

# NetSpeed IP Safety Manual

## *Excerpts For Intel Corporation*

**Version: 1.0**

August 23, 2017

# NetSpeed IP Safety Manual

## About This Document

This Safety Manual for NetSpeed IP provides an overview of the safety requirements applied within the development of the product name in safety-critical automotive and industrial designs according to ISO 26262.

## Audience

This document is intended for users of NetSpeed IP:

- SoC Architects
- NoC Architects
- NoC Designers

## Prerequisite

Before proceeding, you should generally understand:

- Basics of Network on Chip technology
- AMBA 4 interconnect standard

## Related Documents

The following documents can be used as a reference to this document.

- NetSpeed Orion Integration Specification
- NetSpeed Orion Technical Reference Manual

## Customer Support

For technical support about this product, please contact [support@netspeedsystems.com](mailto:support@netspeedsystems.com)

For general information about NetSpeed products refer to: [www.netspeedsystems.com](http://www.netspeedsystems.com)

# Contents

<b>About This Document .....</b>	<b>2</b>
<b>Audience .....</b>	<b>2</b>
<b>Prerequisite .....</b>	<b>2</b>
<b>Related Documents.....</b>	<b>2</b>
<b>Customer Support.....</b>	<b>2</b>
<b>Contents .....</b>	<b>3</b>
<b>Figures .....</b>	<b>7</b>
<b>Tables.....</b>	<b>8</b>
<b>1 Introduction .....</b>	<b>9</b>
<b>2 Assumed Safety Requirements .....</b>	<b>10</b>
2.1 Assumption of Use .....	10
2.2 NetSpeed IP Network on Chip (NoC) .....	10
2.3 NoC IP In System Application .....	10
<b>3 Functional Safety Approach .....</b>	<b>12</b>
3.1 Top-Down Approach.....	12
3.2 Functional Safety - System Level.....	12
3.2.1 Safety Goal & Safety Motivation .....	12
3.3 Functional Safety –Component Level.....	13
3.4 NocStudio to Create NoC IP .....	15
<b>4 Internal Safety Mechanisms.....</b>	<b>16</b>
4.1 NetSpeed Architecture Refresher.....	16
4.2 Summary of Functional Safety, Reliability and Availability .....	16
4.3 Error Detection and Correction Features .....	17
4.3.1 ECC Algorithm .....	17
4.4 NoC-transport Error Detection and Correction .....	18
4.4.1 End-to-end transport integrity.....	18

4.4.2	End-to-end user protection .....	18
4.4.3	Interface Parity .....	19
4.4.4	ARM Cortex R5/R7 Port compatibility .....	19
4.4.5	Hop-to-hop Protection .....	19
4.4.6	End-to-end Packet Integrity .....	19
4.5	Logic protection and redundancy .....	20
4.5.1	Flop structure Parity protection .....	20
4.5.2	Bridge Duplication .....	20
4.5.3	Route Duplication.....	20
4.5.4	Architectural Support for Redundancy .....	21
4.5.5	NoC Register Parity Checking .....	22
4.6	RAM protection features .....	22
4.6.1	Data ECC for RAMs .....	22
4.6.2	Address ECC for RAMs.....	22
4.7	Coherency protection.....	23
4.8	Timeouts .....	23
4.8.1	Target Timeouts .....	23
4.8.2	Initiator Timeouts .....	23
4.8.3	NoC Timeouts .....	23
4.9	Network BIST Architecture .....	23
4.9.1	BIST Control Interface.....	24
4.9.2	BIST Packet Length and Value .....	25
4.9.3	Concurrent BIST and Normal Packets.....	26
4.9.4	Power Requirements .....	27
4.9.5	Reset Sequence .....	27
4.10	Error Logging and Fault Reporting.....	27
4.11	Configuration options through NocStudio .....	27
<b>5</b>	<b>External Safety Mechanisms.....</b>	<b>28</b>

5.1	RAM protection features .....	28
5.1.1	Data ECC for RAMs .....	28
5.1.2	Address ECC for RAMs.....	28
<b>6</b>	<b>Safety Lifecycle .....</b>	<b>29</b>
6.1	Product Life cycle flow .....	29
6.2	Safety Life Cycle – SLC .....	29
6.2	NetSpeed SLC flow .....	30
6.3	Concept .....	30
6.3.1	Requirements .....	30
6.3.2	Safety Requirement tracking.....	31
6.3.3	Safety Requirement Documentation.....	32
6.4	Product Development .....	33
6.5	Post-release phase.....	34
6.5.1	Field monitoring .....	34
<b>7</b>	<b>Development Plan .....</b>	<b>36</b>
7.1	IP Design and Integration .....	36
7.1.1	Hardware Design Flow.....	36
7.1.2	Hardware Design Specification.....	36
7.1.3	Design Methodology.....	37
7.1.4	Design Documentation .....	37
7.1.5	Design Checklist .....	38
7.2	Development Verification Plan .....	38
7.2.1	Verification cycle .....	39
7.2.2	Inputs to Verification Plan .....	39
7.2.3	TestPlan Template .....	39
7.2.4	Levels of testing .....	40
7.2.5	Types of tests.....	41
7.2.6	Result analysis.....	42

---

7.2.7	Safety, Reliability Verification .....	42
7.2.8	Coverage metrics .....	43
7.1	Regression testing.....	43
7.1.0	Bug tracking - JIRA.....	44
7.1.1	Checklist.....	45
<b>8</b>	<b>Safety Analysis Result.....</b>	<b>47</b>
8.1	FMEDA .....	47
8.2	DFA.....	47
<b>9</b>	<b>Terminology.....</b>	<b>48</b>

# Figures

Figure 1 NetSpeed Network IP (NetSpeed components) NoC .....	10
Figure 2 NoC Applications Systems.....	11
Figure 4 Appropriate NoC with Functional Specs through NocStudio NCF .....	15
Figure 5 NetSpeed NoC Architecture .....	16
Figure 6 NetSpeed Safety and Resilience Feature Support.....	17
Figure 7 Route duplication .....	21
Figure 8 Compound bridge to address redundant port checking.....	22
Figure 9 Logical Diagram of Network BIST .....	24
Figure 10 BIST Control .....	25
Figure 11 Router FIFOs .....	26
Figure 3 Safety Life Cycle .....	29
Figure 4 Development flow .....	30
Figure 12 EPIC 17.09 in JIRA – Requirements Traceability.....	31
Figure 13 Safety Requirements as part of EPIC 17.09 in JIRA .....	31
Figure 14 ISO26262-Part2.....	32
Figure 15 Test Plan and Design Flow .....	33
Figure 16 Postsales customer engagement flow .....	34
Figure 17 Design Flow .....	36
Figure 18 Requirement List .....	37
Figure 19 Verification Process .....	39
Figure 20 Verification hierarchy .....	40
Figure 21 Palladium Emulation .....	41
Figure 22 NoCWeaver, CXT, Nemesis.....	41
Figure 23 Coverage .....	43
Figure 24 Build schedule.....	44
Figure 25 Bug Tracking .....	44

# Tables

Table 1 Packet Pattern	25
Table 2 Checklist	45



# 1 INTRODUCTION

---

This Safety Manual for the NetSpeed IP products provides an overview of the safety requirements applied within the development of the product name in safety-critical automotive designs according to ISO 26262.

The manual's purpose is to provide an overview of the e. g. safety processes, flows, safety requirements including description, plans, and reports completed to help design teams accelerate their ISO 26262 certification of their automotive design where NetSpeed IP is embedded.

NetSpeed's Safety Flow considers the NoC development phases mapped to various parts of ISO-26262 requirements and supports the tailoring of the activities to be performed during the lifecycle phases.

## 2 ASSUMED SAFETY REQUIREMENTS

### 2.1 ASSUMPTION OF USE

NetSpeed IP is a configurable interconnect IP supporting various features for interface/protocol, performance, QoS, and coherency, to name a few. The customer can configure the IP per their system requirements. This is true also from the perspective of functional safety and ISO 26262. NetSpeed does not have any further constraints or assumptions on configurations and use. Since this is an IP that will be integrated into an SoC/chip, it is the responsibility of the integrator to define system safe states and the appropriate FTTIs based on system requirements.

### 2.2 NETSPEED IP NETWORK ON CHIP (NoC)

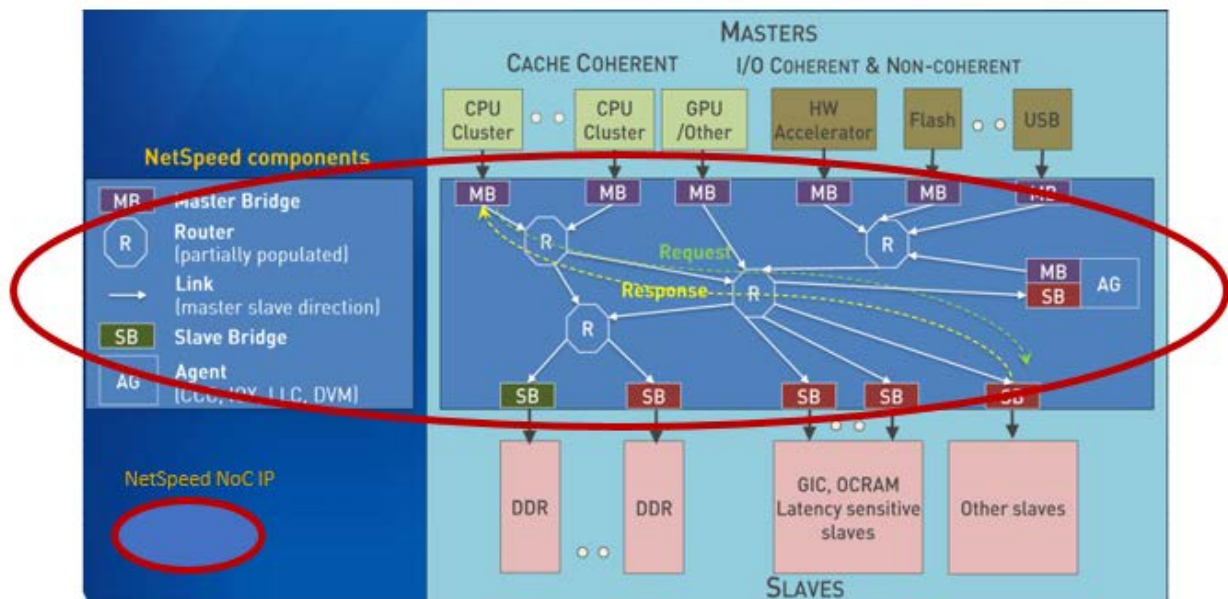


Figure 1 NetSpeed Network IP (NetSpeed components) NoC

NoC is a component in the SoC along with all other hierarchical components like CPU or GPU or DDR. Along with FMEDA and a safety analysis document Safety Manual, all the details required to qualify NoC to required ASIL level are documented.

### 2.3 NoC IP IN SYSTEM APPLICATION

NoC IP gets used in a system application like Automotive ADAS (Advanced Driving Assistance Systems), AR/VR (Augmented Reality/Virtual Reality) or IIoT (Industrial Internet of Things). The

building blocks of NetSpeed NoC IP are bridges, routers, links and agents with caches. Bridges Connect a master or a slave block to the network of routers and links via a well-defined protocol.

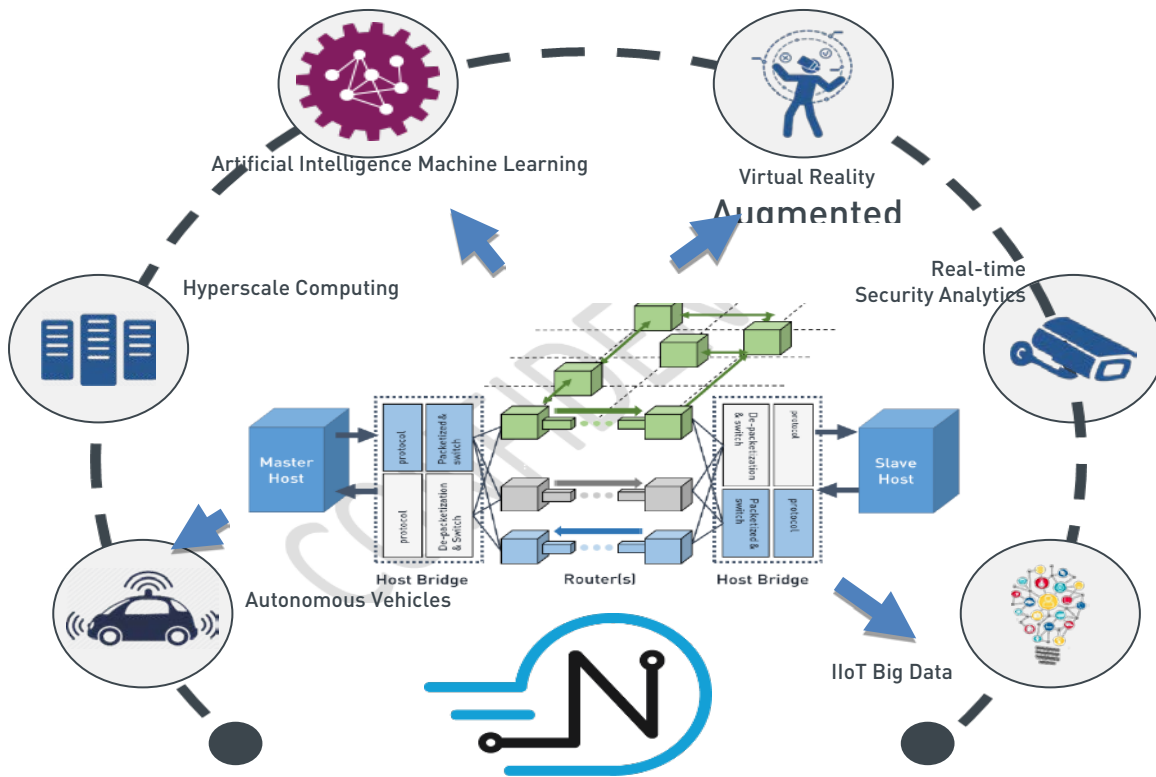


Figure 2 NoC Applications Systems

Bridges packetize the host's transactions into NetSpeed packet format during injection into the network and de-packetizes them during ejection. A router can have four directional links, referred to as north (N), south (S), east (E), and west (W). It also can have up to four additional links to connect to up to four hosts (H, I, J, K). All eight links are identical and can be attached to bridges or to other routers.

