PCI Intelligent I/O Design for High Performance Servers

Byron Gillespie
Strategic Development Manager
Intel Corporation

Abstract

High-speed networks create pressure on servers for improved system level performance. The system I/O performance must keep up with the high-speed network protocols, such as 100-Mbit Fast Ethernet connections. Distributing the I/O processing between the host processor and an intelligent I/O processor balances the server processing for optimal performance. This paper shows system architects some of the issues associated with distributing I/O functions between the host processor and an intelligent network I/O subsystem. Distributed processing enables the system architect to off-load interrupt processing, device specific functions and data algorithms to the intelligent I/O subsystem. This offloads elements that don't scale with the host processor frequency, such as host processor reads from the PCI bus. Some of the issues associated with distributed processing that are addressed include device driver communication, data ownership/movement, and system configuration.

This paper focuses on an 100-Mbit Ethernet—intelligent I/O subsystem. Simulations and actual benchmark indicators formed the basis to create an intelligent I/O architecture such as the i960 RP processor. These indicators represent architectural decisions for determining system performance requirements.

Private or hidden PCI devices, memory spaces, and configuration are just a few of the challenges. In addition, comparisons of an intelligent I/O processor with alternative solutions show increased system level performance.

After this presentation, the system architect will have a good understanding of high-performance PCI system design and with distributed I/O processing subsystems. They will understand architectural issues, balanced distribution concepts and how to achieve increased levels of system performance.

Introduction

Today's client/server computing environment presents a challenge for maximizing server performance. Factors driving the need for intelligent I/O subsystems include:

- The stand-alone computing model is being replaced by networked computing.
- Networked computers increase the vast quantities of data the server systems support.
- Since servers CPUs also run user applications, they need more powerful storage interfaces for accessing larger disk storage areas, in addition to higher reliability offered by RAID storage.
- Simultaneously, the data sizes and type increasingly contain natural data elements like video or audio, in addition to text and graphics.

Increasingly powerful host CPUs must be relieved of I/O processing in order to perform at optimal levels. Creating intelligent I/O subsystems, based on high-performance embedded processors, allows the host CPU to perform more effectively.

An intelligent I/O subsystem handles much of the data congestion, balancing processing between the host CPU and the I/O subsystem.

These intelligent I/O subsystems:

- Enable maximum performance of the servers by utilizing all system resources to the fullest, most effective capabilities.
- Enable servers to scale by increased frequency in the case of symmetric multi-processing (SMP) systems.
- Remove the I/O bottlenecks that limit the scaleability.
- Enable the I/O to scale with the host processing capabilities (more ports equates to more I/O bandwidth)

Today's intelligent I/O application model and the overall performance achieved using a 100-Mbit Ethernet connection to an intelligent I/O card. An intelligent I/O application model improves overall network performance and system performance as compared to a non-intelligent application. The study model discusses the relationship between the PCI system and the intelligent I/O application. The system performance issues include:

- 100-Mbit Ethernet with intelligent I/O
- 100-Mbit Ethernet without intelligent I/O
- system level architecture and chipset variances

Each of these factors affect the performance of the different systems.

The next generation intelligent I/O application model is then discussed, focusing on the newest i960® microprocessor, the i960® RP processor, which was designed for high performance PCI systems. The i960® RP CPU provides the hardware enhancements not found in today's application model and uses these enhancements to achieve improved system level PCI performance.

Also discussed is the balance between the amount and type of processing performed by an intelligent I/O processor and the host processor. Another example of custom hardware on the i960® RP CPU, the messaging unit, improves the system level I/O performance.

Examples are given showing the different hardware units in the i960® RP processor and how they use difference PCI read and write commands to achieve increased PCI utilization.

Intelligent I/O Application Model

Today's intelligent I/O add-in adapter cards demonstrate intelligent I/O innovation. The adapter card used in this case study contains an i960® CF embedded processor, memory, PCI-to-80960 bus bridge, and 100-Mbit Ether peripheral component. Figure 1 describes the Intelligent I/O application model.

