



ExpressNet™ 2.0

Rev 0.1

06/29/2011

1. Introduction

One Stop Systems (OSS) manufactures Computing Blades, which are capable of running various Operating Systems (OS) including, but not limited, to Linux, Solaris, and Windows. Each Computing Blade contains one or more processors, memory, clocks, terminals, disks, network interfaces and other input/output devices. These resources are managed by a single OS and are isolated from other Computing Nodes. The Blades can be aggregated together to create larger systems with the use of PCI Express Bridges and Switches.

A growing number of applications require multiple processors for greater processing power or higher reliability. These Blades can be connected together, in various topologies, to create lightly and tightly coupled multiprocessing systems. Tightly coupled systems share a singular global address space with different memory hierarchies participating on a common bus. Lightly coupled systems contain multiple address spaces interconnected via a high speed communication system. This white paper discusses OSS ExpressNet™ 2.0, an inter-blade communication software stack, designed for lightly coupled multiprocessing systems using an industry standard PCI Express I/O interconnect. In the context of this paper, a domain is defined as an instance of an OS.

PCI Express is a highly scalable, switched, point-to-point, serial I/O interconnect that is backwards compatible with legacy PCI and PCI-X. PCI Express maintains the load-store usage model of PCI without the side-band signaling for interrupts and power management; all information is transmitted in-band such as Quality of Service (QOS), Hot-Plug/Hot-Swap, Data Integrity and Error Handling. In-band traffic communication minimizes the pin count. Each PCI Express lane transmits at 2.5 GB/s per lane, per direction.

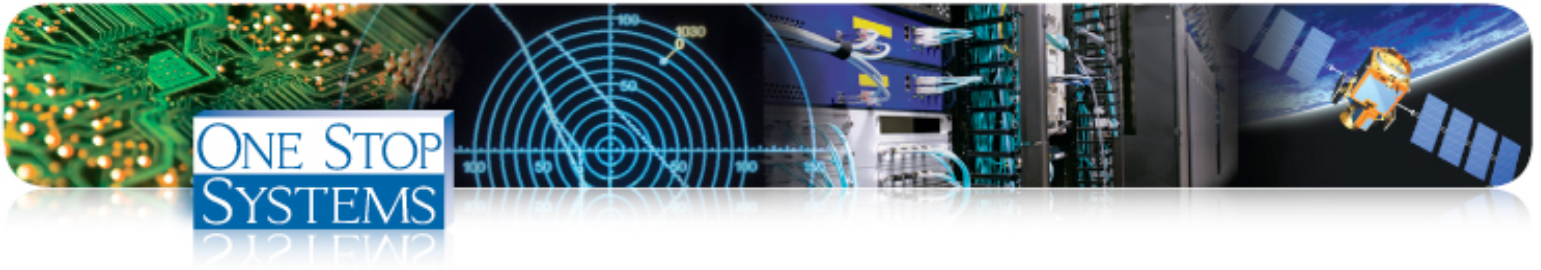
Processor coupling has been advanced with the advent of Non-Transparent Bridges. Prior to Non-Transparent Bridges loosely coupled multiprocessors required the use of custom ASICs for data and address arbitration. The need for custom ASICs is rendered moot with the use of Non-Transparent Bridges, in a lightly coupled multiprocessing system, and offers greater computational bandwidth by providing a mechanism for passing data between logically isolated host processors. These host processors can create symbiotic relationships within a given chassis. The greatest benefit of using Non-Transparent Bridges for loosely coupled



