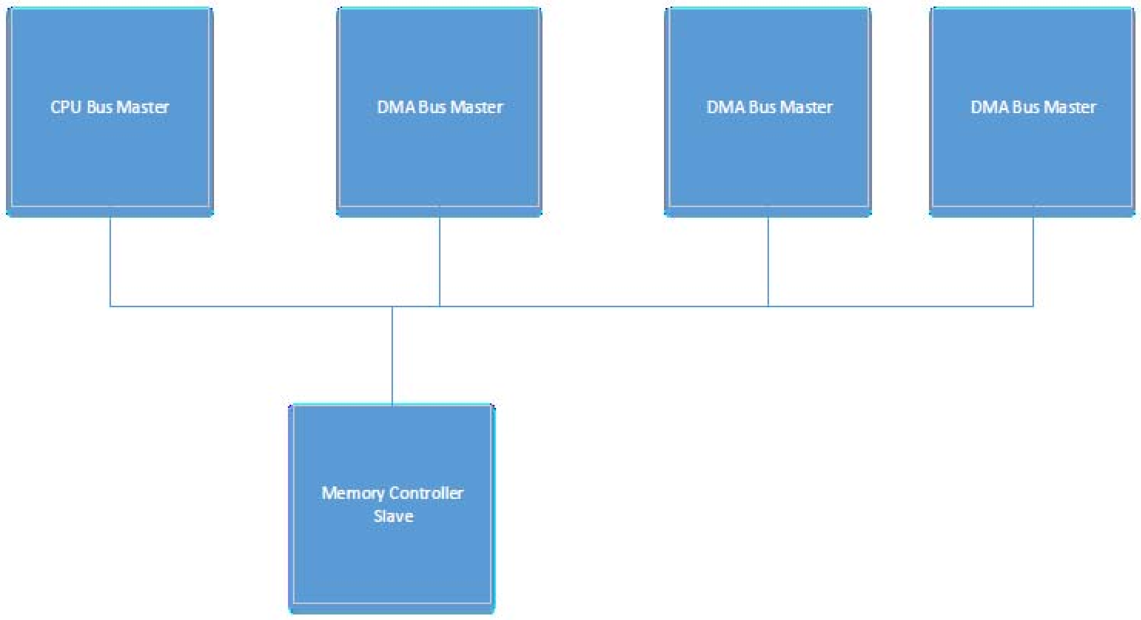
FEEDBACK CONTROLLED DYNAMIC QUALITY OF SERVICE SYSTEM

HP INC

Feedback Controlled Dynamic Quality of Service System

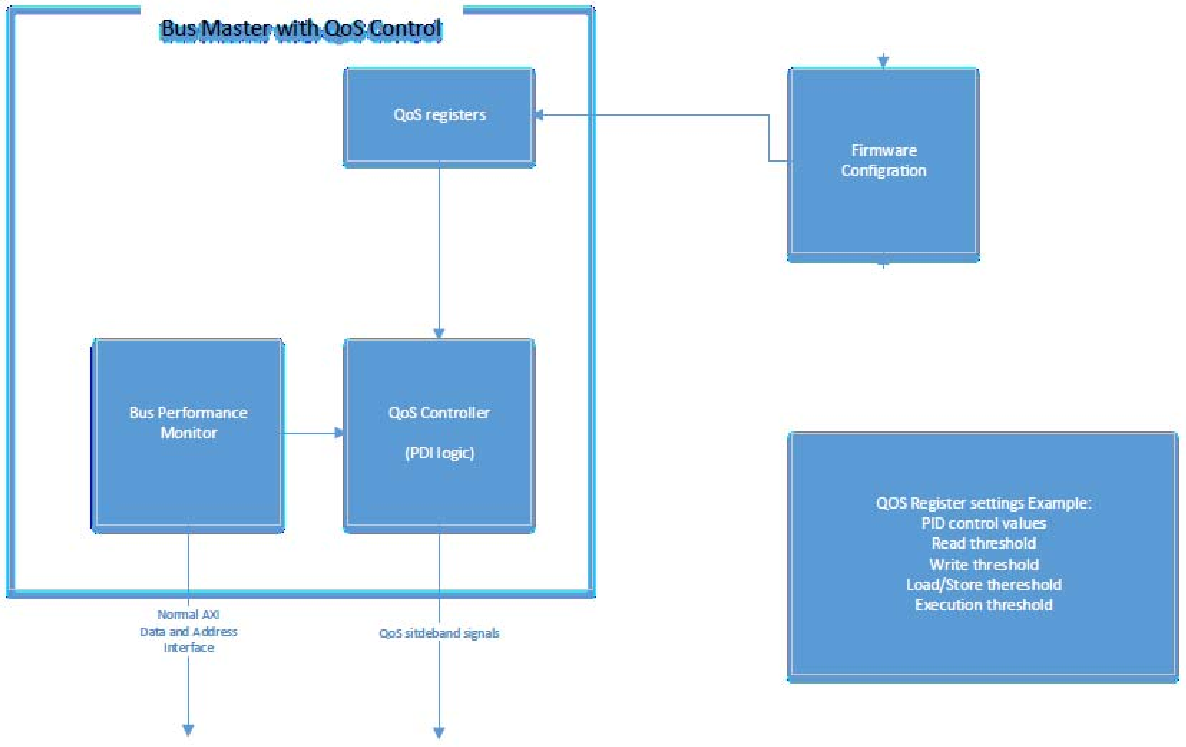


Digital ASICS (application specific Integrated circuit) and SOCs (System on chip) are used as primary centers of compute and processing in multitude of electronic devices including mobile phones, notebooks and tablets. These ASICs/SOCs have a complex architecture of processing blocks that are configured in the master/slave configuration where multiple masters can request resources and or processing from a single/multiple slave block. When this happens, there are contention issues for resource and bandwidth at the slave end. To make sure each master gets the quality of service from the slave, QOS sideband signals are transmitted over the connecting bus from master to slave. These signals are then used for arbitration and prioritization of requests on the salve side.

Limitation of the current approach is that the QOS signals are static and do not adapt to particular product configuration requirement or any other dynamic criteria.

We provide a means to control these QOS signals dynamically through a set of configurable criteria via Firmware which is then monitored by the master and QOS is calculated on the fly.

This allows for dynamic resource throttling as per the requirement for each master.



In a multi master-slave arrangement as in a digital ASIC/SOC, A master node might need to have a certain criterion for certain dynamic quality of service from the slave node. Let's take the following example,

A DMA master like in a USB controller might want a read rate of 1Mb/ms from DRAM slave controller.

A DRAM slave might be serving multiple masters at any given moment in time and hence does not guarantee that rate. USB transfers might then get slow or fail.

To address such a concern, systems like these are provided with QoS sideband signals on the AXI interconnect bus between the master and the slave. Let's say there are N masters, then each master can be configured with a static QoS (Priority) for certain operations like read/write/execute or Load/store and based on the priority of the master's request, those nodes are served prioritized during slave level arbitration if concurrent requests arise at the slave as in the above example. So, USB controller in this example will be given highest priority for read requests.

Current state of the art SoC designs use a single set of QoS signals and those must be programmed when firmware is initialized to one of 5 static levels (Critical, High, Medium, Low and unimportant). Such a solution does not have visibility into what other DMA masters in the system are requesting. If all masters are set to High, then in reality no one gets any preferred access.

