Ncore 3.8 GIU micro-architecture

Release: 3.8

Rev: 0.3, October 6, 2025

ARTERIS® NCORE 3 SYSTEM ARCHITECTURE C2C **CHIP-LET TO CHIP-LET**

Release Information

Version	*Editor*	*Change*	*Date*
	BM	Added the performance monitor chapter with definition of the events. Update to the Scheduler section.	08/11/2025
0.3	BM	First public release. Added to engineering sharepoint	07/01/2025
0.11	BM	Adds the credit control module	02/27/2025
0.1	BM	Initial version	12/01/2024
Legend:	BM Xx	Benjamin Madon Whoever else edited this document	

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 and verification engineer to implement and verify the Gateway Interface Unit.

Using this document

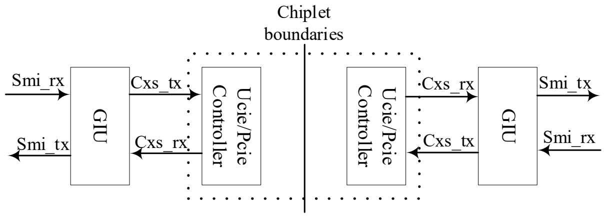
TBD

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.

1 Introduction

The GIU is the unit responsible for formatting ConcertoC messages for their transfer to a different chip. Its role is to convert SMI traffic from Ncore into CXS packets which is fed into the PHY controller. It implements the functionality described in the C2C architecture document. The following figure illustrates how an incoming smi message to a giu from one chiplet is converted into an outgoing smi message from another chiplet.



This micro-architecture intends to describe the hardware implementation of the unit.

2 Terminology

We will use the following terms:

- A **flit** will be reserved for CXS flits
- A **container** is the 256 bytes packing defined in the architecture specification.
- A **granule** is the subdivision of the payload inside the container as defined in the architecture specification. It is 10 Bytes.
- 64 bytes of payload inside the GIU will be referred to as a **beat** of data.
- The term **packet** is used to represent the transfer of a container. It will be used interchangeably for CXS and for internal interfaces.

3 Parameters

Name	Type	Visibility	Min	Max	Default	Description
VC_descriptor.nVC	int	Engineering	x	x	4	Number of VC for the C2C link
VC_descriptor.wCredit	int	Engineering	x	x	5	Width of the credit counters. Max number of credit is 2^5=31

Name	Type	Visibility	Min	Max	Default	Description
VC_descriptor.NumberOfCredit	Array[int]	User	[4,4,4,4]	[31,31,31,31]	[15,15,15,15]	VC_descriptor.NumberOfCredit[vc] is the number of credit for VC vc
packet_descriptor.SMI_ndp_packing_order	Array[str]	Engineering	x	x	["msg_type","targ_id","src_id","ndp_len","msg_id","ndp","msg_user","dp_present"]	Smi ndp packet ordering
packet_descriptor.SMI_dp_packing_order	Array[str]	Engineering	x	x	["data","user"]	SMI dp packet ordering
packet_descriptor.NumberOfGranulePerBeat	int	Engineering	x	x	6	Number of granule in a 64B beat
packet_descriptor.GranuleSizeInBytes	int	Engineering	x	x	10	Size of a granule in bytes
packet_descriptor.StartBits	Array[array[int]]	Engineering	x	x	[[511,510,509,508,507,506],[15,14,13,12,11,10],[511,510,509,508,507,506],[15,14,13,12,11,10]]	packet_descriptor.StartBits[b][gran] is the bit location of the start field for granule gran in beat b
packet_descriptor.PayloadBits	Array[int]	Engineering	x	x	[16,16,16,16]	packet_descriptor.PayloadBits[b] is the bit location at which the payload start in beat b

Name	Type	Visibility	Min	Max	Default	Description
packet_descriptor. CreditreturnBits	Array[Array[int]]	Engineering	x	x	[[0,496,3],[0,21,3],[1,0,3],[1,5,3]]	Packet_descriptor. CreditReturnBits[vc][0] is the beat in which credit of VC vc are returned and Packet_descriptor. CreditReturnBits[vc][1] is the bit location for the start of the field. Packet_descriptor. CreditReturnBits[vc][2] is the width of the field. The value of 3 means maximum 7 credit can be returned in a flit.
packet_descriptor. NumberOfBeatPerContainer	Integer	Engineering	4	4	4	This is the number of beats in a container.
InterleaveInfo	Object	Engineering	x	x	x	Object that contains the interleaving of DMIs and DCEs
SystemParams.engVerId	int	Engineering				Version id
SystemParams.implVerId	int	Engineering				Implementation id
SystemParams.nDces	int	Engineering				Number of DCE on the chiplet
SystemParams.nDve	int	Engineering	1	1	1	Number of DVE on the chiplet
SystemParams.nDmis	int	Engineering				Number of DMI on the chiplet
SystemParams.nDiis	int	Engineering				Number of DII on the chiplet
SystemParams. wFUnitId	int	Engineering				Width of the FUnitId
SystemParams. wFPortId		Engineering				Width of the portId

