TCL User Guide

Doc Name	Revision	Date	Description
100061_UG_3.2.0	3.2.0	Apr 14, 2022	Ncore 3, Version 3.2.0
100061_UG_3.2_RC2	RC2	Dec 10, 2021	Ncore 3, Version 3.2.0_RC2
100061_UG_3.2_RC1	RC1	Nov 19, 2021	Ncore 3, Version 3.2.0_RC1
100061_UG_1.0_Beta3	Beta3	Oct 13, 2021	Beta3 release
100061_UG_1.0_Beta2	Beta2	Sep 13, 2021	Beta2 release
100061_UG_1.0_Beta1	Beta1	June 15, 2021	Beta1 release
100061_UG_1.0_A.2_D1	1.0_A.2_D1	Mar 12, 2021	1.0_A.2_D1 preliminary release
100061_UG_1.0_A.1	1.0_A.1	Nov 23, 2020	1.0 release

Contents

1 Introduction

1.1 About this Document

This document provides information on using the Maestro tool with the Tcl command interface. An annotated demonstration design is provided along with a description of the Maestro Tcl commands and parameters.

1.2 Maestro Overview

Maestro is the complete software environment provided to the user when they license an ArterisIP product.

The Maestro platform implements the client server model of software solutions.

- Server side : This handles the RTL generation of the solution and all the services related directly to it like Test Bench generation, Simulation stimulus, Physical and logic Synthesis scripts, IP-XACT models, Register maps, and Model generation.
- Client Side : This is where the customer can create, constrain, optimize and validate the interconnect solution.

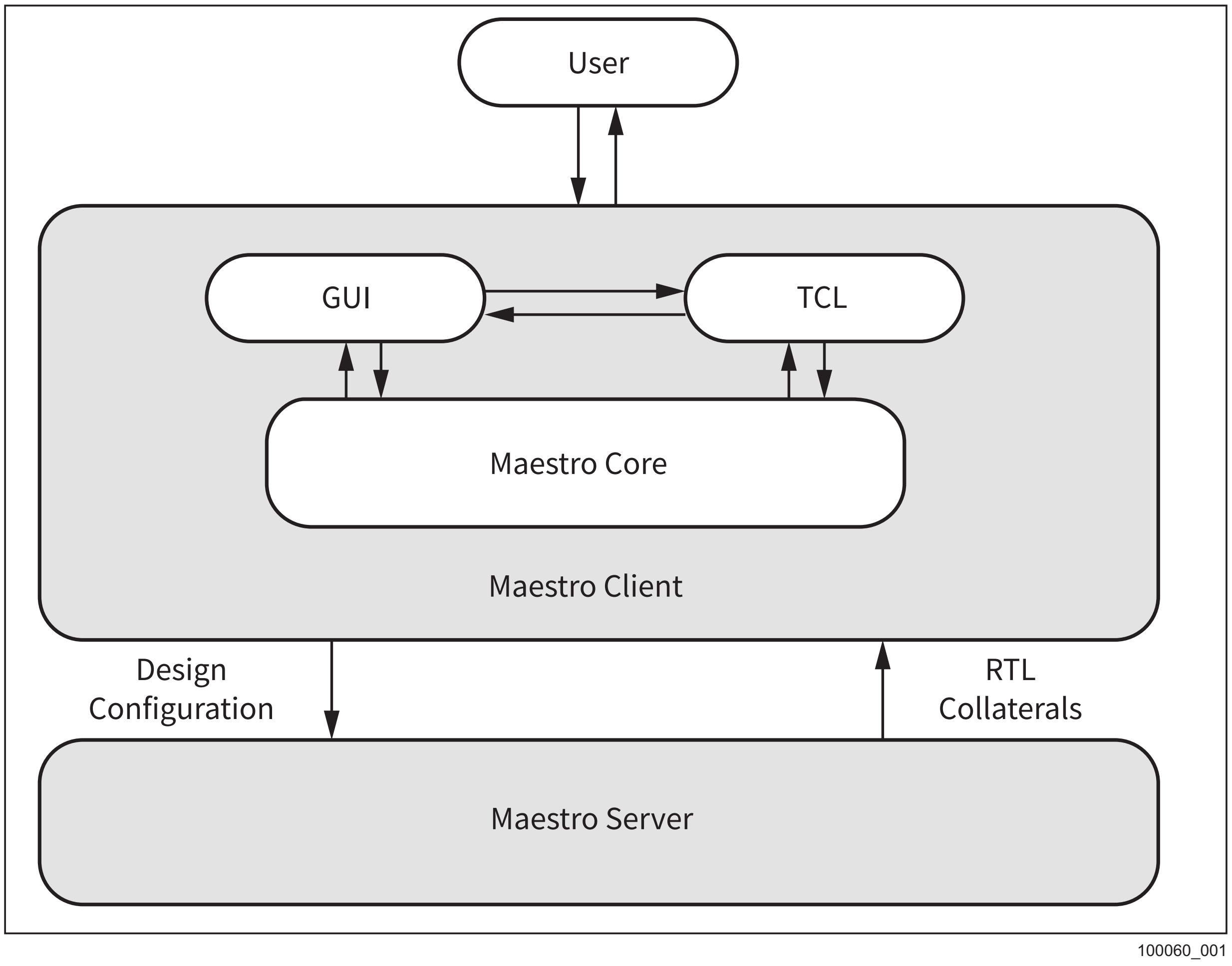
The Maestro client is in turn composed of two facets.

1. The Maestro GUI which can be graphically used to manipulate the design. (Note: Refer to the Ncore User Guide for more information.
2. The Maestro Tcl interface where the user can interact from the command line terminal or write a Tcl script to create and manipulate the design.

Maestro supports assembling multiple ArterisIP products together to create complete communication subsystems for System on Chips of all sizes and complexity. It enables users to configure, explore, and optimize the communication subsystems comprising multiple instances of various ArterisIP products efficiently. After configuration and optimization, Maestro enables users to generate the set of deliverables required for integration and implementation in a chip: simulation, synthesis, backend, verification, documentation, and so on. It implements a unified and consistent workflow which supports the whole life cycle of ArterisIP products design. The workflow guides the user to input data without errors.

Refer to Figure 1 for an architectural overview of the Maestro Software.

Figure 1.Maestro Software Architecture Overview



1.3 Tcl Usage in Maestro

Maestro uses Tool Command Language (Tcl) as the scripting language. Maestro has a list of Tcl commands which provides direct access to the objects in the data model (see “Maestro Tcl Commands”).

1.3.1 Using Tcl in Interactive mode

Individual Maestro Tcl commands can be entered in the Tcl shell to interact directly with Maestro by asking for information from the design or modifying the parameters of an object in the design.

Example: Query information regarding the gating of the clock domain attached to a particular object, modify the parameter values of the unit, or check the existence of an object of a particular type and name.

To see the Maestro interactive commands, start Maestro and type:

```
help
```

To get help for any Tcl command, use the "-help" option for the command.

For example, to get help on the "create_project" command, type:

```
create_project -help
```

The individual commands can be run from the command line or from the Tcl console at the bottom of the Maestro GUI.

1.3.2 Running a Tcl Script

A Tcl script can be sourced from the Tcl command-line option or from the Tcl console in the Maestro GUI in the following way:

```
source <file-name.tcl>
```

When a Tcl script is being run, it displays all the operations taking place to create a design and all other operations from the user are blocked until the script completes.

If the script is interrupted before completion, all the work will be lost since the last save.

1.3.3 Maestro Log Files

When the Maestro executable is invoked, it writes the `maestro.log` file in the same directory where the design is executed. The log file contains a record of all the operations performed at every stage of the design. It categorizes the messages as Info, Warning, Deprecated or Error.

You will not be able to generate the RTL if there are any errors present in the `maestro.log` file.

1.3.4 Tcl Initialization Scripts

Upon invocation, Maestro automatically sources one or more files called `maestro_init.tcl` in the following order:

1. `$HOME/maestro_init.tcl`
   if file exists
2. `$PWD/maestro_init.tcl`
   if file exists

2

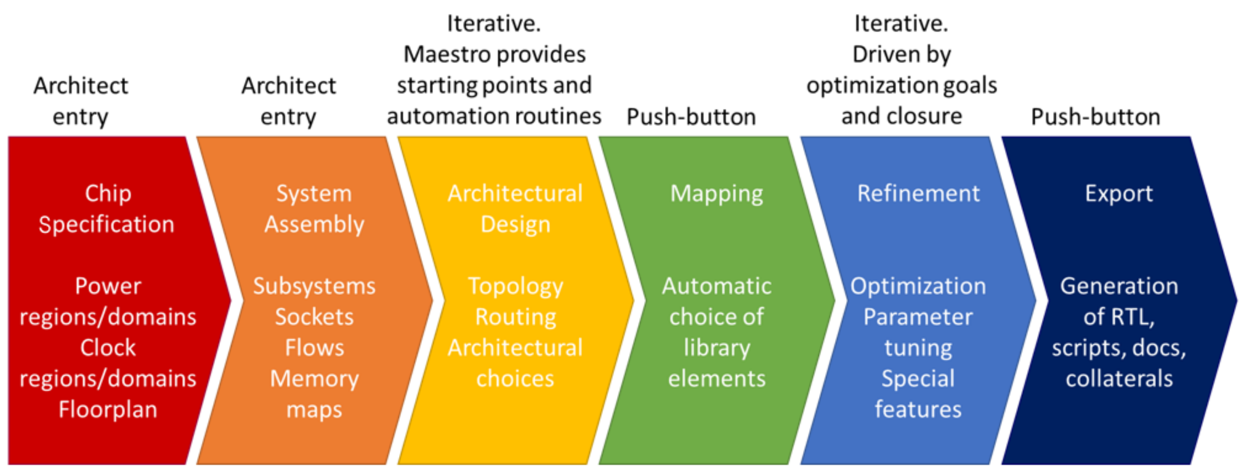
Maestro User Workflow

2.1 Introduction

The Maestro design workflow is the sequence of operations that the user must follow, and the software must enforce, to create a fully configured and ready to implement communication subsystem.

The following diagram shows the phases of Maestro User Workflow.

Figure 2. Maestro User Workflow



The Maestro Phases are defined as follows.

Phase 1. Chip Specification: Define clock and power regions.

Phase 2. System Assembly: Define sockets, connectivity requirements (flows used in Symphony) and the memorymaps.

Phase 3. Architectural Design: This is the step during which one or multiple solutions are conceived to implement the requirements. This step involves creating the fully-connected network topology and also the creation and configuration of Ncore units to offer cache coherence.

Phase 4. Mapping: This step casts the solution of the previous stage into the specific versions of the specific components in the Arteris IP library. This step is fully automatic.

Phase 5. Arcitectural Refinements: Here the user can fine-tune parameters to achieve specific objectives; for example, choosing buffer sizes, tweaking arbitration policies, and so on.

Phase 6. Export: This step invokes the server component of Maestro to generate RTL and collaterals. This process is only allowed if the design is clear of any outstanding errors.

Through the design flow, Maestro provides guidance to the designer about the current and next steps. Several actions such as creation of Ncore units and inserting adapters are automated to minimize the iteration time and the possibility of misconfiguration. Additionally, checks are deployed in the flow to prevent incorrect parameterizations.

2.2 Understanding the Object Hierarchy

Objects in Maestro are user-visible items in the GUI.. They organize and structure the way you can interact with Maestro, through creation, deletion, editing, and various calls to methods using the Tcl commands. Objects are hierarchical and may contain other objects. This hierarchy represents relationships between the information itself, and information contained in a child object only makes sense in the context of its parent. Changes to information typically propagates from parents to children.

The Maestro design hierarchies shown in the following figures provide an overview of the relationship among various objects in the design. Use the figures and descriptions as a guideline for understanding the Maestro design workflow and creating your design using Maestro.

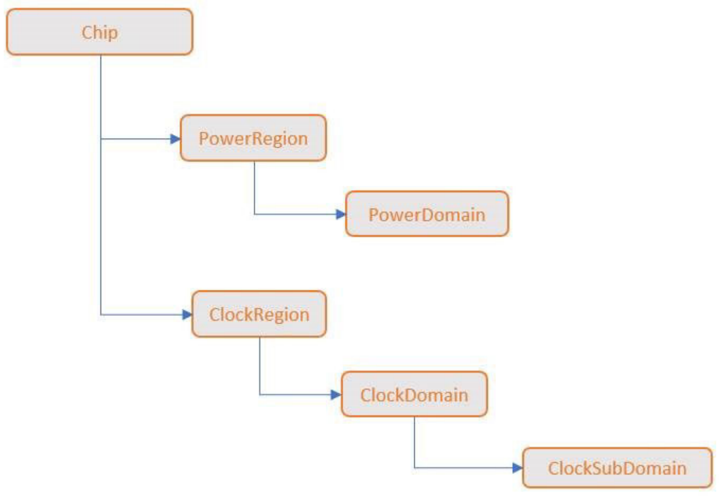
In the following figures, objects are shown in the orange colored boxes for the stages of the Maestro workflow.

- “Phase 1: Chip Specification”
- “Phase 2: System Assembly”
- “Phase 3: Architecture Design”
- “Phase 4: Mapping”
- “Phase 5: Refinement”
- “Phase 6: Export”

2.2.1 Phase 1: Chip Specification

The following objects should be created and modified in Phase 1.

Figure 3. Maestro Design Hierarchy: Phase 1



Project

A project is a container for all information required to create a communication subsystem. All elements of a projects are related to each other. Maestro lets the user work on one project at a time.

Type	Parent
project	None

Example Tcl code snippet:

```
create_project -name "project1"
```

Chip

Chip is the container of multiple components including the power, clocks, units and the networks which constitute the total Interconnect. The notion of a chip correlates to that of a silicon die. Maestro allows you to create only one chip under a project.

Type	Parent Type
chip	project

Example Tcl code snippet:

```
create_object -type chip -name "chip" -parent project
```

Power Region

A power region is a container for design elements supplied by the same power rail. Each region should contain at least one power domain. The frequency is an attribute of the power region.

Type	Parent Type	Parameter
power_region	chip	voltage (mV)

Example Tcl code snippet:

```
set pregion [create_object -name "PwrRegion" -type power_region \
-parent project/chip]
set_attribute -object $pregion -name voltage -value 900mV
```

Power Domain

A power domain is a container for design elements which share a common power supply. This power supply can be turned on or off by an external Power Management Agent (PMA). For the PMA to be enabled, the gating should be selected as ‘external’. For constant power supply, the gating is selected as ‘always on’.

Type	Parent Type	Parameter
power_domain	power_region	gating

Example Tcl code snippet:

```
set pwrDomain [create_object -type power_domain -name "PwrDomain" \
-parent project/chip/PwrRegion]
set_attribute -object $pwrDomain -name gating -value always_on
```

Clock Region

A clock region is a container of design elements which is driven by one external clock source. Each region should contain at least one clock domain. You must bind the clock region to the power region. The frequency of the clock must be set on the clock region.

Type	Parent Type	Parameter
clock_region	chip	Frequency (kHz)
clock_region	chip	unitClockGating Unit clock gating can also be set under the clock region.

Example Tcl code snippet:

Create a clock region named "ClkRegion" with a frequency of 1.6 GHz (`1600000` kHz).

```
set clkregion [create_object -name "ClkRegion" \
-type clock_region -parent project/chip]
```

```
set_attribute -object $clkregion -name frequency -value 1600000
set_attribute -object $clkregion -name unitClockGating -value true
update_object -name $clkregion -bind $pregion
```

Clock Domain

A clock domain is a container for design elements which share a common clock that can be turned on and off. The clock domain must be associated with a power domain. A power domain can span multiple clock domains.

Each clock domain has an associated PMA (Power Management Agent). The PMA is enabled when the gating of the clock domain is set to external—i.e., the clock domain is connected to switchable power domain. Otherwise, the gating can be set to 'always_on'.

Type	Parent Type	Parameter
clock_domain	clock_region	gating

Example Tcl code snippet:

```
set clkdomain [create_object -name "ClkDomain" \
-type clock_domain -parent project/chip/ClkRegion]
set_attribute -object $clkdomain -name gating -value always_on
update_object -name $clkdomain -bind $pwrDomain
```

Clock Subdomain

The clock subdomain is derived from a clock domain. Every object in Maestro must be associated with a clock subdomain.

Type	Parent Type	Parameter
clock_subdomain	clock_domain	divideby

Example Tcl code snippet:

```
set clocksubdomain [create_object -name "ClkSubdomain" \
-type clock_subdomain -parent project/chip/ClkRegion/ClkDomain]
```

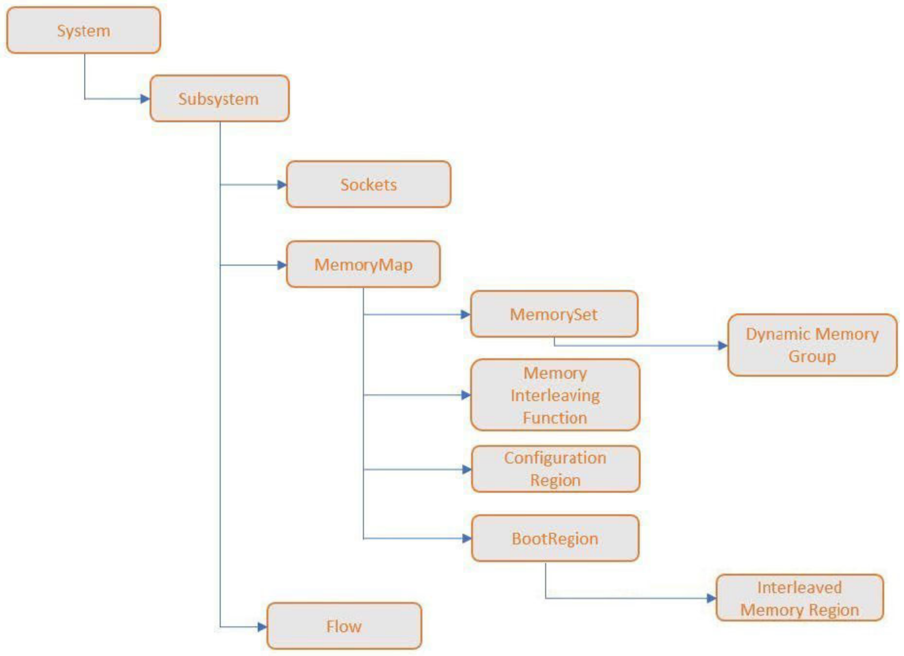
Checklist for Phase 1

State	Title	Necessary completed actions
Domain state	Create chip	Chip is created
	Check domains	At least one power region exists
		At least one power domain exists
		At least one clock region exists
		At least one clock domain exists
		At least one clock subdomain exists
	Check parameters	Power Region -> Voltage is set
		Power Domain -> Gating is set
		Clock Region -> Frequency is set
		Clock Region > unitClockGating is enabled
		Clock Domain -> Gating is set
	Check binding	All clock regions bound to power regions
		All clock domains bound to power domains

2.2.2 Phase 2: System Assembly

The following objects should be created and modified in Phase 2.

Figure 4. Maestro Design Hierarchy: Phase 2



System

A system is a container for information, filled by the user manually and/or through automation. A system can contain only a single instance of a subsystem for a particular project.

Type	Parent Type
system	chip

Example Tcl code snippet:

```
create_object -type system -name "system" -parent project/chip
```

Subsystem

A subsystem is a set of units configured by the user and connected together to implement a high-level function such as a non-coherent interconnect, coherent interconnect, last level cache, and so on.

Type	Parent Type	Paremeter
subsystem	system	subsystem Type

The available subsystem Types are:

- ARTERIS_COHERENT
- ARTERIS_NONCOHERENT

Example Tcl code snippet:

```
create_object -type subsystem -name "subsystem" \
-parent project/chip/system
set_attribute -object $subsystem -name subsystemType \
-value "ARTERIS_COHERENT"
```

Socket

A socket is the interface in the interconnect which connects to external components using one of the ARM-defined protocols. For each Socket, the following information is mandatory:

- A Protocol. Supported Protocol types are CHI-A, CHI-B, ACE, AXI4, ACE-Lite, ACE-Lite-E
- A Function. The function of the Socket is used to infer which type of Ncore Unit is deployed on the inside of the Subsystem. Allowable values are:
    - INITIATOR for any initiators, only for SLAVEs.
    - MEMORY for targets connected to memories, only for MASTERs.
    - PERIPHERAL for targets connected to peripherals, only for MASTERs.
- A Clock Subdomain. This choice should be selected among the Clock Subdomains created in "Clock Subdomain".

Type	Parent Type	Parameters	Associated with
socket	subsystem	protocolType socketFunction params/wAddr params/wData params/wAwUser params/wArUser params/wAwId params/wArId	Clock SubDomain

Example Tcl code snippet:

```
set sock [create_object -type socket -name "dmi2" \
-parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
socketFunction "MEMORY]
update_object -name $sock -bind $clocksubdomain -type domain
```

MemoryMap

This object is the container to store the specifications related to the system address map. There needs to be at least one memory map under a subsystem.

Type	Parent Type
memory_map	subsystem

Example Tcl code snippet:

```
create_object -type memory_map -name "mm" \
-parent project/chip/system/subsystem
```

MemorySet

MemorySet consists of multiple dynamic memorygroups. Each memory set contains the entire set of DMI sockets arranged in groups. There can be a maximum of 2 memorysets under a memorymap. Each memoryset must have unique grouping of DMI sockets.

Type	Parent Type
memory_set	memory_map

Example Tcl code snippet:

```
set memorySet [create_object -type memory_set -name "memorySet" \
-parent project/chip/system/subsystem/mm]
```

Dynamic Memory Group

The DMI sockets in the system can be grouped into distinct groups, each being called a Dynamic Memory Group (DMG). The size of each DMG can be 1, 2, 4, 8, or 16. A given DMI socket can belong to at most 1 DMG.

Type	Parent Type	Associated with
dynamic_memory_group	memory_set	DMI sockets

Example Tcl code snippet:

```
set memoryGroup [create_object -type dynamic_memory_group \
-name "mg01" -parent project/chip/system/subsystem/mm/memorySet]
update_object -name $memoryGroup -value_list \
[list project/chip/system/subsystem/dmi0]
```

Memory Interleaving Function

This object is a container for the primary selection bits. When more than one memory component is assigned to a dynamic memory group, the user must select selection bits to identify the DMI used for memory address interleaving.

- One primary bit must be specified for a DMG of size 2.

- The user must input primary selection bits to identify the MEMORY component used for memory address interleaving.

Type	Parent Type	Parameters
memory_interleave_function	memory_map	primaryInterleavingBitOne primaryInterleavingBitTwo

Example Tcl code snippet:

```
set Func1 [create_object -type memory_interleave_function \
-name "func1" -parent project/chip/system/subsystem/mm]
set_attribute -object $Func1 -name primaryInterleavingBitOne \
-value 6
set_attribute -object $Func1 -name primaryInterleavingBitTwo \
-value 7
```

Contraints Related to Interleaving

Some of the constraints that Maestro checks are as follows:

1. Each address bit in the set of the primary selection bits must be unique—i.e. an address bit must appear only once in the set of primary selection bits.

2. The address bits must be chosen from a range whose upper bound is less than the size of the system address, and the lower bound is greater than or equal to the size of the cacheline.

The constraints on DMI interleaving are as follows:

1. The number of DMI sockets in a Dynamic Memory Group can either be 2,4, or 8.

2. For DMI interleaving, if the number of sockets in a Dynamic Memory Group is *n*, The number of primary selection bits is the binary logarithm of the number of sockets in the group.

The constraints on DCE interleaving are as follows:

1. All the DCEs form a single group and the user does not have the option of segregating down the DCEs into smaller groups.

2. The number of DCEs can only be 1, 2, 4, 8 or 16.

BootRegion

This object is the container for the boot region base address, size, and access point.

The base address and the memory size for the bootregion must be defined by the user. The size of the boot region must be a power of 2 within the range of 4KB and 8TB. The base address of the boot region must be aligned to the size of the region.

The bootcode can be accessed from system memory or peripheral storage. Access from system memory indicates the specific Dynamic Memory Group from Memory Set `0`. The peripheral storage indicates the specific DII socket.

Type	Parent Type	Parameters	Associated with
boot_region	memory_map	memoryBase memorySize	DMG group in memoryset`0` or DII socket

Example Tcl code snippet:

```
set boot [create_object -type boot_region -name "boot" \
-parent project/chip/system/subsystem/mm]
set_attribute -object $boot -name memoryBase -value [expr 0x0]
set_attribute -object $boot -name memorySize -value [expr 0x4000]
update_object -name $boot -bind project/chip/system/subsystem/dii0
```

Configuration Region

This object is the container for the CSR (Control and Status Registers) base address.

- The base address for the CSR region must be provided by the user.
- The default memory size of the Configuration region is 1MB.
- The base address of the configuration region must be 4MB aligned.

