

NetSpeed IP Safety Manual

NetSpeed Orion, Gemini, and Pegasus IP

Version: 1.3

February 13, 2018

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 NoC 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.

For general information about NetSpeed products refer to: www.netspeedsystems.com.

Revision History

Rev.	Date	Updates
1.0	January 24, 2017	Initial
1.1	January 4, 2018	Made updates throughout the document primarily for grammar, style, and consistency in text and graphics.
1.2	January 23, 2018	Updated the “Assumption of Use” section. Moved the section on Network BIST architecture to the “External Safety Mechanisms” chapter. Updated the following sections: <ul style="list-style-type: none">• “Safety Verification Codes”• “Fault/Error Verification” Added “Safety Analysis Results” chapter.
1.3	February 13, 2018	Made updates throughout the document primarily for grammar, style, and consistency in text and graphics. Updated terminology and restructured content in the “Internal Safety Mechanisms” chapter to match FMEDA terminology. Added the “Status of Traffic to Power Gated Blocks” section. Updated content in the “External Safety Mechanisms” chapter.

Contents

About This Document	2
Audience	2
Prerequisite	2
Related Documents.....	2
Customer Support.....	2
Revision History.....	3
Contents	4
Figures	8
Tables.....	9
1 Introduction	10
2 Assumed Safety Requirements	11
2.1 Assumption of Use	11
2.2 NetSpeed IP Network on a Chip	12
2.3 NoC IP in System Applications	12
3 Functional Safety Approach	14
3.1 Top-Down Approach.....	14
3.2 Functional Safety—System Level.....	14
3.2.1 Safety Goal and Safety Motivation.....	14
3.3 Functional Safety—Component Level.....	15
3.4 NocStudio to Create NoC IP	16
3.4.1 Specification Phase	17
3.4.2 Customization Phase.....	17
3.4.3 Generation Phase.....	18
4 Internal Safety Mechanisms.....	19
4.1 NetSpeed Architecture Overview	19
4.2 Summary of Functional Safety, Reliability, and Availability.....	19
4.3 Error Detection and Correction Features	20

4.3.1	ECC Algorithm	20
4.4	NoC-Transport Error Detection and Correction	21
4.4.1	End-to-End Transport Integrity	21
4.4.2	End-to-End User ECC	22
4.4.3	Interface Protection	22
4.4.4	Hop-to-Hop Protection	22
4.4.5	End-to-End Packet Integrity	23
4.5	Logic Protection and Redundancy	23
4.5.1	Flop Structure Parity Protection	23
4.5.2	Bridge Duplication	24
4.5.3	Route Duplication	24
4.5.4	Architectural Support for Redundancy	25
4.5.5	NoC Register Parity	26
4.6	Memory ECC	26
4.6.1	Data ECC for RAMs	26
4.6.2	Address ECC for RAMs	26
4.7	Coherency Protection	27
4.8	Time-Outs	27
4.8.1	Target Time-Outs	27
4.8.2	Initiator Time-Outs	27
4.8.3	NoC Time-Outs	27
4.9	Status of Traffic to Power Gated Blocks	27
4.10	Error Logging and Fault Reporting	27
4.11	Configuration Options Through NocStudio	28
5	External Safety Mechanisms	29
5.1	Network BIST Architecture	29
5.1.1	BIST Control Interface	30
5.1.2	BIST Packet Length and Value	30

5.1.3	Concurrent BIST and Normal Packets.....	31
5.1.4	Power Requirements.....	32
5.1.5	Reset Sequence.....	32
5.2	High Reliability RAM.....	32
5.3	Host Protocol Violations Checker.....	32
5.4	Power-On Tests.....	32
6	Safety Life Cycle	34
6.1	Product Life Cycle Flow	34
6.2	Safety Life Cycle – SLC.....	34
6.2	NetSpeed SLC Flow	35
6.3	Concept.....	35
6.3.1	Requirements	35
6.3.2	Tracking Safety Requirements.....	36
6.3.3	Safety Requirement Documentation.....	37
6.4	Product Development.....	38
6.5	Post-Release Phase	40
6.5.1	Field Monitoring.....	40
7	Development Plan.....	42
7.1	IP Design and Integration	42
7.1.1	Hardware Design Flow.....	42
7.1.2	Hardware Design Specification.....	42
7.1.3	Design Methodology.....	43
7.1.4	Design Documentation	43
7.1.5	Design Checklist	44
7.2	Development Verification Plan	44
7.2.1	Verification Cycle	44
7.2.2	Inputs to Verification Plan	45
7.2.3	Test Plan Template	46

7.2.4	Levels of Testing	46
7.2.5	Types of Tests	47
7.2.6	Result Analysis.....	49
7.2.7	Safety and Reliability Verification.....	49
7.2.8	Coverage Metrics.....	49
7.3	Regression Testing.....	50
7.3.1	Bug Tracking—JIRA.....	51
7.3.2	Checklist.....	52
8	Safety Verification.....	53
8.1	Safety Verification Codes	53
8.2	Fault/Error Verification.....	53
8.2.1	Transport Verification.....	53
8.2.2	CSR Parity Injection Sweep with Interrupt Count.....	54
8.2.3	Interface Parity	54
9	Safety Analysis Results.....	55
9.1	Coverage Assumptions.....	55
9.1.1	ECC.....	55
9.1.2	Parity	55
9.2	Fault Injection.....	56
9.3	FMEDA	56
9.4	DFA.....	56
10	Terminology.....	57

Figures

Figure 1 NetSpeed Network IP (NetSpeed Components) NoC	12
Figure 2 NoC Applications Systems.....	13
Figure 3 Appropriate NoC with Functional Specs through NocStudio NCF	17
Figure 4 NetSpeed NoC Architecture	19
Figure 5 NetSpeed Safety and Resilience Feature Support.....	20
Figure 6 Route Duplication.....	25
Figure 7 Compound bridge to Address Redundant Port Checking.....	26
Figure 8 Network BIST	29
Figure 9 BIST Control	30
Figure 10 Router FIFOs	31
Figure 11 Safety Life Cycle	34
Figure 12 NetSpeed Safety Life Cycle	35
Figure 13 EPIC 17.09 in JIRA – Requirements Traceability.....	36
Figure 14 Safety Requirements as Part of EPIC 17.09 in JIRA	37
Figure 15 ISO 26262-Part2.....	38
Figure 16 Test Plan and Design Flow.....	39
Figure 17 Post-Sales Customer Engagement Flow	40
Figure 18 Design Flow.....	42
Figure 19 Requirement List	43
Figure 20 Verification Process	45
Figure 21 Verification Hierarchy	46
Figure 22 Palladium Emulation	47
Figure 23 NoCWeaver, CXT, Nemesis.....	48
Figure 24 Coverage	50
Figure 25 Build Schedule	51
Figure 26 Bug Tracking	51

Tables

Table 1 Packet Pattern	31
Table 2 Test Plan Template	46
Table 3 Verification Codes	53
Table 4 Diagnostic Coverage for Parity Detection—By Code	55
Table 5 Diagnostic Coverage for Parity Detection—By Error Type	55
Table 6 Terms and Definitions	57

1 INTRODUCTION

This document is a safety manual for NetSpeed IP products. It provides an overview of the safety measures and policies required for developing safety-critical automotive designs. The manual covers safety processes, flows, and requirements such as descriptions, plans, and reports that help design teams attain in a timely manner ISO 26262 certification for automotive designs that have NetSpeed IP embedded within.

ISO 26262 was defined in 2011 by the International Organization for Standardization (ISO) and serves as an international standard for ensuring functional safety of electrical and/or electronic systems in road vehicles.

NetSpeed's safety flow maps network on a chip (NoC) development phases to various parts of the ISO 26262 requirements. The safety flow also supports tailoring of activities to be performed during the life cycle phases.

2 ASSUMED SAFETY REQUIREMENTS

2.1 ASSUMPTION OF USE

The NetSpeed IP is a configurable interconnect IP that supports various features for interfaces, protocols, performance, quality of service (QoS), and coherency, among other things. Customers can configure the IP according to their system requirements. This is also true from a functional safety and ISO 26262 perspective. NetSpeed does not have further constraints or assumptions on configurations and use. Because the NetSpeed IP is to be integrated into a system on a chip (SoC), the integrator is responsible for defining system-safe states and appropriate fault-tolerant time intervals (FTTIs) based on system requirements.

Customers using NetSpeed IP have the flexibility to configure the design to achieve the desired ASIL level.

- ASIL-B & C
 - Users are able to use a partial list of the supported features in NetSpeed IP to achieve ASIL-B & C.
 - If Agent modules are present, the SRAMs that are part of these modules are expected to have a fault coverage of 99% or higher.
- ASIL-D
 - All IP supported functional safety features are required to be enabled.
 - External diagnostic coverage may be required to supplement coverage required for ASIL-D.
 - Agent modules are present, the SRAMs that are part of these modules are expected to have a fault coverage of 99.5% or higher.

2.2 NETSPEED IP NETWORK ON A CHIP

Figure 1 is a top-level view of a NetSpeed NoC.

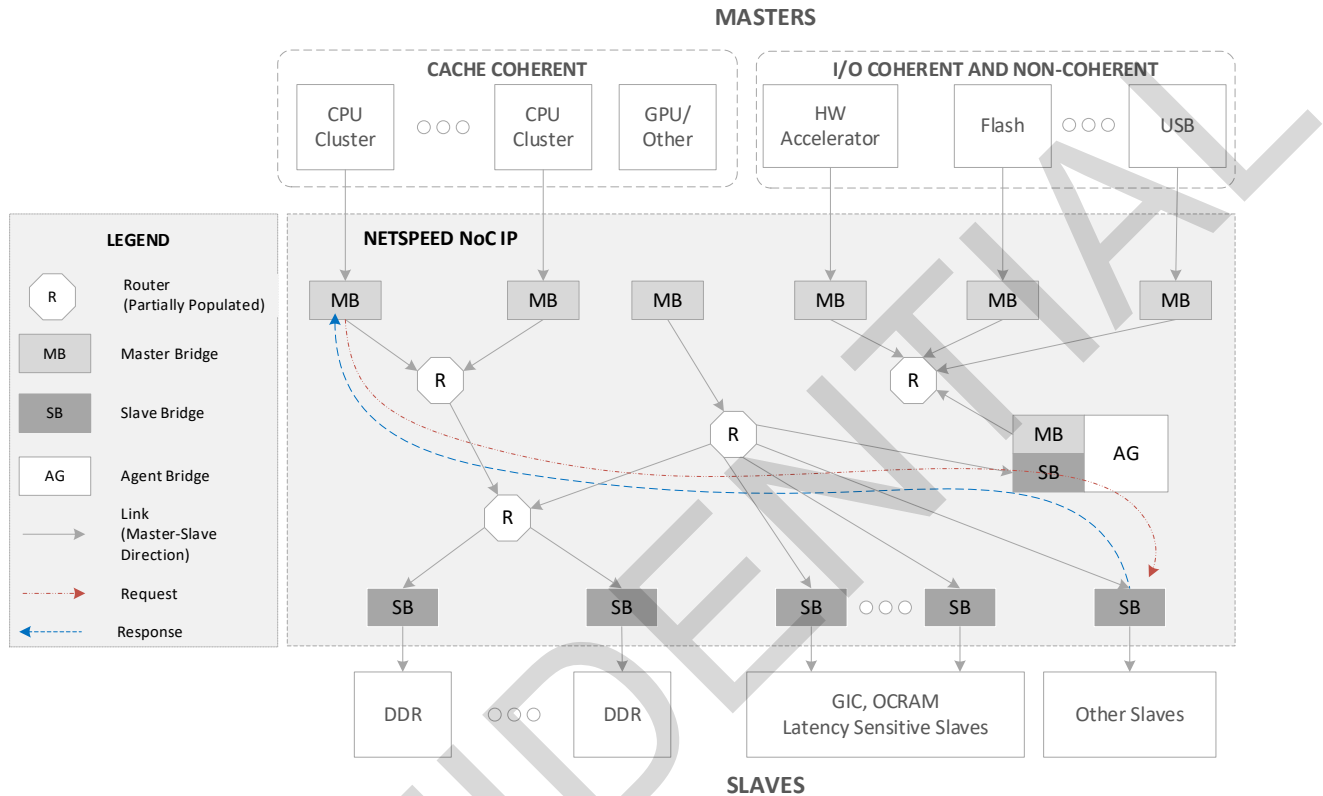


Figure 1 NetSpeed Network IP (NetSpeed Components) NoC

A NoC—like other hierarchical components such as the CPU, GPU and DDR—is a component in an SoC. Along with failure mode effects and diagnostic analysis (FMEDA) and a safety analysis document safety manual, all details required to qualify a NoC to a required automotive safety integrity level (ASIL) level are documented.

