OCP Master Bridge

[1 Introduction 3](#_Toc456876471)

[1.1 Terminology 3](#_Toc456876472)

[1.2 High Level Requirements 3](#_Toc456876473)

[1.3 OCP Configurations Supported 3](#_Toc456876474)

[1.4 OCP Protocol Overview 5](#_Toc456876475)

[2 Microarchitecture 8](#_Toc456876476)

[2.1 Block Diagram 8](#_Toc456876477)

[2.2 Block Details 9](#_Toc456876478)

[3 Other System Level Considerations 15](#_Toc456876479)

[3.1 Posted writes 15](#_Toc456876480)

[3.2 Byte enables 15](#_Toc456876481)

[3.3 Width conversion 16](#_Toc456876482)

[3.4 Ordering 16](#_Toc456876483)

[3.5 MDataInfo and WUSER interoperability 16](#_Toc456876484)

[3.6 Reuse Considerations 16](#_Toc456876485)

[4 Clocking and Reset 18](#_Toc456876486)

[4.1 Clocking 18](#_Toc456876487)

[4.2 Reset 18](#_Toc456876488)

[5 Low Power 19](#_Toc456876489)

[5.1 TODO 19](#_Toc456876490)

| Revision # | Author | Date | Notes |
| --- | --- | --- | --- |
| 0.1 | Perry Wang | 5/16/2016 | Initial Draft |
| 0.2 | Perry Wang | 5/16/2016 | Data width and wrap burst specification |
| 0.3 | Perry Wang | 7/1/2016 | See change bars |
| 0.4 | Perry Wang | 11/3/2016 | See change bars |

# Introduction

## Terminology

This document uses following terms interchangeably:

* Master / Initiator
* Slave / Target
* Strobe / Byte Enable
* Tag / AID

## High Level Requirements

1. OCP 2.2 Compliance
2. Interoperability with AMBA bridges
3. All protocol agnostic features of master bridges should be supported
4. Regbus interface should look as close to AXI master bridge as possible
5. Netspeed LPv2 support
6. Latency: 1-3 cycles depending on configuration
7. Bridge properties and parameters should look as close to AXI4 master bridge as possible for easier NocStudio and DV integration

## OCP Configurations Supported

OCP allows a huge number of configuration options. Following describes the subset that Netspeed will support.  
  
**Basic**

* Flow control signals always required for both request and response
* Only Read and Write, WriteNonPost commands supported
* Data width 32, 64, 128, 256, 512
* Address width 14-60bits
* ~~MAddrSpace support (slave only)~~
* Little and Big Endian support

**Simple Extensions**

* Both read and write byte enable supported
* force\_aligned=1 required for read bursts (this allows us to map to AXI narrows)
* ReqInfo and RespInfo info support

**Burst Extensions**

* Only INCR, STRM, and WRAP supported
* WRAP request total size must be 16B, 32B or 64B, (and in the future, maybe 128 and 256B)
* Must be precise (defined length)
* Single Request Multiple Data mode only
* MDataLast and SRespLast are required
* BurstLength: 1-255 (no 256)
* Netspeed limitation: bursts are not allowed to cross 16KB boundary.
* Master bridge can be configured to split bursts at 64, 128, 256, 512, 1024, 2048, 4096B boundaries
* STRM will be split into single beat transactions when talking to AXI slave bridges. (TBD: for OCP to OCP, should we preserve STRM?)

**Tag Extensions**

* Tag reuse allowed (multiple outstanding for same tag)
* taginorder always 0 (TODO: we might be able to support taginorder fairly easily)
* Only 1 or 0 supported for tag\_interleave\_size

**Thread Extensions**

* ConnID supported
* mthreadbusy\_pipelined supported
* mthreadbusy\_exact must be 1
* Threads can be mapped to traffic classes using add\_traffic commands in nocstudio

**Clocking**

* Async, Ratio Sync supported (clk\_host and clk\_noc)
* ocpClkEnable: TBD

**Reset**

* We only support sreset (an output from master bridge)

**Security**

* Supported via ConnID and ReqInfo
* TBD: security mapping to AXI

**QoS**

* Isochronous traffic support via proprietary sideband signal
* VC priority and arbitration can be configured similar to other bridges

**I/O Coherency**

* TBD

**Low power support**

* Always on, for now, but can coexist in an LP enabled NoC.
* ~~LPv2 support similar to AXI master bridge~~
* ocpClkEnable: TBD