Type	Parent Type	Parameters
configuration_region	memory_map	memoryBase

Example Tcl code snippet:

```
set csr_region [create_object -type configuration_region \
-name "csr" -parent project/chip/system/subsystem/mm]
set_attribute -object $csr_region -name memoryBase \
-value [expr 0x2e6e4000]
```

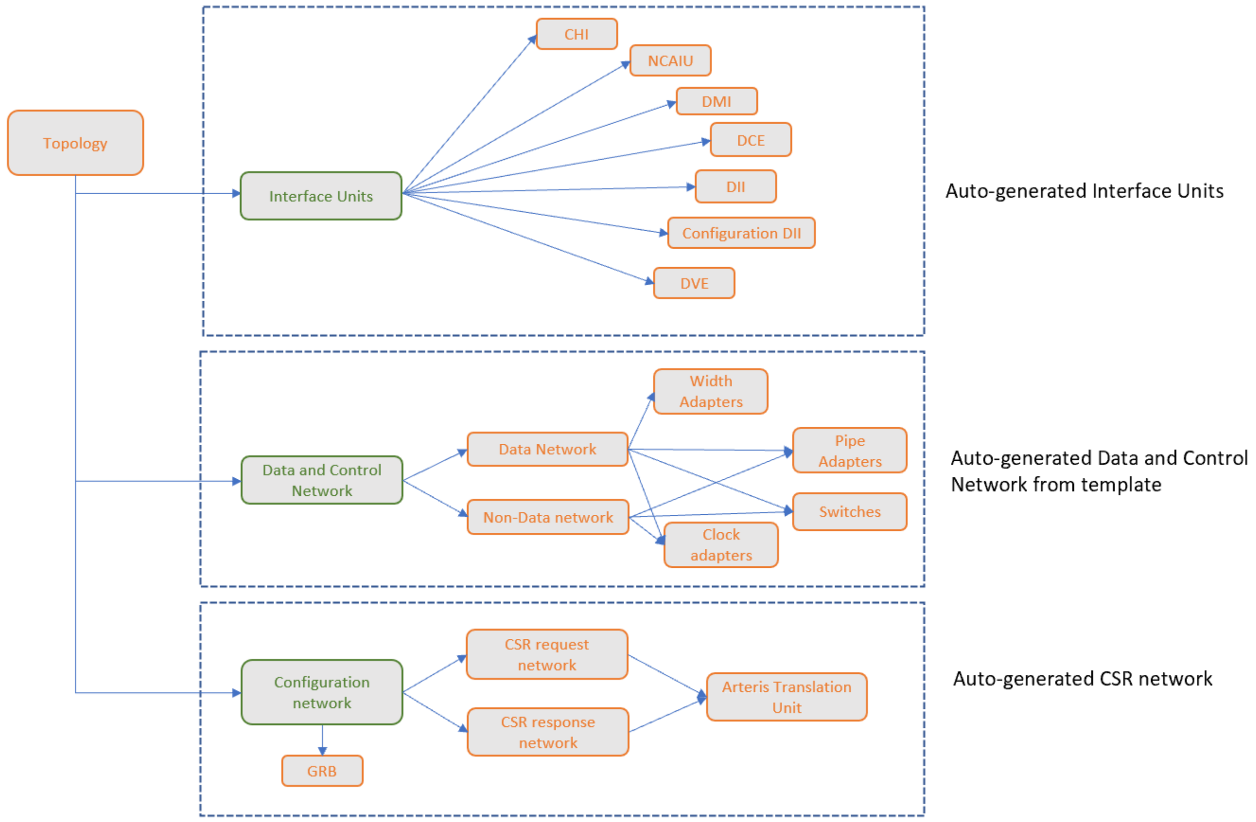
Checklist for Phase 2

State	Title	Necessary completed actions
System state	Initialize the system	System is created
		Subsystem is created
	Check parameters for subsystem	Subsystem -> subsystemType
	Socket state	At least two sockets are created
		At least one socket with SLAVE role and INITIATOR function is created
		At least one socket with MASTER role and MEMORY function is created
	Check parameters	Socket -> protocolType
		Socket -> socketFunction
		Clock Domain -> Gating is set
	Check binding	All sockets bound to clock subdomains
	Memory Map state	Memory Map is created
	Check memories	Maximum of 2 Memory Sets are created
		Memory groups are created under Memory Sets
		Memory group can contain at most 4 MEMORY sockets
		Memory interleaving Function is created
		Boot Region is created
		Configuration Region is created
	Check binding	All MEMORY sockets should be bound to a Dynamic Memory Group
	Check Parameters	Boot_region -> memoryBase Boot_region ->memorySize Configuration_region ->memoryBase

2.2.3 Phase 3: Architecture Design

This phase provides a solution by automatically creating various objects.

Figure 5. Maestro Design Hierarchy: Phase 3



Topology

Topology is the container of the Units and Networks.

Type	Parent Type	Parameter
topology	subsystem	coherentTemplate

Example Tcl code snippet

```
set topology [create_object -type topology -name "topology" \
-parent project/chip/system/subsystem]
set_attribute -object $topology -name coherentTemplate \
-value "OneCtrlOneDataTemplate"
```

Coherent Template

A Template must be chosen for the Topology. The role of the Template is to choose the number of Networks of the Topology, and the partitioning of messages across the Networks.

Three types of networks are automatically generated from Maestro:

1. Datapath network

2. Command network (non-Data network)

3. Configuration network

The coherent template for this topology is selected from the given options:

- OneCtrlOneDataTemplate: This template provides one data and one command network (deprecated).
- TwoCtrlOneDataTemplate: This template provides one data and two command networks.
- ThreeCtrlOneDataTemplate: This template provides one data and three command networks.

Datapath Network

A Datapath network carries data messages. In addition to command and/or control information, a data message carries data that may be read from or written into Ncore Units.

Command Network

A non-data network carries the non-data messages or the command messages. A non-data message carries only command and/or control information.

Configuration Network

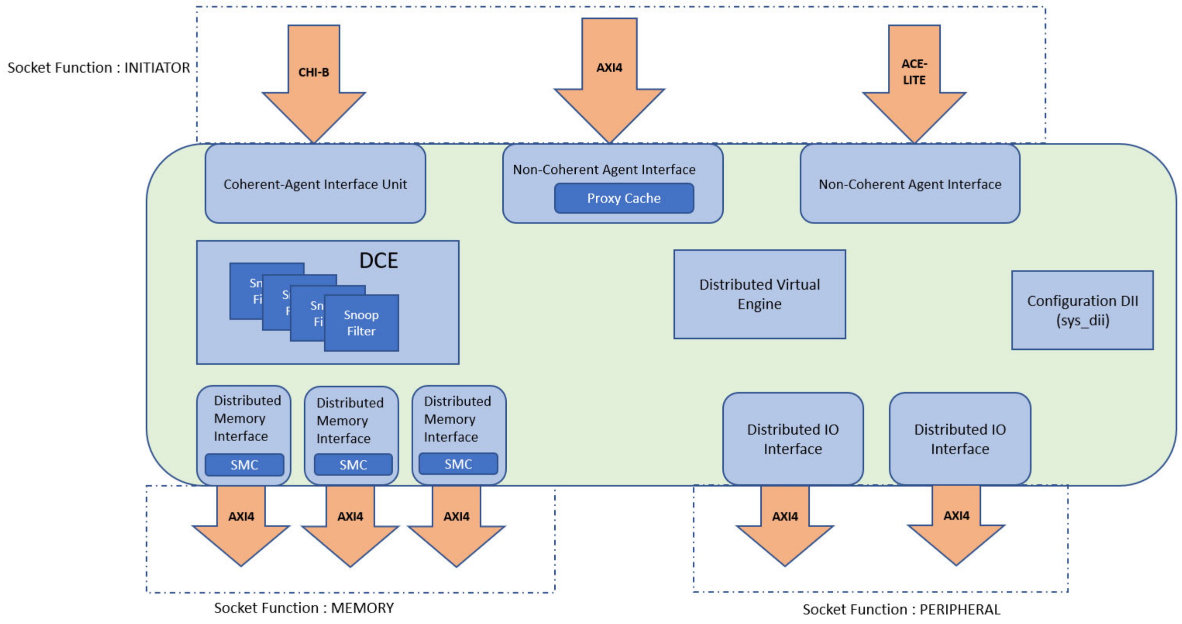
Configuration network is used to write data to or read data from the registers in the Ncore Units.

Ncore Units

A system that is built using the Ncore product can contain the following units. All these units cannot be created by the user and are auto-generated in Maestro, depending on the sockets specified by the user.

Figure 6 shows an example of a structural implementation of an Ncore design with Ncore units.

Figure 6. Structural implementation of an Ncore design with Ncore units


Socket Function : INITIATOR
CHI-B
AXI4
ACE-LITE
Coherent-Agent Interface Unit
Non-Coherent Agent Interface
Proxy Cache
Non-Coherent Agent Interface
DCE
Snoop Filter
Distributed Virtual Engine
Configuration DII (sys_dii)
Distributed Memory Interface
SMC
Distributed Memory Interface
SMC
Distributed Memory Interface
SMC
Distributed IO Interface
Distributed IO Interface
AXI4
AXI4
AXI4
AXI4
AXI4
Socket Function : MEMORY
Socket Function : PERIPHERAL

- **CAIU** (Coherent Agent Interface Unit) : This is a fully coherent unit that supports CHI-A, CHI-B and ACE protocols.
  - The socket attached to this unit is an INITIATOR.
  - For an Ncore3 system, there should be at least one CAIU in the system.
- **NCAIU** (Non-Coherent Agent Interface Unit) : This is an I/O coherent unit that supports AXI4, ACE-Lite and ACE-Lite-E protocols.
  - The socket attached to this unit is an INITIATOR.
  - This unit can contain the Proxy Cache if the protocol is AXI4. There can be zero or more NCAIU units in an Ncore system.
  - NCAIU without a Proxy Cache can operate in coherent or non-coherent mode by setting the NcMode parameter.
- **DCE** (Distributed Coherency Engine) : This unit supports maintaining coherency of the cachelines. There can be one or more DCE units in an Ncore system.
  - If coherent accesses are being made in an Ncore system, however, at least one DCE unit must be present in the system.
  - A DCE does not connect to an external agent and hence it does not have any sockets.
  - Each DCE has one snoop filter by default.
- **DMI** (Distributed Memory Interface) : This unit enables access to the system memory.
  - There can be one or more DMI units per Ncore system.
  - A DMI may optionally contain a cache, called System Memory Cache (CMC), that acts as the last-level cache in-line to memory

  - Multiple DMIs can be accessed in an address interleaved manner using DMI Interleaving.
  - The DMI block offers an AXI4 interface to connect to a memory controller. Hence the socket attached to a DMI has the AXI4 protocol, socket role as MASTER and function as MEMORY.
- **DII** (Distributed IO Interface) : Besides system memory, a system can have additional storage that cannot be, or doesn't need to be, maintained coherently. This is known as peripheral storage.
  - DII unit enables access to peripheral storage. Coherency is not supported for any storage that is accessed via a DII.
  - There can be zero or more DIIs in an Ncore System.
  - The socket attached to a DII has the AXI4 protocol, socket role as MASTER and function as PERIPHERAL.
- **Configuration DII** (sys_dii) : This unit is internally created for the purpose of Configuration network (network to access the Configuration and Status registers within Ncore units).
  - This sys_dii unit is used when the access address falls within the Ncore Register Space.
- **DVE** (Distribute Virtual Engine) : This unit supports accepting DVM messages on the snoop channel.
  - There can be at most one DVE in an Ncore system.

The Ncore units are generated using the command:

```
run_generator name "interface_units" -topology <topology> \
-clock <clocksubdomain> -params <number of directories>
```

The DCEs, DVE and the Configuration DII do not connect to any external socket. They are created automatically using the `run_generator` command mentioned above.

The number of DCEs created is specified by the user in the 'params' options. The clock option associates the given clock_subdomain to the DCEs, DVE and the Configuration DII.

The CAIU, NCAIU, DMI and DIIs will be automatically created with respect to the sockets created in the "Socket" section.

Socket	Socket Function	Interface unit automatically generated
ACE CHI-A CHI-B	INITIATOR	Coherent-Agent Interface Unit (C-AIU)
AXI4 ACE-Lite ACE-Lite-E	INITIATOR	Non-Coherent Agent Interface Unit (NCAIU)
AXI4	MEMORY	Distributed Memory Interface (DMI)

Socket	Socket Function	Interface unit automatically generated
AXI4	PERIPHERAL	Distributed IO Interface (DII)

- Depending on the socket Protocol and Socket Function, the Ncore units will be inferred and created beneath the sockets.
- There should be at least one DCE (the coherency engine with snoop filter).
- By default, one DVE and one Configuration DII (sys_dii) are automatically created along with the above mentioned units.

Example Tcl code snippet:

```
run_generator name "interface_units" \
-topology project/chip/system/subsystem/topology \
-clock project/chip/clkregion/clkdomain/clocksubdomain \
-params [list directories 2]
```

Network Topology

Different network topologies are provided from Maestro to target a specific application and enhance the performance and throughput. See “Automatic Routing” for more information.

Deadlock Analysis

The generated topology should be deadlock free. To check for deadlock:

```
run_checker -name DEADLOCKCHECKER
```

The deadlock can be fixed by specifying the network in which it is present.

Example Tcl code snippet:

```
run_generator -name "fix_deadlock" -topology $topology \
-params [list network ndn]
```

Adapters

Width Adapters, Rate Adapters, and Clock Adapters, which are used to bridge different widths or frequencies of Network elements, are automatically inserted using the `run_generator` command.

The CHI async adapters are also inserted automatically. CHI Asynchronous adapters are inserted in cases where the frequency of the CHI interface is greater than 1.6Ghz. In such instances, the unit CAIU runs at the network frequency and the CHI asynchronous adapter is inserted between the CHI socket and the CAIU unit. Adapters can be optionally inserted using the command given below.

Example Tcl code snippet:

```
run_generator -name adapters -topology $topology
-clock $default_clock
```

See “insert_adapter” example 3 on how to manually insert one adapter into a network.

Configuration Network

The CSR network is needed to modify the registers in the Ncore Units. Each Ncore unit has an APB interface. This APB interface can only be used to perform Read and Write accesses to the registers inside that unit.

The APB interface (also known as Configuration port) of every Ncore unit, including the configuration DII (instance name: sys_dii), will connect to an APB target ATU (Arteris Translation Unit). The AXI4 socket of the configuration DII (instance name: sys_dii) also connects to the AXI4 initiator ATU. The initiator ATU initiates the request on the CSR request network and configures the value in the CSR of the Ncore unit. The target ATU receives the request, configures the Ncore unit, and then sends the response on the CSR response network.

Example Tcl code snippet:

```
run_generator -topology $topology -name "csr" \
-clock $clocksubdomain
```

Interrupt Signals

The design now needs connectivity of its interrupt signals, which is automatically done. The interrupt signals are created by calling the `run_generator` command.

Example Tcl code snippet:

```
run_generator -topology $topology -name "interrupt" \
-clock $clocksubdomain
```

Resiliency

Resiliency can be enabled on the object topology.

Example Tcl code snippet:

```
set_attribute -object $topology -name resilienceEnabled \
-value true
```

Unit duplication is optional.

Example Tcl code snippet:

```
set_attribute -object $topology -name duplicationEnabled \
-value true
```

Each external interface of Interface Unit can be specified to support a protection of its data payload. The protection schemes supported are:

- NONE
- PARITY
- SECDED

All ports of the unit implement the same protection.

Example Tcl code snippet:

```
set_attribute -object $topology -name resiliencyProtectionType \
-value SECDED
```

Enabling Placeholder Signals

There is also an option for the user to specify that the SECDED logic is contained into a placeholder inside the Ncore unit.

Example Tcl code snippet:

```
set_attribute -object $topology -name nativeIntfProtEnabled \
-value true
```

Using placeholders will create additional input to the Ncore unit. In Maestro this is done by creating generic ports to the Ncore unit.

Type	Parent Type	Parameters
generic_port	CAIU NCAIU DMI DII	wireRtlPrefix wireWidth wireName wireDirection

Example Tcl code snippet:

```
set gp [create_object -type generic_port \
-parent project/chip/system/subsystem/topology/cni_caiu/caiu \
-name "placeholder"]
set_attribute -object $gp -name wireRtlPrefix \
-value "caiu_rtl_prefix"
set_attribute -object $gp -name wireWidth -value 4
set_attribute -object $gp -name wireName -value "sig1"
set_attribute -object $gp -name wireDirection -value IN
```

If the design has the Resiliency feature enabled, the necessary stitching can be performed with the Automation.

Example Tcl code snippet:

```
run_generator -topology $topology -name "resiliency" \
-clock $clocksubdomain
```

Quality of Service (QoS)

Maestro supports QoS management for Ncore systems. Qos can be enabled in the system by setting the qosEnabled parameter on the object topology.

Example Tcl code:

```
set_attribute -object $topology -name qosEnabled -value true
```

The QOS value on native interface is mapped to QOS bucket (priority) mapping function. The mapping is provided through predetermined design time parameters. Ncore system supports 16 buckets (priority).

Example Tcl code:

```
set_attribute -object $topology  -name qosMap \
-value_list [list 16’h0780, 16’h0030, 16’h0018, 16’h0006, 16’h0001]
```

The reset value for starvation event counter is 16. The event threshold can be changed by setting the qosEventThreshold parameter on the topology.

Example Tcl code:

```
set_attribute -object $topology -name qosEventThreshold -value 16
```

System Parameters

All the parameters that are global to the entire system should be entered.

Number of General Purpose Address Regions

The ARTERIS_COHERENT susbsystem supports multiple number of address regions for general purpose. The number of regions required must be specified on the topology. These regions can be configured via CSR access.

Example Tcl code snippet:

```
set_attribute -object $topology -name nGPRA -value 6
```

Memory Protection

The ARTERIS_COHERENT susbsystem supports multiple storage structures like proxy cache, system cache, snoopfilter memories. They are protected using the protection schemes:

- NONE
- PARITY
- SECDED

Example Tcl code snippet:

```
set_attribute -object $topology -name memoryProtectionType \
-value SECDED
```

Proxy Cache

The NCAIU with an AXI4 protocolType can be configured with a proxy cache. Proxy Cache must be set before creating routes as it determines what routing paths are needed.

Example Tcl code snippet:

```
set_attribute -object \
project/chip/system/subsystem/topology/cni_ncaiu/ncaiu0 \
-name hasProxyCache -value true
```

System Memory Cache

The DMI can be configured with a System Memory Cache.

Example Tcl code snippet:

```
set_attribute -object \
project/chip/system/subsystem/topology/cni_dmi/dmi0 \
-name hasSysMemCache -value true
```

Snoopfilters

The DCE can be configured with snoopfilters. All the DCE must be configured with the same number of snoopfilters. By default, each DCE contains one snoopfilter with the snoopfilter index `0`.

```
set_attribute -object \
project/chip/system/subsystem/topology/ncc_dce/dce0 \
-name snoopFilters -value 1
```

Each caching agent must be assigned to one and only one snoopfilter.

Example Tcl code snippet:

```
set_attribute -object $caiu -name snoopFilterAssignment -value 0
```

Checklist for Phase 3

State	Title	Necessary completed actions
Solution state	Initialize the topology	Topology is created
	Check parameters for subsystem	Topology -> coherentTemplate
	Check Units	At least one DCE is created in case of Arteris Coherent subsytem
		All units are bound to clock subdomain
		Auto generate units from generator
		Select network topology
		Assign node locations to units in the network topology
		Generate routes in the selected network topology using generator
	Check Data and Control networks	Networks are auto generated
		Packet ports are auto generated
	Check CSR networks	Auto generate CSR network using generator:
		csr_request_nw exists
		csr_response_nw exists
	Check Interrupt accumulator	Auto generate interrupt ports
	Check global parameters	Topology -> nGPRA
		Topology -> memoryProtectionType
	Check unit parameters	NCAIU -> hasProxyCache
		DMI ->hasSysMemCache
		DCE -> snoopFilters
		CAIU -> snoopFilterAssignment
		NCAIU with ProxyCache -> snoopFilterAssignment

2.2.4 Phase 4: Mapping

In this phase of the design flow, Maestro associates specific RTL instances and detailed parameters to the design. The new parameters become accessible simply by clicking on the same units that had been previously populated.

The RTL instances are configured based on all the parameters and structural properties specified to date; for example, in this phase of the flow, the routing tables are populated, based on the information about units, address maps, connectivity and routes.

The map design is done on the object topology. If `map_design` failed, the command will return a "`0`" value. User should not proceed if `map_design` failed.

To check for the status:

```
set mapped [map_design -name $topology]
if {$mapped == 0} {
  puts "ERROR: Mapping has failed"
  exit
}
```

Example Tcl code snippet:

```
map_design -name $topology
```

2.2.5 Phase 5: Refinement

In this phase of the design flow, you can make changes to the existing architecture. The post-map parameters are set in this phase.

Setting of post map parameters (Ncore units, ATUs) is done in the "Maestro Parameters" chapter.

2.2.6 Phase 6: Export

Save the Project

The project can be saved using the command:

```
save_project -file "$designName.postmapping.mpf"
```

The `.mpf` extension stands for "Maestro Project File."

If the file exists, the command will not overwrite the file and project is not saved. To over-write the file, you need to add the "-force" option:

```
save_project -force -file "$designName.mpf"
```

This file format can be loaded onto the Maestro GUI and the configuration can be visualized and modified as required. Note that the `.mpf` file is encrypted.

Generate RTL

In this phase of the design flow, Maestro creates the RTL and all the required collaterals for the design. These can be triggered using the command:

```
gen_collateral -file designName.tgz
```

3 Routing

Routing is part of Phase 3 design creation. Routing the design can begin after the topology and Ncore units have been created. Maestro offers two types of routing:

- "Automatic Routing" — user may choose a predefined topology
- "Manual Routing" — the user creates the connections

3.1 Automatic Routing

Automatic routing is allowed for the following types of topologies:

- "Mesh Topology"
- "Torus Topology"
- "Ring Topology"
- "DoubleC Topology"
- "Full Crossbar Topology"
- "Butterfly Topology"

3.1.1 Automatic Routing Topology Generator

The Automatic Routing Topology Generator (ARTG) is generated using the following two commands.

```
#Create a topology generator
create_object -type generator -subtype <topology_type> \
-parent <the_object_topology> -name <The_name_of_the_generator>

#Execute the generator
update_object -name < The_name_of_the_generator> \
-parent <the_network> -value_list <list_of_parameters>
```

Options:

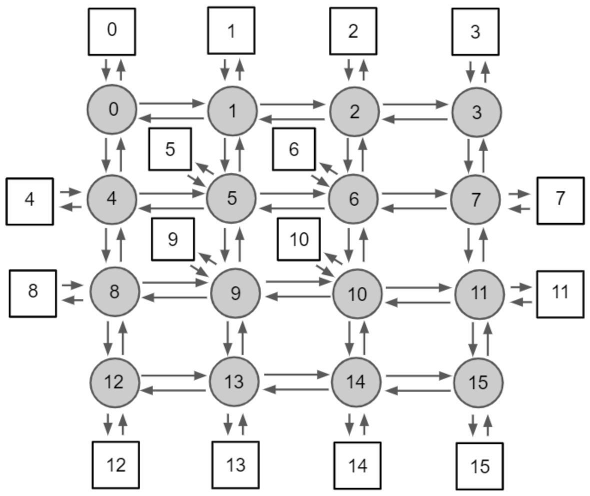
- `params` : Tcl list of name value pairs
- `-name <value>` : must be unique and the same in the `create_object` and `update_object` commands
- `update_object -parent <network to route>` : must the name of the network to route

- `<topology_type>` : one of: mesh, torus, ring, butterfly, full_crossbar, or doubleC
- `<the_object_topology>` : path to the topology. For example: `"project/chip/system/subsystem/topology"`
- `<list_of_parameters>` : list of name value pairs. For example: `[list meshx 4 meshy 2 dataWidth 128 clock $clk method "execute()"]`

3.1.2 Mesh Topology

In this topology, the switches are arranged as a mesh and the routes are generated based on the shortest path from the source to the destination. The switches with 1 input and 1 output port are deleted after the generation of the routes and replaced with a wire in-place.

The size of the mesh can be defined by the parameters 'meshx' and 'meshy'. The bandwidth for routing the data in the data network can be specified by the parameter 'datawidth'. The frequency for routing is defined by the clock subdomain passed in to the parameter 'clock'.



Circles indicate switches, squares indicate the Ncore units

Prior to generating routes, each Ncore unit needs to set a position on how the Ncore unit is attached to the mesh. The command to assign the position is:

```
set_node_position -object $unit -x $x -y $y
```

Maestro allows you to specify which node of the topology each Ncore unit should be associated with. If the units have the same position for all networks, this setup is needed before creating the first topology. For example:

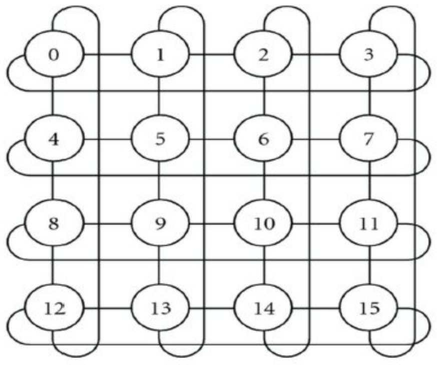
```
set_node_position \
-object project/chip/system/subsystem/topology/cni_caiu/caiu0 \
-x 3 -y 2
```

Example Tcl code snippet:

```
#Create a generator of subtype mesh of size 4x2 under the
#parent topology
#Update the mesh generator with parameters
set meshx 4
set meshy 2
set datawidth 256
set clkSubDomain project/chip/clk_region/clk_domain/clk_subdomain
set params [list meshx $meshx meshy $meshy dataWidth \
$datawidth clock $clkSubDomain -type mesh]
run_generator -name "regular_topology" -topology $topology \
-clock $clkSubDomain -params $params
```

3.1.3 Torus Topology

The parameters of torus are similar to the parameters of mesh. The size of the torus can be defined by the parameters ‘meshx’ and ‘meshy’. The frequency for routing is defined by the clock subdomain passed in to the parameter ‘clock’.



