NetSpeed Systems – Virtual AXI Interfaces Introduction

Revision 1.0

January 31st, 2015

NetSpeed Systems – Virtual AXI Interfaces Introduction

REVISION HISTORY

Rev 1.0	Jan. 31, 2015	Prelimenary Draft Release

About This Document

This document introduces the virtual AXI interfaces available in NetSpeed IP.

Audience

This document is intended for users of NocStudio:

- NoC Architects
- NoC Designers
- SoC Architects
- SoC Designers

Prerequisite

Before proceeding, you should generally understand:

- Basics of Network on Chip technology
- Basics of AMBA Protocols

Related Documents

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

- NetSpeed NocStudio Orion AMBA User Manual

Customer Support

For technical support about this product, please contact support@netspeedsystems.com

For general information about NetSpeed products refer to: www.netspeedsystems.com

Contents

List of Figures

1 ARCHITECTURAL INTENT

It is common to want a slave device to have multiple interfaces so that different kinds of traffic can be sent without interference. For instance, a memory controller may want to have an interface for normal traffic, as well as one for traffic with real-time requirements, such as display or audio. Even the normal traffic may be sub-divided into latency sensitive flows such as CPU traffic, and other traffic that is much less sensitive to latency.

To avoid head-of-line blocking issues, these various traffic classes need to have separate interfaces to the memory controller or other slave. If they share an interface, then one traffic class can block the others.

Multiple physical interfaces can get very expensive because of the number of pins required, as well as the number of physical wires that have to be transported to the slave. Virtual interfaces can reduce both problems.

AXI protocol does not support virtual interfaces. It uses a READY/VALID handshake. When a request is made on the interface, it must stay there until accepted by the other side. If a low priority request is blocked, higher priority requests will get stuck behind them. To support different traffic classes, the slave would have to utilize multiple physical interfaces, with each interface requiring its own bridging logic.

The intent of this specification is to create an interface addition that allows virtual channel awareness across an interface (creating virtual interfaces), without actually changing any of the existing signals, either in number of in meaning. Any protocol checkers that exist on the interface can continue to be used as is.

While NoC to Host interfaces are the focus in this document, the same mechanism can be used for traffic from Host to NoC. This can allow R and B channels of a slave to be virtualized, as well as AR and AW channels from a master.

2 ARCHITECTURE

This section describe the protocol enhancements and flexibility of the solution.

2.1 VIRTUAL CHANNELS/INTERFACES

The following diagram shows the additional interface signals that allow a virtual interface for an AR path.

