Safety, Serviceability and Security Feature Support in NetSpeed IP

Application Note for Automotive, Industrial and Mission-Critical Applications

Version: 1.1

August 15, 2015

Safety, Serviceability and Security Feature Support in NetSpeed IP

About This Document

Application note on Safety, Serviceability and Security Feature Support in NetSpeed IP.

Audience

This document is intended for users of NocStudio:

- NoC Designers
- SoC Designers
- SoC Architects

Prerequisite

Before proceeding, you should generally understand:

- Basics of SoC Architecture
- Basics of NetSpeed NoC IP

Related Documents

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

- NetSpeed NocStudio Orion User 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

Contents

1 INTRODUCTION

Today's mission-critical applications such as enterprise resource planning (ERP) and safety-critical applications such as Advanced Driver Assistance Systems (ADAS) require newer technologies at better price points to deliver to their promise. A failure affecting an SoC can easily cost millions of dollars or worse human lives. All of this points to a need for creating highly scalable and resilient SoCs that are well suited for critical business applications and large-scale consolidation. SoC interconnect IP and Cache-Coherency IP play a critical role in delivering the needed reliability, availability and serviceability requirements.

1.1 TRADITIONAL METHODS

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. Today's SoCs require the handling of unrecoverable hardware errors, while delivering uninterrupted application and transaction services to end users.

1.2 NEW APPROACH

New approaches handle unrecoverable errors throughout the complete stack, from the underlying hardware to the application software itself. Such solutions involve three components: Functional safety, serviceability and security.

1.2.1 Safety

Functional safety is ensuring the protection of data through detection and correction of errors. Error detection ensures that errors are identified at the data and instruction level. Error correction addresses a detected error by restoring the erroneous data bit or bits to their correct value. Error containment ensures that compromised data is flagged as such across all major components and data pathways so that subsystems other than the failing one can take appropriate action should they encounter such errors. Safety mechanisms also include multiple levels of redundancy, automated failover at the silicon and hardware levels.

1.2.2 Serviceability

Serviceability is enhanced by using predictive failure analysis to identify problematic components before they cause uncorrectable errors or actual downtime, thereby simplifying replacement of components in the case of hard failures. System partitioning is used to further

isolate workloads affected by uncorrectable errors from other active workloads running on the same server infrastructure, and facilitates maintenance.

1.2.3 Security

Applications running on mission-critical SoCs have evolved into using open software platforms and applications. These open and potentially untrusted applications create a security risk to the hardware and the communication at an SoC level. Security is ensuring that the interconnect solution should be cognizant of such security risks and provide architectural security support at multiple levels.

1.3 Safety, Serviceability and Security Features of NetSpeed IP

NetSpeed's IP implements a powerful collection of functional safety, serviceability and security features. Safety features enable the interconnect IP to actively and passively monitor all its key components. Serviceability features enable the NoC to proactively and reactively repair known errors and minimize future ones by acting automatically based on configurable options. By providing hook-ups to other SoC IP like fault controller, Power Management Unit, Regbus master, it enables collaborating and tight integration with hardware and software stack. Security features provide multi-tiered security and firewall mechanisms to the SoC.

2 FUNCTIONAL SAFETY FEATURES

2.1 NETSPEED ARCHITECTURE REFRESHER

Figure 1 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.