Circles indicate switches

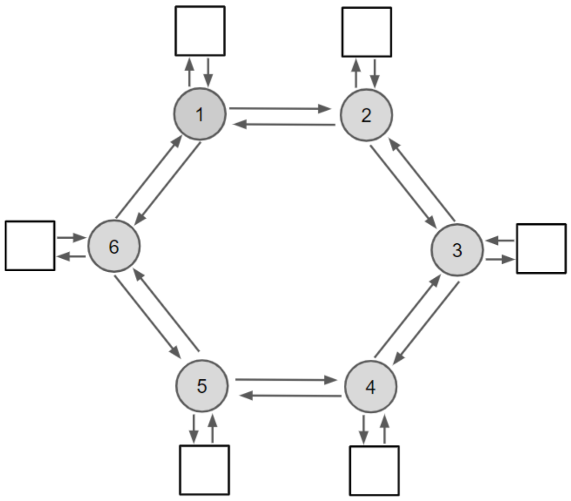
Prior to creating the torus topology, each Ncore unit needs to assign a position as in the mesh topology.

Example Tcl code snippet:

```
#Create a generator of subtype mesh of size 4x4 under
#the parent topology
#Update the torus generator with parameters
set meshx 4
set meshy 4
set datawidth 256
set clkSubDomain project/chip/clk_region/clk_domain/clk_subdomain
set params [list meshx $meshx meshy $meshy dataWidth $datawidth \
clock $clkSubDomain -type torus]
run_generator -name "regular_topology" -topology $topology \
-clock $clkSubDomain -params $params
```

3.1.4 Ring Topology

Here the switches are arranged as a form a ring. Deadlock can be resolved by the use of Virtual Channels. The user will need to set_node_position for each unit as in the x value to specify the order of the ring. The y value is set to `0`.



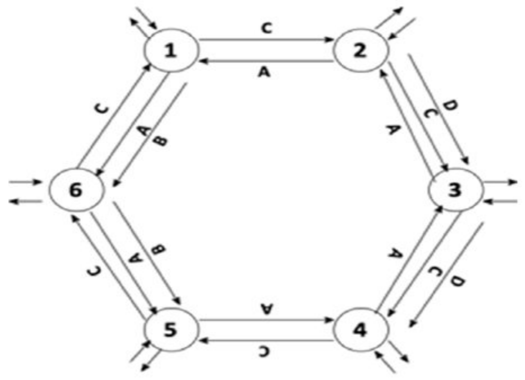
Circles indicate switches, squares indicate the Ncore units

Example Tcl code snippet:

```
#Creating a ring of size 6.
set meshSize 6
set meshx $meshSize
set meshy 1
set datawidth 256
set clkSubDomain project/chip/clk_region/clk_domain/clk_subdomain
#Create the ring generator
#Set the parameters
set params [list meshx $meshx meshy $meshy dataWidth \
$datawidth clock $clkSubDomain -type ring]
#Update the generator with the parameters
run_generator -name "regular_topology" -topology $topology \
-clock $clkSubDomain -params $params
```

3.1.5 DoubleC Topology

Here the switches are arranged as to form a ring. Deadlock is avoided by having parallel routes. This topology also enforces that the y coordinate is always set to `0`.



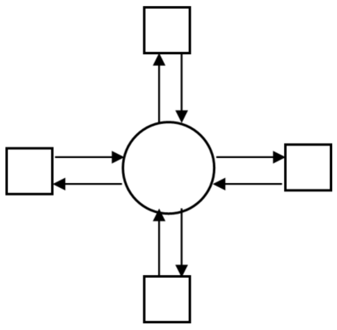
Circles indicate switches

Example Tcl code snippet:

```
#Creating a ring of size 6.
set meshSize 6
set meshx $meshSize
set meshy 1
set datawidth 256
set clkSubDomain project/chip/clk_region/clk_domain/clk_subdomain
#Create the doubleC generator
#Set the parameters
set params [list meshx $meshx meshy $meshy dataWidth \
$datawidth clock $clkSubDomain -type doubleC]
#Update the generator with the parameters
run_generator -name "regular_topology" -topology $topology \
-clock $clkSubDomain -params $params
```

3.1.6 Full Crossbar Topology

This topology consists of a single switch. All ports of all units (of the same network) will be connected to this switch.



Circles indicate switches, squares indicate the Ncore units

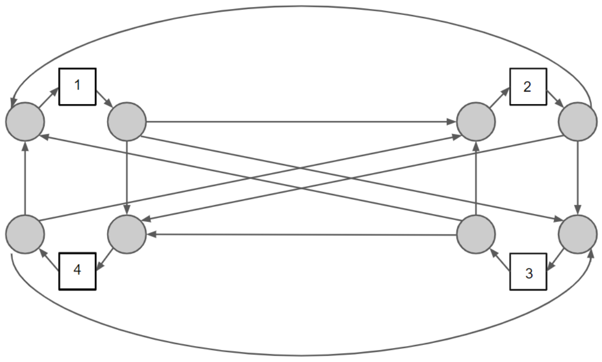
Example Tcl code snippet:

```
#Create the full_crossbar generator for network
#Set the parameters
set params [list dataWidth $datawidth clock $clkSubDomain \
-type fullCrossbar]

#Update the generator with the parameters
run_generator -name "regular_topology" -topology $topology -clock
$clkSubDomain -params $params
```

3.1.7 Butterfly Topology

This topology consists of each unit connected to its own switch. The switch is connected to the other switches on the network. Therefore, each connection between each unit is connected by two switches.



Circles indicate switches, squares indicate the Ncore units

Example Tcl code snippet:

```
#Create the butterfly generator for network
#Set the parameters
set params [list dataWidth $datawidth clock $clkSubDomain \
-type butterfly]

#Update the generator with the parameters
run_generator -name "regular_topology" -topology $topology -clock
$clkSubDomain -params $params
```

Options:

- `params` is a tcl list of name value pairs.
- `-name <value>` must be unique and the same in the `create_object` and `update_object` commands.
- `update_object -parent <network to route>` must the name of the network to route.

3.2 Manual Routing

3.2.1 Steps for Creating Manual Routes in Maestro

Manual routing allows you to specify how to connect the ports on each network.

- A route is created by specifying a path from the port on the source Ncore unit to the port on the target Ncore unit connected via the ports on the switches.
- The ports on the Ncore unit are automatically generated with Ncore units.
- Each port of the Ncore unit has a specific role to send and receive specific protocol messages across the network.

1. Access the required port on the Ncore unit.

Example Tcl commands to find the packet ports of the Ncore units:

```
#Get the output and input ports of caiu0
set caiu1 [get_objects -type caiu -parent $topology]
set caiu1_outports [get_objects -parent $caiu1 -type packet_port \
-subtype out]
set caiu1_inports [get_objects -parent $caiu1 -type packet_port \
-subtype in]
```

For a two control and one data network topology, the Ncore units of type CAIU, NCAIU, DMI and DII have 3 input and output ports each. The Ncore unit of type DMI has 4 input and output ports.

The usernames of the output ports for the caiu1 unit are:

```
project/chip/system/subsystem/topology/cni_caiu1/caiu1_pkt0Z0
project/chip/system/subsystem/topology/cni_caiu1/caiu1_pkt1Z0
project/chip/system/subsystem/topology/cni_caiu1/caiu1_pkt2Z0
```

The usernames of the input ports for the caiu1 unit are:

```
project/chip/system/subsystem/topology/cni_caiu1/caiu1_dpkt0I0
project/chip/system/subsystem/topology/cni_caiu1/caiu1_dpkt1I0
project/chip/system/subsystem/topology/cni_caiu1/caiu1_dpkt2I0
```

2. Access the networks and set up the first route on the network.

Example Tcl commands to access a single network from the list of networks under the topology:

```
set networks [get_objects -parent $topology -type network]
set ndn [lindex $networks 0]
```

3. Create switches.

The switches must be explicitly created by the user. In this example, two switches will be created in the ndn network:

```
#Create two switches on network0: sw1, sw2
set sw1 [create_object -parent $ndn -type "switch" -name "sw1"]
set sw2 [create_object -parent $ndn -type "switch" -name "sw2"]
```

4. Specify the source and destination packet ports of the route.

In this example, the source is port0 of the caiu1 and the destination is the port0 of dmi1.

The username for the source port is:

```
project/chip/system/subsystem/topology/cni_caiu1/caiu1_pkt1Z0
```

The username for the destination port is:

```
project/chip/system/subsystem/topology/cni_dmi1/dmi1_dpkt1I0
```

Example Tcl commands to set the source and destination ports:

```
#Set source and target ports
set srcPort \
project/chip/system/subsystem/topology/cni_caiu1/caiu1_pkt1Z0
set tgtPort \
project/chip/system/subsystem/topology/cni_dmi1/dmi1_dpkt1I0
```

The port index of both units should be the same because the ports must be on the same network. Note: Route cannot be created if outputPort is on `network0` and inputPort is on `network1` (different networks).

Each packet port can have only one connection. Let's assume there is a connection from `sw0` port `Z0` to `sw1` port `I0`. There cannot be a connection from `sw2` port `Z0` to `sw1` port `I0` because `sw1` port `I0` is already connected. `sw2` port `Z0` needs to connect to `sw1` port `I1`.

Different routes can share the same connection. That is, `route 1` has a connection between `sw0` port `Z0` to `sw1` port `I0`. — `route 2` can also go through `sw0` port `Z0` to `sw1` port `I0`.

5. Create the route using the `update_object` command.

The type option specifies the data width of the packet port of the switch. The data network should have a positive value for this option whereas the value of datawidth for other networks is zero.

```
set route_connection [list $srcPort $sw1 $sw2 $tgtPort]
set width 0
```

Example Tcl code to create a route using the `update_object` command:

```
update_object -model route -type $width -add $route_connection
```

3.2.2 Packet Serialization on the Packet Routes

Packet serialization specifies how the data packets are sent through the network. The data may be sent in serial or parallel. If no setting is made, the default is HEADER_SERIAL.

This setting can be set globally at the network level or on each packet port. At the network level, the Tcl command is:

```
set_attribute -object $network -name defaultPacketSerialization \
-value $value
```

where value is HEADER_SERIAL, HEADER_PARALLEL, or FIXED_SERIAL.

When set at the network level, all routes will be using this type of serialization.

The user may further modify the setting at the packet port of the switch and adapters. The Tcl command is:

```
set_attribute -object $packetPort -name packetSerialization \
-value $value
```

Note: If you modify the packet serialization setting, you should call the `insert_adapter` Tcl command to ensure the proper conversion from the different styles by inserting width adapters as needed.

```
insert_adapter -type auto
```

3.2.3 Manual Route Design Example

See “Complete Manual Route Design Example” in the appendix.

4

Maestro Tcl Commands

This chapter describes the Maestro Tcl commands:

- "Tcl Command Summary"
- "Tcl Commands"

4.1 Tcl Command Summary

Table 1. Maestro Tcl Command Descriptions

Tcl Command	Description
`check_object`	Runs a sanity check on a given Maestro object.
`clone_socket`	Creates a copy of selected socket.
`close_project`	Closes the currently open Maestro project.
`connect_network`	Performs initial connectivity for manual editing.
`create_checker`	Creates and configures Maestro checker.
`create_object`	Creates an object in the design hierarchy tree.
`create_project`	Create a new project in Maestro.
`delete_custom_attribute`	Removes the named Custom Attributes from the named object.
`delete_object`	Deletes the named object.
`gen_collateral`	Generates Verilog RTL and the collateral for the complete design.
`get_context`	Returns the name prefix that Maestro is using to identify objects based on the name hierarchy.
`get_license_tokens`	Gets the license tokens available for this project.
`get_node_position`	Gets the (x,y) location of a switch or Ncore unit in a topology used in routing the network.
`get_objects`	Returns the objects of the required type under a given parent.
`get_product_version`	Gets the Maestro version.
`get_routes`	Gets the routes that are going through a packet ports, switches, or adapters.
`get_tasks`	Retrieve the available tasks.
`gui_get_screen_position`	Gets the (x,y) GUI location of a switch, adapter, or Ncore Unit.
`gui_set_screen_position`	Sets the (x,y) GUI location of a switch, adapter, or Ncore Unit.
`help`	Get help for Maestro Tcl commands.
`initialize_routes`	Initializes routes from or to a port, over a given switch.
`insert_adapter`	Inserts adapter between connected network elements.
`insert_switch`	Inserts a switch before or after a component, along certain routes.
`man`	Man pages for Maestro TCL commands.
`map_design`	Operates on a single object or on a whole connected design and casts it onto specific instances of the Arteris IP library.
`merge_switch`	Merge one or more switches into another.
`open_project`	Opens a previously-saved project.
`query_attribute`	Reports the value(s) of the named attribute belonging to the named object.
`query_custom_attribute`	Reports the value(s) of the named custom attribute belonging to the named object.
`query_object`	Check the validity of the object.

Table 1. Maestro Tcl Command Descriptions

Tcl Command	Description
`query_project`	Reports if a project is open.
`redo`	Redoes the previous command.
`rename_object`	Rename an object.
`report_object`	Reports information about the named object.
`report_synthesis_options`	Reports information about an object.
`run_checker`	Invokes and executes a Maestro checker.
`run_generator`	Invokes and executes a Maestro generator.
`run_task`	Runs a Maestro task.
`save_project`	Saves to disk the currently open project.
`set_attribute`	Either select the required template, or sets the value of one or more parameters on the named object.
`set_context`	Sets the path name of an object.
`set_custom_attribute`	Sets the value of one or more Custom Attributes.
`set_node_position`	Sets the (x,y) location of a switch or Ncore unit in a topology used in routing the network.
`set_project_name`	Set new project name.
`set_synthesis_options`	Set synthesis options.
`split_switch`	Split an MxN switch into an Mx1 and an 1xN switch.
`undo`	Undoes the previous command.
`update_object`	Updates the state of the object.

4.2 Tcl Commands

4.2.1 check_object

Runs a sanity check on a given Maestro object.

Usage

```
check_object -name <name of the Maestro object>
```

Options

-name <name> : Object name, type: string

Example

```
check_object -name project/node1/node2
```

4.2.2 clone_socket

Creates a copy of selected socket.

Usage

```
clone_socket -name <path_name>
```

Options

-name <name> : Path to the socket to be cloned.

Example

```
clone_socket -name project/chip/system/subsystem/socket
```

4.2.3 close_project

Closes the currently open Maestro project, if any. Projects with unsaved changes cannot be closed, unless the -force option is used.

Usage

```
close_project
```

Options

[-force] : Force the closing of an unsaved project

Example

```
close_project
close_project -force
```

4.2.4 connect_network

Performs initial connectivity for manual editing.

Usage

```
connect_network -name <name>
```

Options

-name <name> : name of the Network

Example

```
connect_network -name $mynetwork
```

4.2.5 create_checker

Creates and configures Maestro checker.

This command creates a Maestro checker and gives it a name specified by -name. These checkers types are currently available:

- 'data' - a generic sanity check on the project data 'deadlock' - a deadlock freedom checker
- 'packetroutes' - a specific checker for packet route sanity
- 'dependencies' - an advanced checker for parameter consistency.

Usage

```
create_checker -name <name> -type <type>
```

Options

-name <name> : name of the checker to create

-type <type> : type of the checker to create

Example

```
create_checker -name $mychecker -type data
```

4.2.6 create_object

This creates an object in the design hierarchy tree, once the project is created. When an object is created, Maestro will attempt to create it with information about its type. An object should have a qualified name which needs to be unique in the project. Objects are hierarchical. This hierarchy represents relationships between the object and its children. Changes to information typically propagates from parents to children. Hence it is imperative to have the correct parent.

Table 2. create_object objects created

Parent (type of parent object)	Type	Object created
chip	clock_region	clock region under the chip
chip	memory_map	memory map under the chip
chip	power_region	power region under the chip
chip	system	system under the chip
clock_domain	clock_subdomain	clock subdomain under clock domain
clock_region	clock_domain	clock domain under the clock region
memory_map	boot_region	boot region under the memory map
memory_map	configuration_region	Configuration region under the memory map (CSR region)
memory_map	memory_set	memory sets in memory map
memory_set	dynamic_memory_group	memory groups under the memory set
network	switch	switches under the network
power_region	power_domain	power domain under the power region
subsystem	flow	flow in the chip (Symphony only)
subsystem	multi_flow	multi-flows in the chip (Symphony only)
subsystem	socket	socket under the subsystem
subsystem	topology	topology under the subsystem
system	subsystem	subsystem under the system
topology	firewall_adapter	fire wall adapters
topology	pipe_adapter	pipe adapter

Usage

```
create_object -type <type> -name <object_name> \
-parent <parent_name>

create_object -type <type> -group <list of values> \
-parent <parent_name>
```

Options

-name [name of an individual object]

-group [names of groups of objects. e.g.: list of switches, packetizers or depacketizers, memorygroups]

Example 1

```
set project [create_object -type project -name "dry_run"]
set chip [create_object -type chip -name "chip" -parent $project]
set system [create_object -type system -name "system" \
-parent $chip]
```

All the above are examples where we create just one unit of each type.

Example 2

```
create_object -parent $subsystem -type socket -group  \
[list sk0 sk1 sk2]
```

This command creates a list of sockets under the subsystem with names in the list.

Example 3

```
create_object -type memory_set -parent $memorymap \
-group [list ms0 ms1 ms2]]
```

This command creates a group of three memory sets with names `ms0`, `ms1` and `ms2`.

4.2.7 create_project

This command is used to create a new project in Maestro. This command creates a new, empty project using the specified name and optional path. Existing open projects should be closed prior to executing this command. If an existing project is detected a new project is not created. Instead, an error is issued. The license_token is validated with the Maestro license server against authorized license_tokens.

Usage

```
create_project -name <name_of_project>
```

Options

-name <name> : Name of the project, type: string

[-path <path>] : Path used for project data, type: string, default: "./"

[-license_token <license_token>] :
    default: "Arteris/Dev", choices: {Arteris/Dev, Arteris/Other, dryrun, hw_config_two, nxp_cfg_1}

Example

```
create_project -name myproj -path xyz \
-license_token arteris_project_1
```

4.2.8 delete_custom_attribute

This command removes the named Custom Attributes from the named object, if it exists.

Usage

```
delete_object -object <name of the object> \
-name <custom attribute name>
```

Options

-object <object> : Name of the object from which the Custom Attribute is to be removed, type: string

-name <name> : Name of the Custom Attribute to be removed, type: string

Example

```
delete_object -name project/chip0
```

4.2.9 delete_object

Deletes the named object.

Usage

```
delete_object -name <name>
```

Options

-name <name of existing object>

Example

```
delete_object -name project/chip0
```

4.2.10 gen_collateral

Generates Verilog RTL and the collateral for the complete design. It launches the Maestro server which will generate a tarball with synthesizable design RTL along with collateral such as ipXact, testbench and synthesis scripts

Usage

```
gen_collateral
```

Options

[-file <file>] : Path to a tarball file where all results will be held. Default: <date_time>.tgz, ex: Mar05_2020_164847.tgz

[-server_exec <server_exec>] : Full path to dedicated server executable, default: maestro-server found in UNIX $PATH, default: "maestro-server"

[-port_range <port_range>] : Look for available port on local host through which Maestro client and server will communicate, default: "50051:50099"

[-server_run_dir <server_run_dir>] : Directory where Maestro server will run. Must be writeable, default: "auto"

[-server_log_file <server_log_file>] : Redirect server stdout and stderr to this file for debug, default: "./maestro-server.log"

Example

```
gen_collateral -file output.tgz
```

4.2.11 get_context

This command returns the name prefix that Maestro is using to identify objects based on the name hierarchy. Objects are accessible by hierarchical name, and names passed to TCL commands are interpreted as either the passed name (if starting from the project root) or the concatenation of the context plus the passed name.

Usage

```
get_context
```

Example

```
set_context $chip
get_context
project/chip
```

The output shows the context that was set.

4.2.12 get_license_tokens

Gets the license tokens available for this project.

Usage

```
get_license_tokens
```

Example

```
get_license_tokens
```

4.2.13 get_node_position

Gets the (x,y) location of a switch or Ncoreunit in a topology used in routing the network.

Usage

```
get_node_position -object <object>
```

Options

-object : a switch or Ncore unit

Example

```
set pos [get_node_position \
-object project/chip/system/subsystem/topology/cni_dmi0/dmi0]
set x [lindex $pos 0]
set y [lindex $pos 1]
```

This returns the x, y positions of the Ncore object `dmi0`.

4.2.14 get_objects

This command returns the objects of the required type under a given parent.

Usage

```
get_objects -parent <object_name> -type <type_of_object> \
-subtype <particular_option>
```

Options

Table 3. get_objects Return Objects

Parent (type of parent object)	Type	Subtype	Return Objects
<any object>	parent		parent of the object
adapter(clock|pipe|width)	packet_port	in/out/all	packet_ports inside adapter
AdmProjectRoot	chip		name of the chip object
atu	clock_domain		clock domain of the atu
atu	clock_region		clock region of the atu
atu	clock_subdomain		clock subdomain of the atu
atu	interrupt_port		interrupt port of the atu
atu	network		network of the atu
atu	packet_port		packet ports of the atu
atu	socket		corresponding socket of the atu
chip	clock_domain		clock domains under the chip
chip	clock_region		clock regions under the chip
chip	clock_subdomain		clock subdomains under the chip
chip	flow		flows in the chip
chip	memory_map		memory map under the chip
chip	power_region		power regions under the chip
chip	socket		sockets under the chip
chip	system		system under the chip
clock_domain	clock_subdomain		clock subdomains under clock domain
clock_domain	power_domain		power domain belonging to clock domain
clock_region	clock_domain		clock domains under the clock region
clock_region	power_region		power region belonging to the clock region
dynamic_memory_group	socket		sockets in the dynamic memory group
FirewallAdapter	generic_port		
FirewallAdapter	interrupt_port		
flow	socket	source	source socket of the flow

Table 3. get_objects Return Objects

Parent (type of parent object)	Type	Subtype	Return Objects
flow	socket	destination	destination socket of the flow
internal_memory	generic_port		generic port of the internal memory
interrupt_accumulator	interrupt_port		interrupt port of the internal memory
link_segment	packet_port		upstream and downstream ports
mapped_flow	packet_port		packet ports of mapped flow
memory_map	boot_region		boot region of the memory map
memory_map	configuration_region		CSR region under the memory map
memory_map	memory_interleave_funct ion		memory interleave functions of the memory map
memory_map	memory_region		memory regions of the memory maps
memory_map	memory_set		memory sets in memory map
memory_map	peripheral_region		peripheral region of the memory map
memory_map	static_memory_group		static memory group of the memory map
memory_set	dynamic_memory_group		memory groups under the memory set
multi_flow	socket	source	source socket of the multi_flow
multi_flow	socket	destinations	destination sockets of the multi_flow
multicast_station	clock_domain		clock domain of the multi-cast station
multicast_station	clock_region		clock region of the multi-cast station
multicast_station	clock_subdomain		clock subdomain of the multi-cast station
multicast_station	packet_port		packet ports of the multi-cast station
ncoreunit	configsocket		config socket of ncore unit
ncoreunit	internal_memory		list of internal memories in that unit
network	switch		switches inside Network
packet_port	clock_subdomain		clock subdomain of the packet port
packet_port	link_segment		link segment attached to packet port
packet_port	network		network which packet port is on
packet_port	packet_route		packet route which the packet port is on
packet_route	packet_port		all the packet ports in the packet route
power_region	power_domain		power domain belonging to power region
snoop_filter	internal_memory		internal memory of the snoop filter
socket	clock_subdomain		clock subdomain of the socket
socket	ncore_unit		Ncore unit of the socket
subsystem	ClockPort		
subsystem	flow		
subsystem	memory_map		memory map of the subsystem

Table 3. get_objects Return Objects

Parent (type of parent object)	Type	Subtype	Return Objects
subsystem	multi_flow		Multi-flow of the subsystem
subsystem	socket		sockets under the subsystem
subsystem	topology		topology under the subsystem
subsystem	use_case		
switch	network		network of the switch
switch	packet_port		all the ports of the switch
switch | ncoreunit	packet_port	in	the input ports of the switch
switch | ncoreunit	packet_port	out	the output ports of the switch
system	subsystem		subsystem under the system
topology	atu		atu ncore units
topology	caiu		caiu ncore units
topology	clock_adapter		clock adapters
topology	dce		dce ncore units
topology	dii		dii ncore units
topology	dii	configDii	config dii ncore units
topology	dmi		dmi ncore units
topology	dve		dve ncore units
topology	FifoAdapter		fifo adapters in the topology
topology	firewall_adapter		fire wall adapters
topology	generic_port		generic port on the topology
topology	grb		grb ncore uints (global register block)
topology	interrupt_accumulator		interrupt accumulators
topology	interrupt_port		interrupt ports
topology	mapped_flow		mapped flows
topology	multicast_station		multi-cast station in the topology
topology	ncaiu		ncaiu ncore units
topology	ncore_unit		caiu, ncaiu, dmi, dii, dce, dve ncore units
topology	NcoreCXIU		
topology	network		networks under the subsystem
topology	packet_route		routes of the design
topology	pipe_adapter		pipe adapter
topology	rate_adapter		rate adapter
topology	snoop_filter		snoop filters in the topology
topology	socket		sockets in the topology

Table 3. get_objects Return Objects

Parent (type of parent object)	Type	Subtype	Return Objects
topology	switch		switches under the subsystem
topology	width_adapter		width adapter

Examples

```
get_objects -parent project/chip/topology/cni_dmi -type packet_port
```

This returns all the packet_ports connected to the DMI Coherent network interface

```
get_objects -parent $chip -type clock_subdomain
```

This returns all the objects of type clock_subdomain in the chip.