2.3 NoC IP IN SYSTEM APPLICATIONS

The NoC IP is incorporated in system applications such as automotive advanced driving assistance systems (ADASs), augmented reality (AR)/virtual reality (VR), and Industrial Internet of Things (IIoT). Its building blocks include bridges, routers, links, and agents with caches.

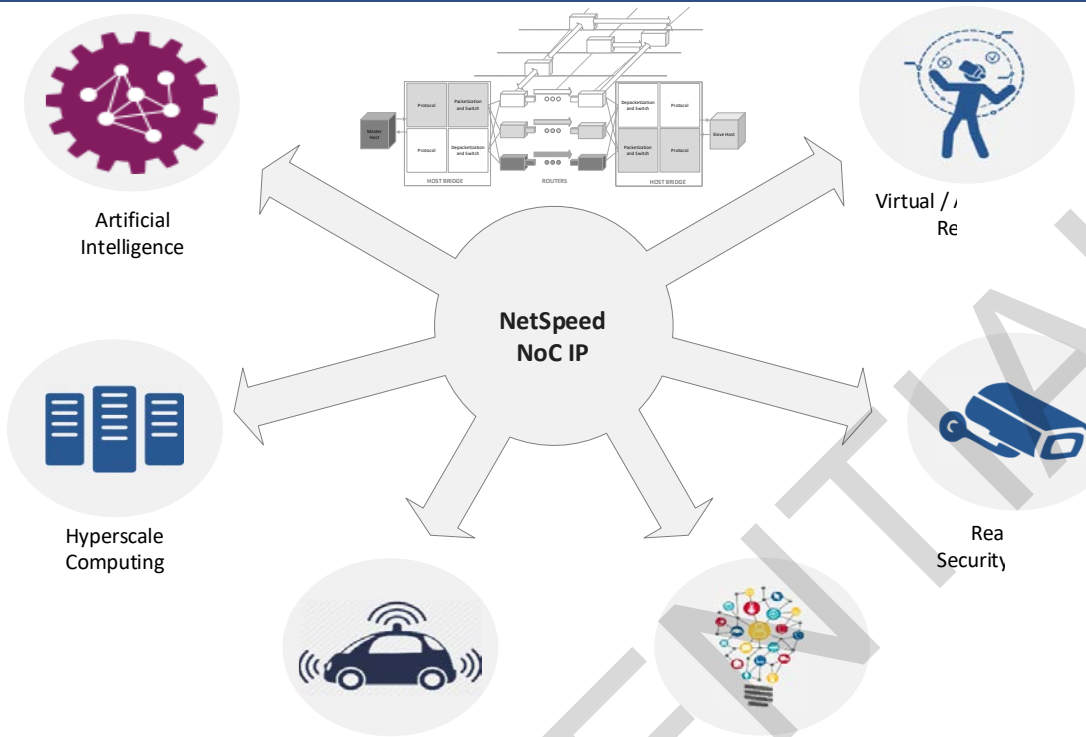


Figure 2 NoC Applications Systems

Bridges connect a master or a slave block to the network of routers and links through a well-defined protocol. The bridges packetize the host's transactions into NetSpeed's packet format during injection into the network and depacketizes 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 as many as four additional links to connect to up to four hosts (H, I, J, and 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

The NetSpeed IP follows a top-down approach, guided by the ISO 26262 safety life cycle. Functional safety base requirements are root requirements. Users specify the base requirements as input in the form of a configuration file to the software that generates the IP—in this case, NocStudio. A NoC with desired system functionality and safety measures are part of this configuration file—in this case again, the NocStudio configuration file that defines functional requirements. The configuration file is validated at the architectural level prior to generating the IP for validation. The generated IP is then verified for correctness prior to release.

3.2 FUNCTIONAL SAFETY—SYSTEM LEVEL

NetSpeed provides the NoC IP as a product with a functionally safe architecture based on customer or end-user use-case requirements for a final SoC. Refer to [AN_505_Safety_Serviceability_Security_Features.pdf](#) for a summary of safety features in all possible NoC designs or configurations.

Safety is not an add-on feature. In other words, safety cannot be added at the end of the product-development process as an afterthought. Safety-feature design is an integral part of the design flow. The product-development phase begins with functionally safe requirements, from concept to requirements, from design to implementation. Safety requirements are handled in the same process as non-safety features and they proceed through development and verification procedures.

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

3.2.1 Safety Goal and 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. Unfortunately, these approaches make it possible for an unrecoverable hardware error to bring down an entire SoC and the applications running on it. This is unacceptable for mission-critical applications such as ADAS and cloud computing—applications where the NetSpeed NoC IP is targeted for use. The SoCs used in these applications must be able to handle unrecoverable hardware errors while continuing to deliver uninterrupted application and transaction services to end-users.

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

3.2.1.2 *Coherency Issue*

Safety-analysis results related 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. Design changes regularly result from new requirements, features, and improvements, and they are tracked with regular automated processes such that all degradations are controlled until the closure of each change.

3.2.1.3 *Plausibility Issue*

Functional safety 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 the ASIL level.

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

3.2.1.4 *Accuracy Issue*

ASIL/Safety phases are assessed in NetSpeed processes without over-engineering, and they are targeted to ensure that 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 the system level; it is also imperative at the component level. Integrated safety devices for NoC customers get detailed directions through parity and/or the error correction code (ECC) to be added selectively as a part of configuration files. They are also described in the technical reference manual.

The NetSpeed NoC IP is expressed as a single component of 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 a link safety or register-level safety, as in ISO 26262: Safety Element out of Context (SEooC).

The basic approach is to assume a system context (or several) for the component. NocStudio helps customers define NoC IP details. The Safety Application Guide—Safety Elements in the NoC Reference Manual—specifies how the safety inside or at the interface of the NoC is applied

correctly in the assumed NoC system context. The NoC Reference Manual is one of the output deliverables from NocStudio (autogenerated by the customer in one or more iterations).

Through bridges and routers that use designated protocols for transportation, the NoC can support data transactions from agents or different types of hosts, such as CPUs, GPUs, and DSPs, as well as peripherals, such as memory controllers and SATA or PCIe devices.

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

NoC functional requirements cover all possible data transportation across components specified in the NocStudio configuration. To ensure safety during packet transportation, various safety features are adopted from the concepts and requirements phases of the development process.

To ensure functional and safe transportation of packets across the NoC fabric, various error detection and correction 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 NocStudio or the configuration file are used for ECC computation based on a user-specified maximum granularity. ECC generation at the transmitting end as well as detection and correction at the receiving end add a cycle each to overall path latency, thus adding PPA with safety feature.

NocStudio provides flexibility to the end-user in selecting safety features. As an alternative to ECC, users can configure parity to be transported with the data. Parity-based protection does not add latency to the path.

3.4 NOCSTUDIO TO CREATE NOC IP

NetSpeed NocStudio is a Network-on-Chip (NoC) architecture platform for ASIC designers and SOC architects. Figure 4 illustrates how the platform incorporates a design flow that involves the following distinct phases, or processes:

1. Specification phase
2. Customization phase

3. Generation phase

This section describes each of these phases in further detail.

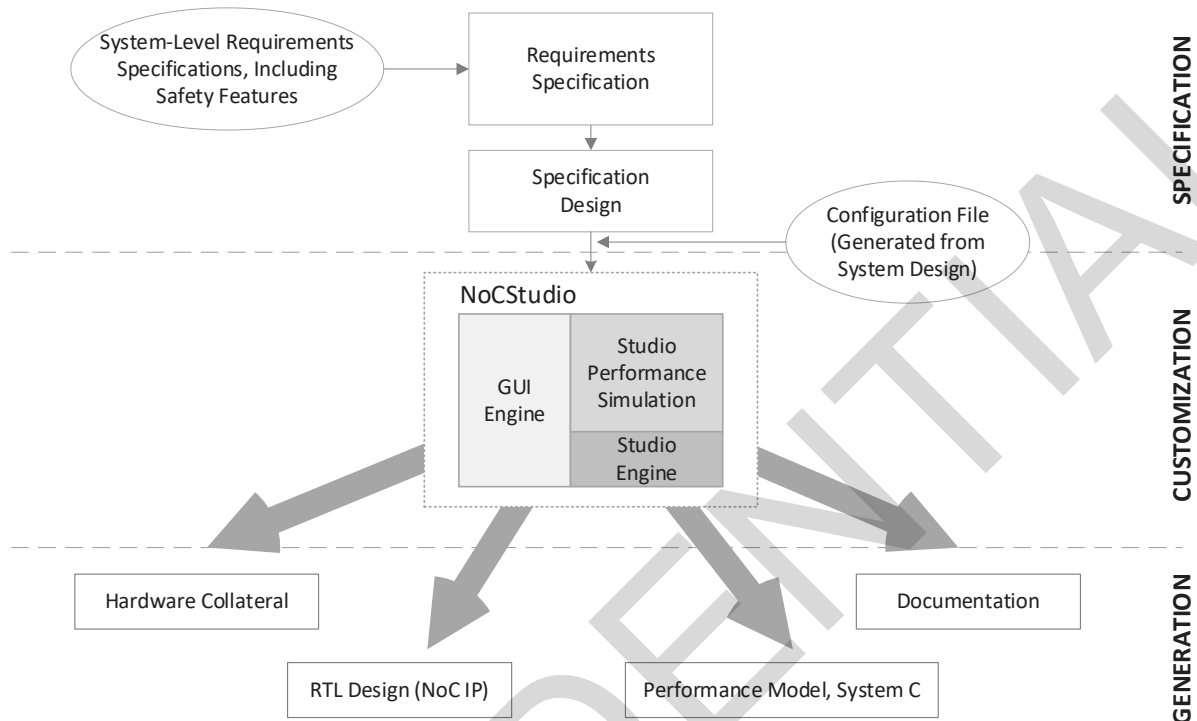


Figure 3 Appropriate NoC with Functional Specs through NocStudio NCF

3.4.1 Specification Phase

During the Specification phase, NoCStudio helps users to specify the following:

- IP hosts, locations, and respective interface protocols
- Address map, connectivity, and dependencies
- Bandwidth and real-time requirements, or QoS, for traffic

3.4.2 Customization Phase

The Customization phase follows the specification phase. In the customization phase, NoCStudio does the following:

- Adds the link and buffer cost
- Adds the routing channels, considering the physical blockages
- Calculates power, performance, and area PPA.

Customers can iterate this step by changing the location of the items. They can also use different traffic requirements, msg_rate/beat rates, or transaction rates to meet target PPAs with respective SoC requirements.

3.4.3 Generation Phase

After all requirements/specifications are determined and the Customization phase is complete, NoCStudio enters the Generation phase. In this phase, NoCStudio generates the following:

- Hardware Collateral
 - IPXACT XML Format—Design Local Information
 - DEF Physical Systems Information
 - SDC Timing Information
- RTL Design—This NoC IP is synthesizable RTL
- Performance Model System C—System simulation
- Documentation—HTML-based NoCStudio User Guide

4 INTERNAL SAFETY MECHANISMS

4.1 NETSPEED ARCHITECTURE OVERVIEW

Figure 4 presents a 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, 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.