Name	Type	Visibility	Min	Max	Default	Description
SystemParams. FPortId		Engineering				?
SystemParams.nRegion		User				Number of GP region
SystemParams.BootInfo		User				Boot region information
SystemParams.CsrInfo		User				CSR region information
interfaces		Engineering				Top level interfaces to the GIU

4 Interfaces

4.1 External interfaces

Name	CPR	Direction	Condition	Description
clk	InterfaceClk	Slave	True	Contains clock and active low reset
checkClk	InterfaceClk	Slave	UseResiliency	Contains clock and active low reset for the duplicate unit
masterTrigger	InterfaceMasterTrigger	Slave	True	Contains the master trigger for the capture
SMI	InterfaceSMI	Master(TX)/ slave(RX)	True	Symphony Message Interface. It receives/transfers the message from the Ncore side
cxsTxInt	InterfaceCXS	Master (TX)	True	Standard ARM Interface to the PHY controller
cxsRxInt	InterfaceCXS	Slave (RX)	True	Standard ARM Interface to the PHY controller
APB	InterfaceAPB	Slave	True	Used To access control and status registers
Irq	InterfaceIRQ	Master	True	Interrupt interface
uId		Slave	True	Static input that contains the unit IDs

Name	CPR	Direction	Condition	Description
uChipletIdInt	InterfaceChipletId	Slave	True	Static input that contains ChipletId and the AssemblyId.
QInt	InterfaceQChannel	Slave	UsePma	Used for power management handshakes
BIST	InterfaceBIST	Slave	UseResiliency	Contains the BIST control signal
bistDebugDisableInt	InterfacePin	Slave		
Fault	InterfaceFault	Master	UseResiliency	Contains the signal sent to the fault controller as well as the late_clk.
User Placeholder	InterfaceGeneric	Master	UseResiliency	Contains user defined signaling to the native placeholder protection block

TABLE 4-1 TOP LEVEL INTERFACES

4.2 Messages mapped to SMI

The mapping below assumes a 4CN/1DN configuration. Messages will be collapsed based on system specifications if an Ncore is configured with fewer CNs.

Interface	Direction	Request	Response
SMI 0 Non-Data (CN0)	RX	CmdReq,StrReq,UpdReq, SysReq	
SMI 0 Non-Data (CN0)	TX	CmdReq,StrReq,UpdReq, SysReq	
SMI 1 Non-Data (CN1)	RX		StrRsp, CmdRsp, UpdRsp, SysRsp
SMI 1 Non-Data (CN1)	TX		StrRsp, CmdRsp, UpdRsp, SysRsp
SMI 2 Non-Data (CN3)	RX		DtwRsp, DtrRsp
SMI 2 Non-Data (CN3)	TX		DtwRsp, DtrRsp
SMI 3 Data (DN)	RX	DtwReq, DtrReq	
SMI 3 Data (DN)	TX	DtwReq, DtrReq	

TABLE 4-2 CONCERTO MESSAGES/ NETWORK DISTRIBUTION

5 Registers

Register Name	Register offset	condition	Description
GIUUIDR	x0	1	GIU Identifcation register. This registers provide informationabout the Ids that belong to this GIU.
GIUUFUIDR	0x4	1	GIU Fabric Unit Identification Register
GIULI	0x8	1	GIU LinkId register
GIUCXSL	0x40	1	GIU CXS Link Register
GIUUTAR	x44	1	GIU Transaction Activity Register. This registers allows software to check if this GIU is currently in use.
GIUUEDR	0x100	1	GIU Uncorrectable Error Detect Register. This register is used to select which type of uncorrectable error are reported
GIUUUEIR	0x104	1	GIU Uncorrectable Error Interrupt Register. This register is sued to select which type of error will trigger an uncorrectable error interrupt.
GIUUUESR	0x108	1	GIU Uncorrectable error status register. This register is used to report uncorrectable errors
GIUUELR0/1	0x10c/0x110	1	GIU Uncorrectable Error Location Registers. Those register are used to give information about the location of an uncorrectable error.
GIUUESAR	0x114	1	GIU Uncorrectable Error Status Alias Register
GIUCECR	0x140	1	GIU Correctable Error Control Register
GIUCESR	0x144	1	GIU Correctable Error Status Register
GIUCESAR	0x150	1	GIU Correctable Error Status Alias Register
GIUCRTR	0x180	1	GIU Correctable Resiliency Threshold Register
GIUQOSCR	0x200	1	
NRSBAR	0x380	1	NRSBAR/NRSBHR identifies the base address of the NRS
NRSBHR	0x384	1	NRSBHR is used to stage the writing of the Base address of the NRS