4.2.15 get_product_version

Gets the Maestro version.

Usage

```
get_product_version
```

Example

```
get_product_version
```

4.2.16 get_routes

Gets the routes that are going through a packet ports, switches, or adapters.

Gets the routes that are going through a packet ports, switches, or adapters. "from", "-to", and "-through" are optional, but at least one must be specified.

Usage

```
get_routes -from <port1> -to <port2> -through [list portA portB]
```

Options

[-from <from>] : route starting from packet port

[-to <to>] : route terminating to packet port

[-through <through>] : going through the list of packet ports, switches, or adapters

Example

```
get_routes -from port1 -to port2 -through [list portA portB]
```

4.2.17 get_tasks

Retrieves the list of available tasks, identified by name.

Usage

```
get_tasks
```

Example

```
get_tasks
```

4.2.18 gui_get_screen_position

Gets the (x,y) GUI location of a switch, adapter, or Ncore Unit in a topology used in routing the network.

Usage

```
gui_get_screen_position -object <object_name> -view <network_name>
```

Options

-object <object> : name of object to set

[-view <view>] : name of network

Example

```
gui_get_screen_position -object project/chip/system/subsystem/
topology/cni_caiu0/caiu0 -view project/chip/system/subsystem/
topology/ndn
```

4.2.19 gui_set_screen_position

Sets the (x,y) GUI location of a switch, adapter, or Ncore Unit in a topology used in routing the network.

Usage

```
gui_set_screen_position -object <object_name> -view <network_name>
-value_list [list]
```

Options

-object <object> : name of object to set

[-view <view>] : name of network

[-value_list <value_list>] : list of values: x $xval y $yval z $zval

Example

```
gui_set_screen_position -object project/chip/system/subsystem/
topology/cni_caiu0/caiu0 -view project/chip/system/subsystem/
topology/ndn -value_list [list x 2000 y 9000 ]
```

4.2.20 help

Help for Maestro TCL commands.

Usage

```
help -category <category> -show_category -brief -verbose -man_page
<name>
```

Options

[-category <category>] : list command by category

[-show_category] : show available categories

[-brief] : show brief command help

[-verbose] : show command arguments

[-man_page] : show full command help

[<name>] : command of interest

Examples

```
help get_routes
help -verbose insert_switch
```

4.2.21 initialize_routes

If the given port, specified with either -from or -to but not both, generates or consumes traffic which has not been routed yet, this automation initializes routes for it. The automation routes the traffic over the named Switch.

Usage

```
initialize_routes -from <port_name> -to <port_name> -via
<switch_name>
```

Options

[-from <from>] : source port of the traffic

[-to <to>] : destination port of the traffic

-via <via> : name of a Switch over which to route the traffic

Example

```
initialize_routes -to $port -via $sw2
```

4.2.22 insert_adapter

Inserts adapter between connected network elements (link direction doesn't matter). Returns a created adapter object.

Usage

```
insert_adapter -from <switch> -to <switch> -type clock_adapter
-name myClockAdapter
```

Options

[-from <from>] : Full path to the first unit, type: string, default: ""

[-to <to>] : Full path to the second unit, type: string, default: ""

[-port <port>] : Port to insert before (input port) or after (output port) the adapter, type: string, default: ""

[-type <type>] : Type of the adapter to insert, default: "none", choices: {width_adapter, clock_adapter, rate_adapter, pipe_adapter, fifo_adapter}

[-name <name>] : Adapter instance name, type: string, default: ""

Example 1

```
set adapter [insert_adapter -name $adapterName \
-type width_adapter -port $pp]
```

The $pp is a packet port. If the packet port is an input port, the adapter is inserted in front of the port. If packet port is an output port, the adapter is inserted after the port.

The $adapterName is the name of the adapter.

The return value is the adapter object created.

4.2.23 insert_switch

Inserts a newly-created Switch either in front of the inputs of the component specified with -before, or in front of the outputs of the component specified with -after or in the link segment specified with -on. Only one of -before and -after and -on can be passed. Optionally, -along specifies a list of routes across the existing component, and restricts the insertion to only occur along the selected routes. This only works when inserting switch providing -before or -after options. If -along is not specified, a new Switch is inserted in front of each input or output port of the component.

Usage

```
insert_switch -on <name> -before <name> -along <list> -on <name>
```

Options

[-before <before>] : insert before named component

[-after <after>] : insert after named component

[-along <along>] : optional list of routes along which to perform the operation

[-on <on>] : insert on named link segment

Example

```
insert_switch -on $myLinkSegmentName
```

4.2.24 man

Man pages for Maestro TCL commands.

Usage

```
man -category <name> -show_category -brief -verbose -man_page
<name>
```

Options

[-category <category>] : list command by category

[-show_category] : show available categories

[-brief] : show brief command help

[-verbose] : show command arguments

[-man_page] : show full command help

[<name>] : command of interest

Example

```
man close_project
```

4.2.25 map_design

This command operates on a single object or on a whole connected design, comprising all Ncore units and all transport units, and casts it onto specific instances of the Arteris IP library. The input of the command is a “generic” NoC description, chiefly consisting of logical units, their connectivity, and general-purpose parameters; the output is a synthesized interconnect, consisting of specific instances of Arteris units with detailed parameterization.

The map_design command carries forward the parameterization that is known at mapping time, including parameters set on the pre-mapping units as well as any parameterization that can be inferred from the structure and connectivity of the design. After running the map_design command, the full unit parameterization (see Ncore/Symphony RTL documentation) becomes available.

If any pre-mapping design properties are modified after the map_design command is run, the command needs to be rerun.

Usage

```
map_design -name <name_of_the_object>
```

Options

-name : Name of the object

The design is fully mapped only if the command is run on the whole topology.

Example

```
map_design -name $topology
```

This command is run after the initial description of the solution (i.e., the population of Ncore units and the definition of the transport topology). It processes the given high-level description of the solution and casts it into a specific set of Ncore and Legato units, setting all the parameters of those units that can be inferred from the solution data accrued until this point. After this command, all the RTL units expose their full parameter set that can be further tuned to optimize the quality of results.

4.2.26 merge_switch

Merge one or more given switches into another. Links and routes are adjusted accordingly so that all the traffic that was going through the switches indicated by -name now goes through the switch indicated by -into. All switches need to belong to the same Network.

Usage

```
merge_switch -name <name_list> -into <name>
```

Options

-name <name> : name of one or more Switches to be merged

-into <into> : name of Switch to merge into

Example

```
merge_switch -name [list sw01 sw03] -into sw02
```

4.2.27 open_project

Opens a previously-saved project using the specified file name. Existing open projects should be closed prior to executing this command.

Usage

```
open_project -file <name>
```

Options

-file <file> : name of the project file

Example

```
open_project -file /my/path/to/file
```

4.2.28 query_attribute

Reports the value(s) of the named attribute belonging to the named object. If no argument is passed for -name, all attributes are reported.

Usage

```
query_attribute -object <object_name> -name <name_of_attribute>
```

Options

-object <object> : Object of interest, type: string

[-name <name>] : Attribute to be reported, or empty to see all, type: string, default: ""

Example

```
set valu [query_attribute -object $ts -name nGPRA]
```

The value of the attribute is stored in variable "valu".

4.2.29 query_custom_attribute

Reports the value(s) of the named custom attribute belonging to the named object. If no argument is passed for -name, all attributes are reported.

Usage

```
query_custom_attribute -object <object_name> \
-name <name_of_attribute>
```

Options

-object <object> : Object of interest, type: string

[-name <name>] : Custom attribute to be reported, or empty to see all, type: string, default: ""

Example

```
query_custom_attribute -object $ts -name xyz
```

4.2.30 query_object

This command is used to check the validity of the object. It returns true if the object with the given path exists.

Usage

```
query_object -object <path_to_object> -type <type>
```

Table 4. query_object return values

Object	Type	Return values
<any_object>	type	type of the object
<any_object>	clock	clock subdomain bound to that object
<any_object>	userName	user name of the object
network	dn	true if the network is data network
network	ndn	true if the network is not a data network

Options

-object : Name of the object created in ADM

-type: The type of attribute

Example

```
query_object -type userName -object $chip/$region
```

4.2.31 query_project

Reports if a project is open. Depending on whether a project is open, returns `0` or 1

Usage

```
query_project
```

Example

```
query_project
```

4.2.32 redo

Redoes the next command in the history list, if there is one.

Usage

```
redo
```

Example

```
redo
```

4.2.33 rename_object

Changes the object name.

Usage

```
rename_object -name <name> -new_name <name>
```

Options

-name <name> : object name

-new_name <new_name> : new name

Example

```
rename_object -name project/node1/node2 -new_name node2a
```

4.2.34 report_object

Reports information about the named object. The project root is a valid target, and will print specific information. The report can optionally be saved to file. Any pre-existing file will be overwritten.

Usage

```
report_object -name <object_name> -file <output file>
```

Options

-name <name> : Object to inspect, type: string

[-file <file>] : File to save the report in, type: string, default: ""

Example

```
report_object -name project/chip0 -file /my/file
```

4.2.35 report_synthesis_options

Reports information about the named object. The project root is a valid target, and will print specific information. The report can optionally be saved to file. Any pre-existing file will be overwritten.

Usage

```
report_synthesis_options -file <name>
```

Options

[-file <file>] : (optional) file to save the report in

Example

```
report_synthesis_options -file /my/file
```

4.2.36 run_checker

Invokes and executes a Maestro checker specified with the argument passed to -name. If no name is passed, all the automatically-created and manually created checkers are launched.

Usage

```
run_checker -name <name_of_checker>
```

Options

[-name <name>] : Name of the checker to invoke, type: string, default: ""

Example

```
run_checker -name $mychecker
```

4.2.37 run_generator

This command invokes and executes a Maestro generator specified with the mandatory argument passed to -name. The generator will work on the interconnect specified by -topology and will use the clock passed to -clock as default. Depending on the chosen generator, different additional arguments may be needed, which are passed as a TCL list to -params.

These generators are currently available:

- '**interrupt**' - Inserts interrupt wires and ports.
- '**csr'** - Inserts a configuration network for the initialization of the Control and Status Registers (CSR). This involves creation of ATUs (Arteris Translation Unit).

  The configuration port (APB socket) of every Ncore unit, including the configuration DII (instance name: sys_dii), will connect to a target ATU (Arteris Translation Unit) via the APB interface. The AXI4 socket of the configuration DII (instance name: sys_dii) also connects to the initiator ATU via the AXI4 interface. The initiator ATU initiates the request on the CSR request network and configures the value in the CSR of the Ncore unit. The target ATU receives the request, configures the Ncore unit and then sends the response on the CSR response network.
- '**resiliency**' - Inserts the logic required to implement resiliency in the system.
- '**interface_units**' - Bootstraps a new solution by inserting the Interface Units needed at every Socket.
  - On an Ncore design, interface units refer to the Ncore units. The number of DCEs to be created is given by the option params.
  - On a Symphony design, interface units refer to the ATUs (Arteris Translation Unit).
- **'fix_deadlock'** - Fixes the deadlock in the network.
- **'adapters'** - Automatically inserts required adapters to bridge the clock frequency or width mismatches.
- **'regular topology'** - Automatically inserts the network topology depending on the type and parameters

Usage

```
run_generator -name <name_of_generator> -topology <topology> \
-clock <clock>
```

Options

-name <name> : Name of the generator to invoke, type: string, choices: {interrupt, csr, interface_units, resiliency, fix_deadlock, adapters}

-topology <topology> : Name of the topology to work on, type: string

-clock <clock> : Name of the Clock Subdomain the generator will use, type: string

[-params <params>] : Additional optional generator arguments, type: string, default: ""

Examples

To auto generate CSR network:

```
run_generator -name csr -topology <path/to/topology> \
-clock <path/to/clock>
```

To fix particular networks (e.g., ndn1 and dn):

```
run_generator -name fix_deadlock -topology $topology \
-params [list network ndn1 network dn]
```

To generate network topology using mesh in ndn network:

```
set route_params [list type mesh name "mesh_topo" meshx 4 \
meshy 3 network project/chip0/sys0/subsys0/topology/ndn \
dataWidth 128]
run_generator -name "regular_topology" -topology $topology \
-clock <path to clock> -params $route_params
```

4.2.38 run_task

This command invokes and executes a Maestro task specified with the argument passed to -name. The list of available tasks can be retrieved with get_tasks.

Usage

```
run_task -name <name> -choice <name>
```

Options

[-name <name>] : name of the task to launch

[-choice <choice>] : if the task offers alternatives, the chosen one

Example

```
run_task -name initialize_solution -choice one_ctrl_one_data
```

4.2.39 save_project

This command saves to disk the project currently open. If a file name is not specified, the project will by default be saved based on the name and path given at project creation or opening time. If a file name is specified but a file with that name already exists, it will not be overwritten, unless -force is also given.

Usage

```
save_project - file <file_to_be_saved>
```

Options

[-file <file>] : Name of the project file, type: string, default: ""

[-force] : force the overwriting of a previously-saved file

Example

```
save_project -file /my/path/to/file -force
```

4.2.40 set_attribute

There are two main uses of this command.

1. To select the required template.

Usage

```
set_attribute -object $topology -name coherentTemplate \
-value <name_of_the_configuration>
```

Options

-value : "TwoCtrlOneDataTemplate", "ThreeCtrlOneDataTemplate"

2.To set the value of one or more parameters on the named object.

Usage

```
set_attribute -object <object_name> -name <attribute_name> \
-value <attribute_value>
set_attribute -object <object_name> \
-value_list [list name value name value]
```

Options

-object : Name of the object

-name : Name of the attribute you can set your value

-value : Value of the attribute

-value_list : List of attributes and values to be assigned.

Example

```
set_attribute -object $dmi -name useCache -value true
set_attribute -object $socket -value_list [list socketRole \
SLAVE socketFunction INITIATOR protocolType ACE-Lite]
```

4.2.41 set_context

This command sets the path name of an object. When you set context for an object, it directly points to the parent of the object. There is no need to explicitly mention the parent when the context is set.

Sets the context for future command invocations. Objects are accessible by hierarchical name, and names passed to Tcl commands are interpreted as either the passed name (if starting from the project root) or the concatenation of the context plus the passed name. This command sets the context to be used for subsequent object access.

```
set_context -name <object_name>
```

Options

-name : Name of the existing object

Example

```
set project [create_object -type project -name "dry_run"]
set chip [create_object -type chip -name "chip" -parent $project]
set_context $chip
```

This command sets the path of the parent system and any object created underneath has the parent “$chip”.

4.2.42 set_custom_attribute

This command sets the value of one or more Custom Attributes. If they do not exist yet, they will be created. A Custom Attribute is an entirely user-specified annotation, with any name, to be used for user purposes; Maestro only guarantees the storage. The command can be triggered in two ways: either by specifying a -object, -name and -value, which will set (after creation if necessary) the named Custom Attribute on the named object; or by specifying a -object and a -value_list list of [name, value] pairs.

Usage

```
set_custom_attribute -object <object_name> \
-name <custom_attribute name> -value <value>
```

```
set_custom_attribute -object <object> \
-value_list <name value name value>
```

Options

-object <object> : Name of the object on which the Custom Attribute is to be set, type: string

[-name <name>] : Name of the Custom Attribute to be set (and maybe created), type: string, default: ""

[-value <value>] : Value for the Custom Attribute, type: string, default: ""

[-value_list <value_list>] : List of Custom Attribute name-value pairs, type: string, default: ""

Example

```
set_custom_attribute -object path/to/object \
-value_list [list abc true wAddr 128]
```

4.2.43 set_node_position

Sets the (x,y) location of a switch or Ncoreunit in a topology used in routing the network.

Usage

```
set_node_position -object <object> -x <int> -y <int> -z <int>
```

Options

-object <object> : Name of switch, or Ncoreunit, type: string

[-x <x>] : X coordinate, type: integer, default: `0`

[-y <y>] : Y coordinate, type: integer, default: `0`

[-z <z>] : Z coordinate, used in rare cases., type: integer, default: `0`

Example

```
set_node_position -object project/chip0/cni_caiu1/caiu1 -x 0 -y 2
```

4.2.44 set_project_name

This command changes project name. New name should be compliant with Verilog HDL standard identifiers.

Usage

```
set_project_name -name <name)
```

Options

-name <name> : new name of the project

Example

```
set_project_name -name project_name2
```

4.2.45 set_synthesis_options

Set synthesis options.

This command sets synthesis options.

Usage

```
set_synthesis_options -tech_node <tech_node>
-check_only <check_only> -topo_mode <topo_mode
-rtl_wrapper_dir <rtl_wrapper_dir -hard_macro_dbs <hard_macro_dbs
-bottom_up_synthesis <bottom_up_synthesis
-max_transition <max_transition>
-output_load <output_load>
-sub_design_instances <sub_design_instances>
-incremental_opt <incremental_opt>
-ulvt_percentage <ulvt_percentage
-compile_command <compile_command -object <object>
```

Options

[-tech_node <tech_node>] : Target technology. When custom, tool generate an empty template file which user should fill in with dc_shell commands loading appropriate tech and library files, etc. default custom

[-check_only <check_only>] : Don't run synthesis, only compile and link the RTL default false

[-topo_mode <topo_mode>] : Run synthesis tool in topographical mode default false

[-clock_uncertainty <clock_uncertainty>] : Clock uncertainty as a percentage of the cycle time of the fastest clock of the synthesized block default 15

[-rtl_wrapper_dir <rtl_wrapper_dir>] : Directory with user written verilog files which instantiate custom cells, such as memories. They override generic behavior verilog files generated by Maestro (which typically implement memories as 'sea of registers') and must be named identically. default {}

[-hard_macro_dbs <hard_macro_dbs>] : List of db files containing compiled models of hard macros instantiated from rtl wrappers. default []

[-bottom_up_synthesis <bottom_up_synthesis>] : Characterize and compile major functional blocks (subdesigns) first and then top level. default true

[-max_transition <max_transition>] : Maximal signal transition constraint, ps. default 150

[-output_load <output_load>] : Capacitive load for all primary outputs, ff. default 100

[-sub_design_instances <sub_design_instances>] : List of instance names or whose modules should be synthesized separately as subdesigns. Can include patterns. Only applicable in bottom-up synthesis flow. default []

[-incremental_opt <incremental_opt>] : Enables DC to perform incremental optimization. This works only on mapped logic. default true

[-ulvt_percentage <ulvt_percentage>] : % of ulvt cells can be used by the tool. default 0

[-compile_command <compile_command>] : DC mapping and optimization command. default compile_ultra -spg -no_boundary_optimization -gate_clock

[-object <object>] : object name, should be chip or subsystem

Example

```
set_synthesis_options -topo_mode true
```

4.2.46 split_switch

Split an MxN switch into an Mx1 and an 1xN switch. Break M input and N output switch into one M input and 1 output switch and 1 input to N output switch.

Usage

```
split_switch -name <name>
```

Options

name <name> : name of switch

Example

```
split_switch -name sw01
```

4.2.47 undo

Undoes the previous command.

Usage

```
undo
```

Example

```
undo
```

4.2.48 update_object

Updates the state of the object.

There are 6 purposes of this command.

1. To change the name of the object.

Usage 1

```
update_object -name <existing object> \
-value_list [list new_name <new_name_of_object>]
```

Options 1

name : Path to the existing object

new_name : New name to be given to the object

2. To make connections between objects.

Usage 2

```
update_object -name <object_name> \
-bind <name of the object to bind>
```

Options 2

-bind : Bind to object of types “socket” or “clock_subdomain”

-type : If the object to bind is a socket, the type is "source" or "target". If the object to bind is a clock, the type is "domain".

When using the bind option, you can bind:

- an object of type “atu” to an object of bind type “configsocket”
- an object of type “unit” to an object of bind type “clock_subdomain”
- an object of type "flow" to a "source socket" and a "target socket"

3. To add a list of values to an object. This is used when we want to associate the DMIs to a Dynamic Memory group.

Usage 3

```
update_object -name <object_name> -value_list <list of objects>
```

Options 3

-name : memorygroup object

-value_list : Names of DMIs

4. To execute the automatic topology generator (mesh, ring, torus, butterfly, full-crossbar).

Usage 4

```
update_object -name <name> -value_list \
[list execute <name of the generator> method "execute()"]
```

Options 4

-name : Name given to the topology generator

-execute : Option to execute the generator previously created

-parent : Network under which you want to create the topology

5. To specify multiple socket destination connections of the multi-flow.

Usage 5

update_object -name $mf -type destinations \
-value_list [list $socket1 $socket2 . . .]

Options 5

-name : The name of the flow

-value_list : The list of sockets to be added to the flow.

6. To remove sockets from the destinations. This command is used in the case of multi flow where there can be multiple destinations for a single source.

Usage 6

```
update_object -name $mf -type delete_destinations \
-value_list [list $socket1 $socket2 . . . ]
```

Options 6

-name : The name of the flow

-value_list : The list of sockets to be deleted from the flow

7. To update a manual route.

Usage 7

```
update_object -model route -type <width of the data network> \
-add <route>
```

Options 7

-type : This option is used only when the data width needs to be provided for the data network. By default the value is zero.

-add : This option is used to include the entire route from the source to the destination

Examples

Example 1:

```
set axi [create_object -type socket -name "caiu0" \
-parent $subsystem]
puts $axi
set axi_new "project/chip/Core_AXI_mem/AXI_mem23"
update_object -name $axi -new_name $axi_new
```

In the above example we change the name of the object. Initially we create an object, then we find the entire name of the name of the object by outputting it to the console. We change the name and set it to a new variable and then use update_object to change the name of the object.

Example 2:

```
update_object -name $slave_socket -type "domain" \
-bind $clk_subdomain
```

This binds the object "socket" to a "clock_subdomain"

```
update_object -name $flow -type "source" -bind $src_socket
```

This binds the object "flow" to a "socket".

Example 3:

```
update_object -name $memoryGroup0 -value_list [list $dmi1 \
$dmi2 $dmi3]
```

This shows all the DMIs present in the interleaving group called memoryGroup0.

Example 4:

```
set MESHGEN [create_object -type generator -subtype mesh \
-name "meshgenerator" \
-parent project/chip/system/subsystem/topology]
update_object -name $name \
-value_list [list execute $MESHGEN method "execute()"]
```

We use this command to execute the method to automatically generate the mesh topology under the given network.

5

Maestro Parameters

This chapter describes the Ncore parameters used by Maestro.

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
0	regionBaseAddr	Str	Boot Region Base Address	Specify boot region base address. This address must be aligned to the size specified.			0x0
1	regionHui	Int	Boot Region Home Unit ID	Specify the home unit ID associated with the boot region. If the home unit is a DII then specify the node ID of the DII. If the home unit is a DMI then specify the associated DMI interleaving group ID.			0
2	regionSize	Int	Boot Region Size	Specifies the size of boot region. Minimum is 4KB and must be a power of two.			4096
3	regionHut	Str	Boot Region Home Unit Type	Specify the home unit Type associated with the boot region. This can be either DMI (Memory) or DII (IO device).			DMI
4	Name	Str	Name of the signal	Name of the soc signal that is to be routed and connected to the memory instance.			
5	Direction	Str	Direction of the signal	Direction of the signal.			
6	Width	Int	Width of the signal	Width of the signal.	1	128	
19	power_ref	HierStr	Power Domain	Power Domain associated with this clock domain.			
20	gating	Str	Gating	The gating value determines whether the clock specified in the parent clock region is subject to gating when sent to a portion of the chip, that should be described as this clock domain. Select the option 'always_on' if no gating should be applied, or 'external' if logic external to the inter-connect can gate this clock.			always_o n
21	power_ref	HierStr	Power Region	Power Region associated with this clock region.			
22	frequency	Int	Frequency	Frequency of the clock signal.	400000	220000 0	500000
24	unitClockGating	Bool	Unit-Level Clock Gating	Whether it is desired that units receiving a clock signal from this Clock Subdomain should be sub-ject to unit-level clock gating, when possible, for power savings. Disabling this optimization might help timing.			FALSE
25	frequency	Int	Frequency (kHz)	This parameter sets Clock frequency for the sys-tem.			300000
26	voltage	Int	Voltage (mV)	Voltage of the power supply of the Power region.			1000
54	nDves	Int	DVE Count	Total Number of DVEs in a given Ncore system.	1	1	

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
60	gating	Str	Gating	Whether the voltage supply specified in the parent Power Region should be subject to disconnection when sent to a portion of the chip, that should be described as this Power Domain. If different gating logic can independently disconnect portions of the chip from the Power Region supply, each must be modeled as a separate Power Domain. Ungated supply should also be modeled as its own Power Domain.			always_on
61	voltage	Int	Voltage	Voltage of the power supply of the Power Region.	1	25000	950
73	enableDVM	Bool	Enable DVM	Add AC channel and enable DVM support.			FALSE
73	enableDVM	Bool	Enable DVM	Add AC channel and enable DVM support.			FALSE
80	AxIdProcSelectBits	Int	Processors ID bits	specify processor ID bits.			
105	REQ_RSVDC	Int	Width of request RSVDC	Width of the request RSVDC/user bits.			0
107	wData	Int	CHI data bus width	Width of data on the CHI interface.			128
109	NodeID_Width	Int	Width of Node Id	Width of the Node ID of the CHI Interface.	7	11	7
110	wAddr	Int	CHI address bus width	Width of the address on CHI interface.	44	52	48
113	REQ_RSVDC	Int	Width of request RSVDC	Width of the request RSVDC/user bits.			0
115	wData	Int	CHI data bus width	Width of data on the CHI interface.			128
121	name	Str	Name of Interface	Name of the interface.			
124	direction	Str	Direction of Interface	Direction of the Evt Interface.			master
174	nTaggedMonitors	Int	Number of Tagged exclusive monitors	Specify the desired number of tagged exclusive monitor per DCE instance. Note that each DCE instance will always have a basic exclusive monitor.	0	8	0
185	useAtomic	Bool	Enable Atomic Engine	This option adds an atomic engine in DMI. It must be enabled when an atomic capable master is present in the system and requires at least a 4KB SMC.			FALSE
207	domain_ref	HierStr	Domain	Clock subdomain associated with this unit.			
214	nonCoherentMode	Bool	Non-Coherent Mode	Only applicable if the interface is AXI. Whether the traffic on the interface should be treated as non-coherent. If a Proxy Cache is used, the traffic is always treated as coherent.			FALSE
220	group_ref	HierStr	Group	Group			
227	wArId	Int	Width of Ar Id	Specify the width of AXI interface ArId bits.	1	20	6
228	wAwId	Int	Width of Aw Id	Specify the width of AXI interface AwId bits.	1	20	6
229	wAddr	Int	AXI address bus width	Specify the width of AXI interface address bits.	12	64	32
231	wData	Int	AXI data bus width	Specify the width of AXI interface data bits. Following limitations apply: AXI interface connected to memory as Ncore master (DMI): 128 & 256. AXI interface connected to peripheral device Ncore master (DII): 64, 128 & 256. AXI interface connected to a master agent accelerator,GPU, GIC etc. as Ncore slave (AIU): 64, 128 & 256.	8	2048	64.

