Regbus Controller and Regbus Master Bridge

Joji Philip

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author | Description |
| 0.1 | May 2019 | Joji Philip | Initial architecture specification for new regbus master and regbus controller. |

RBC-RBM

[1 Regbus Controller (RBC) 2](#_Toc8286942)

[1.1 Block diagram 2](#_Toc8286943)

[1.2 Converters 2](#_Toc8286944)

[1.3 Arbitration 4](#_Toc8286945)

[1.4 RBM interface 4](#_Toc8286946)

[1.5 Trackers 4](#_Toc8286947)

[1.6 Response paths 6](#_Toc8286948)

[1.7 Response flushing from tracker re-order buffer 6](#_Toc8286949)

[2 Regbus master bridge (RBM) 7](#_Toc8286950)

[2.1 Block diagram 7](#_Toc8286951)

[3 Pending 8](#_Toc8286952)

# Regbus Controller (RBC)

Regbus controller will replace existing regbus tunnel module. It allows multiple requesters to get on the regbus layer to access registers in the NoC and user register space.

## Block diagram



## Converters

RBC supports multiple ports with different protocols accessing the regbus layer. AIX4 port is expected to be used for tunneling requests from the main NoC layer to regbus layer. Input ports can have different data width and are primarily used for sending 4B and 8B read and write requests to 32-bit and 64-bit registers. All converters are expected to convert the port protocol to a common RBM interface format which specifies 8B data width.

AXI4 port is capable of sending bursts and these are split to singles using the AXI2AXIL converter. Note that these singles are 4B currently and are not suitable for accessing 8B registers. We will continue with this design for now. But needs to be changed in future

* Bursts must be split at 8B alignment, nominally creating 8B accesses.
* Based on start address alignment, first split can be a 4B access
* Based on length of burst, last split can be a 4B access

These requirements also apply to IOSF-SB converter.

### Factoring tracker credits

Read and write trackers maintain independent and dedicated credits for each input port. Corresponding read and write credit availability indication should be provided to each converter module.

* Converter must use their read, write credit\_available signal to select between reads and write to put on the RBM command interface. (credit\_available is a single bit signal indicating a non-zero value of the credit counters).
* A minimum of 1 read and 1 write credit must be reserved for each input port to ensure that ports do not block each other.

### Converter interface

This is the interface presented by each converter to the arbitration block. Once one of the converters is selected, similar interface is presented to the RBM.

Table 1 Command and write data interface

|  |  |  |
| --- | --- | --- |
| Signal | width | Comments |
| conv\_rbm\_valid | 1 | Request valid from converter |
| conv\_rbm\_read | 1 | Read = 1’b1, write = 1’b0 |
| conv\_rbm\_addrs | 32 | Address: will be 4B or 8B aligned |
| conv\_req\_lbinfo | P\_LBINFO\_WIDTH | Information associated with a request that must be recorded in the tracker for use once the response returns. This is not presented to the RBM. Example: AID of a transaction from the AXI4 port. |
| conv\_rbm\_length | 1 | 4B = 1’b0, 8B = 1’b1 |
| conv\_rbm\_secure | 1 | Secure access = 1’b1, non-secure access = 1’b0 |
| conv\_rbm\_wdata | 64 | For 4B access, data must be at positions [31:0] irrespective of the address alignment.  For 8B access data is little endian |
| rbm\_conv\_grant | 1 | Grant indicating that a command (and wdata for writes) was accepted. |

Table 2 Read data and response interface

|  |  |  |
| --- | --- | --- |
| Signal | width | Comments |
| rbm\_conv\_rvalid | 1 | Valid read response and data |
| rbm\_conv\_rdata | 64 | 4B data must be at [31:0] irrespective of address alignment. 8B data is in little endian format. |
| rbm\_conv\_rlbinfo | P\_LBINFO\_WIDTH | Read request specific information being looped back to corresponding read response |
| rbm\_conv\_rerror | 1 | Address decode, security violation or other errors |
| conv\_rbm\_rgrant | 1 | Grant indicating that the rdata was accepted. |

Table 3 Write response interface

|  |  |  |
| --- | --- | --- |
| Signal | width | Comments |
| rbm\_conv\_bvalid | 1 | Valid write response |
| rbm\_conv\_berror | 1 | Address decode, security violation or other errors |
| rbm\_conv\_blbinfo | P\_LBINFO\_WIDTH | Write request specific information being looped back to corresponding write response |
| conv\_rbm\_bgrant | 1 | Grant indicating that the write response was accepted. |

* Loopback information is a request specific information that RBC or regbus NoC does not use, but needs to be returned with the corresponding response. P\_LBINFO\_WIDTH must be based on the widest information any of the input ports requires. All other ports are expected to drive this full width even if all bits are not used by them.
* All interfaces have single cycle transactions, i.e. if a request or response transaction is valid and is granted in the same cycle, then a new transaction can be posted in the next cycle.
* A valid request is allowed to change while it has not received its grant. For example an input port may post a read request and later change it to a write request if the earlier request wasn’t granted.