---

## 3 FUNCTIONAL SAFETY APPROACH

---

### 3.1 TOP-DOWN APPROACH

NetSpeed's IP follows a top down approach guided by ISO 26262 safety lifecycle. In the product, Functional Safety Base Requirements are the root requirements. They are specified by the user as input in the form of a 'Config file' to the software that generates the IP (NocStudio). NoC with desired system functionality and safety measures are part of this config file. (NocStudio config file defining functional requirements). This is validated at the architectural level prior to generating this IP for validation. The generated IP is then verified for correctness prior to release.

### 3.2 FUNCTIONAL SAFETY - SYSTEM LEVEL

- NetSpeed provides NoC IP as product with functional safe architecture based on customer or end use-case requirements for final SoC. Safety features in all possible NoC designs or configurations are summarized in the document AN\_505\_Safety\_Serviceability\_Security\_Features.pdf
- Safety is not add-on feature, meaning safety cannot be added at the end of product development as additional feature. Product development phase starts with functional safe requirements from the concept to requirements, design followed by implementation phases.
- Safety feature design is an integral part of the design flow. Safety requirements are handling in the same process as non-safety features and go through the development and verification procedures. In other words, safety features are not added on to the NoC as an after thought.

The flow from requirements to release is explained in the subsequent sections of this documentation. All references to non-safety related features of the product apply to safety features fully.

#### 3.2.1 Safety Goal & Safety Motivation

##### 3.2.1.1 *Safety Goal*

Traditional approaches to hardware safety and security features are limited to errors that could be dealt with at the hardware level. In the traditional approach, an unrecoverable hardware error brings down an entire SoC and the applications running on it. This approach is unacceptable to mission-critical applications like ADAS and cloud computing applications where the IP is targeted for usage. These SoCs require the handling of unrecoverable hardware errors, while delivering uninterrupted application and transaction services to end users.

NetSpeed NoC enables SoCs to handle unrecoverable errors end to end: from the underlying hardware to the application software.

#### 3.2.1.2 *Coherency issue*

Safety analysis results relate to the actual design are documented in the FMEDA Spreadsheet (Confluence - NetSpeed SoC Interconnect 20170406). Safety engineers keep track of ongoing changes in design through a detailed process. These changes are resulting from new requirements/features/improvements. They are tracked with regular automated process such that all degradations are controlled till closure of each change.

#### 3.2.1.3 *Plausibility issue*

FuSa features are implemented and verified on a regular basis in a well-defined process. Safety integrity levels in FMEDA are also analyzed with systematic procedures for ASIL level.

Test procedures do fault injections to cover all paths, states and its transitions. They also check parameter value ranges and respective boundary cases.

#### 3.2.1.4 *Accuracy issue*

ASIL/Safety phases are assessed in NetSpeed process without over-engineering and targeted to ensure numerical thresholds are always maintained.

#### 3.2.1.5 *Completeness issue*

NetSpeed Safety engineers enumerate all minimal safety cases and add them as assertions into the product as errors.

### 3.3 FUNCTIONAL SAFETY –COMPONENT LEVEL

Functional safety is not just an issue at system level but also at component level. Integrated safety devices for NoC customer gets detailed directions through parity and/or ecc to be selectively added as a part of configuration file and also described in technical reference manual.

NetSpeed NoC IP is expressed as a single component to the product SoC. NetSpeed NoC is the transport or networking IP and every safety element, such as safety at the boundary through interface parity or safety deep inside the NoC can be added as Link Safety and/or at the register level safety as in ISO 26262: Safety Element out of Context (SEooC).

Basic approach is to assume a system context (or several) of the component. NocStudio helps customer to define details of NoC IP. Safety Application Guide (Safety Elements in NoC

**Reference Manual**) specifies how the safety inside or at the interface of the NoC is applied correctly in the assumed NoC system context.

NoC Reference Manual is one of the output deliverables from NocStudio (auto generated by Customer in one or more iterations).

The purpose of NoC is to support data transactions from agents or different types of hosts like CPU(s), GPU(s), DSP(s) etc. and peripherals like memory controller, SATA, PCIe etc. through bridges and routers using designated or right protocols for transportation.

For safe and reliable operation of a device, error free and fault-tolerant operation of the interconnection networks used in the device is crucial. Random faults can occur in the storage elements and wiring resources used by a system wide interconnect. Such errors must be detected and corrected when possible and all uncorrected errors must be notified to system software for intervention.

NoC functional requirements covers all possible transportation of the data across the specific components as selected by the NocStudio configuration. To ensure safety across such transportation of packets various safety features are being adopted from concepts and requirements phase of the development process.

Ensuring functional safe transportation of packets across NoC fabric, error detection and corrections techniques are incorporated such as

- End-to-end Transport Error Checking
- Hop-to-hop Error Checking
- Configuring Error Checking
- End-to-end packet integrity
- End-to-end packet stream integrity

The inputs selected to the NocStudio or the configuration file will do ECC computation based on user specified maximum granularity. ECC generation at the transmitting end and detection and correction at the receiving end will add a cycle each to overall path latency thus adding PPA with safety feature.

As an alternative to ECC, user may configure parity to be transported with the data. Parity based protection does not add latency to the path. So, NocStudio gives the flexibility to the end user or SoC designer or customer to select safety features.

### 3.4 NOCSTUDIO TO CREATE NOC IP

NocStudio is a Network-on-Chip (NoC) architecture platform for ASIC designers and SOC architects with design flow in three step processes as Specify, Customize and Generate as follows.

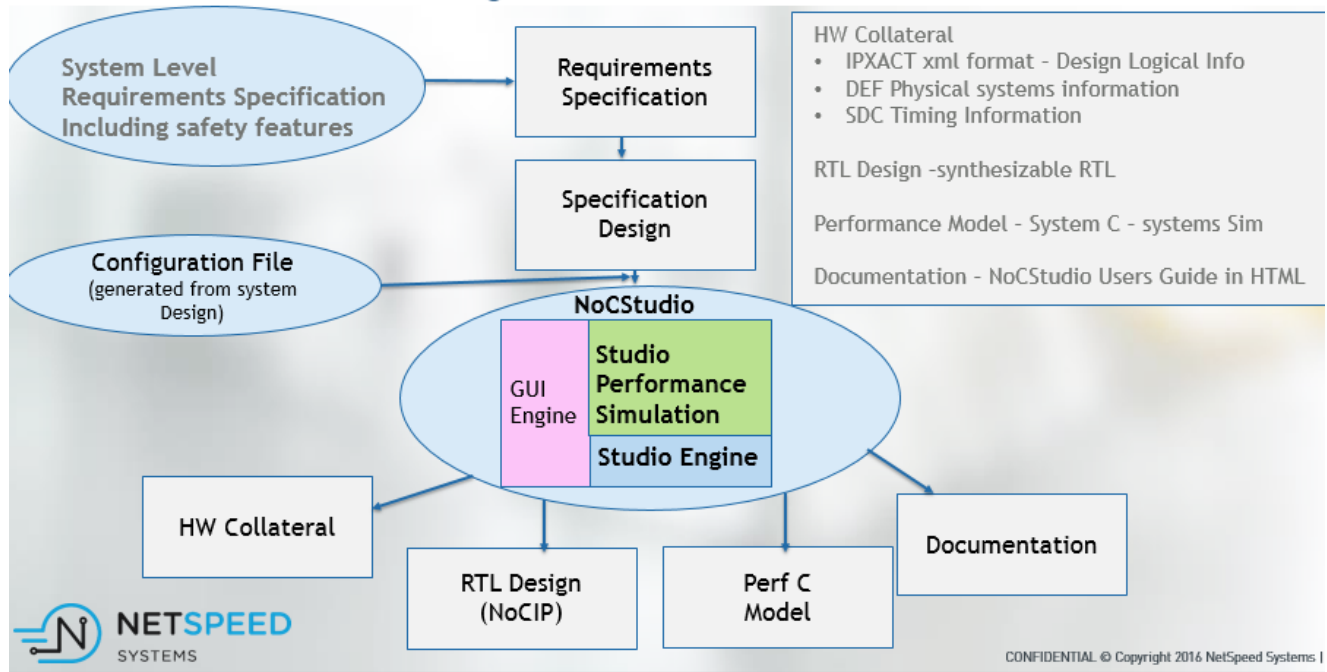


Figure 3 Appropriate NoC with Functional Specs through NocStudio NCF

NoCstudio helps user to **specify** a) IP hosts, locations, respective interface protocols b) Address map, Connectivity, Dependencies and c) the bandwidth and real-time requirement or QoS for the traffic

The next step is **Customize** in this phase the NocStudio

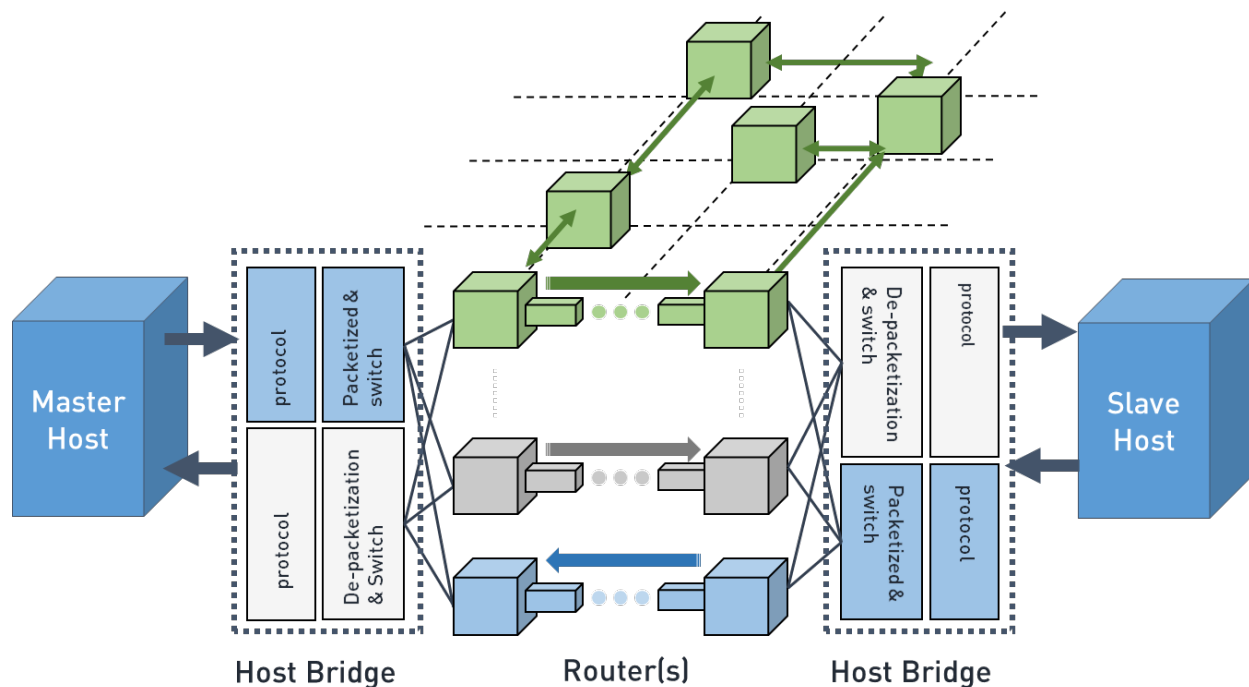
- adds the Link & Buffer cost
- adds the routing channels, considering the physical blockages and then the
- PPA – Power, Performance and Area gets calculated.

Customer can iterate this step by changing the location of the items or can use different traffic requirement or msg\_rate/beat rate or transaction rate and can meet desired PPA with respective SoC requirement.

## 4 INTERNAL SAFETY MECHANISMS

### 4.1 NETSPEED ARCHITECTURE REFRESHER

Figure 5 below shows high level architecture of the NetSpeed NoC IP. A bridge can connect a master or slave block to the NoC and perform the required operations to support the master and slave communication as per the protocol standard. The bridge packetizes the host blocks' transactions into NetSpeed packet format during injection into the NoC and de-packetizes them during ejection. The bridge connects to the router networks. A router can have four directional links, referred to as north (N), south (S), east (E), and west (W). It also can have up to four additional links to connect to up to four hosts (H, I, J, K). All eight links are identical and can be attached to bridges or to other routers.