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
233	generateHierarchy	Bool	Generate Hierarchy	Whether to generate a layer of logical hierarchy to the RTL.			
241	dceInterleaving-PrimaryBits	Int	System Directory primary set select bits	Address bits to be used as interleaving bits for DCE selection.	6	64	6
245	upstream_ref	HierStr	Upstream Packet Port	Upstream packet port in a Link.			
246	downstream_ref	HierStr	Downstream Packet Port	Downstream packet port in a Link.			
262	primaryInterleav-ingBitOne	Int	Primary Interleaving Bit #1	Primary Address bit 1 to address DMI Interleav-ing. This bit forms the LSB in the vector of pri-mary bits. Only this bit is set for a 2-way interleaving Function.	0	8192	0
264	primaryInterleav-ingBitTwo	Int	Primary Interleaving Bit #2	Primary Address bit 2 to address DMI Interleav-ing. In case of 4-way Interleaving Function, address bits should be set for primaryInterleav-ingBitOne and primaryInterleavingBitTwo.	0	8192	0
266	primaryInterleav-ingBitThree	Int	Primary Interleaving Bit #3	Primary Address bit 3 to address DMI Interleav-ing. In case of 8-way Interleaving Function, address bits should be set for primaryInterleav-ingBitOne, primaryInterleavingBitTwo and pri-maryInterleavingBitThree.	0	8192	0
268	primaryInterleav-ingBitFour	Int	Primary Interleaving Bit #4	Primary Address bit 4 to address DMI Interleav-ing. In case of 16-way Interleaving Function, address bits should be set for primaryInterleav-ingBitOne, primaryInterleavingBitTwo, primary-InterleavingBitThree and primaryInterleavingBitFour.	0	8192	0
279	memoryBase	Int	Base Address	The base address of the region.	0		
280	memorySize	Int	Size	Size of the memory. Each region size is power of 2, starting address is aligned to its size.	0		
339	domain_ref	HierStr	Domain	Clock subdomain associated with this unit.			
346	socket_ref	HierStr	Socket	Socket			
367	qosEnabled	Bool	Enable QoS support	Enable QoS capability.			FALSE
368	qosMap	Str	QoS value to priority map	4 bit Native interface QoS value map to 3 bit pri-ority. value 0 has the highest priority and value 7 has the lowest priority.			
371	qosEventThresh-old	Int	QoS event threshold	QoS starvation threshold. Maximum number of high priority requests that can bypass a lower pri-ority request.	1	8192	16
372	domain_ref	HierStr	Domain	Clock subdomain associated with this unit.			
383	domain_ref	HierStr	Domain	Clock subdomain associated with this socket.			
384	protocolType	Str	Protocol	Protocols supported by ARM.			<not set>
385	socketFunction	Str	Function	The function of the Socket is used to infer which type of Ncore Unit is deployed on the inside of the Subsystem.Allowable values are: - INITIA-TOR for any initiators, only for SLAVEs. - MEMORY for targets connected to memories, only for MASTERs. - PERIPHERAL for targets connected to peripherals, only for MASTERs.			INITIA-TOR

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
389	nGPRA	Int	General purpose address regions	Number of general purpose address regions that the system can support. Refer to the reference manual.	1	24	1
391	resilienceEnabled	Bool	Resilience	Enable resilience-related features in the Ncore system. Refer to the reference manual for details on available features.			FALSE
392	duplicationEn-abled	Bool	Unit duplication	Enable unit duplication for all Ncore units only. Memories and interconnect logic are not dupli-cated; they may be protected separately.			FALSE
393	nativeIntfProtEn-abled	Bool	Native interface protec-tion	Enable capability to add protection on native Ncore interfaces like CHI/ACE/AXI4. This adds an empty Verilog module with specified signals at the interface. Protection logic can be added in this Verilog module.			FALSE
395	coherentTemplate	Str	Template	Control and data network options: OneCtrlOne-DataTemplate: Adds support for a single control and a single data network. TwoCtrlOneDataTem-plate: Adds support for two control and a single data network. ThreeCtrlOneDataTemplate: Adds support for three control and a single data net-work. Refer to the reference manual for more details.			
396	interUnitDelay	Int	Duplicate unit delay	Delay between functional unit and delay unit. Delay can be specified in number of clock cycles.	1	4	1
398	resiliencyProtec-tionType	Str	Interconnect protection type	Interconnect protection type. Both data and con-trol header will be protected. Available options are: NONE: no protection. PARITY: Error detec-tion, parity protection. SECDED: Single bit error correction and double bit error detection, ECC protection.			NONE
399	memoryProtec-tionType	Str	Memory Protection	Protection type for all memories in the Ncore sys-tem. Available options are: NONE : no protec-tion. PARITY : Error detection, parity protection. SECDED : Single bit error correction and double bit error detection, ECC protection.			NONE
406	timeOutThreshold	Int	TimeOut threshold	Time out threshold value. This specifies number of clock cycles within which a transaction must complete in an NCORE system. The value speci-fied is at 4096 clock cycle granularity.	1	2147483647	16384
407	nDvmCmdCredits	Int	Number of DVM com-mand request credits	Maximum number of DVM command requests an AIU can have in flight.	1	4	1
408	nDvmSnpCredits	Int	Number of DVM snoop request credits	Maximum number of DVM snoop requests an AIU can have in flight on its native interface.			2
489	outputDomain_ref	HierStr	Output Clock Domain	The output Clock domain.			
495	domain_ref	HierStr	Clock Domain	Clock domain.			
496	depth	Int	Depth	Defines the depth of the rate adapter.	4	1048576	16
496	depth	Int	Depth	The number of words in the SRAM. Can be 4 or greater.	4	1048576	16
501	latency	Int	Latency	Number of clock signals needed to assert error signals out of the UDV module.	0	8192	1

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
507	wireName	Str	Port Name	Port Name for Generic port.			
508	wireWidth	Int	Port Width	Port width for generic port.	0	1024	
510	wireSignaling	Str	Port Signaling	Port Signaling.			LEVEL
511	wireUpf	Str	Port UPF Value	Port UPF value.			
514	domain_ref	HierStr	Domain	The clock domain associated with the Interrupt Accumulator.			
519	snoopFilter_ref	HierStr	Assigned SnoopFilter	Specify the snoop filter associated with this AIU.			
519	snoopFilter_ref	HierStr	Assigned SnoopFilter	Specify the snoop filter associated with this AIU.			
520	nNativeCredits	Int	Credits for Native Proto-col	Specify the maximum number of CHI link credits this AIU should support	2	15	2
521	nOttCtrlEntries	Int	Outstanding transaction table Entries	Specify the maximum number of outstanding native transactions this AIU should support.	4	128	4
522	nDceCmdCredits	Int	Credits per DCE Com-mand Request	Specify the maximum number of credits for coherent transactions per DCE. This should be determined based on required bandwidth and net-work round trip latency.	2	16	2
523	nDmiCmdCredits	Int	Credits per DMI Com-mand Request	Specify the maximum number of credits for non coherent transactions per DMI. This should be determined based on required bandwidth and net-work round trip latency.	2	16	2
524	nDiiCmdCredits	Int	Credits per DII Command Request	Specify the maximum number of credits for non coherent transactions per DII. This should be determined based on required bandwidth and net-work round trip latency.	2	16	2
525	nStshSnpCredits	Int	Stashing Snoop Credits	Specify the maximum number of outstanding stash snoops this AIU should support. These are stash snoops issued on the CHI interface.	1	8	2
527	nDceRbCredits	Int	Number of DCE write request buffer credits	Specify the maximum number of DCE write request buffer credits per DMI. These credits limit number of Coherent writes and includes snoops that can cause a write to DMI.	2	16	2
528	nAiuSnpCredits	Int	Number of snoop credits	Specify the maximum number of snoop request credits per AIU.	2	16	2
529	nDmiMrdCredits	Int	Number of memory read credits	Specify the maximum number of memory read credits per DMI.	2	16	2
530	nAttCtrlEntries	Int	Number of active coher-ent transactions	Specify the maximum number of active coherent transactions tracked by each DCE.	4	64	4
531	nDiiRbCredits	Int	Number of Write request buffer credits	Specify the maximum number of non coherent write request buffer credits.	2	32	2
532	nRttCtrlEntries	Int	Max outstanding read transactions	Specify number of outstanding read transactions on the AXI interface.	4	32	4
533	nWttCtrlEntries	Int	Max outstanding write transactions	Specify number of outstanding write transactions on the AXI interface.	4	32	4
534	nDmiRbCredits	Int	Number of DMI write buffers	Specify the maximum number of non coherent write request buffer credits.	2	64	2

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
535	nRttCtrlEntries	Int	Max outstanding memory read transactions (RTT Entries)	Specify number of outstanding read transactions on the AXI interface.	4	128	4
536	nWttCtrlEntries	Int	Max outstanding memory write transactions (WTT Entries)	Specify number of outstanding write transactions on the AXI interface.	4	64	4
538	hasProxyCache	Bool	Enable Proxy Cache	This options enables Proxy Cache support. This is supported only with AXI interface.			FALSE
539	nOttCtrlEntries	Int	Outstanding transaction table Entries	Specify the maximum number of outstanding native transactions this AIU should support.	4	128	4
540	nDceCmdCredits	Int	Credits per DCE Command Request	Specify the maximum number of credits for coherent transactions per DCE. This should be determined based on required bandwidth and network round trip latency.	2	16	2
541	nDmiCmdCredits	Int	Credits per DMI Command Request	Specify the maximum number of credits for non coherent transactions per DMI. This should be determined based on required bandwidth and network round trip latency.	2	16	2
542	nDiiCmdCredits	Int	Credits per DII Command Request	Specify the maximum number of credits for non-coherent transactions per DII. This should be determined based on required bandwidth and network round trip latency.	2	16	2
565	depth	Int	Depth	Depth of the Pipe Adapter.	1	2	2
573	outputDomain_ref	HierStr	Output Clock Domain	Clock domain associated at the output port of the rate adapter.			
574	width	Int	Data Width	Data width of the SMI Port.	0	8192	32
575	buffers	Int	Number of Buffers	Number of buffers in the SMI Port.	0	8192	0
576	messagetypes_ref	HierStr	Message Types	Reference to the message types.			
612	targ_id	Str	targ_id	Target Id.			
613	route	Str	Route	Route.			
614	name	Str	Instance name	Instance name.			NONE
619	syncDepth	Int	Synchronizer depth	Synchronizer depth - i.e., pipe stages for synchronizer.	2	10	2
636	wireRtlPrefix	Str	RTL Prefix	RTL Prefix. For a given block, all the ports must have the same writeRtlPrefix.			
637	channel	Str	channel	Channel for PMA.			"Q"
649	targ_id	Str	targ_id	Target Id.			
650	egress	Int	Egress Port	Egress port.			
713	wLateClk	Int	Width of Late Clock	Width of the late clock in the fault interface.	0	1	0
715	SrcID	Int	Width of source Id	Width of the Source ID.	7	11	7
716	TgtID	Int	Width of target Id	Width of the Target ID.	7	11	7
717	enPoison	Bool	Enable poison bits	Enable poison port support on CHI interface.			FALSE
718	wAwUser	Int	Width of Aw user	Specify the width of AXI interface Aw user bits.	0	32	0
720	wArUser	Int	Width of Ar user	Specify the width of AXI interface Ar user bits.	0	32	0

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
721	wAddr	Int	width of address	Width of the address bits.			32
730	SrcID	Int	Width of source Id	Width of the Source ID.	7	7	7
731	TgtID	Int	Width of target Id	Width of the Target ID.	7	7	7
732	wAddr	Int	CHI address bus width	Width of the CHI address bus.	44	44	44
733	wAddr	Int	width of address	Specify the width of AXI interface address bits.	12	64	32
734	wArUser	Int	width of ar user	Specify the width of AXI interface Ar user bits.	0	200	0
735	wArId	Int	Width of ar id	Specify the width of AXI interface ArId bits.	0	32	0
736	wAwUser	Int	Width of aw user	Specify the width of AXI interface Aw user bits.	0	200	0
737	wAwId	Int	Width of aw id	Specify the width of AXI interface AwId bits.	0	32	0
740	wData	Int	Width of data	Specify the width of AXI interface data bits.	8	2048	32
747	wQos	Int	Width of QoS	QoS port gets enabled as standard 4 bit wide port when QoS is enabled.	0	4	4
751	nSets	Int	Number of sets for the selected snoop filter	Number of sets for the selected snoop filter. This value should be a power-of-2.	16	1048576	16
753	nVictimEntries	Int	Number of victim buffer entries per DCE for the specified snoop filter.	The number of victim buffer entries for the tag filter.	0	64	2
754	aPrimaryBits	Int	Set Select Bits	Bits that select the set. They need to be as many as log2(number of sets / number of DCEs), they cannot be in the LSBs inside a cache line, and they cannot overlap with DCE interleaving bits. For example, for a system with 64B cache lines, 1024 sets and 4 DCEs interleaved on bits [11 : 10], this needs to be an 8-bit array with values that could be - e.g. [15, 14, 13, 12, 9, 8, 7, 6].			
765	inputBufferDepth	Int	Input buffer depth	Input buffer depth of switch.			0
768	subsystemType	Str	Subsystem Type	Subsystem is a self-standing portion of the architectural design. The enumerated values define which architecture is selected for the design. 'ARTERIS_COHERENT' design supports cache-coherency. 'ARTERIS_NONCOHERENT' design does not support cache coherency. 'ARTERIS_LLC' supports Last Level Cache.			<Undefined>
771	nMainTraceBufSize	Int	Number of trace entries in the buffer	Specifies the number of trace entries in the buffer.	32	1024	64
772	nTraceRegisters	Int	Number of Trace Registers	Specifies the number of trace register sets present in the system per AIU.	1	4	1
791	productVersion	Str	Product Version	Product (Maestro) version.			0.0.0
792	projectLicenseName	Str	Project License Name	Project License Name.			
793	memoryType	Str	Memory Type	This parameter is used to set the type of memory. This parameter is an enumerated type whose values are either FLOP or SRAM.			FLOP
806	nPerfCounters	Int	Number of performance counters	Number of performance counters per Ncore unit.	4	8	4
808	packetSize	Int	Buffer Size	Size of multicast packet, in beats, to be buffered in block. Does not have to be entire packet.	1	1024	2

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
844	nPerfCounters	Int	Number of performance counters	Number of performance counters per Ncore unit.	4	8	4
850	nPerfCounters	Int	Number of performance counters	Number of performance counters per Ncore unit.	4	8	4
856	incrementalOpt	Bool	Incremental Optimization	Enables DC to perform incremental optimization. This works only on mapped logic.			TRUE
857	ulvtPercentage	Int	Ulvt Percentage	% of ulvt cells can be used by the tool.	0		0
860	compileCommand	Str	Compile command	DC mapping and optimization command. Default: compile_ultra -spg -no_boundary_optimization -gate_clock			(see Desc.)
861	nProcessors	Int	Number of Processors	Specify the number of logical processors connected to this interface.			1
862	nPerfCounters	Int	Number of performance counters	Number of performance counters per Ncore unit.	4	8	4
866	fnCsrAccess	Bool	Enable CSR Access	Enables CSR access via this AIU.			TRUE
868	nPerfCounters	Int	Number of performance counters	Number of performance counters per Ncore unit.	4	8	4
869	fnCsrAccess	Bool	Enable CSR Access	Enables CSR access via this AIU.			TRUE
871	multicastLabels	Int	Carried Multicast Labels	Carried Multicast labels	0	8192	0
872	packetSize	Int	Buffer Size	Size of multicast packet, in beats, to be buffered in block. Does not have to be entire packet.	1	1024	2
878	nPerfCounters	Int	Number of performance counters	Number of performance counters per Ncore unit.	4	8	4
996	headerDepth	Int	Header Depth	Header depth	1	1024	1
996	headerDepth	Int	Header Depth	The number of headers that the FIFO can hold. If headerSplit = false, then the number of headers does not apply.	1	1024	1
1046	rtlPrefixString	Str	Memory RTL name	Specify the name of the embedded memory module to instantiate. The system designer must provide the module to be instantiated with this name.			
1047	MemType	Str	Memory Type	Specify the choice of memory implementation, between a Register (flip-fl0p) implementation, or an embedded memory IP core instantiation. Ncore uses Synopsys embedded memory port names and timing.			NONE
1056	nSets	Int	Number of sets	Specify the number of sets/entries in the Cache.			16
1057	nWays	Int	Number of ways	Specify the number of ways/associativity of the cache.	2	16	2
1058	useScratchpad	Bool	Enable Scratchpad	This option enables scratch pad capability which is available only in DMI.			FALSE
1058	useScratchpad	Bool	Enable Scratchpad	This option enables scratch pad capability which is available only in DMI.			FALSE
1064	PriSubDiagAddrBits	Int	Primary Set Select Bits	Specify address bits to be used as primary set select bits.			
1065	TagBankSelBits	Int	Tag Bank Select Bits	Specify tag bank select bit. This bit must be one of the bits from the primary select bits.			

TABLE 5. Maestro/Ncore Parameters

ID	Name	Type	Title	Description	Min	Max	Default
1066	DataBankSelBits	Int	Data Bank Select Bits	Specify data bank select bit. These bits must be from the primary select bits.			
1092	nLargestEndPoint	Int	Size of the largest end point	Specify the size of the largest endpoint device connected to the DII. The size is in KBs. This size is used to achieve endpoint ordering as defined by ARM CHI specification.	4	549755 813888	4
1095	nAddrTransRegis-ters	Int	Number of Address Translation Registers	Specifies the number of address translation regis-ters that are available within the DII. Refer to the address translation capability section in the refer-ence manual.	0	4	0
1103	nAddrTransRegis-ters	Int	Number of Address Translation Registers	Specifies the number of address translation regis-ters that are available within DMI. These regis-ters add capability to translate address on the AXI bus from DMI. Refer to the address translation section of the reference manual.	0	4	0
1105	nWayPartitionin-gRegisters	Int	nWayPartitioningRegis-ters	Specifies the number of cache way partitioning registers. Each register enables configuration capability to assign specific ways to a single agent. The number of registers enabled here should be equal to maximum number of agents that will be configured for way partitioning.	0	16	0
1297	channel_ref	HierStr	Target Socket	Region maps to one target (or an aggregate of multiple identical targets for address striping).			
1298	mg_ref	HierStr	Target DynamicMemo-ryGroup	Defines the memoryGroup this bootregion can be accessed from.			
1496	headerSplit	Bool	Split Header Buffer	Enable split header buffer if true.			FALSE
1496	headerSplit	Bool	Split Header Buffer	Split header buffer.			FALSE
1534	hasSysMemCache	Bool	SMC Enable	This option adds a System Memory Cache in DMI. It must be enabled when an atomic capable master is present in the system and requires at least a 4KB SMC.			FALSE
1535	nTagBanks	Int	Tag Banks	Number of tag banks in the System Memory Cache.			1
1536	nDataBanks	Int	Data Banks	Number of data banks in the System Memory Cache.			1
1540	cacheReplPolicy	Str	Cache Replacement Pol-icy	Cache Replacement Policy.			RAN-DOM
1540	cacheReplPolicy	Str	Cache Replacement Pol-icy	Cache Replacement Policy.			RAN-DOM
1735	nTagBanks	Int	Tag Banks	Number of tag banks in the Proxy Cache.			1
1736	nDataBanks	Int	Data Banks	Number of data banks in the Proxy Cache.			1
2043	checkOnly	Bool	Compile and Link Only	Whether to stop the RTL flow after compilation and linking, or to proceed to synthesis.			
2044	topoMode	Bool	Topographical Mode	Whether to launch the synthesis tools in topo-graphical mode, or WLM. The latter is faster but less accurate.			
2045	techNode	Str	Technology	The technology node for the design.			CUSTOM

TABLE 5. **Maestro/Ncore Parameters**

ID	Name	Type	Title	Description	Min	Max	Default
2046	clockUncertainty	Int	Clock Uncertainty	The default clock uncertainty to assume for clocks, as a percentage (e.g. 15 = 15%). The value can be overwritten in the generated synthesis scripts.	1	99	15
2047	rtlWrapperDir	Str	RTL Wrapper Dir	Directory with user-written Verilog files which instantiate custom cells, such as memories. They override generic-behavior Verilog files generated by Maestro (which implement memories as a ‘sea of registers’) and must be named identically.			
2048	hardMacroDbs	Str	Hard Macros	List of DB files containing compiled models of hard macros instantiated from RTL wrappers.			
2049	bottomUpSynthesis	Bool	Bottom-Up	Characterize and compile major functional blocks (subdesigns) first, and then the toplevel.			TRUE
2050	maxTransition	Int	Max Transition	Max signal transition time constraint, in ps.	0		150
2051	outputLoad	Int	Output Load	Capacitive load for all primary outputs, in fF.	0		100000
2543	region_ref	HierStr	Physical Region	The physical region associated with this group.			
7551	portDataWidth	Int	Port Data Width	The data width of the packet port in a switch.	0	8192	0
7554	portPacketSerialization	Str	Port Packet Serialization	This parameter sets the packet style on the packet port.			HEADER_PARALLEL
10710	csrAccessSupported	Bool	CSR Access	Whether this Socket will be allowed access to the CSR network, if one is inserted. Only applicable to INITIATOR Sockets.			TRUE
25443	preferredCSRdomain_ref	HierStr	Preferred CSR clock	Preferred CSR clock domain.			

6

Demonstration Design

This chapter guides you through the phases of an annotated Maestro demonstration design using Tcl commands and scripting. The unannotated design source is listed in the appendix (see "Tcl Demonstration Design Source").

6.1 Demonstration Design Overview

This demonstration design uses the following configurations:

- Template : 2 Control, 1 Data Network topology (TwoCtrlOneData)
- CAIUs : 2 CHI-B
- NCAIUs : 2 AXI4, 1 ACE-LITE-E
- DCEs : 2
- DMIs : 3
- DIIs : 1
- Topology: Butterfly

6.2 Demonstration Design Workflow

Note: This Tcl design mirrors the Maestro GUI Demonstration Design in the Ncore 3 User Guide. The unannotated Tcl design source is listed in the appendix (see "Tcl Demonstration Design Source").

