Application Note

Arm CMN-600AE Event Interface Connection

Non-Confidential

Issue 1.0

ARM051-799564642-325

Release information

Document history

Issue	Date	Confidentiality	Change
1.0	06-Jul-2021	Non-Confidential	Initial release

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

developer.arm.com

Progressive terminology commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change.

If you find offensive terms in this document, please email **terms@arm.com**.

Contents

1 Introduction

This document describes the event interface connection requirements between MMU-600AE [1] and CMN-600AE [2], [3].

1.1 References

Table 1-1: Referenced documents

Reference	Document number	Author(s)	Title
[1]	101412_0100_00_en	Arm	Arm® CoreLink™ MMU-600AE System Memory Management Unit TRM
[2]	101408_0100_04_en	Arm	Arm® CoreLink™ CMN-600AE Coherent Mesh Network TRM
[3]	101410_0100_04_en	Arm	Arm® CoreLink™ CMN-600AE Coherent Mesh Network Safety Manual

1.2 Conventions

The following subsections describe conventions used in Arm documents.

1.2.1 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 the Arm Glossary for more information: **https://developer.arm.com/glossary**.

This document uses the following terms and abbreviations.

Table 1-2: Terms and abbreviations

Term	Meaning
CMN	Coherent Mesh Network
CSS	Core Sight System-on-chip
DSU	DynamIQ Shared Unit
FMEA	Failure modes and Effects Analysis - a qualitative safety analysis
FMEDA	Failure modes, Effects and Diagnostic Analysis – a quantitative safety analysis
FSM	Finite State Machine
MMU	Memory Management Unit
TRM	Technical Reference Manual

Term	Meaning
WFE	Wait For Event CPU instruction

1.2.2 Typographical conventions

Convention	Use
italic	Introduces citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` **`bold`**	Denotes language keywords when used outside example code.
`monospace` <u>`underline`</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.
Note	This represents an important piece of information that needs your attention.
Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
Remember	This is a reminder of something important that relates to the information you are reading.

1.3 Feedback

To provide feedback on Arm CMN-600AE Event Interface Connection create a ticket on **https://support.developer.arm.com**.

If you have comments on content, send an email to errata@arm.com and give:

- The title Application Note Arm CMN-600AE Event Interface Connection.
- The number ARM051-799564642-325.
- If viewing a PDF version of a document, the page number(s) to which your comments refer.
- If viewing online, the topic names 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 technical content and Frequently Asked Questions (FAQs) on Arm Developer.

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.

2 Problem description

2.1 CPU Event Signaling

Many system IPs from Arm, such as the MMU-600AE, drive an event signal to the CPU that is intended for consumption by software.

In typical usage, the event signal serves as a CPU power-mode wakeup signal when the CPU executes a WFE (Wait For Event) instruction. The CPU event handling allows event reporting to be imprecise, in that events are not required to be reported individually. Multiple events from different sources and back to back events from the same source that occur in the same WFE processing window are consolidated and reported as a single event.

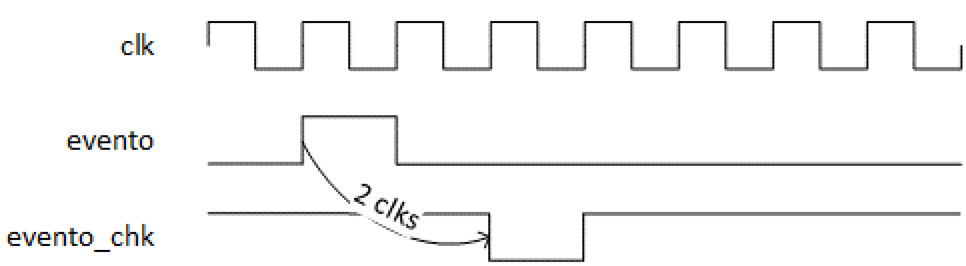
While the destination for the signal is the CPU cluster – more specifically the DSU-AE sub-block within the cluster – the routing pathway often utilizes the Coherent Mesh Network interconnect such as the CMN-600AE.

2.2 MMU-600AE Event Interface

The event interface on MMU-600AE comprises two output signals, evento and its duplicate, evento_chk. The duplicate chk signal is asserted 2 cycles after the corresponding primary signal, and is driven with inverse polarity.

MMU-600AE signals an event by driving a single cycle pulse on these two signals, as shown in Figure 2-1.

Figure 2-1: MMU-600AE event interface signals



2.3 CMN-600AE Event Interface

The event receiver interface on CMN-600AE comprises a REQ/ACK signal pair EVENTOREQ and EVENTOACK, and their duplicates EVENTOREQCHK and EVENTOACKCHK.