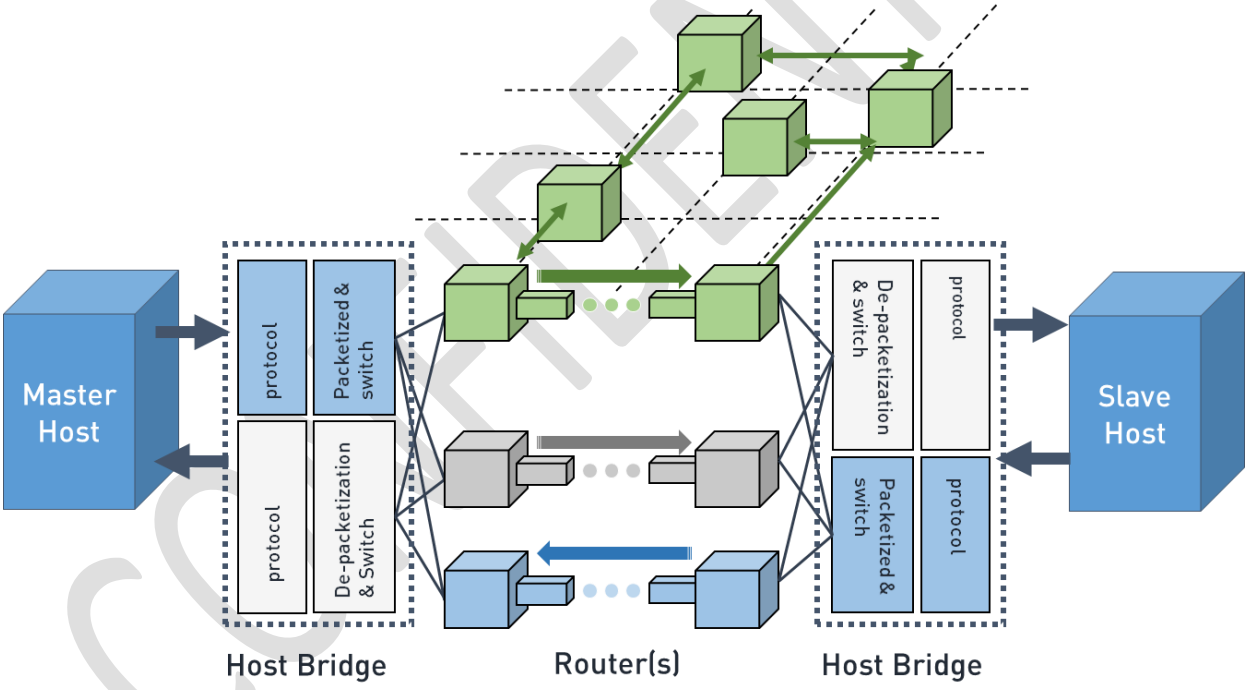

Figure 1 - NetSpeed NoC Architecture

2.2 SUMMARY OF FUNCTIONAL SAFETY, RELIABILITY AND AVAILABILITY

Figure 2 below summarizes the functional safety features provided by NetSpeed across the different interconnect components

