NetSpeed Gemini Register Bus Protocol

Version: GEMINI-15.10

October 5, 2015

NetSpeed Register Bus Protocol

About This Document

This document describes the NetSpeed register bus architecture and protocol. Using NocStudio, users can define NoC architectures, describe specifications and requirements, optimize the NoC design and generate a register bus for the specified NoCs that can access NoC registers as well as customer registers.

Audience

This document is intended for users of NocStudio:

* NoC Architects
* NoC Designers
* SoC Architects

Related Documents

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

* NetSpeed NocStudio User Manual

Related Documents

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

* NetSpeed NocStudio Gemini User Manual
* NetSpeed Gemini Physical Design Guidelines

Customer Support

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

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

Contents

[About This Document 2](#_Toc416806464)

[Audience 2](#_Toc416806465)

[Related Documents 2](#_Toc416806466)

[Related Documents 2](#_Toc416806467)

[Customer Support 2](#_Toc416806468)

[1 Introduction 6](#_Toc416806469)

[2 Register Bus using NocStudio 8](#_Toc416806470)

[3 Area Versus Latency 12](#_Toc416806471)

[4 NoC Registers 13](#_Toc416806472)

[5 Error Responses To Register Accesses 14](#_Toc416806473)

[6 User Register Bus Access 15](#_Toc416806474)

[7 Register Bus Master Interface 16](#_Toc416806475)

[8 Expected Usage of Register Bus Master 20](#_Toc416806476)

[9 Ring Slave to Host Interface 21](#_Toc416806477)

[10 Atomic Operations 22](#_Toc416806478)

[11 Clock Gating 26](#_Toc416806479)

List of Figures

[Figure 1: Regbus Layer Communication 6](#_Toc416806480)

[Figure 2: Regbus Layer Selection in NocStudio 8](#_Toc416806481)

[Figure 3: Register bus master bridge 16](#_Toc416806482)

[Figure 4: Waveform showing read requests and responses at the register bus master interface 19](#_Toc416806483)

[Figure 5: Waveform showing write requests and responses at the register bus master interface 19](#_Toc416806484)

[Figure 6 : Waveform showing ring slave read requests and responses (4B and 8B) 24](#_Toc416806485)

[Figure 7 : Waveform showing ring slave write requests and responses (4B and 8B) 24](#_Toc416806486)

List of Tables

[Table 1: Register attribute table 13](#_Toc416806491)

[Table 2: Register bit attribute table 13](#_Toc416806492)

[Table 3: Response table for NoC Register Accesses 14](#_Toc416806493)

[Table 4: Response table for User Register Bus Accesses 15](#_Toc416806494)

[Table 5: Register Bus Master Interface signals 17](#_Toc416806495)

[Table 6: Register slave to host interface 22](#_Toc416806496)

# Introduction

The NetSpeed register layer is an additional layer that can be visualized as the topmost layer of all the regular NoC traffic layers. NocStudio provides the option of adding this ‘regbus’ layer, which has a single register bus master bridge. Register bus transactions enter the register bus layer through the register bus master bridge. Each node on the register bus layer has a ring to vertically connect all the devices on the lower layers at that specific node. The devices on the ring may be bridges, routers, or hosts themselves. Each ring is mastered by a device called the ring master, and each device on the ring instances a ring slave, which then interfaces to the actual registers within the device.

The register bus layer uses a 32-bit address space (4GB of address space). The bottom 2GB is customer address space. The customer may map any number of devices in this space.

Reg bus Master Bridge

Reg bus Router

Reg bus Router

Ring Master Node “N”

Router Layer0

Str Bridge

Router Layer1

Host

Reg bus Host

Master and slave positions on Ring for Node ‘N’

Ring Slave to Host Interface

Register Bus Master Interface

Figure 1: Regbus Layer Communication

The Regbus layer is physically distinct from the other NoC layers. It is also implemented using Netspeed routers and uses the same topology as the other layers. At each grid point or node in a multilayer NoC, a Ring master unit is instantiated connected with a regbus layer router. All configurable registers in every bridge or router at that node is accessible through a ring interconnect from the ring master. The regbus interconnect thus takes the form of a NoC, with a ring master at every node where a data layer noc element exists. At each such node, a ring interconnect exists that communicates with all the noc element registers. It is assumed that traffic on this layer is low bandwidth and Nocstudio only allows minimal user intervention during the building of this network. It by default tries to minimize the cost of regbus by sizing the data widths to 32b or lower.