*Figure 4 NetSpeed NoC Architecture*

### 4.2 SUMMARY OF FUNCTIONAL SAFETY, RELIABILITY AND AVAILABILITY

Figure 6 below summarizes the functional safety features provided by NetSpeed across the different interconnect components



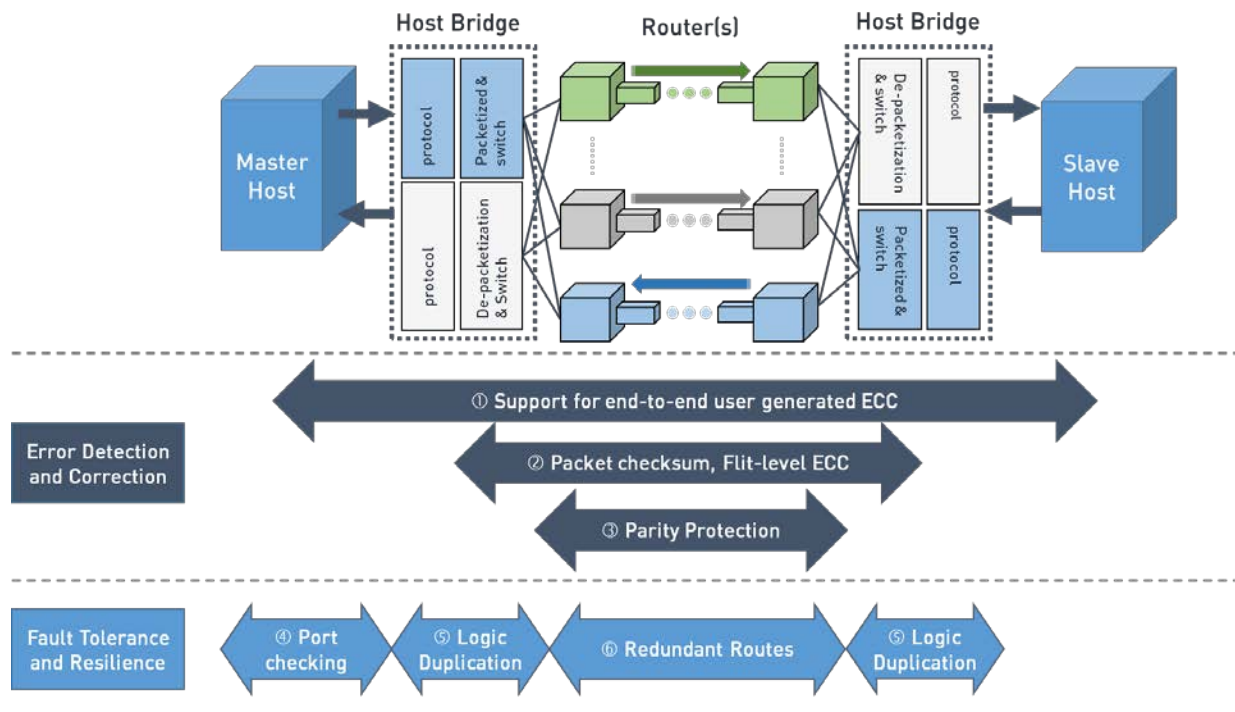


Figure 5 NetSpeed Safety and Resilience Feature Support

### 4.3 ERROR DETECTION AND CORRECTION FEATURES

Handling errors requires first detecting that an error has occurred. The current process for ensuring reliable hardware performance is to detect and correct errors where possible, recover from uncorrectable errors through either physical or logical replacement of a failing component or data path, and prevent future errors by replacing in a timely fashion components most likely to fail. Error correcting codes (ECCs) were devised to enable the detection and correction of errors. One ECC in common use is SECDED (single error correct double error detect), which allows the correction of one bit in an error or detection of a double-bit error in a memory block. Hardware errors can be classified as either (1) detected and corrected errors (DCE) or (2) detected but uncorrected errors (DUE). Handling DCEs is done in silicon using ECCs and can be made transparent to system components. Handling DUEs, needs collaboration from multiple levels of abstraction in the hardware-software stack.

#### 4.3.1 ECC Algorithm

If the NoC or directory is configured to have ECC, the IP implements a customized ECC algorithm. Additional bits are added to the NoC datapath and directory RAM array widths to hold ECC information. The bridge packetization and directory control logic handles generating ECC values and checking them in the NoC destination or in the directory read results to confirm that there is no error. The ECC algorithm uses a hamming code with an additional parity bit,

sometimes referred to as SECDEC (single error correction, double error detection). The algorithm adds the ECC checkbits to the protected data block, so all bits are protected with single-bit correction, and double-bit detection. The hardware supports a register mechanism to directly access the directory RAM, including the ECC checkbits. It supports multiple variants, including a method to take an existing directory entry and flip one or more bits before writing it back into the array. This can be used to test ECC logic within the system. The ECC detection and correction can also be disabled via register access.

## **4.4 NOC-TRANSPORT ERROR DETECTION AND CORRECTION**

### **4.4.1 End-to-end transport integrity**

#### **4.4.1.1 Data ECC Protection**

Within the NetSpeed IP, data (including byte enables) ECC is implemented at the flit/sub-flit level in NetSpeed NoC infrastructure to provide transport integrity. ECC function is single bit correction, double bit detection. To deal with variable width interfaces, ECC is implemented at the granularity of minimum NoC link or user specified granularity, whichever is smaller. Multiple ECC fields are present for wider links. Sideband signal are also protected with ECC.

The ECC is created at the ingress point and the default mode is for ECC to be checked at egress from the network for the packetized transaction. However, at the expense of additional area, error detection and correction can be configured to be added on a per-hop basis inside the NoC increasing robustness.

#### **4.4.1.2 Data Parity Protection**

ECC detection and correction comes at a cost to area, and hence NetSpeed provides the user with the option to implement data parity. The granularity and coverage of the protection is similar to the ECC methodology. Data parity does not cause any latency additions to the path.

#### **4.4.1.3 Sideband ECC or Parity Protection**

NetSpeed IP also protects the information carried in packet sideband, with end-to-end ECC or parity. At the transmitting end, ECC is calculated on sideband segments at the selected granularity and at the receiving end, error detection and correction is performed.

### **4.4.2 End-to-end user protection**

NetSpeed IP also provides the user a configurable option to generate their own ECC and the NetSpeed NoC transports them to the receiving end. NetSpeed IP passes host generated ECC in data and control packets using user-bit fields. The ECC information originates and terminates in the host logic.

#### **4.4.3 Interface Parity**

NetSpeed NoC provides advanced parity protection on the interface to the hosts. This adds protection for the datapath from the host IPs into the NetSpeed bridges. This also offers coverage of the ASYNC FIFO, skid stage and ratio sync buffer.

The coverage and granularity provided by the parity protection depends on the type of signals and varies between the various channels. For example, for the data interface, the granularity is configurable all the way from one bit for all data bits or one bit per 8-bits.

Parity is valid for every beat of information on these interfaces and the parity is checked off at the receiving end of the same interface before any transformation is performed within the bridge. This augmented with the NoC End-to-end transport integrity, provides a true End-to-end from host to host.

#### **4.4.4 ARM Cortex R5/R7 Port compatibility**

With the advent of cores built for these markets, some interfaces already have protection related signals defined and associated as part of the physical ports. The ARM R5/R7 cores have ports protected with ECC and parity for the various parts of the interface. NetSpeed provides the option to generate ECC and parity compatible with the AXI port protection in the ARM Cortex R5/R7 cores. This not only eases user integration but more importantly leverages some of these interface features.

#### **4.4.5 Hop-to-hop Protection**

##### **4.4.5.1 Control Parity Protection**

NetSpeed IP can be configured to generate parity protection for packet fields which can get updated hop-to-hop within the NoC. For example: routing information can be parity protected. These packet fields can undergo modifications and hence it is vital to not only protect but also check, and recompute on a per hop basis.

##### **4.4.5.2 Error detection using e2e ECC/Parity**

Detection and correction of ECC comes at a cost of area and latency. NetSpeed IP provides the user with an additional configurable option of data error detection (only) at a hop-to-hop basis, using the ECC or Parity carried to protect the data and sideband. This does not incur extra latency, since it is only detection, but provides a way to localize any error, and to identify any issue sooner, rather than waiting until a check at the receiver.

#### **4.4.6 End-to-end Packet Integrity**

NetSpeed IP provides robust means of confirming integrity at the packet level to detect missing data or misrouted packets. A packet can be made up of multiple flits, and additional protection

is needed to check for integrity of complete packets exchanged on the NoC. This is done by including a checksum that covers the entire transaction payload (including any address and control fields that must pass unaltered end to end) as well as some basic identifying information such as destination ID, source ID, sequence number, etc. Advanced techniques such as bit interleaved parity and flit identifiers further enhance the robustness of the NetSpeed IP by ensuring tolerance to errors. All these techniques are configurable to provide users with the ability to choose the desired level of protection vs cost tradeoff.

## **4.5 LOGIC PROTECTION AND REDUNDANCY**

Once the transaction is framed into a packet, NetSpeed IP can verify correct transmission through the mechanism described in previous sections. However, to guarantee end-to-end resilience, we need to protect the logic that frames the transaction on ingress and unpacks it at the egress. This is done by having duplicated logic with equivalence check at the NetSpeed bridges.

### **4.5.1 Flop structure Parity protection**

As a first line of defense, NetSpeed provides the option to protect the large logic structures with parity. This comes at a low cost compared to duplication. Key design features including buffers, flop arrays, registers, constant parameter arrays, can be configured to be protected with parity to ensure that faults can be detected, in these structures that are an integral part of the path.

This applies to the flop structures in the following components:

- Bridges
- Routers
- CCC (Cache Coherency Controller)
- IOCB (IO Coherent Bridge)
- LLC (Last Level Cache)
- DAU (Deadlock Avoidance Unit)

### **4.5.2 Bridge Duplication**

For systems that require a higher level of protection, NetSpeed also provides the configurable option to duplicate entire bridges. This provides the utmost protection of the bridges from errors. To ensure that the redundant unit is not similarly affected by error as the original, isolation is achieved by delaying the redundant unit by a clock cycle. Also a separate clock and reset input are provided to isolate them from glitches.

### **4.5.3 Route Duplication**

The one other piece of the datapath that needs protection is the actual routes between the bridges. This is accomplished in an algorithmic way by duplicating entire routes between transmitter and receiver end points. Only one physical route would be active at any given

instant, but under software control the routes can be changed and swapped. If a route is compromised due to errors, the SW would have control to swap to a different route. This is completely under software control and most importantly has a very low area overhead compared to duplication of the routers themselves.

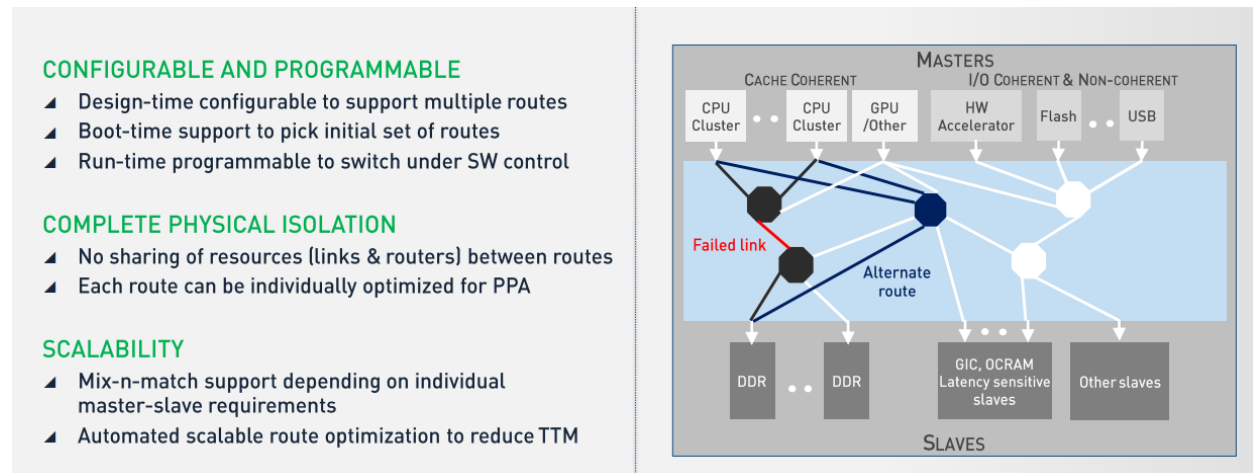
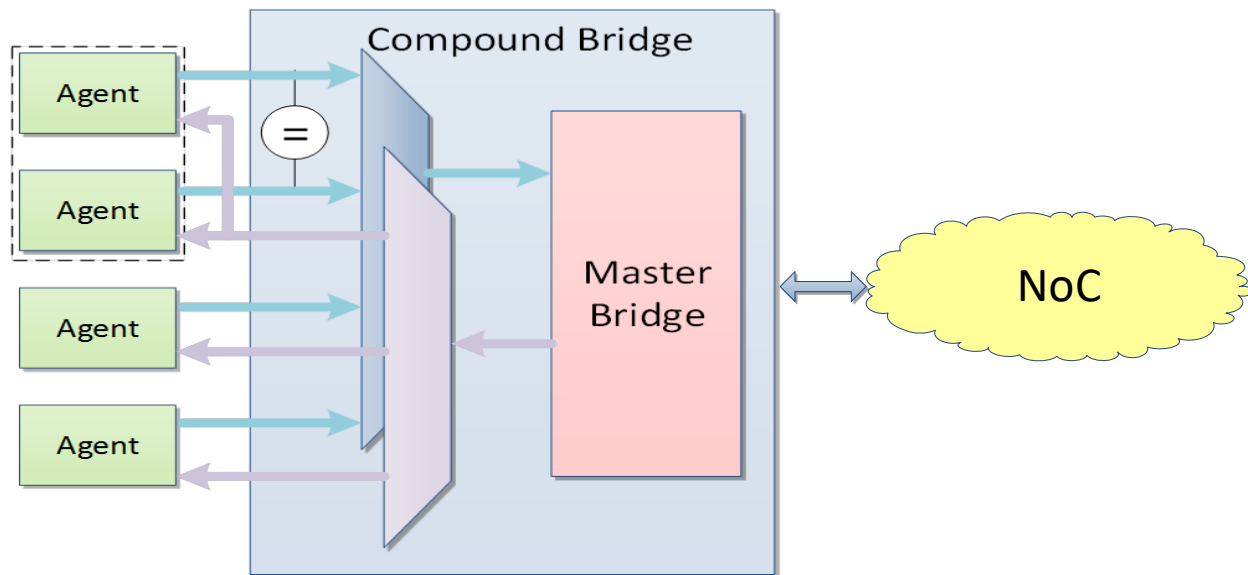


Figure 6 Route duplication

#### 4.5.4 Architectural Support for Redundancy

Hardware errors can affect computed results, data stored in memory, and data in transit between components. Such errors affect the accuracy, reliability, and integrity of computations. Hardware errors fall into two categories: soft errors and hard errors. Soft errors mostly occur because of random events affecting electronic circuits at the molecular level, such as alpha particles or cosmic rays dislodging electrons and therefore moving charges from one part of a circuit to another. Hard errors are permanent physical failures at the hardware level, e.g., a stuck bit in a data bus, a bad bit in a memory module, or a faulty internal circuit in a processor. To address these errors, mission-critical SoCs employ lock step processor cores and other redundant computing elements. To handle these elements, NetSpeed uses a compound bridge as shown in the figure, to compare AXI interfaces and confirms that they are lock-step equivalent.