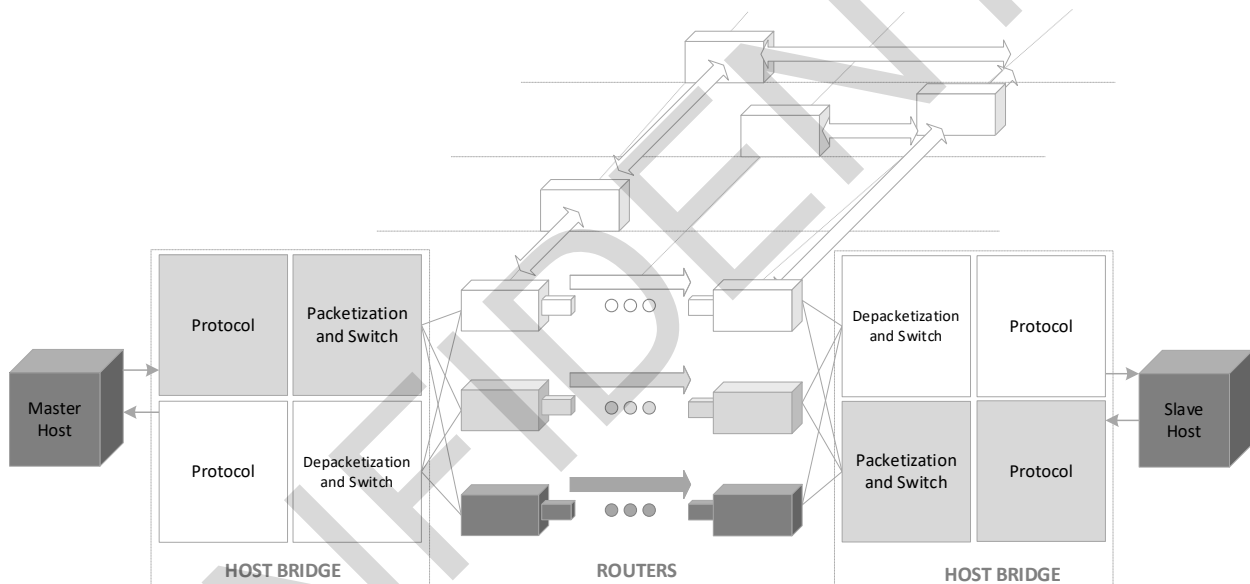


Figure 4 NetSpeed NoC Architecture

4.2 SUMMARY OF FUNCTIONAL SAFETY, RELIABILITY, AND AVAILABILITY

Figure 5 summarizes the functional safety features provided by NetSpeed across the different interconnect components

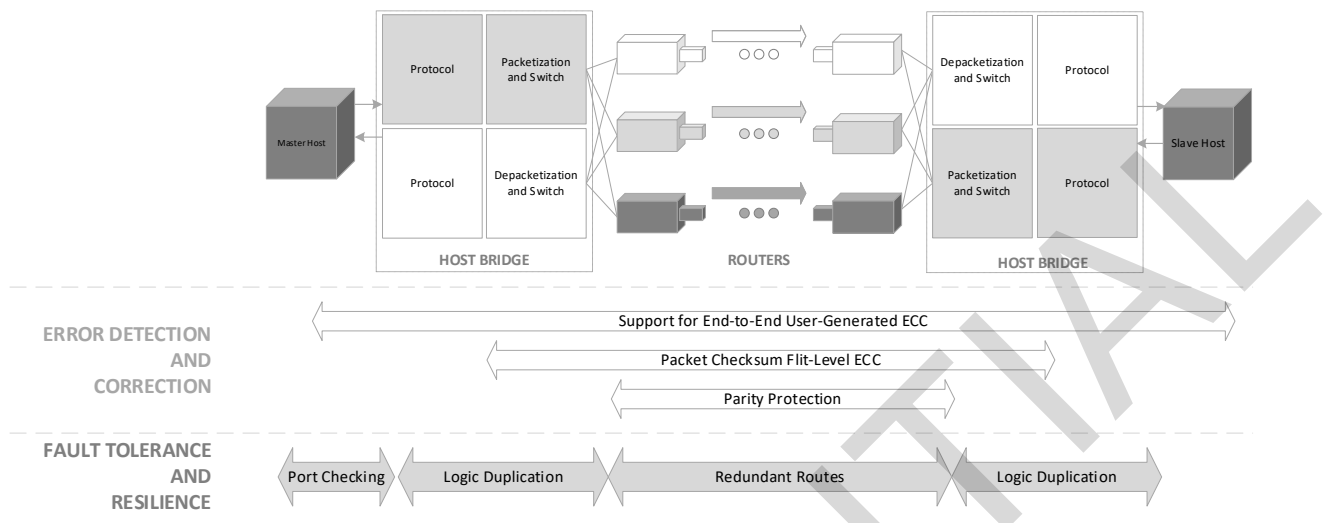


Figure 5 NetSpeed Safety and Resilience Feature Support

4.3 ERROR DETECTION AND CORRECTION FEATURES

The current process for ensuring reliable hardware performance is as follows:

1. Detect and correct errors where possible.
2. Recover from uncorrectable errors through either physical or logical replacement of a failing component or data path.
3. Prevent future errors by replacing in a timely fashion the components most likely to fail.

ECCs were devised to enable the detection and correction of errors. Single-error correction and double-error detection (SECCDED) is just one ECC in common use today. As its name implies, SECCDED allows the correction of one bit in an error or the detection of a double-bit error in a memory block.

Hardware errors can be classified as either of the following:

- DCE—Detected and corrected errors. Handling DCEs is done in the silicon using ECCs and can be made transparent to system components.
- DUE—Detected but uncorrected errors. Handling DUEs requires 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 data path 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 directory read results to confirm that there is no error.

The ECC algorithm uses a SECDED Hamming code with an additional parity bit. The algorithm adds the ECC check bits 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 access the directory RAM directly, including the ECC check bits. 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 the ECC logic within the system. The ECC detection and correction can also be disabled through register access.

4.4 NOC-TRANSPORT ERROR DETECTION AND CORRECTION

4.4.1 End-to-End Transport Integrity

4.4.1.1 *End-to-End NoC ECC Protection*

4.4.1.1.1 Data ECC Protection

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

The ECC is created at the ingress point and the default mode is for the 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, thereby increasing robustness.

4.4.1.1.2 Sideband ECC Protection

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

4.4.1.2 *End-to-End NoC Parity Protection*

4.4.1.2.1 Data Parity Protection

ECC detection and correction comes at a cost to area, so NetSpeed provides users 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 additional latency to the path.

4.4.1.2.2 Sideband Parity Protection

NetSpeed IP also uses end-to-end parity to protect the information carried in packet sideband. At the transmitting end, parity is calculated on sideband segments at the selected granularity; at the receiving end, error detection is performed.

4.4.2 End-to-End User ECC

NetSpeed IP also provides users with a configurable option to generate their own ECC, and the NetSpeed NoC transports them to the receiving end. The 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 Protection

4.4.3.1 Interface Parity

The NetSpeed NoC provides advanced parity protection on the interface to the hosts. This adds protection for the data path 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 they vary between the various channels. For example, for the data interface, granularity can be configured for 1 bit for all data bits or 1 bit per 8-bits.

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

4.4.3.2 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.

4.4.4 Hop-to-Hop Protection

4.4.4.1 Control Parity Protection

The NetSpeed IP can be configured to generate parity protection for packet fields, which can get updated hop-to-hop within the NoC. Routing information, for example, can be parity protected.

These packet fields can undergo modifications, so it is vital to not only protect but also to check and re-compute on a per-hop basis.

4.4.4.2 *Error Detection Using e2e ECC/Parity*

Detection and correction of ECC comes at a cost of area and latency. The NetSpeed IP provides users with the additional configurable option: data-error detection (only) on a hop-to-hop basis, using the ECC or parity carried to protect the data and sideband. This does not incur extra latency because it is only detection, but it does provide a way to localize errors and identify issues sooner, rather than waiting until a check at the receiver.

4.4.5 **End-to-End Packet Integrity**

The NetSpeed IP provides a 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 the integrity of complete packets exchanged on the NoC. This is done by including a checksum that covers the entire transaction payload—all address and control fields that must pass unaltered end to end, as well as some basic identifying information, such as destination ID, source ID, and sequence number. Advanced techniques such as bit-interleaved parity and FLIT identifiers, further enhance the robustness of the NetSpeed IP by ensuring tolerance to errors. All of these techniques are configurable, providing users with the flexibility to choose different levels of protection after carefully considering cost trade-offs.

4.5 **LOGIC PROTECTION AND REDUNDANCY**

Once the transaction is framed into a packet, the NetSpeed IP can verify a correct transmission through the mechanism described in previous sections. However, to guarantee end-to-end resilience, we must 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, and constant parameter arrays—can be configured for parity protection to ensure that faults are detected in these structures, which are integral parts of the path. This applies to the flop structures in the following components:

4.5.1.1 *Flop Structure Parity—Bridges/Routers*

Flop structure parity for bridges and routers covers the flop structures for all master and slave bridges, as well as all routers in the NoC.

4.5.1.2 Flop Structure Parity—Agents

Flop structure parity for agents covers the flop structures in all the agents below:

- Cache coherency controller (CCC)
- IO coherent bridge (IOCB)
- Last level cache (LLC)
- Deadlock avoidance unit (DAU)

4.5.2 Bridge Duplication

For systems that require a higher level of protection, NetSpeed also provides a configurable option to duplicate entire bridges. This option provides the utmost protection from errors for the bridges. 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. A separate clock and reset input are also provided to isolate them from glitches.

4.5.3 Route Duplication

The other piece of the data path—the actual routes between the bridges—also requires protection. This is accomplished algorithmically by duplicating entire routes between transmitter and receiver end points. Only one physical route is active at any given instant, but software can control which routes are being used. If a route is compromised due to errors, for example, the software can switch to a different route (see Figure 7). This is completely under software control, and, most importantly, it has a very low area overhead compared to duplication of the routers themselves.

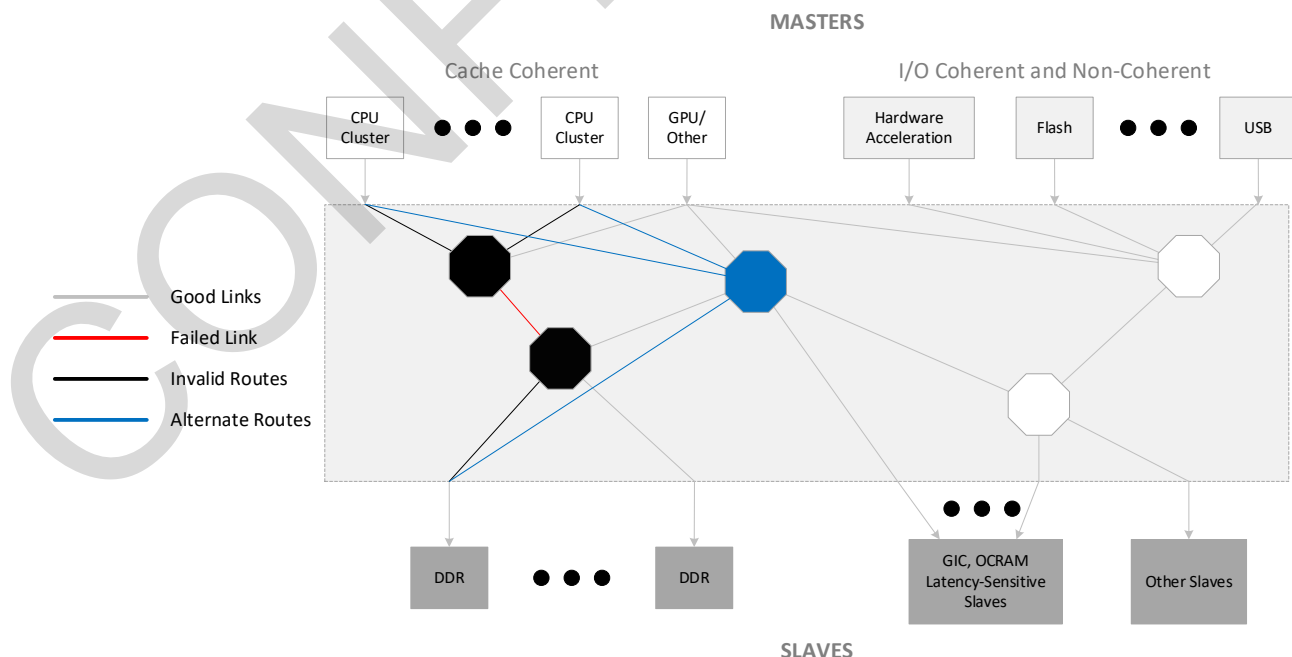


Figure 6 Route Duplication

Route duplication has the following characteristics:

- Configurable and Programmable
 - Design Time—Configurable to support multiple routes
 - Boot Time—Supports selection of initial set of routes
 - Runt-Time—Programmable to switch under software control
- Complete Physical Isolation
 - No sharing of resources (links and routers) between routes
 - Each route can be individually optimized for PPA
- Scalability
 - Mix-and-match support, depending on individual master-slave requirements
 - Automated scalable route optimization to reduce TTM

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—for example, 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 lockstep processor cores and other redundant computing elements. NetSpeed uses a compound bridge to handle these elements, as shown in Figure 8, to compare AXI interfaces and confirm that they are lockstep equivalent.