To conserve power, the Regbus layer runs at a frequency different from that of the rest of the NoC layers. The Regbus Master bridge, regbus routers and ringmaster all run at a common frequency. The Ringmaster implements asynchronous clock domain crossing, and the ring runs at the same frequency as the NoC elements on the Primary layers of the NoC. Regbus, should thus be a power efficient but easily timing closed portion of the NoC.

# Register Bus using NocStudio

1. Adding the register bus to the NoC

A regbus layer is added to the NoC through the new\_mesh command.

new\_mesh 4 4 2 test\_project

To enable Regbus, add “regbus\_enabled” after <project\_name>.

new\_mesh 4 4 2 test\_project regbus\_enabled

1. Visual Representation of the Regbus layer in the NocStudio GUI:

The regbus layer is represented by a ‘R’ in the layer selection buttons.

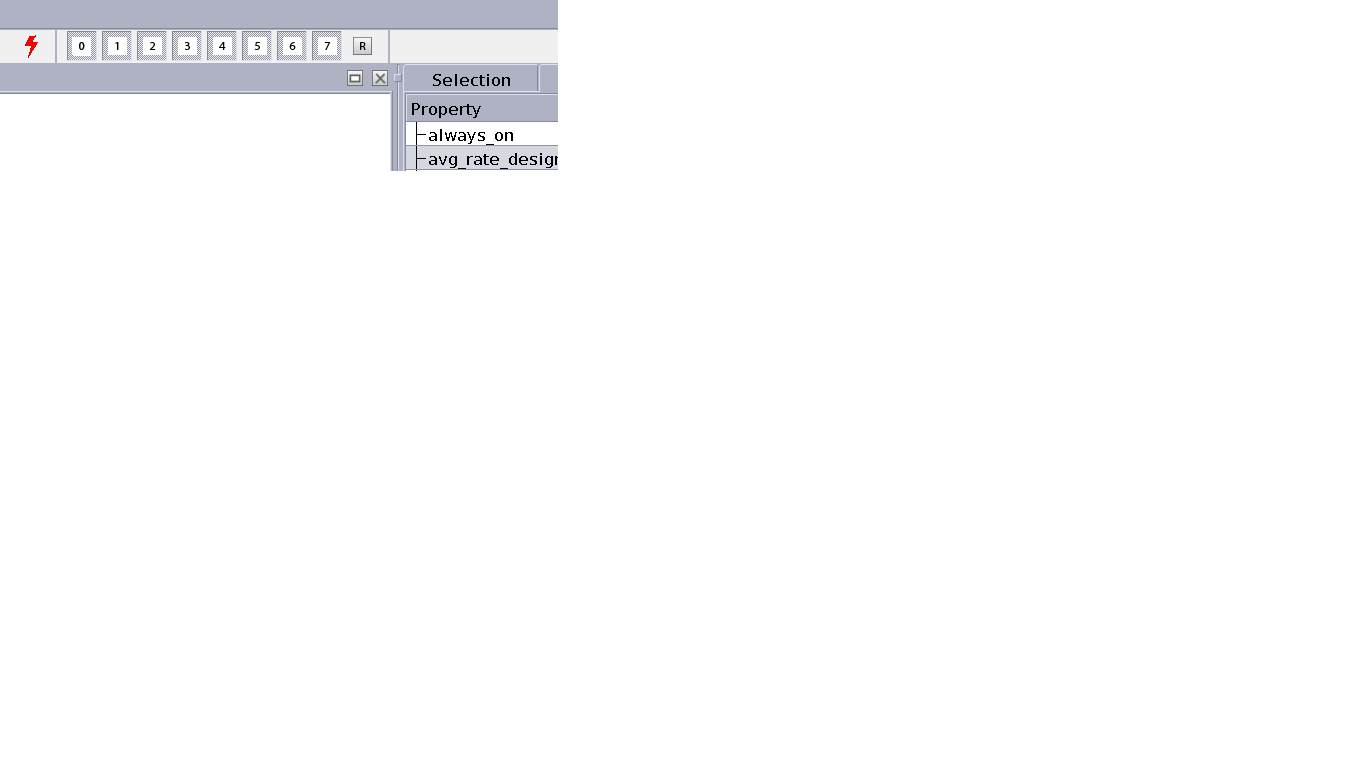


Figure 2: Regbus Layer Selection in NocStudio

1. Adding customer registers:

Customer registers can be added using the add\_host command, and adding a port to a host of the type ‘regbus’. The current release of NocStudio does not support customer registers over the NetSpeed regbus.

add\_host h0 color blue pos 5 bridge t stream bridge r regbus

The customer register address range can be assigned anywhere within the 0 to 2GB range, when addresses are not compressed (See item 7 for compressed addresses).

add\_range h0/r h0\_regbus 0x0-0xffff

1. Enabling downsizing on the regbus rings:

The mesh\_prop ‘regbus\_ring\_width’ can be used to change the width of the regbus ring. Supported widths are 9, 18 and 36, to be able to support integral multiples of an 8-bit data beat accompanied with a 1-bit byte enable. The default width of the ring, when no mesh\_prop is specified, is 9 bits.

This width applies to the link between the regbus layer router and the ring master, as well as the actual ring itself. Reducing this width saves wire routing when the ring master is farther away from the regbus router. It also saves area within the ring master. Increasing this width reduces regbus layer latency.

mesh\_prop regbus\_ring\_width 18

regbus\_ring\_width <bits>: The width of regbus rings

*Allowed values: 9, 18, 36*

Regbus rings can have width 9, 18 or 36; narrower rings have less throughput but less area cost. This property also affects the width of the connection from router to ring master.

1. Enabling address compression:

The default regbus address policy of NocStudio implements a “Node based” address map of NoC elements, with space reserved for the maximum number of nodes (256), the maximum number of layers (8), with a maximum of noc elements per node (32), each with an 8kB space. While this has some nice properties for decoding simplicity, the NoC space allocated is fixed to 256 nodes x 256kB = 64MB, which when future proofed to 8192 nodes expands to 2GB.

A different scheme for Regbus address space is available via the mesh\_prop ‘compact\_regbus\_address\_space’. This scheme moves away from node based addressing, to a more traditional addressing scheme. A mesh\_prop ‘noc\_register\_base’ is available to decide the start of the NoC register addresses. Salient features:

* NoC space is now compressed, and not sparse, resulting in a smaller reserved footprint.
* User control is available to reserve the address range to be allocated for NoC elements, and NocStudio will operate within that space when possible, and indicate an error if the range is insufficient.
* NoC elements are all present in a contiguous address block
* NoC elements may be bridges and routers, and also other NetSpeed agents such as Coherency IP blocks, and the Tunnel block that will allow Regbus access from primary layer host bridges. Other such NetSpeed IP blocks may be added in the future.

mesh\_prop compact\_regbus\_address\_space yes

If true, regbus addresses for NoC will be compacted, default is node/ring-based addressing. Default: enabled.

*Allowed values: yes, no*

Noc elements registers accessible through regbus address space can be compact or spread. Spread addressing requires fewer resources in the regbus master bridge, allowing it to work at higher frequency, but takes up much more address space. Compact addressing of noc elements reduces the address space needed for the NoC, The maximum number of chain per each transaction that will be displayed whenever traffic transactions are expanded.

mesh\_prop noc\_register\_base 0x0

noc\_register\_base <base address>: The base address of noc registers within regbus address map.

*Allowed values: custom*

When using compact regbus address space, the noc registers can be at an arbitrary offset within the regbus space. This property sets that offset. Default is 2GB.

1. Changing regbus layer router fifo depths:

NocStudio can control the depth of the fifos within the routers in the regbus layer. This helps with tuning area versus latency of the regbus.

prop\_default regbus\_vc\_fifo\_depth 1

This reduces the depths of the fifos within the routers in the regbus layer to 1, which is provides the maximum area reduction in terms of fifo depth.