EVENTOREQ and EVENTOREQCHK are input signals of CMN-600AE driven by an external component. EVENTOACK and EVENTOACKCHK are output signals driven by CMN-600AE to the external component.

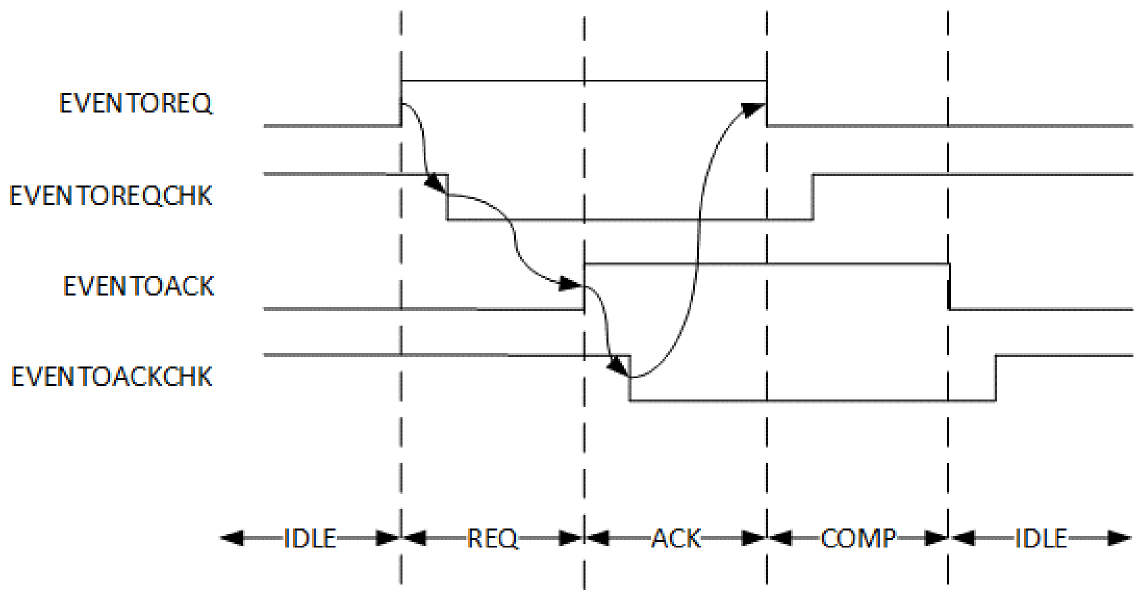
The CHK signals are asserted 2 cycles after the corresponding non-CHK signals and are driven with inverse polarity.

Note

EVENTOREQ* and EVENTOACK* signals can be in two different clock domains. Hence, the external component must synchronize EVENTOACK* signals to its clock domain.

These signals implement a 4-phase REQ/ACK handshake between an external component and CMN-600AE with four states as shown in Figure 2-2.

Figure 2-2: CMN-600AE event interface 4-phase handshake



The initial state is IDLE coming out of reset.

The state sequence shown in the diagram is:

- REQ
  To initiate an event request, the external component must assert EVENTOREQ and EVENTOREQCHK and transition to the REQ state.
- ACK
  Next, CMN-600AE asserts EVENTOACK and EVENTOACKCHK and transitions to the ACK state.
- COMP
  To complete the event request, the external component must deassert EVENTOREQ and EVENTOREQCHK and transition to the COMP state.

- IDLE
  In response to the event completion request, CMN-600AE deasserts EVENTOACK and EVENTOACKCHK and transitions to the IDLE state.

The following rules must be obeyed to adhere to the four-phase handshake protocol.

- When EVENTOREQ* signals are asserted, they must remain asserted until EVENTOACK* signals are asserted.
- When EVENTOREQ* signals are deasserted, they must remain deasserted until EVENTOACK* signals are deasserted.

2.4 Event Interface Protocol Mismatch

The event signaling interfaces on MMU-600AE and CMN-600AE follow different protocols as shown in the previous sections. This protocol mismatch does not allow direct signal connections at the event interface between the two IPs. Additional glue logic is needed to bridge between the protocols.