- "Phase 1 — Chip Specification"
  - "Create Project"
  - "Create Chip"
  - "Create Power"
  - "Create Clock"
- "Phase 2 — System Assembly"
  - "Create System"
  - "Create Subsystem"
  - "Create Sockets"
  - "Create Memory Map"
  - "Create 2-way Interleaving Functions"
  - "Create Boot Region"
  - "Create CSR Region"
- "Phase 3 — Structural Design (Architecture)"
  - "Create Topology"
  - "Select Coherent Template"
  - "Auto Generate Units"
  - "Set Node Positions of the Units in the Topology"
  - "Create Snoopfilters"
  - "Generate Topology for Routing"
  - "Auto Generate the Control and Status Register (CSR) Network"
  - "Auto Generate Adapters"
  - "Set preMap Parameters"
- "Phase 4 — Mapping"
  - "Start Mapping Process"
- "Phase 5 — Refinement"
  - "Set postMap Parameters"
- "Phase 6 — Export"
  - "Export Setup"
  - "Generate RTL"

6.3 Phase 1 — Chip Specification

```
###########################################################################################
# Description
#
# This configuration uses the following
#  Template -> ThreeSysThreeMemTemplate
#  CAIUs    -> 2 CHI-B
#  NCAIUs   -> 2 AXI4, 1 ACE-LITE-E
#  CCEs     -> 2
#  DMIs     -> 3
#  DIIs     -> 1
#
#####################################################################################
# Main
#####################################################################################
# PHASE 1
# Create project
# Create chip
# Create Power
# Create Clock
# *************************************************************************************
```

6.3.1 Create Project

```
#Create project
set designName native_flow
set fileName $designName.mpf
if {[file exists $fileName] == 1} {
    file delete $fileName
}
set project [create_project -name $designName -license_token Arteris/Dev]
```

6.3.2 Create Chip

```
#create chip
set chip    [create_object -type chip    -name "chip"   -parent $project]

# Set Synthesis options
set_synthesis_options -object $chip -tech_node tsmc7
set_synthesis_options -object $chip -topo_mode true
set_synthesis_options -object $chip -check_only true
set_synthesis_options -object $chip -clock_uncertainty 30
set_synthesis_options -object $chip -bottom_up_synthesis true
set_synthesis_options -object $chip -max_transition 200
set_synthesis_options -object $chip -output_load 150
set_synthesis_options -object $chip -sub_design_instances [list ".*"]
set_synthesis_options -object $chip -incremental_opt false
set_synthesis_options -object $chip -ulvt_percentage 10
set_synthesis_options -object $chip -compile_command "test command"

set frequency 1600; #MHz
set voltage 1000;#mV
```

6.3.3 Create Power

```
#Create power
# Create a power region
set pregion [create_object -name "PR0" -type power_region    -parent $chip]

# Set a voltage on the power region
set_attribute -object $pregion -name voltage -value $voltage

# Create the power domain
set pdomain [create_object -name "PD0" -type power_domain    -parent $pregion]

#Set the gating on the power domain
set_attribute -object $pdomain -name gating -value always_on
```

6.3.4 Create Clock

```
#Create Clock
# Create a clock region
set clkregion [create_object -name "clk_R0" -type clock_region    -parent $chip]

# Bind the power region to the clock region
update_object -name $clkregion -bind $pregion -type "powerRegion"

# Set the frequency on the clock region
set_attribute -object $clkregion -name frequency -value $frequency

# Create a clock domain
set clkdomain [create_object -name "clk_D0" -type clock_domain    -parent $clkregion]

#Bind the clock domain to the power domain
```

```
update_object -name $clkdomain -bind $pdomain -type "powerDomain"


#Create a clock subdomain
set clocksubdomain [create_object -name "clk_SD0" -type clock_subdomain -parent $clkdomain]
```

6.4 Phase 2 — System Assembly

```
#####################################################################################
# PHASE 2
# Create system
# Create subsystem
# Create sockets
# Create memory map
#************************************************************************************
```

6.4.1 Create System

```
# Create a system
set system  [create_object -type system  -name "system" -parent $chip]
```

6.4.2 Create Subsystem

```
#Create a subsystem
set subsystem  [create_object -type subsystem  -name "subsystem" -parent $system]
set_attribute -object $subsystem -name subsystemType -value ARTERIS_COHERENT
```

6.4.3 Create Sockets

```
#Create Sockets
#Create the CHI-B socket and Bind the clock subdomain to every socket.
set sock [create_object -type socket -name "caiu0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType CHI-B \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/NodeID_Width 7 \
                                      params/enPoison false \
                                      params/REQ_RSVDC 0]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the CHI-B socket
set sock [create_object -type socket -name "caiu1" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType CHI-B \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/NodeID_Width 7 \
                                      params/enPoison false \
                                      params/REQ_RSVDC 0]

update_object -name $sock -bind $clocksubdomain -type domain

#Create the AXI4-INITIATOR socket
set sock [create_object -type socket -name "ncaiu0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 10 \
                                      params/wAwId 10]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the AXI4-INITIATOR socket
set sock [create_object -type socket -name "ncaiu1" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 10 \
                                      params/wAwId 10]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the ACE-Lite-E-INITIATOR socket
set sock [create_object -type socket -name "ncaiu2" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType ACE-Lite-E \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 10 \
                                      params/wAwId 10]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the DMI socket
set sock [create_object -type socket -name "dmi0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "MEMORY" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain
#Create the DMI socket
set sock [create_object -type socket -name "dmi1" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "MEMORY" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the DMI socket
set sock [create_object -type socket -name "dmi2" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "MEMORY" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the DII socket
set sock [create_object -type socket -name "dii0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "PERIPHERAL" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain
```

6.4.4 Create Memory Map

```
#Create Memory Map

#create the object MemoryMap
set memorymap [create_object -type memory_map -name "mm" -parent $subsystem]

# Create the memory groups under memorySet0
set memorySet0 [create_object -type memory_set -parent $memorymap -name "ms0"]
set memoryGroup00 [create_object -type dynamic_memory_group -parent $memorySet0 -name "mg0"]
set memoryGroup01 [create_object -type dynamic_memory_group -parent $memorySet0 -name "mg1"]

#Assign DMI sockets to the memory group00
update_object -name $memoryGroup00 -value_list [list $subsystem/dmi0 $subsystem/dmi1]
update_object -name $memoryGroup01 -value_list [list $subsystem/dmi2]

# Create the memory groups under memorySet1
set memorySet1 [create_object -type memory_set -parent $memorymap -name "ms1"]
set memoryGroup10 [create_object -type dynamic_memory_group -parent $memorySet1 -name "mg0"]
set memoryGroup11 [create_object -type dynamic_memory_group -parent $memorySet1 -name "mg1"]
set memoryGroup12 [create_object -type dynamic_memory_group -parent $memorySet1 -name "mg2"]

#Assign DMI sockets to the memory group01
update_object -name $memoryGroup10 -value_list [list $subsystem/dmi0]
update_object -name $memoryGroup11 -value_list [list $subsystem/dmi1]
update_object -name $memoryGroup12 -value_list [list $subsystem/dmi2]
```

6.4.5 Create 2-way Interleaving Functions

```
# Create 2-way interleaving functions
set twoWayIntFunc1 [create_object -type memory_interleave_function -name "twowayif1" \
-parent $memorymap]
set_attribute -object $twoWayIntFunc1 -name primaryInterleavingBitOne -value 18
set twoWayIntFunc2 [create_object -type memory_interleave_function -name "twowayif2" \
-parent $memorymap]
set_attribute -object $twoWayIntFunc2 -name primaryInterleavingBitOne -value 19
```

6.4.6 Create Boot Region

```
#create BootRegion

set boot [create_object -type boot_region -parent $memorymap -name "bootregion"]


#Set the base address on the bootregion

set_attribute -object $boot -name memoryBase -value [expr 0x0]


#Set the memory size on the bootregion

set_attribute -object $boot -name memorySize -value [expr 0x4000]


#access boot code from DMI

update_object -name $boot -bind $memoryGroup00 -type "memoryGroup"
```

6.4.7 Create CSR Region

```
#create CSR region

set csr_region [create_object -type configuration_region -parent $memorymap -name "csrregion"]


# Set the base address for CSR region

set_attribute -object $csr_region -name memoryBase -value [expr [expr 0x2e6e4000]*1024]; \
# in bytes

#EBH value in user guide 0x2e6e4000
```

6.5 Phase 3 — Structural Design (Architecture)

```
#####################################################################################
# PHASE 3: ARCHITECTURAL DESIGN
# Create topology
# Select coherent Template
# Auto generate units
# Set the node positions of the units in the topology
# Generate topology for routing
# Auto generate adapters
# Auto generate the Control and Status Register (CSR) network
# *************************************************************************************
```

6.5.1 Create Topology

```
#Create the topology
set topology [create_object -type topology -name "topology" -parent $subsystem]
```

6.5.2 Select Coherent Template

```
#Select Template
#Set the template as "TwoCtrlOneDataTemplate". This contains one data network and one \
control network
set_attribute -object $topology -name coherentTemplate -value "TwoCtrlOneDataTemplate"
```

6.5.3 Auto Generate Units

```
puts "############################################"
puts "####     Interface Unit Automation.     ####"
puts "############################################"
#Auto generate units
#Run automatic generation of Ncore units. The clock is given for the DCE and the DVE.
#The params option defines the number of DCEs
run_generator -name "interface_units" -topology $topology -clock $clocksubdomain \
-params [list directories 2]


puts "############################################"
puts "####  Interface Unit Automation Done.   ####"
puts "############################################"
```

6.5.4 Set Node Positions of the Units in the Topology

```
set caius [get_objects -parent $chip -type caiu]
set caiu0 [lindex $caius 0]
set caiu1 [lindex $caius 1]

set diis [get_objects -parent $chip -type dii]
set dii0 [lindex $diis 0]
set sys_dii   [get_objects -parent $chip -type dii -subtype configDii]

set dmis [get_objects -parent $chip -type dmi]
set dmi0 [lindex $dmis 0]
set dmi1 [lindex $dmis 1]
set dmi2 [lindex $dmis 2]

set dces [get_objects -parent $chip -type dce]
set dce0 [lindex $dces 0]
set dce1 [lindex $dces 1]

set ncaius [get_objects -parent $chip -type ncaiu]
set ncaiu0 [lindex $ncaius 0]
set ncaiu1 [lindex $ncaius 1]
set ncaiu2 [lindex $ncaius 2]

set dves [get_objects -parent $chip -type dve]
set dve0 [lindex $dves 0]
```

6.5.5 Create Snoopfilters

```
#Create 1 snoopfilters
set sf0 [create_object -type snoop_filter -parent $topology -name "sf0"]
set sf1 [create_object -type snoop_filter -parent $topology -name "sf1"]

#Set parameters on the snoopfilters
set_attribute -object $sf0 -name nSets           -value 64
set_attribute -object $sf0 -name nWays           -value 4
set_attribute -object $sf0 -name aPrimaryBits    -value_list [list 7 8 9 10 11]
set_attribute -object $sf0 -name nVictimEntries  -value 0

set_attribute -object $sf1 -name nSets           -value 64
set_attribute -object $sf1 -name nWays           -value 4
set_attribute -object $sf1 -name aPrimaryBits    -value_list [list 7 8 9 10 11]
set_attribute -object $sf1 -name nVictimEntries  -value 0

#Assign CAIUS to Snoopfilter 0 (sf0)
update_object -name $caiu0 -type snoopFilter -bind $sf0
update_object -name $caiu1 -type snoopFilter -bind $sf0

# Set ProxyCache on NCAIUs 0, 1
set_attribute -object $ncaiu0 -name hasProxyCache -value true
set_attribute -object $ncaiu1 -name hasProxyCache -value true
set_attribute -object $ncaiu2 -name hasProxyCache -value false

#Assign NCAIUS to Snoopfilter 1 (sf1)
update_object -name $ncaiu0 -type snoopFilter -bind $sf1
update_object -name $ncaiu1 -type snoopFilter -bind $sf1

set networks [get_objects -parent $chip -type network]
set datawidth 128



foreach net $networks {
  # Skip mesh generation if it is CSR network
  set request  [query_object -object $net -type "csr_request"]
  set response [query_object -object $net -type "csr_response"]
  if {$request == "true" || $response == "true"} {
    continue
  }

  set_context -name $net
  set pos [string last "/" $net]
  set nam [string range $net $pos+1 end]

  #Set the priority count on Data and Non data networks
  #EBH caused an error set_attribute -object $net -name priorityCount -value 8
```

6.5.6 Generate Topology for Routing

```
  # Generate the topology only on Data network and Non-data networks
  # Create the mesh generator. Assign the datawidth which is needed for the data network.
  set dataWidth 0
  if {$nam == "dn"} {
    set dataWidth $datawidth
  }

  #Butterfly Topology
    set route_params [list type butterfly name $nam meshx 0 meshy 0 network $net \
dataWidth $dataWidth optimizeDataWidth false maxVC 0]
```

```
    run_generator -name regular_topology -topology $topology -clock $clocksubdomain \
-params $route_params


}
puts "############################################"
puts "####       Butterfly Generation Done.        ####"
puts "############################################"
```

6.5.7 Auto Generate the Control and Status Register (CSR) Network

```
puts "############################################"
puts "####      CSR Insertion Automation.     ####"
puts "############################################"

#Run automatic generator for the CSR network
run_generator -topology $topology -name "csr" -clock $clocksubdomain

#Run automatic generator for interrupt accumulator
run_generator -topology $topology -name "interrupt" -clock $clocksubdomain

puts "############################################"
puts "####   CSR Insertion Automation Done.   ####"
puts "############################################"
```

6.5.8 Auto Generate Adapters

```
puts "############################################"
puts "### Adapter insertion.   ####"
puts "############################################"

#Run automatic generator for insertion of adapters
run_generator -topology $topology -name "adapters"

puts "############################################"
puts "### Adapter insertion Done.   ####"
puts "############################################"
```

6.5.9 Set preMap Parameters

```
puts "############################################"
puts "####   Pre map parameters               ####"
puts "############################################"

# atus
# Extract APB-ATUs in apbAtus
# Extract AXI-ATUs in axiAtus
# set objects [get_objects -type atu -parent $chip]

# The ATUs are created in the CSR(Control and Status Registers) Network.

set objects [get_objects -type atu -parent $chip]
set axiAtus [list]
set apbAtus [list]
foreach atu $objects {
   set sk [get_objects -parent $atu -type socket]
   set func [query_attribute -object $sk -name socketFunction]
   if {[string compare $func "CONFIGURATION"] == 0} {
       lappend apbAtus $atu
   } elseif {[string compare $func "CONFIG_INTERFACE"] == 0} {
       lappend axiAtus $atu
   }
}

#-------------------------------------------------------------------
#System parameters
#-------------------------------------------------------------------
set_attribute -object $topology -name nDvmCmdCredits -value 4
set_attribute -object $topology -name nDvmSnpCredits -value 4
set_attribute -object $topology -name nGPRA -value 6


# DCE interleaving selection bits since we have 2 DCEs
set_attribute -object $topology -name dceInterleavingPrimaryBits -value_list [list 40]

#-----------------------------------------------------------------------
# Unit level parameters
#-----------------------------------------------------------------------

#---------------------------------------------------------------
# PREMAP for DCE0
#-------------------------------------------------------------
set_attribute -object $dce0 -name nAttCtrlEntries    -value 48
set_attribute -object $dce0 -name nDceRbCredits      -value 2
set_attribute -object $dce0 -name nAiuSnpCredits     -value 2
set_attribute -object $dce0 -name nDmiMrdCredits     -value 2

#-----------------------------------------------------------------
# PREMAP for DCE1
#---------------------------------------------------------------
set_attribute -object $dce1 -name nAttCtrlEntries    -value 48
set_attribute -object $dce1 -name nDceRbCredits      -value 2
set_attribute -object $dce1 -name nAiuSnpCredits     -value 2
set_attribute -object $dce1 -name nDmiMrdCredits     -value 2



#-----------------------------------------------------------------
# PREMAP for CAIUS
#-----------------------------------------------------------------
#Set the premap  attributes on the caiu0
set_attribute -object $caiu0 -name nNativeCredits        -value 15
set_attribute -object $caiu0 -name nOttCtrlEntries       -value 48
set_attribute -object $caiu0 -name nDceCmdCredits        -value 8
set_attribute -object $caiu0 -name nDmiCmdCredits        -value 4
set_attribute -object $caiu0 -name nDiiCmdCredits        -value 4
set_attribute -object $caiu0 -name nStshSnpCredits       -value 2
set_attribute -object $caiu0 -name nProcessors           -value 1

#Set the premap  attributes on the caiu1
set_attribute -object $caiu1 -name nNativeCredits    -value 15
set_attribute -object $caiu1 -name nOttCtrlEntries   -value 48
set_attribute -object $caiu1 -name nDceCmdCredits    -value 8
set_attribute -object $caiu1 -name nDmiCmdCredits    -value 4
set_attribute -object $caiu1 -name nDiiCmdCredits    -value 4
set_attribute -object $caiu1 -name nStshSnpCredits   -value 2
set_attribute -object $caiu1 -name nProcessors       -value 1

#-----------------------------------------------------------------
# PREMAP for NCAIUS
#-----------------------------------------------------------------
#Set the premap  attributes on the ncaiu0
set_attribute -object $ncaiu0 -name nOttCtrlEntries  -value 64
set_attribute -object $ncaiu0 -name nDceCmdCredits   -value 8
set_attribute -object $ncaiu0 -name nDmiCmdCredits   -value 4

set_attribute -object $ncaiu0 -name nTagBanks -value 1
set_attribute -object $ncaiu0 -name nDataBanks -value 1
set_attribute -object $ncaiu0 -name cacheReplPolicy -value "RANDOM"
```

```
#------------------------------------------------------------------------
# 2 Tag_Memory of NCAIU0
#------------------------------------------------------------------------
set Tagmemories [get_objects -parent $ncaiu0 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#------------------------------------------------------------------------
# 2 Data_Memory of NCAIU0
#------------------------------------------------------------------------
set Datamemories [get_objects -parent $ncaiu0 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"

#Set the premap  attributes on the ncaiu1
set_attribute -object $ncaiu1 -name nOttCtrlEntries  -value 48
set_attribute -object $ncaiu1 -name nDceCmdCredits   -value 8
set_attribute -object $ncaiu1 -name nDmiCmdCredits   -value 4

set_attribute -object $ncaiu1 -name nTagBanks -value 1
set_attribute -object $ncaiu1 -name nDataBanks -value 1
set_attribute -object $ncaiu1 -name cacheReplPolicy -value "RANDOM"


#Set the premap  attributes on the ncaiu2 (No Proxy Cache)
set_attribute -object $ncaiu2 -name nOttCtrlEntries  -value 32
set_attribute -object $ncaiu2 -name nDceCmdCredits   -value 4
set_attribute -object $ncaiu2 -name nDmiCmdCredits   -value 4
set_attribute -object $ncaiu2 -name nDiiCmdCredits   -value 4



#-----------------------------------------------------------------
# PREMAP for DMI
#-----------------------------------------------------------------
#Set the premap attributes on the dmi0
set_attribute -object $dmi0 -name nRttCtrlEntries    -value 48
set_attribute -object $dmi0 -name nWttCtrlEntries    -value 32
set_attribute -object $dmi0 -name nDmiRbCredits      -value 16

set_attribute -object $dmi0 -name hasSysMemCache -value true
set_attribute -object $dmi0 -name nTagBanks -value 1
set_attribute -object $dmi0 -name nDataBanks -value 1
set_attribute -object $dmi0 -name cacheReplPolicy -value "NRU"

#-----------------------------------------------------------------------
# 2 Tag_Memory of DMI0
#-----------------------------------------------------------------------
set Tagmemories [get_objects -parent $dmi0 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#-----------------------------------------------------------------------
# 4 Data_Memory of DMI0
#-----------------------------------------------------------------------
set Datamemories [get_objects -parent $dmi0 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"

#-----------------------------------------------------------------------
# 1 Write Data_Memory of DMI0
#-----------------------------------------------------------------------
set WrDatamemories [get_objects -parent $dmi0 -type internal_memory -subtype WDATA]
set wrdm0 [lindex $WrDatamemories 0]
set_attribute -object $wrdm0 -name memoryType -value "FLOP"

#Set the premap attributes on the dmi1
set_attribute -object $dmi1 -name nRttCtrlEntries    -value 48
set_attribute -object $dmi1 -name nWttCtrlEntries    -value 32
set_attribute -object $dmi1 -name nDmiRbCredits      -value 16

set_attribute -object $dmi1 -name hasSysMemCache -value true
set_attribute -object $dmi1 -name nTagBanks -value 1
set_attribute -object $dmi1 -name nDataBanks -value 1
set_attribute -object $dmi1 -name cacheReplPolicy -value "RANDOM"

#-----------------------------------------------------------------------
# 2 Tag_Memory of DMI1
#-----------------------------------------------------------------------
set Tagmemories [get_objects -parent $dmi1 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#-----------------------------------------------------------------------
```

```
# 4 Data_Memory of DMI1
#-------------------------------------------------------------------------
set Datamemories [get_objects -parent $dmi1 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"

#-------------------------------------------------------------------------
# 1 Write Data_Memory of DMI1
#-------------------------------------------------------------------------
set WrDatamemories [get_objects -parent $dmi1 -type internal_memory -subtype WDATA]
set wrdm0 [lindex $WrDatamemories 0]
set_attribute -object $wrdm0 -name memoryType -value "FLOP"

#Set the premap attributes on the dmi2
set_attribute -object $dmi2 -name nRttCtrlEntries    -value 32
set_attribute -object $dmi2 -name nWttCtrlEntries    -value 16
set_attribute -object $dmi2 -name nDmiRbCredits      -value 16
set_attribute -object $dmi2 -name hasSysMemCache     -value true
set_attribute -object $dmi2 -name nTagBanks -value 1
set_attribute -object $dmi2 -name nDataBanks -value 1
set_attribute -object $dmi2 -name cacheReplPolicy -value "RANDOM"

#-------------------------------------------------------------------------
# 2 Tag_Memory of DMI2
#-------------------------------------------------------------------------
set Tagmemories [get_objects -parent $dmi2 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#-------------------------------------------------------------------------
# 4 Data_Memory of DMI2
#-------------------------------------------------------------------------
set Datamemories [get_objects -parent $dmi2 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"

#-------------------------------------------------------------------------
# 1 Write Data_Memory of DMI2
set WrDatamemories [get_objects -parent $dmi2 -type internal_memory -subtype WDATA]
set wrdm0 [lindex $WrDatamemories 0]
set_attribute -object $wrdm0 -name memoryType -value "FLOP"
```

```
#diis
set diis [get_objects -type dii -parent $chip]
set dii0 [lindex $diis 0]
set sysdii [lindex $diis 1]

#Set the premap attributes on dii0
set_attribute -object $dii0 -name nDiiRbCredits      -value 16

puts "##################################################"
puts "####   Setting Pre map parameters  complete   ####"
puts "##################################################"

puts "############################################"
puts "####   User Design Creation complete.   ####"
puts "############################################"

#Save the project
set fname ${designName}.premap.mpf
save_project -file $fname
```

6.6 Phase 4 — Mapping

```
########################################################################
################## Phase 4: Mapping
########################################################################
```

6.6.1 Start Mapping Process

```
puts "############################################"
puts "####    Start mapping process.          ####"
puts "############################################"
set stat [map_design -name $topology]

if {$stat == 0} {
  puts "ERROR in map_design. Exiting ..."
  exit 1
  return
}

puts "############################################"
puts "####    Mapping process complete.       ####"
puts "############################################"
```

6.7 Phase 5 — Refinement

```
########################################################################
################## Phase 5: Refinement
########################################################################
```

6.7.1 Set postMap Parameters

```
puts "#############################################"
puts "####   Setting post-map parameters.     ####"
puts "#############################################"



#DCE settings
#DCE0
set dcePrefix mappedObjects/dce0_mapped
set_attribute -object $dce0 -name $dcePrefix/ImplementationParameters/nTaggedMonitors \
-value 4


#DCE1
set dcePrefix mappedObjects/dce1_mapped
set_attribute -object $dce1 -name $dcePrefix/ImplementationParameters/nTaggedMonitors \
-value 4


#ncaiu settings


#Setting parameters related to proxyCache
set prCachePrefix mappedObjects/ncaiu0_mapped/Cache
set_attribute -object $ncaiu0 -name $prCachePrefix/nSets                              -value 16
set_attribute -object $ncaiu0 -name $prCachePrefix/nWays                              -value 2
set_attribute -object $ncaiu0 -name $prCachePrefix/SelectInfo/PriSubDiagAddrBits \
-value_list [list 6 7 8 9]


set prCachePrefix mappedObjects/ncaiu1_mapped/Cache
set_attribute -object $ncaiu1 -name $prCachePrefix/nSets                              -value 16
set_attribute -object $ncaiu1 -name $prCachePrefix/nWays                              -value 2
set_attribute -object $ncaiu1 -name $prCachePrefix/SelectInfo/PriSubDiagAddrBits \
-value_list [list 6 7 8 9]



# DMI settings
#DMI0
set dmiPrefix mappedObjects/dmi0_mapped
set_attribute -object $dmi0 -name $dmiPrefix/useAtomic                                          -
value false
```

```
# Settings for DMI configured with System Memory Cache
set smcPrefix mappedObjects/dmi0_mapped/Cache
set_attribute -object $dmi0 -name $smcPrefix/nSets                              -value 16
set_attribute -object $dmi0 -name $smcPrefix/nWays                              -value 2
set_attribute -object $dmi0 -name $smcPrefix/useScratchpad                      -value false
set_attribute -object $dmi0 -name $smcPrefix/SelectInfo/PriSubDiagAddrBits \
-value_list [list 6 7 8 9]
set_attribute -object $dmi0 -name $smcPrefix/SelectInfo/TagBankSelBits          -value_list []
set_attribute -object $dmi0 -name $smcPrefix/SelectInfo/DataBankSelBits         -value_list []

#DMI1
set dmiPrefix mappedObjects/dmi1_mapped
set_attribute -object $dmi1 -name $dmiPrefix/useAtomic                                              -
value false

# Settings for DMI configured with System Memory Cache
set smcPrefix mappedObjects/dmi1_mapped/Cache
set_attribute -object $dmi1 -name $smcPrefix/nSets                              -value 16
set_attribute -object $dmi1 -name $smcPrefix/nWays                              -value 2
set_attribute -object $dmi1 -name $smcPrefix/useScratchpad                      -value false
set_attribute -object $dmi1 -name $smcPrefix/SelectInfo/PriSubDiagAddrBits \
-value_list [list 6 7 8 9]
set_attribute -object $dmi1 -name $smcPrefix/SelectInfo/TagBankSelBits          -value_list []
set_attribute -object $dmi1 -name $smcPrefix/SelectInfo/DataBankSelBits         -value_list []

#DMI2
set dmiPrefix mappedObjects/dmi2_mapped
set_attribute -object $dmi2 -name $dmiPrefix/useAtomic                                    -value
false
set smcPrefix mappedObjects/dmi2_mapped/Cache
set_attribute -object $dmi2 -name $smcPrefix/nSets                              -value 16
set_attribute -object $dmi2 -name $smcPrefix/nWays                              -value 2
set_attribute -object $dmi2 -name $smcPrefix/useScratchpad                      -value false
set_attribute -object $dmi2 -name $smcPrefix/SelectInfo/PriSubDiagAddrBits \
-value_list [list 6 7 8 9]
set_attribute -object $dmi2 -name $smcPrefix/SelectInfo/TagBankSelBits          -value_list []
set_attribute -object $dmi2 -name $smcPrefix/SelectInfo/DataBankSelBits         -value_list []


#DII settings
set diiPrefix mappedObjects/dii0_mapped
set_attribute -object $dii0 -name $diiPrefix/nLargestEndPoint -value 262144
```

```
puts "#####################################################"
puts "####   Setting post-map parameters complete.     ####"
puts "#####################################################"
```

6.8 Phase 6 — Export

```
#############################################################
# PHASE 6: EXPORT
##############################################################
```

6.8.1 Export Setup

```
file mkdir ./json
set intermediateJson ./json/top.level.json
set postmapMpf       ${designName}.mpf

export_design -format flat -file $intermediateJson
save_project -file $postmapMpf
puts "Design mpf created in $postmapMpf"
```

6.8.2 Generate RTL

```
#GENERATE RTL
#----------------------------------------------------------------------
#gen_collateral

exit
```

6.9 Graphical View of the Networks

Graphical views of the networks are shown in the following figures.