NRSBLR	0x388	1	NRSBLR is used to load the NRSBAR from NRSBHR
DMIAMIGR	0x3c0	1	Active Memory Interleave Group Register
DMIMIFSR	0x3c4	1	Memory interleave Function Select registers
GIUBRAR	0x390	1	Boot region attribute register
GIUBRBLR/GIUBRBHR	0x394/0x398	1	Boot region address register
GIUCNTCR	0xb00	derived.nPerfCounters(+0x10)	PMON Counter Control Register
GIUCNTVR	0xb04	derived.nPerfCounters(+0x10)	PMON Counter Value Register
GIUCNTSR	0xb08	derived.nPerfCounters(+0x10)	PMON Counter Saturation Register
GIUBCNTFR	0xbc0	derived.nPerfCounters(+0x10)	PMON BandWidth Counter Filter Register
GIUMCNTCR	0xc80	derived.nPerfCounters?	PMON Main Counter Control Register
GIULCNTCR	0xc84	derived.nPerfCounters?	PMON Latency Counter Control Register
GIUUEVIR	0xff4	1	GIU Unit Engineering Version Id Register
GIUINFOR	0xffc	1	GIU Information Register

6 Latency

Commented [RS1]: If these values are coming from the uArch, I think its better to report them after explaining the uArch details.

Path	Target Latency	Detail
xReq -> CXS packet	5 cycles	
CXS packet -> xREq	5 cycles	

TABLE 6-1 LATENCY TABLE

7 Memories

There are no memories in the GIU.

