CFG SCF NocStudio Addendum

Version: 19.07L

Revision: 0.0

CFG SCF NocStudio Addendum

About this document

This document describes the architecture of the SCF part of NocStudio. This includes details of the IP components and instructions on how to add the SCF components bridges and how to configure and use it.

Audience

This document is intended for users of NocStudio:

- NoC architects, designers and verification engineers
- SoC architects, designers and verification engineers

Prerequisites

Before proceeding, you should generally understand:

- Basics of Network on Chip technology
- Basic knowledge of Scalable Coherent Fabric (SCF)

Customer support

For technical support about this product and general information, contact CFG Support.

Revision History

Revision	Date	Updates
0.0	Jul 19, 2019	Initial Version

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 SCF OVERVIEW

Scalable Coherent Fabric (SCF) is generally used in server mesh designs. The SCF fabric comprises of a connected ring based design. These rings can be both, Horizontal and Vertical.

The figure below shows a sample SCF Fabric design in NocStudio.

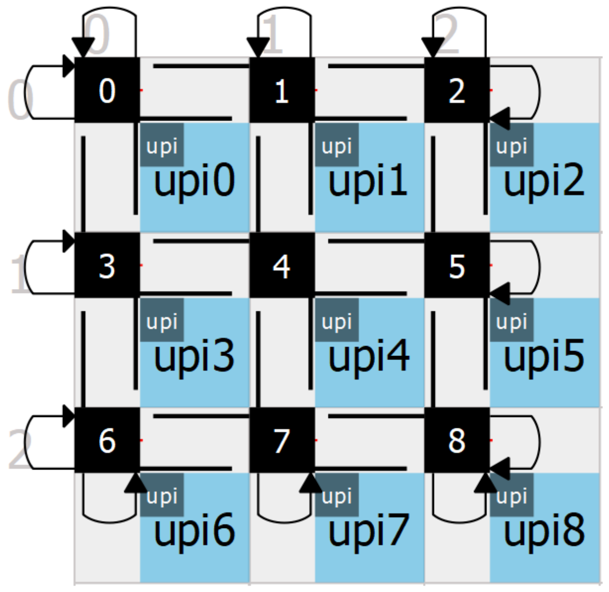

Figure 1 General SCF Mesh View in NocStudio

1.1 SCF COMPONENTS

1.1.1 Common Mesh Stop (CMS)

The Common Mesh Stops (CMSs), are the numbered black boxes represented in the figure above. They are Arbitration points in the fabric, which are connected in a full mesh fashion. The CMS consists of direction ports and add/drop interfaces. Agents connect to the CMS through the Add/Drop ports. Each CMS has 2 add/drop ports. In addition to these add/drop ports, CMS has directional ports to connect to the adjacent ring stops. A Transgress buffer (TG) is present inside the CMS which holds flits that are changing direction Y to X.

The SCF Models follow a strict "Y-X" routing. However, an agent inside the CMS to help and mimic X-Y routing.

