Ncore 3.4 - SysCmd Architecture Specification

Release: 3.4.22

Rev: 0.56, March 3, 2023

ARTERIS® NCORE 3.4 - SYSCMD ARCHITECTURE SPECIFICATION

Release Information

Version	Editor	Change	Date
0.1	MF/MK	Initial Document template created	10/24/2019
0.2	MF	Document started from template	04/20/2021
0.3	MF	added SysCo protocol	05/16/2021
0.4	MF	added SysEvent protocol	05/23/2021
0.41	MF	redefined opcodes for SysReq/SysRsp, adjust width of RMessageID	05/28/2021
0.42	MF	Included feedback from MK	06/21/2021
0.43	MF	Modified event flow to send all requests to DVE for distribution	07/18/2021
0.50	MF	Updated version number	08/13/2021
0.52	MF	Upgraded the spec version --> scaled down version for Ncore 3.2	08/13/2021
0.53	MF	Updated SysCoReq FSM --> scaled down version for Ncore 3.2	11/17/2021
0.54	MF	Carry forward to Ncore 3.4	06/03/2022
0.55	MF	SysEvt - simpler broadcasting scheme removes signaling from DCE --> DVE	06/08/2022
0.56	MF	updated legal configurations for unit interfaces CONC-11481, added Table 6	02/08/2023
Legend:	MK	Mohammed Khaleeluddin	
	MF	Michael Frank	
	AA	Arkadi Avrukin	
	JU	Junie Um	
	KJ	Kjeld Svendsen	
	Xx	Whoever else edited this document	

Note:

- Notification of coherent agents for SysCoReq/Ack is structurally the same as a DVM message being distributed to multiple agents -- this is a typical multicast problem
- Distributing events to all agents on the notification list is another multicast problem
- Should we consider creating a generalized SysReq agent (derived from DVE) that receives all SysReq messages and performs multicast distribution and consolidation of the responses?

Confidentiality Status

This document is Confidential and Proprietary. This document may only be used and distributed in accordance with the terms of the agreement entered into by Arteris IP and the party that Arteris IP delivered this document to.

Product Status

The information in this document is ***Preliminary***.

Web Address

http://www.arteris.com

Table of Contents

Table of Figures

Table of Tables

Preface

This preface introduces the Arteris® Network-on-Chip Hierarchical Coherency Engine Architecture Specification.

About this document

This technical document is for the Arteris Network-on-Chip Hierarchical Coherency Engine Architecture. It describes the subsystems and their function along with the system's interactions with the external subsystems. It also provides reference documentation and contains programming details for registers.

Product revision status

TBD

Intended audience

This manual is for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses or intend to use the Arteris Network-on-Chip Hierarchical Coherency System (ANoC-HCS).

Using this document

TBD

Glossary

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

Typographic conventions

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.

monospace

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

monospace italic

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name. monospace italic Denotes arguments to monospace text where the argument is to be replaced by a specific value. monospace bold Denotes language keywords when used outside example code.

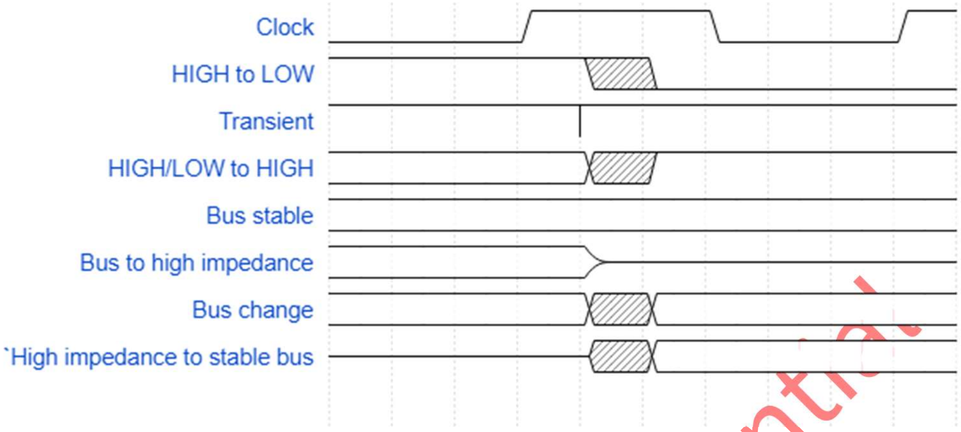
SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the Arteris® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

History of the World II, Mel Brooks.

1. Overview

Ncore 3.0 implements a coherent communication system (Network on a Chip = NoC) for System-On-A-Chip data transport service. It is responsible for data transactions between a large number of requesters (any agent that initiates a transaction) and completers (any agent that responds to a request).

Ncore 3.0 uses the following request and response message pairs to implement data and coherency service:

TABLE 1: OVERVIEW NCORE 3.0 MESSAGING

Message Class	Direction	Request	Response	Reference
CMD (coherent)	iAIU →DCE	CMDreq	CMDrsp	
CMD (non-coherent)	iAIU →DCE iAIU → DMI/DII	CMDreq	CMDrsp	
SNP	DCE → sAIU	SNPreq	SNPreq	
MRD	DCE → DMI	MRDreq	MRDrsp	
Virtualization	iAIU → DVE DVE → sAIU	DVM Snoop	SNPrsp	
State Reply	DCE → iAIU	STRreq	STRrsp	
Directory Update	AIU → DCE	UPDreq	UPDrsp	
Request Buffer Request	DCE → DMI	RBRreq	RBRrsp	
Request Buffer Usage	DMI → DCE	RBUreq	RBUrsp	
Completion	DMI/DII → iAIU	--	CMPrsp	
Concerto Message Error	DMI/DII → iAIU	--	CMErsp	
Data Reply	DMI, DII, sAIU → iAIU	DTRreq	DTRrsp	
Data Write	iAIU → DMI/DII	DTWreq	DTWrsp	
System Message	iAIU → DVE iAIU → DCE/DVE DVE → sAIU/cAIU DCE → sAIU/cAIU	SysReq + OpCode	SysRsp + CMStatus	EventIn for broadcasting to other agents Coherence On/Off Request EventOut & Coherence Ack/Grant SysEvent, SysCo Broadcast & Coherence Ack/Grant

Note:

- *PCredits (option) – this response may carry back credits to an initiating agent to improve roundtrip latency for credits. If no credits are transported,*
- *Legend:*
  iAIU – initiator AIU – AIU sending the first message in a transaction,
  sAIU – snooper AIU – AIU being snooped by DCE or receiving a DVMOp from DVE

This document defines a new service, system messages. The purpose of system messages is to convey operating state information or state transitions for agents or asynchronous events within the endpoints of an Ncore domain. In the future they will be used to implement new functionality in the general class of system related activities.

This service will be implemented by a new message class, **System**. System messages will be sent from a source agent to one or more destination agents.