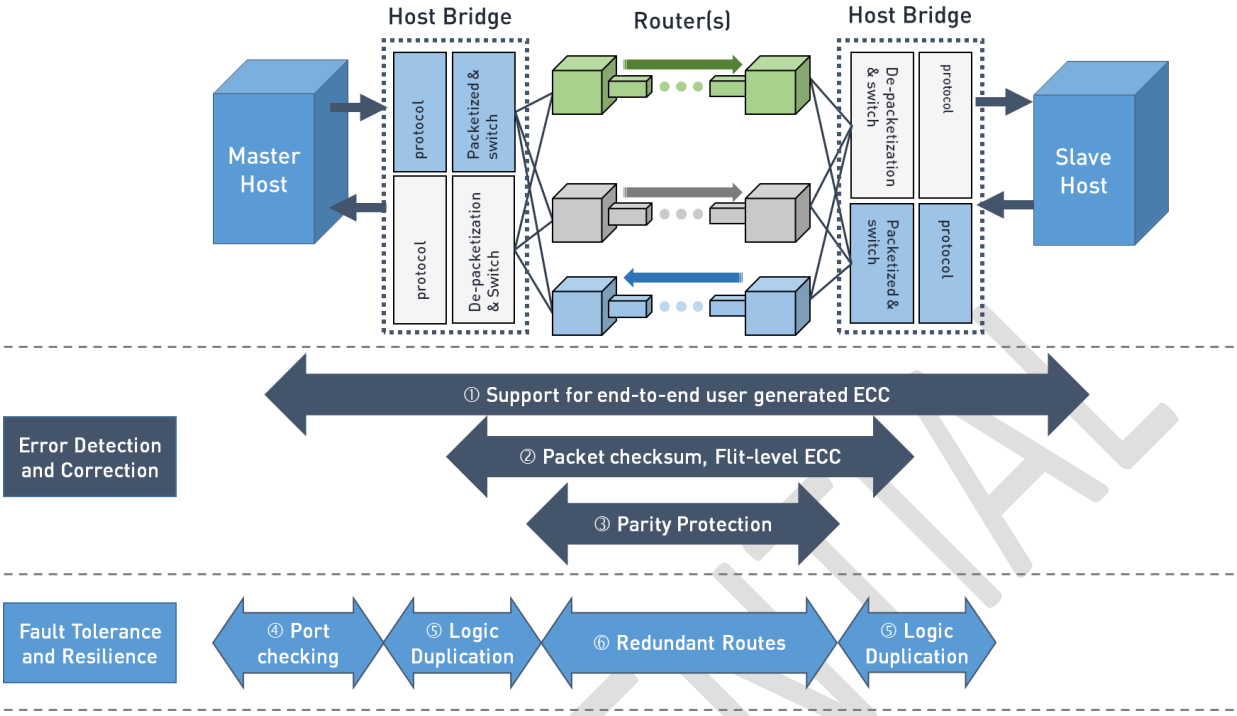

Figure 2 - NetSpeed Safety and Resilience Feature Support

2.3 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.

2.3.1 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 dectection. 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.

2.4 NoC-transport Features

2.4.1 End-to-end transport integrity

Within the NetSpeed IP, 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 data width. Multiple ECC fields are present for wider links. Sideband signal are also be protected with ECC. 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.

2.4.2 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.

2.4.3 Packet Validity Checking

NetSpeed IP provides robust means of confirming integrity at the packet level to detect missing data or misrouted packets. 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.

2.5 Bridge-Level Features

2.5.1 End point protection

Once the transaction is framed into a packet, NetSpeed IP can verify correct transmission through the mechanism described in Section 2.4. 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.

2.6 NetSpeed Coherency IP Features

2.6.1 ECC for RAMs in Coherency IP

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.

2.7 NetSpeed NoC Registers

2.7.1 Registers Parity Checking

NetSpeed IP can be configured to enable parity on all NoC registers. If enabled, parity bits are stored with write and verified by SW on reads. Parity is generated at regbus master (see 3.1 for regbus master description) 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. For example: address table parity failure would force a DECERR response that terminates the transaction.

2.8 Host IP Block

2.8.1 Host Terminated ECC

NetSpeed IP can also be configured to pass host generated ECC in data and control packets using user-bit fields. The ECC information originates and terminates in the host logic.

2.8.2 Architectural Support for Redudancy

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 Figure 3 to compare AXI interfaces and confirms that they are lock-step equivalent.

