Ncore 3.4 - SkidBuffer Architecture Specification

Release: 3.4.20

Rev: 0.5, February 2, 2022

ARTERIS® NCORE 3.4 - SKIDBUFFER 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	01/26/2022
0.5	MF	Added details for each item and cleaned up	02/02/2022
Legend:	MK	Mohammed Khaleeluddin	
	MF	Michael Frank	
	JU	Junie Um	
	KJ	Kjeld Svendsen	
	Xx	Whoever else edited this document	

Note:

- The initial description of these modifications has been presented as CCB request on Jan 21, 2022

Issues to be discussed:

- Do we want to apply this mechanism to write data buffers within DMI? In this case we would create a ***nCohWriteBufSiz*** parameter for each DMI and use a configuration register in each DMI to distribute them among the DCE

Product Status

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

Web Address

http://www.arteris.com

Table of Contents

Table of Figures

1. *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.

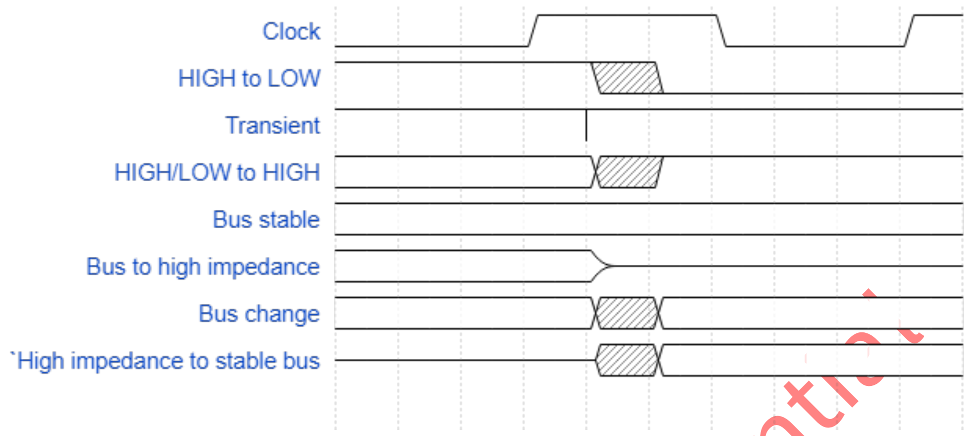
SMALL CAPITALS

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.

Definitions

Flow

Communication between two end points in the protocol. Includes sending a message from the initiator of a transaction (sender), for example an AIU, to the completer of the transaction (receiver), and returning a response back.

Target

The endpoint of a flow, Ncore architecture implements the following targets:

- DCE, DCE - commands from CAIU, NCAIU
- DMI - commands from CAIU, NCAIU and DCE; data from CAIU, NCAIU
- DII - commands & data from CAIU, NCAIU
- CAIU - snoop commands from DCE

2. *Overview*

Credit scheme, buffering etc.

Ncore 3.x architecture uses an end-to-end credited protocol scheme.

Each **credit** represents a free entry at the receiver and a sender needs to have at least one credit available before sending a message.

The receiver in each target will be configured by Maestro to implement sufficient buffering (**skid buffer**) to store messages arriving from all communicating initiators.

The communication requirements are determined by Maestro, based on:

- *Address map support required by the SoC use case*
- *Interleave factor - directories and memories can be interleaved to improve performance*

A more detailed discussion including the definition and manipulation of the interconnect matrix can be found in other documents (System Architecture Specification).

The number of buffers is calculated by Maestro, based on the number of initiators, and for each initiator the number of allowed flows is defined as a design configurable parameter. A single parameter for an initiator defines the number of supported flows between that initiator and **all** targets. Therefore:

- *All skid buffers associated with an initiator have the same depth*

The max. bandwidth available to an agent depends on the

- *Number of transactions it can have in flight*
- *The roundtrip latency for sending the request and receiving a response back*
- *The amount of data moved per transaction request*

The Nocre architecture only supports a single, design time defined parameter for each initiator. The user must determine the number of outstanding transactions based on the longest latency that initiator has to any agent it communicates with. As a result, this

- *Leads to oversizing the total buffer in target (DCE/DMI/DII)*
- *Allocation is fixed at design time – buffer space may be assigned even for agents that will not communicate in the real system (e.g. due to address map configuration)*

Current Skid buffer architecture

Each target implements a skid buffer. This structure consists of a n-entry deep queue to store arriving requests and an arbiter that picks the oldest request, given a certain priority/QoS level. The implementation of an age buffer for the arbitration grows with the square of the number of entries. The data for each request occupies about 145 bits in the queue storage.

