WHITE PAPER

Introduction to QoS Virtual Networks (QVN)

Ashley Stevens
Senior FAE, Systems IP

December 2013

The Problem

Today's SoCs often consist of dozens of masters and slaves, but typically most masters only target main memory (i.e. SDRAM), while the majority of slaves are only accessed by a very limited number of masters, primarily the CPU(s). It's common for a large number of masters to contend for access to main memory and for the memory system to become a key bottleneck in the system. QoS regulation and prioritization systems attempt to regulate traffic flow based on the requirements of the masters, e.g. bandwidth or latency targets and the capability of the memory system. Memory controllers typically contain a transaction queue and can re-order transactions based on QoS priority, but in heavily congested systems the memory controller queue may saturate resulting in the controller stalling the interconnect and ultimately creating back-pressure on masters. This can cause blocking in the interconnect and no matter what QoS schemes are employed, latency will be indeterminate and uncontrolled. As a result, high priority traffic may get blocked behind low priority traffic, and cannot make progress to enter the memory controller queue where it could be prioritized appropriately. It would be desirable to enable traffic to flow un-interrupted from the master through the interconnect to the memory controller queue, where it can be prioritized accordingly.

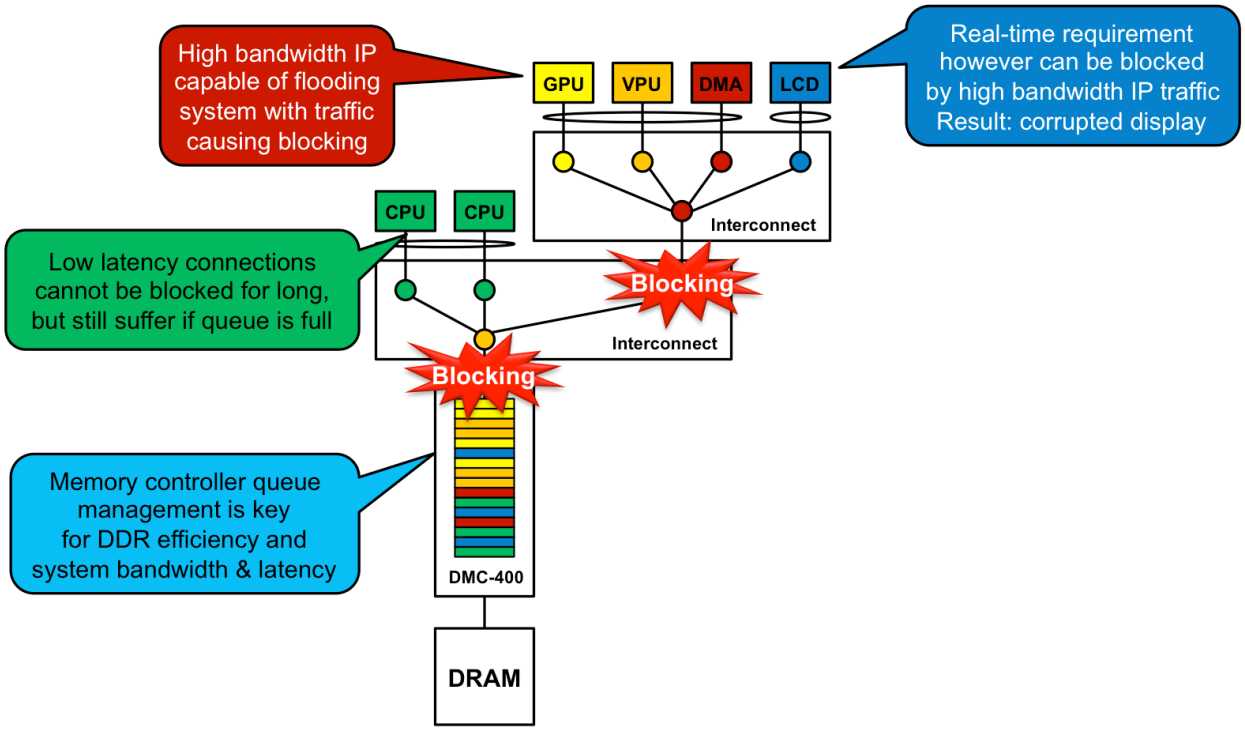

Figure 1: *Without QVN the high-bandwidth GPU can block the low-latency LCD*

QoS Virtual Networks (QVN)