Every system message must receive a response to acknowledge receipt by the destination agent - depending on the protocol implemented by a specific message pair, receipt of response may initiate additional activity (for example a state transition at the source agent).

1.1. SYSreq and SYSrsp Messages

SysReq message is an implementation of the new system message class. The purpose of these messages is to propagate system related information between agents or functional units within the system.

SysReq messages are asynchronous, do not use credit management as there shall be only one outstanding transaction per target, it shall be acknowledged by the receiver returning the SysRsp message. The SysRsp response implements status and error reporting in the CMStatus:

Command error[1]:

- command was not properly formatted
- No support for this operation
- Wrong unit - this unit is not supporting the OpCode or unknown OpCode
- Unit is busy and cannot perform the requested operation at this time
- Ok - no error occurred, command completed as expected

SysReq OpCode:

- SysReq command uses CMType = 0x7B[2]
- SysRsp response uses CMType = 0xFB[3]

TABLE 2: SYSREQ COMMAND LAYOUT

Name of Field	Width Param	Param Values	Description	Direction	Source
CMH:			**CCMP Message Header**		
TargetId	wTargetId	6 - 12	Target Identifier	In/Out	
InitiatorId	wInitiatorId	6 - 12	Initiator Identifier	In/Out	
CMType	wCMType	8	Type of Message - SysReq Opcode	In	
MessageId	wMessageId	6 - 12	MessageId	In/Out	
HProtection	wHProt	Derived	Protection for the Concerto Message Header fields	In/Out	
CMHE:			**CCMP Message Header Extension**		
TTier	wTTier	0 - 4	Traffic Tier of the message for use by the transport fabric	In/Out	
Steering	wSteering	0 - 4	Steering. This field is used to indicate steering of the message the transport fabric should apply to this message	In/Out	
Priority	wPriority	0 - 4	Priority of the message to be used by the transport fabric	In/Out	
QL	wQL	0 - 4	QoS Label. This field is used by the transport fabric for QoS functions such as bandwidth provisioning, isochronicity property, etc. to be applied for this message	In/Out	
CMB:			**CCMP Message Body**		
RMessageID	wRMessageID	6 - 12	MessageID for the command associated with this response	In/Out	
CMStatus	wCmStatus	8	Status - using standard command status layout	In/Out	
SysReqOp	wSysReqOp	4	As of now - only 16 different operations for this command class	In/Out	
~~TimeStamp~~[4]	~~wTimeStmp~~	~~32~~	~~Timestamp~~	~~In/Out~~	~~local~~
NDProt	wSysReqProt	0 - 10	Protection bits for SysReq Message	In/Out	

[1] Some or all of these will not be supported in the initial release Ncore 3.2
[2] This opcode is available - between 0x7A (STRreq) and 0x7C (RBRreq)
[3] This seems to be one of the few free opcodes within the response block (0xF0 - 0xFF)

TABLE 3: SYSRSP RESPONSE LAYOUT

Name of Field	Width Param	Param Values	Description	Direction	Source
CMH:			**CCMP Message Header**		
TargetId	wTargetId	6 - 12	Target Identifier	In/Out	
InitiatorId	wInitiatorId	6 - 12	Initiator Identifier	In/Out	
CMType	wCMType	8	Type of Message - SysReq Opcode	In	
MessageId	wMessageId	6 - 12	MessageId	In/Out	
HProtection	wHProt	Derived	Protection for the Concerto Message Header fields	In/Out	
CMHE:			**CCMP Message Header Extension**	**In/Out**	
TTier	wTTier	0 - 4	Traffic Tier of the message for use by the transport fabric	In/Out	
Steering	wSteering	0 - 4	Steering. This field is used to indicate steering of the message the transport fabric should apply to this message	In/Out	
Priority	wPriority	0 - 4	Priority of the message to be used by the transport fabric	In/Out	
QL	wQL	0 - 4	QoS Label. This field is used by the transport fabric for QoS functions such as bandwidth provisioning, isochronic- ity property, etc. to be applied for this message	In/Out	
CMB:			**CCMP Message Body**		
RMessageID	wRMessageID	6 - 12	MessageID for the command associated with this response	In/Out	
CMStatus	wCmStatus	8	Status - using standard command status layout	In	
~~Filler~~	~~wSysReqOp~~	4	~~Unused field - aligns timestamp~~	~~In/Out~~	
~~TimeStamp~~[4]	~~wTimeStmp~~	~~32~~	~~Timestamp~~	~~In/Out~~	~~local~~
NDProt	wSysRspProt	0 - 10	Protection bits for SysRsp Message	In/Out	

[4] Timestamp will not be part of the initial implementation in Ncore 3.2

1.1.1. SysReq Commands

Being not credited imposes restrictions on the use of SysReq messages. It requires that a functional unit must always be able to process a single SysReq command and generate a meaningful response using SysRsp. In the first implementation in NCore 3.2 these message operations will be supported:

Table 4: SysReq Command Operation Payload

Command	OpCode	Payload	Function
SysReq.NOP	0x00	TimeStamp	No function, beyond carrying the timestamp in both directions. This transaction may be used to measure the roundtrip latency between two agents. If an agent does not implement a timestamp counter, the value used in the response shall be -1 (all ones)
SysReq.Attach	0x01	TimeStamp SysCoReq	Attach the requester to the coherency domain. This command is used by a cache coherent initiator (CAIU) to attach itself to the coherency engine (DCE). The command needs to be sent to all interleaved DCE. The OK response signals successful attachment, only after that the agent will be allowed to send coherent requests and participate in snoop protocol
SysReq.Detach	0x02	TimeStamp SysCoReq	Detach the requester from the coherency domain. This command removes the agent from the set of snooped coherent agents managed by DCE
SysReq.Event	0x03	TimeStamp EventReq	System Event - these events are generated within various agents[5] in the system and convey asynchronous events between event producers and event-consumers . These agents are CPUs, monitors for exclusive transactions, SMMU, Accelerators or other agents. Agents may be producer, consumers or both.

[5] In Ncore 3.2 only DCE (exclusive monitors) and AIU (EventInReq) will issue Event messages

1.1.2. Command CMStatus

The CMStatus[7:0] field is shown only as input in the direction column, implying that the field is not transmitted along with the rest of the *SysReq* message at the initiator, but is an input-only field at the target of the message.

The CMStatus[ ] field is used to report the result of a processing step of the operation, including detection of errors, to an Ncore unit. However, *SysReq* being the start of the operation, there are no results to report.

If any type of error is detected in the native request transaction, the initiator AIU must not issue a corresponding *SysReq* message into the Ncore system to commence the processing of the native operation. The operation must be locally terminated and error reported back to the native agent.

A *SysReq* message may encounter a transport error along its path to the target. This error must be reported to the target via the CMStatus[ ] field constructed at the transport interface of the target Ncore unit.