1. Reducing regbus fifo\_depth individually:

There ingress fifo of a router on the regbus layer can be individually controlled, specified by the “in” link. Similarly, the ingress fifo of a ring master can be individually controlled, specified by the “out” link of the router it is connected to.

vc\_prop 7/44h.1.in fifo\_depth 1

vc\_prop 7/44h.1.out fifo\_depth 2

fifo\_depth <# flits>: [TX/Async] Fifo depth for the tx stream interfaces and for the async interfaces.

*Allowed values: custom*

The FIFO depth at the streaming bridge tx interfaces and all bridge async interfaces. For streaming bridge tx interfaces, FIFO is needed for the flow credit flow control, and for all async rx and tx interfaces of all bridges, FIFO is needed for needed for async clock crossing; this prop indicates the depth of the FIFO.

# Area Versus Latency

In the previous section, some regbus properties have been outlined. Area optimization on the regbus can be performed using those properties. For example:

* Regbus ring width can be set to the lowest of 9 bits (mesh\_prop regbus\_ring\_width 9).
* The regbus router layer can be reduced in size by reducing the depths of the fifos to 1 (prop\_default regbus\_vc\_fifo\_depth 1).

The effect of reducing area is an increase in latency. A regbus ring width of 9 means that it has four times as many data beats to send as compared to a regbus with ring width of 36, and twice as many data beats to send as compared to a regbus with ring width of 18.

The regbus router layer fifo depth, when reduced, means a smaller number of credits available at hand, and hence an increase in latency for a packet to make its round-trip journey.

The time taken by regbus sanity test provided as part of the NetSpeed release can be controlled by these properties. Increasing ring width and increasing regbus\_vc\_fifo\_depth will reduce latency, thereby reducing the time duration of the regbus sanity test.

# NoC Registers

NoC registers are automatically created by NocStudio and placed in a fixed register bus address map. This address map is unrelated to any address map within the main NoC design.

For details of the registers and register address map, refer to noc\_reference\_manual.html and noc\_registers.csv (which only appears if register bus is enabled) generated by NocStudio in the project directory.

Registers can be 32-bit wide, or 64-bit wide. Register sizes are indicated by the width of their reset values inside noc\_registers.csv (or noc\_reference\_manual.html). Within noc\_registers.csv, the following register attribute nomenclature is followed.

Table 1: Register attribute table

|  |  |
| --- | --- |
| **Register attribute** | **Description** |
| rw | Read-Write register. All bits in this register are writable (except for u, A, B) |
| r | Read-only register. All bits in this register are read-only, and cannot be written to. These are usually status registers |
| wzc | Write-zero-to-clear register. This register contains fields that must be written with zeroes to clear. These are usually error registers |

Each individual bit inside a register has fine-grained bit attributes. Reset values of the registers are concatenations of each of these bit attributes in bit order.

Table 2: Register bit attribute table

|  |  |
| --- | --- |
| **Register bit attribute** | **Description** |
| u | Unused. These bits have no associated flops and return 0 when read |
| r | Reserved. These bits are reserved for future expansion, and have associated flops. Flop reset value is 0 |
| A | Unwritable 0. These bits are part of a bigger field, but do not have associated flops to save area |
| B | Unwritable 1. These bits are part of a bigger field, but do have associated flops to save area |
| 0 | Reset value of 0. These bits have an associated flop |
| 1 | Reset value of 1. These bits have an associated flop |

# Error Responses To Register Accesses

NetSpeed NoC registers can be 32-bit wide or 64-bit wide. The register bus master allows 32-bit as well as 64-bit accesses to the register space. Some accesses may return errors due to decode failures. Below is a list of combinations and their expected error responses.

