Integrating Rack Level Connectivity into a PCIe Switch

Hot Chips 2013

Silicon without Software is Just Sand
Authors and Acknowledgments

- **Architecture Team**
  - Jack Regula
  - Mani Subramaniyan
  - Jeff Dodson

- **Executive Team**

- **Engineering Team**
The Opportunity

- PCIe is to servers as Ethernet is to the rest of the world
  - It is used to connect everything
  - Therefore extend PCIe for use as a converged rack level interconnect for IPC as well as for I/O

- Questions
  - Where best to bridge to Ethernet?
  - How to share expensive endpoints (SSD)?
  - How to architect modular compute platform where CPU blade is just processor and memory

- Benefits of PCIe as a rack level fabric are:
  - Saves cost, space, and power within the blades by removing NICs, HBAs, and disk drives from them
  - Low latency, high throughput IPC at low cost/power/area
The State of the Competition Within the Rack

- Competition is Data Center Ethernet and IB
  - 10GigE LOM per server moving to 40G and 100G uplinks
  - QDR (40 Gbps) IB now available

- PCIe Gen 3 at x8 has 1.6x QDR IB link speed w/o needing HCAs

- Existing PCIe switches and fabrics:
  - Non-transparent bridging is problematic because of the change in programming model
  - Don’t support redundant links well
  - Limited scalability due to limit of 256 BUS numbers
  - Marginal security
  - But do serve to prove the concept
Challenges

- Retaining PCIe compatibility
  - Only 100% standard PCIe packets on edge links
    - Existing CPU chipsets and endpoints can be used
  - Use Vendor Defined extension mechanisms on fabric links
    - Vendor Defined Message
    - Vendor Specific DLLP
    - Vendor Defined End to End Prefix

- Not changing the programming model
  - BIOS, drivers, OS, applications

- Non-technical challenges:
  - Added features must be marginal increase in switch cost or product isn’t economically viable
    - PLX has been shipping 96 lane PCIe Gen3 switch for >9 mos.
  - Normal product development schedule and budget constraints must be met
Architectural Goals

- The ability to share storage and communication end points among a large number of processors without driver changes
- High BW, low latency IPC using both Ethernet tunneling and RDMA over PCIe
- Remove PCIe scalability limits
- Support for diverse multi-stage fabric topologies
  - Load balancing over redundant paths
  - Congestion avoidance and management
  - Resiliency
- Complete compatibility with the PCIe specification
  - Use only standard PCIe TLPs on edge links
  - Use these plus PCIe Vendor Defined extension mechanisms on fabric links
System Solution Outline

Tell story via builds on next slide

- The Switch
  - 24 x4 ports plus an x1 port,
  - x4 ports combine to from wider ports
  - 4 types of ports:
    - Management port
    - Host ports that connect to servers
    - Fabric ports that connect to other switches
    - Downstream ports that connect to endpoints

- The Fabric
  - Single switch or multiple stages
  - Diverse topologies: Fat Tree, Mesh, Torus
  - Sweet spot = 3-5 stage fat tree >= rack scale

- The Management CPU
  - Manages and configures all so that standard server software (drivers, OS, applications) work w/o change
  - Handles only “events” during run time
System Block Diagram

The switch: 24 x4 ports + x1 port x4’s merge to x8, x16

MCPU configures and manages the fabric and IO. Handles “events” during run time

Host port includes RNIC for IPC

4 types of ports: Host Downstream Fabric Management

Single switch or multiple stages Support for multiple paths Diverse topologies ID routed tunneling

VF assigned to servers per color code

Shared EP
- PF
- VF
- VF
- VF
- VF
- VF
- VF
- VF
ID Routing Solves Multiple Problems

- **Scalability:**
  - An 8-bit Domain ID added to the PCIe ID to create a 24-bit Global ID.
  - Each Domain is a separate PCIe BUS number space
  - Fabric scales to (65K-256) nodes, each with 256 FUNs

- **Producer/Consumer Ordering over redundant paths**
  - In PCIe, memory requests route by address while completions route by ID but stay ordered because there is only one path
  - In ExpressFabric™, both types route by ID. Same choice of redundant paths is made so ordering is maintained. Different ordered streams take different paths so all paths are used.

- ID Routing simplifies and in some cases eliminates need for mappings between global and host-local address spaces

- PCIe Vendor Defined End to End Prefix is added when packet type is not natively ID-routed or destination is in a different Domain
Congestion Avoided via Pull Protocol

- **Pull protocol**
  - Read completions of pull protocol can take any path, as can other unordered traffic
  - Unordered traffic is round robin spread over redundant paths absent congestion feedback (>70% of IPC traffic)
  - Avoids paths where congestion is indicated
  - Work Request and Remote Read Requests Outstanding limits function as end to end sliding windows flow control and manage congestion on source and destination host links, respectively

- BECN using VD DLLP used to balance loads on redundant paths

- Credit based flow control for DMA WR VDMs to avoid deadlock
QoS

- TC Queuing on output queued switch
  - 4 TC Qs on x4, 8 on x8 or x16
  - Mix of priority and WRR scheduling
  - Interoperates and provides benefits with standard CPUs and endpoints as link partners, not just on fabric links
Integrated Messaging Engine

- Single HW messaging engine per 4_x4_port module of the switch
  - Virtualized and presented to hosts as multiple VFs
  - Each VF is an RDMA-NIC and can be assigned to an SI on the host

- Transfers can be done in two modes:
  - NIC mode tunnels Ethernet via standard TCP/IP stack
  - RDMA mode uses OFED stack and is a secure and reliable zero copy transfer from source application buffer to destination application buffer
  - Transfers use PCIe Vendor Defined Messages on fabric links