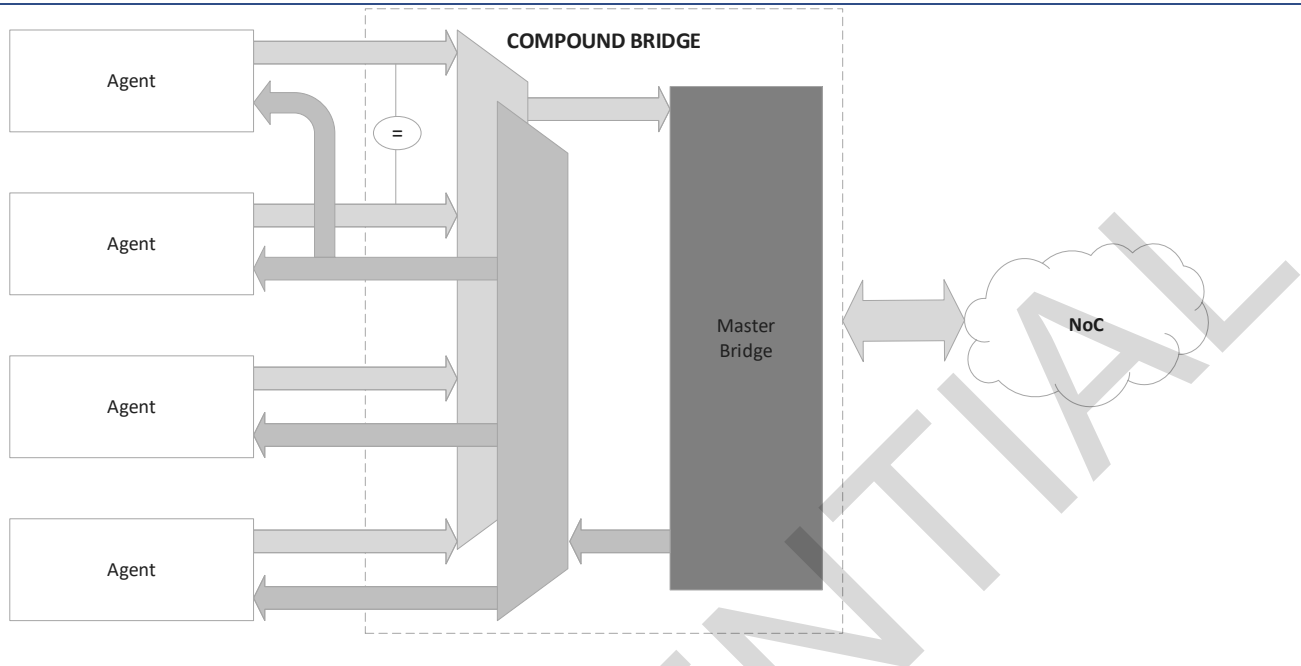


Figure 7 Compound bridge to Address Redundant Port Checking

4.5.5 NoC Register Parity

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

4.6 MEMORY ECC

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 check bits 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, allowing failures in that logic to be detected. The goal 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. They apply to coherent as well as non-coherent components of the NetSpeed IP.

4.8 TIME-OUTS

Various configurable options are available for handling time-outs in the NetSpeed IP using high resolution counters with programmable timestamps. A maskable interrupt is also raised to the CPU with a detailed syndrome of the timed-out request

4.8.1 Target Time-Outs

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

4.8.2 Initiator Time-Outs

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

4.8.3 NoC Time-Outs

Another layer of time-outs occurs based on backpressure from the slave device for requests and master devices for responses from the NoC. This can cause backups in the NoC, potentially blocking other traffic. Time-outs for these events can be configured to start dropping requests or responses at the destination and raise fatal interrupts for CPU intervention.

4.9 STATUS OF TRAFFIC TO POWER GATED BLOCKS

Interrupt status bits are maintained that flag any errors causing the traffic to be misrouted to a gated power domain.

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 the element that raised an alarm. For additional granularity, multiple types of interrupt signals are also tracked from each element. The detailed information is accessible

through regbus access to status stored in the element that sources the alarm. Errors are counted where they happen, and alarms (interrupts) are raised and delivered to the fault controller. Correctable ECC errors do not halt progress, but are counted and an appropriate interrupt is raised.

4.11 CONFIGURATION OPTIONS THROUGH NOCSTUDIO

Different end-user applications, from servers to data-center storage to automotive applications, often require different levels of resilience and reliability. Most options described above can be configured through NocStudio. All the resilience options are a trade-off between robustness and area/power. Architects can evaluate these trade-offs and make appropriate design decisions using NocStudio.

5 EXTERNAL SAFETY MECHANISMS

For designs that target ASIL-C or ASIL-D, it may be necessary to add external diagnostics in addition to the safety coverage provided internal to the Netspeed IP.

5.1 NETWORK BIST ARCHITECTURE

Figure 9 is a logical diagram of the NetSpeed Network BIST, an on-chip feature that performs a Built-In Self-Test (BIST) on the network to verify that it is functioning properly. Network BIST issues a sequence of packets into the network, then verifies that these packets are received without data corruption on the far side. The feature checks for stuck-at values in the network links and all network storage structures.

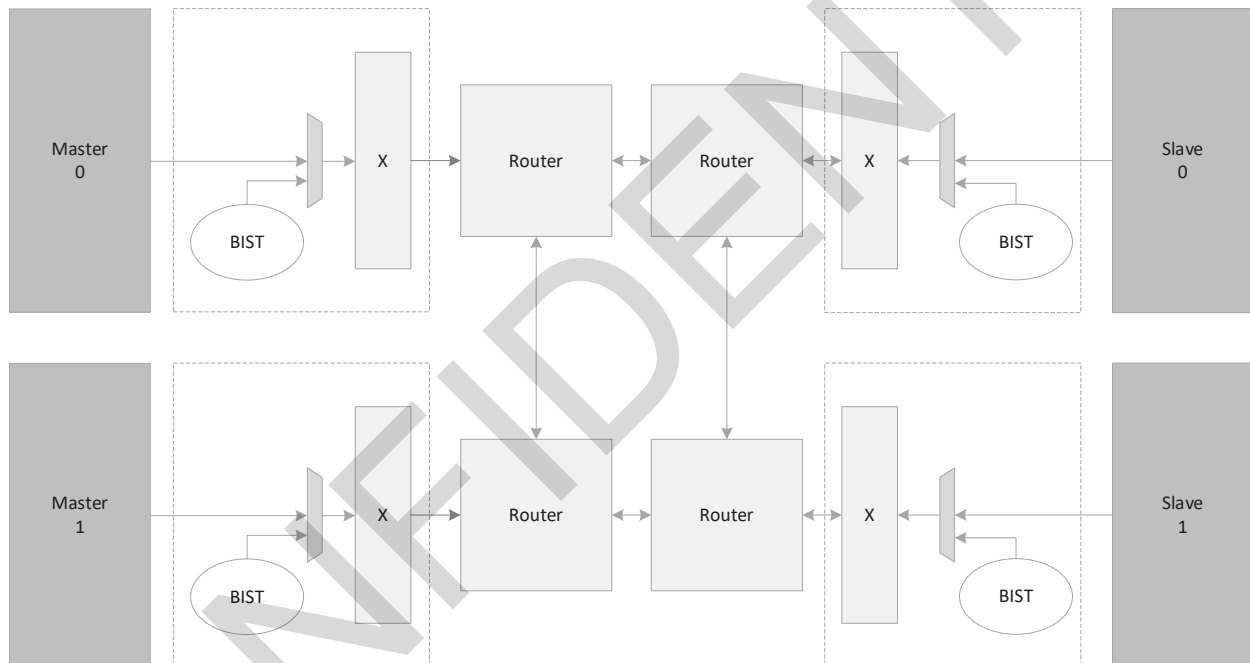


Figure 8 Network BIST

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 to expect for BIST. This way they can identify whether all packets have arrived and whether they all have the correct values.

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

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

5.1.1 BIST Control Interface

The BIST Control module is a global state machine that controls Network BIST (see Figure 10).

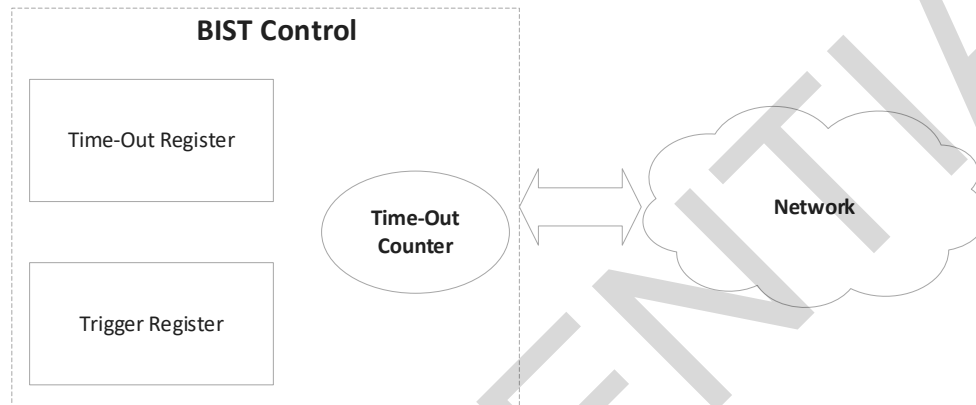


Figure 9 BIST Control

The Network BIST sequence is triggered by a register access to the BIST Control. When triggered, the BIST signals to all bridges through a sideband mechanism that the BIST sequences should begin. Each bridge initiates its sequence of packets.

The BIST Control also has a time-out value register that indicates the amount of 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 that no false failures are flagged. When the BIST sequence is started, a timer is triggered and it counts until the programmed value is hit. If any packets have not arrived at the destination, an interrupt is signaled to indicate that a BIST error has occurred.

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

5.1.2 BIST Packet Length and Value

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

To check stuck-at faults, the BIST sends multiple packets through each route in the network. Each packet carries a different payload value to detect stuck-at values. The packets 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. For these patterns to reliably hit each data path storage element, each packet must be long enough to fill the longest FIFO along its path.

In Figure 11, 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. Packets sent along the west-to-east path must be long enough to fill the entire 8-entry FIFO. This ensures that each FIFO entry is written with each of the patterns.