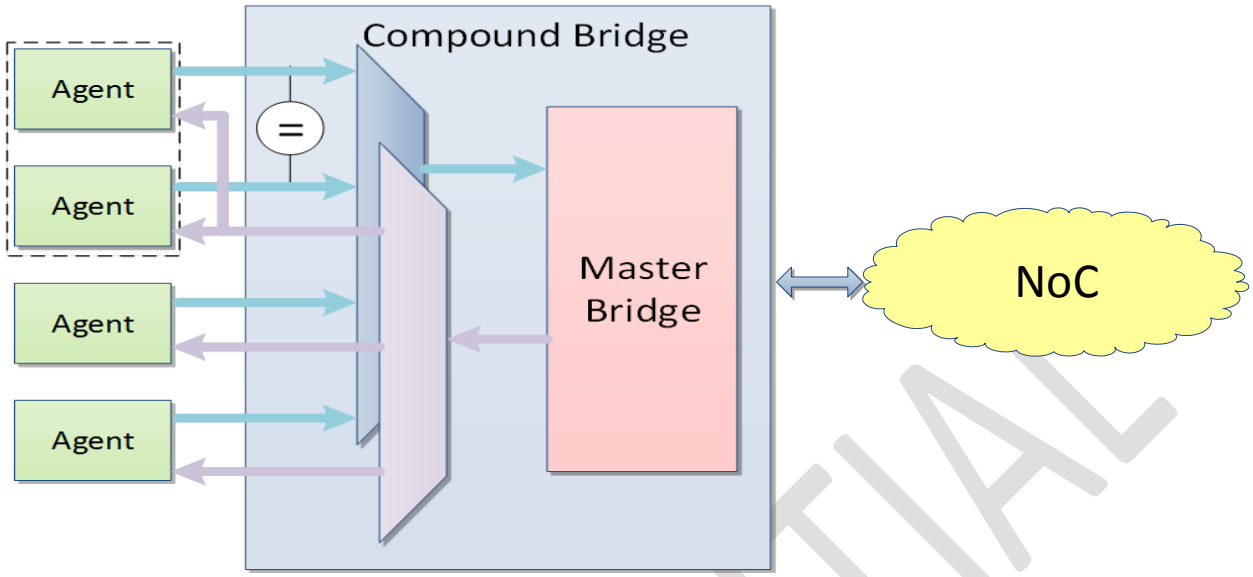

Figure 3 - Compound bridge to address redudant port checking

2.9 END-TO-END

2.9.1 Transactional Timeout

There are couple of configurable options of handling transaction timeouts in NetSpeed IP. 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. Maskable interrupt is also raised to the CPU with detailed syndrome of the timed-out request including address, AID, request type etc. On initiator side bridges, timeouts can be maintained for transactions outstanding on the NoC. These timeout allow detection of requests potentially dropped or stuck in the NoC. Timeout intervals are individually programmable and share timers for low cost implementation. Another layer of timeouts occur based on backpressure from the slave device for requests and master devices for response 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.

2.9.2 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.

2.9.3 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.

3 SERVICEABILITY FEATURES

NetSpeed bridges and routers support registers for address ranges, QoS weights, error logging, event counting, and interrupt generation and masking. These registers can be used to debug or configure the network and must be accessed by a privileged host, using an access layer that remains active even when the data layers are stalled. NocStudio provides the option of adding a *Regbus* layer that meets these requirements, accessed using a single Regbus master bridge.

3.1 REGBUS MASTER BRIDGE

The privileged master unit that manages the network must interact with the Regbus layer through the Regbus master bridge. A block diagram of the bridge is shown in Figure 4. The Regbus master bridge is a specialized version of an AXI bridge with the following restrictions:

- The AXI interface assumes a 32-bit master.
- AxLEN is restricted to 0 or 1 to allow either 32-bit or 64-bit register access.
- The NoC bridge address and router elements are determined and allocated by NocStudio. These are not user modifiable.
- The register-bus master bridge can be configured to have up to 16 outstanding read requests and 16 outstanding write requests.

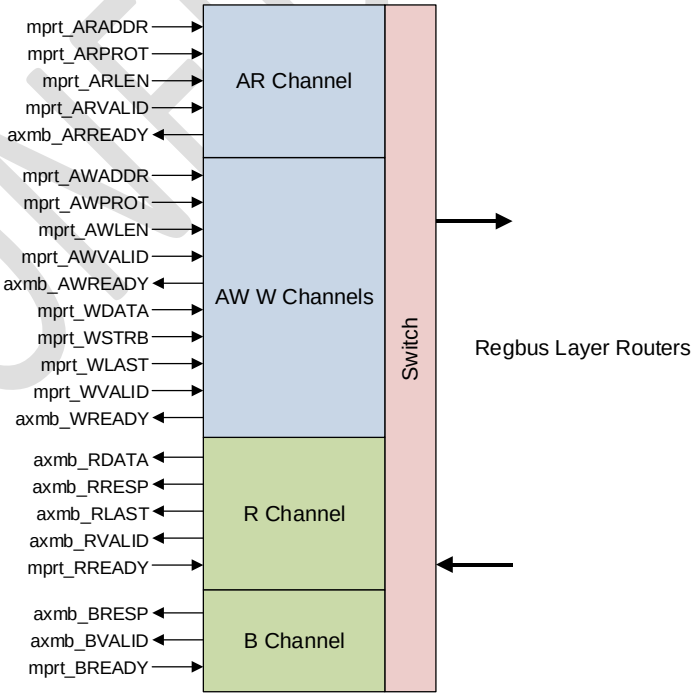

Figure 4. Regbus Master Bridge

3.2 THE REGBUS LAYER

As shown in Figure 5, the Regbus layer is physically separate from other NoC layers. It is implemented using NetSpeed routers and uses the same topology as the other layers. At each grid point or node in the multilayer NoC, a *RingMaster* unit is connected to a Regbus layer router. All configurable registers in every bridge or router at that node are accessible through a ring interconnect from the RingMaster. It is assumed that traffic on this layer is low bandwidth. By default, NocStudio attempts to minimize the Regbus cost by sizing data widths to 32-bit or lower. NocStudio allows minimal user intervention during build of this network.