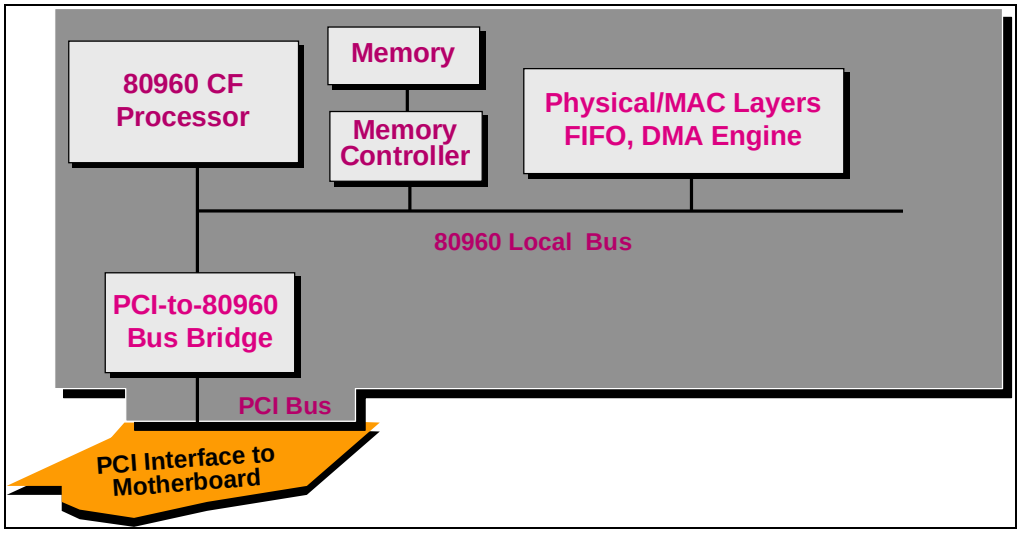


Figure 1 Intelligent I/O Application Model

The data flow for Ethernet traffic entering and exiting the server is shown in the following steps:

Receive Data:

1. The Ethernet packet arrives from the physical media into the LAN controller (Intel 82556 LAN controller).
2. The LAN controller is a bus mastering device, which transfers the packet from the physical media to the local memory on the add-in card.
3. An interrupt is generated to the i960® CF processor to notify the software that a new packet has arrived.
4. The i960® CF processor reads the header information and determines whether the packet is forwarded to the host memory or discarded.
5. When forwarding data, the i960® CF processor programs the DMA controller to "push" the data to the host memory.
6. When the data transfer is complete, the adapter card interrupts the host processor for notification of new data.

Transmit Data:

1. The host builds transmit packet information in host memory.
2. The host device driver notifies the I/O subsystem of new data to transfer via the PCI bus.

3. The intelligent I/O subsystem programs the DMA controller to "pull" the data from the host memory and transfers the data to the i960® CPU local memory. Once complete, the DMA controller notifies the 80960 CF processor of the data transfer via an interrupt.
4. The 80960 CF processor packetizes the data and adds the header information.
5. The 80960 CF processor performs the low level programming of the LAN controller to begin the data transmit.
6. The LAN controller reads the data from the 80960 local memory and serializes the data for the Ethernet physical media.

This is the typical data flow for an intelligent I/O adapter card with this type of architecture. Each time a packet arrives or a block of data is transmitted, the LAN controller generates an interrupt.

Role of an I/O Processor

With non-intelligent adapter cards, the interrupt is sent up to the host processor. This adds significant latency to the packet processing, increases the risk of losing packets from the LAN controller, and wastes PCI bus bandwidth performing low-level LAN controller programming.

One of the roles of an intelligent I/O processor is to reduce the host CPU interrupt handling requirements. This I/O processor compresses multiple receive packet interrupts into a single "data ready" interrupt. The "data ready" interrupt is generated when all the packets have been received and the entire block of data has been transferred to the host memory. The LAN controller receive interrupts, which would be destined for the host in a non-intelligent I/O system, are intercepted by the i960® CF processor which supports the low interrupt latency requirements from the LAN controller and filters the received packets to reduce the amount of unnecessary traffic on the PCI bus.

Cost of Interrupt Analysis

Since the intelligent I/O processor packs the data into complete data blocks for transfer, the overhead associated with each packet (such as the packet header, and checksum) can be striped from the pack to reduce the number of PCI bus transfers. This reduces the amount of wasted PCI bandwidth.

The interrupt analysis was performed with two different configurations to determine the host processing requirements.

The first configuration used one, two , three and four intelligent I/O adapter cards within the system. The second configuration used one, two, three, and four dumb I/O adapter cards. With the dump adapter cards, the host CPU utilization for a single adapter card was approximately 50% while achieving relatively high network throughput. The increased interrupt latency from the host CPU was one of the causes for the network failing to achieve the theoretical maximum throughput. This throughput difference appeared n the inter-frame spacing between the network packets transmitted. The spacing increased approximately 50 percent; 8 microseconds for intelligent I/O cards versus 12 microseconds for dumb I/O cards.

System Analysis

Besides the host interrupt processing, other factors to consider for intelligent I/O adapter cards include:

- host processor bus utilization
- contention for host memory
- PCI traffic to service the LAN controller interrupts
- PCI traffic for overhead of packet headers on individual packets

Every time the host processor performs a context switch for an interrupt, the code must be fetched from the host memory. This utilization of the processor bus is costly.