Table 3: Response table for NoC Register Accesses

|  |  |
| --- | --- |
| **Type of Access** | **Response** |
| 32-bit access to defined 32-bit register | Okay |
| 64-bit access to defined 64-bit register | Okay |
| 64-bit access to defined 32-bit register | Okay |
| 32-bit access to defined 64-bit register | Decode Error signaling access is not atomic |
| 32-bit access to non-existing register address | Decode Error (Accesses that have a register at its 64b aligned address will realign the access and return Okay) |
| 64-bit access to non-existing register address | Decode Error |

# User Register Bus Access

The NocStudio User Manual contains the description on how to add access for a user’s registers via the NetSpeed Register Bus. Please check your release version to see if this is supported for your release.

There are four protocols via which this can be done: AHB-lite, AXI4-lite, APB and a NetSpeed Native Register Protocol. Data width may be 32-bits or 64-bits wide. Narrow accesses are not supported on any of these interfaces. Responses to narrow accesses are returned as decode errors.

Table 4: Response table for User Register Bus Accesses

|  |  |
| --- | --- |
| **Type of Access** | **Response** |
| 32-bit access to 32-bit interface | Okay |
| 64-bit access to 64-bit interface | Okay |
| 64-bit access to 32-bit interface | Decode Error |
| 32-bit access to 64-bit interface | Decode Error |

# Register Bus Master Interface

The register master is the entry port into the register layer. This privileged master unit that manages the register bus network must interact with this layer through the Regbus master bridge. The Regbus master bridge is a specialized version of an AXI bridge.

* Interface on the AXI side assumes a 32b master.
* AxLEN restricted to 0,1 to allow either 32b or 64b register access
* Address of NoC bridge and router elements are decided and allocated by Nocstudio. These are not user modifiable.
* The register bus master bridge can be configured to have as many as 16 outstanding requests on reads and 16 on writes

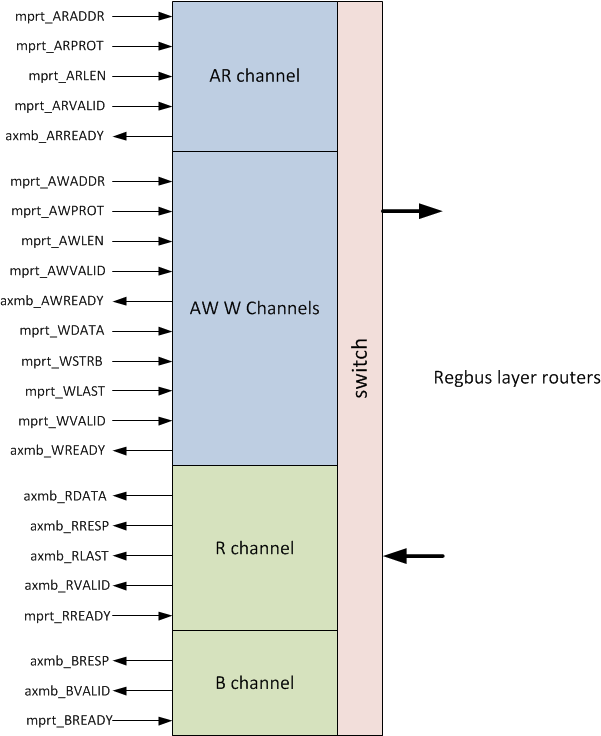


Figure 3: Register bus master bridge

The list of input signals is specified below:

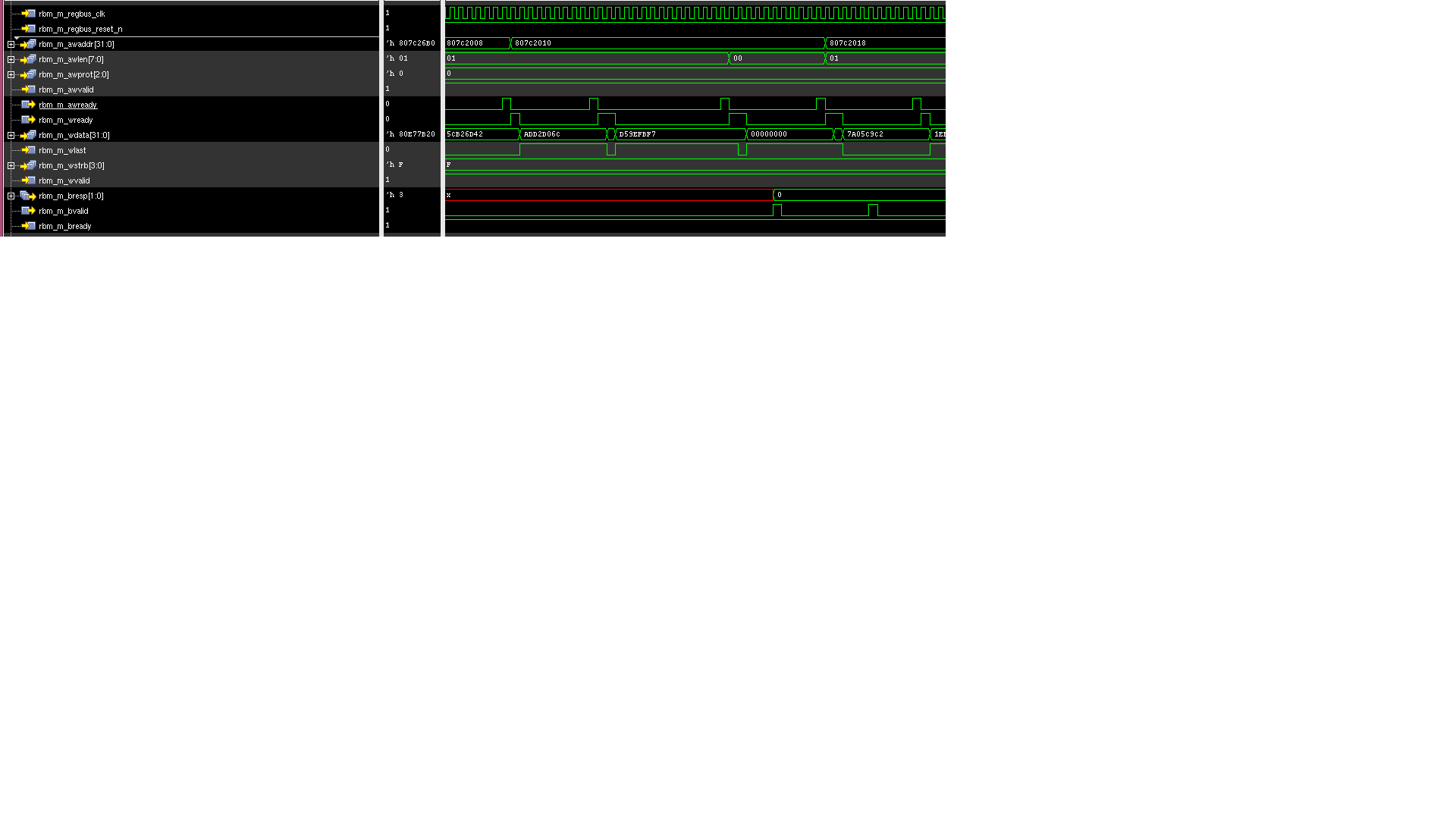
Table 5: Register Bus Master Interface signals