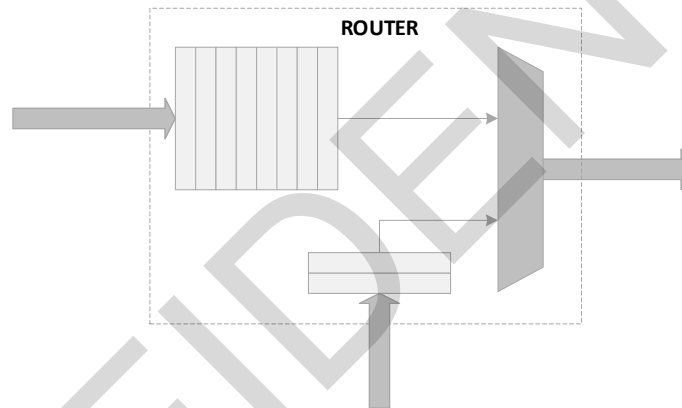


Figure 10 Router FIFOs

Because upsizing and downsizing in the network is possible, the packet length might 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.

5.1.3 Concurrent BIST and Normal Packets

The BIST packets are marked with an additional bit to indicate that they are for BIST only. This allows the end-point to discard the packet instead of sending it through to the agent, as well as allowing it to track the number of packets that have arrived.

Because BIST and normal packets are distinguishable from one another, they can use the network simultaneously. 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 backpressure in the network, for example, causing the BIST execution time to be longer. The time-out 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, however, allows Network BIST to be executed multiple times, instead of just at reset. The BIST could be triggered periodically to test that the network continues to function correctly.

5.1.4 Power Requirements

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

5.1.5 Reset Sequence

While the BIST engine can be triggered through register accesses, doing so requires the network to be active. An active network also allows any other accesses needed to get that far in the sequence. It might be desirable to have the BIST engine triggered on system reset, before any of the agents are active, to test the network before anyone uses it. This can be configured in NocStudio.

5.2 HIGH RELIABILITY RAM

NetSpeed IP provides wrappers for the SRAMs, which the SoC integrators replace with real SRAM modules. It is required to ensure these SRAMs have a fault coverage of at least 95%.

5.3 HOST PROTOCOL VIOLATIONS CHECKER

To ensure coverage against faults that result in protocol violations, the SoC that is instantiating NetSpeed IP must be capable of checking for the protocol correctness on the interfaces. This can detect any malformed transactions due to faults.

5.4 POWER-ON TESTS

To ensure the reliability of the chip, the system must be equipped with mechanisms to conduct appropriate structural tests.

First, the system must have a fault injection mechanism to validate error detection. All large SRAM structures, such as caches and coherency tables in the Agents and Bridges, have the ability to inject faults through software. Diagnostics to inject faults into these structures and validate that

the faults are detected and reported accurately ensures proper operation of the error detection and reporting circuitry.

Moreover, structural tests such as MBIST for memories and LBIST for logic, as well as any other structural tests at every power-on of the devices, ensures the logic and memories continue to be reliable. This pre-emptively uncovers a class of errors that would cause safety violations during normal operation of the device that expressed NetSpeed IP.

CONFIDENTIAL

6.2 NETSPEED SLC FLOW

The NetSpeed SLC flow is mapped as described below.

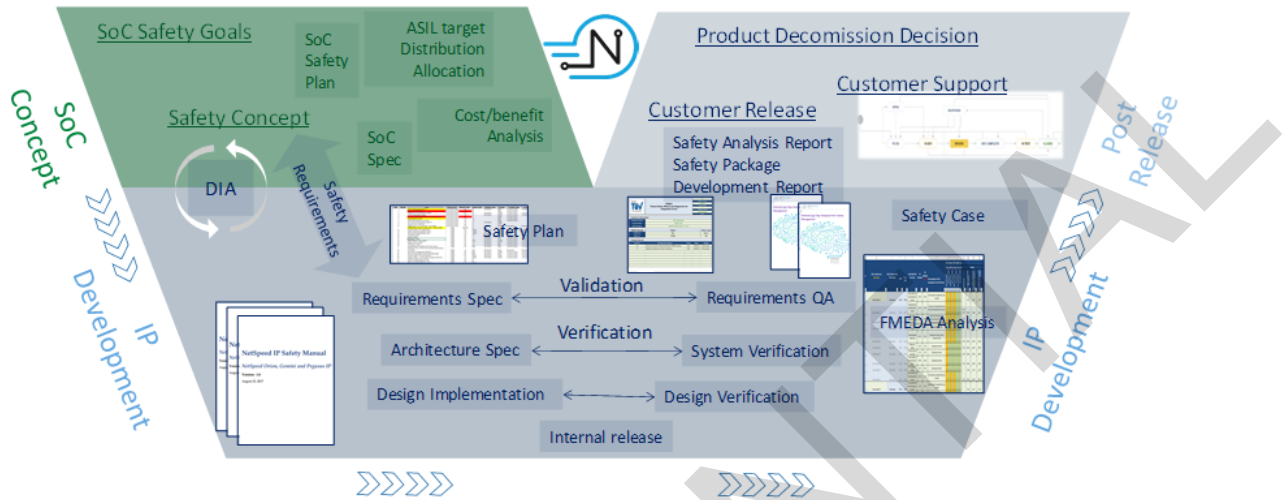


Figure 12 NetSpeed Safety Life Cycle

6.3 CONCEPT

The NoC development process starts with the Concept phase, in which design requirements are gathered from customers and internal roadmap targets. Safety features and reliability/quality requirements are enforced during the requirements-analysis portion of this 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 must be documented with a unique ID (subdivided if required). Every requirement can be identified and its status in the development cycle can be tracked with a unique ID in JIRA (see Figure 12).

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

Each of these requirements has a corresponding architecture requirement, which in turn have micro-architecture requirements. These are all defined, modified, coded, verified, validated, and prepared for releases. And they are all tracked individually to ensure that all requirements have been filled.

6.3.2 Tracking Safety Requirements

All functional safety (FuSa) requirements in the JIRA database are part of the NS-2350 group and are documented with unique IDs, as shown in Figure 12.

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

Sub-tasks




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

FuSa requirements such as NS-2350 are grouped based on their context. The NS-2350 FuSa improvement requirement, for example, comprises three sub-requirements: NS-2358—ECC on RAM index, NS-2359—End-to-end parity, and NS-2368—Parity on registers.

Functional requirements for all safety aspects are documented in detail and are version-controlled in the SVN database. In the example presented in Figure 14, all safety, reliability, and serviceability features are documented in the FuSa Req, Document A.

An example can be found here:

https://app.assembla.com/spaces/netspeed_noc/subversion/source/HEAD/trunk/doc/planning/Planned%20FuSa%20features.xlsx

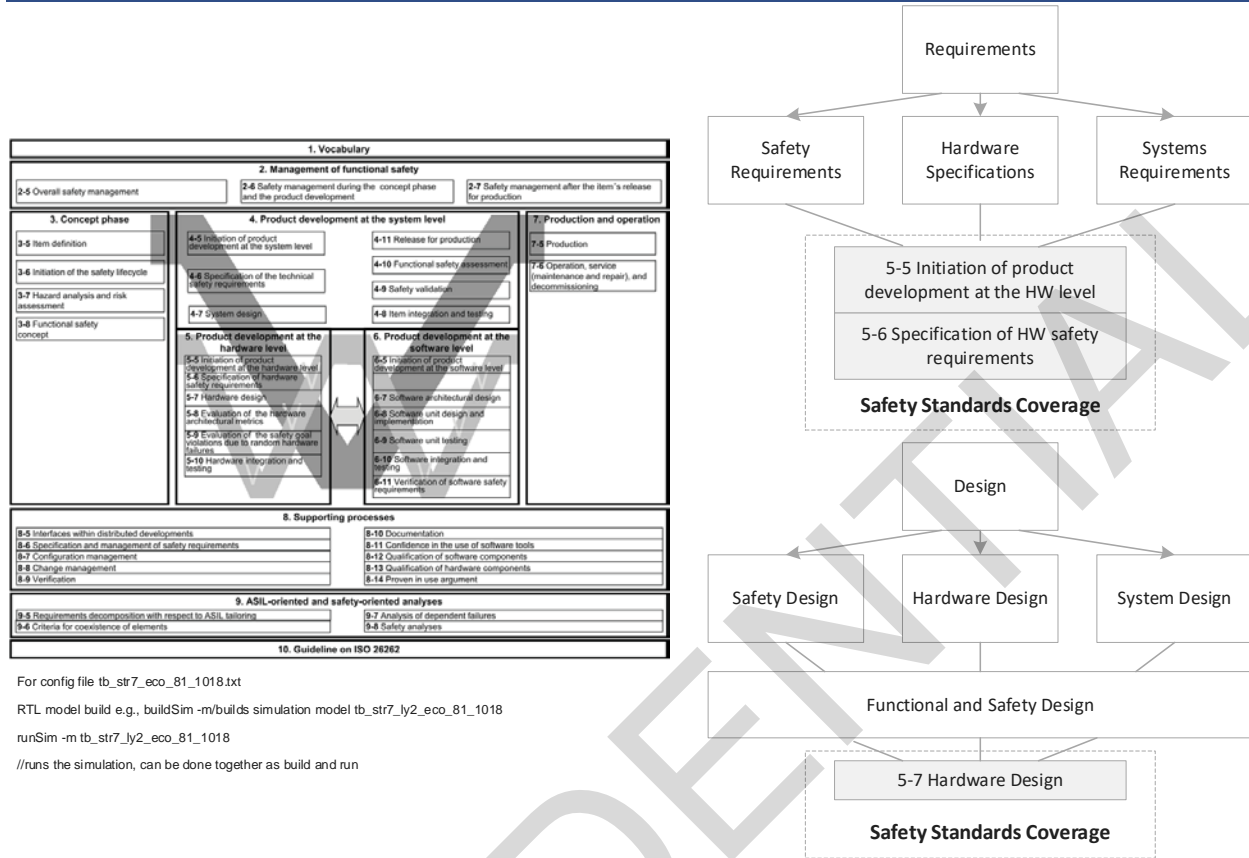


Figure 15 ISO 26262-Part2

Document B in the example is a detailed technical specification for respective parts of the design or architecture; it is version-controlled in SVN. Document C is the NocStudio configuration file, which describes the complete system environment or the complete servicing flow. This configuration creates the customer IP and 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 Product Development phase. In this phase, the architects translate the requirements into specifications. Safety features are an integral part of this architecture.

An example can be found here:

https://app.assembla.com/spaces/netspeed_noc/subversion/source/HEAD/doc/ccNoC/Coherency%20Architecture.docx

Figure 15 illustrates the release process, from Test plan, to test design, through execution, to reports with metrics results until all planned items are passed or tested with success and then released. For more information on regression tests and code coverage, see section 7.3, Regression Testing and section 7.2.8, Coverage Metrics, respectively.

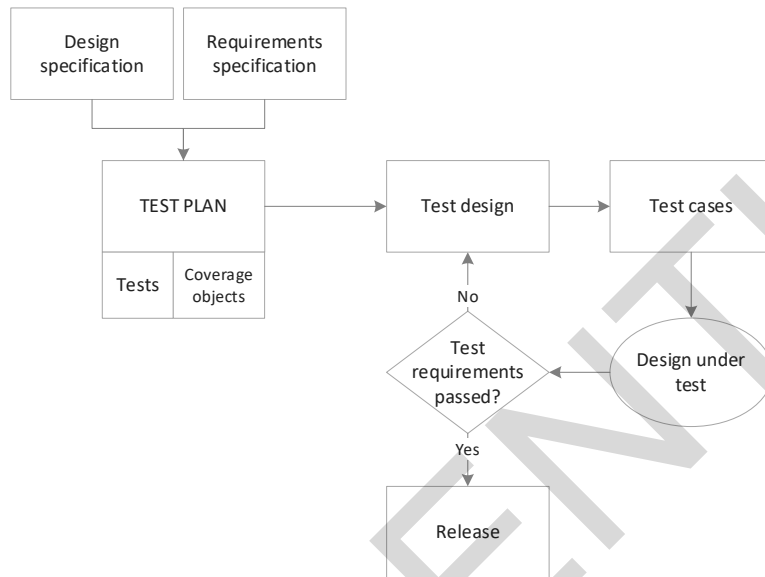


Figure 16 Test Plan and Design Flow

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

An example can be found here:

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. The verification team develops a test plan using architecture specifications.