*Figure 7 Compound bridge to address redundant port checking*

#### 4.5.5 NoC Register Parity Checking

NetSpeed IP can be configured to enable parity on all NoC registers. If enabled, parity bits are stored with write (or at reset) and verified by SW on reads. Parity is generated at regbus master and carried through the NoC. The hardware components that use register values, checks for parity whenever they use the value, and if there is a parity mismatch, the operation is modified as appropriate for the circumstance. Apart from this the parity is also checked every cycle. For example: address table parity failure would force a DECERR response that terminates the transaction.

## 4.6 RAM PROTECTION FEATURES

### 4.6.1 Data ECC for RAMs

The coherency directory and last level cache RAMs support ECC single-bit correction and double-bit detection. The number of ECC checkbits is derived from the number of data bits needed. This information is available in the NocStudio generated NoC Reference Manual.

### 4.6.2 Address ECC for RAMs

Apart from the data array, NetSpeed also protects the address decode/lookup functionality of the RAM too allowing failures in that logic to be detected. The goal here is to have the ECC computed not just based on the data but also the array address. This level of protection is vital in safety critical applications to detect potential issues causing incorrect rows to be read from the RAMs.

## **4.7 COHERENCY PROTECTION**

NetSpeed protects all its coherency components, logic and memory included. The various mechanisms are covered in different sections of this document, apply to coherent as well as non-coherent components of the NetSpeed IP.

## **4.8 TIMEOUTS**

There are various configurable options for handling timeouts in NetSpeed IP, using high resolution counters with programmable timestamps. Maskable interrupt is also raised to the CPU with detailed syndrome of the timed-out request

### **4.8.1 Target Timeouts**

To detect unresponsive targets, timeouts track requests outstanding to slave devices at the target side NoC bridges. When responses are not received from the target within timeout intervals, dummy error responses can be optionally auto-generated and sent back to the initiator. This allows recovery of reserved resources in the NoC and the initiator.

### **4.8.2 Initiator Timeouts**

On initiator side bridges, timeouts can be maintained for transactions outstanding on the NoC. These timeouts allow detection of requests potentially dropped or stuck in the NoC. Timeout intervals are individually programmable and share timers for low cost implementation.

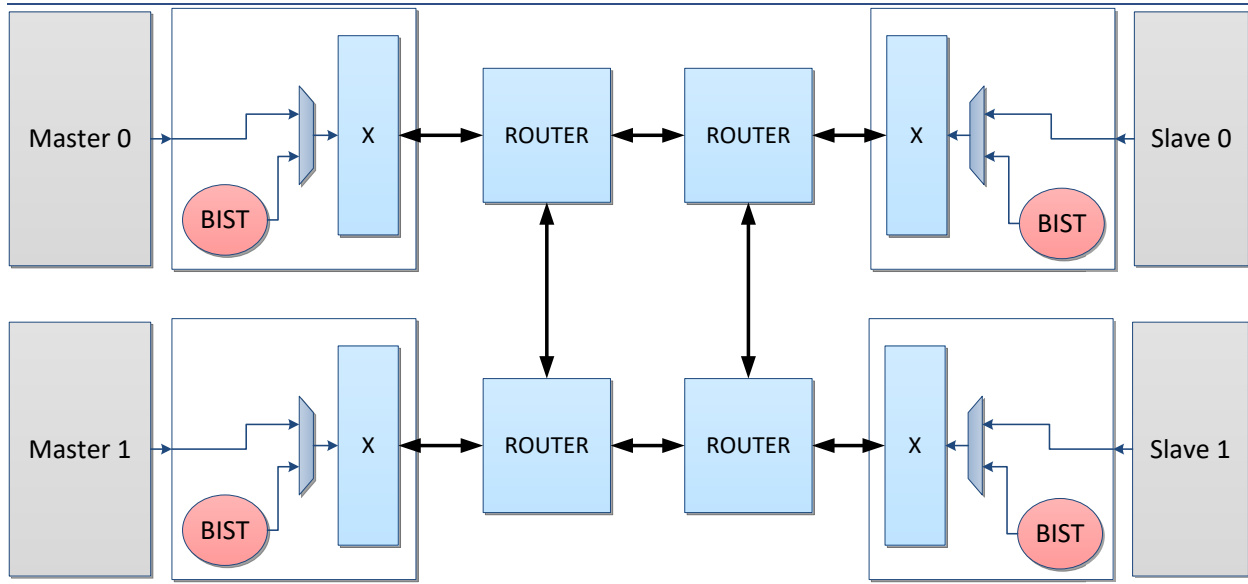
### **4.8.3 NoC Timeouts**

Another layer of timeouts occur based on backpressure from the slave device for requests and master devices for responses from NoC. This can cause backup in the NoC potentially blocking other traffic. Timeout for these events can be configured to start dropping requests or response at the destination and raise fatal interrupts for CPU intervention.

## **4.9 NETWORK BIST ARCHITECTURE**

NetSpeed Network BIST (Built-in Self Test) is an on-chip feature that performs a test of the network to verify that it is functioning correctly. It issues a sequence of packets into the network and verifies that they are received on the far side and that the data isn't corrupted. It is intended to check for stuck-at values in the network links and all network storage structures.





*Figure 8 Logical Diagram of Network BIST*

The Network BIST inserts packets into the network in the bridge logic (both master and slave bridges). The packets are sent through every available path in the network, through every route and every layer.

NocStudio automatically configures the BIST logic based on the constructed network. For each bridge, it creates a sequence of packets to send into the network for each interface, ensuring that every possible path is exercised. The bridges are also programmed on the receiving side to know what packets are expected for BIST, to identify whether all packets have arrived and whether they all have the correct values in the packet.

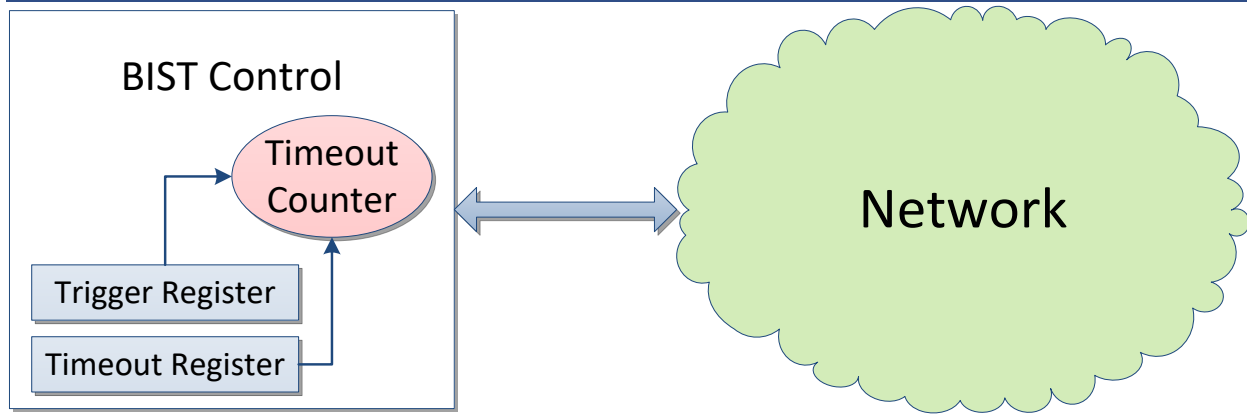
If any packets are lost or corrupted, the network will be able to generate an interrupt to alert the system that a problem has been detected.

If packets are stuck in the network for any reason, the Network BIST will indicate the network is down through the same mechanism.

#### 4.9.1 BIST Control Interface

The BIST control module is a global state-machine that controls the Network BIST.





*Figure 9 BIST Control*

The Network BIST sequence is triggered by a register access to the BIST Control. When triggered, the BIST will signal through a side-band mechanism to all bridges that the BIST sequences should begin. Each bridge will initiate there sequence of packets.

The BIST Control also has a timeout value register. This register is used to indicate how much time the BIST engine is allowed to run before the packets are considered stuck or lost in the network. Ample time should be programmed to ensure no false failures are flagged. When the BIST sequence is started, a timer will be triggered and count until the programmed value is hit. If any packets haven't arrived at the destination, an interrupt will be signaled indicating that a BIST error has occurred.

The BIST error detection logic can be stimulated in hardware by setting the timeout value to a very small value.

#### 4.9.2 BIST Packet Length and Value

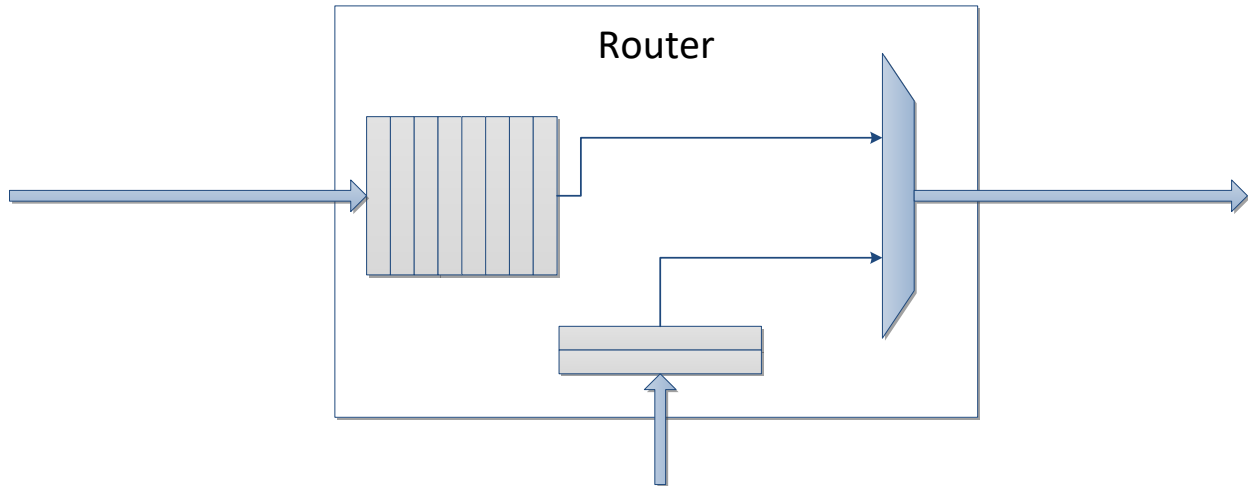
The packets generated by the Network BIST are intended to check that the routes are functional, as well as checking for stuck-at values within the NoC.

To check stuck-at faults, the BIST will send multiple packets through each route in the network. Each packet will carry a different payload value to detect stuck-at values. The packets will send the following patterns repeated for the length of the packet:

*Table 1 Packet Pattern*

Packet Pattern (Repeated)
0000
1111
0101
1010

The length of the packet is determined by the network itself. In order for these patterns to reliably hit each datapath storage element, each packet must be long enough to fill the longest FIFO along its path.



*Figure 10 Router FIFOs*

In the figure above, a router has two input FIFOs going to the east port. The FIFO on the west port has a depth of 8. The FIFO on the south port has a depth of 2. Packet sent along the west->east path must be long enough to fill the entire 8 entry FIFO. This will ensure each FIFO entry will be written with each of the patterns.

Since upsizing and downsizing in the network is possible, the packet length may need to be longer than the largest FIFO depth along the path. If the 8 entry FIFO is 512 bits wide, an agent sending 128 bit beats would have to send packets of 32 beats long.

#### 4.9.3 Concurrent BIST and Normal Packets

The BIST packets are mark with an additional bit to indicate that it is for BIST only. This will allow the end-point to discard the packet instead of sending through to the agent, as well as allowing it to track how many packets have arrived.

Since BIST and Normal packets are distinguishable, they can use the network at the same time. The system does not need to wait for the BIST sequence to complete before executing normal requests into the network. However, the concurrent execution of these packets can adversely affect each other. Normal packets can cause back-pressure in the network, causing the BIST execution time to be longer. The timeout value would need to be adjusted accordingly. Similarly, BIST packets can cause some additional network congestion, increasing the latency of normal packets.

The ability to run BIST concurrently with normal traffic allows the Network BIST to be executed multiple times, instead of just at reset. The BIST could be triggered periodically to test that the network is still functioning correctly.

