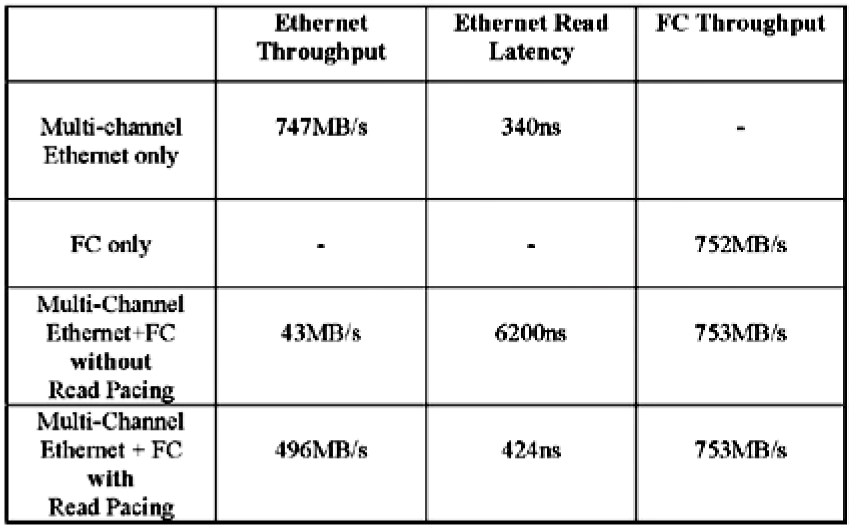
Using nextgen PCI Express switches to eliminate network I/O bottlenecks

By Steve Moore, PLX Technology

Controllers in today's network-connected embedded systems often are overwhelmed by the data streaming to and from the various I/O sources; it can be difficult for the system's root complex to absorb high-speed bursty traffic such as 10Gig Ethernet when it competes with very fast streaming data from sources such as InfiniBand and Fibre Channel (FC) storage elements.

For example, when a few bytes of Ethernet data get stuck behind large packets of FC data in the root complex, the latency that is introduced by this congestion will severely impact system response time and create bandwidth limitations (see **Table 1 below**).



	Ethernet Throughput	Ethernet Read Latency	FC Throughput
Multi-channel Ethernet only	747MB/s	340ns	-
FC only	-	-	752MB/s
Multi-Channel Ethernet+FC without Read Pacing	43MB/s	6200ns	753MB/s
Multi-Channel Ethernet + FC with Read Pacing	496MB/s	424ns	753MB/s

Table 1. Ethernet latency bandwidth tradeoffs

The next generation of PCI Express (PCIe) switches have added many new features to mitigate the effects of having to process competing data protocols, thereby improving overall system performance.

Advanced new features such as Read Pacing, enhanced port configuration flexibility, dynamic buffer memory allocation, and the deployment of PCIe Gen2 signaling are reducing I/O bottlenecks, providing dramatic improvements in system performance in server and storage controllers.

Performance Limited by "Endpoint Starvation"
When two or more endpoints are connected to a root complex through a PCIe switch, with unbalanced upstream versus downstream link-widths (and hence unbalanced bandwidths) and an uneven number of read requests are being made by the endpoints, one endpoint inevitably dominates the bandwidth of the root complex queue. The other endpoints suffer reduced performance as a result. This is known as "endpoint starvation," which can make it appear as if the system is congested and not performing optimally.

Figure 1 below shows a typical root complex connected to two endpoints through a PCIe switch. In this example, there is a x8 upstream port and two x4 downstream ports. The FC HBA is a good example of an endpoint that could dominate the bandwidth of the root complex queues.

In this example, the FC HBA makes several 2KB read requests, which are then queued by the root complex, filling up the queues in root complex.