## Arbitration

A simple round-robin arbitration can be performed between requests from multiple converters. Selected command and write data has to be muxed and presented on the RBM interface.

* Free location index in the read or write tracker where the command will be allocated is used to tag the command to RBM with a unique ID
* Once a selected requested is granted, appropriate credit pool in the read or write tracker is decremented.

## Security

For the IOSF sideband master port, incoming SAI must be translated into secure/non-secure attribute of the regbus transaction.

* 3 special registers, RAC, WAC and CP have to be added in the RBC CSR space
* These registers have RW access and are of ‘secure’ type (only accessible by secure RBM transactions)
* Access by non-secure transaction cause a decode error: writes are dropped and reads issue a 0 response.
* Reset value of the 3 registers must be derived from dedicated strap input pins
* A local address decode of access to these 3 registers is needed. When any access targets these special registers, incoming SAI value must lookup CP register to mark the packet as secure/non-secure. Only certain special SAI values have secure access to these policy registers, all other SAI get marked as non-secure effectively denying them access to these registers.
* For all other register addresses in the NoC, 64-bit RAC/WAC register specifies the secure/non-secure status of incoming read/write transaction based on the SAI value in the IOSF transaction



Figure SAI to secure/non-secure translation

## RBM interface

* On the interface to RBM, **req\_info** field is not present
* On the interface to RBM, **req\_id** is presented. Its width P\_ID\_WIDTH is the larger of the number of bits required to encode number of entries in read and write trackers.
* Read and write response interfaces have **rid** and **bid** fields representing the transaction ID **req\_id** of the original request sent to the regbus NoC

## Trackers

Two independent trackers are maintained for read and write requests sent to the RBM. Parameters P\_MAX\_OUTSTANDING\_RD, P\_MAX\_OUTSTANDING\_WR size the number of entries in the read and write trackers respectively. Following descriptions apply to both trackers.

### Tracker reservations and flow control

As mentioned earlier, each input port must have dedicated reservations in the trackers to prevent them from being blocked by other ports. This must be implemented as follows.

* Tracker entries are divided into multiple credit pools, 1 credit reserved for each input port and all the remaining entries belonging to a shared credit pool
* credit\_available signal to a given input port is asserted either if the shared pool has credits, or if the port’s 1 credit is unused
* Every time a request is granted by the arbiter, if shared credit is available then it is decremented by 1 and the entry allocated for the request in the tracker is marked to have used shared credit
* When a request is granted by the arbiter and shared credit pool is empty, credit dedicated for that input port is decremented and the tracker entry is marked to have used dedicated credit
* When an entry is deallocated during response flush, credit is returned to the shared pool or to the port’s dedicated pool based on the marked credit type in the entry.

### Tracker data structure

Trackers are logically structured as dynamically allocated linked-list queues. Independent linked-list queues are maintained for each input port. Responses can be received out-of-order from the regbus NoC. So the responses are held in the tracker till they can be flushed out in-order to each input port. This ensures that each input port receives responses in its original request order and that an input port cannot block any other port from making independent progress on request and response paths.

* One logical linked-list queue maintained per source port
* Each entry in the tracker goes through the following logical states

**Free🡪 request allocated-waiting for response 🡪 response received-waiting for ordered flush 🡪 Free**

* For read tracker each entry has following fields

{Source port ID}

{lbinfo}

{64-bit of space reserved for rdata, 1-bit reserved for rerror},

{credit type} // 1’b1 : shared pool, 1’b0 dedicated port credit

{entry state}

{linked\_list pointer}

* For write tracker each entry has following fields

{Source port ID}

{lbinfo}

{1-bit reserved for berror},

{credit type} // 1’b1 : shared pool, 1’b0 dedicated port credit

{entry state}

{linked\_list pointer}

* Once granted, first free entry (also used for tagging **req\_id**) is used to record the request and is added to the tail of the selected input port’s linked-list.

## Response paths

Independent read and write response paths are available from RBM

* Response IDs **rid, bid** are used to index into the corresponding tracking tables and record the received response data and error status.

## Response flushing from tracker re-order buffer

As described earlier, tracker has logical linked-list FIFOs for each input port with list’s head-to-tail order holding oldest-to-newest requests sent from that input port. Response for these can be received out of order from the regbus NoC. However, responses have to be sent back to the input port in the original request order.