3 Workarounds

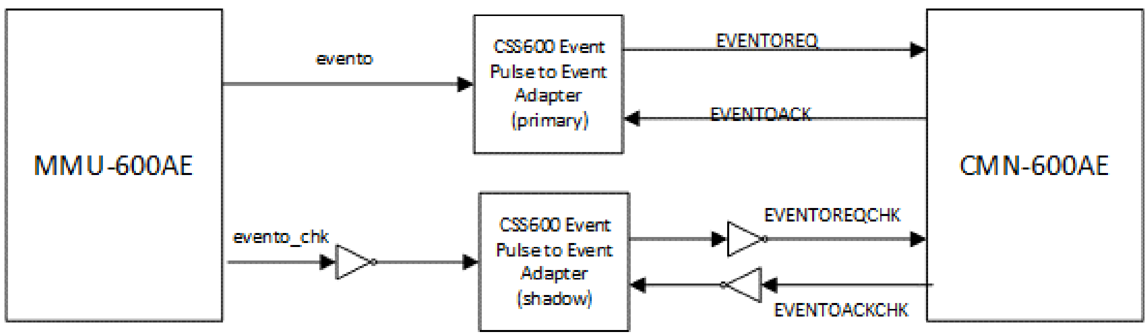
Arm recommends one of the following approaches to address this problem:

- CSS600 event pulse to event adapter solution in section 3.1
- Generic event pulse to event adapter solution in section 3.2

3.1 CSS600 event pulse to event adapter solution

Customers who have access to Arm CoreSight™ SoC-600 IP can use the CSS600 event pulse to event adapter component to connect the event interface signals. Two copies of the adapter component are needed, one to convert the primary set of signals, and the other to convert the duplicate set of CHK signals, as shown in Figure 3-1.

Figure 3-1: Event interface connections using CSS600 component



The following table provides details of the CSS600 event adapter I/O list and the required connections.

Table 3-1: CSS600 event adapter pin connections

Adapter instance	I/O	Direction	Connection Information
Primary	pulse_in	Input	Connect to MMU-600AE **evento** output
	pulse_req	Output	Connect to CMN-600AE **EVENTOREQ** input
	pulse_ack	Input	Connect to CMN-600AE **EVENTOACK** output
	clk_s	Input	Connect to same clock as MMU-600AE **aclk**
	reset_s_n	Input	Connect to same reset as MMU-600AE **aresetn**
	clk_s_qactive	Output	Connect to the MMU-600AE clock controller
	pwr_qreq_n	Input	Connect to MMU-600AE power controller
	pwr_qaccept_n	Output	
	pwr_qactive	Output	
Shadow	pulse_in	Input	Connect to inverted MMU-600AE **evento_chk** output

	pulse_req	Output	Invert and connect to CMN-600AE EVENTOREQCHK input
	pulse_ack	Input	Connect to inverted CMN-600AE EVENTOACKCHK output
	clk_s	Input	Connect to same clock as MMU-600AE aclk_fdc
	reset_s_n	Input	Connect to same reset as MMU-600AE aresetn_fdc
	clk_s_qactive	Output	Connect to the MMU-600AE clock controller
	pwr_qreq_n	Input	Connect to MMU-600AE power controller
	pwr_qaccept_n	Output	
	pwr_qactive	Output	

3.2 Generic event pulse to event adapter solution

Customers who do not have access to Arm CoreSight SoC-600 IP can design their own generic event pulse to event adapter. This section provides a functional description of a generic event pulse to event adapter block.

Two copies of the adapter should be instantiated, one instance to convert the primary set of signals and the other instance to convert the duplicate set of CHK signals as shown in Figure 3-1.

3.2.1 Adapter block I/O list

The adapter block interface should be designed with a set of input and output signals as defined in the table below.

Table 3-2 : Generic event adapter block I/O list

I/O	Direction	Description
pulse_in	Input	Event pulse input
pulse_req	Output	Event request signal for 4-phase handshake
pulse_ack	Input	Event acknowledge signal for 4-phase handshake. pulse_ack input should be synchronized to the adapter clock domain before it is used.
clk_s	Input	Clock input
reset_s_n	Input	Reset input
clk_s_qactive	Output	Indicates when adapter block is active and needs the clock

Note

Additional signals related to power controller interface in CSS600 adapter shown in Table 3-1 are not required.