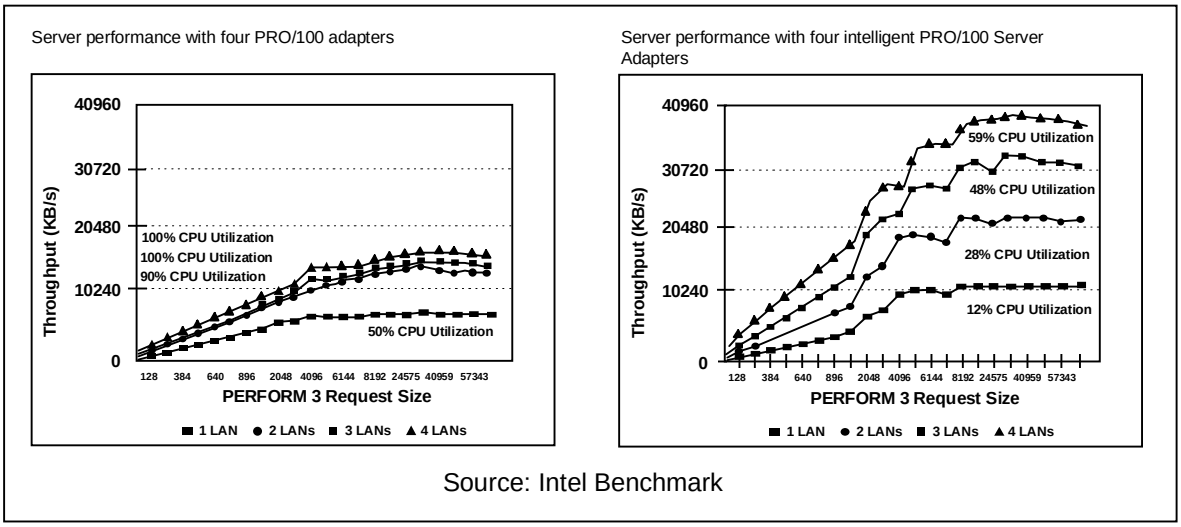
By the same token, the contention for the host memory affects the network throughput. Since the host processor builds the individual packets in this memory, the amount of contention for this bus increases.

Most PCI systems utilize a chipset to access the PCI bus. These accesses can burden the chipset with unnecessary PCI transfer requests. The access latency through the PCI bus can vary depending on the traffic loads required by the other PCI agents. Also, as the LAN controllers transfer the data packets across the PCI bus, the overhead for each packet is also transferred unnecessarily.

An intelligent I/O processor reduces the PCI and host memory traffic, host processor bus utilization, and improves the LAN performance by keeping the transmit pipe in the LAN controller full all the time. The receive packet buffering and data packing also improves PCI and host performance.

Performance Benchmark for Intelligent I/O Networking System

Figure 2 shows a benchmark using intelligent I/O versus non-intelligent I/O.



Source: Intel Benchmark

Figure 2, Intelligent I/O Benchmark

The chart on the left shows the LAN performance and host CPU utilization for configurations consisting of one, two, three and four dumb Ethernet card(s) installed within a system.

The single card installation shows relatively good LAN performance with approximately 50% host CPU utilization. However, when multiple dumb Ethernet cards were installed within the system, the data did not track with the single card installation. It is apparent that the host CPU became the bottleneck within the system. Therefore, multiple dumb Ethernet cards within the system did not improve the server's ability to support the LAN throughput capabilities of four 100-Mbit Ethernet cards.

The chart on the right show the LAN performance and host CPU utilization for configurations consisting of one, two, three, and four intelligent Ethernet card(s) installed within the same system.

With the single intelligent I/O card installation, the LAN Ethernet performance improved. This can be attributed to the improved interrupt latency offered by the 80960 CF processor. Also, the intelligent card had the ability to keep the transmit pipe full, reducing the interframe spacing between Ethernet packets. The host CPU utilization reduced to a mere 12% for a single intelligent I/O card configuration.

When multiple intelligent I/O adapter cards were installed, the LAN performance tracked well with the single card installation. Unlike the non-intelligent case where the host CPU is at full utilization with only 2 adapter cards. The intelligent I/O processor enables the server to support the LAN throughput capabilities of four 100-Mbit Ethernet cards.

These charts are based on a file server benchmark with the transmit case oriented. There is even a greater payback for receive traffic. The intelligent I/O reduces host CPU utilization on the received data by approximately 60% which is good for client server applications.

Focus On System Performance

The system configuration can directly effect the LAN performance, host processor performance, and PCI performance.

For example, the slot configuration of a server system contains many variables which may or may not directly effect the system performance. If a PCI card slot shares interrupts with other PCI devices, the behavior of the intelligent I/O subsystem can be affected. With shared interrupts, the host processor must look at all of the devices programmed to share an interrupt to determine which device generated the interrupt. This increases interrupt latency for all devices on the same interrupt line. Also, PCI cycles are required by the host processor when looking for the actual device which generated the interrupt. Therefore, PCI bandwidth is not utilized efficiently. Also, PCI reads do not scale with host processor frequency. For example, a host CPU running at 200 Mhz requires 20-30 CPU clocks where the PCI write cycles contain no additional delays for increased processor frequency.

The surrounding PCI devices also effect PCI performance and interrupt latency. For example, if a PCI chipset does not take advantage of the hints generated by the PCI command cycles (for example, memory read line, memory read multiple, memory write and invalidate), the host memory performance can reduce the PCI bandwidth utilization. The PCI specification version 2.1 helps make more efficient use of the PCI bandwidth, however, if the system contains components which mix PCI specification version 2.0 and version 2.1 devices, the system performance be affected.

Next Generation Intelligent I/O Application Model

The next generation of intelligent I/O add-in adapter cards demonstrate intelligent I/O innovation by using the PCI bus to an even greater extent. The adapter card used in this case uses an i960® RP intelligent I/O processor, memory, and peripheral component. Figure 3 shows the next generation intelligent I/O application model.