* Once response to the oldest request (head) is received, that input port’s queue is ready to flush
* Walking the list, all successive entries behind the head that have received their responses can also flush
* Queue must stop flushing when first entry which hasn’t received a response is encountered during the walk (Or if the queue goes empty). A new eligible queue has to be selected for flush next.
* Multiple input port queues can be eligible for flushing at the same time. Some fair arbitration has to select among the ready queues.
* Once an entry is flushed and de-allocated, credit is returned to the appropriate pool in the read or write tracker.

# Regbus master bridge (RBM)

## Block diagram



* Performs address and security lookup for read and write transactions using a common address lookup table.
* Decode errors or security violations detected by address table cause a looped back error response from within the RBM
* RBC write data interface is 8B but write data interface to regbus NoC is 4B. So 8B requests from RBC will be downsized and sent as 2 data flits. 4B requests are sent as a single data flit.
* Similarly 4B read responses are received as a single data flit. 8B read responses are received as 2-data flits which are upsized and sent to RBC on the 8B interface.
* A simple tracker indexed by **req\_id** is used to record the dependent power domains of a transaction while its response is outstanding in the regbus NoC. This is used to implement fence and drain functionality in the RBM
* 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 a common Rxswitch interface.

# RBC-RBM Low power



## Regbus end to end

* RBC-RBM combination is in a single power domain
* Two hops for handling power gating
  + Mainband master bridge considers RBC-RBM as a slave power domain. It tracks requests outstanding to the RBC-RBM. It is responsible for fence-drain/autowake for all power domains on mainband only up to and including RBC-RBM
  + RBM tracks requests outstanding to various register slaves on the regbus NoC. RBM manages fence-drain/autowake for all downstream power domains corresponding to regbus routers, ring master, ring slaves.
  + Ring slaves are always in the power domain of their host

## RBC-RBM power domain

* From the master bridges’ perspective, RBC-RBM can be an auto-wake or decode error power domain.
  + If autowake, if any transactions are outstanding to it, any attempt to bring down RBC-RBM power domain will receive a fence\_deny. Similarly, any transactions to the regbus layer (RBC-RBM) will cause a wakeup request for RBC-RBM power domain if it is gated.
  + If decode error only, all transactions bound for RBC-RBM must receive decode error at the master bridge when the power domain is gated. Master bridge is required to perform full fence and drain of transactions to RBM-RBC before accepting power down request for RBC-RBM domain
* For the other slave ports (non-tunnel) on the RBC, corresponding master agents must be in the same power domain as RBM-RBC. No power, voltage or clock domain boundary is planned to the agents ports.
* It is the responsibility of the corresponding masters to drain all outstanding transactions before RBC-RBM domain can be powered down. Once gated, any new transaction can only be issued after waking up the RBC-RBM domain. Agents must handle this with higher system power manager.
* RBC may be asked to fence-drain on its slave ports (non-tunnel ports only) when asked to power down. This can be done for the AXI, APB ports. Unclear if credit IOSF-SB interface can be disconnected and fenced. **(OPEN)**
* RBC-RBM will indicate idle status when no transactions are outstanding in RBC and RBM and states are restored to reset==idle condition.

## RBM: Regbus NoC power domains

* RBM maintains an indexed table for transactions outstanding on the NoC layer. Table is indexed with transaction ID of the requests and responses.
* Outstanding flag in the entry is set when a transaction is sent to the regbus NoC and is reset once the corresponding response is received. Table also records the power target ID of each transaction.
* Power target ID is obtained from the RBM address lookup. It can either represent a specific register slave on the regbus layer or a group of slaves.
* Target ID of a transaction is used to lookup all the power domains required by a transaction (request and response).
* Some power domains can be ‘decode error only’, while others can be auto-wake
* On the RBM, pd\_active for RBC-RBM power domain must be tied 1’b1. This prevents it from fencing for its own power domain and blocking the interface from RBC

### Tracking

* When power down request is received for a power domain, a fence\_deny is returned if the domain is autowakable and any transactions are outstanding to it. A fence\_ack can be returned when there are no transactions outstanding to the power domain.
* Outstanding requests to each power domain in monitored using the transaction tracking table and the parameter table mapping each power target to the dependent power domains as shown below.



Figure PD busy status based on outstanding transactions

### State machine and transaction handling

* For any incoming register transaction, if pd\_active is low for any ‘decode error only’ power domains, then the transaction is returned with a decode error.
* Else, If pd\_active is low for ‘autowake power domain’
  + If the fence request for that PD was denied, the transaction should be allowed to make progress
  + Else, If the fence request was accepted, RBM interface is halted and auto wake request is issued for those power domains
* Below figure is a broad schematic. Fence ack/deny state machine should be reused from ACE master bridge



Figure PD status check for new requests

# Opens

* User and NoC register space address lookup in RBM
* Any clock, voltage, power domain crossing on the interfaces