#### **4.9.4 Power Requirements**

The Network BIST issues packets throughout the entire network. It can only be triggered when all network power domains are powered up, or packets won't be able to make it through the network correctly. Additionally, the BIST does not support fence/drain style of power-management. It won't detect if it is legal to send the packet based on active power domains. This is why the network must be fully powered up during the BIST sequence.

#### **4.9.5 Reset Sequence**

While the BIST engine can be triggered through register accesses, those would require the network to be active in order to access the registers, along with any other accesses needed to get that far in the sequence. It may be desirable to have the BIST engine triggered on System reset, before any of the agents are active, in order to test the network before anyone uses it. This can be configured in NocStudio.

### **4.10 ERROR LOGGING AND FAULT REPORTING**

Interrupt and fault control signals from each NoC element are reported to a centralized fault controller. These signals are asynchronously delivered, locally synchronized and captured into interrupt status and mask registers. A simple register interface, accessible over regbus, is provided to show which element raised an alarm. For additional granularity, multiple types of interrupt signals are also tracked from each element. The detailed information is accessible via regbus access to status stored in the element that sources the alarm. Errors are counted where they happen, and alarms (interrupts) will be raised and delivered to the fault controller. Correctable ECC errors will not halt progress, but will be counted and an appropriate interrupt will be raised.

### **4.11 CONFIGURATION OPTIONS THROUGH NOCSTUDIO**

Different end-user applications, from servers to data center storage to automotive applications, may require different levels of resilience and reliability. Most of the options discussed above are configured through NetSpeed's interconnect synthesis engine - NocStudio. All the resilience options are a tradeoff between robustness and area/power. These tradeoffs can be evaluated and architects can make the appropriate decision using NocStudio.

## 5 EXTERNAL SAFETY MECHANISMS

---

### 5.1 RAM PROTECTION FEATURES

#### 5.1.1 Data ECC for RAMs

The coherency directory and last level cache RAMs support ECC single-bit correction and double-bit detection. The number of ECC checkbits is derived from the number of data bits needed. This information is available in the NocStudio generated NoC Reference Manual.

#### 5.1.2 Address ECC for RAMs

Apart from the data array, NetSpeed also protects the address decode/lookup functionality of the RAM too allowing failures in that logic to be detected. The goal here is to have the ECC computed not just based on the data but also the array address. This level of protection is vital in safety critical applications to detect potential issues causing incorrect rows to be read from the RAMs.

## 6 SAFETY LIFECYCLE

### 6.1 PRODUCT LIFE CYCLE FLOW

NetSpeed's product flow follows the Safety Life Cycle ranging from the specification, to design, implementation, integration, verification, validation, and production release. Each phase of product flow corresponds to the ISO 26262 Safety Life Cycle (SLC) top-down flow. Organizational structure is maintained to support such a process.

### 6.2 SAFETY LIFE CYCLE – SLC

Safety life cycle based on the ISO26262 standard is described below, with 3 main phases. NetSpeed SLC follows the guidelines from this standard.

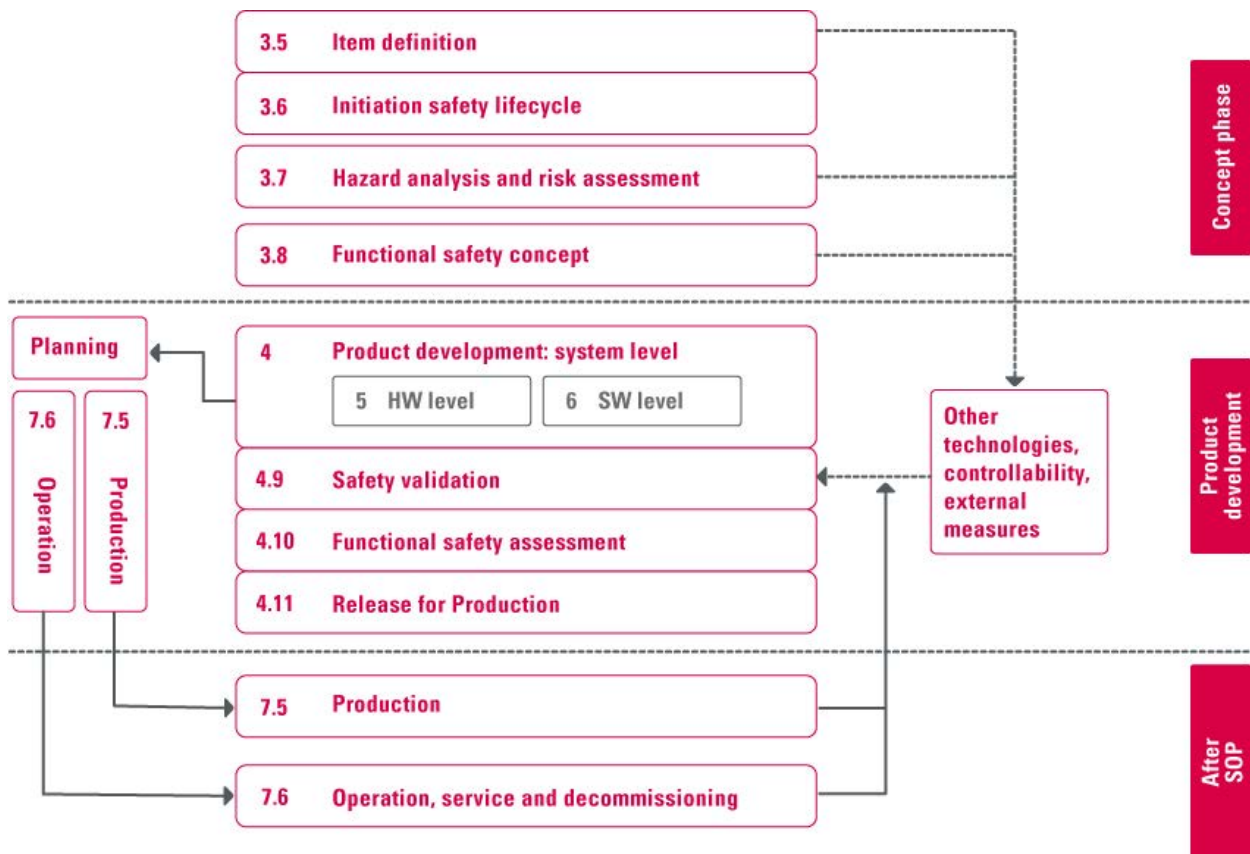


Figure 11 Safety Life Cycle

## 6.2 NETSPEED SLC FLOW

NetSpeed's SLC flow is mapped as described below. There is clear independence and demarcation between the three main phases.

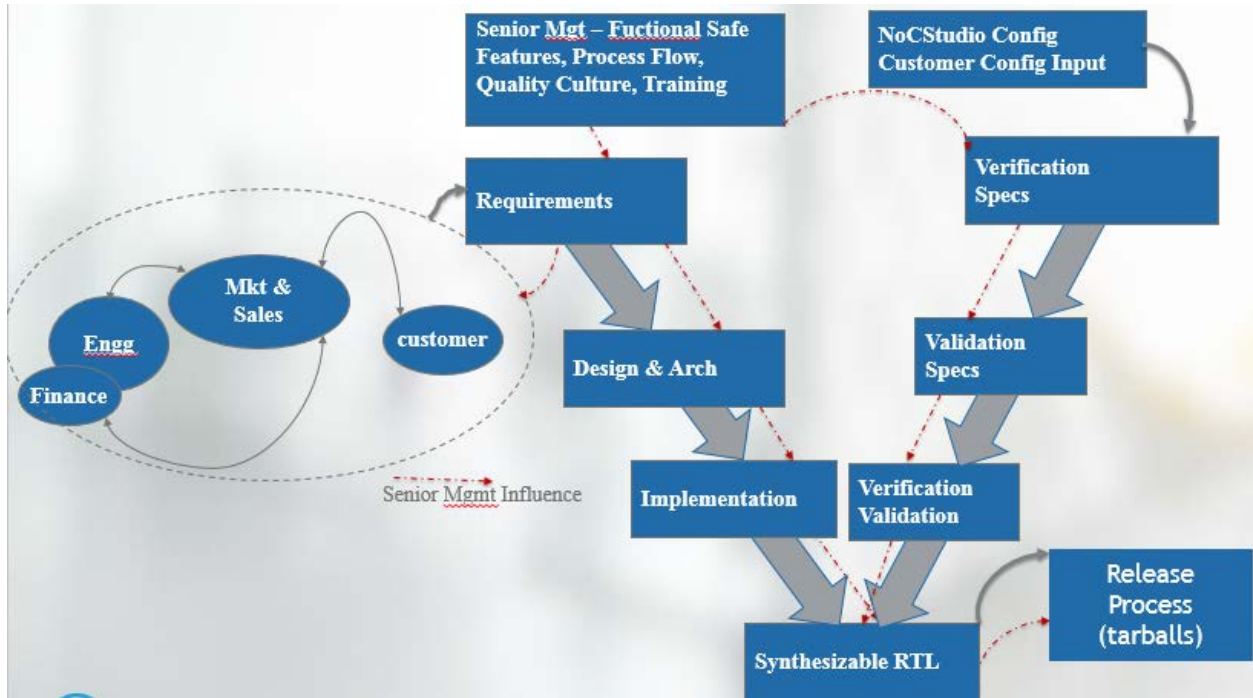


Figure 12 Development flow

## 6.3 CONCEPT

NoC development process starts with Design requirements gathering from customers and internal roadmap requirements. Safety features or reliability or quality requirements are enforced at the requirements analysis phase. These requirements are carried forward for detailed design and implementation. Every phase of development quality is ensured through rigid checklists.

### 6.3.1 Requirements

Every requirement is documented with their unique id and sub-divided if required, but for all cases every requirement can be identified with its unique ID in JIRA.

Here are the Epics filtered for 17.09 release (Open < status < Closed):

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	NS-2492	SystemC - Gemini	reza	reza	↑	TO DO	Unresolved	Jun 13, 2017	Jun 13, 2017	Jul
	NS-2466	Help message improvements	Bhumeshwar Sarswat	John Bainbridge	↑	DEV COMPLETE	Unresolved	Jun 08, 2017	Jun 15, 2017	Jun
	NS-2454	Accurate (um) SoC Floorplan Spec	Nishant Rao	Sailesh Kumar	↑	IN DEV	Unresolved	Jun 06, 2017	Jun 14, 2017	Jun
	NS-2431	Bridge-based clock-crossing improvements	Sailesh Kumar	John Bainbridge	↑	TO DO	Unresolved	Jun 01, 2017	Jun 09, 2017	
	NS-2422	Gemini Agent Modeling	Akshay Ramachandran	Akshay Ramachandran	↑	TO DO	Unresolved	May 31, 2017	May 31, 2017	
	NS-2398	Industry standard host template and library of standard IPs	Amish Shah	Sailesh Kumar	↑	TO DO	Unresolved	May 26, 2017	May 26, 2017	
	NS-2397	Eval winning mode	Sailesh Kumar	Sailesh Kumar	↑	TO DO	Unresolved	May 26, 2017	May 26, 2017	

Figure 13 EPIC 17.09 in JIRA – Requirements Traceability

And these requirements are tracked for respective architecture, micro-architecture being defined, modified, coded, verified, validated and prepared for releases.

### 6.3.2 Safety Requirement tracking

Every functional safe requirement in the database JIRA are part of NS-2350 and are documented with unique ID being assigned as depicted in the picture.

relates to	NS-2358 ECC on RAM index	↑	TO DO
	NS-2359 End to end parity (Netspeed hos...	↑	TO DO
	NS-2368 Parity on registers	↑	TO DO
<b>Sub-tasks</b>			
1. ECC on RAM index		TO DO	Babu Turumella
2. End to end parity (Netspeed host support)		TO DO	Babu Turumella
3. Parity on registers		TO DO	Babu Turumella

Figure 14 Safety Requirements as part of EPIC 17.09 in JIRA



### 6.3.3 Safety Requirement Documentation

Functional safety requirements as NS-2350 are grouped based on the context. NS-2350 FuSa improvement requirement consists of three other sub-requirements as NS-2358 – ECC on RAM index, NS-2359-End-to-End parity and NS-2368 – parity on registers.