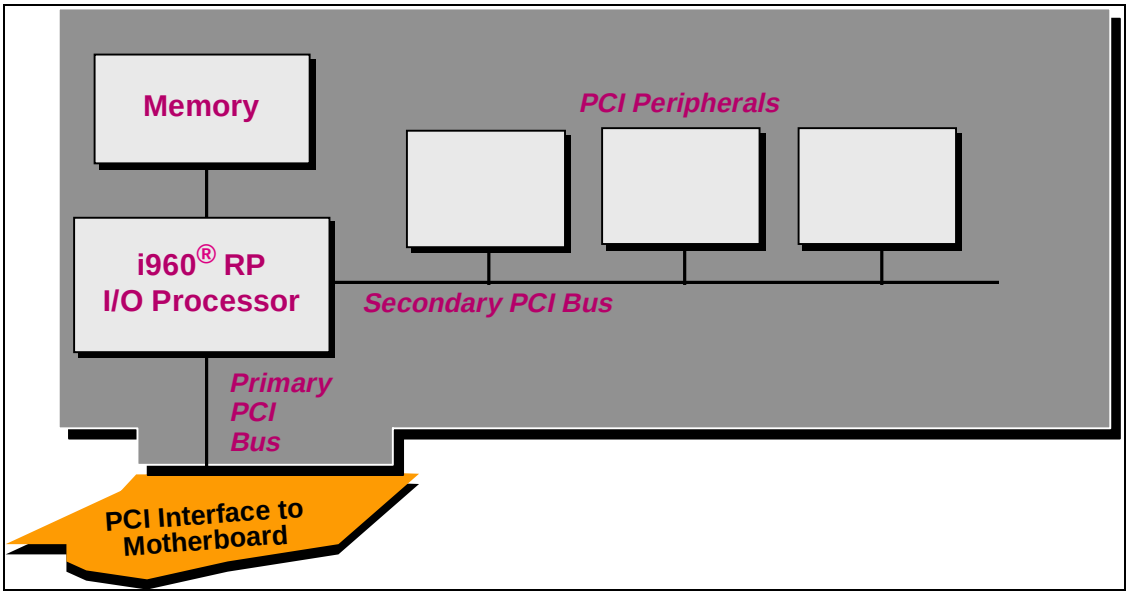


Figure 3 Next Generation Intelligent I/O Application Model

The data flow for Ethernet traffic coming into the server is shown in this architecture is very similar to the previous architecture. However, this architecture utilizes PCI peripheral components to provide the physical interface to I/O devices.

The PCI local bus standardization has migrated to an abundant number of peripheral components with a direct connection the PCI bus. These PCI components take advantage of the extended bursting capabilities provided by the PCI local bus and PCI commands, which provide hints to the slave devices for improving system performance.

Utilizing these components with an embedded I/O processor creates the intelligent I/O subsystem. The PCI bus has migrated onto the card via a PCI-to-PCI bridge. This enables system designers to take advantage of the lower cost peripherals available to create an high performance, intelligent I/O subsystem.

i960® RP Processor

The i960® RP processor architecture creates a PCI intelligent I/O subsystem. The i960 RP processor core is the i960 JF processor. Integrated around the i960 JF processor core are peripherals to complete requirements of the intelligent I/O subsystem. The i960 RP processor contains:

- PCI-to-PCI Bridge to support two PCI buses
- 31 VAX MIPS i960 JF microprocessor
- Dual address translation unit for direct access between PCI and the i960 CPU local bus
- Messaging unit for host to intelligent I/O inter-processor communication
- Three-channel DMA controller
- Integrated memory controller supporting DRAM, ROM, FLASH, SRAM or local bus peripherals
- I/O Advanced Programmable Interrupt Controller (I/O APIC interface)
- I^2C Interface for server management peripherals
- Interrupt routing support

Figure 4 shows the block diagram for the i960® RP processor.

