPUMNTIRQ[N:0].
- When GICCDISABLE is tied HIGH, the nVIRQ and nVFIQ inputs are driven by the external GIC. These can also be tied HIGH if not in use.

nSEI[n:0] and nREI[n:0] are not related to GICs:

- The GIC-400 can only signal IRQ and FIQ exceptions.

With the GIC CPU interface disabled, the processor still supports the ARM Architecture v8 exception behavior. The nSEI, nREI, and nVSEI signals remain functional and can be used to cause their respective exceptions. However, the GIC-400 does not support these signals as they are related to ARMv8 exceptions, and so will not be aware of their existence. During a power-down sequence, the processor must be able to mask all external interrupts, including SErrors generated by nSEI, nREI and nVSEI. If there is no SoC-logic outside of the Cortex-A5x/A72 processor that needs to report these system errors, it is recommended that these signals are tied HIGH.

Memory Error Signals

It is recommended that memory error signals from the processor (nINTERRIRQ, nEXTERRIRQ) and equivalent signals from other IP are routed through the GIC. The resulting interrupts should be delivered as FIQs (secure) to the processor.

Power Management and Wakeup Events

In the GIC-400, wakeup events are signaled using nIRQOUT and nFIQOUT. nFIQOUT and nIRQOUT should be connected to the power controller. These signals indicate if the GIC is trying to signal an interrupt on the corresponding core. The role of the power controller is to wake up the core from a power-down state when the GIC asserts these signals. Unlike the normal interrupt outputs, nIRQOUT and nFIQOUT are not masked by disabling the CPU interface in the GIC-400.

GIC-400 Clocking

The GIC-400 has a single common clock with which all the AXI and other interfaces are synchronized. PPI and SPI inputs from the cores and peripheral devices should be synchronized with the GIC-400 clock before fed into the GIC-400.

An asynchronous bridge should be used for the AXI interface of the GIC-400.

GIC-400 Programming

The GIC-400 Technical Reference Manual and GIC Architecture Specification version 2.0 should be the starting point for programming information.

AxUSER Requirements

In a multi-cluster system, each cluster will output a unique transaction ID on AxIDM to identify the type of transaction. For a read and write transaction, AxIDM[1:0] indicates the processor that originated the transaction.

Once inside the interconnect, the interconnect will identify the originating cluster by post-pending a few bits to the AxIDM. At this point, the transaction is uniquely identified as having come from a particular core.

Unfortunately, the GIC-400 does not use the AxID bits in this way because it expects the information to be presented on AxUSER. Therefore, it requires the system integrator to use their knowledge of the system (including, for example, its AxID bits) to generate the AxUSER signals.

Memory Map

- The input PERIPHBASE[43:18] in the Configuration Base Address Register (CBAR_EL1) of a Cortex-A57/A72 processor sets the location of the Interrupt Controller registers in the memory map. For Cortex-A53 processors, the corresponding input is PERIPHBASE[39:18].
- Each core sees its own banked copy of the CPU interface registers.
- The use of PERIPHBASE is optional with GICCDISABLE tied HIGH. It is simply a way for software to know the address of the external GIC in the system.

Table 6-1 summarizes the GIC memory map:

Table 6-1 GIC Memory Map

Offset from PERIPHBASE	Interrupt Controller Block
0x0000 – 0x0FFF	Reserved
0x1000 – 0x1FFF	IC Distributor
0x2000 – 0x3FFF*	IC Physical CPU interface
0x4000 – 0x4FFF*	IC Virtual CPU interface (Hypervisor view for requesting CPU)
0x5000 – 0x5FFF*	IC Virtual CPU interface (Hypervisor view for all cores)
0x6000 – 0x7FFF*	IC Virtual CPU interface (Virtual Machine view)[1]

[1] Each core sees its own banked copy of the CPU interface registers

7 Example: Integrating a GIC-400 with Cortex-A53 processors

This section describes an example where two Cortex-A53 clusters with 4 cores are integrated with a GIC-400. The legacy FIQ/IRQ signals in the GIC-400 are used as PPIs. Figure 7-1 gives a block-diagram overview of the system. Table 7-1and Table 7-2 list the interrupt-related connections on the Cortex-A53 side and the GIC-400 side, respectively. The GIC-400 connects to the AXI interconnect through a 32-bit AXI4 slave interface. For information on the AXI interconnect, refer to the AMBA AXI and ACE Protocol Specification.

In this example, to generate the AxUSER bits used by the GIC-400 to identify which core is the source of a transfer, one solution could be connecting AxIDM[1:0] directly from the cluster outputs to AxUSER[1:0] of the GIC-400, and connecting one of the ID bits appended by the interconnect to AxUSER[2] input to identify which cluster is performing the access.