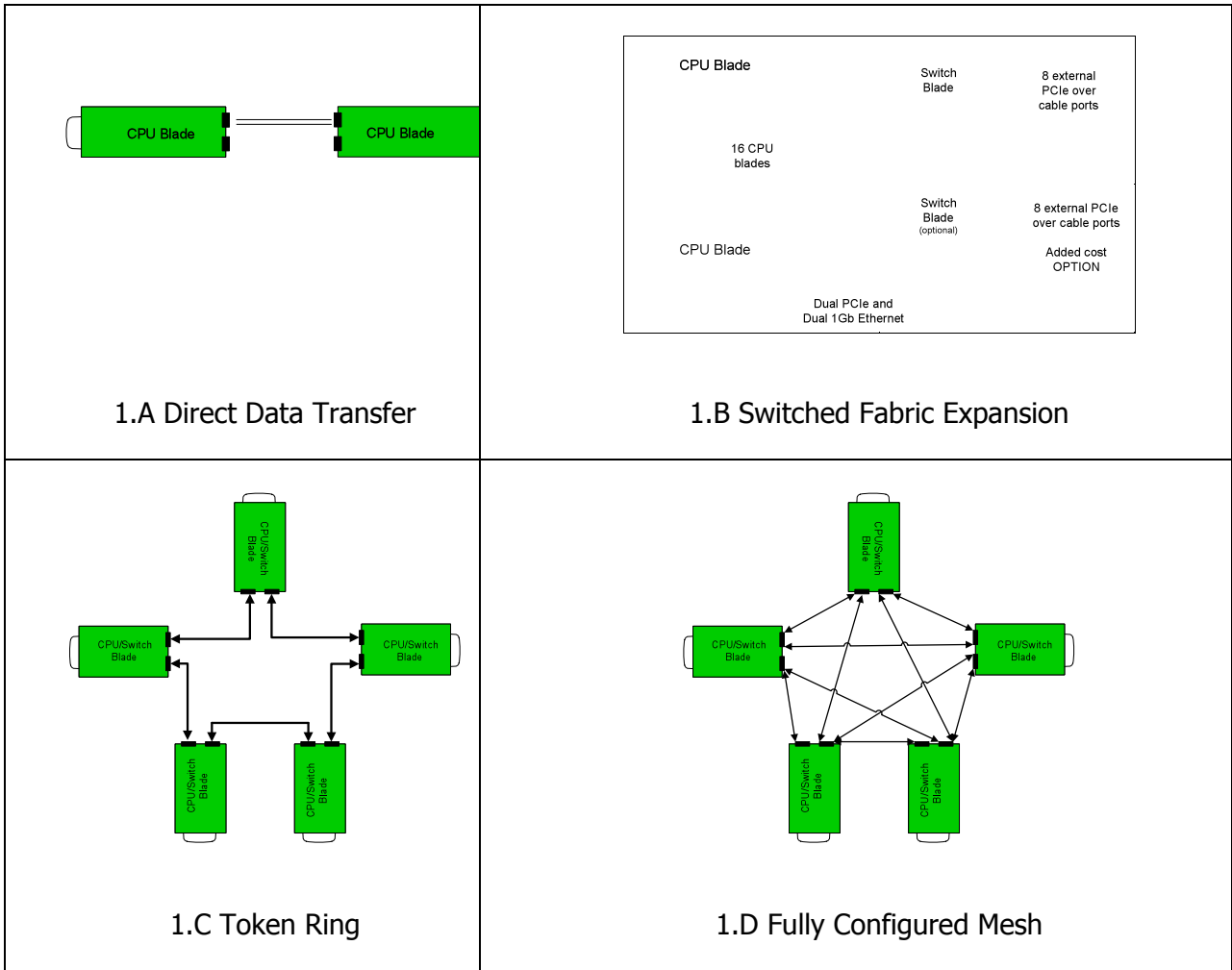
multiprocessing system is that it is based on a non-proprietary industry wide standard PCI technology. An ancillary benefit to using Non-Transparent Bridges provides a mechanism to couple disparate Operating Systems in a given chassis because the interconnection offers a standard communication protocol.

Devices on the PCI bus identify themselves as either switches/bridges or endpoints by their header format. Endpoints identify themselves to enumeration and discovery software by a "TYPE 0 Header"; while, bridges identify themselves by a "TYPE 1 Header". Bridges, by definition, provide a path between independent buses; therefore, both Non-Transparent and Transparent Bridges are functionally the same. They differ in the fact that Non-Transparent Bridges masquerade as endpoints to discovery software. Additionally, devices identify themselves by a Class Code register in the standard CSR header. A PCI-PCI Transparent Bridge uses a class code of 0x060400. A Non-Transparent bridge identifies itself as a RAM Controller via a class code of 0x050000. This identification reflects the fact that its typical use is to map memory space, containing memory rather than memory mapped I/O, from one address domain into another.