With our proposed systems,

Each master node gets to choose a set of criteria for resource acquisition or processing that the slave must meet. These are expectations for operations like read/write/execute/load or store configured per master in QoS configuration registers.

If we take the example of the USB controller (DMA) expecting a certain data read rate of 1Mb/ms then that's the expected Read rate as configured in QoS Register.

DRAM bandwidth signals are monitored and provided as feedback to the QoS controller on the Master side. QoS controller then updates an internal control loop to update the gain to QoS value for the master based on expected threshold (in this case 1Mb/ms). If the provided rate is greater that 1Mb/ms it reduces the QoS to yield the bandwidth to other masters.

We can configure these control loop parameter values (PID values) that then dynamically change the QoS value such that we reach the expected transfer rate from the slave.

These PID value also govern the responsiveness of the masters, it might not be critical to reach a certain QoS setpoint for certain masters quickly, but in some other cases that might be.

We also provide a Max QoS or QoS saturation threshold. It's used to prevent starvation of other masters during concurrent access to a slave resource.

Here is a case when multiple masters are involved in the contention.

Let's say there are 3 masters and one slave

We can preset a priority for each master such that if all 3 masters have concurrent access to slave, the control loop saturates to the configured QOS for that master (max QOS per master is preset via configuration in a QoS config register) This so that there is no starvation for other masters.

unlike static QOS, where we don't verify if a certain set criterion is met, dynamic QoS system informs the master which criteria are being met and system self-throttles. At the slave end if there is contention, then the QOS values will saturate to the most optimal configuration for those criteria (due to max QOS saturation config) for each master.

Conclusion:

With dynamic QoS and saturation, multiple masters can have their own control loop to reach the required level of performance when accessing the system resources(slaves) and yet work cooperatively and yield priority when their demands are met

Following issues have been addressed by the Dynamic QoS system with saturation

a) Non adaptive resource requests
b) Starvation of other masters (due to lack of configurability)
c) limited levels of control

d) Masters can’t yield to others when criteria are met
e) Inefficient prioritization policies reducing performance of the system

Disclosed by Vijaykumar Nayak, Mark John Reed and Douglas James Hillmer, HP Inc.