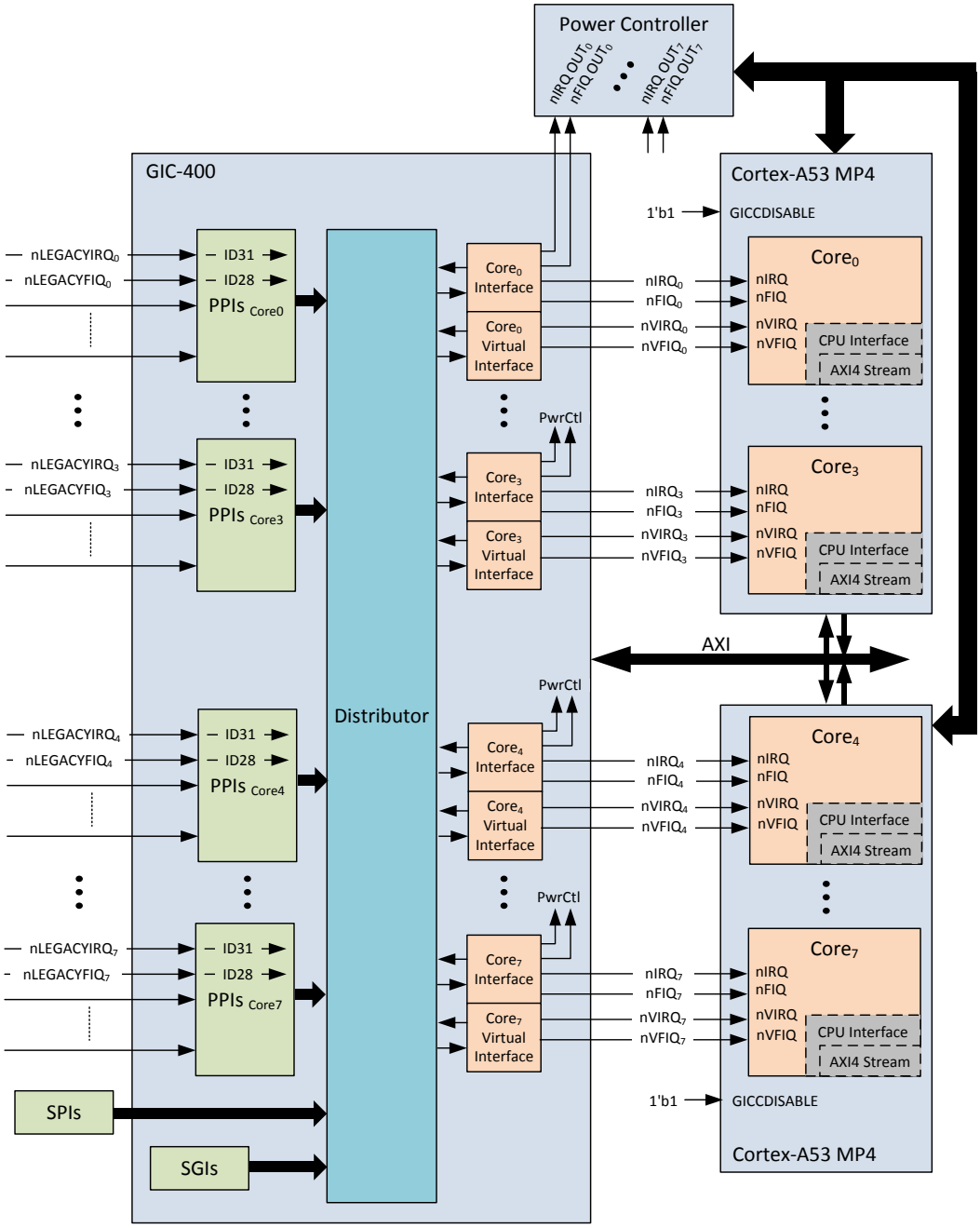

Figure 7-1 Main connections between two Cortex-A53 clusters and a GIC-400

Table 7-1 summarizes the main connections for the Cortex-A53 processor:

Table 7-1 Cortex-A53 Processor Main Connections

Signals	Direction	Size	Connect to Unit	Connect to Signal
GICCDISABLE	Input		constant	1'b1
nFIQ[N-1:0]	Input	[N-1:0]	GIC-400	nFIQCPU[N-1:0]
nIRQ[N-1:0]	Input	[N-1:0]	GIC-400	nIRQCPU[N-1:0]
nVFIQ[N-1:0]	Input	[N-1:0]	GIC-400	nVFIQCPU[N-1:0]
nVIRQ[N-1:0]	Input	[N-1:0]	GIC-400	nVIRQCPU[N-1:0]
ICDTVALID	Input		constant	1'b0
ICDTREADY	Output		unconnected	
ICDTDATA[15:0]	Input	[15:0]	constant	1'b0
ICDTLAST	Input		constant	1'b0
ICDTDEST[1:0]	Input	[1:0]	constant	1'b0
ICCTVALID	Output		unconnected	
ICCTREADY	Input		constant	1'b0
ICCTDATA[15:0]	Output	[15:0]	unconnected	
ICCTLAST	Output		unconnected	
ICCTID[1:0]	Output	[1:0]	unconnected	
nVCPUMNTIRQ[N-1:0]	Output	[N-1:0]	unconnected	

Table 7-2 summarizes the main connections for the GIC-400:

Table 7-2 GIC-400 Main Connections

Signal	Direction	Size	Connect to Unit	Connect to Signal
IRQS[NUM_SPIS-1:0]	Input	[NUM_SPIS-1:0]	GIC-400 SPIs	SPIs
nLEGACYIRQ[N-1:0]	Input	[N-1:0]	GIC-400 PPIs	PPI ID31 (if used)
nCNTPNSIRQ[N-1:0]	Input	[N-1:0]	GIC-400 PPIs	PPI ID30
nCNTPSIRQ[N-1:0]	Input	[N-1:0]	GIC-400 PPIs	PPI ID29
nLEGACYFIQ[N-1:0]	Input	[N-1:0]	GIC-400 PPIs	PPI ID28 (if used)
nCNTVIRQ[N-1:0]	Input	[N-1:0]	GIC-400 PPIs	PPI ID27
nCNTHPIRQ[N-1:0]	Input	[N-1:0]	GIC-400 PPIs	PPI ID26
nFIQCPU[N-1:0]	Output	[N-1:0]	Cortex-A53	nFIQ[N-1:0]
nIRQCPU[N-1:0]	Output	[N-1:0]	Cortex-A53	nIRQ[N-1:0]
nVFIQCPU[N-1:0]	Output	[N-1:0]	Cortex-A53	nVFIQ[N-1:0]
nVIRQCPU[N-1:0]	Output	[N-1:0]	Cortex-A53	nVIRQ[N-1:0]
nFIQOUT[N-1:0]	Output	[N-1:0]	Cortex-A53	System Power Controller
nIRQOUT[N-1:0]	Output	[N-1:0]	Cortex-A53	System Power Controller

8 GIC-400 Legacy Interrupt Signals

The GIC-400 supports a legacy mode that enables a mode of operations analogous to how older ARM® processors handled interrupts. Older processors used two physical interrupt input lines, traditionally referred to as nIRQ and nFIQ. The nLEGACYIRQ and nLEGACYFIQ inputs to the GIC-400 are analogous to the nIRQ/nFIQ inputs of the Cortex-A15 processor or the Cortex-A7 processor.

As PPIs for the corresponding core:

- $nLEGACYIRQ_{[NUM\_CPUS–1:0]}$ connected to $PPI_{[NUM\_CPUS–1:0]}$ with interrupt ID 31.
- $nLEGACYFIQ_{[NUM\_CPUS–1:0]}$ connected to $PPI_{[NUM\_CPUS–1:0]}$ with interrupt ID 28.

As legacy IRQ or FIQ for the corresponding core, bypassing the GIC logic:

- $nLEGACYIRQ_{[NUM\_CPUS–1:0]}$ connected to $nIRQCPU_{[NUM\_CPUS–1:0]}$ (Legacy IRQ signal).
- $nLEGACYFIQ_{[NUM\_CPUS–1:0]}$ connected to $nFIQCPU_{[NUM\_CPUS–1:0]}$ (Legacy FIQ signal).

To use the legacy mode, two conditions must be true:

- The corresponding GIC-400 CPU interface must be disabled.
- The GIC-400 bypass function must be active.

The GIC-400 bypass functionality is enabled by default. When a CPU interface in the GIC-400 is disabled, the bypass functionality allows the nLEGACYIRQ and nLEGACYFIQ signals to bypass the GIC logic and drive their corresponding nIRQCPU or nFIQCPU signals.

The bypass functionality itself can be disabled in the GIC-400 through the CPU Interface Control Register (GICC_CTLR). This ensures that when a CPU interface in the GIC-400 is disabled, the legacy interrupt signals cannot bypass the GIC logic and wake up the core. Bypassing is typically disabled when powering down a processor, to avoid unwanted core wakeups from legacy interrupt inputs.

If the bypass functionality is disabled:

- nIRQCPU[NUM_CPUS–1:0] is de-asserted.
- nFIQCPU[NUM_CPUS–1:0] is de-asserted.