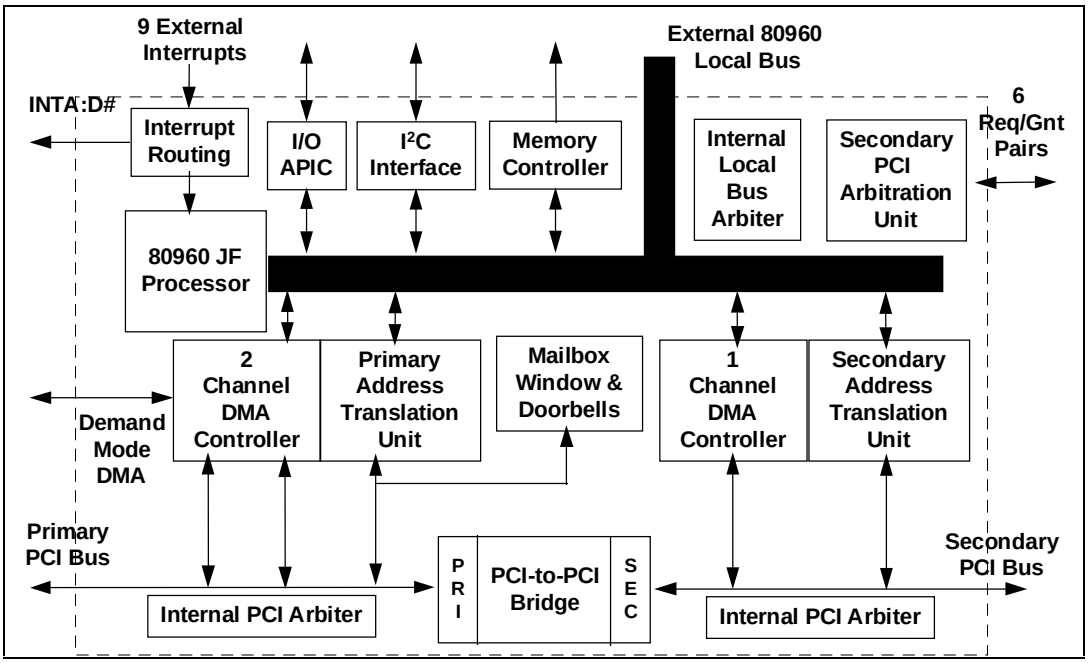


Figure 4 i960® RP Processor Block Diagram

The i960 RP processor supports the data flow requirements of the early intelligent I/O architecture in addition to supporting one additional path. The PCI-to-PCI bridge enables a communication path between the primary PCI bus and the PCI bus used for the intelligent I/O peripherals local to the add-in adapter card.

The i960® RP processor is designed with system performance in mind. In creating an intelligent I/O processor, extensive research defined the hardware mechanisms to support a high performance subsystem.

The first step in increasing system performance is the use of a PCI local bus on the add-in adapter card. This enables the customer base to use widely available PCI peripherals designed for high performance I/O. The PCI peripherals with bus mastering capabilities provide extended burst lengths to limit the amount of overhead for data transfers.

Secondly, taking advantage of the PCI commands enables the i960 RP processor to intelligently perform data transfers to increase PCI performance. Commands such as memory read line, memory read multiple, and memory write-and-invalidate can intelligently transfer data to and from the host memory.
These commands are further described in the next two slides.

Third, the i960 RP provides a hardware interface to simplify the communication between the host processor and the intelligent I/O subsystem. This hardware mechanism, called the messaging unit, simplifies and reduces the driver software to improve throughput. The messaging unit is described in the following slides.

Increasing PCI Read Performance

There are three PCI read commands. They are *Memory Read, Memory Read Line,* and *Memory Read Multiple*. The first PCI read command, *Memory Read*, does not provide any hints to the slave device to intelligently prefetch data to anticipate the amount of data requested by the PCI master. Utilizing the additional PCI *Memory Read Line* and *Memory Read Multiple* commands yields much higher PCI read performance. As shown in the chart, the memory read line command improves the read throughput to achieve approximately 50 Mbytes/sec. Figure 5 shows the performance obtainable by utilizing the various PCI read commands.

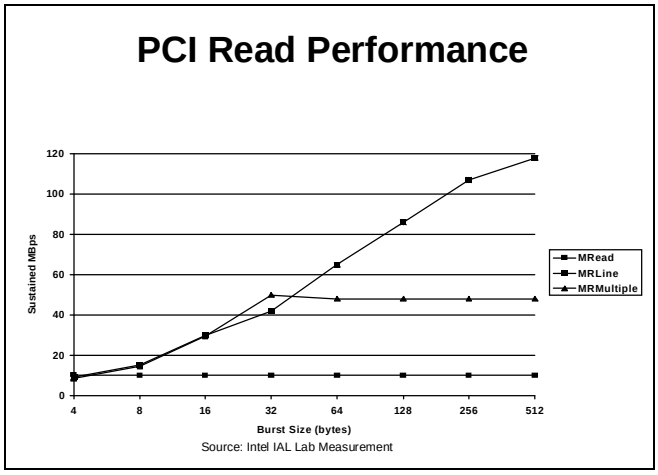


Figure 5 PCI Read Performance

The PCI memory read multiple command rapidly approaches the highest possible throughput achievable in a PCI system. The i960® RP processor intelligently utilizes these commands to prefetch data from the PCI bus. These commands in conjunction with multiple 64-byte queues enables an high performance intelligent I/O subsystem.

PCI Write Performance

Figure 6 shows the performance advantages of using the various memory write commands. The memory write performance varies depending on the processor data caches. For example, if a memory location in host memory is dirty, meaning the data is currently cached in the processor data cache of an L2 cache, the cache lines must be flushed from the caches prior to the data being transferred from the PCI bus. Therefore, the PCI performance of the memory write command varies extensively.

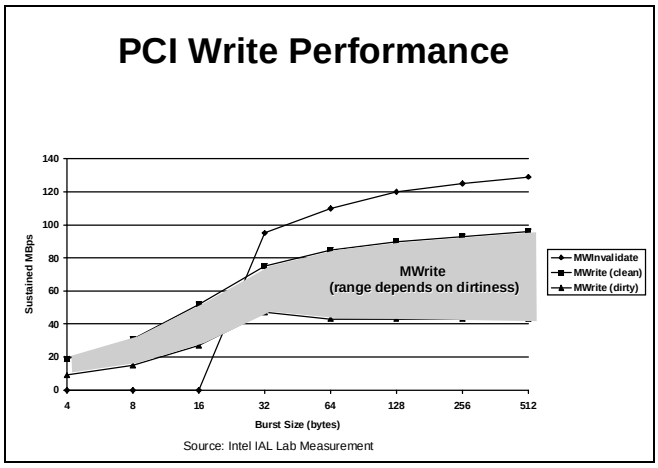


Figure 6 PCI Write Performance

However, the PCI memory write-and-invalidate command can achieve very high PCI performance because the data caches are not required to be flushed prior to the PCI memory write-and-invalidate command being executed. The data caches will simply invalidate the entire cache line tags in the data cache tag array.