3.2.2 Adapter block operation

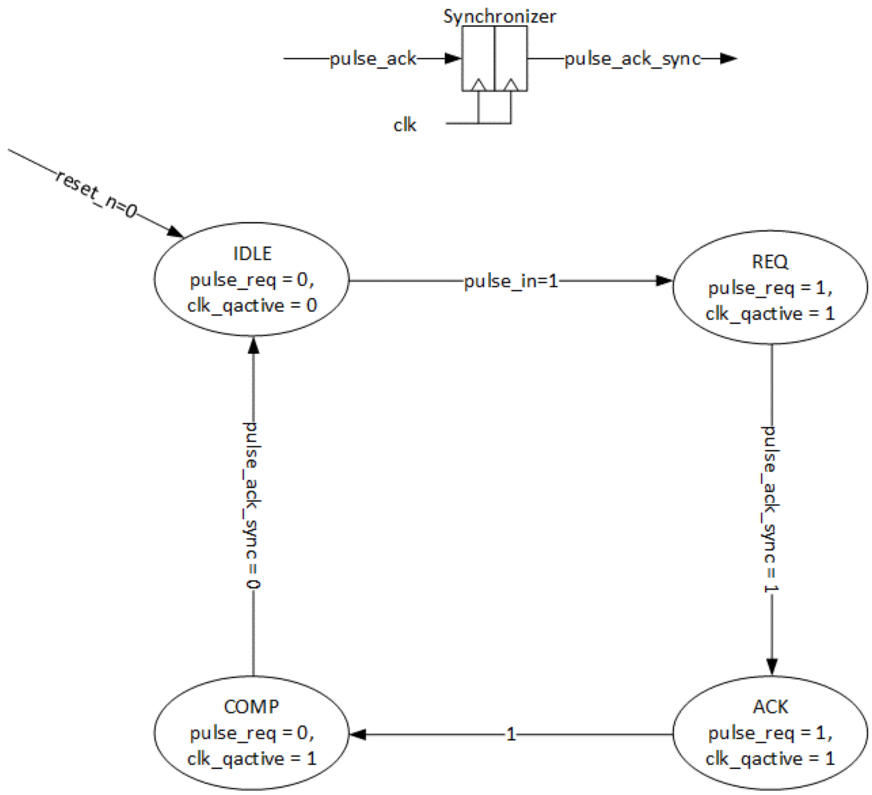
The adapter should be designed to operate in the same clock and reset domain as the MMU-600AE event interface logic and use the MMU-600AE's clock and reset signals.

The adapter should implement an FSM comprising the four states based on the 4-phase handshake protocol as shown in **Figure 2-2**. The adapter outputs should be driven as described below.

- IDLE. In the IDLE state, both pulse_req and clk_qactive outputs should be deasserted.
- REQ. In the REQ state, both pulse_req and clk_qactive outputs should be asserted.
- ACK. In the ACK state, both pulse_req and clk_qactive outputs shall remain asserted.
- COMP. In the COMP state, pulse_req should be deasserted while clk_qactive output should continue to remain asserted.

The state transitions are illustrated in the FSM diagram shown in **Figure 3-2**.

Figure 3-2: Adapter FSM diagram



The adapter block asserts clk_qactive output when it is in a non-IDLE state. To ensure that the adapter's clk is ON during adapter operation, the clk_qactive output should be sent to the clock controller.

Note

When an event is received from MMU-600AE and processed by the adapter, there is a 12-15 clock cycle window (based on synchronizer delays), in which the FSM cycles through the state transitions to get to IDLE state, before being ready to receive a second event. Events reported within the 12-15 cycle window can be considered as combined with the first reported event. This is OK because the WFE processing window is much longer than 15 clock cycles.

3.3 Qualitative FMEA

This section provides an example qualitative FMEA and describes the failure modes applicable to the event interface connection logic when using the adapters.

The main function of the adapter is to transfer the event pulse from MMU-600AE to CMN-600AE. Errors resulting from random hardware faults in the adapter block and/or interface signal connections are mitigated through two safety mechanisms. These are

- SM1: Adapter block duplication. The dual set of interface signals and adapter instances provide redundant event transfer paths between MMU-600AE and CMN-600AE
- SM2: Asynchronous signal interface checkers in CMN-600AE. This mechanism detects interface signal faults such as incorrect or illegal signal transitions by comparing the outputs of the adapter blocks and throwing a timeout error in CMN-600AE. Details of the asynchronous interface checker can be found in [3]

The failure modes applicable to the event interface connection logic are described in Table 3-3.

Table 3-3 : Example FMEA

Potential failure mode	Potential effect(s) of failure mode	Potential cause(s) of failure	Fault model	Fault type	Safety mechanism
Dropped event	Event not reported preventing CPU wake-up	pulse_in input stuck-at-0, adapter logic fault	Permanent, Transient	Random	SM1 and SM2
Spurious event	Spurious event reported causing premature CPU wakeup	pulse_in input stuck-at-1, adapter logic fault	Permanent, Transient	Random	SM1 and SM2
REQ/ACK protocol violation	Event not reported preventing CPU wake-up	pulse_req/pulse_ack stuck-at fault, adapter logic fault	Permanent, Transient	Random	SM1 and SM2

3.4 Functional safety requirements

The following requirements must be fulfilled to maintain the functional safety integrity of the product.

3.4.1 Functional verification

The event interface connections between MMU-600AE and CMN-600AE using the adapter block must be verified at the system level by exercising the event interface.

3.4.2 FMEDA

A quantitative failure mode analysis must be performed on the event interface connections, using the netlist data, and incorporated into the system FMEDA. The qualitative FMEA example can be used as a guide in determining the failure modes applicable to the event interface logic.