By masquerading as an endpoint, Non-Transparent Bridges insulate subsystems from each other and forward translated addresses across the bridge. Both sides of the bridge expose a separate "TYPE 0 Header" with BARs defining tunnels into each other's memory space. Non-Transparent bridges provide inter-domain communications primitives such as Doorbells, Scratchpad and I/O apertures using Base Address Registers (BARs). Doorbell registers are used to send interrupts from either side of the bridge. PCI Express capability structures allow these interrupts to be legacy interrupts (INTx) or Message Signaled Interrupts (MSIs). Scratchpad registers can be accessed from both sides of the bridge for communications. There are two sets of BARs: a Primary and Secondary BAR. Each BAR has a "setup" register, which defines the size and type of its apertures, and an address translation.



2. System Topologies



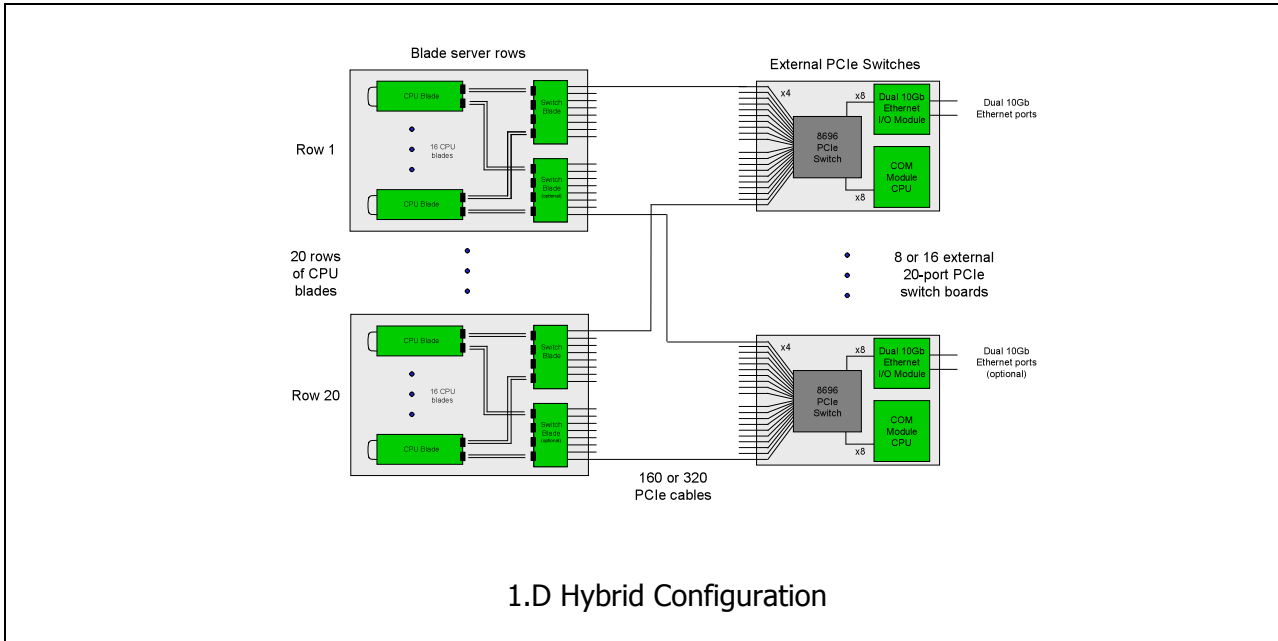
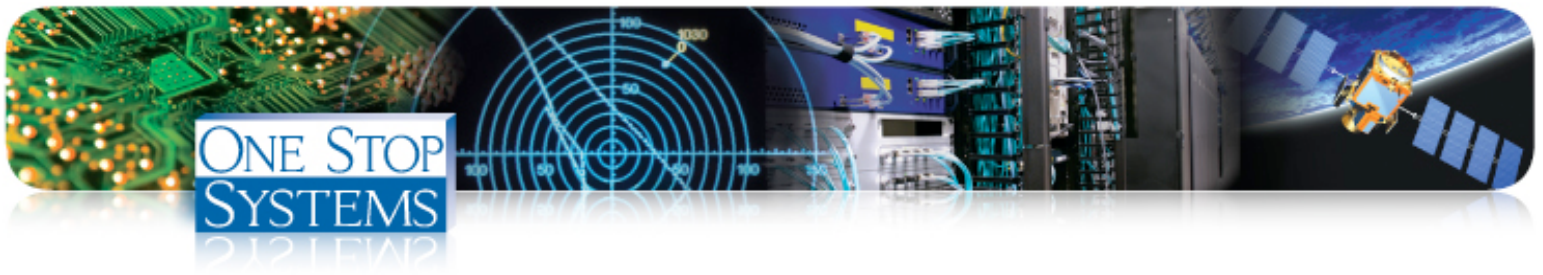