QoS Virtual Networks (QVN) operates by ensuring that a transaction can always be accepted at its target before it's initiated. Masters wishing to initiate a transaction must first obtain a 'token' from the slave. The slave arbitrates token requests and hands over tokens to the masters based on the priority of the requests and the slave's internal state. For example, a memory controller with a certain number of slots in its transaction queue may present tokens only to the highest priority masters when the queue length exceeds a pre-defined threshold. Thus it can reserve certain transaction queue slots for transactions above a pre-determined level of priority. QVN enables arbitration by the slave using knowledge of its own internal state. Putting the slave in charge of transaction arbitration emancipates the slave from the drudgery of dumbly following the ordering of transactions presented to it from the masters in the system. The result is a memory-centric design in which the slave is in control of the masters, picking and choosing which master to service next to make the best use of scarce memory bandwidth resources, based on knowledge of memory system state (e.g. open DRAM rows) and QoS priority requirements. In the past the approach taken has often been to use a memory controller with multiple slave ports. In the days of AMBA® AHB™ interfaces multi-port memory controllers enabled a form of multiple outstanding transactions even though the AMBA AHB protocol didn't actually support outstanding

transactions over the interface. But many designs continue to use multi-ported memory controllers even with the AMBA 3 AXI™ protocol because although the protocol supports multiple outstanding transactions this approach provides un-contended paths to the memory controller and enables the memory controller to perform transaction arbitration. With QVN it's possible to obtain all these benefits without the wiring and area overhead of multiple AMBA AXI interfaces on the memory controller.

QVN Signaling Over AXI

By way of a quick reminder, the AMBA AXI interconnect consists of 5 independent channels of communications. In the forward direction (from master to slave) there are:

```
AR -  Read address
AW -  Write address
W -   Write data
```

and in the return path (slave to master) there's:

```
R -   Read data
B -   Write response
```

QVN works by augmenting each of the forward path channels with additional signals to request and receive a 'token'. There can be multiple virtual channels over a single physical channel, up to a maximum of 16 virtual channels per physical channel[1]. For each virtual network there's a set of QVN signals. The QVN signals comprise of a VALID signal and a QoS priority field from the master to the slave and a READY signal from the slave to the master. When both READY and VALID are asserted a token is transferred from the slave to the master. The slave uses the QoS priority field to determine whether to grant the master a token based on it's internal state and the QoS priority of other masters requesting tokens. This signaling is present on the AR and AW channels. The W (write data) channel has the READY/VALID handshake but not the QoS priority signaling. The possession of a token grants the master the right to initiate a transaction on the interface and guarantees that the transaction will be accepted by the entity providing the token. Note that each beat on the W write data interface requires a separate token. A master may request more than one token when it has more than one transaction to initiate. Each of the three forward direction channels has a single bit-field to indicate which virtual network the transaction is being initiated on. This enables the slave to identify which token in the master's possession it's 'spending' on the transaction.

[1] Note that this is the limit of the protocol, individual IP blocks may have lower limits. For example CoreLink™ NIC-400 supports 8 VNs per physical network and 4 VN interfaces per master interface.