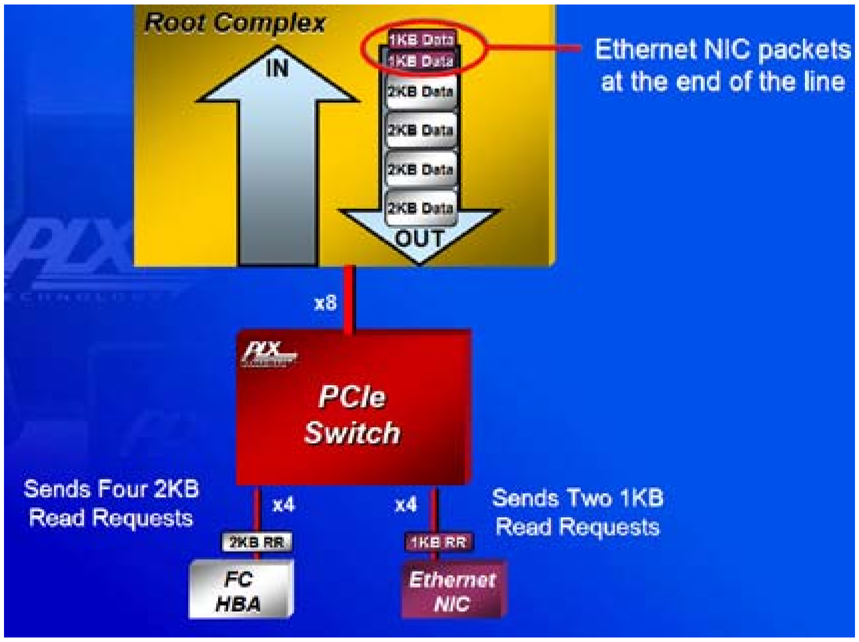


Figure 1. Endpoint starvation

While the queues are full, the Ethernet NIC makes two 1KB read requests. The Ethernet NIC must wait for the root complex to service all of the read requests from the FC HBA before they're serviced. Thus the NIC is "starved."

Read Pacing "Feeds" the Starving Endpoint
Endpoint starvation is solved and the endpoint is "fed" -- with a new PCIe switch feature called Read Pacing, which is available on the latest Gen 2 PCIe switches.

Read Pacing provides increased system performance with a more balanced allocation of bandwidth to the downstream ports of the switch. With Read Pacing, the switch can apply rules to prevent one port from overwhelming the completion bandwidth or buffering in the system.

Figure 2 below shows the same example, with a FC HBA and an Ethernet NIC on the downstream ports of a switch which aggregates traffic into a root complex. The FC HBA makes several 2KB read requests.

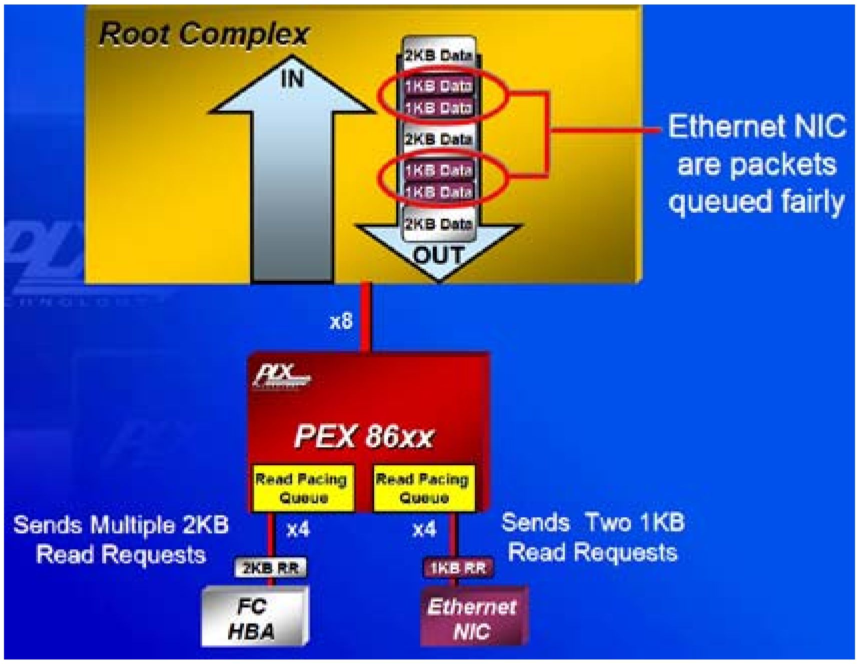


Figure 2. Read pacing eliminates endpoint starvation

With Read Pacing, the switch controls the number of the FC HBA's read requests forwarded through at a time. Programmable registers in the switch control the number of read requests forwarded to the root complex.

As the Ethernet NIC makes its two 1KB read requests, the switch allows both read requests through, thus balancing the flow of data from both endpoints. As shown in **Figure 2**, a 2KB read for the FC HBA through the root complex is immediately followed by two 1KB reads for the Ethernet NIC, resulting in balanced traffic for each endpoint.

Read Pacing allows the Ethernet NIC to be serviced more frequently without impacting the bandwidth of the FC HBA. Hence, endpoint starvation is eliminated with Read Pacing. The chart below compares the performance improvement that can be achieved with and without using Read Pacing in a real world system, where the FC issues 16 4K read requests ahead of the Ethernet single 1K read request.