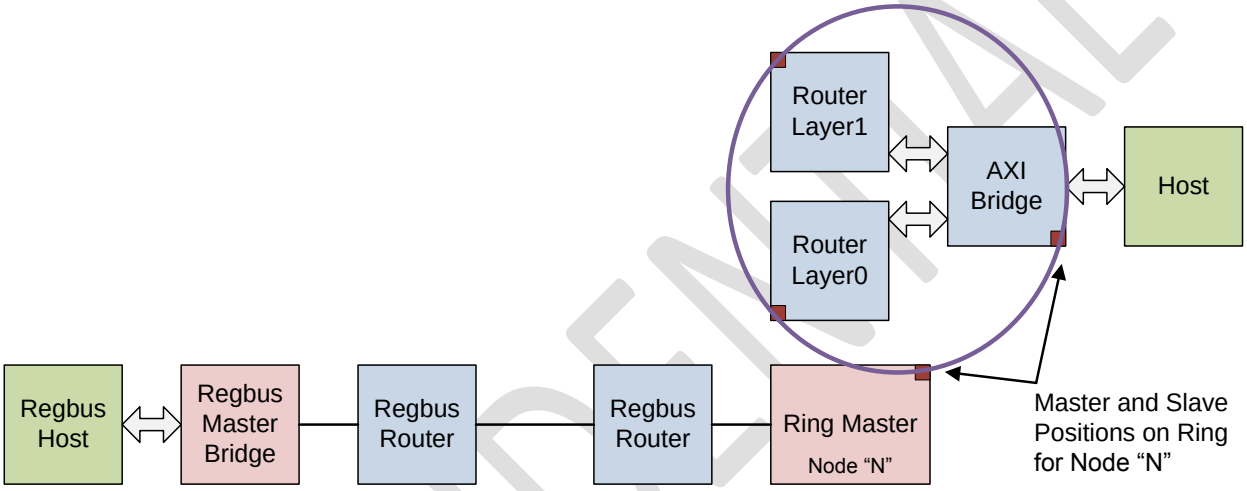

Figure 5. Regbus Layer Communication

To conserve power, the Regbus layer may run at a different frequency than the other NoC layers. The Ringmaster implements asynchronous clock-domain crossing, and each ring runs at the same frequency as the NoC elements on the primary NoC layers. Thus, Regbus should be power efficient and timing should be easy to close.

NocStudio assigns all NoC elements to a contiguous address space and programs the addresses into the Regbus master-bridge address tables. The addresses are not user modifiable.

3.2.1 Connecting to a Regbus Master over the Primary NoC

Occasionally, a host on the Primary NoC layer (such as a CPU) might need to configure another NoC host, access its internal registers to monitor status, or collect information for performance and debug. The CPU might not have an additional port to connect to the Regbus master bridge. To handle this, the NoC architecture provides a NetSpeed *tunnel block* that acts as a slave on the primary NoC layers and as a master to the Regbus master bridge.

Figure 6 shows how a CPU that is a primary NoC layer host connects to the Regbus master bridge. This provides connectivity to the Regbus layer, and access to NoC internal registers and host registers through configuration ports on the Regbus ring.