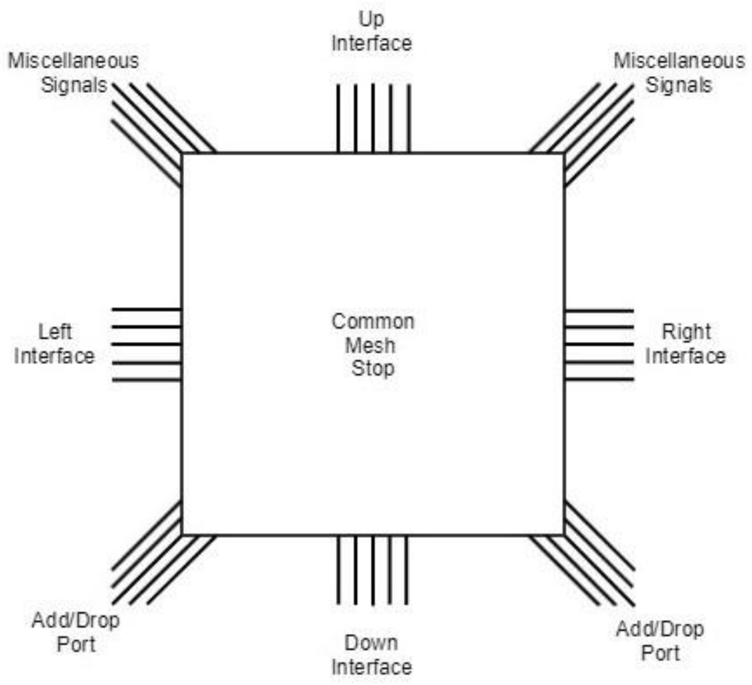

Figure 2 CMS Interfaces

The interfaces of the CMSs are named as shown in the figure above. The Up, Down, Left and Right interfaces of the CMSs are used to communicate with the adjacent CMSs, vertically and horizontally.

There are interfaces at the corners of the CMS as shown. The corner interfaces on the lower side of the CMS are dedicated connections to the Add or Drop ports of the host block. There can be one or two add and drop interfaces, based on the host connected to the CMS.

The interfaces present in the upper corners of the CMS are for miscellaneous signals. These signals can be clock, resets, Jtags etc.

There are 3 types of CMSs.

1. CHA CMS
2. CMS NCAP and
3. CMS TALL

1.1.2 Transgress Buffer (TG)

Transgress Buffers are the buffers used to move a flit from Y direction to X direction or to drop a flit off from Y direction to a drop port, in a CMS. TG has an ingress buffer and an egress buffer. The ingress buffer is split internally to accommodate different classes and types of messages (both bounce-able and credited) separately to avoid deadlocks.

In some cases, Dual ported ingress buffers help the transgress buffer to accept one message from either directions in each cycle, and send the message in each direction, going to a separate port.

1.1.3 Turn Agent (TA)

In addition to Transgress Buffers, the CMS has another logic inside it known as Turn Agent(TA). This TA is designed to accept a packet from the horizontal ring at a CMS and inject it vertically, so that a Y-X routing effectively becomes X-Y routing. This Turn Agent muxes to one of the add-drop ports and has its own buffer and credits.

1.1.4 Message Types

The SCF model has various message and interfaces types. These message types are as follows:

Table 1 SCF Message Types

Message Type	Acronym	Use
ad	Address Ring	Command Request
bl	Block (Data) Ring	Data Message
ak	Acknowledgement Ring	Response/Go Message
akc	AK Credit Ring	End-to-End Credit Return
ak2/iv	Invalidate (Core Snoop) Ring	Snoops
tgc	Transgress Credit Return	Additional Credit Return

Each and every message type has its own set of wires to communicate. These set of wires are represented as layers in NocStudio. Among these message types; ad, bl, ak and akc are single flit packets. In some cases, iv or ak2, is a two flit packet and requires both the flits to be sent back-to-back. Each BL message is 32 byte wide. But BL messages are sent as 64 Byte chunks per credit, which means 2 BL messages will be grouped together and sent , whenever data is requested, consuming 1 credit. AD and BL layers are credited when needed.

There are 2 main types of traffic in SCF model.

1. **End to End Credited traffic**

   When the Coherency Home Agents (CHA) talk to memory, the requests they send are end to end credited, i.e., the credits are reserved by each source to the drop port buffer of the destination. E2E crediting implies traffic is also credited to the TG, and is guaranteed to get onto the rings and off the rings.

2. **Non Credited - Bounceable traffic**

   Non-credited traffic gets on the rings based on slot availability. This traffic is un-credited to the destination and to the intermediate buffers. It makes best effort to get on and off the ring to the intermediate and final destination buffers. If flit cannot exit the ring, it continues on the ring, i.e. it bounces.

On the vertical rings, only the messages which are credited or un-credited to Transgress buffer or Staging Buffer (SBO) are transported. On the horizontal rings, along with the credited and bounceable traffic, uncredited but guaranteed sink i.e., messages with no credit yet guaranteed to be sunk, are transported. In this scenario, the previously sent messages allocates resources for the sub-sequent messages.

Hence the Transgress Buffer is configured to handle both credited and bounceable messages, using separate buffers to avoid blocking each other.