Increase Performance by Optimizing Buffer Size Dynamically
Early PCIe switch architectures provided each port with a fixed amount of buffer RAM. **Figure 3 below** compares a typical type of buffer allocation, seen in the older switch designs, with the new Dynamic Allocation scheme found in the latest Gen2 switches.

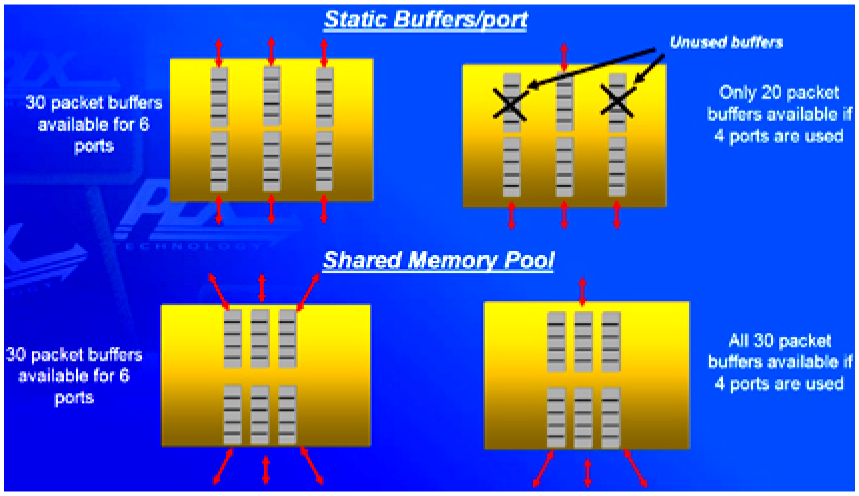

Static Buffers/port
Unused buffers
30 packet buffers available for 6 ports
Only 20 packet buffers available if 4 ports are used
Shared Memory Pool
30 packet buffers available for 6 ports
All 30 packet buffers available if 4 ports are used

Figure 3. Dynamic allocation leads to more buffers

In this example, a six-port switch is designed with a total of 30 packet buffers, with five buffer segments available on each port. If only four ports are used, then the buffers allocated to the two unused ports are wasted.

Since a larger buffer will translate into better performance, it would be nice if that unused memory could be used to increase the size of the buffers on the four ports that are being used.

In the latest Gen2 switches, it is possible to do just that. This feature is known as Dynamic Buffer Allocation, where a shared memory pool is available to any port, and the size of the buffer is allocated dynamically depending on the number of ports in use.

Increasing Performance by Sizing Buffers Dynamically

Figure 4 below compares a static buffer per port scheme with a Dynamic Scheme on a switch which is configured with three differing port widths. Since the smaller width ports require less bandwidth than the wider ports, they should require fewer packet buffers as well.