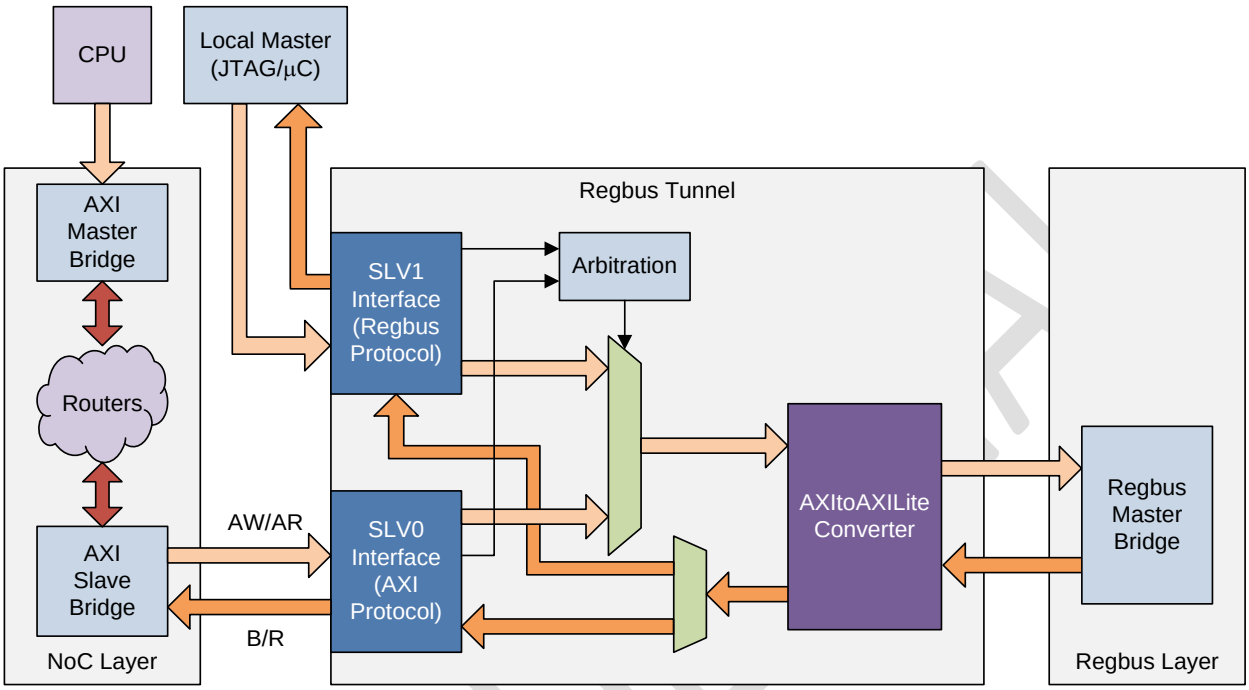

Figure 6. Regbus Tunnel Connects Primary NoC Layer to Regbus Layer

Traffic flows must be set up between one or more privileged masters on the NoC and the tunnel slave bridge. The tunnel address range can be a contiguous space covering all host configuration-register spaces and can have secure access attributes defined through NocStudio. This allows only privileged code running on the CPU to access this secure space. Host configuration-register space is defined on host configuration bridges and is mapped to the Regbus master bridge by NocStudio.

An additional port is provided on the tunnel unit for other masters, such as a JTAG or boot controller. Arbitration between the ports is done within the tunnel.

3.3 CONFIGURING THE REGBUS

The NetSpeed NoC Register Bus provides access to the registers of the NoC elements. In addition to NetSpeed's own registers, we provide the feature of providing register bus access to a user's own host registers. This access is made via the Register Bus Master (or through a host via the Tunnel). The Register Bus Master packetizes the access onto the register bus layer, to the specified host.

There are four interfaces available to connect the host's registers: APB, AHB lite, AXI4 lite and a NetSpeed Native Register interface. To add a user's host regbus connection:

```
add_host h1 bridge r1 rb_axi4l
```

Note here the bridge type is rb_axi4l, for Register Bus AXI4 lite. The bridge_type can be chosen to be any one of:

rb_axi4l – Register Bus AXI4 Lite Protocol
rb_ahbl – Register Bus AHB Lite Protocol
rb_apb – Register Bus APB Protocol
rb_native – Register Bus Native Protocol

All of the above bridge types support 32-bit and 64-bit data widths, except for rb_apb, which supports only 32-bit data width. Narrow accesses are not allowed on these interfaces, and will result in decode errors.

To allocate a register address range to the host, the following command can be used:

```
add_range h1/r1 range_r1 0x100-0x1ff
```

The address range can be specified as low address-high address or base:mask.

The above two commands are sufficient to have an AXI4 lite interface for user register port h1/r1. If there is no address compaction, and no specification of noc_register_base, the default user register space available is from 0x0 to 0x7fff_ffff (NoC register space begins at 0x8000_0000).

If address compaction is enabled, the user register address space will remain the same as that specified in the add_range command. The mesh_prop noc_register_base is used to signal the

beginning of NoC register space. The noc register base must be specified on an 8KB boundary. User space may hence be above or below the noc register space.

```
mesh_prop noc_register_base 0x0010_0000
```

Note that the entire register address space must be within a 4GB window, aligned at 4GB.

3.4 USAGE WITH TUNNEL

When accessing the register bus via the Tunnel, the tunnel range comes into play. Example:

```
add_range rbm/s rbm_s_tunnel_range 0x1_0000_0000:0xfff_ffff_0000_0000
        programmable 0
```

Address compaction, if enabled, will begin at the low address.

It is up to the user to size the tunnel range, and adjust the noc_register_base so that the tunnel range encompasses the entire user register space plus the NoC register space.