Figure 7. Data network (Mesh)

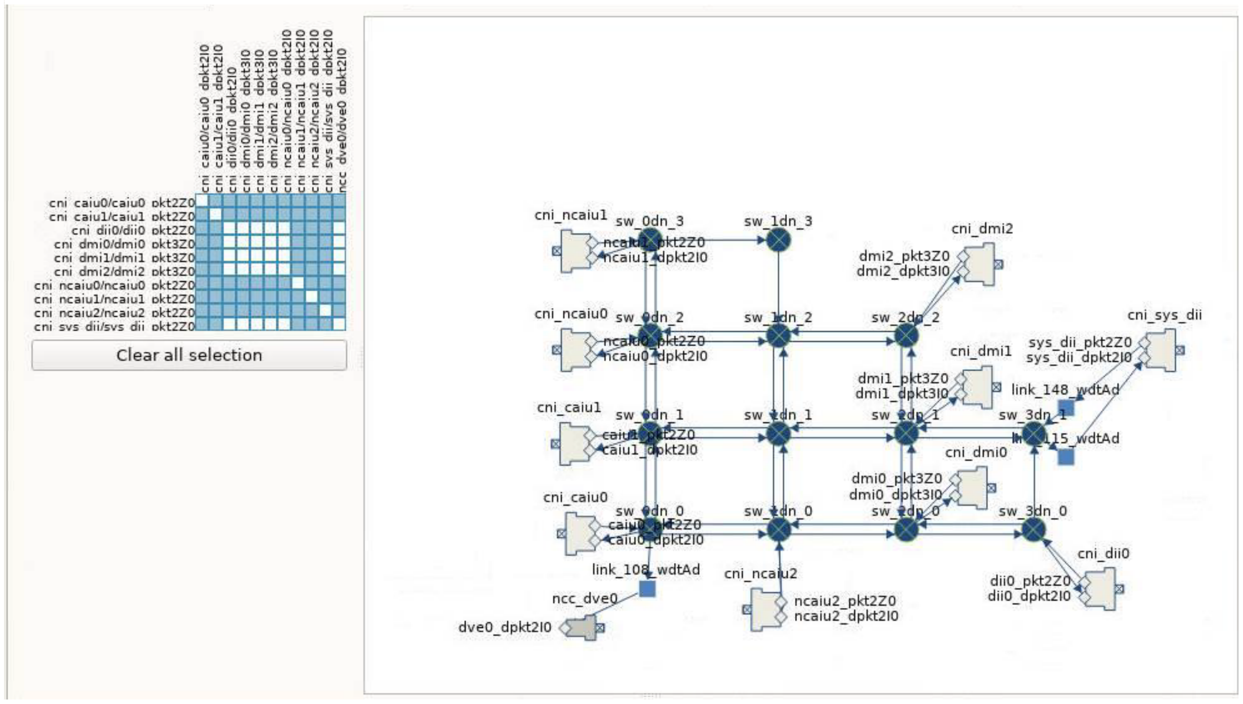


Figure 8. Non Data network (Mesh)

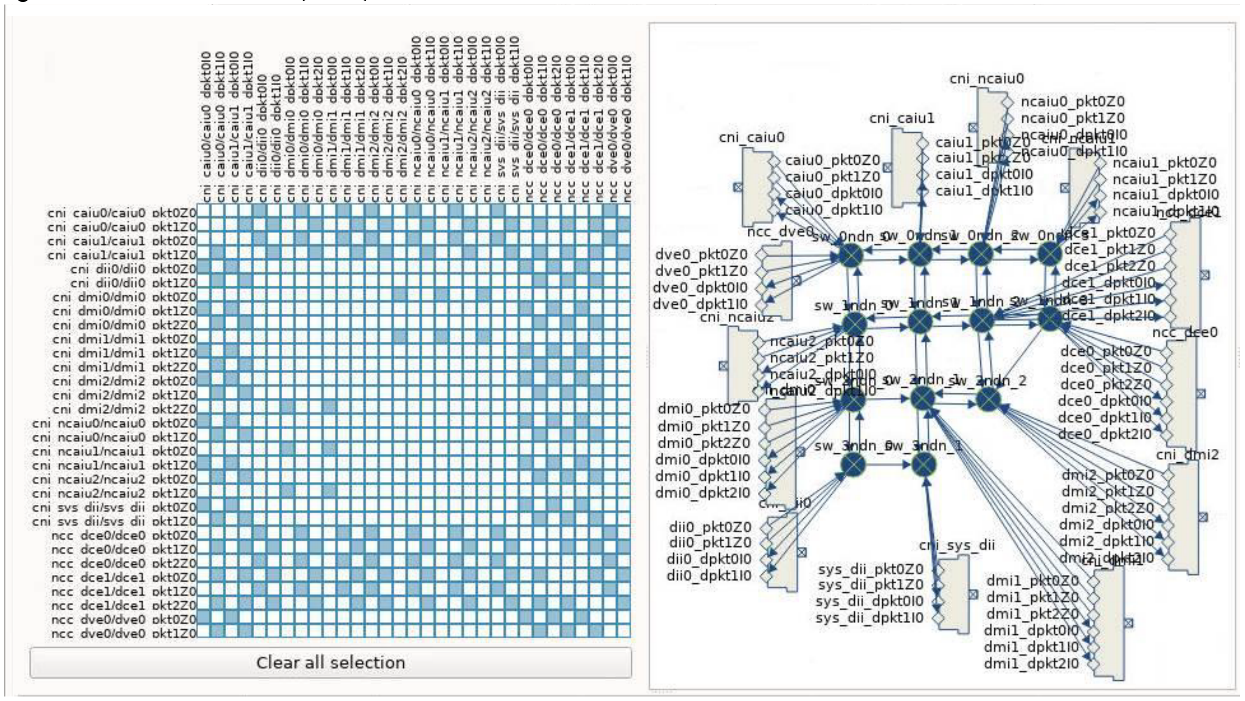


Figure 9. CSR request network (tree structure 1)

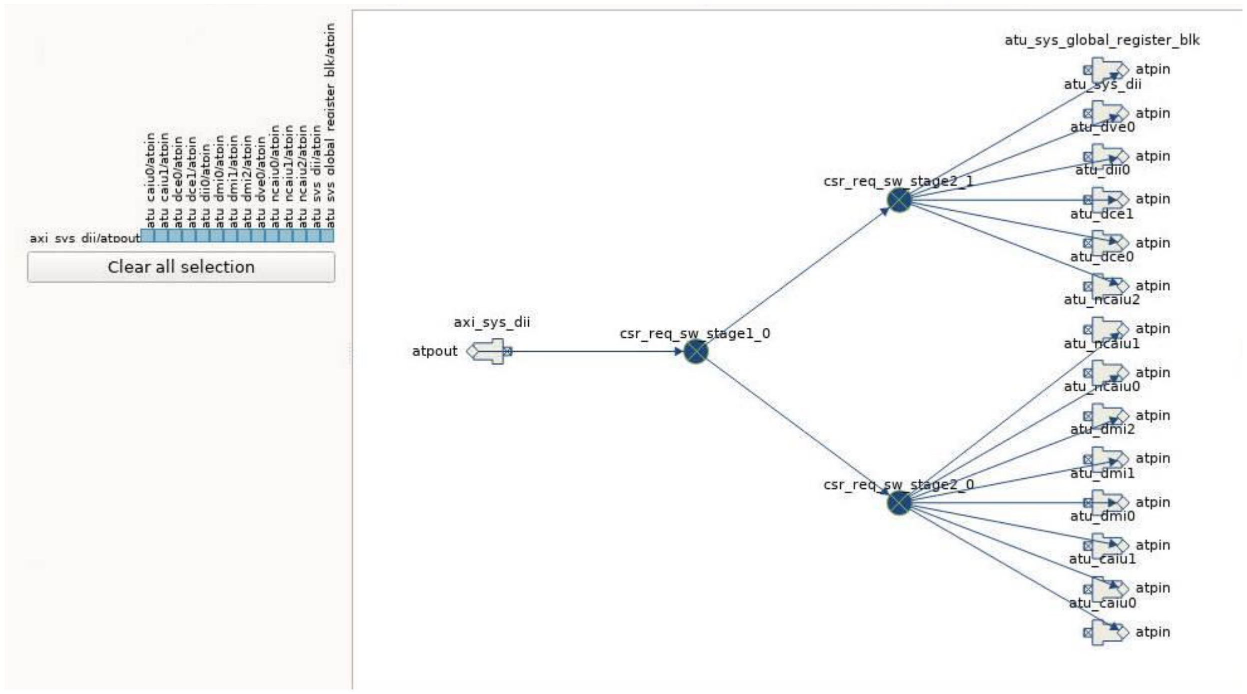
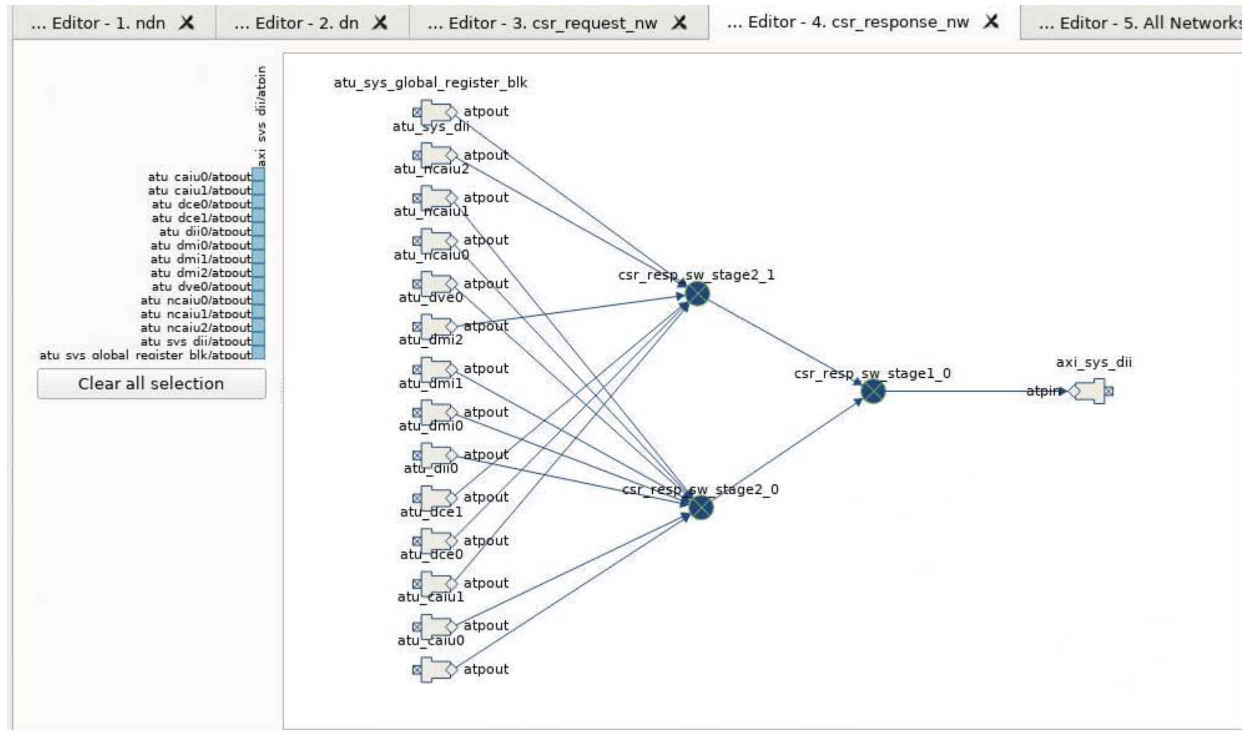


Figure 10. CSR response network (tree structure 2)



A

Appendix

1.1 Glossary of Terms

Table 6. Terms and Abbreviations

Terms	Descriptions
ATU	Arteris Transport Unit used in Non-Coherent Transport Interconnect.
C-AIU	Coherent Agent Interface Unit. Can be CHI-A, CHI-B or ACE. CHI AIU provides native interface for connecting CHI-A/CHI-B processors (processors compatible with ARM interface standards like Coherent Hub Interface Issue-A and Issue-B) with the Ncore3 Coherency system. Used interchangeably with CHI-AIU.
Chip	Object to store the design that the user is trying to build.
Clock Adapter	Unit to synchronize clock frequencies
Clock Domain	One of the main clock signals
Clock Region	Collection to hold a group of clock signals
Clock Subdomain	Clock derived from one of the main clocks
CNI	Coherent Network Interface, used internally in Maestro
ConfigSocket	Socket used for the Configuration Networks
Database	The database contains all the information entered by the user, as well as information regarding the state of the tool, for a project.
DCE	Model for Distributed Coherency Engine. It is the central block in the coherence system and is responsible for maintaining memory coherence across the set memory locations mapped to it.
Depacketizer	Converts the packets back to native layer transactions. Packetizer is present in the Coherent Network Interface for every Ncore unit (see Packetizer).
DII	Model for Distributed IO Interface. The DII unit is a configurable IP block which acts as a gateway to the I/O devices.
DMI	Model for Distributed Memory Interface Unit. It is a configurable IP block which interfaces with the next level memory hierarchy (e.g., system memory).

Table 6. Terms and Abbreviations

Terms	Descriptions
DVE	Distributed Virtual Memory Management Engine. It is the central block in the Ncore system for Distributed Virtual Memory transactions and is the point of serialization for all the DVM requests sent to it.
Dynamic Memory Group	Container for DMI sockets. It can hold maximum of 4 DMI sockets.
Fabric	A Fabric is a set of interconnected packets switching and transport hardware units.
Generator	Engine in Maestro used to carry out a functionality.
GRB	Global Register Block. It manages memory address.
GUI	Graphical User Interface. It is the software module which provides visual representation of items and enables the user to interact with them.
IO-AIU	Model for Non Coherent Agent Interface Unit. The IOAIU serves the master agent using its native AXI/ACE interface and exchanges protocol messages with other blocks in the system using the internal messaging protocol.
IP	Intellectual Property refers to the hardware components delivered as RTL to customers who use them to build ASIC. ArterisIP products are considered as IP.
IP Generator (IPG)	The IPG concept is used by the Maestro user to access high-level functions. An IPG instance contains information usually filled by the user and/or through automation.
IP Generator Unit	The IP Generator unit is a basic element reflecting the structure of the IP created by the IP Generator.
Memory Map	Group of memory sets in the design. Only one active memory map is possible in the design.
Memory Set	Set of Memory Groups. The memory groups in the set must include all the DMIs.
MPF	Maestro Project files
NC-AIU	Non-Coherent Agent Interface Unit
NCNI	Non-coherent Network Interface
Network	The Network is a combination of switches connected through links, and other internal elements such as buffers and pipelines. Its function is to transport requests and responses between sources and destinations at the edge.
NoC	Network on Chip can be described as a class of communication fabrics. Arteris IP interconnects are based on NoC designs.
Object	An object describes a user-visible container of information elements, organized in a structured manner; as well as a set of methods to interact with the information.
Packet port	It is a port in the design, used to send packets
Packetizer	Converts the native layer transactions into packets to be transported into the network. It is present in the Coherent Network Interface for every Ncore unit (see Depacketizer).
Port	A Port is a simple input or output of a hardware unit. They can have multiple bits.
Power Domain	Power source for the Ncore unit
Power Region	Collection of power domains
Project	A project is a container of all information required to create a Communication Subsystem. All elements of a projects are related to each other. Maestro lets the user work on one project at a time.

Table 6. Terms and Abbreviations

Terms	Descriptions
Route	Route is an ordered list of Transport units where traffic can go through from a source NIU to a destination NIU. Routes uniquely define the set of physical and virtual connections in a network.
Routing	Contains information describing the connectivity between Sockets and the address decode and transformation function used to route requests between them.
Session	It is the time duration when the Maestro tool is running continuously on a computer. During a session, information entered in the tool is assumed to be kept in memory.
SMI Port	Symphony Message Interface Port
SoC	System on a Chip can be described as a class of Integrated Circuits built by assembling multiple components (CPU, memory, accelerators, etc) around a Communication Subsystem.
Socket	A Socket is a bundle of inputs and/or outputs of a hardware unit, that follows a communication protocol. Example of Sockets protocols are AMBA/AXI, OCP-IP, TileLink.
Subsystem	A Subsystem is any combination of ArterisIP products including single instance of a particular IP and last level cache. The user creates and optimizes the subsystem using Maestro. Subsequent to RTL generation, the Subsystem is used in the SoC.
Switch	Entity in a network which receives packets and forwards it to the destination unit.