Figure 1 Topologies

Figure 1, illustrates the various system topologies addressed by OSS's ExpressNet™ 2.0. A doublet configuration is first depicted. A doublet is defined as a point to point direct connection which provides Direct Data Transfer (DDT)™ between computing nodes. Next a PCI Express Fabric Expansion is illustrated; a Switch is used to increase the number of blades that can be interconnected into a loosely coupled multiprocessing systems. Figure 1.C, shows a Token Ring which contains a bi-directional links between adjacent nodes. Each Computing/Switch Node transmits data sequentially in either clockwise or counter-clockwise direction. In the next topology, a Fully Connected Network is shown. Each of the nodes is directly connected to each other. Fully Connected Networks don't require the use of switching or broadcasting. The last topology is a Hybrid Configuration which is a mixture of the mesh, token ring and switched fabric configuration. Each Topology introduces its own inherent challenges for Inter-Blade Communication: hardware addressing, device enumeration and discovery, traffic routing, interrupt distribution and fabric management.



3. ExpressNet 2.0 Specification

Hardware Addressing

There are three ways in which addressing can be assigned for Non-Transparent Bridges: static, configurable and dynamic. Static hardware addressing can be assigned across the entire system using a combination of Chassis Serial Number Slot and Row Number. Configurable hardware addressing can be set by the user using EPROM settings. Dynamic hardware address assignment is designed around the boot-up or power on sequence of the bridge or after reset is encountered on the fabric. Hardware addressing is required to provide a Singular Global Name Space.

ExpressNet™ 2.0 supports a combination of static and configurable hardware addressing; this combination is coined as Re-Plug. To support true Hotplug requires the dynamic creation and tear down of PCI Express Bridges and Switches.

PCI Enumeration and Device Discovery

Since its conception, PCI has been host processor centric enumeration and device discovery scheme; a given system identifies all the devices, with their memory and I/O space. If additional processors are added to a given system, all processors will attempt to enumerate and memory map the entire system. This will result in conflicts, multiple processors will attempt to service the same system resources.

A series of control status registers (CSRs) are used by the discovery and configuration software. For Transparent Bridges, the CSR with a "Type 1 Header" informs the processor to keep enumerating beyond this bridge as additional devices lie downstream. These registers are primary, secondary and subordinate bus numbers, which, when programmed by the host, define the CSR addresses of all downstream devices. Endpoint devices have a "TYPE 0 Header" in their CSRs to inform the processor that no additional devices lie downstream. These CSRs include BARs used to request memory and I/O apertures from the host. Both header types include a class code that indicates the type (1 or 0) as well as sub class, device ID and vendor ID fields. Through this header format and addressing rules, the host searches through the entire topology until it reaches all endpoint. Until then, it reads all the class codes of all the discovered devices and assigns bus numbers to all bridges. At the end of this discovery, the system knows which devices are present along with their memory and I/O space requirements.

After discovery and device resource requirements, the range of addresses within a BAR are the addresses a port may respond to as a completer (target). PCI Hotplug provides a



framework for the addition/removal of hardware components from a running operating system without the need to reboot. Hotplug provides an industry standard methodology and various flavors of hot plug are implemented by most Operating System. As discussed above, the desire to dynamically add and remove devices, from a live system will require dynamic reconfiguration of bridges.

PCI Traffic Routing

There are two types of transactions in PCI parlance, Posted and Non-Posted transactions. Posted transactions do not require a completion. Traffic routing in PCI can be managed by three distinct schemes: address routing, ID routing and implicit routing. The Type field in a received TLP indicates if the packet is to be routed using an address, ID, or implicitly. Address routing uses the address located within its BARS to verify if the address falls within its ranges or it will forward the packet either upstream, downstream or reject the packet as unsupported. ID routing is based on the logical position (Bus Number, Device Number, Function Number) of a device function within the PCI bus topology. For non-posted transactions on a Non-Transparent Bridge, the bridge must own the transaction by replacing the originating device's BDF with its own. Implicit routing is based on the intrinsic knowledge of PCI Express devices. Routing packets upstream and downstream requires the existence of single PCI Express Root Complex at the top of the PCI Express topology. The PCI Traffic Routing scheme utilized by ExpressNet™ 2.0 is dictated by the intrinsic topology of a given system topology.