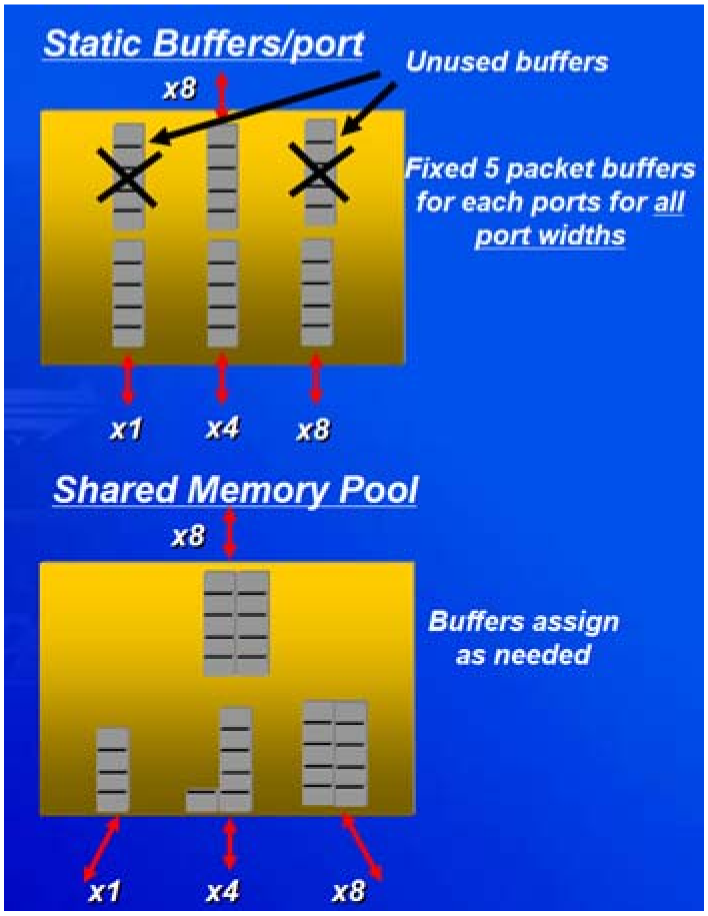


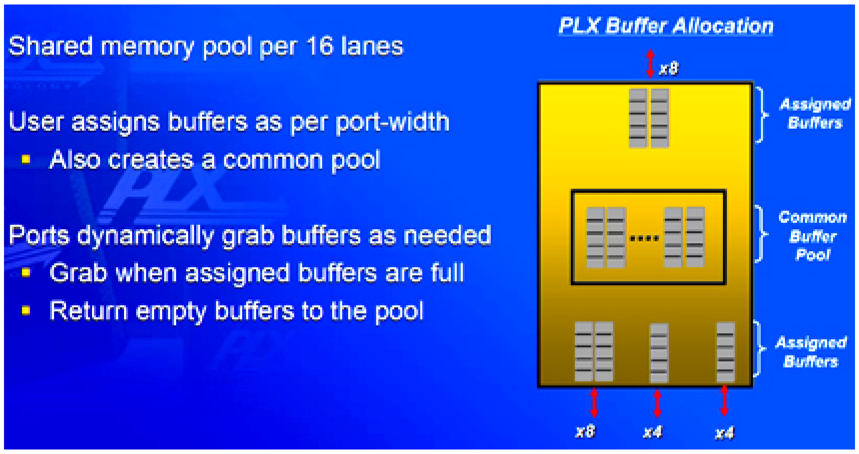
Figure 4. Dynamic allocation allows more appropriate buffer sizing

In this example, a x8 upstream port is servicing three downstream ports, one a single x1 port, one a x4 port and third one a x8 port. With a static fixed buffer per port architecture, the x1 port is allowed the same buffer size as the x8 ports. Not only is this not the optimal buffer assignment, but there are two unused groups of packet buffers.

With Dynamic Allocation, buffers are assigned as needed to each port based on the width of each port. Since there are no unused buffers, a larger total amount of buffer is available, increasing the size of buffer that may be applied in the ports that need the extra bandwidth.

In this example, in the bottom half of **Figure 4**, ten packet buffers are allocated to each of the x8 ports, whereas six buffers are given to the x4 port and four buffers are available for the x1 port. Thus the amount of buffer available on a given port is dynamically assigned based on the traffic loading on each port, resulting in higher overall system performance.

Real-World Implementation of Dynamic Buffer Allocation
A real-world implementation of Dynamic Allocation can be seen in **Figure 5 below**. Here, a 24-lane PCIe Gen2 switch is configured with a x8 upstream port, a x8 downstream port and two x4 downstream ports.


Shared memory pool per 16 lanes

User assigns buffers as per port-width

- Also creates a common pool

Ports dynamically grab buffers as needed

- Grab when assigned buffers are full
- Return empty buffers to the pool

Figure 5. Dynamic allocation using a 24 lane switch

This switch's configuration has been set up by the user with assigned buffer space for each port and an uncommitted common (or shared) buffer pool per 16 lanes. The buffers have been assigned proportional to the port width, i.e., the x8 ports each have 10 packet buffers, the x4 ports four each.

A common buffer memory pool is set up with five buffer packets for each of the 16 downstream lanes. Each of the ports may dynamically grab buffers as needed to support its own traffic bandwidth.

For example, a port may grab buffers when its assigned buffer memories are full; conversely, a port may return buffers to the pool when they are empty. This dynamic reallocation has two benefits in switch design: it makes full use of the buffer memory on-chip and it requires less overall memory to achieve optimal performance.

Port Flexibility Improves Performance, Simplifies Layout
In the previous generations of PCIe switches, one port was fixed as the upstream port while all other ports were defined as downstream, with severely limited lane count/port count combinations.

A new wave of PCIe Gen 2 switches now offers flexible and versatile port configuration schemes, with ports configurable as x1, x2, x4, x8, and x16 for maximum port bandwidth ranging from 250MB/s (x1 port, Gen 1 signaling) to 8GB/s (x16 port, Gen 2 signaling), with several intervals in between. This means it is easier to optimize lane bandwidth and power dissipation and port layout trace-width from port to port.