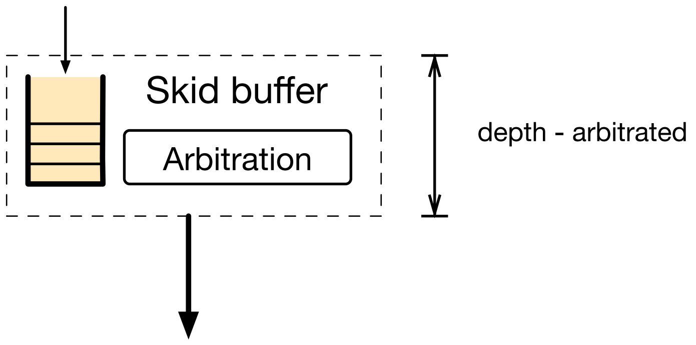


FIGURE 1: SKID BUFFER OVERVIEW

The physical size of the skid buffer is highly dependent on the total number of entries and grows quadratically with the number of entries.

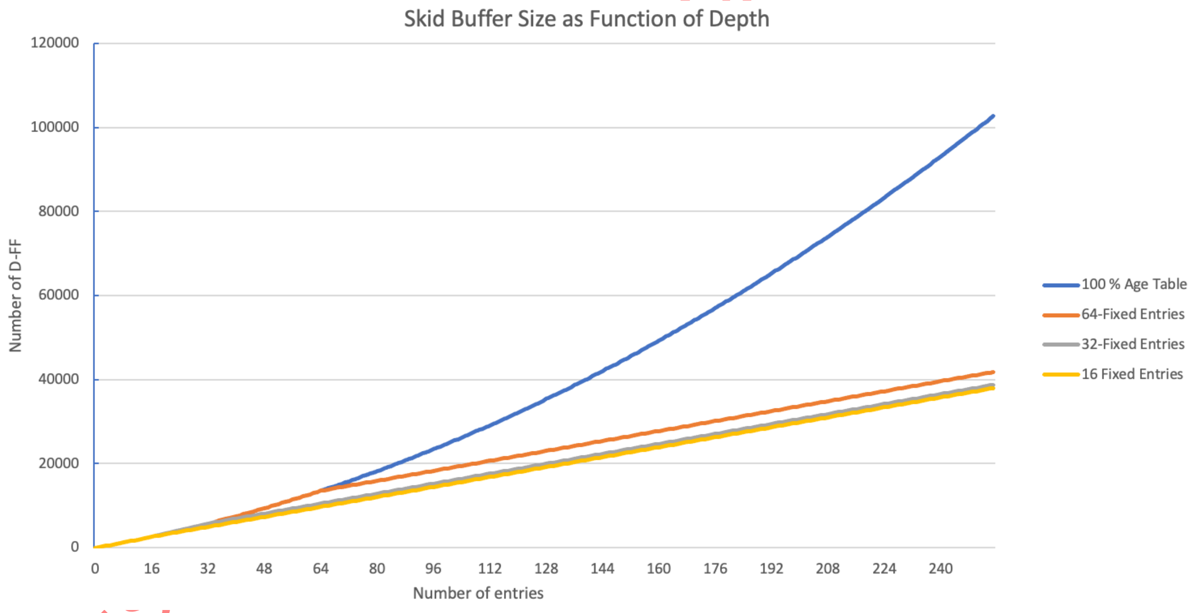


FIGURE 2: SKID BUFFER SIZE AS FUNCTION OF DEPTH

The logic structure to implement the arbitration is growing proportionally to the age buffer size and the complexity leads to significant loss in performance (speed).

As the total useful buffer is determined by an agent's bandwidth and latency – there is no use in frontloading a DMI with 100's of GB worth of traffic – an upper bound of skid buffer size for a target agent can be determined by looking at the total BW supported in steady state operation, the temporal distribution of requests and the number of interconnect buffers in the request path (queueing model interarrival time λ, Little's Law, service time μ).

3. *Modifications*

The following chapters will describe several modifications to the current implementation to address the problem of skid buffer explosion. Each proposal covers a different aspect of the issue and the combination of them will improve the power and area efficiency of Ncore and increase the flexibility of each design for the future.

Command Buffer Partitioning

What issues will be addressed by this change:

1. The current implementation of Ncore implements the command buffers as a single, monolithic queue where arriving command requests are stored as they arrive from initiators and retrieved based on their age and priority
2. The logic structure (age buffer) attached to this queue grows with the square of the number of entries
3. In steady state traffic, the skid buffer will use only some part of the entire structure

Description:

The command buffer is sized to hold '**d**' entries

- Static credit budgeting leads to overprovisioning of buffer space
- The number of credits actively used may be limited by SW scheduling – For some applications, the agents' credit may be adjusted by SW (driver) before kicking off active burst
- For dynamic credit management overflow buffer can be small – just to protect against backup
- The command buffer shall have sufficient depth to store all credited transactions – for example during overlapped activity on multiple agents – without backing up traffic into interconnect (danger of deadlock)
- In steady state operation, the command buffer is only partially filled