TABLE 5: SYSCMD AND SYSRSP COMMAND STATUS FIELD ENCODING

Status Type	**CMStatus [7:6]**	**CMStatus [5:0]**	
Success (No error)	2b00	SysRsp: System Response carries back the status of the command execution at the target agent	
		[5:3] = 3b000	[2:0]: **xx0:** No Operation performed - this may not be an error! **001:** Completion - Execution state machine is still running **011:** Final - Execution state machine went to IDLE after sending this response **101/111:** reserved (TBD)
		SysReq: {Reserved} - there should be no error on the incoming command, except transport problems	
Error	2b01	SysRsp: System Response carries back the status of the command execution at the target agent	
		[5:3] = 3b000	[2:0]: **xx0:** No Operation performed - unit signals error **001:** Completion - with unknown error (conditions TBD) **011 - 110:** Format Error - reserved **111:** Unit is busy and cannot perform operation at this time

Transport related error status uses the same representation as CmdRsp.

1.2. SysCoReq Implementation

The SysCo protocol is a 4 state handshake protocol implemented by ARM processor DSU. The purpose of the protocol is to attach and detach the agent to the coherent domain.

See ARM document *dsu_trm_100453_0401_03_en.pdf,* section A4.9 Cluster powerdown for a detailed description of the Initiator Agent handshake protocol.

- On power up
  - Adds a request agent (DSU--CAIU) to the coherency domain
  - After de-asserting RESET:
    - SYSCOACK is inactive
    - Agent will respond to snoop and DVM requests but will not yet make cacheable requests
      - ➢ Agent will try to CONNECT by asserting SYSCOREQ signal
    - Interconnect is ready to receive cacheable requests
      - ➢ Interconnect shall assert SYSCOACK
- On power down
  - Remove a request agent (DSU--CAIU) from the coherency domain
  - At this time SYSCOREQ and SYSCOACK are asserted
  - Flushing caches and quiescing traffic:
    - Agent halting execution (WFI/WFE/HALT)
    - Agent de-asserts SYSCOREQ
    - Agent shall continue to respond to snoop and DVM requests but may no longer make cacheable requests
    - The interconnect has no longer active transactions for this agent and the system directory's snoop filters have not any cache lines recorded
    - De-assert SYSCOACK
    - Shut down the agent (assert RESET, power-down)

1.2.1. System Architecture

In support of SysCo (Attach/Detach) each coherent agent (CAIU/ProxyCache[6]) shall implement a SysCo-engine.

In support of SysEvent each agent with event sources or receivers shall implement a SysEvent-engine.

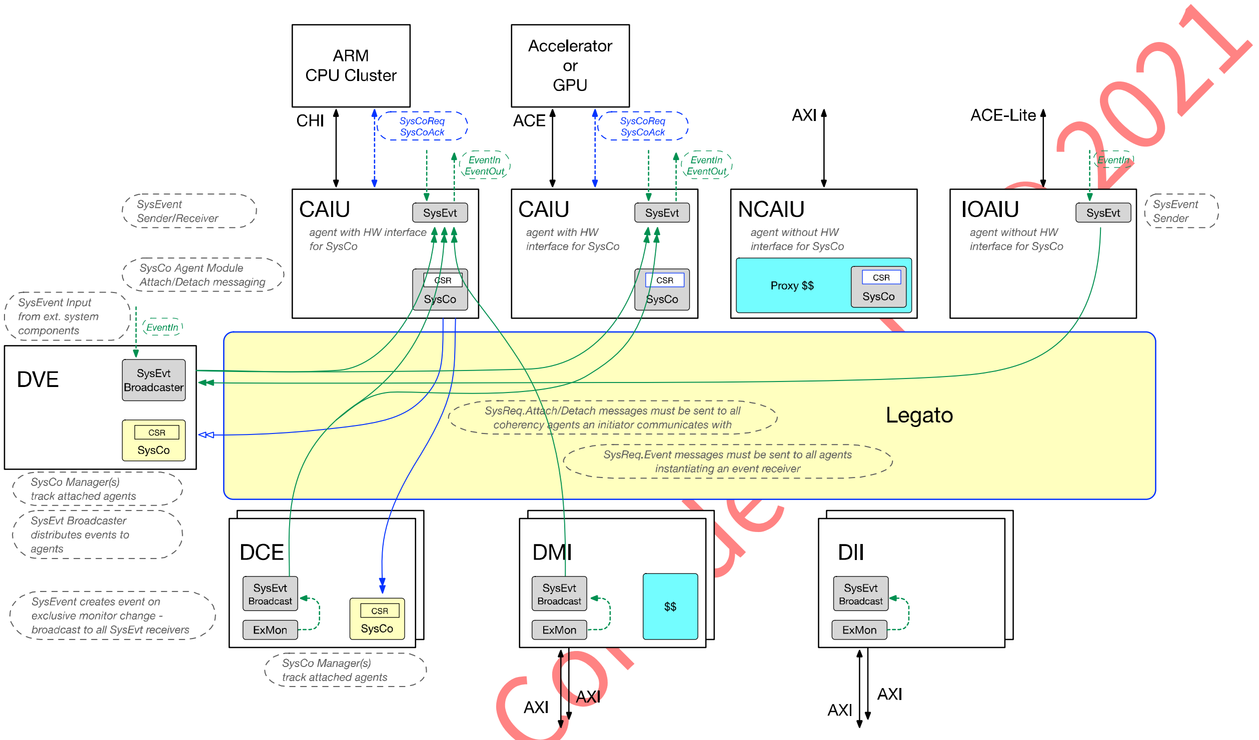


FIGURE 1: SYSCO TOP LEVEL ARCHITECTURE[7]

After reset, an agent will not be part of a coherency domain, it may only send non-coherent transactions and will not respond to snoop requests.

Ncore will use SysReq messaging to notify all coherency managers (DCE) when an agent requires to be attached or detached from the coherency protocol.

[6] Any agent participating in the coherency protocol that does not support SysCoReq/Ack needs to implement a Sysco-engine using CSR protocol to attach/detach to the coherency domain

[7] This high level architecture shows optional exclusive monitors and SysEvent-engines in DMI/DII, these are not part of Ncore 3.2

1.2.2. Attach/Detach Protocol

The initial state of any CAIU is detached - any CAIU that connects to a coherency agent with SYSCOREQ/SYSCOACK (SysCo) support must implement the SysCoReq interface and the SysCo message protocol engine.

Any other coherent initiator agent (CAIU connecting agents that do not support SysCo, NCAIU with ProxyCache) must implement the SysCo message protocol engine. The engine must provide a CSR interface to initiate the coherency attach/detach process by SW (write to register).

A coherency agent must not send any coherency transactions (Snoop/DVMSnoop) to a functional unit, unless that unit has been attached to this coherency agent by using the SysCo protocol.