The i960 RP processor supports the high performance memory write-and-invalidate command. Since the queues within the DMA controller and the PCI-to-PCI bridge are larger than the cache lines on the Intel Pentium® processor and the next generation Pentium® Pro processor, the i960 RP processor can easily guarantee transferring a full cache line successfully.

Intelligent I/O Interrupt Structure

The host processor and intelligent I/O subsystems have implemented a request and response queue model to perform messaging passing between the host and the I/O processor. This model greatly improves performance by reducing host interrupts, thus improving the host processing performance, PCI bus performance, and host memory performance.

The advent of symmetric multi-processing systems (SMP) brings forth new I/O performance issues. With SMP systems, spin locks waste CPU utilization, PCI local bus bandwidth, and create host memory contention. The spin lock is a case where the host processor polls the queues waiting for the other host processor (in an SMP system) to release the lock on the queues.

Request and Response Queue Model

The hardware messaging unit in the i960® RP processor implements the request and response queue model. This unit works in conjunction with the host OS and the intelligent I/O OS to alleviate the spin lock condition. The request and response queue model provides a software mechanism for reducing the number of interrupts generated to both the host processor and the I/O subsystem. Figure 7 shows the typical operation of the request and response queue model.

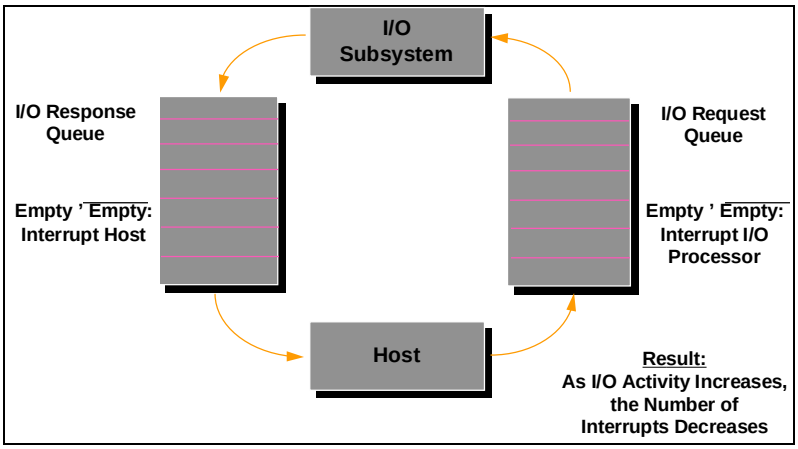


Figure 7 Request and Response Queue Model

The operation of these queues is as follows for host initiated transactions:

1. The host processor builds Event Control Blocks (ECB), which are data structures used to transfer packets between the adapter and the server memory. Each ECB contains a pointer (defined in host memory), which is the address to access the ECB.
2. Once the ECBs are built, the pointers (for each ECB) are written into the I/O request queue by the host processor to enable the I/O subsystem to perform the operations specified by the ECB.
3. An interrupt is generated to the I/O processor only when the queue transitions from empty-to-not-empty.
4. Upon the I/O subsystem receiving the interrupt, the I/O processor reads the pointer and transfers the ECB to local I/O memory.
5. The I/O processor processes the ECB and upon completion, places the ECB pointer in the I/O response queue.
6. The host processor reads the pointer from the I/O response queue and completes the transaction.
7. The ECB pointer is then allowed to be reused for another host transaction.

These steps are very similar for I/O subsystem initiated transactions. There are additional I/O request and response queues for the I/O subsystem to initiate transactions to the host processor.

I960® RP Processor Messaging Unit

The i960® RP processors messaging unit contains specialized hardware to provide an optimized host access interface. The messaging unit, from the PCI interface, appears as a simple memory port. The host processor accesses the port through a single PCI address interface. The host or external PCI agent accesses the same address for reading and writing data through the port. The messaging unit hardware implementation accesses circular queues implemented in memory located in the i960 RP CPU address space. These queues contains head and tail pointers, managed by a combination of the i960 RP CPU software and the messaging unit hardware. Figure 8 shows the circular queue structure for the i960® RP processor messaging unit.

