# NoC Serviceability: Regbus Layer

## The Register Bus

CFG 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 multiple *Regbus* layers that meets these requirements, accessed using a single Regbus master bridge for each layer.

### The Regbus Layer

As shown in Figure 26, a Regbus layer is physically separate from other NoC layers. It is implemented using CFG 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 ring interconnects from the RingMaster. 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 26. Regbus Layer Communication

By default, the Regbus master bridge, Regbus routers, and RingMaster all run at a common frequency. However, different clock domains can be defined on the regbus layer similar to normal NoC layers. The Ringmaster runs synchronously with the connected regbus router but can serve one or more rings each in its own clock domain. Each ring is in the same clock domain as the normal layer bridge or router elements it is serving.

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.

### Regbus Controller

From 20Q1 release, Regbus Controller (RBC) will replace the previously existing Regbus Tunnel module. It allows multiple requesters to get on the regbus layer to access registers in the NoC and user register space. RBC supports multiple ports with different protocols accessing the regbus layer, which are:

* AXI4S port for tunneling requests from the main NoC layer to the Regbus layer.
* APBM interface
* IOSF-SB interface
* IOSF-APB interface
* AXI4LM



### Regbus Master

The Regbus Controller has a new regbus master bridge on it called RBC-RBM, shown in the figure below. The RBC-RBM connects to the RBC on the host side and an RSSB module on the NoC side. The RBC-RBM cannot connect to the legacy non-RSSB router, or interface with the old Tunnel Host module. The RBC must contain exactly one RBM, but all the other ports such as the AXI4 tunnel slave, APBM, IOSFSB, etc are optional. Unlike the previous RBM, read and write requests are sent to the NoC through a single common Txswitch interface, so preserve read-write ordering set by RBC. Read and write responses are also received over separate Rxswitch interfaces. There can be multiple RBCs instantiated in the design, one associated with each regbus layer.



### 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 CFG *tunnel block* that acts as a slave on the primary NoC layers and as a master to the Regbus master bridge.

Figure 27 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 27. 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 and NoC 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. This port can be configured as a 32-bit AXI-Lite port or an APB port. Arbitration between the ports is done within the tunnel.

### Configuring the regbus

The CFG NoC Register Bus provides access to the registers of the NoC elements. In addition to CFG’s own registers, we provide the feature of providing register bus access to a user’s 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 CFG Native Register interface.

#### 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

The above command defines the system address space for registers which is accessible through tunnel. This encompasses both the user register space and the CFG NoC register space.

NoC address space can be allocated in two configurations. In compacted mode, the amount of address space taken up by NoC registers is lesser. Non-compacted mode consumes more address space but allows simpler decoding in the regbus master bridge.

Address space for user registers are assigned using add\_range command. For example, following command assigns a range to a user register port h1/reg1

add\_range h1/reg1 h1\_reg\_1 0x0000\_5000-0x0000\_50FF 0

Mesh property noc\_register\_base can be used to define the base address of NoC registers within regbus address map. By default, NocStudio assigns the address above the last user host register range to internal NoC registers. It is up to the user to size the tunnel range, and adjust the noc\_register\_base so that the tunnel range covers the entire user register space plus the NoC register space.



Figure 28: Regbus Address Map

## 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 3: 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 4: 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

CFG NoC registers can be 32-bit wide or 64-bit wide. All NoC registers are aligned to 64-bit addresses. Each NoC register also has a secure/non-secure attribute. 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 5: 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 | Okay. Each half of the 64-bit register can be accessed using 32-bit access |
| 32-bit access to non-existing register address | Decode Error |
| 64-bit access to non-existing register address | Decode Error |
| 64-bit access to an address which is aligned to 32-bits | Decode Error |
| Read access to secure register with AxPROT[1] = 1 | No read performed. 0 data and decode error response is returned |
| Write access to secure register with AxPROT[1] = 1 | No write performed. Decode error response is returned |
| Read/Write access to non-secure register with any AxPROT[1] | Okay |

## User Register Bus Access

The NocStudio User Manual contains the description on how to add access for a user’s registers via the CFG 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 CFG 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 6: 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

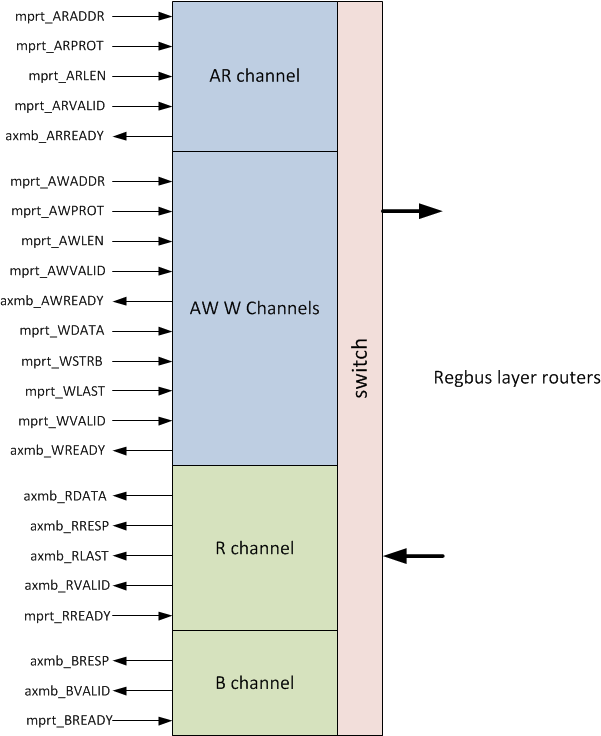


Figure 29: Register bus master bridge

The list of input signals is specified below:

Table 7: 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-CFG registers) |
| rbm\_m\_arprot | 3 | Mandatory | 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-CFG registers) |
| rbm\_m\_awprot | 3 | Mandatory | 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 30: Waveform showing read requests and responses at the register bus master interface

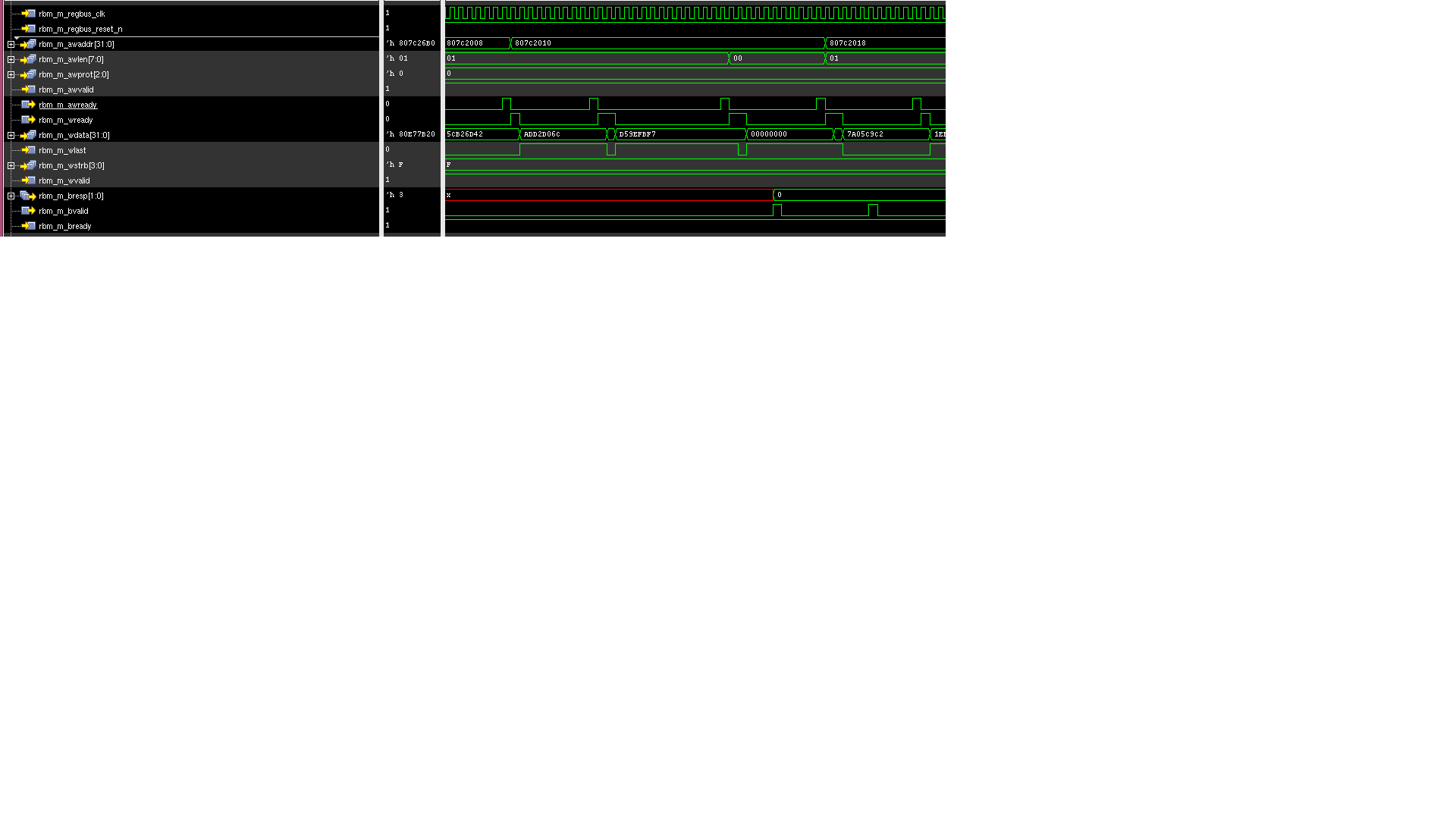


Figure 31: Waveform showing write requests and responses at the register bus master interface

## Expected Usage of Register Bus Master

The CFG 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 8: 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 32 : Waveform showing ring slave read requests and responses (4B and 8B)

Figure 32 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 33 : Waveform showing ring slave write requests and responses (4B and 8B)

Figure 33 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.

## Restrictions

* Regbus master bridge implements a customized AXI protocol with 32b data width, AxLEN of 0, 1 to support 32b, 64b register accesses over a 32b down-sizeable NoC layer interface.
* User reg bus, have the following restrictions:
* Transaction size needs to equal the interface size:
  + If you have a 32-bit rb native interface, 32-bit user reg bus accesses are supported
  + If you have a 64-bit rb native interface, 64-bit user reg bus accesses are supported. 32-bit accesses are not supported.
* Host errors returned from the rb native interface are not supported.
* Async interface at rb native is not supported.
* Write strobes (byte enables) are not supported. This is similar to what we support for the NOC internal registers.