An example can be found here:

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 test plan and are run on the design using the simulation tools from Cadence and Synopsys. Automated test methods and systematic regression processes are used to verify the design. Details of this process are covered in detail in Section 7.2, Development Verification Plan.

At the completion of the verification cycle, an elaborate release process makes the IP ready for shipment to the customer.

6.5 POST-RELEASE PHASE

6.5.1 Field Monitoring

The SLC continues after the product is released. The Post-Release phase involves handling of customer issues, which goes through a well-defined process. Each customer has a point of contact (PoC) in the application engineering team. After initial triage and communication with the customer, the problem is transferred if it requires a design change. It goes through the development-and-release process before the application engineer makes a release, or provides documentation with a workaround, available to the customer.

This data is tracked in a JIRA-based tracking system and is used to mine information in the future. The data is maintained at

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

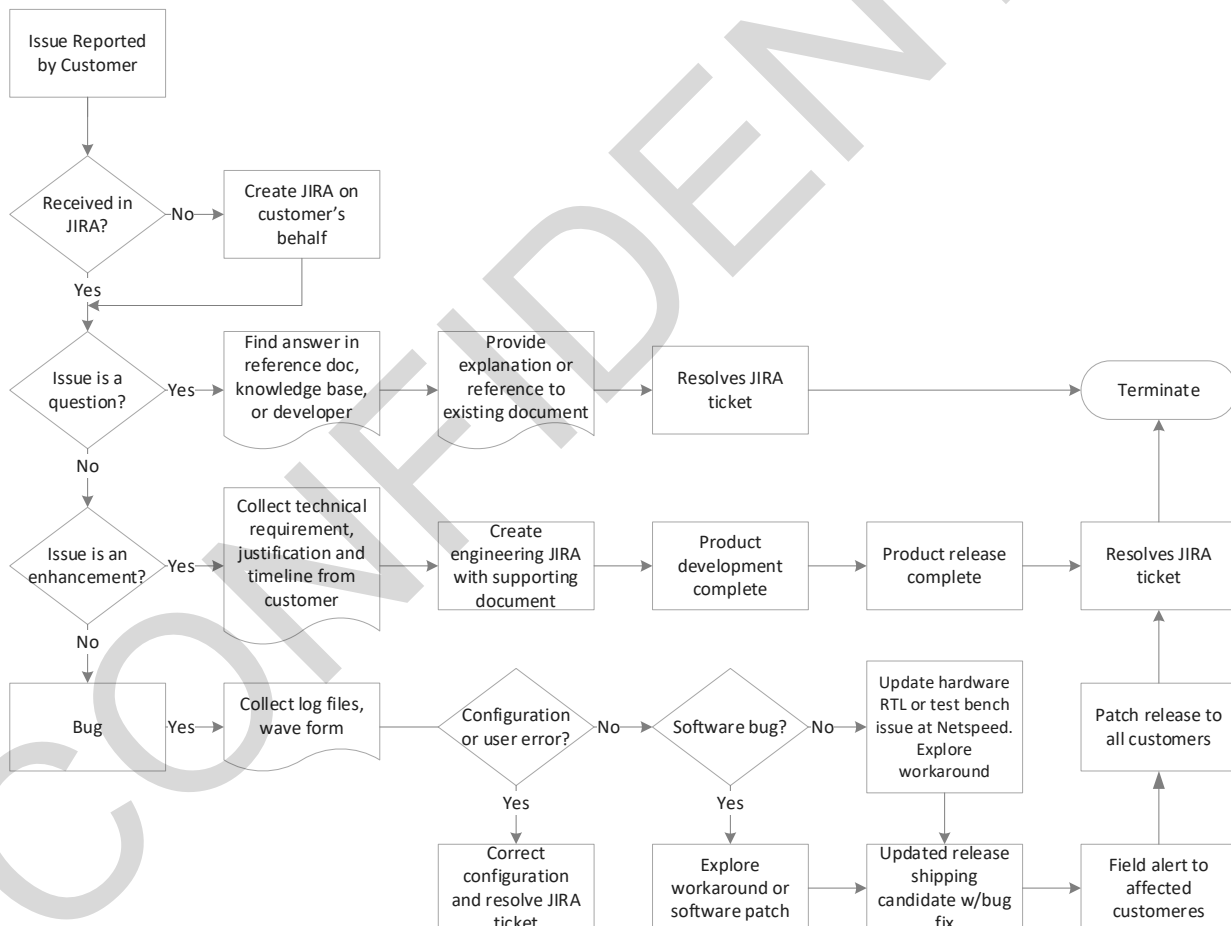


Figure 17 Post-Sales Customer Engagement Flow

The data from the customer JIRA database is periodically mined for analysis, and findings 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 and address the problems that escaped to the previous releases, ensuring continuous quality improvement.

Product-decommissioning decisions are made by a steering committee based on 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 is done on these products.

CONFIDENTIAL

7 DEVELOPMENT PLAN

7.1 IP DESIGN AND INTEGRATION

7.1.1 Hardware Design Flow

The design flow, shown in Figure 17, starts with a requirements specification as an input and ends with a fully verified product as the output.

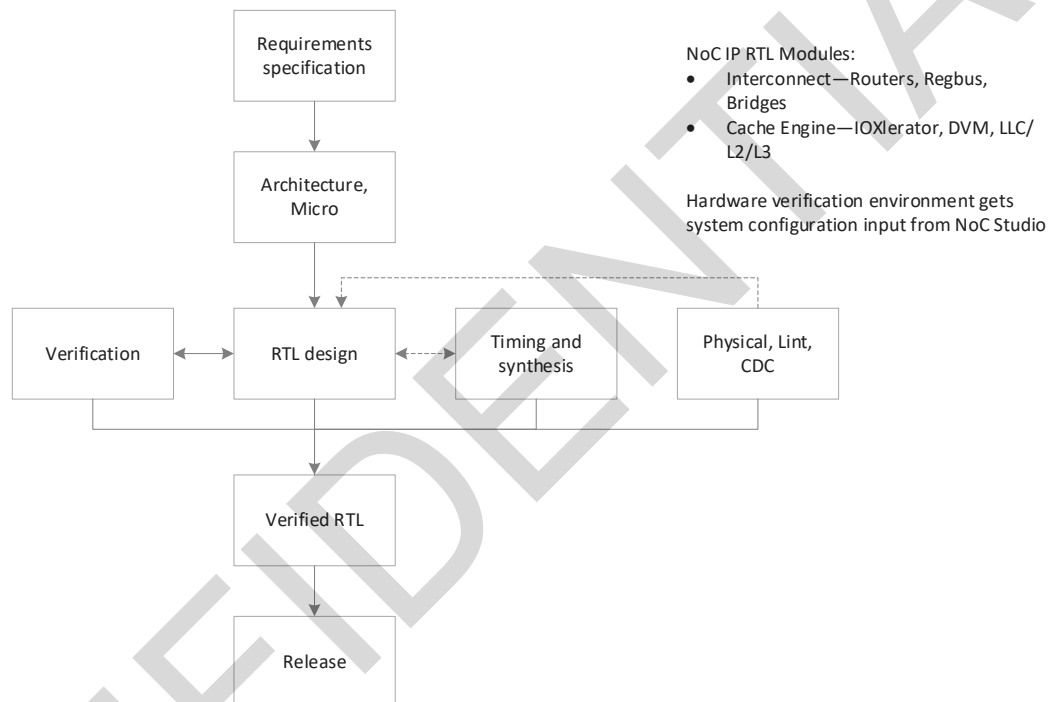
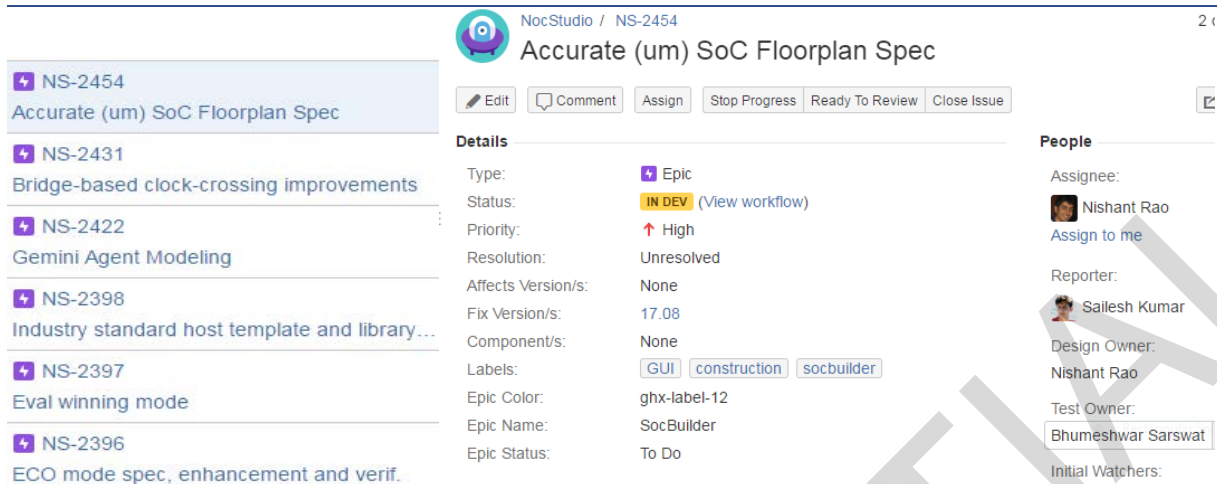


Figure 18 Design Flow

7.1.2 Hardware Design Specification

IP requirements are assigned unique ID numbers and are documented in a common repository in JIRA with their respective priorities, resolution status to track the implementation, testing, review, and closure.

Figure 18 is an example of a list of requirements and their respective unique ID numbers, shown as NS-XXXX.



The screenshot shows the JIRA interface for a project named 'NocStudio / NS-2454'. The left sidebar displays a list of requirements, with 'NS-2454 Accurate (um) SoC Floorplan Spec' highlighted. The main area shows the details for this item, including its type (Epic), status (IN DEV), priority (High), and resolution (Unresolved). It also lists the assignee (Nishant Rao), reporter (Sailesh Kumar), design owner (Nishant Rao), and test owner (Bhmeshwar Sarswat). The right sidebar shows the 'People' section with the same information.

Figure 19 Requirement List

In the example, Item NS-2454 is highlighted on the left side of the window, displaying several fields on the right side of the window. These fields provide further details about the highlighted item, including the priority of the item, status of the specification feature/requirement, and whether the item is closed or still unresolved. All other fields in JIRA, such as the owner of the item, assigned person for development, and testing, are used for project management.

7.1.3 Design Methodology

Design methodology comprises RTL coding styles and release procedures; they are controlled by detailed checklists. These conventions are documented and ensure 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 or modification to these documents is version-controlled.

Example:

Ring Master and 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 must go through a detailed checklist covering the following areas:

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

A detailed checklist is maintained at:

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

7.2 DEVELOPMENT VERIFICATION PLAN

The NetSpeed verification process starts with a verification plan based on the functional specification. The process involves developing a verification test bench infrastructure, test stimuli to exercise various states of the design, and checkers to verify the correctness of the design. This is an iterative process until the verification is complete. Completeness is determined by verification metrics that involves coverage completeness and a dropping of the bug rate to an acceptable level.

7.2.1 Verification Cycle

Figure 19 depicts the verification process flow with a verification plan, with a test-bench infrastructure for the execution flow. Once all planned tests pass, the module, or eventually the NoC IP, is ready for signoff.