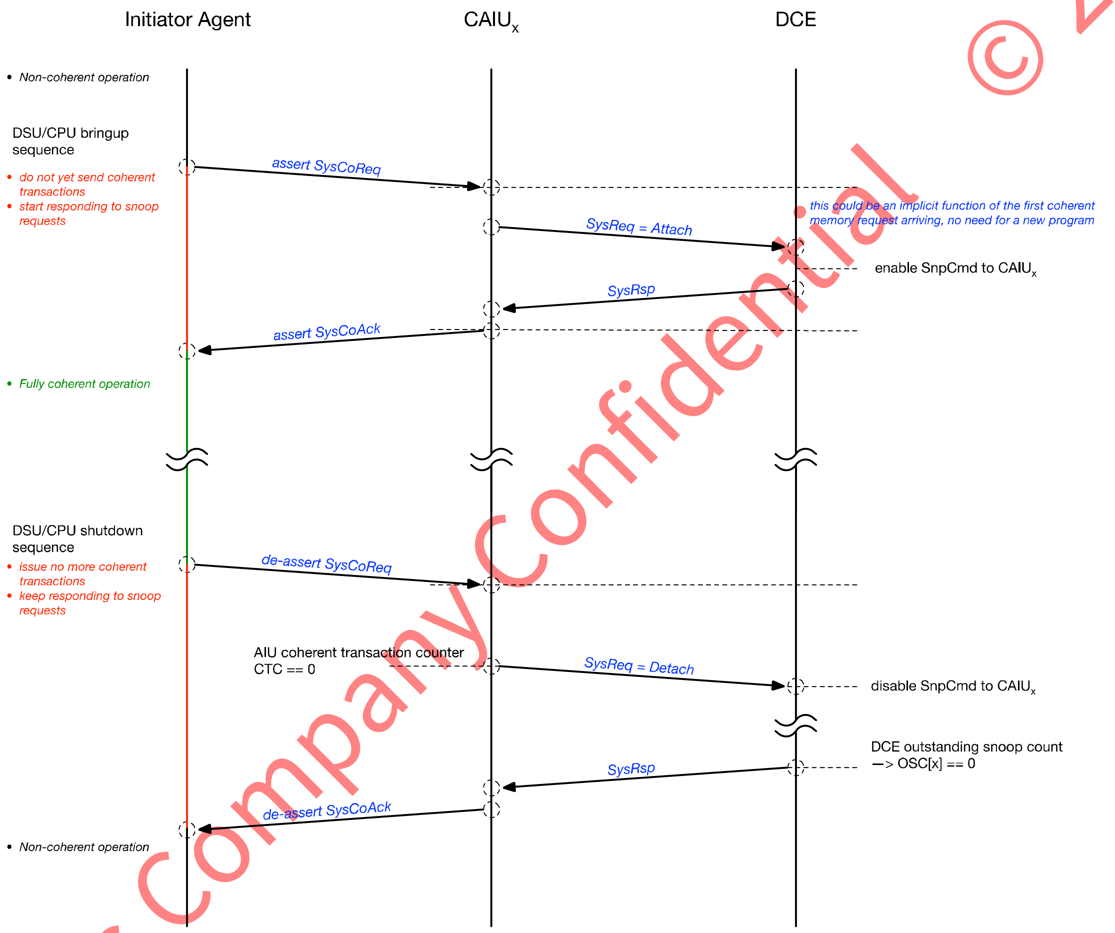

Initiator Agent
CAIUx
DCE
Non-coherent operation
DSU/CPU bringup sequence
do not yet send coherent transactions
start responding to snoop requests
assert SysCoReq
SysReq = Attach
this could be an implicit function of the first coherent memory request arriving, no need for a new program
enable SnpCmd to CAIUx
SysRsp
assert SysCoAck
Fully coherent operation
DSU/CPU shutdown sequence
issue no more coherent transactions
keep responding to snoop requests
de-assert SysCoReq
AIU coherent transaction counter CTC == 0
SysReq = Detach
disable SnpCmd to CAIUx
DCE outstanding snoop count —> OSC[x] == 0
SysRsp
de-assert SysCoAck
Non-coherent operation

FIGURE 2: SYSCO FLOW

Figure 2 shows the details of the SysCo protocol and the associated exchange of messages. For clarity, only messaging with a single DCE is shown in the flow chart - in a real system a CAIU needs to attach to all DCE and DVE covering the coherency domain by sending messages to each one. The CAIU may start using coherent transactions only after all SysRsp have been received.

While in *detached* state, CAIU must not send coherent transactions, it is permitted (but not required) to respond to snoop requests.

CAIU must keep track of outstanding requests and keep a running count of outstanding coherent transactions. When a transition from attached to *detached* state has been requested, either by HW SysCo protocol or by a CSR write request, it must stop making coherent transaction into the system (ARM DSU will no longer make coherent request after SysCoReq has been deasserted) and wait until all outstanding coherent transactions have been retired, before making a SysReq.Detach to DCEs.[8]

A coherency agent (DCE/DVE) shall no longer snoop an agent after receiving SysReq.Detach from it[9]

A coherency agent must keep track of outstanding requests and keep a running count of outstanding coherent transactions to an agent, SysReq.Detach shall only be acknowledged by a coherency agent after all outstanding coherent transactions for a snooped agent have been completed.[10]

1.2.3. SysCo Engine at Initiator Agent

The SysCo engine manages the attach/detach process for each coherent initiator agent.

For agent interfaces with HW protocol support 2 signals (SysCoReq = please attach me, SysCoAck = you are part of the coherency domain) will be provided. These signals may be asynchronous to the internal clock domain and shall be properly synchronized using proper CDC implementation.

For agent interfaces without HW protocol support, the CSR interface may be used to request attachment or detachment from the coherency system.

Internally, each SysCo Engine shall implement (also see Figure 3):

- CSR with 4 bits
  - C: Connecting, read-only
    - On read returns ‘1’ when the agent is waiting to connect (FSM.state == ***Connect***)
  - Req: Activate, read-write
    - Write to ‘1’ to trigger FSM.state transition from ***Idle*** to ***Connect***
    - Write to ‘0’ to trigger FSM.state transition from ***Connect*** to ***Detach***
    - Cleared by reset
  - A: Attached, read-only
    - On read returns the current state of the interface (0 - detached, 1 - attached)
  - Err: Error, read, WZ (write-to-zero)
    - read as ‘1’ when a protocol error or time-out has been signaled
    - read as ‘0’ after reset - or after no protocol error has been detected in a transaction
    - write to zero - clears the bit
- SysCoFSM with 6 states:
  - ***Idle*** – after reset and when disconnected - *SysCoAck* shall be de-asserted in this state
    - On SysCoReq assertion or CSR.Req set, transition to ***Connect***
  - ***Connect*** – request to connect by sending a *SysReq.Attach* message to all coherency agents this agent communicates with and wait for acknowledgement by protocol, a time-out counter shall be implemented to report error on no response.
    - Transition to ***Attached*** when *SysRsp.Ok* received
    - Transition to ***Attach-Error*** when *SysRsp.Error* or time-out
  - ***Attach-Error*** - transient state, sets CSR.Err and transitions to ***Detach***, reach a safe state
  - ***Attached***– connected, normal operating state - *SysCoAck* shall be asserted in this state