1.1.5 Slots and Rings

In the SCF model, the fabric has Horizontal and Vertical rings that are connected in a fully connected mesh structure. These rings made up of slots, that move around every cycle. These slots accommodate the flits/message packets, which enter and leave the ring at CMS, through the Add/Drop ports.

The various types of slots are:

1. **Normal Slot**

   There are no special restrictions for this type of slot. All types of messages can take these slots.

2. **Anti-Deadlock Slot (ADS)**

   These slots can only be taken by credited messages. This ensures that the credited messages have an assured passage on the ring, without starving the system. These slots are present on the vertical and horizontal rings on both AD and BL layers, as they have credited traffic. These slots can have one of three states:

- free state - In this state, the ADS is free and can accept a credited flit
- In Transit state - In this state, the ADS is carrying a flit
- Delivered state - In this state, the credited flit is delivered, but the slot needs to be freed at the source.

3. **Inter-ring Anti Deadlock Slot (IRADS)**

   This slot is meant to be taken up by messages that are not crossing die, i.e. not going to the SBO. This ensures that two rings are not full of bounceable traffic that is trying to go to the other ring but it can't due to the ring being full. These slots are present only in a multi-die fabric (MDF).

4. **Anti-Starvation Slot (Reserved Slot)**

   Add ports of some CMSs could starved of progress due to loss of arbitration to through traffic. In such cases, the add ports can temporarily reserve a slot, so that it can only be used by the add port that reserves it. This is done to ensure some forward progress for each add port.

1.2 MULTI DIE FABRIC

In SCF, Multi die designs can be implemented using two important components.

1. CRS – Common Ring Stop
2. SBO – Staging Buffer

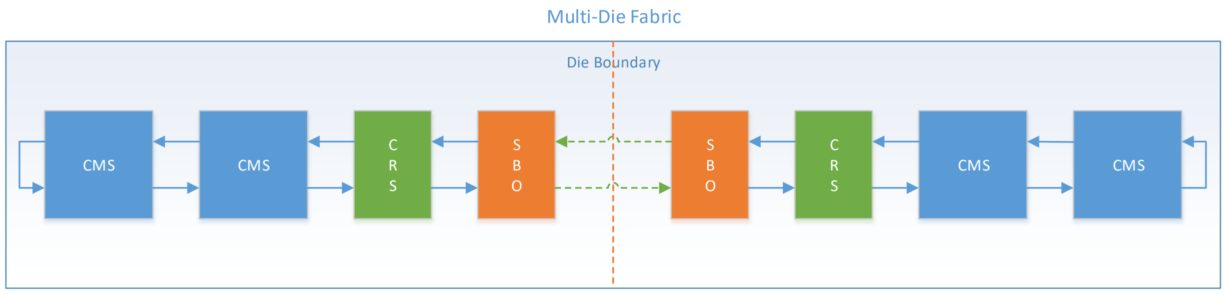

Figure 3 SBO,CRS and CMS in a Multi Die Fabric

The figure above illustrates a Multi-Die Fabric (MDF). Each die has its own sets of hosts and ports associated with the ports. Based on these ports, different types of CMSs are instantiated for the information or communication to occur between the ports. The messages get onto the horizontal or the vertical rings through the add ports at the source and are dropped at the destination through the drop ports.

In an MDF, when this message has to propagate from one die to another, the CMSs in the first die send the message towards the CMS present closer to the die boundary. This CMS sends the information to the CRS, where the message address is verified and sent to the SBO. The SBO holds on to the information and sends it across to the SBO of the adjacent die, through a physical medium (wires). This SBO, then transfers the information to the CRS connected to it, which places the information in the horizontal or the vertical rings, based on slot availability to be propagated to the destination, through the CMSs along the path.

The interfaces involved in the CMS, CRS and the SBO are represented in the following figure.

