# Feature Updates

## NoC Construction Improvements

### Isolate and reduce congestion

NocStudio can now take special consideration while mapping high bandwidth traffic (based on the traffic rate or overall load on the transmitting and receiving interfaces) to isolate them from other traffic so as to reduce congestion in the NoC.

* The user can also sort traffic flows so as to map high bandwidth traffic flows first using the option “high\_rate\_traffic\_flows” in the sort order argument to the map\_opt command. NocStudio also automatically uses “high\_rate\_traffic\_flows” as one of the many sort orders to sort traffic flows in each iteration of map\_opt.
* The user can choose to make NocStudio avoid the mapping of high rate traffic flows and low rate traffic flows on the same VC as part of its mapping algorithm by using the “separate\_high\_rate\_traffic\_flows” argument in map\_opt.

### Automated FIFO sizing

Automatic sizing of FIFO depth on VC’s with high bandwidth traffic during tune\_links to prevent congestion in the NoC and head of line blocking.

### Automated tune\_links

*tune\_links* is now run automatically at the end of map\_opt to tune the link widths and FIFO depths. *tune\_links* is also a separate command that a customer can choose to run.

## GUI Enhancements

### Hide toolbars to increase screen real estate

Press Ctrl+H to hide/unhide the drawing toolbar and the top toolbar to increase the screen real estate.

### Minimize the toolbars and the property panels to side tabs

The drawing toolbar, top toolbar and the property toolbar can be minimized to the sides to appear as tabs.

### Hide inactive layers after mapping

The layers that don’t have any traffic mapped on them are automatically hidden after mapping.

### New way to add/remove link and port blockages in the Blockage View

In the blockage view, left click on a link to add/remove a link blockage (add/remove blockage to that link on all layers), and right click to add/remove a port blockage (add/remove blockage on the link only on the clicked layer).

### Collapse all the buttons to add the different agents and bridges to single ‘Add’ button

The add buttons of different agents (Bridge and SIB) are all collapsed into one ‘Add’ button.

### Errored commands remain on console for editing

An errored command remains on the console so that the user can edit it rather than having to retype the entire command.

### Change color of link when clicked

When a link is clicked, its color changes to pink so that the user knows which link he is currently viewing the state/properties of.

## Multi-NoC

RTL from multiple NoCs can now be integrated into the same design for simulation. Each NoC must have a unique mesh name and the following NocStudio property setting:

prop\_default tag\_project\_name yes

Two or more NoCs can be integrated with no upper limit to the number of NoCs. Only one coherent NoC is supported per set of NoCs. Crux NoCs cannot be mixed with Orion/Gemini NoCs.

## Functional Safety

For safe and reliable operation of the device, NocStudio now supports:

1. End to end transport error checking, including Data/Sideband ECC Protection, Data/Sideband Parity Error Detection
2. Hop to hop error checking of packet fields
3. Error reporting

For more details, please refer to Chapter “Safety and Reliability” in the TRM.

## Synchronizer Depth and FIFO Sizing

NocStudio now supports programmable synchronizer depths, both for general synchronizers (ns\_demet.v) and for reset synchronizers (ns\_rst\_n.v). Users may select synchronizer depths, based on their own library/process requirements, on a per-clock-domain granularity.

## Traffic Class Optimization

In many cases, it may make sense to have requests and responses have different traffic classes. Traffic flows to memory, for instance, may want to have different traffic classes for the different requests. This is particularly true when different priorities are used for the traffic classes.

The response path may not need separate traffic classes. If a low priority response is stuck leaving the memory controller, it can block high priority responses as well. It will often make sense to have these responses share a single traffic class.

This can also significantly reduce area within a NoC. Traffic classes require separate virtual or physical channels, which requires more storage and possibly more links. While this may be useful for traffic going to memory, the expense of the response path may be unnecessary. And since data paths are often wider than request paths, this can be a significant area impact.

NocStudio allows each hop of a traffic flow to indicate a separate traffic class, to provide user with the flexibility to configure various requests and responses to either use different traffic classes or share them.

## User Defined System Resource Override

Support for user defined NOC system level resource for compatibility of families of products. In NoC synthesis flow, the tool automatically calculates the resource needed for a given project to provide the most optimized solution. But in terms of derivative or re-spin, due to some ECO changes, NocStudio may expand the resource (e.g. AID width). In order for the NocStudio to preserve the NOC resource, SOC architect/designer needs to predict the address width, AID width, User width, etc. and use mesh\_prop “sys\_\*\_width <N>” to override the System level NOC resource.

## Verification Updates

### Updated mailbox syntax

A new argument “proj\_tag” has been added as follows:

Old syntax:

*`NS\_E2E\_CHECKER\_TOP.ns\_transaction\_src\_mbox.try\_get(ns\_src\_transaction)*

New syntax:

*`NS\_E2E\_CHECKER\_TOP.ns\_transaction\_src\_mbox[proj\_tag].try\_get(ns\_src\_transaction)*

where “*proj\_tag*” is a string that should be,

- Set to project\_name if the noc is using ‘*tag\_project\_name*’

- Set to an empty string if the noc is not using ‘*tag\_project\_name*’

## System C Model

In order to integrate C++ based models into custom environments it is necessary to define interfaces and APIs to enable integration and interacting with 3rd party models, debuggability and analysis of results. NetSpeed provides C++ based models for such integration purposes based on the SystemC TLM2 standard. The current mode specifically addresses the need for fast functional modeling in order to fit into platform models for software and firmware verification and development. Such models are typically refered to as LT (loosely timed) or PV (programmers view).