# Integration of Multiple NoCs

To integrate RTL from multiple NoCs into the same design for simulation the following steps must be followed:

* The NocStudio configuration file for each NoC must have a unique project name and contain the command:

prop\_default tag\_project\_name yes

This allows each NoC to have unique top-level module names and log file names.

* Run NocStudio on each configuration file to generate project directories for multiple NoCs. For each configuration file, NocStudio generates the following:

Table 8 Directory and files with tag\_project\_name

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Type** |
| noc\_verif\_ip\_proj/ | Custom verification files specific to each NoC. | Verification |
| ns\_<project>\_soc\_ip.v | ns\_soc\_ip.v for each NoC. | RTL |
| ns\_<project>\_fabric.v | ns\_fabric.v for each NoC. | RTL |
| ns\_<project>\_fabric\_modules.v | ns\_fabric\_modules.v for each NoC. | RTL |
| ns\_<project>\_group\_modules.v | ns\_group\_modules.v for each NoC. | RTL |
| ns\_<project> \_pp\_to\_pd\_map.txt | ns\_ pp\_to\_pd\_map.txt for each NoC. NOTE: Only generated for Low Power NoCs. | Verification |
| ns\_<project> \_pd\_to\_pdnum\_map.txt | ns\_pd\_to\_pdnum\_map.txt for each NoC. NOTE: Only generated for Low Power NoCs. | Verification |
| ns\_<project>\_preloader\_declare.sv | ns\_preloader\_declare.sv for each NoC. NOTE: Only present for Pegasus | Verification |
| ns\_<project> \_preloader\_init.sv | ns\_preloader\_init.sv for each NoC. NOTE: Only present for Pegasus | Verification |
| ns\_<project> \_preloader\_set.sv | ns\_preloader\_set.sv for each NoC. NOTE: Only present for Pegasus | Verification |
| ns\_<project>\_bind\_checkers.svh | ns\_bind\_checkers.svh for each NoC. | Verification |

* Once multiple NoC project directories exist, the next step is to construct a combined NoC project directory. This can be done either manually, or using the sample script, $NOCSTUDIO\_HOME/scripts/ns\_multi\_noc.pl where $NOCSTUDIO\_HOME is the installation directory.

## Automated Integration of Multiple NoCs

* The ns\_multi\_noc.pl sample script constructs a combined NoC project directory, stitches a combined NoC testbench and performs a sanity check by compiling and elaborating the generated NoC RTL.
* To get a list of options, invoke the following command:

$NOCSTUDIO\_HOME/scripts/ns\_multi\_noc.pl -h

* To run ns\_multi\_noc.pl, change to the installation directory and invoke the following:

If you are using Cadence Incisive Simulator, run:

$NOCSTUDIO\_HOME/scripts/ns\_multi\_noc.pl \

-i <project1> <project2> ... <projectN> \

-o <combined\_project>

If you are using Synopsys VCS, run:

$NOCSTUDIO\_HOME/scripts/ns\_multi\_noc.pl \

-i <project1> <project2> ... <projectN> \

-o <combined\_project> \

-VCS

Where,

* <project1> <project2> ... <projectN> denote the paths to input NoC project directories.
* <combined\_project> denotes the desired name of the output combined NoC project directory.

On a successful compile and elaboration, the following will appear at the prompt:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* BUILD PASSED \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A file named SIM\_SKIPPED will be generated in the combined NoC project directory indicating that only the build phase was performed. The log file ns\_multi\_noc.log will log the steps performed by the ns\_multi\_noc.pl script and any errors encountered. Depending on the simulator, the log file run\_test\_vcs.log or run\_test\_incisiv.log will also list any errors encountered during the build phase. The build logs, named build.log, are located in the model/model\_\*/ directory.

After a successful NoC sanity testbench build, the generated combined NoC RTL and verification IP are ready for integration into the user’s environment.

## Manual Integration of Multiple NoCs

* Create a combined NoC project directory and recursively copy the directories and files listed in Table 8 Directory and files with tag\_project\_name from each NoC project directory, along with the following directories, into the combined project directory.

Table 9 Additional directories to copy into combined NoC project directory

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Type** |
| noc\_modifiable\_rtl/ | RTL modules, such as RAM, that can be replaced by user implementation | RTL |
| noc\_rtl/ | NoC RTL library | RTL |
| noc\_verif\_cust/ | Holds ns\_global\_defines.vh file used for integration of CFG verification IP into customer environment | Verification |
| noc\_verif\_ip/ | NoC verification checkers IP library | Verification |
| noc\_rtl\_agents/ | NoC RTL agents library. | RTL |
| noc\_verif\_agents/ | NoC verification agents IP library | Verification |
| cpfs/ | NoC hierarchical CPF files. NOTE: Only generated for Low Power NoCs. | CPF |