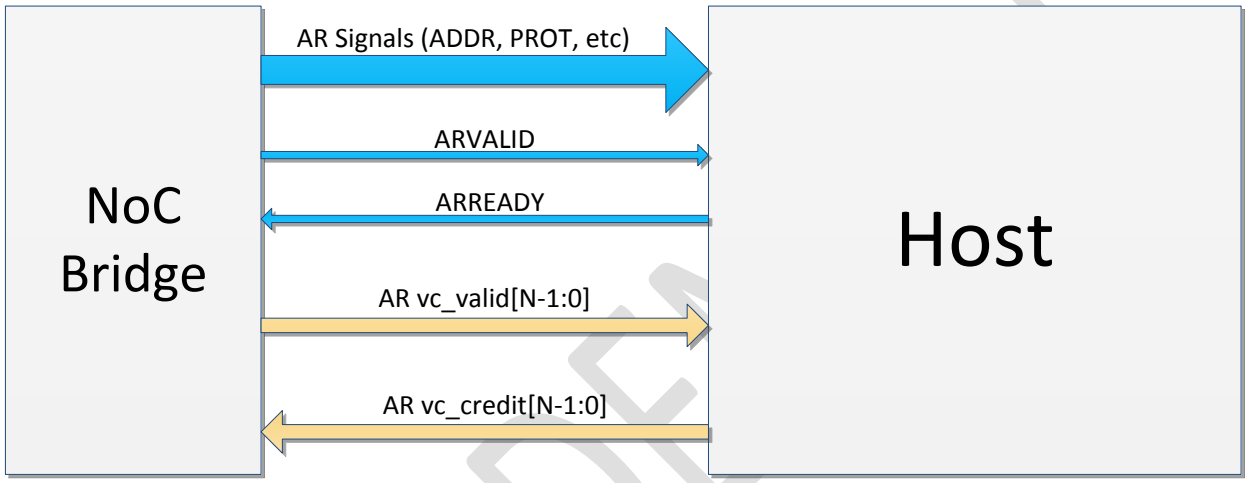

Figure 1: AR interface additions

The top three signal sets, in blue, are the standard AXI interface signals. These don't change. There is an ARVALID and ARREADY for flow control, as well the other AR signals.

The bottom two signal sets are new, and are used as an add-on to the interface to allow virtual interfaces. These signals are vc_valid and vc_credit. Each of these signals has a width equal to the number of virtual interfaces desired (N). So if 3 interfaces are needed, each of these signals will be 3 bits wide.

The credit information is a method by which the Host can send information to the NoC bridge to indicate that it has a dedicated resource available for that virtual channel. By communicating with these credits, the bridge is able to know ahead of time whether the Host will be able to accept a request of that type. If no credit is available, the bridge will not send a request to the Host for that traffic type, since once it does, it can create head-of-line blocking.

When the bridge detects that a credit is available for a traffic type, it can choose to send that traffic type to the Host. It will do so by setting the normal ARVALID signal along with other signals. In the same cycle, it will indicate to the Host that which virtual channel the request is for using the vc_valid signal.

The host must still assert ARREADY for the request to complete. If the ARREADY signal is deasserted, the bridge will continue to attempt to send that AR request until it successfully sees ARREADY with ARVALID. Since dedicated storage is expected, and that storage has been communicated via a credit signal, it is possible that the host will never deassert ARREADY, since any request being sent on the interface could drain.