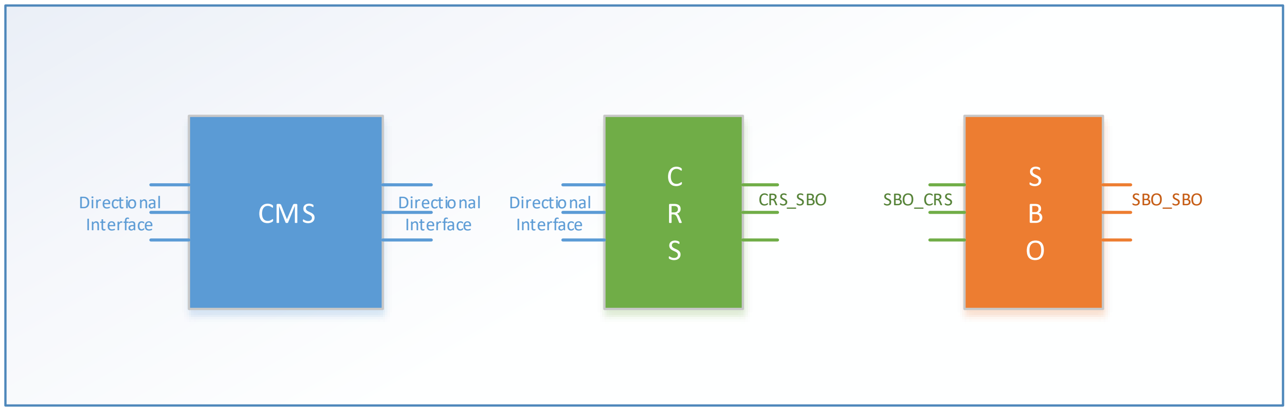

Figure 4 CMS, CRS and SBO interfaces

2 SCF NocStudio

SCF NocStudio was a standalone product (iNocStudio) being used to develop the Sapphire Rapids (SPR) and the upcoming Granite Rapids (GNR) fabric. Beginning 1907, SCF NocStudio has been combined into the same binary as the traditional NocStudio, being license controlled with and "SCF" feature.

2.1 Important NocStudio Commands

SCF NocStudio config script has the same structure as that of the traditional NocStudio config file. Let's take a deeper look SCF config flow and some of the important commands involved.

2.1.1 enter_scf_mode

After opening NocStudio with the said SCF License enabled, to enable the exclusive SCF mode in NocStudio, ***enter_scf_mode*** command has to be entered. This command enables the SCF NocStudio and gets rid of the traditional NocStudio properties we have in the help menu and display only the commands needed to design the SCF Mesh.

2.1.2 set_emb

EMB (Embedded Multi-Die Interconnect Bridge) is the partition between any 2 Die in a Multi-Die design. ***set_emb*** command allows users to set rows and/or columns on which EMB will be placed.

The syntax is as follows:

set_emb [-row {<row>+}] [-col {<col>+}] [-region <scf_region_name>]

<row> specifies the top of two rows between which the EMB will be placed.

<col> specifies the left of two columns between which the EMB will be placed.

2.1.3 add_ihost

This command creates a 1x1 host with one or two ports.

The syntax is as follows:

add_ihost -name <name> [-color <color>] [-pos <pos>] [-ports <port1> [port2]] [-full_ring_col_swap <layers>]

Based on the name of the port, NocStudio automatically decides what type of CMS is to be assigned to the corresponding CMS. NocStudio follows the following convention:

Table 2 CMS Types for Coretools

Port Name	Type of CMS
Core, Cha	CHA_CMS
UPI, MEM, MC etc	SA_CMS

Note: When it comes to coretools, SCF has 2 types of CMSs. CHA_CMS and SA_CMS. SA_CMS combines the CMS_NCAP and CMS_TALL.

2.1.4 set_ifce_noc_layer

The transactions involved in SCF NoC are discussed in *Table 1 SCF Message Types.*

set_ifce_noc_layer command is used to sort these traffic transaction types to different layers to avoid congestion. Assigning a type of transaction onto a particular layer enables NocStudio to create the fabric without traffic congestion. This helps in debug and in traffic analysis.

2.1.5 set_credit_return_layers

This command allows users to choose layers on which the End-to-End Credits and the Transgress credits are returned.

Generally, these credit return layers for e2e and tg are set to the same layers as the transaction types akc and tgc respectively.