4 SECURITY FEATURES

NetSpeed IP is cognizant of multiple security risks at the SoC level and provides architectural security support at these levels. NetSpeed IP provides several security and firewall mechanisms in the NoC. These mechanisms filter selected traffic to specific target-address regions to prevent access to secure data by non-secure traffic. Figure 7 below summarizes NetSpeed's multi-tiered approach to security.

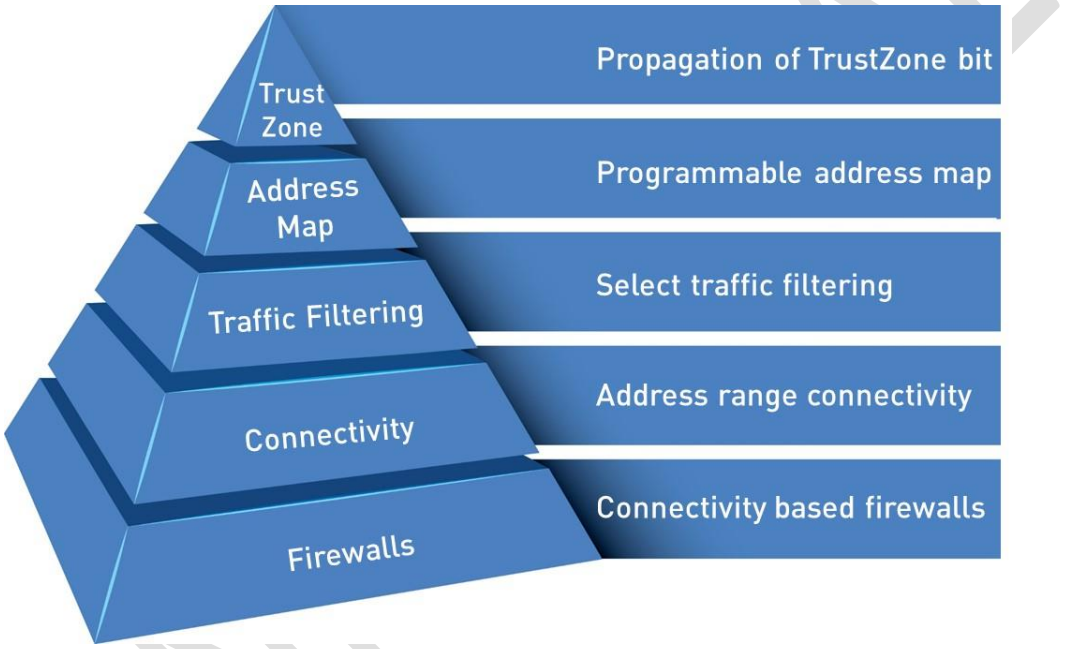

Figure 7 - NetSpeed's Multi-Tiered Security SupportWeighted QoS Mechanism

4.1 CONNECTIVITY-BASED FIREWALLS

When NocStudio generates a NoC, it creates routing paths between hosts using the traffic descriptions listed in the NocStudio configuration. The NoC is built minimizing connectivity, so if traffic wasn't listed between two hosts, there won't be a connection between those hosts. Even if they reside in a common network, the hosts are not logically connected and traffic cannot be sent between them. This can be used to prevent selected masters from communicating with selected slaves. Any request from a master that targets an unconnected slave causes a decode error in the bridge connected to the master. The request will not be sent to the slave.

Because connectivity is created during NoC construction, this acts as a static and permanent firewall. Each master can be individually configured for connectivity.

This security option can be controlled using the **add_traffic** command in NocStudio.

4.2 Address Range Connectivity

During NoC construction, a slave address range can be divided into multiple parts. Each address range can have different security control. A master can be set up to communicate to a subset of a slave's address range. For example, a device can have two address ranges, one for secure traffic and one for non-secure traffic. Non-secure hosts can be configured to access the non-secure address range and prevented from accessing the secure address range. This capability provides finer control over connectivity. Connectivity can be controlled on a per-address range basis, rather than a per-slave basis.

In NocStudio, master address-range assignment is done using **add_range_to_master**. If this is not used, address-range assignment occurs automatically based on traffic from the master to the slave.

4.3 Propagation of TrustZone Bit

The standard security feature in AXI-based interconnects is filtering based on packet protection bits. AXI networks support a 3-bit protection field as part of the read-request and write-request packets. The ARPROT[2:0] and AWPROT[2:0] fields can be used to propagate security information from the master to the destination. If the slave supports TrustZone filtering, or if TrustZone controller IP is instantiated before the slave, filtering can be handled outside of the NoC.