Architecture Requirements:

1. The physical implementation of the command buffer shall consist of two parts
   - A fixed size, limited skid buffer - This limits depth of (arbitration) visible entries – only these entries will require arbitration in age-buffer (growing with $O(N^2)$)
   - An overflow buffer to maintain total depth, by adding a FiFo in front of skid buffer – most efficient implementation, only needs D-FF/SRAM for txn-data storage

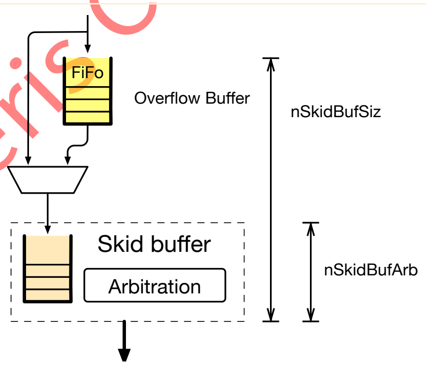


FIGURE 3: SKID BUFFER PARTITIONING

Figure 3 shows how the skid buffer shall be partitioned into an arbitration visible window at the bottom and an overflow buffer at the top. Two parameters are introduced to describe the depth of the entire skid buffer (nSkidBufSiz) and the size of arbitration window (nSkidBufArb).

Software initialization of credit counters

What issues will be addressed by this change:

1. The current implementation of Ncore allocates a fixed amount of buffer space for each flow
2. The allocation is fixed at design time and may not be changed
3. The assignment of credits for an initiator treats all targets to the same amount of credits, independent of differences in round trip latency and BW requirements

Architecture Requirements:

1. The value of credits shall be programmable at each initiator
   a. For each reachable target, a CSR shall provide a field to modify the allocation of credits for a flow
   b. Maestro shall provide path vectors (calculated based on connectivity) to support generation of the required Credit Control Registers:
      - For each CAIU:

Name	**Elements**	**Description**
boolDcePath[i]	# of DCE in system	a boolean, indicating a command path exists from CAIU to DCE[i]
boolDmiPath	# of DMI in system	a boolean, indicating a command path exists from CAIU to DMI[i]
boolDiiPath	# of DII in system	a boolean, indicating a command path exists from CAIU to DII[i]

      - For each NCAIU:

Name	**Elements**	**Description**
boolDmiPath	# of DMI in system	a boolean, indicating a command path exists from CAIU to DMI[i]
boolDiiPath	# of DII in system	a boolean, indicating a command path exists from CAIU to DII[i]

      - For each DCE:

Name	**Elements**	**Description**
boolDmiPath	# of DMI in system	a boolean, indicating a command path exists from CAIU to DMI[i]

   c. The CSR shall provide a means to determine if a communication path between this initiator and a specific target exists
   d. The max. number of credits available for a flow shall be programmable at runtime
2. A mechanism to determine the configured skid buffer size for each target shall be provided by implementing a read-only CSR
3. The initial allocation (at power-on-reset) shall be determined as follows:
   a. If a Boot Region has been configured at an initiator, the non-coherent target defined by the MIG used for the initial boot region shall be initialized to 2 credits
   b. If a Boot Region has been configured at an initiator, the target(s) associated with the system DII shall receive two credits - this needs to consider possible interleaving (unlikely to be used, but possible if DMI used to host the boot region?)
   c. all other credit counts in CAIU/NCAIU/DCE shall receive 0 credits - the boot loader firmware shall determine the available budget for each agent by reading the Skid Buffer Size Register (SBSR) for each
4. This proposal does not apply to credits allocated to snoop transactions or DV messages

Implementation:	

The format of the credit control field shall be 8-bits wide: 3.a

Bit	Type	Counter State	Description
7:5	RO	000: Normal Operation	Counter is operating normally, some outstanding credits
		001: Empty (== 0)	All credits have been used - should be a transient state during which no additional transactions may be sent
		010: Negative (< 0)	All credits have been used - this state would normally be reached when SW reduces the credit limit to a smaller value by an amount that exceeds the currently available credits. This should be a transient state, during which no transactions may be sent. As outstanding credits arrive, the counter will increment through zero to allow requests to resume
		100: Full (== Limit)	No outstanding transactions, all credits are back
		111: No Connection	Special signature to indicate a non-existing connection between this initiator and the target represented by this control field. Maestro implements a connectivity map where some routes may not be required, this code allows firmware to recover the implemented connectivity map between initiators and targets
		Credit Limit	
4:0	RW	0 .. 31	This 5-bit value may be programmed to set the credit limit available for this connection. It is the user firmware's responsibility to set this value correctly, so that the total number of credits assigned between all initiators does not exceed the available buffer space

The credit counters:

The credit counter shall implement a two's complement counter using 6 bits. The counter shall be updated:

- when a credit is consumed - decrement counter
- when a credit is returned by receiving a response - increment counter
- when the CreditLimit is written - add the value calculated by newCreditLimit - oldCreditLimit. This value may be negative if the new value is smaller than the old value, effective decreasing the available credit

All 3 events may occur in the same clock cycle!