The syntax is as follows:

set_credit_return_layers -e2e <layer> -tg <layer>

2.1.6 set_initial_e2e_credits

This command is used to setup credited flows from a source to a destination on an interface.

The syntax is as follows:

set_initial_e2e_credits -src <bridges> -dest <bridges> -ifce <ifce> [-turn_agent] -credits <n> [-return_layer <lay>] [-refill_rate <0-1>] [-class <list of classes 0-15>]

2.1.7 set_credits

This command is used to setup credited flows from a source to a destination where the source is a MeshStop/SBO/TG and the destination is a MeshStop/SBO/TG/interface.

The syntax is as follows:

set_credits -src <MeshStop|SBO|TG> -dest <MeshStop|SBO|TG|interface> -class <classes+> -credits <n>

TGs are represented using TG<Layer_id>.<Node_id> such as TG0.21.

SBOs are represented using SBO<Layer_id>.<Node_id> such as SBO0.31.

Mesh Stops are represented using R<Layer_id>.<Node_id> such as R0.45.

Interfaces are represented using <host_name>/<bridge_name>.<ifce_id>, such as tile1/core.ad.

The valid credit loop combinations are :

1. TG to HSBO.

2. TG to drop port interface.

3. TG to turn agent (same as drop port interface).

4. VSBO to TG.

5. VSBO to VSBO.

6. HSBO to drop port interface.

7. HSBO to turn agent (same as drop port interface).

8. HSBO to HSBO.

9. port credits from mesh stop to TG.

10. port credits from mesh stop to VSBO.

3 NocStudio Collateral

As a part of the **gen_ip** command, NocStudio generates the connectivity information of the configuration sourced in NocStudio. This information, along with the mesh corekits, is fed into collage to design the SoC.

3.1 Coretools

As a part for SCF NocStudio, corekit files related to a config can be generated, provided the related license feature, " *Nocstudio_COREKIT*", is available. If this feature is a part of the license file, NocStudio generates Corekits for the SCF config. The corekits can be found under the ***coretools*** directory, inside the NocStudio created project directory.

The following picture shows the core kits file structure. Let's look at them in detail:

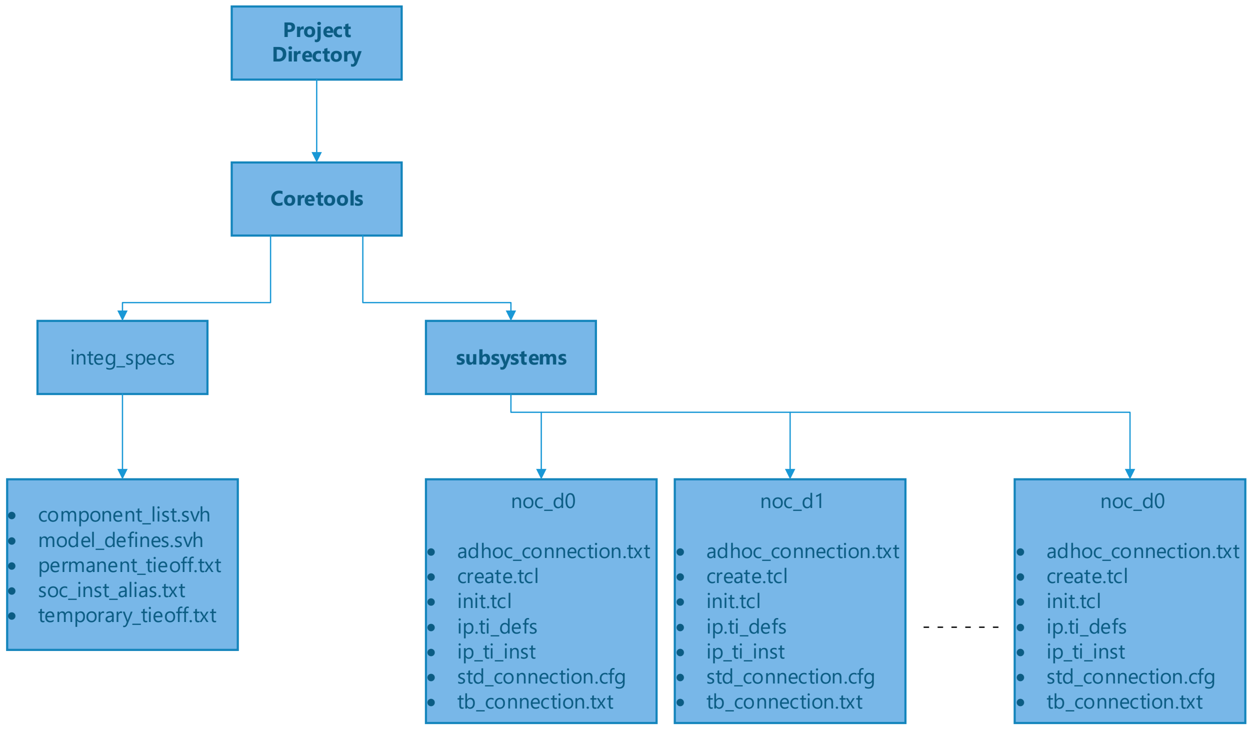

Figure 5 Coretools Directory Structure

3.1.1 Integ_specs

The integ_specs directory contains the files which provide information about the corekits, like the locations, tie-offs etc.

The files are individually explained below:

Table 3 Files under Integ_specs directory

File Name	Usage
connections_list.svh	This file contains a list of tiles and their locations present in the config.
model_defines.svh	This file contains some defines used by our pre processing script for the chassis tb initialize file.
permanent_tieoff.txt	This file is expected by collage for permanent tieoffs. Since mesh doesn't have any, this file is empty.
soc_inst_alias.txt	This file sets up some defines to be used by our TI files and model ral env set up. It will set up all the defines for tiles.
temporary_tieoff.txt	This file is for temporary tieoffs. It will handle adhoc connections for the non-SS corekits.

3.1.2 Subsystems

The Subsystems directory contains sub-directories, corresponding to each of the die in a multi-die config. These sub-directories are based on the number of EMB's (Embedded Multi-Die Interconnect Bridge) present, creating these multiple dies. Each of these dies have the same set of files related to the components present in the die. The list of files is as follows:

Table 4 Files under individual die directories under SubSystems

File Name	Usage
adhoc_connection.txt	This file is used for specifying adhoc connections. In SCF connectivity, it is used for exporting clock and reset pins to the next level in the RTL.
create.tcl	This file contains the instances of each corekit in the subsystem, and is consumed by collage to create instances of each corekit with the defined name, and connect their pins and interfaces appropriately.
init.tcl	This file is used to initialize and load the corekits used in each subsystem. It is consumed by collage to load all the corekits that will be used in stitching up that subsystem.

ip.ti_defs	This file is used for setting up the test islands (TIs) and their input/output interfaces for each tile.
ip.ti_inst	This file is for setting up the TI parameters for each TI.
std_connection.cfg	This file is used to represent the connectivity between the interfaces of different instances of corekits in the subsystem. In SCF collage collateral, this file lists out the interface connectivity between all the directional interfaces (Cring interfaces), as well as tie-offs for the u-turn connections at the ends of the ring.
tb_connection.txt	This file is for setting up connections for the adhoc signals in the TI (e.g. clock/reset signals on the TI interfaces). This needs to exist for each corekit instance. The connections exist for each add/drop interface that the particular TI has (see ip.ti_defs section for the list).

4 NocStudio Commands For SCF

4.1 Commands

Command Name	Comment
list_node_regbus_layers	This command lists the association between nodes and regbus layer, with all elements (bridges, routers, and agents) associated.
reset_node_regbus_layer	This command resets the association between regbus layer and node back to default value.
set_node_regbus_layer	This command changes the regbus layer associated to one or more nodes specified in the list.
query_links	This command is used to list the links with certain properties between rtl groups. Note that links with ILDC aren't listed.
enter_scf_mode	This command is used to reset the mesh and enter into SCF NocStudio Mode.
set_initial_e2e_credits	This command is used to setup credited flows from source to a destination on an interface.
set_credits	This command is used to setup credited flows from a source to a destination where the source is a MeshStop/SBO/TG and the destination is a MeshStop/SBO/TG/Interface.
add_slots	This command used to add a type of slot with access permissions for deadlock analysis.
set_credit_return_layers	This command is to choose layers on which e2e and tg credits are returned
add_ihost	This command is used to creates a 1x1 host with one or two ports for SCF mesh.
print_flits_in_noc	This command prints the flits in all the channel buffers of the NoC.