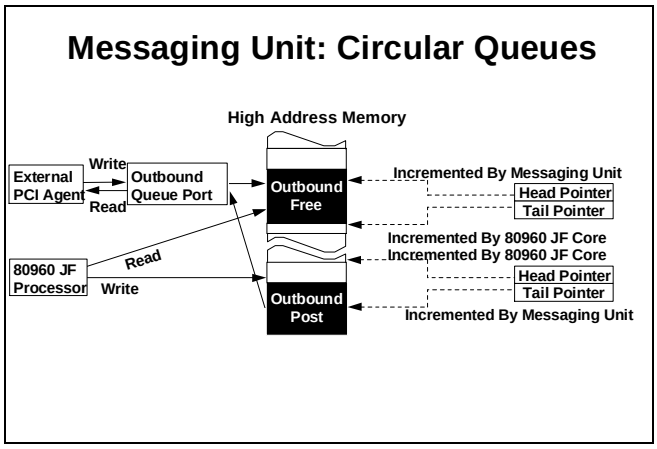


Figure 8 i960® RP Processor Messaging Unit

The following describes the action taken by the messaging unit hardware and i960 RP processor software for the various transaction types:

PCI write to the outbound queue port: The data value from the PCI write is written to the i960 RP CPU local memory specified by the address in the outbound free port head pointer. When the data is written, the messaging unit hardware will increment the outbound free head pointer register. The i960 RP processor is responsible for maintaining the outbound free queue tail pointer. An interrupt is generated only when the head pointer increments to the tail pointer, meaning the queue is full.

PCI read from the outbound queue port: The PCI read is signaled by a retry from the messaging unit. Meanwhile, the data value for the PCI read is retrieved from the outbound post queue pointed to by the outbound queue tail pointer. A single data word is read by the messaging unit and held in the outbound queue port. When the PCI agent returns after the retry, the messaging unit will return the data to the PCI master and increment the outbound queue tail pointer. The i960 RP processor is responsible for maintaining the outbound queue port tail pointer.

i960 RP processor read from the outbound free queue: The i960 RP processor is responsible to read the address pointed to the by the outbound free tail pointer. Upon completion, the i960 RP processor must increment the outbound free tail pointer.

i960 RP processor write to the outbound post queue: The i960 RP processor will write data to the outbound post queue. The address it writes to is contained in the outbound post head pointer. Upon completion, the i960 RP processor must increment the outbound free tail pointer.

The i960 RP processor messaging unit contains an identical structure for the inbound post and free queues. This structure is for the host processor to post new ECBs for the i960 RP to process.

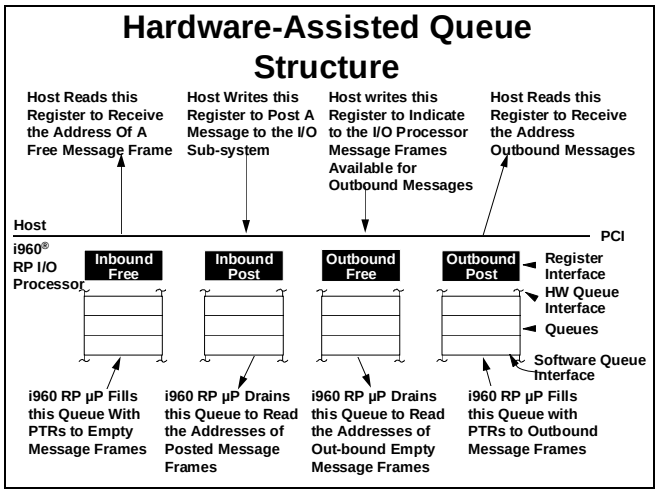


Figure 9 Messaging Unit Queue Description

Figure 9 shows the four different queues contained in the i960® RP processor messaging unit. These queues are:

1. Inbound free queue
2. Inbound post queue
3. Outbound free queue
4. Outbound post queue

Two of the queues are dedicated for inbound messages. The inbound messages are for the host processor to send messages to the intelligent I/O subsystem. These messages are pointers to ECBs in host memory that the i960 RP processor must execute.

Two additional queues are dedicated for outbound messages. The outbound messages are for the i960 RP processor to send messages to the host processor. The messages are pointers to ECBs in memory that the host processor must execute.

This free/post queue structure prevents overflow conditions because both the host processor and the I/O processor pre-allocates the empty ECBs. Therefore, when a processor reads an empty queue, the value returned is a -1. This notifies the respective processor the queue has no ECBs to process or available to use.

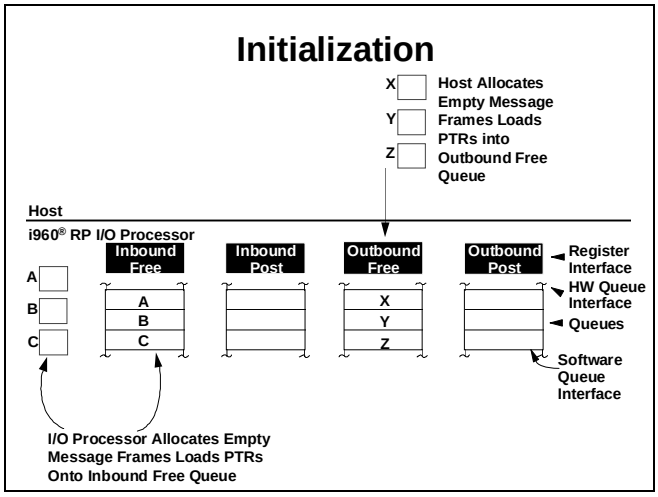


Figure 10 Messaging Unit Queue Initialization

Upon reset, the i960 RP processor messaging unit initializes all of it registers to zero. The i960 RP software must configure the messaging unit queue sizes, queue base address registers, head and tail pointers for each of the queues prior to the host processor accessing the inbound and outbound queue ports.