* Edit the ns\_<project>\_bind\_checkers.svh files in the combined project directory. For multiple NoCs, there will be bind statements for the same module in more than one file. Resolve this by removing redundant bind statements for the same module.
* Create a combined ns\_node\_id\_table.sv file in the combined project directory by concatenating the contents of ns\_node\_id\_table.sv from each NoC.
* Create a combined ns\_routing\_table.sv file in the combined project directory by concatenating the contents of ns\_routing\_table.sv from each NoC.
* Create a combined ns\_power\_map\_table.sv file in the combined project directory by concatenating the contents of ns\_power\_map\_table.sv from each NoC. NOTE: Only present for Low Power NoCs.
* Create a combined ns\_global\_defines.vh by taking noc\_verif\_cust/ns\_global\_defines.vh from the first NoC project directory then adding any `defines from noc\_verif\_cust/ns\_global\_defines.vh of subsequent NoCs that are not already present. There should be no duplicate `defines. Remove any `defines pointing to checker instances that do not have a bind statement in ns\_<project>\_bind\_checkers.svh. For example,

`define NS\_<PROJECT1>\_NOC\_TOP <Hierarchical path to first NoC top module>

`define NS\_E2E\_CHECKER\_TOP \

`NS\_<PROJECT1>\_NOC\_TOP.ns\_amba\_noc\_e2e\_checker

`define NS\_REGBUS\_E2E\_CHECKER\_TOP \

`NS\_<PROJECT1>\_NOC\_TOP.ns\_regbus\_e2e\_checker

`define NS\_GCT\_TOP \ `NS\_<PROJECT1>\_NOC\_TOP.ns\_ace\_noc\_gct\_coherency\_checker

`define NS\_<PROJECT2>\_NOC\_TOP <Hierarchical path to second NoC top module>

`define NS\_ASYNC\_FIFO\_CHECKER\_TOP \

`NS\_<PROJECT2> \_NOC\_TOP.ns\_noc\_async\_fifo\_checker

Where,

* <project1> denotes the name of the project for the first NoC in lower case.
* <PROJECT1> denotes name of the project for the first NoC in upper case.
* <project2> denotes name of the project for the second NoC in lower case.
* <PROJECT2> denotes name of the project for the second NoC in upper case.
* Consolidate ns\_noc\_files.f from each NoC to create final simulation file list for multiple NoCs. For example,

+libext+.v+.sv

+incdir+$COMBINED\_PATH/noc\_rtl

+incdir+$COMBINED\_PATH/noc\_rtl\_agents/tunnel

+incdir+$COMBINED\_PATH/noc\_verif\_ip

+incdir+$COMBINED\_PATH/noc\_verif\_cust

-y $COMBINED\_PATH/noc\_rtl

-y $COMBINED\_PATH/noc\_modifiable\_rtl

-y $COMBINED\_PATH/noc\_rtl\_agents/tunnel

-y $COMBINED\_PATH/noc\_verif\_ip

-y $COMBINED\_PATH/noc\_verif\_ip\_proj

$COMBINED\_PATH/noc\_verif\_ip/ns\_amba\_struct.sv

$COMBINED\_PATH/noc\_verif\_ip/ns\_brdg\_cfg.sv

$COMBINED\_PATH/noc\_verif\_ip/ns\_mstrbrdg\_cfg.sv

$COMBINED\_PATH/noc\_verif\_ip/ns\_aximpabrdg\_cfg.sv

$COMBINED\_PATH/noc\_verif\_ip/ns\_regbus\_struct.sv

$COMBINED\_PATH/ns\_<project1>\_fabric\_modules.v

$COMBINED\_PATH/ns\_<project1>\_fabric.v

$COMBINED\_PATH/ns\_<project1>\_group\_modules.v

$COMBINED\_PATH/ns\_<project1>\_soc\_ip.v

-f $COMBINED\_PATH/noc\_verif\_bench/noc\_verif\_bench.vc

-f $COMBINED\_PATH/noc\_verif\_bench/noc\_verif\_bench.gemini.vc

+incdir+$COMBINED\_PATH/noc\_rtl\_agents/ip

-y $COMBINED\_PATH/noc\_rtl\_agents/ip

$COMBINED\_PATH/ns\_<project2>\_fabric\_modules.v

$COMBINED\_PATH/ns\_<project2>\_fabric.v

$COMBINED\_PATH/ns\_<project2>\_group\_modules.v

$COMBINED\_PATH/ns\_<project2>\_soc\_ip.v

Where,

* <project1> denotes the project name of the first NoC.
* <project2> denotes the project name of the second NoC.
* $COMBINED\_PATH denotes the path to the combined NoC project directory.
* For each additional NoC, simply replicate the same lines of <project2> to point to the unique files of the new NoC.

This mechanism can be used to integrate more than two NoCs with no upper limit to the number of NoCs.