In addition, these new switches support auto-negotiation of the port width, reducing the number of lanes that are active in a port down to match endpoints that are connected. For example, if a NIC with a x4 port is connected to a x8 (or x16) port on the switch, the switch will automatically reduce the number of active lanes for that port down to a x4 configuration.

Selectable Upstream Port Simplifies High Performance Layout
These newer switches also support a moveable upstream port. Any port, in fact, can be defined

as the upstream port in these devices. This can be optimized to meet the needs of the traffic through each port of the switch.

Additionally, the layout of a system board is enhanced by this flexible upstream port assignment. **Figure 6 below** illustrates how, in a storage application, a flexible upstream port assignment allows spreading of high-speed traces evenly on a system board with a 16-lane switch configured with one four-lane upstream (US) port and three x4 downstream (DS) ports. The system on the left uses a switch with a fixed US port.

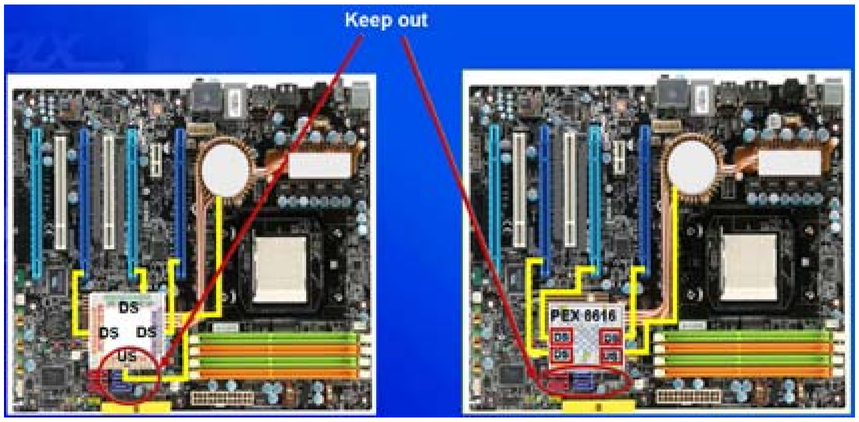


Figure 6. Port flexibility enhances board layout

The fixed US port creates severe trace congestion since the DS ports are required to route through the SATA connectors, creating an undesirable crosstalk environment. The photo on the left shows the same system with a switch that has a flexible US port. This flexibility allows the layout designer to avoid routing the high-speed PCIe lanes through the equally powerful SATA2 data paths, thus reducing crosstalk, enhancing signal integrity and improving transmission margin.

Dual Cast

In addition to balancing bandwidth and improved buffer allocation, these new switches also support Dual Cast, a feature that allows for the copying of data packets from one ingress port to two egress ports, allowing for higher performance in dual-graphics, storage, security, and redundant applications.

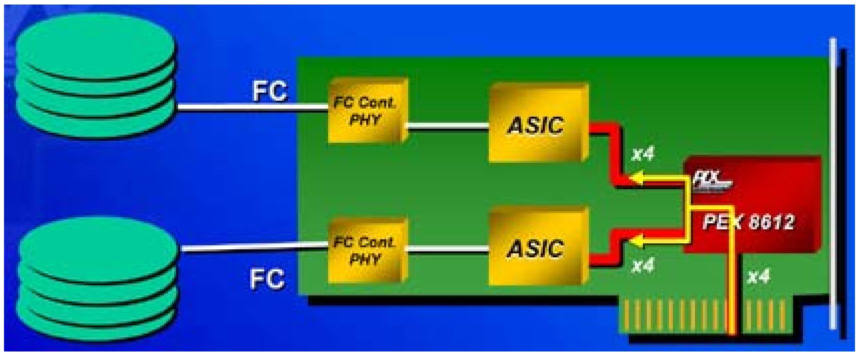


Figure 7. Dual cast fiber channel HBA

Without Dual Cast, the CPU must generate twice the number of packets, requiring twice the processing power. **Figure 7 above** illustrates a redundant storage array, where a PCIe Gen2 switch uses Dual Cast to store data on two RAID disk arrays. Additionally, the same card can be used for non-redundant applications

Summary

This new generation of PCIe switches supports Gen2 signaling, doubling the throughput per lane of the previous devices. Furthermore, new data-flow architectures are being deployed in these switches to optimize the bandwidth and memory utilization while minimize latency and power dissipation. Each of these features makes significant contributions to dramatic improvements in system and I/O performance in embedded systems.

*Steve Moore is senior product marketing manager at **PLX Technology**, Sunnyvale, Calif.*