## OCP Protocol Overview

Here is a table that summarizes the OCP protocol **relevant to our profile**:

|  |  |  |  |
| --- | --- | --- | --- |
|  | OCP 2.2 | AXI4 | Comment |
| Basic Flow Control Handshake | Valid / Accept | Valid / Ready |  |
| Protocol channels (w/ independent data flow) | cmd, wdata, resp | AR,AW,W,R,B | See TX switch section. Write data and command channels are merged at switch interface. |
| Single command multiple data | Optional in OCP, but required for Netspeed bridges | Always SCMD |  |
| Support for multiple outstanding transaction (within a thread) | Tags  Write interleaving between different tags not allowed.1  Read data interleaving allowed between different tags.  Multiple outstanding allowed for same tag, but must be in order.  Read and write share the same pool of tags | AID  Write interleaving between different AIDs not possible.  Read data interleaving allowed between different AIDs.   Multiple outstanding allowed for same AID, but must be in order.  Read and write have different pools of AIDs | We only support tag\_interleave\_size=1 for OCP |
| Threading | ThreadID, can switch thread every cycle | Not supported | Netspeed Virtual AXI can approximate threading, but more coarse grained |
| Ordering | Write data must follow same order as write command.   Single command channel, ordering maintained between read and write.  No ordering requirement whatsoever between threads.  No ordering required between tags in the same thread, unless there’s address overlap (interconnect responsible). | W and AW must follow same order. Requests of same AID within AW and AR channel must be in order.  There are no ordering requirement between AW and AR.  No ordering requirement between different AIDs.   Interconnect not responsible for address overlap based ordering. |  |
| Narrows | No | Yes | AXI can perform partial reads and write using narrow. |
| Narrow burst | OCP doesn’t support narrow, so bursts address only increments by full word size. | Address is calculated based on AxSize . | This means that if OCP master maps partial read to AXI narrow, we have to break up AXI reads into singles. |
| Partial Write/Read | MDataByteEn  MByteEn | WSTRB | OCP does not support narrows so it needs to use read byte enable to issue partial read. |
| Addressing | Byte Address | Byte Address |  |
| Data Width (bits) | 32, 64, 128, 256, 512 | 32, 64, 128, 256, 512 |  |
| Unaligned access | Not allowed.  Must be aligned to word size (data bus) | Allowed |  |
| Burst | Netspeed supports:  - precise burst - 1-255 beat bursts - INCR, STRM, WRAP burst  (TBD: XOR) | - undefined burst not allowed by spec - AXI4 allows up to 1-256 beat  - INCR, FIXED, WRAP burst types |  |
| Wrap | Netspeed supports 16,32,64B transaction size, 2-64 beat | Netspeed supports 16,32,64Btransaction size, 2-64 beat2 |  |
| Page Alignment | No restriction, but netspeed limits crossing to 16KB for timing purpose | Transaction must not cross 4KB boundary. |  |
| User Signals | MDataInfo,MReqInfo,  SDataInfo,SRespInfo  MDataInfo contains both per byte and per transaction info. | AWUSER, ARUSER, WUSER,RUSER,BUSER  WUSER only contains per byte info. |  |

1. From OCP spec:   
     
   *For tagged write transactions with datahandshake enabled, the datahandshake phase must observe the same order as the request phase. The master cannot interleave requests or datahandshake phases with different tags within a transaction.*
2. Release 1604 and prior releases only supports wrap size up to 64B.

TODO: cache, parity, security, etc  
interrupt, errors (out of band), and other sideband (out of band) signaling

# Microarchitecture

## Block Diagram

OCP Rsp

OCP Cmd

Serialize

Serialize

Merge

RX Switch

RDATA Endian Convert

Serialize

Re

Reorder

Join

Fork

WDATA Endian Convert

TX Switch

Req Sched

Address Overlap Table

Tag (AID)  
Table

Address Decode Table

Serialize

Serialize

Rsp I/F

Serialize

Serialize

Req I/F

Serialize

Serialize

Split

Serialize

Serialize

Serialize

CSR

Reorder Buffer

Serialize

Re

Rsp Gen / Drop

clk   
crossing?

clk crossing

## Block Details

### Fork

This block is responsible for interfacing between the-per thread Req IF and the OCP initiator. Its main job is to dispatch request and deal with per thread flow control.