|  |  |  |  |
| --- | --- | --- | --- |
| Signals | Width (number of bits) | Usage | Description |
| Inputs |  |  |  |
| rbm\_m\_regbus\_clk | 1 | Mandatory | Register bus clock (may or may not be the same as the chosen noc clock) |
| rbm\_m\_regbus\_reset\_n | 1 | Mandatory | Active low reset |
| rbm\_m\_araddr | 32 | Mandatory | 32-bit register read address (Bit 31 set to 0 for non-NetSpeed registers) |
| rbm\_m\_arprot | 3 | Optional | Read protection bits |
| rbm\_m\_arvalid | 1 | Mandatory | Read valid signal |
| rbm\_m\_arlen | 1 | Mandatory | Read length. 0 indicates 32B read. 1 indicates 64B read |
| rbm\_m\_rready | 1 | Mandatory | Read response ready signal indicating acceptance of read response |
| rbm\_m\_awaddr | 32 | Mandatory | 32-bit register write address (Bit 31 set to 0 for non-NetSpeed registers) |
| rbm\_m\_awprot | 3 | Optional | Write protection bits |
| rbm\_m\_awvalid | 1 | Mandatory | Write valid signal |
| rbm\_m\_awlen | 1 | Mandatory | Write length. 0 indicates 32B read. 1 indicates 64B read |
| rbm\_m\_wdata | 32 | Mandatory | 32-bit Write data |
| rbm\_m\_wstrb | 4 | Mandatory | Write strobe or byte enables |
| rbm\_m\_wvalid | 1 | Mandatory | Write data valid signal |
| rbm\_m\_wlast | 1 | Mandatory | Indicates the last beat of data. Set on the first beat if 32B, set on second bit if 64B |
| rbm\_m\_bready | 1 | Mandatory | Write response ready signal indicating acceptance of write response |
|  |  |  |  |
| Outputs |  |  |  |
| rbm\_m\_arready | 1 | Mandatory | Read ready signal indicating acceptance of read request |
| rbm\_m\_rdata | 32 | Mandatory | 32-bit response data |
| rbm\_m\_rresp | 2 | Mandatory | 2-bit read response. 2'b00-okay, 2'b11-decode error, 2'b10-slave error |
| rbm\_m\_rvalid | 1 | Mandatory | Read response valid signal |
| rbm\_m\_rlast | 1 | Mandatory | Indicates the last beat of data. Set on the first beat if 32B, set on second bit if 64B |
| rbm\_m\_awready | 1 | Mandatory | Write command ready signal indicating acceptance of write request |
|  |  |  |  |
| rbm\_m\_wready | 1 | Mandatory | Write data ready signal indicating acceptance of write data |
| rbm\_m\_bresp | 2 | Mandatory | 2-bit read response. 2'b00-okay, 2'b11-decode error, 2'b10-slave error |
| rbm\_m\_bvalid | 1 | Mandatory | Write response valid signal |

Figure 4: Waveform showing read requests and responses at the register bus master interface



Figure 5: Waveform showing write requests and responses at the register bus master interface



# Expected Usage of Register Bus Master

The NetSpeed Bridges and Routers support registers for QoS weights, error logging, event counting, and interrupt generation and masking. As these registers can be used to debug the state of the network, they must be accessed by a privileged host, and by an access layer that remains alive even if the data layers are stalled. Host registers connected to the regbus layer are also extended the advantage of debug through the regbus layer if the data layers are stalled.

The privileged host, or the ‘Register Bus Master’, can be part of a larger agent that handles configuration, power, reset and debug. It may also have a port on the data layers of the NoC through which it is controlled by CPUs so that the CPUs can access the regbus layer indirectly.

# Ring Slave to Host Interface

On the ring slave to host interface, a combined read/write bus is used. The interface is very similar to an AXI-lite interface. It follows the same flow control ready/valid protocol. This interface runs on the chosen NoC clock. It also has an active high reset.

Rules:

* If more than one request is permitted to be outstanding to the host, the host must return the responses to the ring slave in order. Read responses must be returned in order with respect to each other. Similarly, write responses must be returned in order with respect to each other. Read response ordering with respect to write responses (or vice versa) is not expected. Read and write responses may come back out of order with respect to each other, as long as they are ordered within their respective channels.
* The address requested on the bus is the lowest address being requested. For example, a 32-bit or 4B write request to an address 0x40 indicates that the write is meant for byte offsets 0x43, 0x42, 0x41, 0x40.
* Flow control by means of a ready signal is present on this interface. The valid signal, if asserted, must remain asserted until it receives a ready. All fields on the interface must also remain unchanged until the ready has been received. There are two sets of valid/ready signals: req\_valid/req\_ready, rsp\_valid/rsp\_ready.
* A ring slave can be allowed to have multiple outstanding requests to the host indicated by the programmable parameter P\_REGBUS\_RSLV\_NUM\_OUTSTANDING.

# Atomic Operations

On the ring slave to host interface, each request and response is transferred in a single cycle. Whether a write is a 32-bit write or a 64-bit write, all bits of write data are presented on the interface at the same time. The same is true for read response data. The single cycle transfer makes all transactions on this interface inherently atomic.