set_emb	This command is used to set rows and/or columns on which EMB will be placed.
set_row_bridging	This command is used to set the bridging between a pair of rows.
set_col_bridging	This command is used to set the bridging between a pair of columns.
reset_row_bridging	This command is used to reset the row bridging specified using set_row_bridging
reset_col_bridging	This command is used to reset the column bridging specified using set_col_bridging
set_throttling	This command sets the throttling in specified directions for the routers.
set_polarity	This command sets row and column polarity (even/odd/none) from sources to destinations on a given interface.
show_polarity	This command shows polarities (even/odd/none) between two bridges for a single interface_id
set_route_direction	This command sets row injection direction (east/west/shortest) and column injection direction (north/south/shortest) from sources to destinations on a given interface.
show_route_direction	This command shows route directions (east/west/shortest), (north/south/shortest) between two bridges for a single interface_id.
set_tg_node	This command sets the position of the TG to be used for pairs of sources to destinations on a given interface.
show_tg_node	This command shows positions of TG's that will be used between two bridges for a single interface_id.
set_two_flit_layer	This command sets a layer to support two flit transactions.

set_ads_slot_layer	This command sets the layers to contain the specified number of ADS or IR-ADS slots on all row and column rings.
set_packet_groups	This command sets the layers/message types in a group (group A) to be used in 1x mode in simulation.
list_packet_groups	This command lists the layers and message types in each packet group.
reset_packet_groups	This command resets the packet group of each layer/message type to none.
set_two_beat_credited_packets_layer	This command sets a layer in which two credited packets are sent as a part of the same hop of the transaction but consume only one credit.

4.2 Default Properties

Property Name	Default Value	Comment
compress_strap_enable	No	This property when enabled sets the Strap drivevalue ports to local. config_strap_id command is used as mux to pass values.
tcl_print_mode	Warn-only	This property enables printing all warning messages in tcl mode.
log_all_packets	Yes	This property decides whether to save every packet template for detailed statistics to be computed after simulation.
stats_view_green_threshold	0	This command is to set the minimum value of throughput/occupancy at which tiles in the stats view are colored green. This threshold must be less than the stats_view_orange_threshold.
stats_view_orange_threshold	60	This command is to set the minimum value of throughput/occupancy at which tiles in the stats view are colored

		orange. This threshold must be more than the stats_view_green_threshold and less than the stats_view_red_threshold.
stats_view_red_threshold	100	This command is to set the minimum value of throughput/occupancy at which tiles in the stats view are colored red. This threshold must be more than the stats_view_orange_threshold.
scf_deadlock_analysis_enable	Yes	This property when set to true, will enable SCF deadlock detection analyzer to detect possible deadlocks during mapping based on the traffic added in SCF NocStudio.
emb_latency	0	This property sets the default number of cycles taken by a data or credit message to cross the EMB i.e. from SBO ingress on one die to the SBO egress on another die.
scf_stamping_enable	no	This property when set to true, will enable multi-instancing of SCF CMS modules.

4.3 MESH PROPERTIES

Property Name	Default Value	Comment
security_interrupt_enable	no	This property is used to expose/tie-off a security interrupt signal from all modules.
visa_enable	no	This property when set to true, the VISA pins are generated for the mesh stops during gen_ip.
intel_inst_enable	no	This property when set to true, the pins guarded by INTEL_INST_ON in the RTL