[8] This requirement guarantees termination of concurrent traffic at the source and, assuming that software initiated a proper shutdown of local caches, also maintains consistent memory state

[9] This guarantees termination of snoop traffic initiated by other agents

[10] Only after all coherent traffic has completed, the agent may be transitioning to detached state and acknowledge the transion to the DSU

    - On *SysCoReq* de-assertion or *CSR.Req* cleared, transition to ***Detach***
  - ***Detach*** – requesting to disconnect by sending a *SysReq.Detach* message to all coherency agents this agent communicates with and wait for acknowledgement by protocol, a time-out counter shall be implemented to report error on no response.
    - Transition to ***Idle*** when *SysRsp.Ok* received
    - Transition to ***Detach-Error*** when *SysRsp.Error* or time-out
  - ***Detach-Error*** - transient state, sets CSR.Err and transitions to ***Idle***, reach a safe state

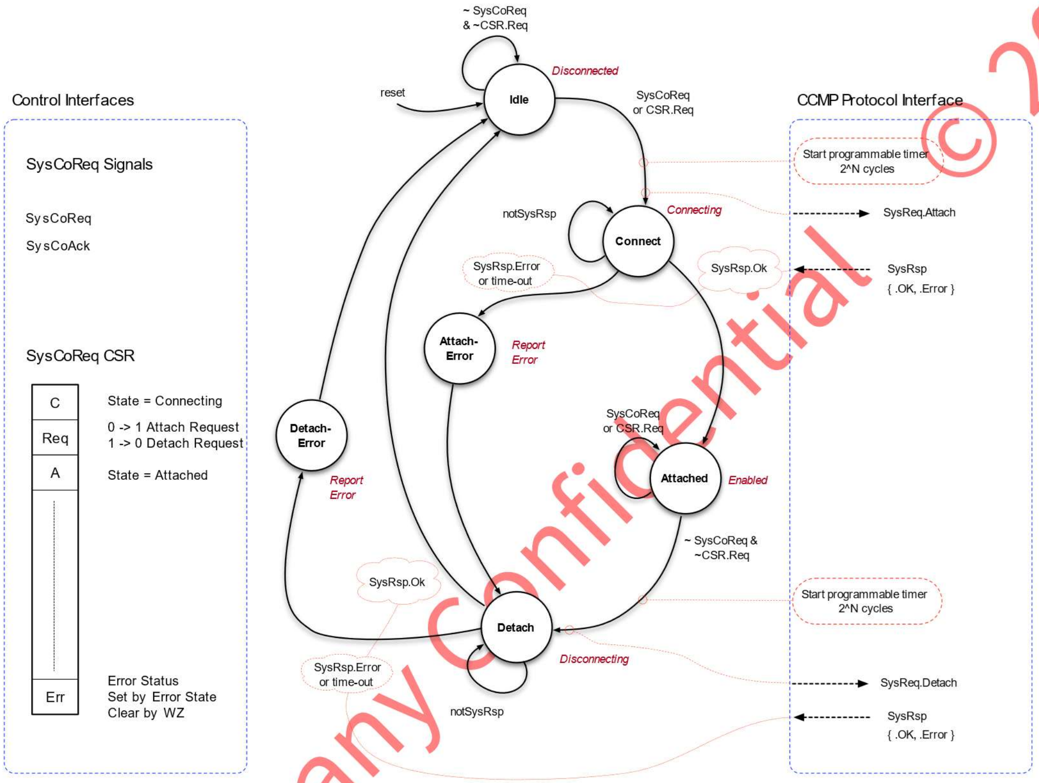


FIGURE 3: SYSCO STATE MACHINE

1.2.4. SysCo Engine at Coherency Agent

The SysCo engine manages the attach/detach process for each coherency agent. Coherency agents are functional units which issue Data Snoop or DVM Snoop requests. These agents need to know if message target is active and will respond to a request. Snooping an inactive agent would break the protocol and, even though it could be detected by using the currently implemented time-out mechanism, the performance impact of an unreliable transaction is not acceptable to customers.

The coherency agent's SysCo engine manages the participation of initiator agents in the snoop protocol by preventing snoops being sent to detached agents.

Agents that want to participate in the coherency protocol must register by sending a SysReq.Attach command to any coherency agent (DCE/DVM) they expect to be communicating with.

Agents that transition to sleep or any other inactive state and can no longer participate in the coherency protocol by responding to snoops, must detach from the coherency domain by sending a SysReq.Detach command to all coherency agents within said domain.

The coherency agent will track all outstanding transactions to agents and will respond by acknowledging the detach request as soon as no more active transactions exist for that agent (all outstanding snoop responses have been received).

Internally, each coherency agent's SysCo engine shall implement:

- At least one CSR with (at least) one bit T[i] to tracked each agent's state, where i is the unique ordinal index of the initiator agent in the ordered set of initiators as assigned by Maestro
    - T[i] == 0 indicates agent$_i$ is not actively participating in the coherency protocol and no transaction with that agent is allowed.
    - T[i] == 1 indicates the agent$_i$ is active and participating in the coherency protocol, snoop or DVM transactions may be issued.
    - For DCE:
        - The coherency protocol allows cache lines to be silently evicted from any agent's cache. As a consequence, the tracking snoop filters implemented in DCE may not correctly reflect the agent's cache state and falsely indicate presence of a line.
        - It is required that agents that are no longer meant to participate in the coherency protocol go through a well defined sequence of steps:
            i. Clean and invalidate the contents of the device's cache
            ii. Initiate transition out of the coherency domain, by following the HW handshake provided for Arm DSU or by SW procol initiating the transition using the provided CSR interface
            iii. Entering sleep or power down state
        - After steps (i) and (ii), it can be safely assumed that the agent does no longer hold valid data and any presence in the snoop filter is a result of silent evictions.
        - If a presence is indicated by a snoop filter and the agent is in detached state (T[i] == 0), no snoop to that agent shall be sent, the snoop filter entry for that agent shall be updated to remove the cache line and a spurious snoop filter event shall be generated to the performance counters
    - The set of CSR holding the T[i] bits shall provide read-only access to retrieve the current state (attached or detached)
    - An engineering register within CSR space shall be provided - writes to this register shall modify the vector T[i]
    - For DVE:
        - No DVMSnoop shall be sent to a detached agent

- SysCoMgr to receive SysReq messages from initator agents:
    - On receipt of SysReq.Attach:
        - Map the SourceID of the arriving SysReq to the associated T[i] and set the bit
        - Issue a SysRsp.Ok if the CMStatus of SysReq is OK
        - Issue a SysRSP.Error if
            i. CMStatus or SysReq indicates an error
            ii. target does not exist within the T-vector
            iii. target is already set - we should not see duplicate attempts to set/clear a bit, this would be considered a protocol violation!
    - On receipt of SysReq.Detach:
        - Map the SourceID of the arriving SysReq to the associated T[i] and clear the bit
        - Issue a SysRsp.Ok if the CMStatus of SysReq is OK
        - Issue a SysRSP.Error if
            i. CMStatus or SysReq indicates an error
            ii. Target does not exist within the T-vector
            iii. Target is already clear - we should not see duplicate attempts to set/clear a bit, this would be considered a protocol violation!