1.2 Tcl Demonstration Design Source

Copy and paste the design into a Tcl file. Make sure to remove any document headers/footers before running the design in Maestro.

```
#############################################################################################
# Description
#
# This configuration uses the following
#  Template -> ThreeSysThreeMemTemplate
#  CAIUs    -> 2 CHI-B
#  NCAIUs   -> 2 AXI4, 1 ACE-LITE-E
#  CCEs     -> 2
#  DMIs     -> 3
#  DIIs     -> 1
#
#######################################################################################
# Main
#######################################################################################
# PHASE 1
# Create project
# Create chip
# Create Power
# Create Clock
# ***************************************************************************************
#Create project
set designName native_flow
set fileName $designName.mpf
if {[file exists $fileName] == 1} {
    file delete $fileName
}
set project [create_project -name $designName -license_token Arteris/Dev]

#create chip
set chip    [create_object -type chip    -name "chip"   -parent $project]

# Set Synthesis options
set_synthesis_options -object $chip -tech_node tsmc7
set_synthesis_options -object $chip -topo_mode true
set_synthesis_options -object $chip -check_only true
set_synthesis_options -object $chip -clock_uncertainty 30
set_synthesis_options -object $chip -bottom_up_synthesis true
set_synthesis_options -object $chip -max_transition 200
set_synthesis_options -object $chip -output_load 150
set_synthesis_options -object $chip -sub_design_instances [list ".*"]
set_synthesis_options -object $chip -incremental_opt false
set_synthesis_options -object $chip -ulvt_percentage 10
set_synthesis_options -object $chip -compile_command "test command"

set frequency 1600; #MHz
set voltage 1000;#mV

#Create power
# Create a power region
set pregion [create_object -name "PR0" -type power_region    -parent $chip]

# Set a voltage on the power region
set_attribute -object $pregion -name voltage -value $voltage

# Create the power domain
set pdomain [create_object -name "PD0" -type power_domain    -parent $pregion]

#Set the gating on the power domain
set_attribute -object $pdomain -name gating -value always_on

#Create Clock
# Create a clock region
set clkregion [create_object -name "clk_R0" -type clock_region    -parent $chip]

# Bind the power region to the clock region
update_object -name $clkregion -bind $pregion -type "powerRegion"

# Set the frequency on the clock region
set_attribute -object $clkregion -name frequency -value $frequency

# Create a clock domain
set clkdomain [create_object -name "clk_D0" -type clock_domain    -parent $clkregion]

#Bind the clock domain to the power domain
update_object -name $clkdomain -bind $pdomain -type "powerDomain"

#Create a clock subdomain
```

```
set clocksubdomain [create_object -name "clk_SD0" -type clock_subdomain -parent $clkdomain]

#######################################################################################
# PHASE 2
# Create system
# Create subsystem
# Create sockets
# Create memory map
#**************************************************************************************
# Create a system
set system  [create_object -type system  -name "system" -parent $chip]

#Create a subsystem
set subsystem  [create_object -type subsystem  -name "subsystem" -parent $system]
set_attribute -object $subsystem -name subsystemType -value ARTERIS_COHERENT

#Create Sockets
#Create the CHI-B socket and Bind the clock subdomain to every socket.
set sock [create_object -type socket -name "caiu0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType CHI-B \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/NodeID_Width 7 \
                                      params/enPoison false \
                                      params/REQ_RSVDC 0]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the CHI-B socket
set sock [create_object -type socket -name "caiu1" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType CHI-B \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/NodeID_Width 7 \
                                      params/enPoison false \
                                      params/REQ_RSVDC 0]

update_object -name $sock -bind $clocksubdomain -type domain

#Create the AXI4-INITIATOR socket
set sock [create_object -type socket -name "ncaiu0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 10 \
                                      params/wAwId 10]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the AXI4-INITIATOR socket
set sock [create_object -type socket -name "ncaiu1" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 10 \
                                      params/wAwId 10]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the ACE-Lite-E-INITIATOR socket
set sock [create_object -type socket -name "ncaiu2" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType ACE-Lite-E \
                                      socketFunction INITIATOR \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 10 \
                                      params/wAwId 10]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the DMI socket
set sock [create_object -type socket -name "dmi0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "MEMORY" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain
#Create the DMI socket
set sock [create_object -type socket -name "dmi1" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "MEMORY" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the DMI socket
set sock [create_object -type socket -name "dmi2" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "MEMORY" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain

#Create the DII socket
set sock [create_object -type socket -name "dii0" -parent $subsystem]
set_attribute -object $sock -value_list [list protocolType AXI4 \
                                      protocolType "AXI4" \
                                      socketFunction "PERIPHERAL" \
                                      params/wAddr 48 \
                                      params/wData 128 \
                                      params/wAwUser 0 \
                                      params/wArUser 0 \
                                      params/wArId 14 \
                                      params/wAwId 14]
update_object -name $sock -bind $clocksubdomain -type domain

#Create Memory Map

#create the object MemoryMap
set memorymap [create_object -type memory_map -name "mm" -parent $subsystem]

# Create the memory groups under memorySet0
set memorySet0 [create_object -type memory_set -parent $memorymap -name "ms0"]
set memoryGroup00 [create_object -type dynamic_memory_group -parent $memorySet0 -name "mg0"]
set memoryGroup01 [create_object -type dynamic_memory_group -parent $memorySet0 -name "mg1"]

#Assign DMI sockets to the memory group00
update_object -name $memoryGroup00 -value_list [list $subsystem/dmi0 $subsystem/dmi1]
update_object -name $memoryGroup01 -value_list [list $subsystem/dmi2]

# Create the memory groups under memorySet1
set memorySet1 [create_object -type memory_set -parent $memorymap -name "ms1"]
set memoryGroup10 [create_object -type dynamic_memory_group -parent $memorySet1 -name "mg0"]
set memoryGroup11 [create_object -type dynamic_memory_group -parent $memorySet1 -name "mg1"]
set memoryGroup12 [create_object -type dynamic_memory_group -parent $memorySet1 -name "mg2"]

#Assign DMI sockets to the memory group01
update_object -name $memoryGroup10 -value_list [list $subsystem/dmi0]
update_object -name $memoryGroup11 -value_list [list $subsystem/dmi1]
update_object -name $memoryGroup12 -value_list [list $subsystem/dmi2]

# Create 2-way interleaving functions
set twoWayIntFunc1 [create_object -type memory_interleave_function -name "twowayif1" -parent $memorymap]
set_attribute -object $twoWayIntFunc1 -name primaryInterleavingBitOne -value 18
set twoWayIntFunc2 [create_object -type memory_interleave_function -name "twowayif2" -parent $memorymap]
set_attribute -object $twoWayIntFunc2 -name primaryInterleavingBitOne -value 19

#create BootRegion
set boot [create_object -type boot_region -parent $memorymap -name "bootregion"]

#Set the base address on the bootregion
set_attribute -object $boot -name memoryBase -value [expr 0x0]

#Set the memory size on the bootregion
set_attribute -object $boot -name memorySize -value [expr 0x4000]

#access boot code from DMI
update_object -name $boot -bind $memoryGroup00 -type "memoryGroup"
```

```
#create CSR region
set csr_region [create_object -type configuration_region -parent $memorymap -name "csrregion"]

# Set the base address for CSR region
set_attribute -object $csr_region -name memoryBase -value [expr [expr 0x2e6e4000]*1024]; # in bytes
#EBH value in user guide 0x2e6e4000

######################################################################################
# PHASE 3: ARCHITECTURAL DESIGN
# Create topology
# Select coherent Template
# Auto generate units
# Set the node positions of the units in the topology
# Generate topology for routing
# Auto generate adapters
# Auto generate the Control and Status Register (CSR) network
# ***********************************************************************************
#Create the topology
set topology [create_object -type topology -name "topology" -parent $subsystem]

#Select Template
#Set the template as "TwoCtrlOneDataTemplate". This contains one data network and one control network
set_attribute -object $topology -name coherentTemplate -value "TwoCtrlOneDataTemplate"

puts "###########################################"
puts "####     Interface Unit Automation.     ####"
puts "###########################################"
#Auto generate units
#Run automatic generation of Ncore units. The clock is given for the DCE and the DVE.
#The params option defines the number of DCEs
run_generator -name "interface_units" -topology $topology -clock $clocksubdomain -params [list directories 2]

puts "###########################################"
puts "####  Interface Unit Automation Done.   ####"
puts "###########################################"

set caius [get_objects -parent $chip -type caiu]
set caiu0 [lindex $caius 0]
set caiu1 [lindex $caius 1]

set diis [get_objects -parent $chip -type dii]
set dii0 [lindex $diis 0]
set sys_dii   [get_objects -parent $chip -type dii -subtype configDii]

set dmis [get_objects -parent $chip -type dmi]
set dmi0 [lindex $dmis 0]
set dmi1 [lindex $dmis 1]
set dmi2 [lindex $dmis 2]

set dces [get_objects -parent $chip -type dce]
set dce0 [lindex $dces 0]
set dce1 [lindex $dces 1]

set ncaius [get_objects -parent $chip -type ncaiu]
set ncaiu0 [lindex $ncaius 0]
set ncaiu1 [lindex $ncaius 1]
set ncaiu2 [lindex $ncaius 2]

set dves [get_objects -parent $chip -type dve]
set dve0 [lindex $dves 0]


#Create 1 snoopfilters
set sf0 [create_object -type snoop_filter -parent $topology -name "sf0"]
set sf1 [create_object -type snoop_filter -parent $topology -name "sf1"]

#Set parameters on the snoopfilters
set_attribute -object $sf0 -name nSets            -value 64
set_attribute -object $sf0 -name nWays            -value 4
set_attribute -object $sf0 -name aPrimaryBits     -value_list [list 7 8 9 10 11]
set_attribute -object $sf0 -name nVictimEntries   -value 0

set_attribute -object $sf1 -name nSets            -value 64
set_attribute -object $sf1 -name nWays            -value 4
set_attribute -object $sf1 -name aPrimaryBits     -value_list [list 7 8 9 10 11]
set_attribute -object $sf1 -name nVictimEntries   -value 0

#Assign CAIUS to Snoopfilter 0 (sf0)
update_object -name $caiu0 -type snoopFilter -bind $sf0
update_object -name $caiu1 -type snoopFilter -bind $sf0

# Set ProxyCache on NCAIUs 0, 1
set_attribute -object $ncaiu0 -name hasProxyCache -value true
set_attribute -object $ncaiu1 -name hasProxyCache -value true
set_attribute -object $ncaiu2 -name hasProxyCache -value false
```

```
#Assign NCAIUS to Snoopfilter 1 (sf1)
update_object -name $ncaiu0 -type snoopFilter -bind $sf1
update_object -name $ncaiu1 -type snoopFilter -bind $sf1

set networks [get_objects -parent $chip -type network]
set datawidth 128


foreach net $networks {
  # Skip mesh generation if it is CSR network
  set request  [query_object -object $net -type "csr_request"]
  set response [query_object -object $net -type "csr_response"]
  if {$request == "true" || $response == "true"} {
    continue
  }

  set_context -name $net
  set pos [string last "/" $net]
  set nam [string range $net $pos+1 end]

  #Set the priority count on Data and Non data networks
  #EBH caused an error set_attribute -object $net -name priorityCount -value 8

  # Generate the topology only on Data network and Non-data networks
  # Create the mesh generator. Assign the datawidth which is needed for the data network.
  set dataWidth 0
  if {$nam == "dn"} {
    set dataWidth $datawidth
  }

  #Butterfly Topology
    set route_params [list type butterfly name $nam meshx 0 meshy 0 network $net dataWidth $dataWidth \
optimizeDataWidth false maxVC 0]
    run_generator -name regular_topology -topology $topology -clock $clocksubdomain -params $route_params

}
puts "##############################################"
puts "####       Butterfly Generation Done.         ####"
puts "##############################################"


puts "##############################################"
puts "####      CSR Insertion Automation.     ####"
puts "##############################################"

#Run automatic generator for the CSR network
run_generator -topology $topology -name "csr" -clock $clocksubdomain

#Run automatic generator for interrupt accumulator
run_generator -topology $topology -name "interrupt" -clock $clocksubdomain

puts "##############################################"
puts "####   CSR Insertion Automation Done.   ####"
puts "##############################################"

puts "##############################################"
puts "### Adapter insertion.   ####"
puts "##############################################"

#Run automatic generator for insertion of adapters
run_generator -topology $topology -name "adapters"

puts "##############################################"
puts "### Adapter insertion Done.   ####"
puts "##############################################"

puts "##############################################"
puts "####   Pre map parameters               ####"
puts "##############################################"

# atus
# Extract APB-ATUs in apbAtus
# Extract AXI-ATUs in axiAtus
# set objects [get_objects -type atu -parent $chip]

# The ATUs are created in the CSR(Control and Status Registers) Network.

set objects [get_objects -type atu -parent $chip]
set axiAtus [list]
set apbAtus [list]
foreach atu $objects {
   set sk [get_objects -parent $atu -type socket]
   set func [query_attribute -object $sk -name socketFunction]
   if {[string compare $func "CONFIGURATION"] == 0} {
        lappend apbAtus $atu
    } elseif {[string compare $func "CONFIG_INTERFACE"] == 0} {
        lappend axiAtus $atu
    }
}

#--------------------------------------------------------------------
#System parameters
#--------------------------------------------------------------------
set_attribute -object $topology -name nDvmCmdCredits -value 4
set_attribute -object $topology -name nDvmSnpCredits -value 4
set_attribute -object $topology -name nGPRA -value 6


# DCE interleaving selection bits since we have 2 DCEs
set_attribute -object $topology -name dceInterleavingPrimaryBits -value_list [list 40]

#------------------------------------------------------------------------
# Unit level parameters
#------------------------------------------------------------------------

#------------------------------------------------------------------
# PREMAP for DCE0
#----------------------------------------------------------------
set_attribute -object $dce0 -name nAttCtrlEntries    -value 48
set_attribute -object $dce0 -name nDceRbCredits      -value 2
set_attribute -object $dce0 -name nAiuSnpCredits     -value 2
set_attribute -object $dce0 -name nDmiMrdCredits     -value 2

#------------------------------------------------------------------
# PREMAP for DCE1
#----------------------------------------------------------------
set_attribute -object $dce1 -name nAttCtrlEntries    -value 48
set_attribute -object $dce1 -name nDceRbCredits      -value 2
set_attribute -object $dce1 -name nAiuSnpCredits     -value 2
set_attribute -object $dce1 -name nDmiMrdCredits     -value 2


#------------------------------------------------------------------
# PREMAP for CAIUS
#------------------------------------------------------------------
#Set the premap  attributes on the caiu0
set_attribute -object $caiu0 -name nNativeCredits        -value 15
set_attribute -object $caiu0 -name nOttCtrlEntries       -value 48
set_attribute -object $caiu0 -name nDceCmdCredits        -value 8
set_attribute -object $caiu0 -name nDmiCmdCredits        -value 4
set_attribute -object $caiu0 -name nDiiCmdCredits        -value 4
set_attribute -object $caiu0 -name nStshSnpCredits       -value 2
set_attribute -object $caiu0 -name nProcessors           -value 1

#Set the premap  attributes on the caiu1
set_attribute -object $caiu1 -name nNativeCredits    -value 15
set_attribute -object $caiu1 -name nOttCtrlEntries   -value 48
set_attribute -object $caiu1 -name nDceCmdCredits    -value 8
set_attribute -object $caiu1 -name nDmiCmdCredits    -value 4
set_attribute -object $caiu1 -name nDiiCmdCredits    -value 4
set_attribute -object $caiu1 -name nStshSnpCredits   -value 2
set_attribute -object $caiu1 -name nProcessors       -value 1

#------------------------------------------------------------------
# PREMAP for NCAIUS
#------------------------------------------------------------------
#Set the premap  attributes on the ncaiu0
set_attribute -object $ncaiu0 -name nOttCtrlEntries  -value 64
set_attribute -object $ncaiu0 -name nDceCmdCredits   -value 8
set_attribute -object $ncaiu0 -name nDmiCmdCredits   -value 4

set_attribute -object $ncaiu0 -name nTagBanks -value 1
set_attribute -object $ncaiu0 -name nDataBanks -value 1
set_attribute -object $ncaiu0 -name cacheReplPolicy -value "RANDOM"

#-------------------------------------------------------------------------
# 2 Tag_Memory of NCAIU0
#-------------------------------------------------------------------------
set Tagmemories [get_objects -parent $ncaiu0 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#-------------------------------------------------------------------------
# 2 Data_Memory of NCAIU0
#-------------------------------------------------------------------------
set Datamemories [get_objects -parent $ncaiu0 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"
```

```
#Set the premap  attributes on the ncaiu1
set_attribute -object $ncaiu1 -name nOttCtrlEntries  -value 48
set_attribute -object $ncaiu1 -name nDceCmdCredits   -value 8
set_attribute -object $ncaiu1 -name nDmiCmdCredits   -value 4

set_attribute -object $ncaiu1 -name nTagBanks -value 1
set_attribute -object $ncaiu1 -name nDataBanks -value 1
set_attribute -object $ncaiu1 -name cacheReplPolicy -value "RANDOM"


#Set the premap  attributes on the ncaiu2 (No Proxy Cache)
set_attribute -object $ncaiu2 -name nOttCtrlEntries  -value 32
set_attribute -object $ncaiu2 -name nDceCmdCredits   -value 4
set_attribute -object $ncaiu2 -name nDmiCmdCredits   -value 4
set_attribute -object $ncaiu2 -name nDiiCmdCredits   -value 4


#-----------------------------------------------------------------
# PREMAP for DMI
#-----------------------------------------------------------------
#Set the premap attributes on the dmi0
set_attribute -object $dmi0 -name nRttCtrlEntries    -value 48
set_attribute -object $dmi0 -name nWttCtrlEntries    -value 32
set_attribute -object $dmi0 -name nDmiRbCredits      -value 16

set_attribute -object $dmi0 -name hasSysMemCache -value true
set_attribute -object $dmi0 -name nTagBanks -value 1
set_attribute -object $dmi0 -name nDataBanks -value 1
set_attribute -object $dmi0 -name cacheReplPolicy -value "NRU"

#------------------------------------------------------------------------
# 2 Tag_Memory of DMI0
#------------------------------------------------------------------------
set Tagmemories [get_objects -parent $dmi0 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#------------------------------------------------------------------------
# 4 Data_Memory of DMI0
#------------------------------------------------------------------------
set Datamemories [get_objects -parent $dmi0 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"

#------------------------------------------------------------------------
# 1 Write Data_Memory of DMI0
#------------------------------------------------------------------------
set WrDatamemories [get_objects -parent $dmi0 -type internal_memory -subtype WDATA]
set wrdm0 [lindex $WrDatamemories 0]
set_attribute -object $wrdm0 -name memoryType -value "FLOP"

#Set the premap attributes on the dmi1
set_attribute -object $dmi1 -name nRttCtrlEntries    -value 48
set_attribute -object $dmi1 -name nWttCtrlEntries    -value 32
set_attribute -object $dmi1 -name nDmiRbCredits      -value 16

set_attribute -object $dmi1 -name hasSysMemCache -value true
set_attribute -object $dmi1 -name nTagBanks -value 1
set_attribute -object $dmi1 -name nDataBanks -value 1
set_attribute -object $dmi1 -name cacheReplPolicy -value "RANDOM"

#------------------------------------------------------------------------
# 2 Tag_Memory of DMI1
#------------------------------------------------------------------------
set Tagmemories [get_objects -parent $dmi1 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#------------------------------------------------------------------------
# 4 Data_Memory of DMI1
#------------------------------------------------------------------------
set Datamemories [get_objects -parent $dmi1 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"

#------------------------------------------------------------------------
# 1 Write Data_Memory of DMI1
#------------------------------------------------------------------------
set WrDatamemories [get_objects -parent $dmi1 -type internal_memory -subtype WDATA]
set wrdm0 [lindex $WrDatamemories 0]
set_attribute -object $wrdm0 -name memoryType -value "FLOP"
```

```
#Set the premap attributes on the dmi2
set_attribute -object $dmi2 -name nRttCtrlEntries    -value 32
set_attribute -object $dmi2 -name nWttCtrlEntries    -value 16
set_attribute -object $dmi2 -name nDmiRbCredits      -value 16
set_attribute -object $dmi2 -name hasSysMemCache     -value true
set_attribute -object $dmi2 -name nTagBanks -value 1
set_attribute -object $dmi2 -name nDataBanks -value 1
set_attribute -object $dmi2 -name cacheReplPolicy -value "RANDOM"

#-----------------------------------------------------------------------
# 2 Tag_Memory of DMI2
#-----------------------------------------------------------------------
set Tagmemories [get_objects -parent $dmi2 -type internal_memory -subtype TAG]
set tm0 [lindex $Tagmemories 0]
set_attribute -object $tm0 -name memoryType -value "FLOP"

#-----------------------------------------------------------------------
# 4 Data_Memory of DMI2
#-----------------------------------------------------------------------
set Datamemories [get_objects -parent $dmi2 -type internal_memory -subtype DATA]
set dm0 [lindex $Datamemories 0]

set_attribute -object $dm0 -name memoryType -value "FLOP"

#-----------------------------------------------------------------------
# 1 Write Data_Memory of DMI2
set WrDatamemories [get_objects -parent $dmi2 -type internal_memory -subtype WDATA]
set wrdm0 [lindex $WrDatamemories 0]
set_attribute -object $wrdm0 -name memoryType -value "FLOP"

#diis
set diis [get_objects -type dii -parent $chip]
set dii0 [lindex $diis 0]
set sysdii [lindex $diis 1]

#Set the premap attributes on dii0
set_attribute -object $dii0 -name nDiiRbCredits      -value 16

puts "###################################################"
puts "####   Setting Pre map parameters  complete   ####"
puts "###################################################"

puts "#############################################"
puts "####   User Design Creation complete.   ####"
puts "#############################################"

#Save the project
set fname ${designName}.premap.mpf
save_project -file $fname

########################################################################
################## Phase 4: Mapping
########################################################################
puts "#############################################"
puts "####   Start mapping process.           ####"
puts "#############################################"
set stat [map_design -name $topology]

if {$stat == 0} {
  puts "ERROR in map_design. Exiting ..."
  exit 1
  return
}

puts "#############################################"
puts "####   Mapping process complete.        ####"
puts "#############################################"

########################################################################
################## Phase 5: Refinement
########################################################################
puts "#############################################"
puts "####   Setting post-map parameters.     ####"
puts "#############################################"


#DCE settings
#DCE0
set dcePrefix mappedObjects/dce0_mapped
set_attribute -object $dce0 -name $dcePrefix/ImplementationParameters/nTaggedMonitors       -value 4

#DCE1
set dcePrefix mappedObjects/dce1_mapped
set_attribute -object $dce1 -name $dcePrefix/ImplementationParameters/nTaggedMonitors       -value 4
```

```
#ncaiu settings

#Setting parameters related to proxyCache
set prCachePrefix mappedObjects/ncaiu0_mapped/Cache
set_attribute -object $ncaiu0 -name $prCachePrefix/nSets                               -value 16
set_attribute -object $ncaiu0 -name $prCachePrefix/nWays                               -value 2
set_attribute -object $ncaiu0 -name $prCachePrefix/SelectInfo/PriSubDiagAddrBits       -value_list [list 6 7 8 9]

set prCachePrefix mappedObjects/ncaiu1_mapped/Cache
set_attribute -object $ncaiu1 -name $prCachePrefix/nSets                               -value 16
set_attribute -object $ncaiu1 -name $prCachePrefix/nWays                               -value 2
set_attribute -object $ncaiu1 -name $prCachePrefix/SelectInfo/PriSubDiagAddrBits       -value_list [list 6 7 8 9]


# DMI settings
#DMI0
set dmiPrefix mappedObjects/dmi0_mapped
set_attribute -object $dmi0 -name $dmiPrefix/useAtomic                                            -value false

# Settings for DMI configured with System Memory Cache
set smcPrefix mappedObjects/dmi0_mapped/Cache
set_attribute -object $dmi0 -name $smcPrefix/nSets                               -value 16
set_attribute -object $dmi0 -name $smcPrefix/nWays                               -value 2
set_attribute -object $dmi0 -name $smcPrefix/useScratchpad                       -value false
set_attribute -object $dmi0 -name $smcPrefix/SelectInfo/PriSubDiagAddrBits       -value_list [list 6 7 8 9]
set_attribute -object $dmi0 -name $smcPrefix/SelectInfo/TagBankSelBits           -value_list []
set_attribute -object $dmi0 -name $smcPrefix/SelectInfo/DataBankSelBits          -value_list []

#DMI1
set dmiPrefix mappedObjects/dmi1_mapped
set_attribute -object $dmi1 -name $dmiPrefix/useAtomic                                            -value false

# Settings for DMI configured with System Memory Cache
set smcPrefix mappedObjects/dmi1_mapped/Cache
set_attribute -object $dmi1 -name $smcPrefix/nSets                               -value 16
set_attribute -object $dmi1 -name $smcPrefix/nWays                               -value 2
set_attribute -object $dmi1 -name $smcPrefix/useScratchpad                       -value false
set_attribute -object $dmi1 -name $smcPrefix/SelectInfo/PriSubDiagAddrBits       -value_list [list 6 7 8 9]
set_attribute -object $dmi1 -name $smcPrefix/SelectInfo/TagBankSelBits           -value_list []
set_attribute -object $dmi1 -name $smcPrefix/SelectInfo/DataBankSelBits          -value_list []

#DMI2
set dmiPrefix mappedObjects/dmi2_mapped
set_attribute -object $dmi2 -name $dmiPrefix/useAtomic                                       -value false
set smcPrefix mappedObjects/dmi2_mapped/Cache
set_attribute -object $dmi2 -name $smcPrefix/nSets                               -value 16
set_attribute -object $dmi2 -name $smcPrefix/nWays                               -value 2
set_attribute -object $dmi2 -name $smcPrefix/useScratchpad                       -value false
set_attribute -object $dmi2 -name $smcPrefix/SelectInfo/PriSubDiagAddrBits       -value_list [list 6 7 8 9]
set_attribute -object $dmi2 -name $smcPrefix/SelectInfo/TagBankSelBits           -value_list []
set_attribute -object $dmi2 -name $smcPrefix/SelectInfo/DataBankSelBits          -value_list []


#DII settings
set diiPrefix mappedObjects/dii0_mapped
set_attribute -object $dii0 -name $diiPrefix/nLargestEndPoint -value 262144


puts "#####################################################"
puts "####   Setting post-map parameters complete.     ####"
puts "#####################################################"


#############################################################
# PHASE 6: EXPORT
##############################################################
file mkdir ./json
set intermediateJson ./json/top.level.json
set postmapMpf       ${designName}.mpf

export_design -format flat -file $intermediateJson
save_project -file $postmapMpf
puts "Design mpf created in $postmapMpf"


#GENERATE RTL
#----------------------------------------------------------------------
#gen_collateral

exit
```

1.3 Complete Manual Route Design Example

```
set tclLib "MAESTRO_TCLLIB"

if {[info exists env($tclLib)] } {
 set tclLibDir $env($tclLib)
 puts "Found envVar $tclLib = $tclLibDir"
} else {
 set script_path [ file dirname [ file normalize [ info script ] ] ]
 set tclLibDir [file join $script_path "../tcllib"]
 puts "Found local $tclLib = $tclLibDir"
}

if {[file isdirectory $tclLibDir]} {
 lappend auto_path $tclLibDir
 puts "auto_path = $auto_path"
 set pkgs [file join $tclLibDir pkgs.tcl]
 puts "Sourcing $pkgs"
 source $pkgs
} else {
 puts "$tclLibDir is not a directory"
 return
}

proc checkFile { fileName } {
 if { [file exists $fileName] == 1} {
  file delete $fileName
 }
}

set design_name packetStyle
checkFile $design_name.mpf

set project [create_project -name $design_name]
set chip [create_object -type chip -name "chip" -parent $project]

set powerreg [create_power_region -name pwrRegion -voltage "1.0V" -parent $chip]
set powerdom [create_power_domain -name pwrDomain -parent $powerreg -gating "always_on"]

set clkRegion [create_clock_region -name clk_reg -parent $chip -frequency "1.6GHz" \
-power_region $powerreg]
set clkDomain [create_clock_domain -name clk_dom -parent $clkRegion \
-gating always_on -power_domain $powerdom]
set clkSubdomain [create_clock_subdomain -name clk_subdom -parent $clkDomain]

set system [create_object -type system -subtype system -name "system" -parent $chip]
set subsystem [create_object -type subsystem -name subsystem -parent $system]

set subsystemType ARTERIS_COHERENT ; # ARTERIS_NONCOHERENT
set_attribute -object $subsystem -name subsystemType -value $subsystemType

set sk0 [create_interface -name sk0 -protocol "CHI-B" -function INITIATOR -role SLAVE \
-wdata 128 -waddr 48 -clock $clkSubdomain]
set sk1 [create_interface -name sk1 -protocol "CHI-B" -function INITIATOR -role SLAVE \
-wdata 128 -waddr 48 -clock $clkSubdomain]
set sk2 [create_interface -name sk2 -protocol AXI4 -function MEMORY -role MASTER -wdata 128 \
-waddr 48 -clock $clkSubdomain]

set mm [create_object -type memory_map -name "mm" -parent $subsystem]

set topology [create_object -type topology -name "topology" -parent $subsystem]
set_attribute -object $topology -name coherentTemplate -value "TwoCtrlOneDataTemplate"

set nDCE_directories 1
run_generator -name "interface_units" -topology $topology -clock $clkSubdomain \
-params [list directories $nDCE_directories]

# get the ncoreunits of the sockets
set ncoreunits [list ]
set sockets [get_objects -parent $chip -type socket]
foreach sk $sockets {
 set nc [get_objects -parent $sk -type ncoreunit]
 lappend ncoreunits $nc
 puts "$sk -> $nc"
}

# get the output and input ports of unit0 (socket0)
set unit0 [lindex $ncoreunits 0]
set unit0_outports [get_objects -parent $unit0 -type packet_port -subtype out]
set unit0_inports [get_objects -parent $unit0 -type packet_port -subtype in]

# get the output and input ports of unit1 (socket1)
set unit1 [lindex $ncoreunits 1]
set unit1_outports [get_objects -parent $unit1 -type packet_port -subtype out]
set unit1_inports [get_objects -parent $unit1 -type packet_port -subtype in]

# get the output and input ports of unit2 (socket2)
set unit2 [lindex $ncoreunits 2]
set unit2_outports [get_objects -parent $unit2 -type packet_port -subtype out]
set unit2_inports [get_objects -parent $unit2 -type packet_port -subtype in]
set networks [get_objects -parent $topology -type network]
set nNetw [llength $networks]
puts "Number of networks: $nNetw"

# create routes on the first network (network0)
set width 0
set i 0
set netw [lindex $networks $i]
set_context -name $netw

# create two switches on network0: sw1, sw2
set sw1 [create_object -parent $netw -type "switch" -name "sw1"]
set sw2 [create_object -parent $netw -type "switch" -name "sw2"]

# specify the ports of the route
# from unit0:outputPort0 thru sw1 thru sw2 to unit2:inputPort0
# source port: unit0 output port $i
set srcPort [lindex $unit0_outports $i]
set srcNetw [get_objects -parent $srcPort -type network]
puts "srcPort: $srcPort :: $srcNetw"
# target port: unit2 input port $i
set tgtPort [lindex $unit2_inports $i]
set tgtNetw [get_objects -parent $tgtPort -type network]
puts "tgtPort: $tgtPort :: $tgtNetw"

# the port index of both units should be the same so the ports are on the same network
# note: cannot create route if outputPort is on network0 and inputPort is on network1 (different networks).

# the source and destination are ports, the intermediate objects are switches
# this route goes thru sw1 and sw2
set ports [list $srcPort $sw1 $sw2 $tgtPort ]

# create the route using the ports
update_object -model route -type $width -add $ports

# create another route:
puts "Another route ... "
# source port: unit1 output port $i
set srcPort [lindex $unit1_outports $i]
set srcNetw [get_objects -parent $srcPort -type network]
puts "srcPort: $srcPort :: $srcNetw"

# target port: unit2 input port $i
set tgtPort [lindex $unit2_inports $i]
set tgtNetw [get_objects -parent $tgtPort -type network]
puts "tgtPort: $tgtPort :: $tgtNetw"

# the source and destination are ports, the intermediate objects are switches
# this route goes thru sw1 only
set ports [list $srcPort $sw2 $tgtPort ]

# create the route using the ports
update_object -model route -type $width -add $ports
```

Index