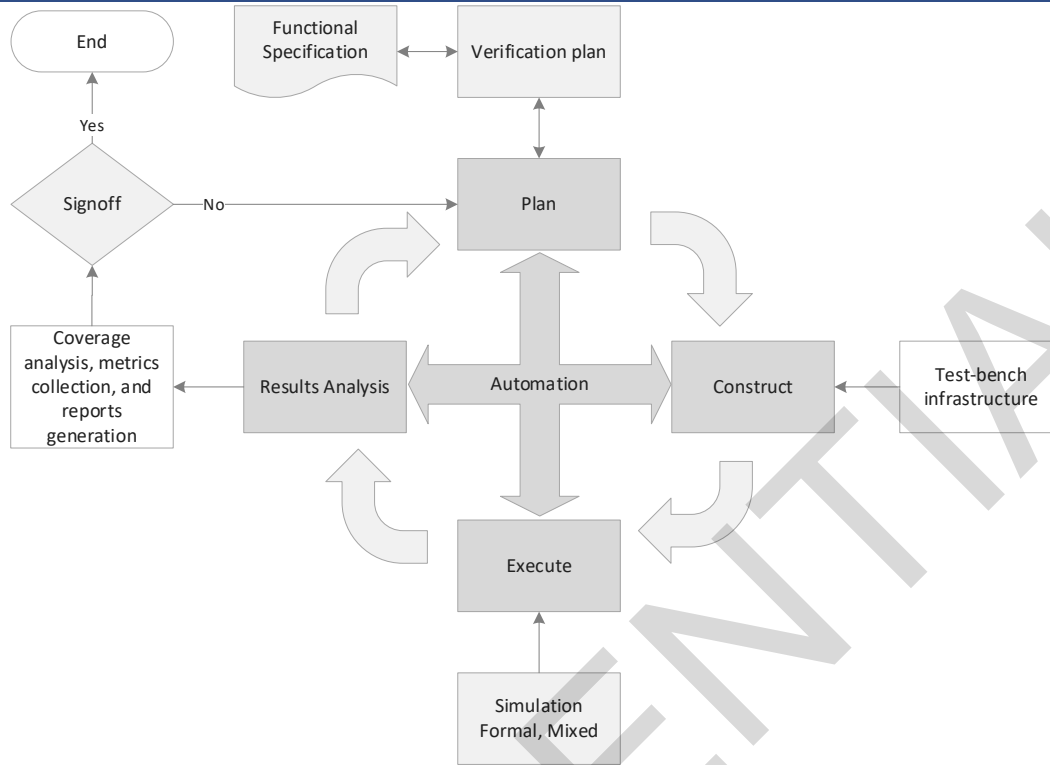


Figure 20 Verification Process

7.2.2 Inputs to Verification Plan

The main inputs to a verification plan are as follows:

- Functional requirements
- Design specification
- Infrastructure and methodology

Each of these inputs is described below.

7.2.2.1 Functional Requirements

These requirements can be seen in section 6.3.1 Requirements, of this document.

7.2.2.2 Design Specification

The specifications are discussed in section 7.1.2, Hardware Design Specification, of this document.

7.2.2.3 Infrastructure and Methodology

Information about Infrastructure and Methodology is available in Confluence as *Infrastructure_and_Methodology*.

7.2.3 Test Plan Template

The template for the test plan is shown as follows:

Table 2 Test Plan Template

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
.....
.....
.....

Test plans are documented and maintained in 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

7.2.4 Levels of Testing

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

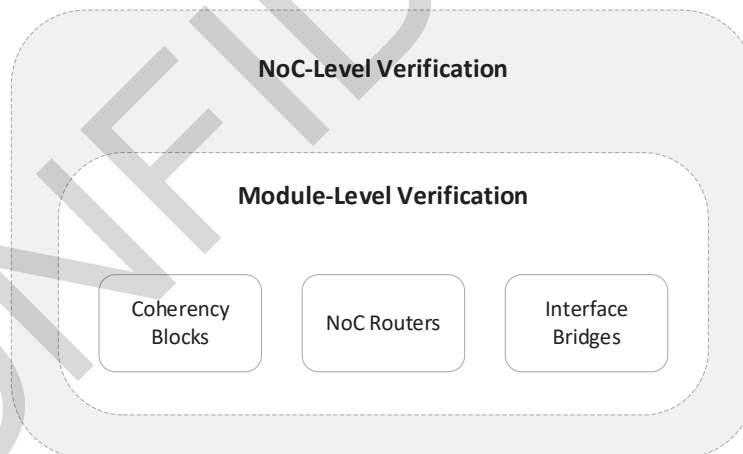


Figure 21 Verification Hierarchy

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

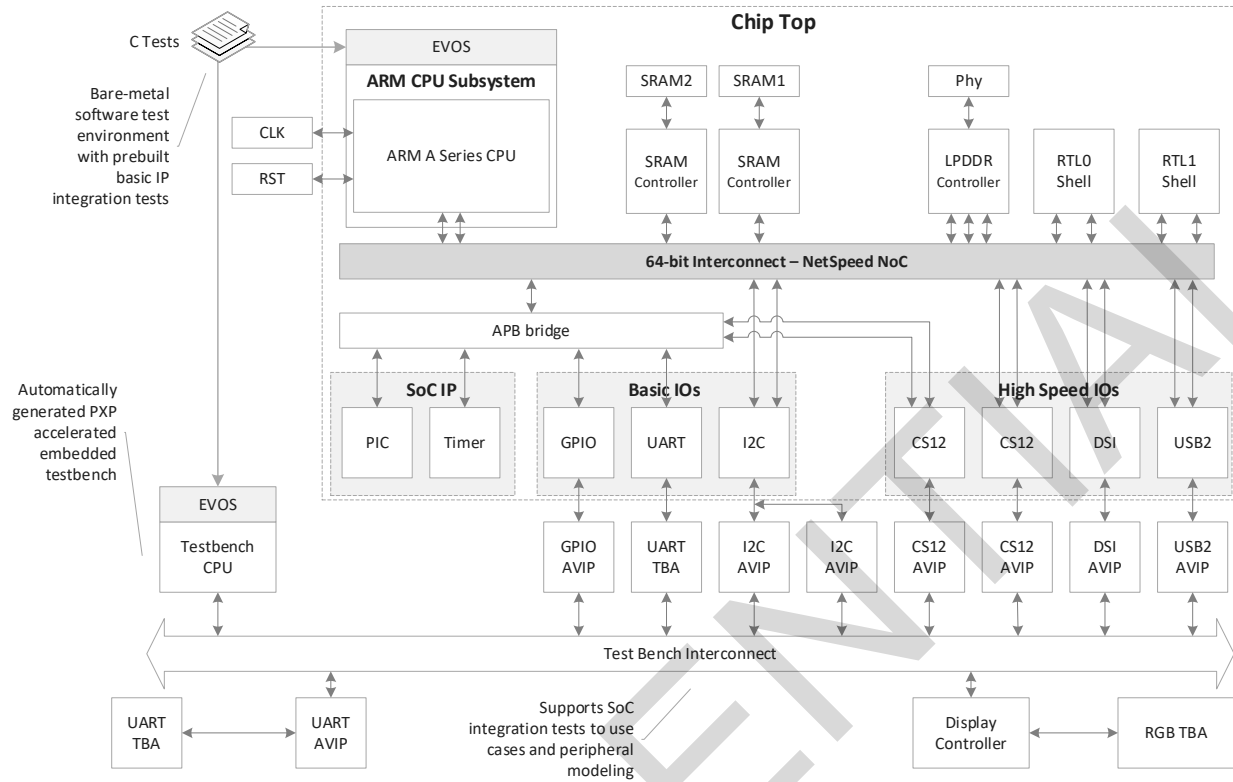


Figure 22 Palladium Emulation

7.2.5 Types of Tests

The test stimulus comprises directed and constrained random tests. Directed tests are hand-coded to verify one or more targeted conditions, whereas constrained random tests are automatically generated to verify targeted functions and unforeseen conditions.

7.2.5.1 Constrained Random Tests

Figure 22 illustrates the main components used in automated constrained random testing.

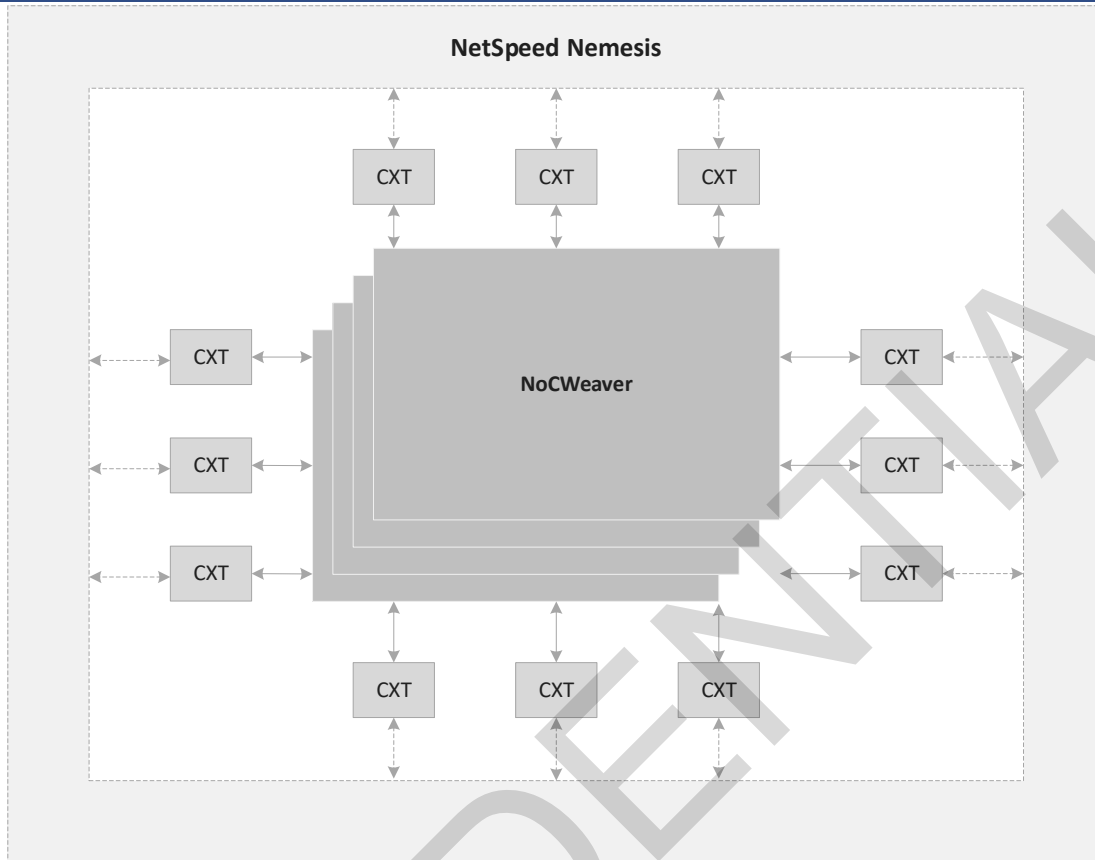


Figure 23 NoCWeaver, CXT, Nemesis

The main items are as follows:

- NoCWeaver—Pseudorandom NoC configuration generator
- Configurable Transactor (CXT)—Emulation-ready transactor to model agent behavior
- Nemesis—Sophisticated traffic-generation system that creates complex, coordinated tests covering all possible or allowed sequences that are coordinated and randomized. Most importantly, the cache non-coherent memory accesses are examined for the correct coherency controls in verification and the results are reported accordingly.

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

Approximately 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 stimuli and rely on checkers to ensure the design functions correctly in response to the stimulated condition. Various checkers used in the simulations are listed below:

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

7.2.7 Safety and Reliability Verification

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

Each functional safety feature designed or implemented gets verified step-by-step as the design evolves in a 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 and functional coverage of the predefined cover points ensure that the verification test exercises and checks a complete design.