Functional Requirements considering all safety aspects are documented in detail and is version controlled in SVN database. All safety, reliability or serviceability features are documented. For example, one of such version of the document (A) is [https://app.assembla.com/spaces/netspeed\\_noc/subversion/source/HEAD/trunk/doc/planning/Planned%20FuSa%20features.xlsx](https://app.assembla.com/spaces/netspeed_noc/subversion/source/HEAD/trunk/doc/planning/Planned%20FuSa%20features.xlsx)

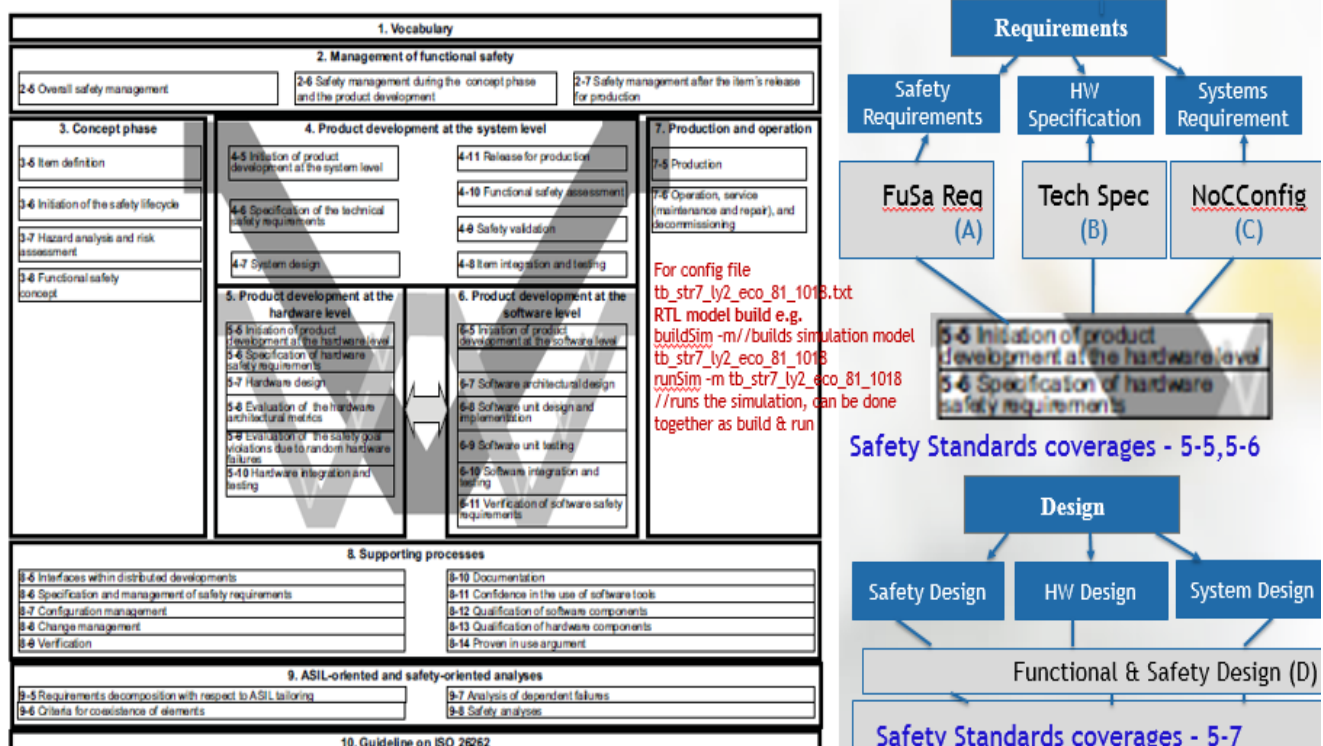


Figure 15 ISO26262-Part2

Detailed technical specification (B) for respective parts of the design or architecture is version controlled in SVN. NocStudio configuration file (C) describes the complete system environment or the complete servicing flow. Such a configuration that creates the customer IP is generated from the NocStudio with customer input. Usage of NocStudio is documented in the *Technical reference user manual* or the help menu of NocStudio.



## 6.4 PRODUCT DEVELOPMENT

Product requirements from the 'Concept Phase' are taken as an input to the 'Development Phase'. In this phase, the requirements are translated into specifications by the Architects. Safety features are an integral part of this architecture.

Example:

[https://app.assembla.com/spaces/netspeed\\_noc/subversion/source/HEAD/doc/ccNoC/Coherency%20Architecture.docx](https://app.assembla.com/spaces/netspeed_noc/subversion/source/HEAD/doc/ccNoC/Coherency%20Architecture.docx)

Test Plan, Test Design, execution, reports with metrics results till all planned items are passed or tested with success and then release process is depicted in the below picture.

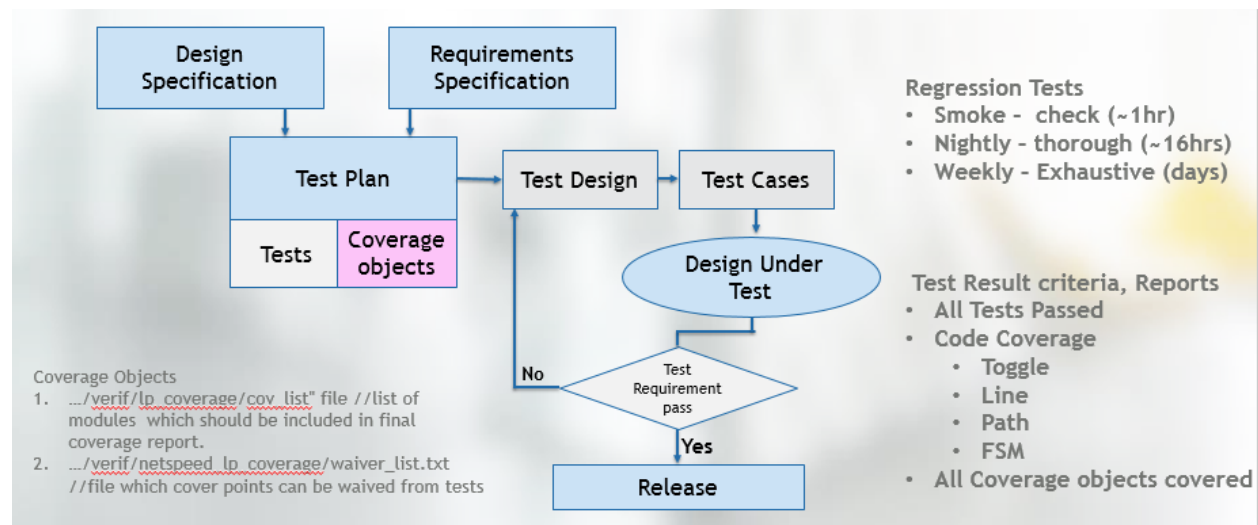


Figure 16 Test Plan and Design Flow

Using these documents as a specification, design team implements the design in Verilog RTL.

Example:

[https://app.assembla.com/spaces/netspeed\\_noc/subversion/source/HEAD/trunk/src/hw/ccs/rtl/\\*.v](https://app.assembla.com/spaces/netspeed_noc/subversion/source/HEAD/trunk/src/hw/ccs/rtl/*.v)

Verification and Design are independent groups with no overlap. Verification team develops a testplan using the architecture specifications.

Example:

[https://app.assembla.com/spaces/netspeed\\_noc/subversion/source/HEAD/trunk/doc/release\\_docs/Xena%20Specific%20Docs/NetSpeed%20Gemini%20Verification%20Plan%20-%20Xena.docx](https://app.assembla.com/spaces/netspeed_noc/subversion/source/HEAD/trunk/doc/release_docs/Xena%20Specific%20Docs/NetSpeed%20Gemini%20Verification%20Plan%20-%20Xena.docx)

Verification tests are developed based on this testplan and run on the design using the simulation tools from Cadence and Synopsys. Automated test methods and systematic regression process is used to verify the design. Details of this process are covered in detail in the *Verification chapter*.

At the completion of the verification cycle, an elaborate release process makes the IP ready for customer shipment. This process is explained in detail in the *Release process chapter*.

## 6.5 POST-RELEASE PHASE

### 6.5.1 Field monitoring

Customer issues go through a well-defined process. Each customer has a point of contact (POC) in the application engineering team. After the initial triage and communication with the customer, the problem will be transferred if it requires a design change. It will go through the development and release process before the application engineer makes the release or a documentation with a workaround available to the customer.

This data is tracked in JIRA based tracking system and used to mining in future. The data is maintained at

<https://netspeedsystems.atlassian.net/issues/<filter # for customer name>>

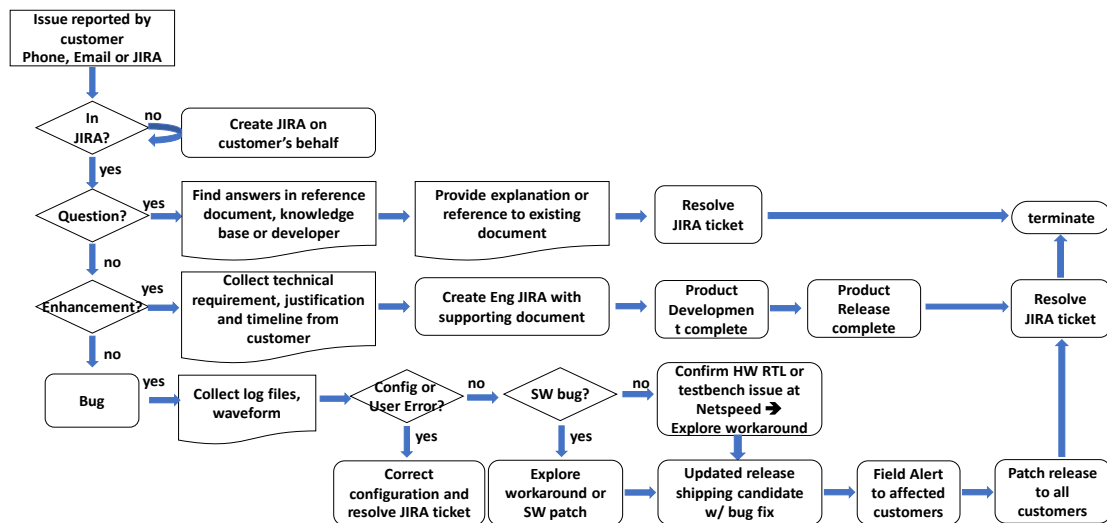


Figure 17 Postsales customer engagement flow

The data from customer JIRA database is mined for analysis periodically and learnings with corrective actions are fed back to the engineering development process to eliminate such issues in future releases. In many cases, tests are augmented to improve coverage to address the problems that escaped to the previous releases, ensuring continuous quality improvement.

Product decommissioning decisions are made by the steering committee based on the input from the marketing and sales teams. Generally, this happens when the product can be replaced by a newer product from NetSpeed. Support for the decommissioned product is continued to the existing customers, but no new feature development will be done on these products.

## 7 DEVELOPMENT PLAN

### 7.1 IP DESIGN AND INTEGRATION

#### 7.1.1 Hardware Design Flow

Design flow starts with Requirement specification as an input and a fully verified product as the output.

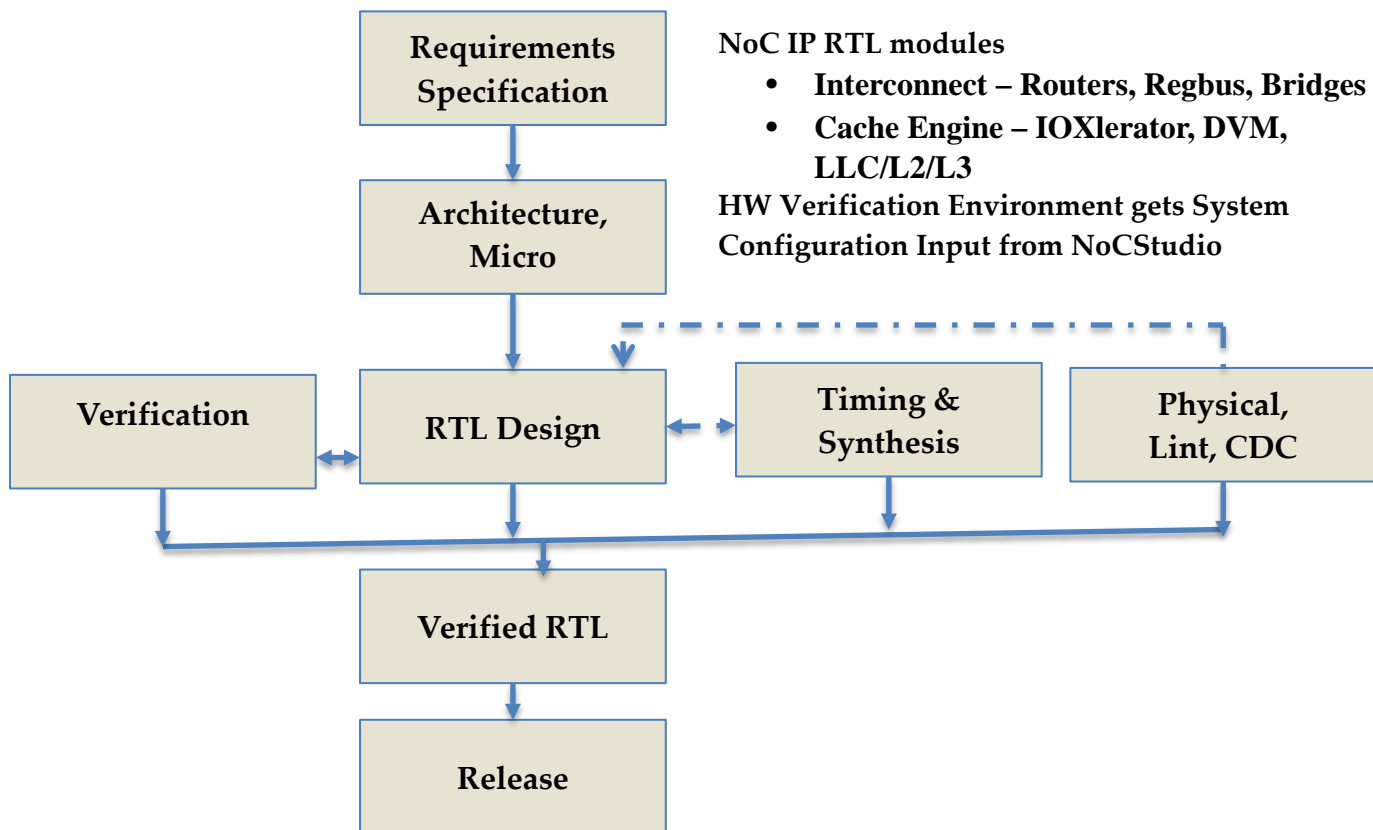
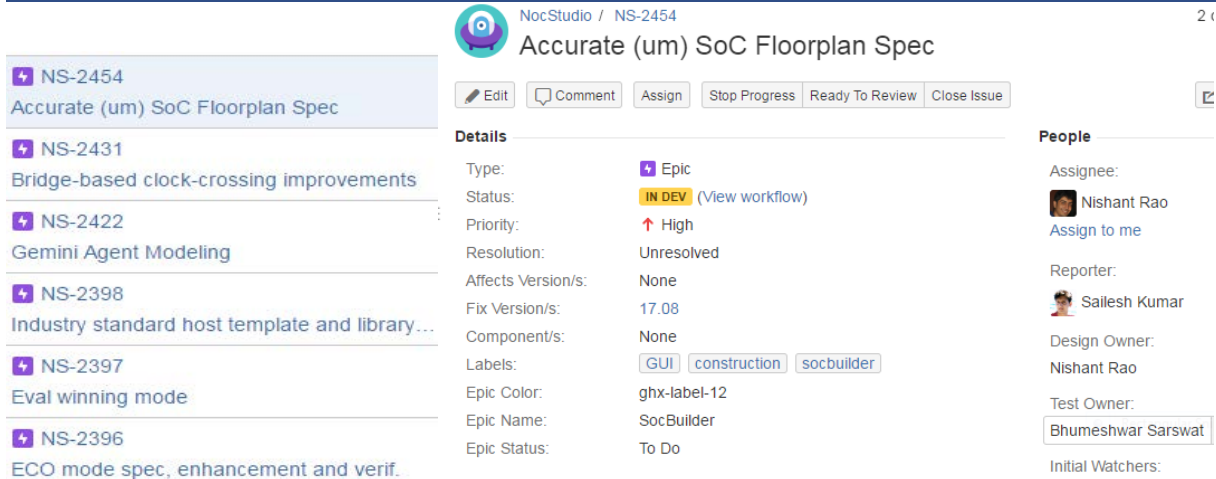


Figure 18 Design Flow

#### 7.1.2 Hardware Design Specification

IP requirements are uniquely numbered and are documented in common repository in JIRA with its respective priority, resolution status to track the implementation, testing, review and closure.

