New Register Bus Protocol and Microarchitecture

**Version 0.2**

[Register Bus Master to Ring Master Packet Format 2](#_Toc384655054)

[Ring Packet Format 5](#_Toc384655055)

[Timing Diagram 9](#_Toc384655056)

[High Bandwidth Example 10](#_Toc384655057)

[Microarchitecture 11](#_Toc384655058)

[Register Slave to Client Interface 12](#_Toc384655059)

[Parameters 14](#_Toc384655060)

[Clock Gating 15](#_Toc384655061)

[Clock Crossing 16](#_Toc384655062)

[Power Gating 17](#_Toc384655063)

# Register Bus Master to Ring Master Packet Format

The format of the packet sent between the Register Bus Master to the Ring Master on the node is shown below. The RD versus WR bit is at bit 8, sequence number is at bits 4:0 and prot bits are at bits 7:5. Additional information added to the packet is the region field to Flit 0 of the write and read packets. They occupy bits 18:15.

To accommodate 64-bit reads and writes, an additional data flit is added.

In the new register address map, for a NetSpeed register, the ring ID is bits 18:14 of the 32-bit address.

Bit 31 – Set to 1 to indicate NetSpeed space, not User space

Bits 30:19 – Node ID (26:19 are used)

Bits 18:14 – Ring ID (Specific 16 KB offset for a router, bridge, coherency, or NoC information within the node--32 such regions)

Bits 13:0 - Register offsets within the 16 KB space

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Flit 0 | Dummy | Region | Prot | SeqNum | Length (Command[0]) | Converter | Command[2:1] | Host Regbus | Ring ID |
|  | 35:22 | 21:18 | 17:15 | 14:10 | 9 | 8 | 7:6 | 5 | 4:0 |
|  |  |  |  |  |  |  |  |  |  |
| Flit 1 | Dummy | Address | | | | | | | |
|  | 35:32 | 31:0 | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Flit 2 | Wstrb[3] | Data[31:24] | Wstrb[2] | Data[23:16] | Wstrb[1] | Data[15:8] | Wstrb[0] | Data[7:0] |  |
|  | 35 | 34:27 | 26 | 25:18 | 17 | 16:9 | 8 | 7:0 |  |
|  |  |  |  |  |  |  |  |  |  |
| Flit 3 | Wstrb[7] | Data[63:56] | Wstrb[6] | Data[55:48] | Wstrb[5] | Data[47:0] | Wstrb[4] | Data[39:32] | <Only for  64b write> |
|  | 35 | 34:27 | 26 | 25:18 | 17 | 16:9 | 8 | 7:0 |  |

**Figure 1: write request packet. Cmd = 3’b010 for 32b request, 3’b011 for 64b request.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | B |  |  |  |  |  |  |  |  |
|  | Flit 0 |  |  |  |  | Dummy | bresp[1:0] | Seqnum |  |
|  |  | 35:7 | 6:5 | 4:0 |  |
|  |  |  |  |  |  |  |  |  |  |

**Figure 2: Write response packet.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Flit 0 | Dummy | Region | Prot | SeqNum | Length (Command[0]) | Converter | Command[2:1] | Host Regbus | Ring ID |
|  | 35:22 | 21:18 | 17:15 | 14:10 | 9 | 8 | 7:6 | 5 | 4:0 |
|  |  |  |  |  |  |  |  |  |  |
| Flit 1 | Dummy | Address | | | | | | | |
|  | 35:32 | 31:0 | | | | | | | |

**Figure 4: Read request packet. command = 3’b000 for 32-bit reads, 3’b001 for 64-bit read requests.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | R |  |  |  |  |  |  |  |  |
|  | Flit 0 | {rresp[1], rdata[31:24]} | | | | {rresp[1], rdata[23:16]} | {rresp[1], rdata[15:8]} | {rresp[1], rdata[7:0]} |  |
|  |  |  |  |  |  |  |  |  |  |
|  | R USRSB |  |  |  |  | Dummy | rresp[0] | Seqnum |  |
|  |  | 35:6 | 5 | 4:0 |  |
|  |  |  |  |  |  |  |  |  |  |

**Figure 5: 32-bit read response packet.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | R |  |  |  |  |  |  |  |  |
|  | Flit 0 | {rresp[1], rdata[31:24]} | | | | {rresp[1], rdata[23:16]} | {rresp[1], rdata[15:8]} | {rresp[1], rdata[7:0]} |  |
|  | Flit 1 | {rresp[1], rdata[63:56]} | | | | {rresp[1], rdata[55:48]} | {rresp[1], rdata[47:40]} | {rresp[1], rdata[39:32]} |  |
|  |  |  |  |  |  |  |  |  |  |
|  | R USRSB |  |  |  |  | Dummy | rresp[0] | Seqnum |  |
|  |  | 35:6 | 5 | 4:0 |  |
|  |  |  |  |  |  |  |  |  |  |

**Figure 6: 64-bit read response packet.**

# Ring Packet Format

The protocol on the ring will be a flit-based wormhole protocol. Credits are maintained between each hop on the ring, and a flit is released upon availability of a credit.