### Req I/F (per thread)

Request Interface deals with the commands and write data interfaces. It is responsible for:

1. Clock synchronization logic (sync, async, ratio sync)
2. Output registering (for timing)
3. Buffering
   1. required in async mode
   2. when threads>1, shallow buffer is required to absorb the latency of the pipelined threadbusy.

### Split (per thread)

There are a few reasons why OCP transactions might need to be split:

1. If the transaction interacts with coherent domains, in which case we need to split into 64B.
2. AXI requires that transactions never cross 4KB boundary. AHB requires that transactions never cross 1KB boundary.
3. Slave’s max burstlength is smaller than the current burst. This sort of split is done at the slave side to avoid keeping track of slave widths in the master.
4. When a burst crosses slave boundary. Master needs to detect the end of the slave address range and split accordingly.

### Serialize (per thread)

After the transactions are split, the master bridge needs to meet ordering rules of OCP. Specifically:

* For a given thread, transactions with address overlap need to be completed in order. This is required even if they have different tags.
* All responses for the same tag needs to come back in order.   
  There are a few reasons why responses for the same tag could come back out of order, despite individual slaves guaranteeing completion in order. For example,  
    
  1. If layer VC changes  
  2. Requests go to different slaves

In addition to ordering hazards, there are other reasons where serialization is required. Following is a summary of the serialization behavior. Note that the decision to stall is based on the OR of the applicable table entries. So there’s no particular row based priority implied.

**Yes**: stall the thread pipeline and wait for all outstanding transactions for the **same tag**, in the same thread, to come back

**Yes/All**: stall the thread pipeline and wait for all outstanding transactions for the **all tags**, in the same thread, to come back  
**Yes/Addr**: stall the thread pipeline and wait for address overlapped transactions in the same thread, to come back

|  |  |  |  |
| --- | --- | --- | --- |
|  | With Reorder Buffer | Without Reorder Buffer, tag\_interleave\_size=1 | Without Reorder Buffer, tag\_interleave\_size=0 |
| Address overlaps with outstanding transactions | Yes/Addr | Yes/Addr | Yes/Addr |
| Route different from outstanding transactions of same tag due to QoS change1 | Yes | Yes | Yes |
| Destination slave different from current outstanding transactions of the same tag | No | Yes | Yes |
| Start or end of a split, and interleave responses enabled | No | No | Yes/All |
| Tag overlaps withoutstanding transactions, and if unique tag mode is turned on | Yes | Yes | Yes |
| Tag overlaps with outstanding transactions, and there’s a potential for RAW or WAR ordering hazard at AMBA slave2 | Yes3 | Yes | Yes |
| No tag overlap, no address overlap | No | No | No |

1. Change of route could cause even requests to go out of order. Requests using the same tag need to arrive the slave in order. So even with reorder buffer we have to stall on the request side.
2. AXI slave has independent AR and AW channel. So read and write responses could come back out of order even if OCP master issues on a single channel with the same tag.
3. If there are separate read and write reorder buffers, the ordering between read and write will not be maintained. If there’s a way to have just one reorder buffer, then it might be possible.

### Address Decode Table

This table provides the mapping between transaction address and slave information. Slave information includes:

* Slave ID, routing information.
* Size of the slave address space. This is needed for splitting transaction that crosses slave boundaries.
* Split size select. Each master supports 2 split sizes, and this bit allows selecting between the two sizes per slave.
* Type of slave. AMBA or OCP. This information is useful to allow OCP specific information to be preserved for OCP to OCP transactions, such as read byte enable, and posted/nonposted. The information will also be used to determine the destination RX switch host interface.
* Security, reloc, etc
* Power domain dependency information for selective fencing
* Load balancing

Note that the address decode table is statically programmed at RTL generation time via parameters and not programmable at run time. Parameters are passed to each module that use the table, and a common address decoder is instantiated for each instance that require simultaneous access to the table.

### Tag table (a.k.a. AID Table)

This table stores information about outstanding transactions. It is the same module as the one in AXI master bridge, except that it’s configured to be in OCP mode. For each tag there can be a link list. The link list keeps tracks of the order of the outstanding transactions using the same tag. The main functions of this table is as follows:

1. Contain information to allow us to figure out if current transaction is going to the same target as the outstanding transactions. Serialization decisions sometimes depend on whether the request goes to the same target.
2. Store ordering information for the same Tag/AID, so that reorder buffer can reorder the response.
3. Stores the starting timestamp for each outstanding transaction (used by timeout logic).