Following example shows list of requirements with its respective unique ID numbers as NS-XXXX.



**NS-2454**  
Accurate (um) SoC Floorplan Spec

**NS-2431**  
Bridge-based clock-crossing improvements

**NS-2422**  
Gemini Agent Modeling

**NS-2398**  
Industry standard host template and library...

**NS-2397**  
Eval winning mode

**NS-2396**  
ECO mode spec, enhancement and verif.

**Details**

Type: Epic  
Status: **IN DEV** (View workflow)  
Priority: **High**  
Resolution: Unresolved  
Affects Version/s: None  
Fix Version/s: 17.08  
Component/s: None  
Labels: GUI, construction, socbuilder  
Epic Color: ghx-label-12  
Epic Name: SocBuilder  
Epic Status: To Do

**People**

Assignee: Nishant Rao  
Assign to me

Reporter: Suresh Kumar

Design Owner: Nishant Rao

Test Owner: Bhmeshwar Sarswat

Initial Watchers:

Figure 19 Requirement List

For further details of NS-2454 there are several fields mentioning priority of the item, status of the specification feature or requirement. Resolution field mentions if the item is closed or still unresolved. All other fields in JIRA such as owner of the item, assigned person for development, testing etc. are used for project management.

### 7.1.3 Design Methodology

Design methodology comprises of RTL coding styles, release procedures and controlled by detailed checklists. These conventions are documented and ensured all design engineers are trained on these procedures.

RTL Coding styles are documented and maintained in Confluence at:

<https://netspeed.atlassian.net/wiki/spaces/EN/pages/1212524/Hardware+Design+Methodology>

Design quality is ensured by using a number of static checks described in:

<https://netspeed.atlassian.net/wiki/spaces/EN/pages/80260465/NS-2309+-+Design+Rule+Check+Lint>

Elaborate checks are imposed prior to releasing the design to verification team for testing. These steps are described in:

<https://netspeed.atlassian.net/wiki/spaces/EN/pages/7602404/HW+Regression+Hierarchy>

### 7.1.4 Design Documentation

All Design details are documented and are part of wiki pages in confluence. Every update/modification to these documents is version controlled.

Example:

Ring Master & Ring Splitter Design Details are part of confluence with diagrams as

<https://netspeed.atlassian.net/wiki/display/EN/Ring+Master+and+Ring+Splitter+Design+Details>

Functional Safety features are part and parcel of these documents, wherever its appropriate. For example – cache coherency engine supports parity design as documented in <https://netspeed.atlassian.net/wiki/display/EN/GACE-58+CCC>.

### **7.1.5 Design Checklist**

Any new feature implementation has to go through a detailed checklist covering the following areas:

- NocStudio (SW)
- Release
- Documentation
- RTL Design
- Synthesis/Backend
- Lint/CDC
- HW Verif
- SW Verif

Detailed checklist is maintained at:

<https://netspeed.atlassian.net/wiki/spaces/EN/pages/73367686/New+Feature+Checklist>

## **7.2 DEVELOPMENT VERIFICATION PLAN**

NetSpeed verification process starts with verification plan made from the Functional specification. It involves developing verification testbench infrastructure, test stimulus to exercise various states of the design and checkers to verify the correctness of the design. Its an iterative process until the verification is complete. Completeness is determined by verification metrics that involves coverage completeness and dropping of bug rate to acceptable level.

## 7.2.1 Verification cycle

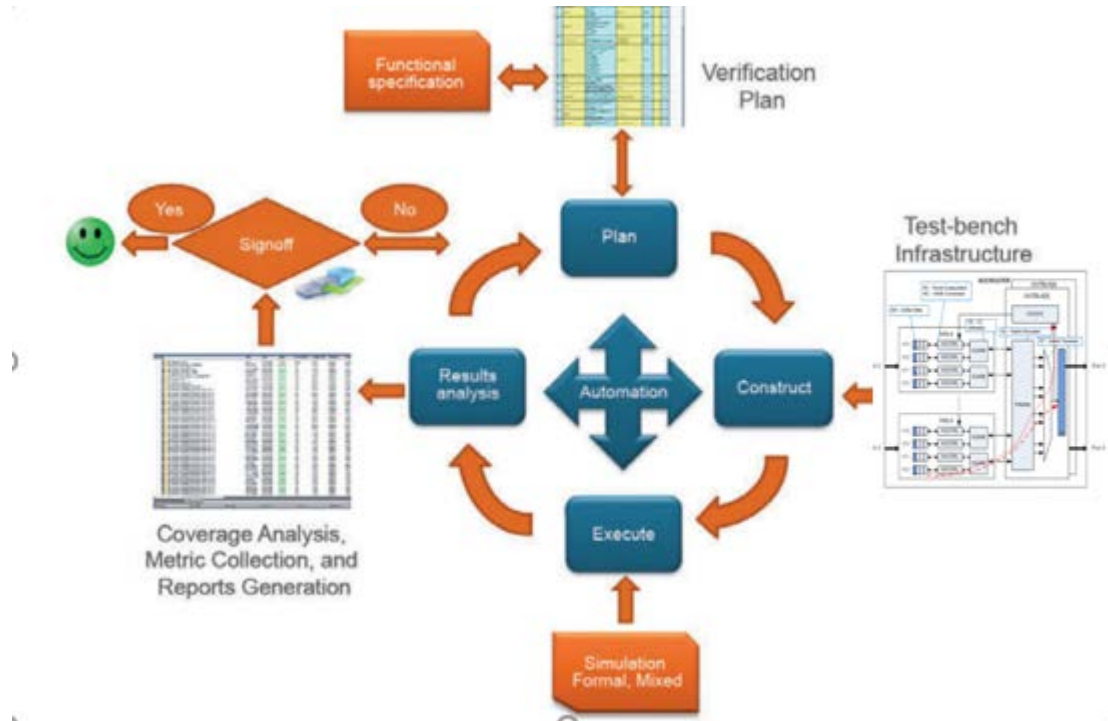


Figure 20 Verification Process

The above figure depicts the verification process flow with verification plan, its test-bench infrastructure for the execution flow resulting in report generations with coverage and other metrics collection. Once all planned tests pass, the module or eventually the NoC IP is ready for signoff.

## 7.2.2 Inputs to Verification Plan

There are three main inputs to a verification plan. They are:

### 7.2.2.1 Functional Requirements

These requirements can be seen in section 9.1 *NOC Functional Requirements*, of this document.

### 7.2.2.2 Design Specification

The specifications are discussed in the section 11 *Hardware Design Specification*, of this document.

### 7.2.2.3 Infrastructure and Methodology

It is available in confluence as *Infrastructure\_And\_Methodology*.

## 7.2.3 TestPlan Template

The template for the testplan is shown as follows:

Function	Feature	Diag	Checker	Coverage	Coverpoint Name	Verif Owner	Status	RTL Done	Test/Cover coded	Test Pass/ Cover hit	CHI Issue A Spec	CHI Issue B Spec	Notes
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....

Testplans are documented and maintained at SVN. An example of such a document is

[https://app.assembla.com/spaces/netspeed\\_noc/subversion/source/HEAD/trunk/doc/internal\\_docs/LP%20Test%20and%20Coverage%20Plan.xlsx](https://app.assembla.com/spaces/netspeed_noc/subversion/source/HEAD/trunk/doc/internal_docs/LP%20Test%20and%20Coverage%20Plan.xlsx)

#### 7.2.4 Levels of testing

Verification is done at a module level for exhaustive coverage and at NoC level for full validation and inter-module interactions.

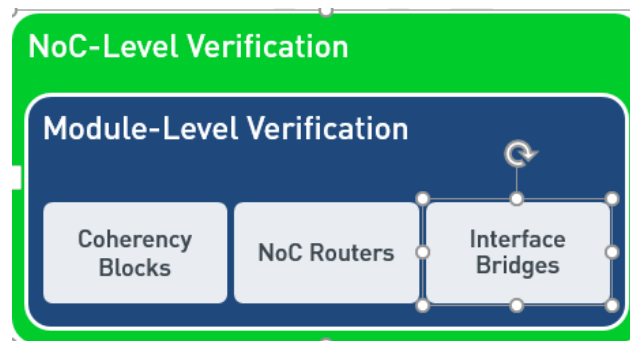


Figure 21 Verification hierarchy

The NoC IP is then validated in a system configuration. Each stage has a detailed testplan based on functional specifications and checklists to ensure completeness.



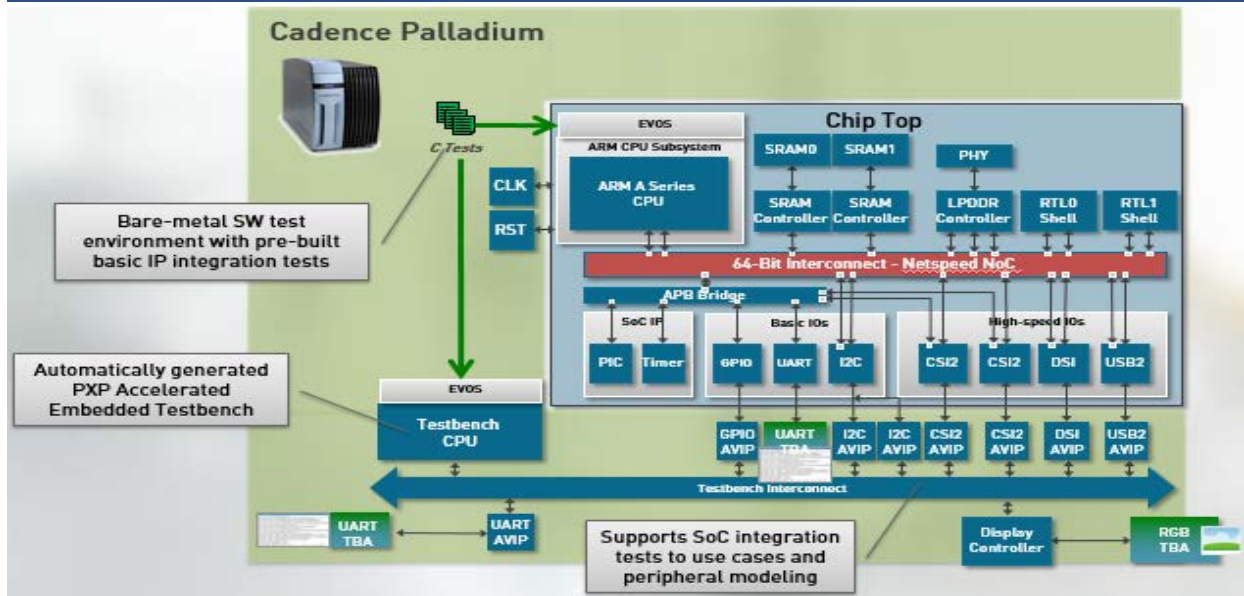


Figure 22 Palladium Emulation

## 7.2.5 Types of tests

Test stimulus comprises of directed and constrained random tests. Directed tests are handcoded to verify one or more targeted conditions, whereas constrained random tests are automatically generated to verify targeted functions as well as unforeseen conditions.

### 7.2.5.1 Constrained random tests

Automated constrained random testing is done using three main items as NoCWeaver or random NoC configurations generator, Configurable Translator or CXT and Nemesis.