As has been previously mentioned, the GIC-400 supports wakeup events in systems that require power management. It signals these wakeup events using nIRQOUT or nFIQOUT. Disabling the GIC-400 bypass functionality does not affect nIRQOUT or nFIQOUT, and as a result, they are always enabled.

Figure 8-1 shows a block diagram of the same system as in Figure 7-1, but with the GIC-400 working in legacy (bypass) mode. The nLEGACYIRQ and nLEGACYFIQ inputs bypass the distributor logic in the GIC-400. Power controller outputs (nIRQOUT/nFIQOUT) are not shown.

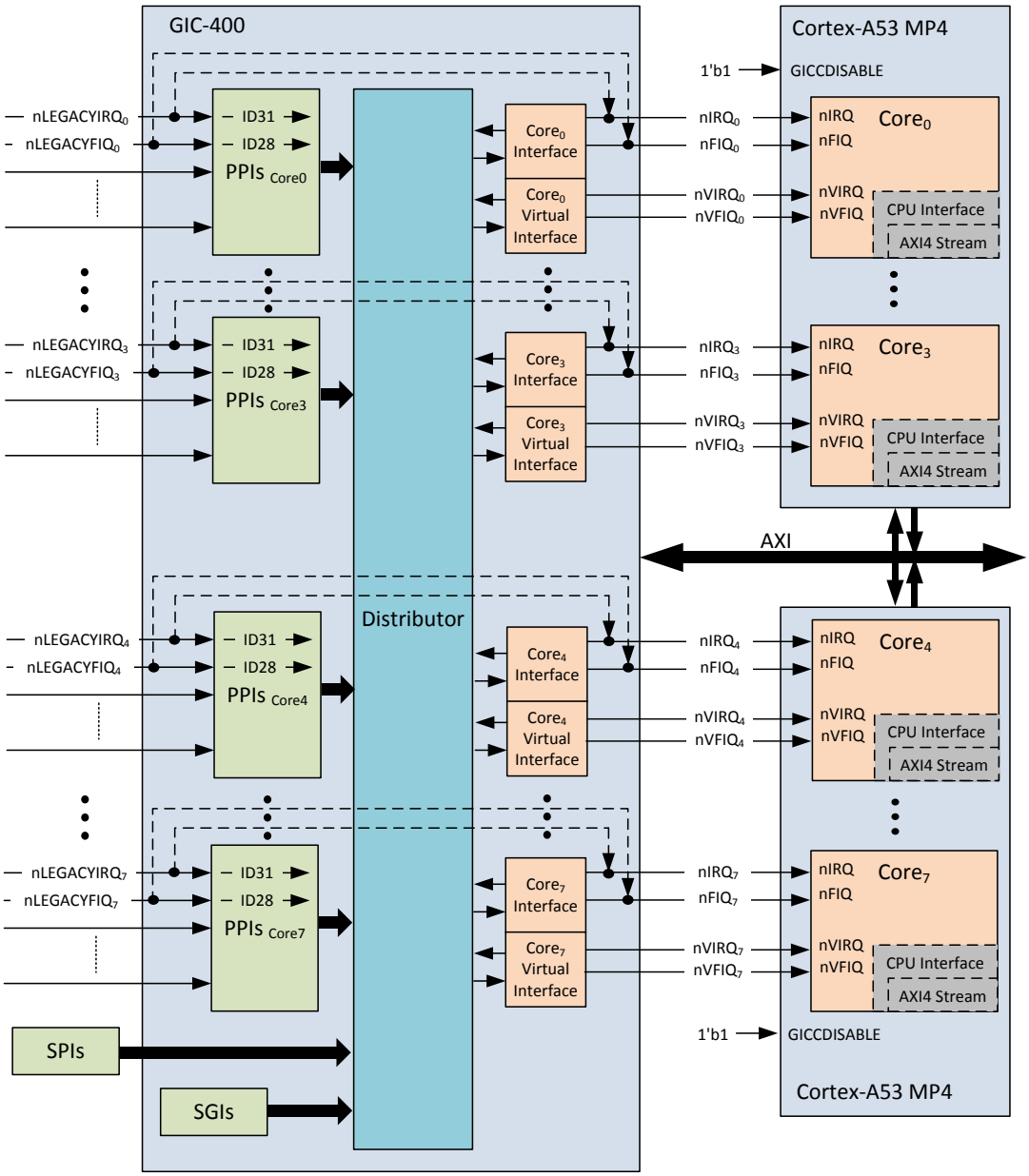

Figure 8-1 Cortex-A53 clusters and GIC-400 operating in legacy mode