Signals on the ring are:

Inputs:

ring\_data\_in [31:0]

ring\_data\_in\_valid

ring\_credit\_in

ring\_wakeup\_in

Outputs:

ring\_data\_out[31:0]

ring\_data\_out\_valid

ring\_credit\_out

ring\_wakeup\_out

Each head flit is a command flit, containing control information. The ring ID is encoded into the first 5 bits. It is required first for the slave to make a decision on whether to pass it to the next slave in the ring, or to accept it as its own packet.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Flit 0 | Dummy | Region | Prot | SeqNum | Length (Command[0]) | Converter | Command[2:1] | Host Regbus | Ring ID |
|  | 35:22 | 21:18 | 17:15 | 14:10 | 9 | 8 | 7:6 | 5 | 4:0 |
|  |  |  |  |  |  |  |  |  |  |
| Flit 1 | Dummy | Address | | | | | | | |
|  | 35:32 | 31:0 | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Flit 2 | Wstrb[3] | Data[31:24] | Wstrb[2] | Data[23:16] | Wstrb[1] | Data[15:8] | Wstrb[0] | Data[7:0] |  |
|  | 35 | 34:27 | 26 | 25:18 | 17 | 16:9 | 8 | 7:0 |  |
|  |  |  |  |  |  |  |  |  |  |
| Flit 3 | Wstrb[7] | Data[63:56] | Wstrb[6] | Data[55:48] | Wstrb[5] | Data[47:0] | Wstrb[4] | Data[39:32] | <Only for  64b write> |
|  | 35 | 34:27 | 26 | 25:18 | 17 | 16:9 | 8 | 7:0 |  |

**Figure 7: Ring write packet**

On a response, the ring ID is changed to ring ID=0, indicating that the destination is the Ring Master, which has ring ID 0. The error bit indicates slave error, the case in which the slave exists but the register offset does not. Decode error (slave not present) is detected and returned by the Ring Master.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Write response packet format | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Flit 0 | Dummy | | | | Error | SeqNum | Command | Ring ID |
|  | 35:14 | | | | 13 | 12:8 | 7:5 | 4:0 |

**Figure 8: Ring write response**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Flit 0 | Dummy | Region | Prot | SeqNum | Length (Command[0]) | Converter | Command[2:1] | Host Regbus | Ring ID |
|  | 35:22 | 21:18 | 17:15 | 14:10 | 9 | 8 | 7:6 | 5 | 4:0 |
|  |  |  |  |  |  |  |  |  |  |
| Flit 1 | Dummy | Address | | | | | | | |
|  | 35:32 | 31:0 | | | | | | | |

**Figure 9: Ring read request**

On a response, the ring ID is changed to ring ID=0, indicating that the destination is the Ring Master, which has ring ID 0. The error bit indicates slave error, the case in which the slave exists but the register offset does not. Decode error (slave not present) is detected and returned by the Ring Master.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Read Response packet format | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Flit 0 | Dummy | | | | Error | SeqNum | Command | Ring ID |  |
|  | 35:14 | | | | 13 | 12:8 | 7:5 | 4:0 |  |
|  |  |  |  |  |  |  |  |  |  |
| Flit 1 | rresp[1] | Data[31:24] | rresp[1] | Data[23:16] | rresp[1] | Data[15:8] | rresp[1] | Data[7:0] |  |
|  | 35 | 34:27 | 26 | 25:18 | 17 | 16:9 | 8 | 7:0 |  |
|  |  |  |  |  |  |  |  |  |  |
| Flit 2 | rresp[1] | Data[63:56] | rresp[1] | Data[55:48] | rresp[1] | Data[47:0] | rresp[1] | Data[39:32] | <Only for  64b read  response> |
|  | 35 | 34:27 | 26 | 25:18 | 17 | 16:9 | 8 | 7:0 |  |

**Figure 10: Ring read response**

|  |  |
| --- | --- |
| Command Encoding |  |
| 3'b000 | 32-bit Read Request |
| 3'b001 | 64-bit Read Request |
| 3'b010 | 32-bit Write Request |
| 3'b011 | 64-bit Write Request |
| 3'b100 | 32-bit Read Response |
| 3'b101 | 64-bit Read Response |
| 3'b110 | 32-bit Write Response |
| 3'b111 | 64-bit Write Response |