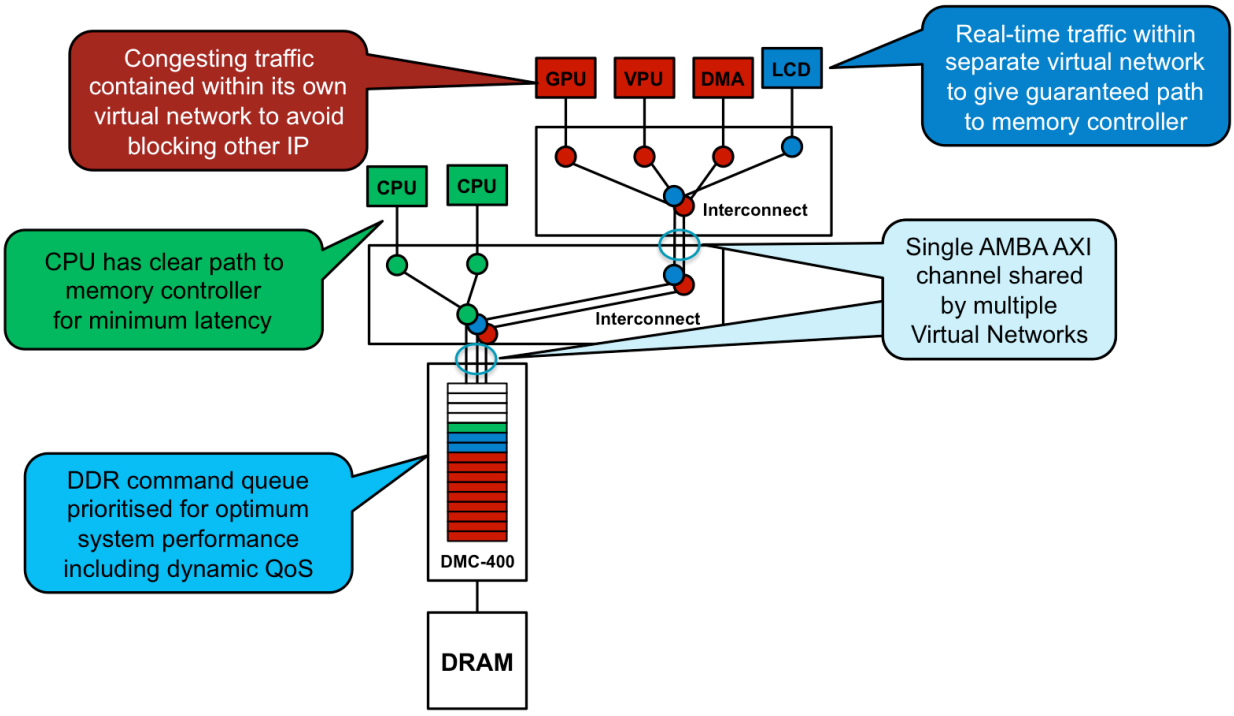

Figure 2: *QVN enables virtual channels over the same physical AMBA AXI interface*

Although this description has been from the point of view of the master, it's important to understand that in a QVN system it's very much the slave that's the key element. The benefit of QVN is that it enables the slave (memory controller) to control the flow of traffic in the system. It enables the creation of an intelligent slave that can optimally schedule transactions and manage traffic in the system. To utilize QVN the slave should be QVN-enabled because QVN is intended to put the slave in control, to facilitate intelligent slaves able to schedule master transactions. If the slave's not smart then there's no point in putting it in control of the system. That said, it's also important to understand that it's typically only one target slave that needs to support QVN, the dynamic memory controller (SDRAM controller) plus of course the entire path to it from the masters, i.e. the interconnect. Other slaves that don't have complex scheduling requirements and don't support multiple outstanding transactions aren't required to support QVN. Within the AMBA Designer GUI environment it's possible to terminate a virtual network targeting such slaves within the interconnect.

In a QVN system the interconnect must support QVN because it needs to support QVN signaling pass-through from the master to slave and slave to master. Since an interconnect is a master to a downstream memory controller slave, it needs to request a token from the memory controller before it can initiate a transaction. A master connecting to an interconnect must request a token from the interconnect. Tokens therefore flow from the slave, via the interconnect to the master. To reduce latency, in

addition to the request/grant (ie valid/ready) handshake mechanism for obtaining tokens, tokens may also be pre-allocated at reset time. When a master has pre-allocated tokens, the slave will start out with commensurately fewer tokens available to hand out. Pre-allocated tokens are supported on the address channels AR and AW, not on the write data channel W and there can be a maximum of one pre-allocated token for each of the AR or AW channels per virtual network.

An interconnect may support multiple outstanding transactions, therefore it may make available more than one token and may accept more than one transaction. However the key concept is that transactions will always be accepted and will not be stalled by de-assertion of READY either by the interconnect or by the target slave. An interconnect supporting multiple outstanding transactions can enable higher priority transactions to overtake and pass lower priority ones within the interconnect and thus avoid blocking.

At the time of writing the QVN protocol is currently supported in[2] :

- CoreLink™ DMC-400 Dynamic Memory Controller
- CoreLink NIC-400 Network Inter-Connect (with QVN-400 plug-in)
- CoreLink CCI-400 Cache Coherent Interconnect (rev r1 onwards)
- CoreLink ADB-400 AMBA Domain Bridge (rev r1 onwards)

Note: The CoreLink NIC-400 supports QoS Virtual Networks (QVN) only with the addition of the QVN-400 plug-in which provides QVN functionality in the interconnect. QVN-400 requires a separate license from the NIC-400 and requires a separate license feature to be enabled.

Summary

QVN makes system latency and bandwidth deterministic and predictable; preventing blocking in the interconnect by ensuring that a transaction can be accepted before it's initiated. A master must obtain a token from the slave indicating that a transaction slot in the slave is available before it can initiate a transaction. QVN enables a slave to stall a master without stalling the network interconnect between the master and the slave. Virtual networks enable multiple masters to share the same network wiring without causing blocking. A slave supporting multiple outstanding transactions may reserve certain transaction slots for high priority masters. QVN enables slave-driven arbitration without the requirement for a multi-ported memory controller and multiple interconnects. To reduce initial latency a master may be provided with pre-allocated tokens at reset. Together with QoS regulation of the masters, QVN provides a complete QoS solution for on-chip interconnects. QVN functionality is enabled in the CoreLink NIC-400 with the QVN-400 plug-in and is also available in CoreLink CCI-400 revision r1 and later. Further information on the QVN protocol is available in the ARM Confidential document *CoreLink QVN Protocol Specification ARM HI 0063B,* available under license.

[2] Note: This paper is not intended to address AMBA 5 CHI™ or the products that use it like CoreLink CCN-504.