8 Design description

8.1 Top level view

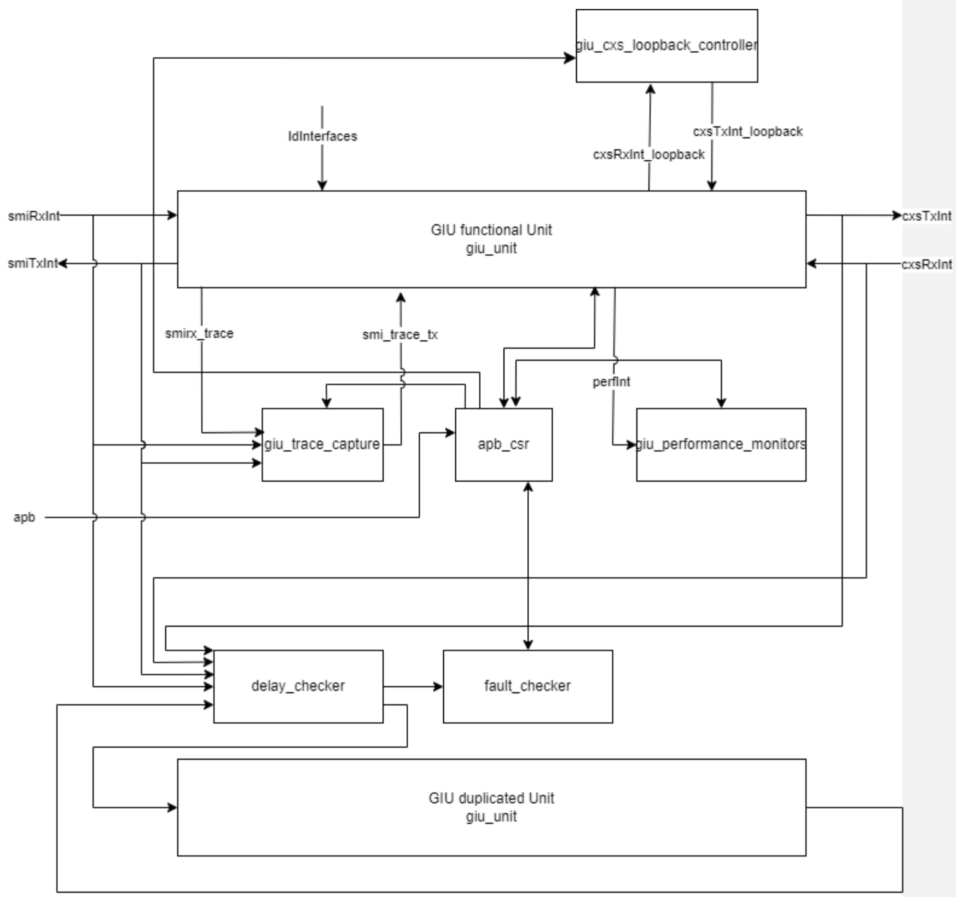


FIGURE 8-1 TOP LEVEL DIAGRAM

The top level of the GIU will contain the functional unit which role is to packetize SMI messages into CXS messages and operate the translation required by the protocol. Trace capture and performance monitor block are kept at the top level as they do not participate in the main functionality of the GIU. Memories are also kept at the top level to be uniform with the rest of the Ncore unit.

8.2 GIU functional unit

The functional unit is composed of the following main functionalities:

- The packetizer block will pack the SMI messages based on the packing scheme described in the architecture document. Its output is a ready valid data bus. It will contain storage to optimize the packing. This unit will be written as a library element and is meant to be flexible in its use (in particular it should be able to create CHI-c2c packet out of CHI-C2C input channels with minor modification).
- The depacketizer will decode the packet into SMI messages.
- VC credit counter keeps track of the number of outstanding per virtual channel.
- The CXS controller maintains the CXS link including link credits, activation and deactivation of the link. Its input is a ready/valid/last interface of 64 byte, the entire data is passed over to the data channel of the CXS link.
- SMIx_processing will update the TargetId field for CmdReq and won't do anything for other message. Consequently, only SMI0_processing will contain logic.
- The giu_smi_rx2_demux decodes the type of message. If it is a dtw_dg_rsp message it sends it to the trace capture block, otherwise it goes through normal D2D processing.

Each block will have its own clock gator using a separate enable.

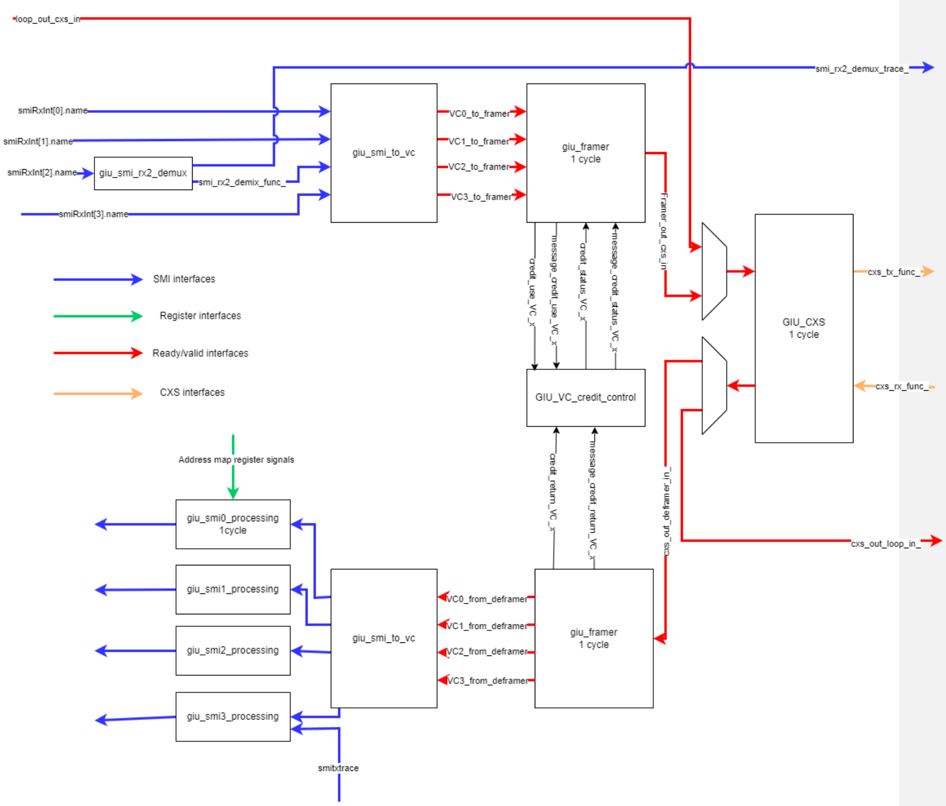


FIGURE 8-2 GIU FUNCTIONAL UNIT DIAGRAM

8.2.1 SMI to VC

This module is responsible for assigning each SMI message to a VC and to pad them such that the resulting interface is an integer number of granules.