Software Architecture

Figure 2, illustrates the ExpressNet 2.0 Software Architecture. ExpressNet 2.0 supports higher level protocols such as TCP/IP, PXI-MC and MPI. These industry standard protocols are supported thru the ExpressNet 2.0 exported library, libXnet2.0, to user level applications and makes privileged calls to the ExpressNet 2.0 Driver. All of the topologies, in Figure 1, require a fabric manager to store and update the latest route patterns for data communication. These route patterns are essential for higher levels software stack, i.e. TCP/IP, for addressing packets to their desired destination. Either address or ID routing require this device and resource map for distributed memory communication. The Fabric Manager must maintain the resource management interfaces to manage bus resources globally in the system. A simple linked list data structure is needed to store the base and length of each aperture with a global mutex lock to protect the list of resource maps. Each local domain must communicate a desire to change its resource map or to ascertain the address and/or BDF for a given peer that it wishes to communicate with.

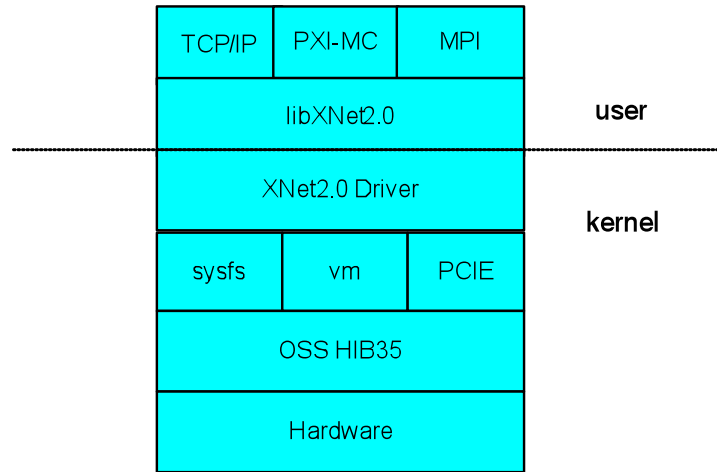
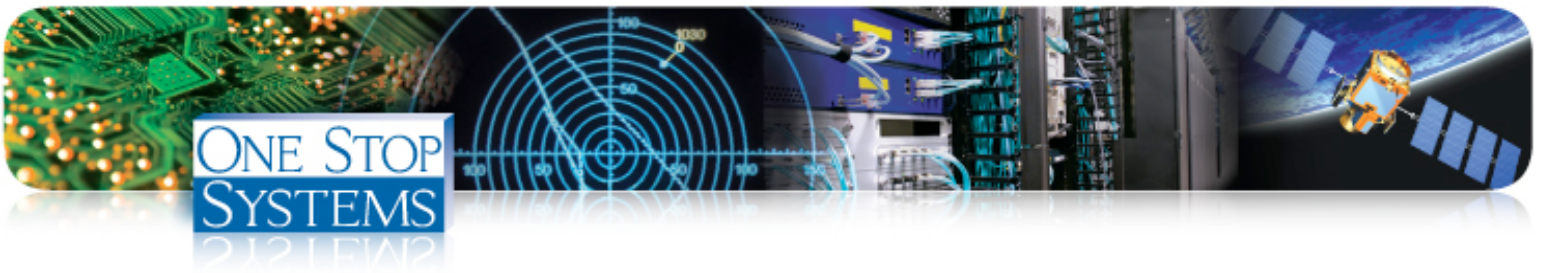


Figure 2 Software Stack

Packet Size

ExpressNet 2.0 optimizes the inter-blade communication for throughput and latency using DMA or Write Combining PIO, based on the message packet size. The ExpressNet 2.0 Driver exposes these optimization parameters to be application tunable.

4. Project Plan

Phases	Date
1) Direct Data Transfer over PCIe Gen 2.0	8/1/2011
2) 9 Port System w/ External Switch & XNet Demo Application	9/12/2011
3) 21 Port System	12/23/2011
4) 164 Port Hybrid System	1/23/2012