Table 6: Register slave to host interface

|  |  |  |  |
| --- | --- | --- | --- |
| Signals | Width (number of bits) | Usage | Description |
| Inputs |  |  |  |
| Clk | 1 | Mandatory | Same as chosen noc clock |
| Reset | 1 | Mandatory | Active high reset |
| regslv\_rsp\_valid | 1 | Mandatory | When 1, indicates a valid response from the host |
| regslv\_rsp\_rnw | 1 | Mandatory | When 1, indicates a read response. When 0, indicates a write response |
| regslv\_rsp\_rdata | 32 or 64 (parameter) | Mandatory | The data is transferred in the same cycle as regslv\_rsp\_valid. If size=0, the least significant 32 bits are the ones returned to the regbus master |
| regslv\_rsp\_err | 2 | Mandatory | 2-bit. Indicates slave error when slave exists, but no register at the location specified. The slave is free to return a decode error instead of a slave error if it so chooses. (AMBA spec: 2’b10=Slave error (slave exists, but no register at the location specified). 2’b11=Decode error (no slave exists). Decode error will be returned by the ring master when it receives a request back from the ring that wasn’t accepted by any slave) |
| regslv\_req\_ready | 1 | Mandatory | When asserted at the same time as regslv\_req\_valid, indicates the acceptance of that request |
|  |  |  |  |
| Outputs |  |  |  |
| regslv\_req\_valid | 1 | Mandatory | When 1, indicates a valid request from ring slave to the host |
| regslv\_req\_addr | 31 or less (parameter) | Mandatory | Register read or write address |
| regslv\_req\_rnw | 1 | Mandatory | Read not Write. When regslv\_req\_valid=1, regslv\_req\_rnw=1, a read is being requested. When regslv\_req\_valid=1, regslv\_req\_rnw=0, a write is being requested |
| regslv\_req\_size | 1 | Mandatory | 0 indicates a 32-bit request. 1 indicates a 64-bit request |
| regslv\_req\_region | 4 | Optional | Passes along the address map sub-slave information for devices behind this device |
| regslv\_req\_prot | 3 | Optional | Passes along the 3-bit ARPROT/AWPROT field presented to the register bus master for this transaction |
| regslv\_req\_wdata | 32 or 64 (parameter) | Mandatory | The data is transferred in the same cycle as regslv\_req\_valid. P\_REGBUS\_RSLV\_DATA\_WIDTH can be 32-bit or 64-bit. If P\_REGBUS\_RSLV\_DATA\_WIDTH=64 and size=0, it indicates the least significant 32 bits should be accessed, that is, bits 31:0 |
| regslv\_req\_wstrb | 4 or 8 | Optional | Indicates the write strobes or byte enables for write data |
| regslv\_rsp\_ready | 1 | Mandatory | When asserted at the same time as regslv\_rsp\_valid, indicates the acceptance of that request |



Figure 6 : Waveform showing ring slave read requests and responses (4B and 8B)

Figure 3 shows examples of 4B and 8B read requests and their responses. In this example, read responses show decode errors. Write data (regslv\_req\_wdata) is don’t-care because these are read requests.



Figure 7 : Waveform showing ring slave write requests and responses (4B and 8B)

Figure 4 shows examples of 4B and 8B write requests and their responses. In this example, the write responses are decode errors. Read data (regslv\_req\_rdata) is don’t-care because these are write requests.

# Clock Gating

1. Ring-level clock gating

The ring master is responsible for waking up all the slaves on the ring by asserting its ring\_wakeup output. This is a signal on an always-on clock and goes around the ring, waking up all the slaves on the ring. The ring master asserts this signal on receiving a request, and de-asserts this signal when there are no outstanding requests on the ring.

1. Slave-level clock gating

When the first entry in the four-entry buffer of a ring slave is filled, it wakes up the slave. Packets are counted as they enter and leave. The clock enable is deasserted when all packets have left the bounds of that slave.

2670 Seely Avenue

Building 11

San Jose CA 95134

[www.netspeedsytems.com](file:///C:\Users\AE1\Downloads\www.netspeedsytems.com)