Additionally, at the beginning of each message it will add a VC field which will only be used for GIU to GIU communications. It will encode the VC that this message belongs to and will be used to facilitate the decoding of the packet on the receiving end.

The input to this module is SMI and the output is a ready-valid interface with width equal to the message length for this particular VC.

The order used is described by packet_descriptor. SMI_ndp_packing_order and packet_descriptor. SMI_dp_packing_order arrays. The VC field wich is 2 bits will always be first, followed by ndp fields and dp field using javascript indexing order.

One such module is instantiated per VC.

8.2.2 VC to SMI

Its function is the reverse of SMI to VC.

Commented [BM2]: More details to come but that should be sufficient for now.

8.2.3 SMI0_processing

8.2.3.1 Parameters

8.2.3.2 Interfaces

8.2.3.3 Functional diagram

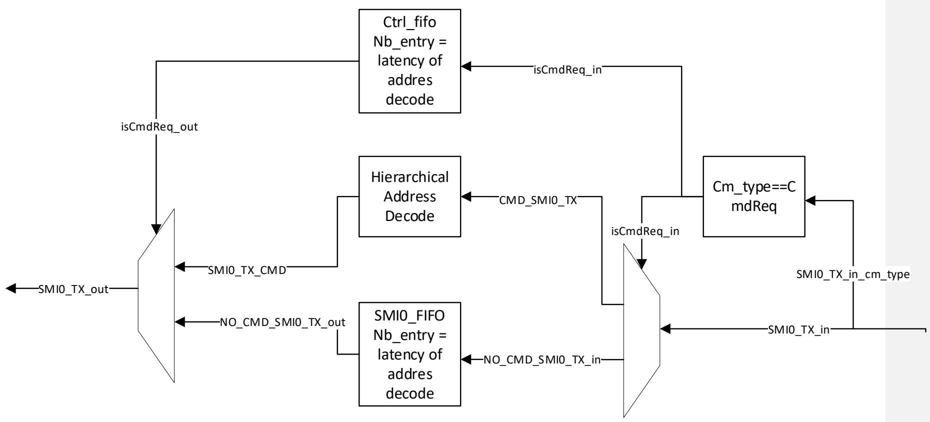
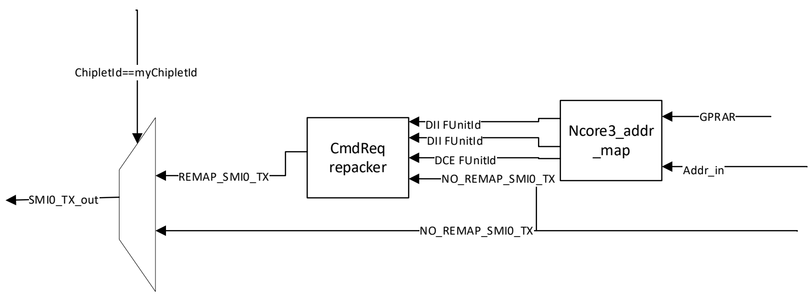


TABLE 8-1 SMI0_PROCESSING FUNCTIONAL DIAGRAM

8.2.4 Hierarchical address decode



8.2.5 Framer

The packetize module is meant to be a flexible module which could be reused to packetize CHI-C2C messages. Consequently, its IOs are generic, and it will take several parameters to configure it.

8.2.5.1 Parameters

Name	Type	Description
nVC	integer	This is the number of VC; it corresponds to the number of VC used for the C2C link. Each channel, will have its own interface
interfaces	object	Contains all the interfaces used for IOs
NGperFlit	integer	Number of granules contained per 64B.
GranuleSizeInBytes	integer	Number of bytes in a granule. (10 in our case)
MessageSizeInGranule	Array[integer]	This is the number of granule used for each VC. The size of the array is nVC.
NumberOfCredit	Array[integer]	The number of credit for each VC.
NumberOfBeatPerPacket	integer	This is the number of 64B beat per packet. (4 in our case)
CreditBytes	Array[Array[integer]]	This is the location within the packet of the credit for each VC. For example CreditBytes[0][:] are the bits used for the credits of VC0.
StartBits	Array[integer]	Location at which the start vector starts for each beat within 64B subdivision. For example, StartBits[0] indicate the location at which the start vector begins.