SysReq messages are not credited - special care must be taken to avoid loss of messages and guarantee proper accounting through SysRsp. Initiating agents shall generate no more than one outstanding message per each coherency agent at a time. This limits the total number of transactions arriving at a coherency agent.

It is an implementation choice to use a queue on the receiver side and execute SysReq only when it has the resources to respond with SysRsp.

1.3. Event Propagation

Events are asynchronous messages and will be used by agents to notify other agents in a system that an action shall be taken. Other than interrupts, events do not change the flow of program execution. Agents may be waiting on events, entering a lower-power state while doing so.

In the ARM ecosystem events may be used to quickly synchronize execution between agents by explicitly waiting for (WFE instruction) and sending events (SEI instruction). Other sources of events may be defined, for example, exclusive monitors changing state or SMMU service requests completed.

Events generated at a source within an agent will enter Ncore through the *EventInputInterface* and will be forwarded to DVE for distribution. DVE will broadcast event messages to all active[11] agent interfaces (CAIU) with an Event Receiver in a system. The Event Receiver's *EventOutputInterface* will deliver events to the connected target (e.g. a processor cluster (DSU) or accelerator).

Event interfaces use a 4-phase handshake protocol implemented with 2 signals: EventReq, EventAck.

Events are not commulative, multiple events arriving at an agent interface will only generate a single event handshake on the interface. Event messages arriving while a handshake is actively taking place will be registered and will initiate another handshake after the previous has completed.

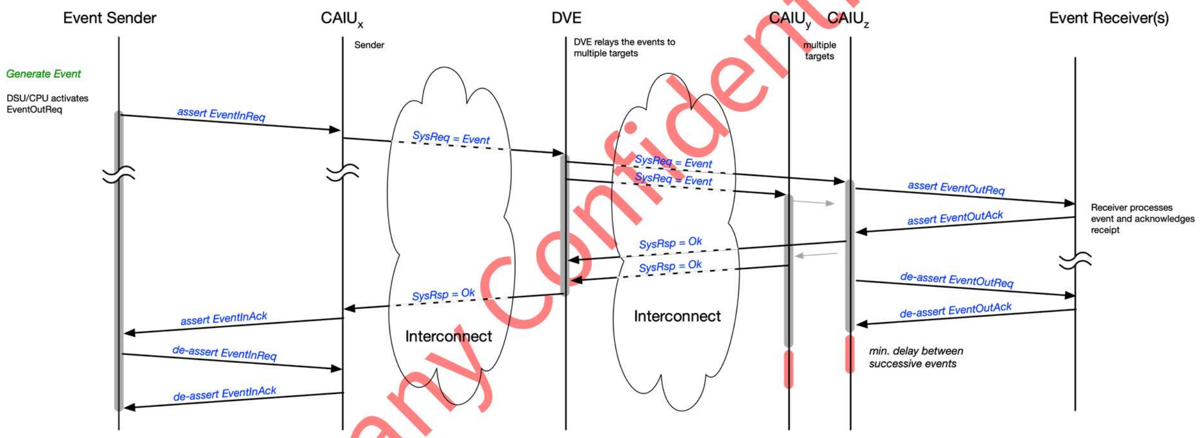


FIGURE 4: EVENT MESSAGING

Requirements:

- Each AIU shall support a bi-directional event interface
- The event interface may receive events from the attached agent following the REQ/ACK-protocol
- The event interface shall forward events, generated internally or received from another agent, to the attached agent
- A received/generated event will be sent to DVE
- Events will be transported within Ncore using SysMsg transactions
- Event messages will be broadcast to all active AIU by DVE
- Events may originate within Ncore (when an exclusive monitor is cleared) - these events will be broadcast by the agent hosting the exclusive monitor
- Events entering Ncore at any AIU shall be forwarded to all AIUs

[11] An interface is considered active if it has registered as participating in coherency traffic by SysCoReq protocol - note, at this point this may be restrictive and we may consider introducing a separate activation protocol in the future

1.3.1. Event Messaging Architecture

Please refer to Figure 1 for the system overview.

The event architecture knows different agents:

Source = any system function that generates events, either as a consequence of state changes or by program execution (e.g. instructions SEV (send event))

- Processors executing SEV
- Page table activity, misses/errors/service requests in SMMU
- Accelerators, finishing tasks or computations
- Global exclusive monitors (DCE/~~DMI/DII~~) changing state

TABLE 6: EVENT MESSAGING INTERFACES - SOURCES AND DESTINATIONS

Component	Type	Signal	Present	Function
C-AIU CHI	Source Sender Receiver	EventInReq, EventInAck EventOutReq, EventOutAck	yes yes	CPU cluster sending events to system CPU cluster receiving events from NoC
C-AIU ACE	Source Sender Receiver	EventInReq, EventInAck EventOutReq, EventOutAck	yes yes	CPU/accelerator cluster sending events to system CPU cluster receiving events from NoC
NCAIU ACE_Lite + DVM	Source Sender	EventInReq, EventInAck	yes/opt	peripheral I/F - with SMMU
NCAIU AXI + ProxyCache	Receiver	--	yes	AXI with Proxy Cache receives Event message from DCE for Exclusive Monitor event and does nothing about it. AXI with Proxy Cache issues SysReq.Attach/Detach to DCE for System Coherency.
NCAIU ACE_Lite	--	--	no	
NCAIU AXI	--	--	no	
DCE	Source	--	yes	No external interface exposed, source of events is internal
DMI	--	--	no	no internal source of events
DII	Source	--	not in 3.4/3.6	possible use to receive events from peripherals connected downstream

Note:

1. Optional - allow configurability within Maestro to populate individual peripheral interfaces. NCAIU with ACE-Lite E, connecting to an SMMU shall implement the interface because SMMU is capable of emitting events

Sender - Block within an agent interface that implements event inputs (EventInReq, EventInAck). A sender will convert the arriving event (assertion of EventInReq) into SysReq.Event messages, passing the message to DVE and acknowledge the event using the above defined 4-phase handshake. The sender will receive the SysRsp.OK from DVE and shall report an error on timeout (not receiving OK responses to sent message) or if response indicates an error.

Receiver - Block within an agent interface that implements event outputs (EventOutReq, EventOutAck). A receiver will convert arriving SysReq.Event messages into event signals, asserting EventOurReq. The receiver shall respond back to the initiator by returning SysRsp.OK.