At reset the counter shall be cleared to zero or preset to a positive value 5.a

Any counter value ≤ 0 indicates that no credits are available and requests must not be sent

The Credit Control Registers (CCR) in CSR space:

- Ncore architecture may eventually support up to 32 DCE/DMI/DII and we shall plan our register map accordingly
- Each agent will be assigned an ordinal number withing its group (0..31)
- For AIU/NCAIU, each 32-bit Funit.CSR[index] shall pack credit control fields for all agent types (DCE, DMI, DII) using the same index, starting at the least significant byte, for example:

  AIU[x].CCR[0] = {0xE0, DII0.CCF, DMI0.CCF, DCE0.CCF}

  AIU[x].CCR[1] = {0xE0, DII1.CCF, DMI1.CCF, DCE1.CCF]

- For DCE the register layout may be optimized to pack control fields for 4 DMI targets into a single register:

  DCE[x].CCR[0]= {0xE0, 0xE0, DMI1.CCF, DMI0.CCF} - example with 2 DMIs, the other 2 slots indicate that there is no connection to DMI2 and DMI3, should they exist. This scheme supports optimization for interleaved DCE/DMI configurations

- Registers shall be filled, starting at address 0xblah within each unit
- The number of registers created shall be determined by the unit with the highest index
- Partially filled slots in registers shall be initialized with 0xE0, indicating to firmware that either no unit exists with this index or that the connection between this initiator and the target, mapped into this slot, is not implemented (no path exists)

The Skid Buffer Size Info Register (SBSIR) in CSR space:

Each unit implementing a skid buffer at its frontend shall announce the implemented size of the skid buffer in a CSR read-only register or a read-only field in an existing register.

The register shall be using this format:

Bit	Type	Meaning	Description
7:0	RO	SkidBufArb (0 .. 255)	Number of entries visible to arbitration in s split skid buffer configuration, value must be ≤ SkidBufSiz This value is supplied by a new parameter: **nSkidBufArb**
15:8	RZ	Reserved (0)	Keep for future expansion
25:16	RO	SkidBufSize (.. 1023)	Total Size of skid buffer implementation - this value may be identical to SkidBufArb if no split skid buffer has been implemented This value is supplied by a new parameter: **nSkidBufSize**
30:26	RZ	Reserved	Keep for future expansion
31	RO	Valid = 1	Valid bit, indicates presence of the register

The CSR address space for DCE/DMI/DII has an empty slot at 0xFF8 which shall be used to implement this register

Software Support (Maestro)

New parameters are required to configure skid buffer:

Name: nSkidBufSize					**Visibility:** User
	Architecture		**Release**		**Default**
	min.	max.	min.	max.	
Value	4	256	4	256	nSkidBufArb
Constraints	≥ nSkidBufArb				
Customer Description	Total depth of skid buffer for this unit. The skid buffer is used to stage transaction requests from initiator agents. The number of required entries may be determined by traffic requirements and analysis using performance modeling. This value sets the total budget of protocol credits available for distribution.				
Engineering Description	This value sets the total budget of protocol credits available for distribution to communicating initiators. It is recommended to allow at least 2 credits for each active connection. The lower number of entries (4) may apply for a DII where only one or two active flows terminate, all other units may require a minimum of 16 or more.				

Name: nSkidBufArb					**Visibility:** User
	Architecture		**Release**		**Default**
	min.	max.	min.	max.	
Value	4	64	4	64	16
Constraints	≤ nSkidBufSize, restricted granularity of 4, 8, 16, 32, 64 suggested				
Customer Description	Depth of skid buffer visible to arbitration. This value determines the size of the arbitration window within which arriving requests are selected based on QoS, priority and arrival time. It is recommended to start with a reasonably value for performance analysis - the area of a skid buffer grows with the square of this number and larger options will also significantly impact timing				
Engineering Description	This value sets the number of entries within the skid buffer that is visible to arbitration				

UI impact - what does the user see, options

New parameters to configure skid buffer:

- The user interface shall provide a means to define two size parameters, **nSkidBufSiz** and **nSkidBufArb** for each DCE/DMI and DII (command input buffer).
- Each function unit may be assigned a different value for these parameters
- The design time assigned value shall be reflected in the unit's **Skid Buffer Size Info Register**