For Ncore 3.8 which tunnels CCMP messages, the parameters will be as follows :

Name	Value	Description
nChannel	5	Number of VC which the module need to handle
Chan_interface_0	SMI0	Interface signal bundle for VC0
N_granule_0	3	Number of granule of a message in VC0
Chan_interface_1	SMI1	Interface signal bundle for VC1
N_granule_1	3	Number of granule of a message in VC1
Chan_interface_2	SMI2	Interface signal bundle for VC2
N_granule_2	2	Number of granule of a message in VC2
Chan_interface_3	SMI3	Interface signal bundle for VC3
N_granule_3	9	Number of granule of a message in VC3
Chan_interface_4	SYST	Interface signal bundle for VC4
N_granule_4	2	Number of granule of a message in VC4
GranuleSize	10	Number of Byte per granule
DATAInterface	{valid :1, ready : -1, last:1, data : 512}	Interface signal bundle of the data interface
PacketSize	256	Number of Byte per packet.

Name	Value	Description
LinkBytes	[0,1,63,64]	Byte which can't be used in the flit for the data payload in 64B granularity.
Credits	{VC0:"{flit:1,bits : "[2:0]"}, VC1:"{flit:1,bits : "[7:5]"}, VC2:"{flit:3,bits : "[2:0]"}, VC3:"{flit:3,bits : "[7:5]"} }	Location of the bits used for credit return in each
StartBits	{Flit0:[63,62,61,60,59,58], Flit1: [15,14,13,12,11,10], Flit2: [63,62,61,60,59,58], Flit3: [15,14,13,12,11,10]}	

8.2.5.2 Interfaces

Name	Object Name	Direction	Description
clk	InterfaceClk	Slave	Contains clock and active low reset
SMI0	SMI0	Slave	Contains valid, ready and an SMI0 message (CN0)
SMI1	SMI1	Slave	Contains valid, ready and an SMI1 message (CN1)
SMI2	SMI2	Slave	Contains valid, ready and an SMI2 message (CN3)
SMI3	SMI3	Slave	Contains valid, ready and an SM3 message (DN
SYST	SYST	Slave	Contains valid, ready and a SYST packet.
DATA	DATAInterface	Master	Contains a valid, a ready, a last and 64B payload. Last assert at the end of the flit.
Fault	ProtectionInterface	Master	Goes to the fault controller

8.2.5.3 Functional diagram

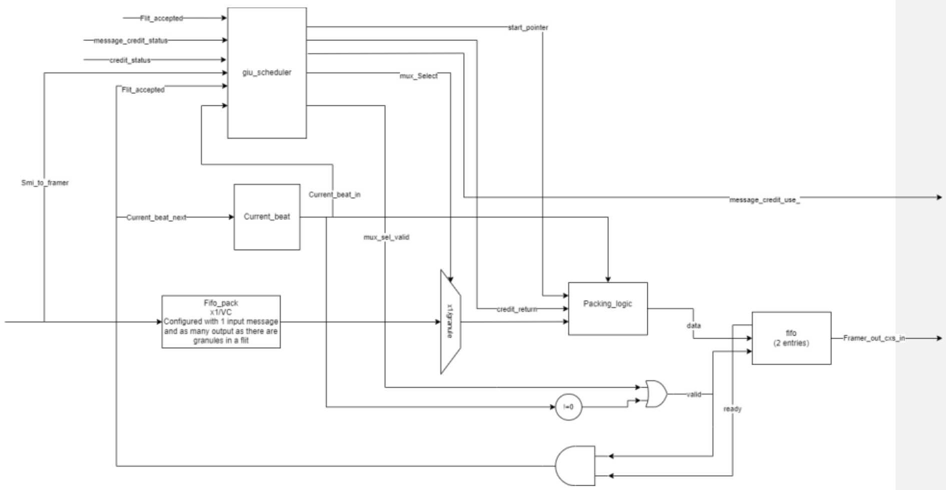


FIGURE 8-3 FRAMER SCHEMATIC

The packetization has five main functional elements:

- An n input m output fifo where n is the number of granules for the message being stored and m is the maximum number of granule which can be retrieved from the fifo. The storage necessary in those fifo is to be studied using modeling. This should be designed as a library element. On the input side it will either push all or none of the granule to the fifo.
- Multiplexing to create the packet from the granule available inside the fifo
- A scheduler which selects the messages to send based on availability of message and credits and granule carried over from the previous cycle. It is also responsible to return the credits to the other side.
- The packing logic puts the granule, start pointers and credit return in the correct location to form a 64B flit.
- An output fifo which is present strictly for timing and may be rendered optional.

This micro-architecture should be easily modifiable to accommodate a future CHI-C2C implementation.

8.2.5.4 Giu_scheduler

The giu_scheduler implements a large case statement which selects which message need to be sent based on fifo occupation, the number of granule available in the current cycle and the number of credit.