TODO: AXI has separate write and read tables. The size of these tables are determined by max write and read outstanding. OCP only has 1 pool of tags. Do we still preserve two separate max outstanding parameters?

### Address Overlap Table

OCP spec requires that ordering must be maintained between transactions (of the same thread) going to overlapping addresses. For each outstanding transaction, master bridge must keep track of the address range that it accesses. When a new request arrives and overlaps with any of the outstanding transactions, the bridge must hold off the request in the thread until there are no more outstanding transaction for the thread (TODO: optimization: only flush transactions that overlap?). Due to the nature of the table look up, a separate structure is created instead of adding information to AID table.

The size of address overlap table will equal to the number of outstanding transaction configured for this master bridge. Multiple threads can share this table. In order to prevent threads from blocking each other, each thread will have reserved entries. The extra entries will be shared between the threads.

### Req Scheduler

This scheduler decides which thread to choose from to send to the TX switch. The decision is based on priority, fairness, and whether the VC is available.

TODO: more details on arbiter  
TODO: Isochronous.

### Write Data Endian Conversion

Netspeed NoC assumes little endian data in packets (for packing/unpacking). All data will be converted to little endian before being sent to the TX switch.

### TX Switch

TX Switch interfaces with routers in different layers and provides request packet interfaces to the host side logic in master bridges. For OCP, there will be a single host interface, but VC can switch every cycle, and there will be a per VC flow control.

The outgoing flit contains a field to specify the destination host interface (channel). If the destination slave bridge is AXI, the destination host interface will be one of the 2 host interfaces (AR, AW+W). If the destination slave bridge is OCP, there will only be 1 destination channel for request.

TX switch sometimes also handles width conversion. Please see the width conversion section.

### RX Switch

RX switch interfaces with routers in different layers and provides response packet interfaces to the host side logic in master bridges. For OCP, there will be two host interfaces for responses. When an AXI slave bridge is sending response to OCP master bridge, R and B host interfaces will be destined for the respective channel. When an OCP slave bridges sends response to OCP master bridge, it will always send to the R channel at destination. This preserves the response order for OCP to OCP transaction.

Note: it is possible to have one host interface in RX switch, but AXI slave bridges will need to know which master ID is OCP, and send R and B to the same destination host interface. Netspeed slave bridges currently don’t have that knowledge about the master bridges.

RX switch sometimes also handles width conversion. Please see the width conversion section.

### RDATA Endian Conversion

Netspeed NoC assumes little endian data in packets (for packing/unpacking). All response data coming from the NoC will be converted to the host endianness.

### Rsp Gen / Drop (per thread)

Posted writes from OCP master will always be converted to non-posted before being sent across the NOC. As such, write responses need to be dropped. This module also will log any write response errors and issue an error interrupt.

Read and write responses sometime need to be fabricated if the request address does not result in any legal target address space.

One of the reason for converting posted writes to non-posted is that the slave/target could be AXI and it always requires responses to writes. It is also useful for master to implement some sort of barrier for ordering purpose.

### Reorder (per thread)

If the reorder buffer is present, this block will manage access toreorder buffer, with the help of tag table. Potentially out of order responses will be written to the buffer according to its sequence number, and will be read out in order before progressing up the response data path. Note that reorder buffer only reorders transactions with the same tag.

### When the complete response for the original transaction has been drained from reorder buffer, the tag will be retired from the tag table as well as address overlap table.Reorder Buffer

Reorder buffer stores out of order responses so that the reorder logic can read out the responses in order. This is especially important when a transaction has been split and the response must be presented to the master in a single burst.

When reorder buffer exists in the master bridge, some of the serialization in the request datapath can be avoided. However the reorder buffer is fairly expensive in area.

### CSR

This block contains all the control and status registers for the master bridge. It implements the ring slave and is similar to AXI master bridge.

### Merge (per thread)

Requests are sometimes split in the ingress datapath of the master bridge. We need to merge the response before presenting to the OCP master upstream.

Error status in the responses also need to be merged in a pessimistic fashion.