Code coverage measures the following types:

- Block
- Expression
- Line
- Path
- Toggle
- FSMs

Tools are developed to automate the measurement process.

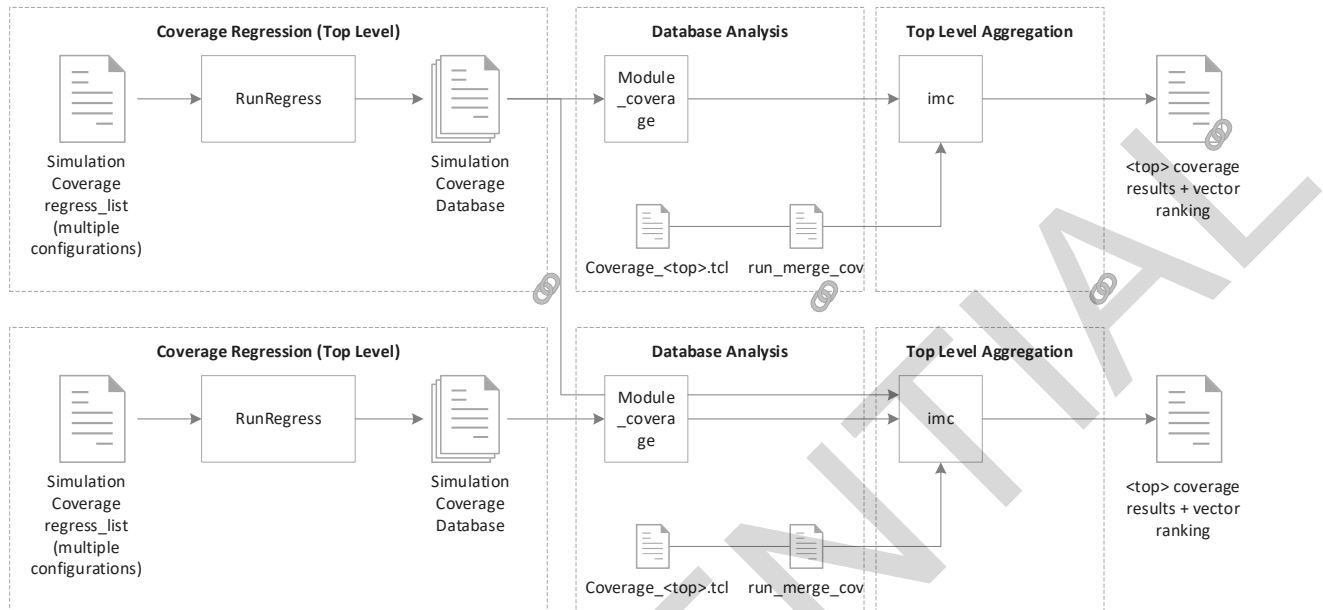


Figure 24 Coverage

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

7.3 REGRESSION TESTING

Continuous regressions are run of all passing tests to ensure the design stability is maintained. The types of regressions are as follows:

- Smoke—This is a quick check. The regression takes approximately an hour to complete
- Nightly—A thorough check is done, taking approximately 16 hours to complete.
- Weekly—This is an exhaustive regression tests that often takes several days to complete.

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

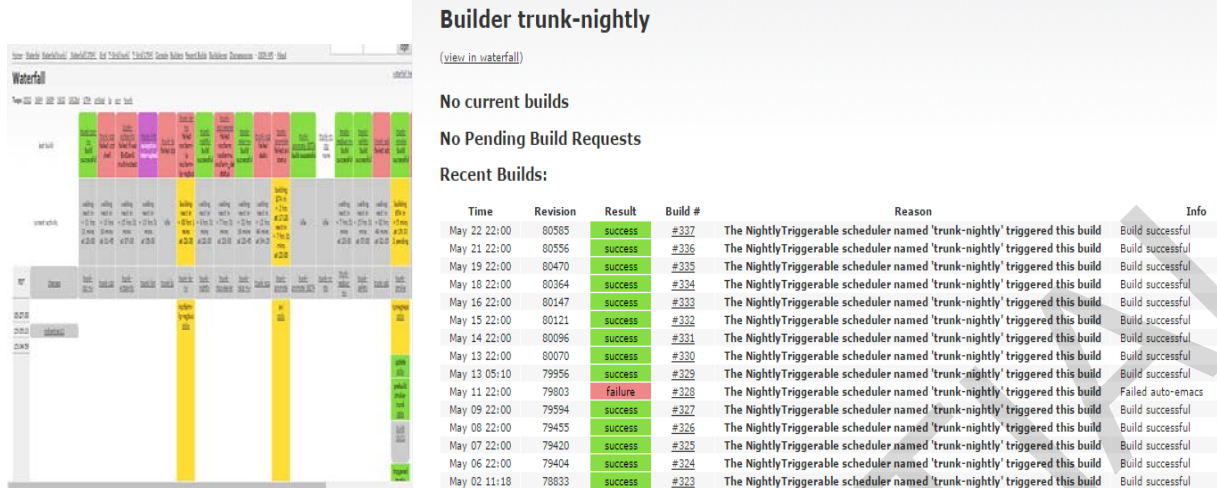


Figure 25 Build Schedule

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

7.3.1 Bug Tracking—JIRA

All design defects, or bugs, are tracked in the JIRA bug-tracking system. The bug-tracking flow from the Opened state to the Closed state is shown in Figure 25.

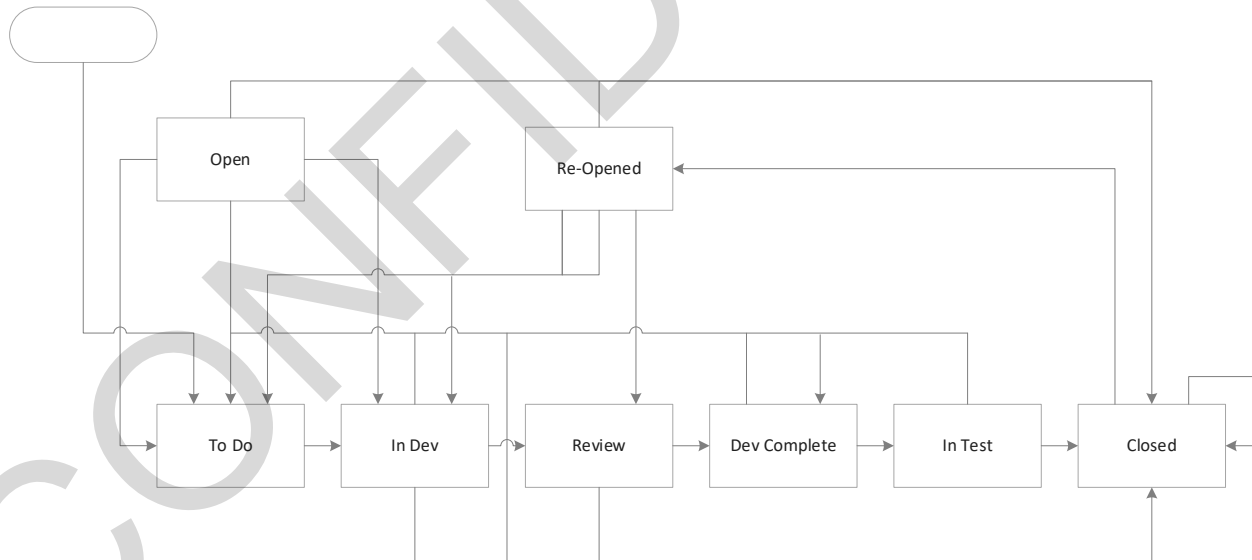


Figure 26 Bug Tracking

Open bugs go into development, followed by testing. Review phases appear in both the development and testing phases. Review defects are logged into JIRA with responses often resulting in iterative phases of development and testing. Additional items can be opened or implemented. If the issues can be tested and reviewed with no defects, the issue is closed.

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

7.3.2 Checklist

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

These checks include the following:

- Test plan
- 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 can be found here:

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

8 SAFETY VERIFICATION

8.1 SAFETY VERIFICATION CODES

Table 2 lists and describes the safety verification codes.

Table 3 Verification Codes

Code	Description	Verification Method
DI001	Control and status register (CSR) for parity protection	CSR parity injection sweep with interrupt count
DI002	Unused	Unused
DI003	Flop structure parity protection	Error injection of each bit of the flop based control structures
DI004	Data ECC protection	Transparent 1-bit and 2-bit error injection at every valid link on the NoC
DI005	Data parity protection	Parity error injection at every valid link on the NoC per granularity of protection
DI006	Status of traffic to power gated blocks	Error injection with checkers monitoring power boundaries for validation
DI007	Transport ECC	Directed testing of 2-bit errors and random injection of correctable errors
DI008	Time-outs detecting requests dropped, stuck, or incomplete	Validation by stalling slaves to generate time-outs at the relevant bridges
DI009	Unused	Unused
DI010	SRAM data and index ECC	Directed testing of ECC logic and error injection into the RAMs using preload and/or diagnostic features

8.2 FAULT/ERROR VERIFICATION

Safety features such as link-level parity checks and end-to-end ECC correction are verified by injecting an error at select points, then observing the detector response.

8.2.1 Transport Verification

The following methods are used to verify the transport operation of the design.

- A parity error is injected on a parity-enabled subfield, then the interrupt response is monitored.
- Parity checkers are enabled by an active VC on the link. The error bit then breaks protocol or corrupts data and higher-level protocol checkers detect it
- An uncorrectable error is injected. This is similar in effect to a parity-error injection on a parity-only link.
- A correctable error is injected (single bit error on a data or usr_sideband signal) and the interrupt response is monitored.
- Correctable ECC errors are randomly injected into regressions of non-error tests and results are checked to ensure normal operation

8.2.2 CSR Parity Injection Sweep with Interrupt Count

For each eligible CSR register, use the following sequence for verification:

1. Inject a fault on an eligible bit.
2. Allow the fault to propagate to the top-level interrupt pin.
3. Clear the fault and interrupt.
4. At the end of the simulation, assert that the total number of interrupts pulses equals the number of eligible CSR registers.

8.2.3 Interface Parity

Injectors and checkers implemented in the test bench along with random test stimulus

9 SAFETY ANALYSIS RESULTS

9.1 COVERAGE ASSUMPTIONS

This section describes coverage assumptions that are made in the FMEDA analysis.

9.1.1 ECC

ECC coverage is assumed to be 99%.

9.1.2 Parity

Table 3 and Table 4 show calculations upon which parity coverage is based. These involve accurate assumptions based on semiconductor error distributions.

Table 4 Diagnostic Coverage for Parity Detection—By Code

	Data (Bits)						
Code	8	16	32	64	128	256	512
1	91%						
2		95%					
4			96.82%				
8				98.81%			
16					99.42%		
32						99.70%	
64							99.85%

Table 5 Diagnostic Coverage for Parity Detection—By Error Type

	Data (Bits)						
Parity	8	16	32	64	128	256	512
1% 1-Bit Error	100%	100%	100%	100%	100%	100%	100%
9% 2-Bit Errors	0%	50%	66%	87.50%	93.75%	96.875%	98.4375%
90% >2-Bit Errors	50%	75%	87.50%	93.75%	96.875%	98.4375%	99.21875%

9.2 FAULT INJECTION

Error injection capabilities are available to be exercised on the silicon that expresses Netspeed IP. They are available for RAM-based structures, which will be expanded in future to large flop-based structures.

9.3 FMEDA

FMEDA analytical analysis is done qualitatively at this stage. The results are documented in a separate file.

Fault coverage measurement and validation is expected to be done in the near future using commercial EDA tools.

9.4 DFA

DFA is documented in a separate file.

10 TERMINOLOGY

Table 6 Terms and Definitions

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

Verification	Testing to ensure implementation follows Specification
Virtual Circuit	Logical path between two end points in the NoC

CONFIDENTIAL

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