- Short packets are pushed; long packets are pulled
  - Congestion avoidance benefits vs 2^{nd} pass thru fabric
Short Packet Push

- **Source RAM**
- **Src DMA**
  - Host writes DMA doorbell
  - DMA reads Src Descriptors
  - Src Descriptors in DMA Cpl
- **Dest DMA**
  - DMAC sends Vendor Defined (completion) Message to Source
  - Short Message in WR VDM to Destination DMA
  - DMAC Write to Buffer
  - DMAC Writes Rx
  - Completion Queue
  - DMAC 0-Byte Read
  - All ACKs/Cpl of 0 Byte Read
- **Destination RAM**
- **HOST 1**
- **HOST 2**

**Color Key**
- **Host Involved**
  - MRd, MWr, CplD
  - PCIe Vendor Defined Msg

**Source**
- RAM

**Dest**
- DMA
Fabric Topologies

- Support for multi-stage switch fabrics includes
  - Routing and load balancing over redundant paths
  - Diverse topology support
    - Fat tree
    - 3D fat tree
    - Mesh
    - 2D and 3D Torus

- 3D Fat tree
  - Illustrating Domains
    - Each stands alone with its own MCPU
    - Share I/O within Domain
    - Communicate host to host across Domains
  - Align domain boundaries with packaging boundaries
    - Domain per cabinet
    - Domain per rack
3D Fat Tree

- Each grey box and its outside edge nodes is a standalone Domain & BUS# space w MCPUs.
- Domain fabric derived from 2n+1 stage fat tree by deleting n+1 columns of switches:
  - 3 stages => 1 stage; 5 stages => 2 stages
  - The former central rank forms inside edge rank
- Each Inter-domain switch can be a single chip or multiple stage fabric.
Domain with 72 x4 Ports
μServer Aggregation Fabric
Overcoming the SW Challenge

- Management CPU sleight of hand
  - MCPU manipulates fabric and IO configuration space to match standard enterprise IO programming model
  - MCPU configures mapping and routing tables in switches transparently to host software
  - Fabric and host ports are hidden from MCPU OS and BIOS so management application can manage them directly

- Leverage the existing SW infrastructure
  - TCP/IP Stack for Ethernet Tunneling
  - OFED Stack for RDMA
  - Supporting applications that use the APIs that feed these stacks

- PLX developed SW (SW layer diagram next slide)
  - Management application
  - Host to host drivers and adaptation layers
ExpressIOV™

- Sharing of I/O by the assignment of the VFs of SR-IOV endpoints, functions of multi function endpoint, and single function endpoints connected to the fabric to hosts within the same fabric Domain

- Applications:
  - Shared SSD within fabric scale cluster
  - Shared NICs or CNAs to connect ExpressFabric hosts into the “cloud”
  - Fabrics in which GPUs are dynamically assigned to hosts and communicate with hosts and each other either directly in memory space or via RDMA
Model for Sharing an SR-IOV VF

- MCPU configures ID-routed tunnels between hosts and the I/O VFs assigned to them
- Vendors’ PF drivers run on MCPU
- Vendors’ VF drivers run on hosts
Performance Benchmark

➢ Throughput
   ▪ Results from an IPERF benchmark on 2, 8 core Xeon servers connected back to back, Linux RHEL 5.6
   ▪ Gen 2 x8 and Gen3 x4 using memory copy DMA

<table>
<thead>
<tr>
<th></th>
<th>PLX NIC software on PLX NT</th>
<th>10Gbe Server Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>22 Gb/sec</td>
<td>9.8 Gb/sec</td>
</tr>
<tr>
<td>Receive CPU % overhead</td>
<td>11%</td>
<td>11.5%</td>
</tr>
</tbody>
</table>

➢ Total Latency (including HW & SW)
   ▪ 25 µsec PCIe
   ▪ 30 µsec on 10Gbe server adapter
## System HW Latency Estimation

### DMA and TWC Latency Comparison vs Number of Fabric Stages

<table>
<thead>
<tr>
<th></th>
<th>Number Fabric Stages</th>
<th></th>
<th>Number Fabric Stages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TWC PIO Write</strong></td>
<td></td>
<td><strong>Short Packet Push DMA</strong></td>
<td></td>
</tr>
<tr>
<td>Posted write thru fabric</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Land in memory</td>
<td>150</td>
<td>450</td>
<td>750</td>
</tr>
<tr>
<td></td>
<td>77</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>Total (ns)</td>
<td>227</td>
<td>527</td>
<td>827</td>
</tr>
<tr>
<td><strong>TWC PIO Read</strong></td>
<td></td>
<td><strong>NIC Mode Pull DMA</strong></td>
<td></td>
</tr>
<tr>
<td>forward path</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Read of host/RC memory</td>
<td>150</td>
<td>450</td>
<td>750</td>
</tr>
<tr>
<td>Return path</td>
<td>230</td>
<td>230</td>
<td>230</td>
</tr>
<tr>
<td>Land in register</td>
<td>77</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>Total (ns)</td>
<td>607</td>
<td>1207</td>
<td>1807</td>
</tr>
<tr>
<td></td>
<td>Parameters in ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host Read round trip read latency</td>
<td>230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch fall through latency</td>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RxCQ Interrupt Received</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total (ns)</td>
<td>910</td>
</tr>
</tbody>
</table>
Simulated Throughput vs Payload Size
Questions

➢ JRegula@plxtech.com