Once initialized, the host processor will allocate empty message frames and load the pointers into the outbound free queue. These messages are used by the i960 RP processor for generating messages the host processor must execute.

At the same time, additional message frames are allocated and the pointers are loaded in the inbound free queue. These messages are used by the host processor for generating messages the i960 RP processor must execute.

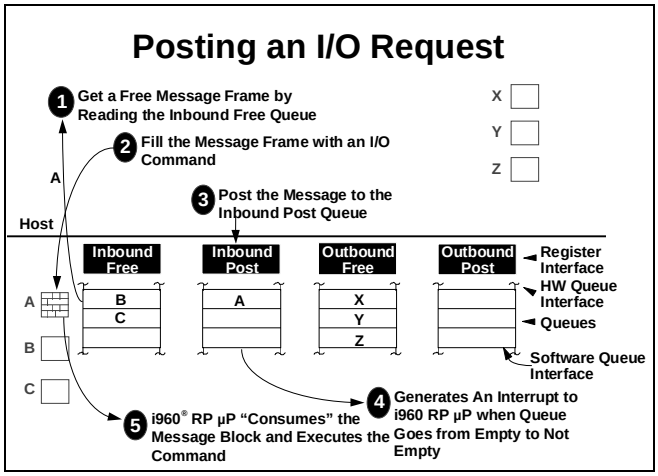


Figure 11 Posting an I/O Request

Figure 11 shows the procedure for posting an I/O request. When the host processor needs to post an I/O request to be processed by the i960® RP processor, the following describes the actions performed:

1. The host processor reads from the inbound free port to get the address of an ECB to fill.
2. The host processor fills the message frame with an I/O command
3. The host processor writes the address of the ECB to the inbound post queue.
4. An interrupt is generated to the i960 RP processor when the queue goes from empty to not-empty.
5. The i960 RP processor executes the commands in the ECB.
6. When processed, the i960 RP processor places the address of the ECB at the end of the inbound free queue.

After processing a ECB from the inbound post queue, the i960 RP processor will verify if additional messages have been posted to be executed. The i960 RP processor will continue reading from the inbound post queue until the queue becomes empty.

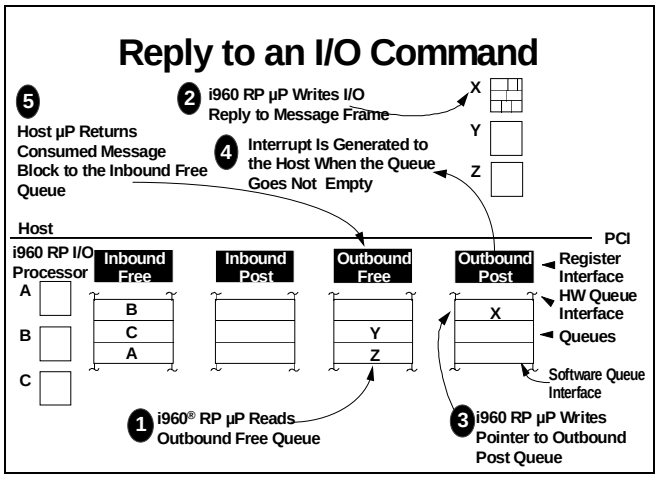


Figure 12 Reply to an I/O Command

When the i960 RP processor needs to post an I/O request or reply to an I/O command, as shown in figure 12, the following describes the actions performed:

1. The i960 RP processor reads from the outbound free port to get the address of an ECB to fill.
2. The i960 RP processor fills the message frame with an I/O command
3. The i960 RP processor writes the address of the ECB to the outbound post queue.
4. An interrupt is generated to the host processor when the queue goes form empty to not-empty.
5. The host processor executes the commands in the ECB.
6. When processed, the host processor places the address of the ECB at the end of the outbound free queue.

After processing a ECB from the outbound post queue, the host processor will verify if additional messages have been posted to be executed. The host processor will continue reading from the outbound post queue until the queue becomes empty.

Summary

Distributing the I/O interrupts to the I/O subsystem for server I/O enables the enterprise class I/O to PC servers. An intelligent I/O subsystem enables high performance I/O and removes the I/O bottlenecks for servers. Distributing the I/O processing improves the I/O throughput and relieves the congestion on the PCI bus and system memory. It also enables servers and the server I/O to scale with the frequency of the host CPU.

An intelligent I/O processor, such as the i960® RP processor, exploits the advantages of PCI architectures by providing a high performance processor with PCI specific integration. The i960® RP processor architectural requirements, taken from applications ranging from networking and storage add-in cards to server motherboard systems, comprise today's intelligent PCI I/O subsystem.

Biography

Byron Gillespie
Strategic Development Manager
Intel Corp. Enterprise Computing I/O
(602) 554-2653

Byron Gillespie works as a strategic development engineer for Intel's Embedded Processor Division. He is responsible for working with customers to define the requirements for future i960® microprocessor products. Before coming to Intel in 1991, he had eight years experience writing software for embedded avionics applications. The embedded applications used a variety of Intel processors including the superscalar i960® CA microprocessor. Gillespie received a B.S. in computer science from Northern Arizona University in 1983.