Each agent participating in the Event protocol shall implement a CSR to provide:

- Timeout value (implementation dependent)
- Status Bit(s) to indicate Error or detailed Error Status (implementation dependent)

- Enable Bit to enable send protocol - Even when disabled, the sender shall acknowledge event requests on the 2-wire interface
- Enable Bit to enable receiving - Even when disabled, the Receiver shall properly terminate all received SysReq.Event messages by responding with SysRsp.Ok

A first (protocol) timeout value shall be provided by a CSR, the timeout shall be programmable to provide at least 4096 clock cycles of timeout - it may be sufficient to only allow powers of two - this timeout will be used to detect fatal protocol or initialization errors.

A second (handshake) timeout value shall be provided by a CSR, the timeout shall be programmable to provide a smaller range (64 to 256 clock cycles) - this timeout will be used to detect a non-responding target of events.

1.3.2. Event Sender

The sender state-machine will be idle after reset. When the EventInReq is asserted by the source, the state machine will enter the Send state and start sending SysReq.Event messages to all receivers in the system. Maestro shall provide a vector, listing all receivers.

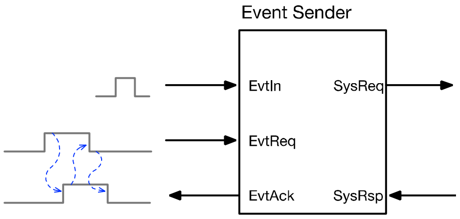


FIGURE 5: EVENT SENDER - INTERFACES

Event messages, like all SysReq transactions, are not credited and a sender must not send more than one transaction to each target agent.

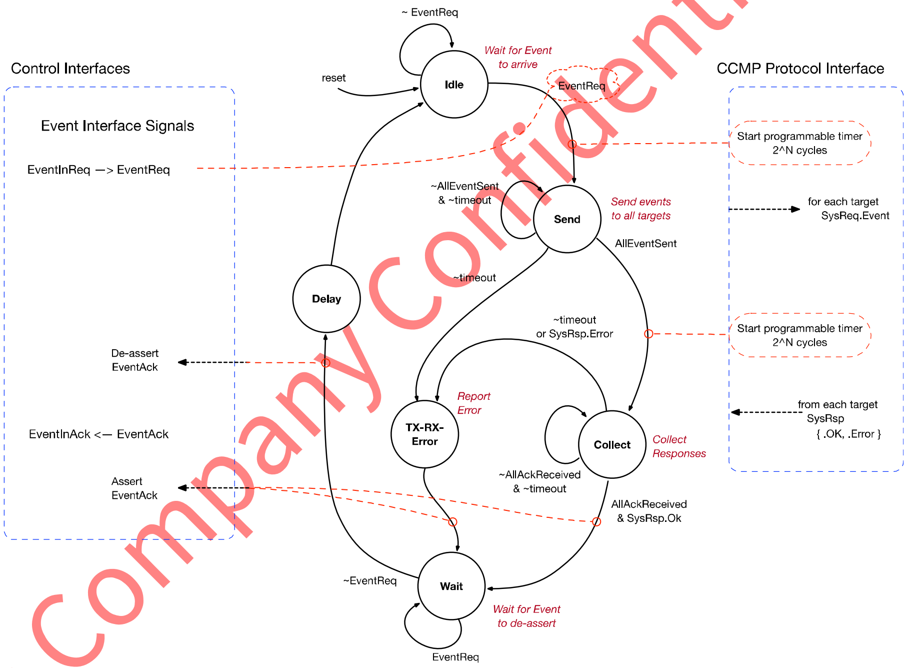


FIGURE 6: EVENT SENDER FSM

After sending out all messages, the sender shall provide means to verify that all messages receive responses - as events will be handled one-at-a-time and no more than one message will be sent to each agent, counting the number of responses is sufficient.

An error is considered:

- not all outbound transactions receive a response within the timeout period
- one or more SysRsp return an error status - the status shall reflect accumulated error from all received responses (most severe error within CMSTATUS -- List to be defined)

1.3.3. Event Receiver

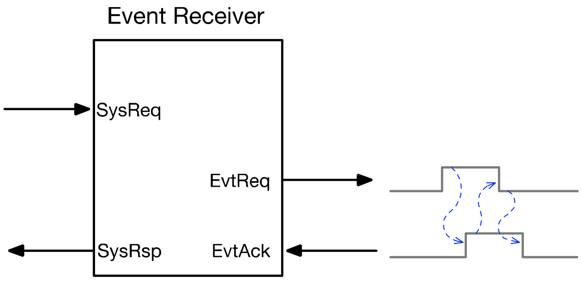
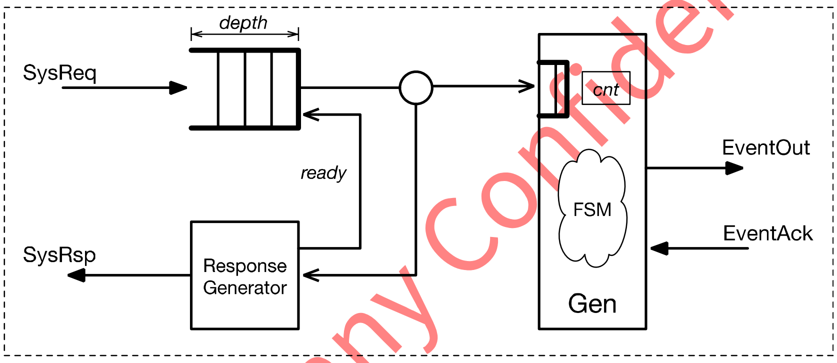


FIGURE 7: EVENT RECEIVER INTERFACE

Whenever a SysReq.Event message arrives, it will be recorded within the input queue. The queue shall provide one dedicated storage location for each source of events. Possible sources of events are: CAIU, NCAIU, DCE, DMI, DII etc.

Events are indistinguishable from each other and may be aggregated - all arriving messages within a certain time period, for example while the interface is occupied with a previous event, may be combined into a single event.



depth - one entry per messaging agent

counter **cnt** increments when a response sent and a FiFo entry has been consumed

FIGURE 8: EVENT RECEIVER ARCHITECTURE

The output of the queue feeds into the event generator and the response generator - every arriving event must be responded to.

If the arriving message does not indicate an error status, the response (order of severity) shall be:

- Ok - if the agent is enabled to receive events (least severe error)
- Busy - if the agent is disabled
- Error - The event generator did not receive EventAck within the timeout period (note, the timeout period for the event handshake may be hard-coded to a significantly smaller value than the protocol timeout)
- Error - The received command message indicated an error (most severe error)

The Event Receiver State Machine, shown in Figure 9, receives arriving event messages and converts them into the 4-phase handshake protocol. Even though the messaging protocol allows fast bursts of event messages to arrive - multiple event messages may trigger only a single event sequence through the state machine.