In coherent systems, the AxPROT[1] bit is used as an additional address bit for coherent requests. This ensures that coherency IP treats non-secure and secure accesses to an address as if they were different addresses, preventing an access to one address from accessing the data in the other.

These mechanisms are supported in NetSpeed IP.

4.4 Selective Traffic Filtering

Instead of blocking all requests from a master to a slave, it can be useful to selectively filter traffic from the master to the slave. An example is a CPU that can send both secure and non-secure traffic. The hardware must allow request filtering on a per-address range using the security bit, AxPROT[1].

NetSpeed allows selective request filtering. Four bits of packet information are available for filtering. AxPROT[2:0] bits can be used for filtering and specifying read versus write. For each of these, the NoC can be configured to only allow traffic to be sent when one or more of these bits

matches a predefined value. Each filter bit has an enable bit indicating the filter bit is included in the security lookup, and a polarity bit indicating which logical value is required to pass the security check. For example, AxPROT[1] can be enabled and the polarity set to zero so that only secure accesses (AxPROT[1]=1b1) are allowed. All of the security bits can be used concurrently if desired.

Selective traffic filtering can be controlled per-address range and per-master. The per-address range control allows a slave to have multiple address ranges with different security options. A slave that has a secure and non-secure address range can allow hosts to access either range. Only secure traffic can access the secure range. The non-secure range can be configured to permit access by both types of traffic or only non-secure traffic.

Each master connected to the NoC has individually-controlled security options. Different masters can have different security requirements for the same address range. A secure slave can permit all traffic from some secure hosts and only selected traffic from other hosts.

A request filtered through this mechanism causes a decode error. This is because the checks are done as part of the address-map lookup. If the traffic does not match the security requirements, the address lookup fails and functions as if the address range is not mapped to a target. The request stops in the bridge connected to the master port and is not transferred to the target slave.

Security options can be added during NoC construction using NocStudio. The **add_range** and **add_range_to_master** options can be used. The **add_range** option can create a default security option for all masters sending traffic to that address range. The **add_range_to_master** option provides a per-master security control.

4.5 PROGRAMMABLE ADDRESS MAP

NetSpeed IP supports programmable address ranges, including per-address range security-control features. This enables multiple additional security options.

The programmable address registers reside in bridges connected to masters. Because each bridge has its own registers, they can be individually controlled.

4.5.1 Disabling Address Ranges

Each programmable address range has a control bit that can disable an address range. This can be used to dynamically isolate a slave address range from a master, preventing requests between them. This functions similarly to fabric-connectivity filtering, but can be enable or disabled as needed.

4.5.2 Changing the Address Map

Because the address map is programmable, address ranges used by security features can be changed. A slave with secure and non-secure regions can be modified to change the region sizes or locations. This can be combined with address range enable/disable to change the number of security ranges.

NOTE: Address range registers are associated with a specific target slave. Although the ranges can change, the range target is unchanged. An unused range register cannot target a different slave.

NOTE: Programmable address registers are specified during construction. Any additional registers must be included at that time. Those registers can be initially disabled, if desired.

4.5.3 Modifying Traffic Filtering

The selective traffic-filtering controls are part of the programmable address registers. Thus, security filtering can be dynamically enabled or disabled. For example, during construction a slave address range can be specified as accessible by all hosts, and software can later modify the security requirements.

4.6 INTERFACE OVERRIDES

Because AxPROT bits are passed through the network to the destination, the system designer might need to override the bits coming from the interface. This can be true if an IP component cannot be trusted to set the protection bits correctly. For example, if AxPROT is set to indicate TrustZone secure, the host would have access to secure data.

An interface can optionally override the AxPROT[2:0] bits to ensure the system designer has full controllability. This provides control over the AxPROT bits that are transmitted within the network. If the override is set to non-secure, all requests from that host are tagged as non-secure. The override can be specified during NoC construction.

4.7 USER BITS

NetSpeed supports the propagation of user bits (AxUSER) within the network. This can be used to pass additional information with read and write packets, such as security options not supported natively in NetSpeed IP.

As an example, user bits with additional security information can be sent through the NoC to the target destination. The target destination can then reject requests based on the user-bit content.

2670 Seely Ave
Building 11
San Jose, CA 95134
(408) 914-6962
http://www.netspeedsystems.com