It also calculates how many granules will spill over to the next flit to use in the next cycle to calculate how many granule are available for new messages.

Currently the algorithm takes in a list of priority per vc and sends as many messages as it can for the highest priority then, to the next etc…

The number of granule used for each message class, the number of credit for each VC, the order of priority and the maximum number of message for each VC in a flit are passed down by maestro.

8.2.6 Deframer

The depacketize module will operate the reverse operation to the packetize one consequently it will take the same parameters in.

8.2.6.1 Parameters

Name	Type	Description
nVC	integer	This is the number of VC; it corresponds to the number of VC used for the C2C link. Each channel, will have its own interface
interfaces	object	Contains all the interfaces used for IOs
NGperFlit	integer	Number of granules contained per 64B.
GranuleSizeInBytes	integer	Number of bytes in a granule. (10 in our case)
MessageSizeInGranule	Array[integer]	This is the number of granule used for each VC. The size of the array is nVC.
NumberOfCredit	Array[integer]	The number of credit for each VC.
NumberOfBeatPerPacket	integer	This is the number of 64B beat per packet. (4 in our case)
CreditBytes	Array[Array[integer]]	This is the location within the packet of the credit for each VC. For example CreditBytes[0][:] are the bits used for the credits of VC0.
StartBits	Array[integer]	Location at which the start vector starts for each beat within 64B subdivision. For example, StartBits[0] indicate the location at which the start vector begins. The first pointer correspond to the last granule

8.2.6.2 *Functional diagram*

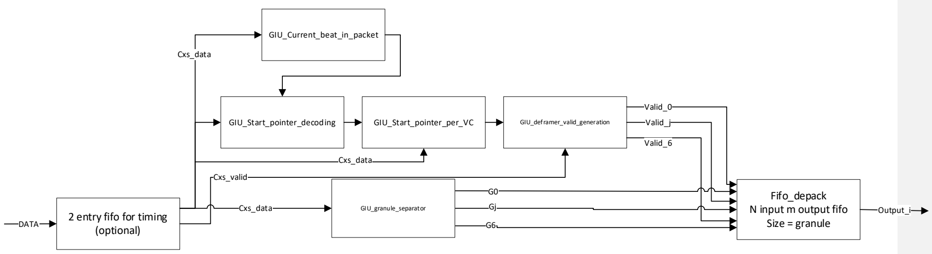


FIGURE 8-4 DEPACKETIZE SCHEMATIC

8.2.7 CXS controller

8.2.7.1 *Description*

This module is responsible for:

- Generating valid CXS packet from a ready/valid interface
- Handle the link layer credit of the CXS interface
- Handle the Initialization of the CXS interface.

The initialization routine and the acceptable CXS packet will need to be parametrizable depending on which controller will be connected to the unit.

8.2.8 VC credit control

8.2.8.1 *Description*

This unit keeps track of the link credit required to exchange messages between two GIUs. It keeps track of two things :

- The number of credit that need to be sent to the other GIU.
- The number of credit available to send messages.

Those are counted per virtual channel. The width of these counters is controlled by a fixed parameter called VC_descriptor.wCounter.

8.2.8.2 Parameters

Name	Type	Description
interfaces	object	Contains all the interfaces used for IOs
VC_descriptor	object	This object is propagated from previous level in the hierarchy
packet_descriptor	object	This object is propagated from previous level in the hierarchy

8.2.8.3 Interfaces

Name	Object Name	Direction	Description
clkInt	InterfaceClk	Slave	Contains clock and active low reset
Message_credit_return_VC_\=vc=_	InterfaceGenVld	Slave	Contains valid, and data field indicates the number of credits were returned by the GIU on the other side of the link. There is one such interface per VC.

Name	Object Name	Direction	Description
Message_credit_use_VC_\=vc=_	InterfaceGenVld	Slave	Contains valid and the data field indicates how many credits were used to send a message to the other GIU. There is one such interface per VC.
Message_credit_status_VC_\=vc=	InterfaceGenData	master	Contains the number of credits which can be used to send a message to the other GIU. There is one such interface per VC.
Credit_return _VC_\=vc=	InterfaceGenVld	Slave	Contains valid, and the data field contains the number of credits which can be sent back to the other side (will always be 1). There is one such interface per VC.
Credit_use_VC_\=vc=	InterfaceGenVld	Slave	Contains a valid and the data field contains the number of credit that were sent to the GIU on the other side of the link. There is one such interface per VC.