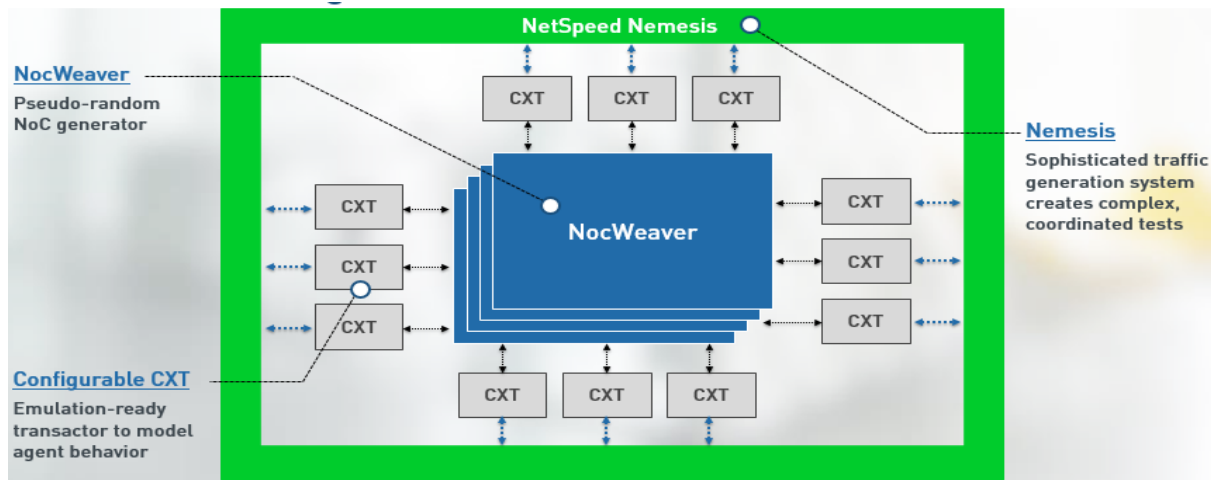


Figure 23 NoCWeaver, CXT, Nemesis

- NoCWeaver - Pseudo-random NoC configuration generator

- Configurable Transactor, CXT - Emulation-ready transactor to model agent behavior
- Nemesis – is Sophisticated traffic generation system which creates complex, coordinated tests covering all possible or allowed sequences in coordinated, randomized and most importantly the cache non-coherent memory accesses are checked for right coherency controls in verification and report results accordingly.

The generator takes inputs from two files, weights and knobs, and user specified NoCs. These files and NoC configuration files accumulated and saved in SVN at, *trunk/src/hw/nemesis/*.

About 1000+ configurations are generated and tested every night. Failures are triaged every day, looking for defects in the design. The completion of automated testing is determined by the coverage measurement. For each feature, cover points are developed that are specified in the test plan. Once these are completely covered, along with code coverage, the testing is considered complete.

#### **7.2.6 Result analysis**

The tests focus on generating right stimules and rely on checkers to ensure the design functions correctly in response to the stimulated condition. Various checkers used in the simulations are:

- Verification Global checker at NoC level
  - Global coherency tracker
  - NoC End-to-End checker
  - Register bus End-to-End checker
- Module checker binds to each instance
  - Bridge checkers
  - Router checkers
  - Coherency component checkers

Test stimulus combined with these checkers ensure verification quality is maintained. Functional safety feature verification is integral to this process and follows similar methodology.

#### **7.2.7 Safety, Reliability Verification**

NoC IP is verified with unrecoverable errors throughout the complete stack, from the underlying hardware to the application software. Such solutions involve three components: Functional safety, serviceability and security.

Each functional safe feature designed or implemented gets verified step-by-step as the design evolves in systematic way. Verification is an iterative process, where feedback from each stage is fed to the next stage to make it robust.

### 7.2.8 Coverage metrics

Code coverage of the RTL design as well as functional coverage of the pre-defined cover points ensures the verification tests exercised and checked complete design.

Code coverage measures four types a) Block b) Expression c) Toggle and d) FSMs. Tools are developed to automate the measurement process.

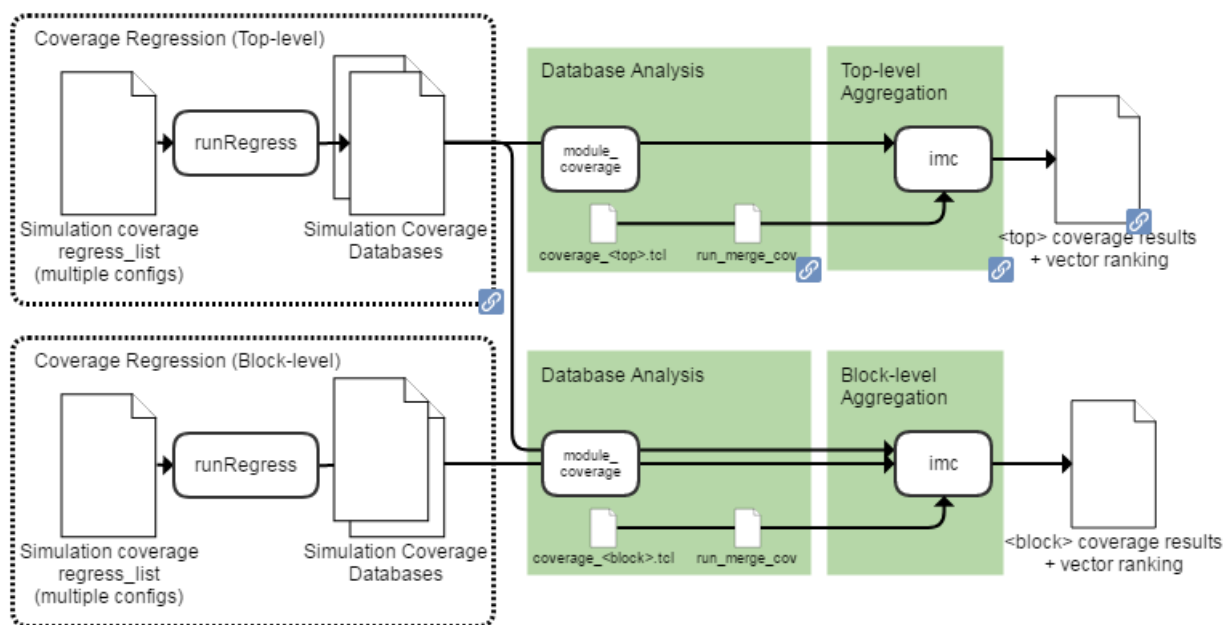


Figure 24 Coverage

For verification signoff, the coverage must be 100% or uncovered items are waived after a review.

## 7.1 REGRESSION TESTING

Continuous regressions are run of all passing tests to ensure the design stability is maintained. There are three types of regressions:

1. Smoke – Quick check – takes about an hour
2. Nightly – Thorough check is done which takes about 16 hours
3. Weekly – This is exhaustive regression tests and often takes several days for completion.

Regression testing is fully automated and results are triaged by tools and notified to the engineers.

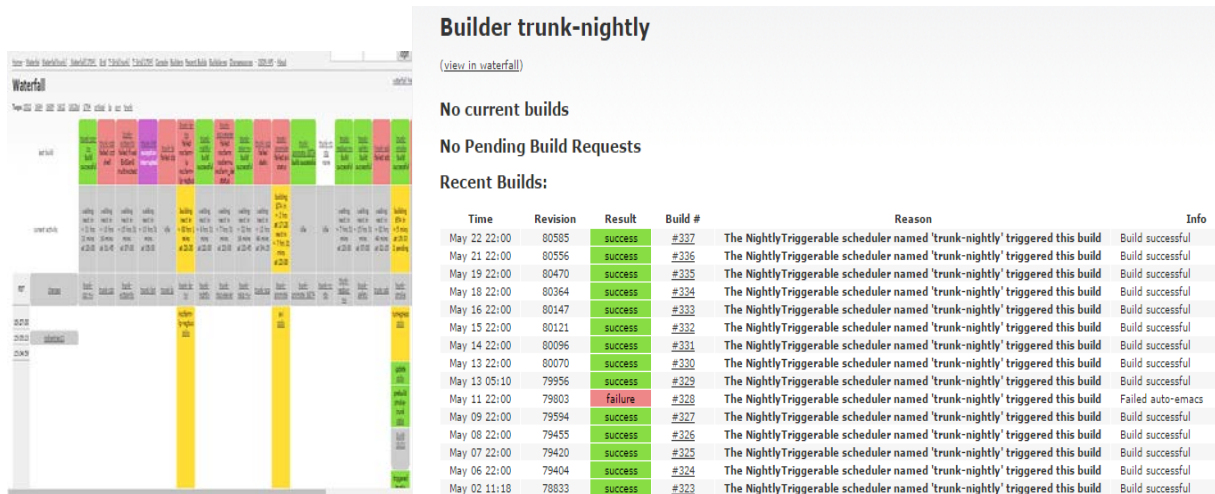


Figure 25 Build schedule

All results of the regressions are saved in SVN database as Buildbot file with summary of execution results can be found at <http://172.29.106.32:8010/waterfall?tag=trunk>

### 7.1.0 Bug tracking - JIRA

All the design defects/Bugs are tracking in JIRA bug tracking system. The flow that any bug transitions from the time it Opened to Closed is shown below:

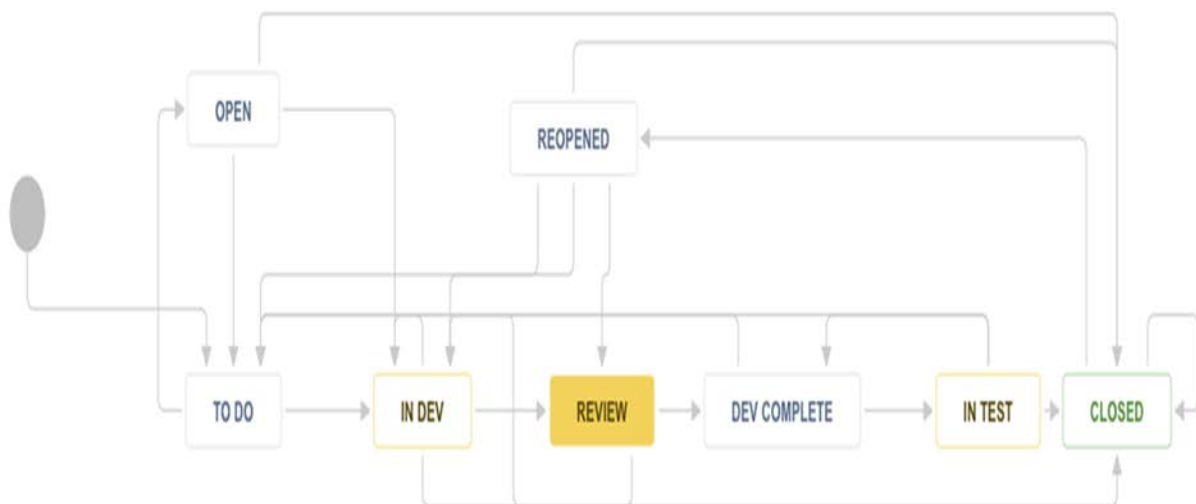


Figure 26 Bug Tracking

The bugs that is open goes into development followed by testing. There are review phases both during the development and testing phases. Review defects are logged into JIRA with responses of iterative phases of development and testing or reviewed for improvement or additional items to be open or implemented, tested and reviewed with no defects and hence closed. Details of the flow is captured in the picture.

Every update to the JIRA entry is broadcast as an email to all the concerned people. This enables the communication across the organization working on that issue. JIRA allows report generation that's used widely to track the progress on the issues and project decisions.

### 7.1.1 Checklist

Completion of verification is checked by the verification engineers using a checklist. Checklists are maintained on Confluence. The checks include:

*Table 2 Checklist*

Testplan
VCS Compatibility
RTL Param Coverage
NS Prop Coverage
Generate block coverage
Code Coverage
Functional Coverage
External sanity
External UVM
RTL performance verification
End to End checkers
Other Assertions/Checkers
Randomized config generation and TB generation (usually this means genStreamX and NocWeaver)
LP
Negative Testing

xprop
Important configs added to smoke/promotion
Regbus / Interrupt
Gate level sim for important customer configs
Run regression on important customer configs

An example of such a document is

<https://netspeed.atlassian.net/wiki/spaces/EN/pages/73367686/New+Feature+Checklist>

## 8 SAFETY ANALYSIS RESULT

---

### 8.1 FMEDA

Documented in a separate file.

### 8.2 DFA

Documented in a separate file.

## 9 TERMINOLOGY

Term	Definition
<b>Agent</b>	Modules in NoC that handle specific functions such as coherency
<b>ASIL</b>	Automotive Safety Integrity Level
<b>Coherency</b>	Rules to ensure the data sharing and ordering rule integrity is ensured
<b>CPU</b>	Central Processing Unit
<b>ECC</b>	Error correction code
<b>EDA Tools</b>	Tools used in the semiconductor design flow
<b>FMEDA</b>	Failure modes, effects, and diagnostic analysis
<b>IP</b>	Intellectual property: RTL design and related collateral
<b>IPXACT</b>	Industry standard format used to express interfaces between modules and other information needed to integrate IP modules
<b>ISO</b>	International Standard Organization
<b>LLC Last level cache</b>	Cache subsystem that connect to the memory
<b>Master Bridge</b>	Connection of the master modules in a SoC to NetSpeed NoC
<b>NoC</b>	Network on a Chip
<b>NocStudio</b>	Software tool infrastructure for NetSpeed IP
<b>QA</b>	Quality assurance
<b>Router</b>	Modules in NoC that handle decisions on the path to transfer packets in the NoC
<b>Slave Bridge</b>	Connection of the slave modules in a SoC to NetSpeed NoC
<b>SoC</b>	System on a Chip
<b>V &amp; V</b>	Verification and Validation
<b>Validation</b>	Testing to ensure Implementation works in a higher level usecase
<b>Verification</b>	Testing to ensure implementation follows Specification
<b>Virtual Circuit</b>	Logical path between two end points in the NoC



2870 Zanker Road, Suite 210  
San Jose, CA 95134  
(408) 914-6962  
<http://www.netspeedsystems.com>