		are also generated in the wrapper during gen_ip.
sbo_to_ta_credit_enable	yes	This property when false, the source gets credit to TA. Else, the SBO gets credit to TA.
two_flit_col_polarity_mode	even_odd	This property is used to set the mode of injection of flits into the vertical ring from the host port, based on the polarity.
two_flit_row_polarity_mode	even_odd	This property is used to set the mode of injection of flits into the horizontal ring from the TG, based on the polarity.
global_back_pressure_type	none	This property is used to set the global back pressure mechanism in the NoC.
back_pressure_threshold	3	This property sets the number of messages of space left in buffers to enable global.
throttling_hysterisis	10	This property sets the number of cycles for the throttled sender to remain throttled before resuming at usual rate.
sink_rule_policy	stripes	This property is used to set the policy used to set the sink rules on all bridges.
enable_fast_map	yes	When this property is set to true, the mapping speed is increased.
throttling_rate_factor	0.5	This property sets the throttle rate factor for the mesh stop that the injection rate of the throttled mesh stop will be reduced by.
enable_1x_mode	no	This property enables 1x mode in simulation.

4.4 Bridge Properties

None

4.5 Host Properties

None

4.6 Interface Properties

Property Name	**Default Value**	**Comment**
rate_limit_incr_trigger	no	When a credited packet with this label is sent or received, add a token to the rate limiter for that packet's src/dest.
rate_limit_decr_trigger	no	When a credited packet with this label is sent, use a token from the rate limiter for that packet's dest.
drop_bounce_buffer_size	4	This property is used to define the DROP port's bounce buffer size.
credits_from_west_sbo	8	This property is used to set the number of credits available for the west sbo on the same layer as this interface, to send to this interface.
credits_from_east_sbo	8	This property is used to set the number of credits available for the east sbo on the same layer as this interface, to send to this interface.
credits_from_tg	8	This property is used to set the number of credits available for the Transgress buffer on the same layer as this interface, to send to this interface.

4.7 Link Properties

Property Name	Default Value	Comment
emb_data_latency	0	This property sets the number of cycles taken by a data message to cross the EMB i.e. from SBO ingress on one die to the SBO egress on another die.
emb_credit_latency	0	This property sets the number of cycles taken by a credit message to cross the EMB i.e. from SBO ingress on one die to the SBO egress on another die.

4.8 CMS Properties

Property Name	Default Value	Comment
tg_ingress_bounce_buffer_size	8	This property is used to define the bounce input buffer size in the Transgress buffer.
tg_ingress_credit_buffer_size	8	This property defines the credit input buffer size in the Transgress buffer.
tg_egress_bounce_buffer_size	8	This property is used to define the egress bounce buffer size in the Transgress buffer.
tg_egress_credit_buffer_size	8	This property is used to define the egress credit buffer size in the Transgress buffer.
add_blocked_cycles_until_reserve	n/a	This property is to set the number of add blocked cycles before reserving a slot.

tg_blocked_cycles_until_reserve	n/a	This property sets the number of tg blocked cycles before reserving a slot.
credits_from_sbo	n/a	This property sets the number of credits from SBO in column of <n> to TG at the given position for the specified traffic class.
credits_to_sbo	n/a	This property sets the number of credits from the TG to the SBO on its horizontal ring.
credits_to_other_sbo	8	This property sets the number of credits from the current SBO to the other SBO.
sbo_ingress_bounce_buffer_size	8	This property sets the bounce buffer size of the ingress buffer on the SBO.
sbo_ingress_credit_buffer_size	8	This property sets the credit buffer size of the ingress buffer on the SBO.
sbo_egress_bounce_buffer_size	8	This property sets the bounce buffer size of the egress buffer on the SBO.
sbo_egress_credit_buffer_size	8	This property sets the credit buffer size of the egress buffer on the SBO.
turn_agent_bounce_buffer_size	4	This property sets the turn agent bounce buffer size on the router.
turn_agent_latency	4	This property sets the turn agent latency on the router.
throttling_threshold	512	This property sets the throttling threshold for the router, which is the number of cycles that the mesh stop is unable to send packets from bridges connected to it, after which the throttling signal is sent.
port_credits_to_tg	8	This property is used to set he initial number of TG credits for ports to send through CMS

port_credits_to_sbo	8	This property is used to set the initial number of credits to the SBO on the same vertical ring for each initiator
sink_rule	n/a	This property is used to set the sink rule for the rx interface.