### Rsp I/F (per thread)

Response Interface deals with the OCP read and write responses. It is responsible for:

1. Clock synchronization logic (sync, async, ratio sync)
2. Output registering (for timing)
3. Buffering, if in async mode is used

### Join

This block muxes responses from different threads to a single OCP interfaces. Arbitration is done every cycle (not transaction boundary) and based on fairness and SThreadBusy and SDataThreadBusy.

TBD: weighted roundrobin or roundrobin?

# Other System Level Considerations

## Posted writes

OCP master bridge will always convert a posted write to a non-posted write so that the master knows when a write has completed. This is required because masters sometimes need to flush the writes to maintain ordering. This is also used to maintain ordering between read and write to overlapping addresses.

If an OCP master (initiator) talks an OCP slave (target), and both supports posted and nonposted, it is possible that customers want to preserve the ability to control type of write command on per-transaction basis. To support this, extra information needs to be transported in the NoC packets so that OCP target can issue the type of write the initiator specifies. Without such information, the target would have to resort to static mapping. For example, all writes will be issued to the OCP target as posted or nonposted writes. Static mapping is still required in case the request comes from AXI master.

## Byte enables

### Read Byte Enables

OCP does not support narrows. So in order issue partial reads, byte enable must be used. To maintain AXI compatibility (AXI does not support read byte enables), we require force\_aligned to be 1. Force\_aligned==1 allows all OCP partial accesses to be mapped to AXI narrows.

An OCP master bridge will always convert OCP read byte enable into narrows before forming the command in the request NoC packet.

An OCP slave bridge will always receive request packets with size information (narrow), and convert it to the proper read byte enable.

(TODO: can we say that force\_aligned=1 is required only for read only? Theoretically we can support arbitrary write enable)

### Write Byte Enables

OCP write byte enables maps to AXI byte enables pretty well. Because OCP address is always word aligned, OCP master bridge can safely pass on the byte enable even to AXI slave bridges as is. The only exception is when the OCP initiator is big endian. OCP master bridge will transform the write byte enable as well as data so that we’re compatible with the AXI slaves.

## Width conversion

In general, master bridges have fixed width data path. Master bridges do not have knowledge about the data width of the slave. Conceptually, width conversion happens at the slave bridges. In reality, only the request command transformation (size, length) is done at the slave bridges. The packing and unpacking of data can happen at any of the 3 places:

1. Tx switch
2. Rx switch
3. Router

Packing and unpacking logic assume little endian, and NocStudio determines the optimal location of the packing/unpacking.

## Ordering

AXI has 5 channels (AR/AW/W/R/B) at the interface, and OCP has 3 channels (cmd/wdata/resp). At the switch interface, the write command and data are merged into a single channel so we have 4 independent channels for AXI and 2 for OCP. If a transaction goes from AXI master to OCP slave, AXI ordering rules will be preserved. However, if a transaction goes from OCP master to AXI slave, ordering between read and write will be lost. The only way to preserve ordering in that case is to only allow one outstanding per tag for that data flow. (TBD: should this be a property?)

## MDataInfo and WUSER interoperability

In OCP, MDataInfo contains both per transaction user bits as well as per byte. However, AXI’s WUSER only contains per byte information. In order to interoperate with AXI slave, OCP master bridge will create NOC packets such that the per transaction portion of MDataInfo is appended to the MReqInfo (upper part).

## AXI to OCP error response mapping

|  |  |
| --- | --- |
| **AXI** | **OCP** |
| OKAY | DVA |
| EXOKAY | DVA |
| SLVERR | ERR |
| DECERR | ERR |

Note: FAIL resonse is only allowed for WRC command so we can’t use it for DECERR.

## Reuse Considerations

1. AID table reuse
2. Reorder buffer reuse
3. Address table reuse
4. ch\_valin and ch\_valout reuse
5. AID table, reorder buffer, address tables might need to have multiple look up ports, if applicable. Use reservation for threads.
6. Tx switch change needed to switch threads every cycle. Upsizing downsizing could be an issue.
7. Allocate bits in flits to transport read byte enable and posted write, in case the destination is OCP slave
8. Almost full support for ch valin.
9. Selective fencing reuse?

# Clocking and Reset

## Clocking

OCP master bridge allows 2 clock domains. Host clock and noc clock.

TODO: do we still want to allow clock crossing in RX switch?

## Reset

TODO

# Low Power

## TODO