Name	Object Name	Direction	Description
Credit_status_VC_\=vc=	InterfaceGenData	Master	Contains the number of credit available to be sent to the GIU ont eh other side of the link. There is one such interface per VC.

8.2.8.4 *Functional diagram*

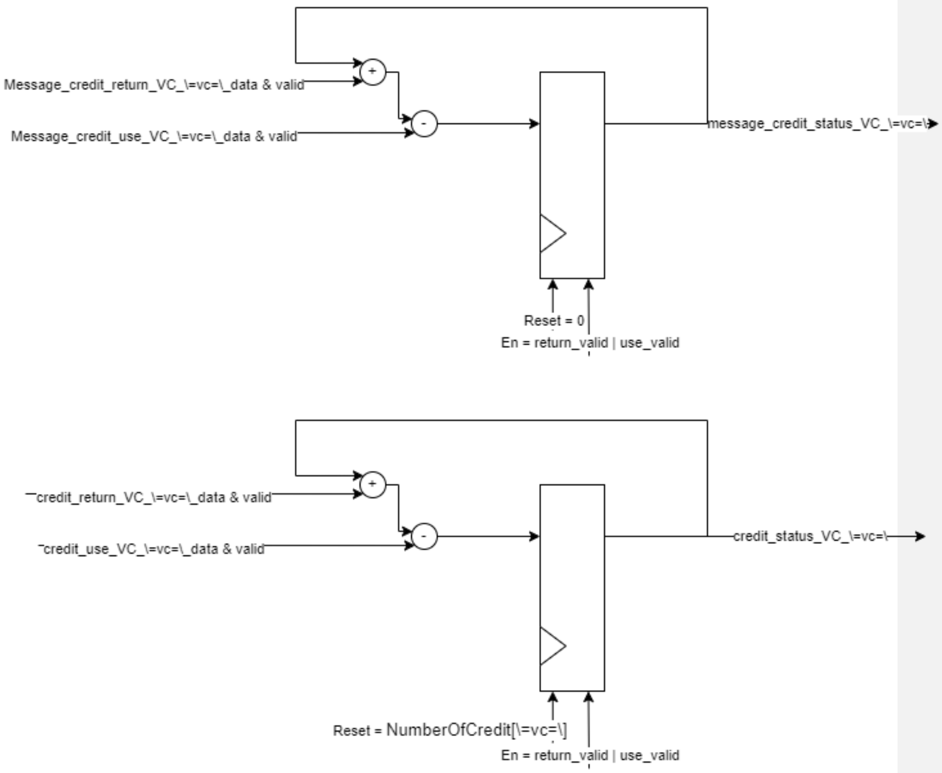


FIGURE 8-5 FUNCTIONAL DIAGRAM OF THE VC CREDIT CONTROL BLOCK

The interfaces prefixed message_credit_ refer to the credit that allow the GIU to send messages to the GIU on the other side of the link.

The interfaces prefixed credit_ refer to the credit that the GIU needs to provide to the GIU on the other side of the link.

The words "used" and "return" refer to the action on the counter. The respectively correspond to incrementing and decrementing the counter. The count is only activated when the valid of the interface is set.

The message_credit_return comes from the deframer. It will contain credit sent by the GIU on the other side of the link.

The message_credit_use comes from the scheduler in the framer. It tells the credit counter the arbitration decision made by the scheduler.

The message_credit_status is provided to the scheduler in the framer. It allows the framer to make packing decisions.

The credit_return interface comes from the deframer when a message is removed from the corresponding skid buffer.

The credit_use interface comes from the scheduler in the framer. It tells the credit counter how many credits were sent to GIU on the other side of the link.

The credit_status interface is provided to the framer. It allows sending credit to the GIU on the other side of the link.

8.3 Performance monitoring

GIU uses the same registers and hardware block as other Ncore units for the performance monitoring.

The list of event are as flows :

Event index	Event description
0	CXS TX flit valid : event index 0
1	CXS RX flit valid : event index 1
2	SMI0 message received
3	SMI1 message received
4	SMI2 message reveived
5	SMI3 message received
6	SMI0 message sent
7	SMI1 message sent
8	SMI2 message sent
9	SMI3 message sent
10	SMI0 not ready to receive
11	SMI1 not ready to receive
12	SMI2 not ready to receive
13	SMI3 not ready to receive
14	SMI0 not ready to send
15	SMI1 not ready to send
16	SMI2 not ready to send
17	SMI3 not ready to send
18	No valid flit on CXS TX interface

19	Not valid on CXS RX interface
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Clock divided by 16
30	Reserved
31	Reserved

TABLE 8-2 PERFORMANCE EVENT DESCRIPTION