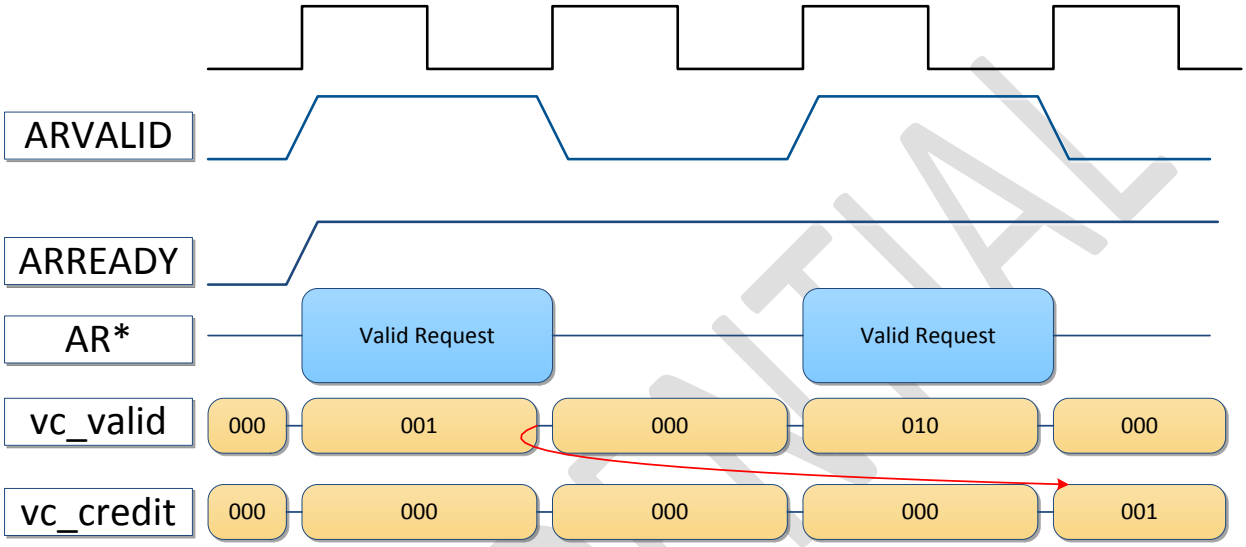

Figure 2: Interface Timing Diagram

The diagram above shows example cycles on the interface. At the beginning of time, the bridge has credits for all of the VCs. In the first cycle, it decides to send a valid request on the AR channel. It indicates which virtual channel will be used by setting the vc_valid signal to indicate that the request is for VC0 (indicated with value 001). Since ARREADY is high, the request is accepted and deasserted in the next cycle.

Two cycles later, the bridge decides to send another request, this time using VC1 (indicated with value 010).

In the last cycle, we see the Host sending a credit back to the bridge for VC0. This credit can be returned on any cycle after the request has been accepted, and is only dependent on the Host freeing up a resource.

Note that whenever ARVALID is asserted, one and only one vc_valid signal is asserted. The ARVALID is a logical OR of the vc_valid vector. However, it retains its original protocol meaning, allowing protocol checkers to continue to monitor the interface.

2.2 THE AW/W INTERFACES

The AR channel is relatively straightforward to virtualize. The AW and W signals have more complex scenarios, and so require special attention.

To maintain compatibility with the AXI protocol, AW and W channels must be coordinated so that the order of requests are the same in each channel. This mean we cannot interleave W data between the two virtual interfaces. Interleaving is on a per-request basis only. This has a number of implications.

Only AW is virtualized

The AW channel will allow virtual channels, but the W channel will not. If a request is sent on an AW channel, it defines the usage of the W channel. There is no benefit to virtual channels for the W channel as there can and will be some head-of-line blocking.

W will only send if corresponding AW is available

While the W channel will not have virtual channels, the bridge will not send data on the W channel unless the corresponding AW has an available credit. If low priority requests are blocked, the W channel will not attempt to go, as that would lock out the high priority requests.

Large Requests can cause blocking

Since an AW request can be up to 4KB long, this can have the effect of blocking high priority channels until all 4KB can be accepted and consumed within the target. It may be preferable to limit the size of requests to a virtual slave in order to avoid this issue. On top of that, if request size is limited, it may be beneficial for the slave to preallocate storage for the data before sending a credit for a channel. This will ensure the traffic can drain quickly and free up the interface for higher priority requests.

2.3 VIRTUAL INTERFACE FLEXIBILITY

The virtual interfaces of a device have significant flexibility. The AR, AW, R, and B channels need not all have virtual channels, or have the same number.

For instance, it may be beneficial to have a virtual channel for AR reads to a memory controller for display reads only. Since display only performs reads, there may be no need for a virtual AW channel. There is no requirement to have both.

The R and B channels can also have virtual channels. It may be beneficial to have a matching set of channels for AR and R, or AW and B. However, this is not required. It is likely that congestion in a system usually happens on the AR or AW path to memory, while the response paths from memory can drain easily. In this case, only the AR or AW paths are needed.

2.4 NoC Virtual Channels and Virtual Interfaces

The NetSpeed NoC already provides the ability to separate traffic flows into multiple virtual channels in order to avoid head-of-line blocking within the network. This is done by specify the traffic class "QOS" inside the add_traffic* commands. While additional virtual channels may be used for deadlock, this traffic class specification provides guarantees that the traffic flows can be independent. The traffic classes within the NoC can be specified as having equal priority, or different priority.

When a virtual interface is specified in NocStudio, the traffic classes that can send to that physical interface can be mapped to one of the virtual interfaces. So if there are two AR channels, it could be set up so NoC QOS level 0 maps to the first channel, while NoC QOS level 3 maps to the second. Each NoC QOS level must be mapped to one and only one of the virtual interfaces. If a virtual interface has no traffic mapped to it, an error will be flagged by NocStudio.

2670 Seely Avenue
Building 11
San Jose CA 95134
www.netspeedsytems.com