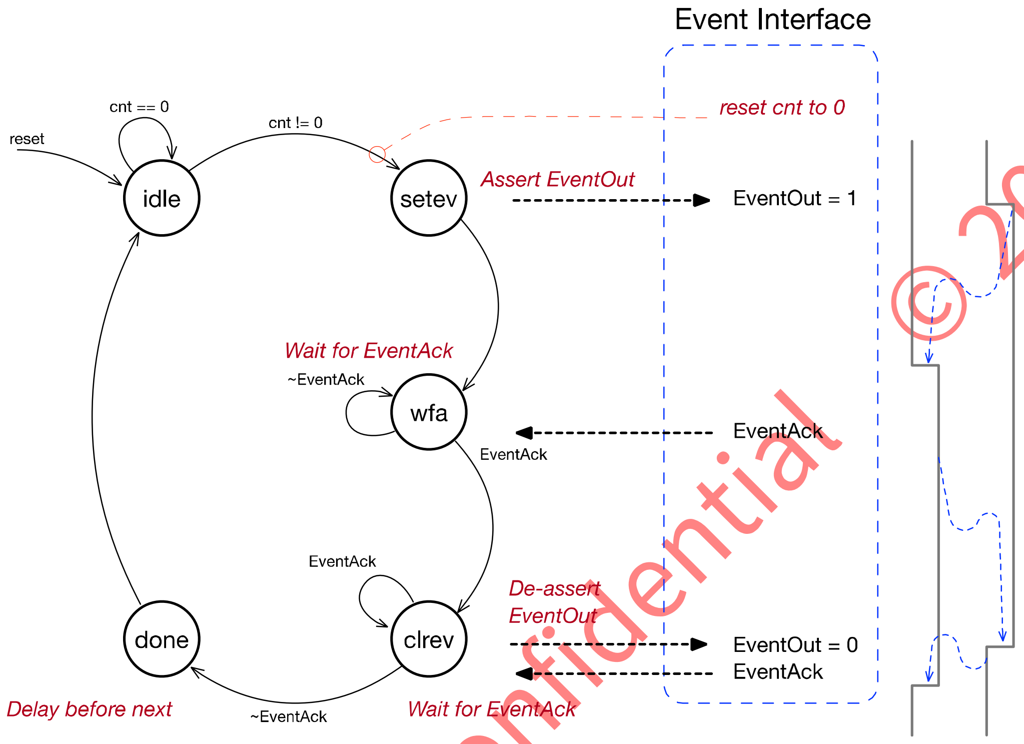


FIGURE 9: EVENT RECEIVER STATE MACHINE

2. *Document References*

2.1. Supported standards and specifications

- Arm CHI A/B/C/D/E
- Arm AXI-4
- Arm v8.1A, v8.2A, v8.3A, v8.4A, v8.5
- Arteris Symphony Concerto-C NCore 3.0

2.2. Standard relevant documents and specifications

ARM DDI 0598A.b Architecture Reference Manual Supplement - Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A

ARM IHI 0050B AMBA® 5 CHI Architecture Specification

ARM IHI 0050C AMBA® 5 CHI Architecture Specification

ARM IHI 0050D AMBA® 5 CHI Architecture Specification

ARM IHI 0050E.a AMBA® 5 CHI Architecture Specification

ARM AES 0003 AMBA® 5 CHI Issue E Specification v7.0 (PDF, obtained by email)

ARM Architecture Specification

3. *Opens*

Future Ideas:

~~state transitions implement local state machine within each agent~~

~~events/requests at the agent interface initiate transitions~~

~~entering certain states will result in communication between agents by sending messages~~

~~message aggregation~~

~~responses to messages~~

~~list source and destinations for messages~~

~~start simple with coherency~~

~~events~~

~~event aggregation and forwarding~~

power messages - shutting down domains - request --> execution --> response --> halted state --> wakeup --> ...

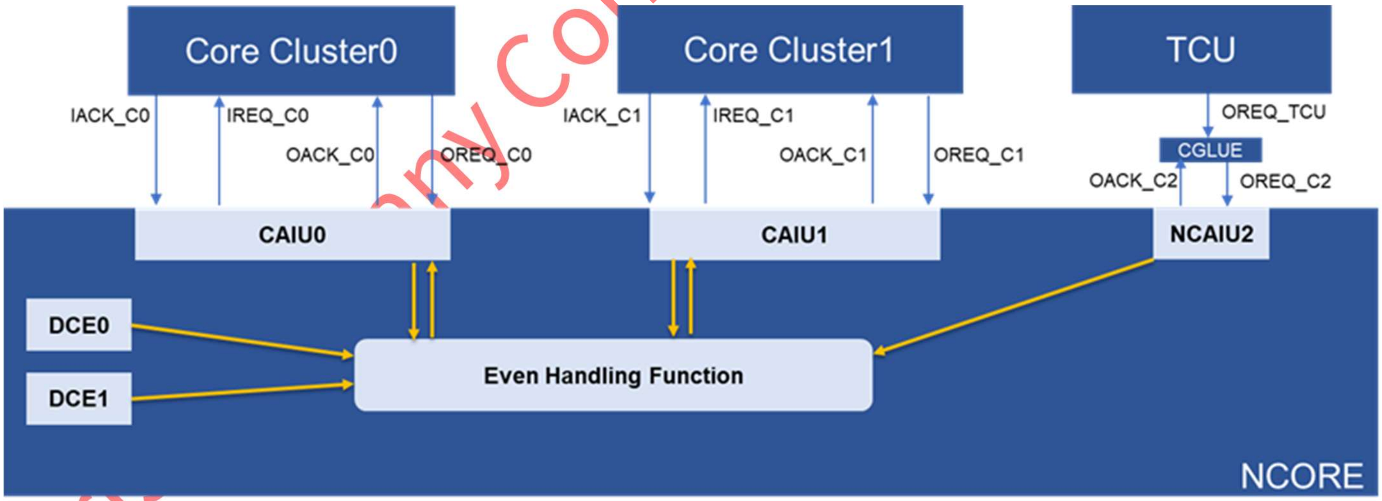
4. *Notes*

Event Messaging and Distribution System

i. explain why we need system messages
ii. state transition within agents need to be communicated
iii. event forwarding - broadcasting
iv. what else?
v. general class of sytem related activities
vi. 1. Event Sources
vii. 2. Event Receivers
viii. 3. Single Ended Event signals (TCU) - cross clock domains, implement glue logic that converts the arriving event pulse into a 4 state handshake compatible with
ix.

So this is what I interpret:

- DCE would generate Event message whenever there is a change in the state of Exclusive monitor to “Open”
- TCU/Clusters would generate Event request on the execution of SEV instruction
- Each Event request would be propagated to all active agents
  - If the Event request was not for the specific cluster, it would go back to sleep after the check (?)
- In case of race, the interconnect should be able to “queue” the event requests from different agents
- No Event request should be “lost”



x.