**Table 1: Ring Protocol Command Encoding**

# Timing Diagram

Flit0

command

Flit1

Address

Flit2

Data0

Flit3

Data1

ring\_data\_in[31:0]

ring\_data\_in\_valid

ring\_data\_in\_credit

clk

**Figure 11: Timing diagram of the input interface of a register bus slave receiving a 64b write request**

# High Bandwidth Example

Flit0

command

Flit1

Address

Flit2

Data0

Flit3

Data1

ring\_data\_in[31:0]

ring\_data\_in\_valid

Flit0

command

ring\_data\_in\_credit

Flit0

command

Flit1

Address

64b Write Request

32b or 64b Write Response

frpm previous slaves on ring

32b or 64b Read Request

clk

**Figure 12: Timing diagram of the input interface of a register bus slave receiving a number of back to back requests**

# Microarchitecture

Register Bus Client (Host or Bridge)

regslv\_req\_\*

regslv\_rsp\_\*

Register Bus Slave

4-flit wide full packet storage

Response from this client

Request for this client

2

1

ring\_data\_in[31:0]

ring\_data\_out[31:0]

Packet sent to next

register bus slave in the ring

2-flit wide credited storage

Packet not for this client

ring\_credit\_in

ring\_credit\_out

**Figure 13: Internal organization of a register bus slave**

Arbitration block #2 will lock the input path if a response has begun. Similarly, it will lock the response path if the input path has begun.

# Register Slave to Client Interface

Rules and Assumptions:

1. If more than one request is permitted to be outstanding to the client, the client will return the responses to the slave in order. This means that reads and writes are also ordered with respect to each other. (I probably can handle if reads and writes are ordered within themselves, and not with respect to each other, but starting with the fully ordered assumption).
2. The address put on the request bus is the address belonging to the least significant bits being requested.
3. The same is true for read responses.
4. 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.

Outputs:

regslv\_req\_valid : When 1, indicates a valid request from register slave to the host.

regslv\_req\_addr[P\_ADDR\_WIDTH-1:0]

regslv\_req\_rnw : 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: 0 indicates a 32-bit request. 1 indicates a 64-bit request.

regslv\_req\_region[3:0]: Passes along the 4-bit ARREGION/AWREGION field presented to the register bus master for this transaction.

regslv\_req\_prot[2:0]: Passes along the 3-bit ARPROT/AWPROT field presented to the register bus master for this transaction.

regslv\_req\_wdata[P\_REGBUS\_RSLV\_DATA\_WIDTH-1:0]: 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\_rsp\_ready – When asserted at the same time as regslv\_rsp\_valid, indicates the acceptance of that request.

Inputs:

regslv\_req\_ready – When asserted at the same time as regslv\_req\_valid, indicates the acceptance of that request.

regslv\_rsp\_valid : When 1, indicates a valid response from the host.

regslv\_rsp\_rdata[P\_REGBUS\_RSLV\_DATA\_WIDTH-1:0]: 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-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).

# Parameters

1. P\_REGBUS\_RING\_ID – Can take value 1 to 31. A value of 0 is reserved for the ring master. The slave logic matches this value to the first 5 bits of the ring packet to decide whether to send this to its own host, or to the next slave in the ring.
2. P\_REGBUS\_RSLV\_NUM\_OUTSTANDING – A slave could be allowed to have multiple outstanding requests to the host. Storage equivalent to sequence number bits (5) + length bit (1) times P\_REGBUS\_RSLV\_NUM\_OUTSTANDING will be instanced to be able to store the sequence numbers until the responses are received from the host. The value of this parameter is hence a number indicating the depth of the storage.
3. P\_REGBUS\_RSLV\_ADDR\_WIDTH – The number of bits required to convey register offset to a host varies depending on how much of the 16KB offset that host is using. A maximum of 16KB is allowed per host, so the maximum number of bits is 14. In the case of the router, it takes only 1KB at the 16KB offset, which is 10 bits. Similarly, a streaming bridge will require 4KB, which is 12 bits. Parameterizing this can also reduce the size of the register bus slave.
4. P\_REGBUS\_RSLV\_DATA\_WIDTH – Takes value of either 32 or 64. For clients that have only 32 bit registers, this parameter can be set to 32. Clients that have this set to 64 can still accept 32-bit requests.
5. P\_REGBUS\_RDATA\_WIDTH – This is the width of the actual ring data path, from the ring master to slave as it hops around the ring. This is currently set to 32, and can be downsized in the future.

# 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 slave is filled, it wakes up the slave. Packets will be counted as they enter and leave. The clock enable can be deasserted when all packets have left the bounds of that slave.

# Clock Crossing

This item is TBD.

# Power Gating

This item is TBD.