Overview of Intel® QuickPath Interconnect System Initialization

ABSTRACT: After two components based on Intel® QuickPath Interconnect (Intel® QPI) technology establish a connection between themselves, they can reliably send information back and forth. Consider this quote by Henry David Thoreau: "Reacting to the news that Maine and Texas had been connected by telegraph lines—The point at issue is not whether Maine and Texas may now talk to one another, but rather whether they have anything significant to say." What shall we do with that connection? This article describes the first steps in the effective use of those communication pathways, the process of initializing the entire system so that the system can perform the overall function for which it was intended.

Basic Concepts of System Initialization

The beginning of the platform initialization process is to apply power, stabilize all critical clocking circuits, and then control the release of various *reset* signals so that the computing system will begin operation in a predictable and controlled manner. In this context, *Reset* is defined to be a set of hardware-based events that establishes an initial hardware state. *Initialization* is defined to be the set of instruction sequences that follow Reset and which prepare the hardware for execution of boot firmware. The *boot firmware* then prepares the system for the loading of and transferring control to an operating system. A significant goal of these hardware and boot firmware activities is to get their tasks completed and get out of the way so that control can be handed over to an operating system.

Platforms based on the Intel QuickPath Architecture are no different in that basic regard, but they do have several additional capabilities that are vital to the task of getting started. These interconnect features are essential in supporting the wide variety of platform configurations that are served by these systems. These features also introduce new areas that the boot firmware in particular needs to be concerned about, things that either did not exist in previous platforms or that were not as configurable in how they operated. These areas include:

- *Link resource initialization* – Some links may require configuration and explicit startup instructions; others may come up automatically. In either case, some links may fail to initialize at all, or could come up in a reduced operational state.
- *Topology discovery* – Certain devices may or may not be populated, or may not be fully functional. A policy for discovering all the interconnected resources must be devised and implemented.
- *Distributed memory configuration* – There typically would be multiple memory controllers, each controlling a differing amount of physical memory.

In all of the above areas, an Intel QPI system opens up new possibilities in how these issues can or should be handled. The following sections outline the basic steps for many of these initialization actions. The approach taken in this article is to define and describe these building blocks of the initialization process, but we do not prescribe a specific formula for how the overall boot process is performed. There are several ways to put these blocks together and still accomplish the overall task of getting ready to run the operating system. Those choices are left to the platform architects and development teams.

Before Software Runs

Of course, the goal of initializing the computing system is to enable useful work to be conducted, to allow the processors to execute the operating system and application programs that they are intended to execute. In order for that to take place, many things must be operational prior to the initial code fetch from the reset vector. As we shall see, there are many possibilities to choose from and several things that the hardware design takes care of prior to handing over control to the boot firmware.

Multiple Reset Domains
Systems based on Intel QPI are expected to continue to operate in the presence of certain conditions that could result in system downtime if not for the error recovery features. They also are called upon to support sophisticated operating system features such as dynamic reconfiguration and partitioning. In order to facilitate those advanced features, the hardware is designed to be able to independently control the resetting and initialization of portions of the hardware without disrupting other portions. As we shall see in later sections, the control of these resets may take place by portions of the Intel QPI layers, for example the link layer may force a reset of the physical layer. Or these reset functions may take place under control of boot firmware or other layers of software, using control registers to influence the hardware.

Boot Modes
Platform designers may have multiple choices in terms of how the boot firmware device is connected to the system. A common choice for smaller-scale systems is to use a firmware hub (FWH), which is connected to the ICH device and then to the legacy IOH device (LIOH). The LIOH in turn is connected through at least one Intel QPI link to the processor(s).

Figure 1 depicts a two processor and single IOH configuration with the FWH connected to the only IOH, which by default is the legacy IOH. We have seen this basic diagram before; what is added here are the ICH and FWH connections. The hardware typically would initialize itself such that transactions from either CPU

can reach the FWH.

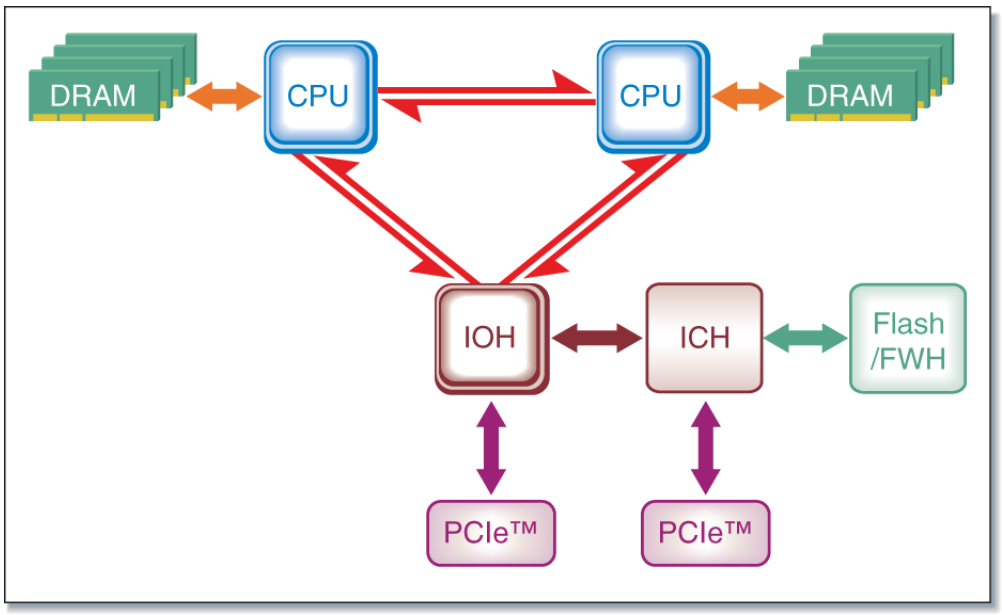


Figure 1 Dual CPU, Single IOH System Topology

Figure 2 shows a more complex four processor dual IOH configuration. In this configuration, two of the CPUs are not directly connected to the LIOH but instead are one Intel QPI “hop” distant from the LIOH.

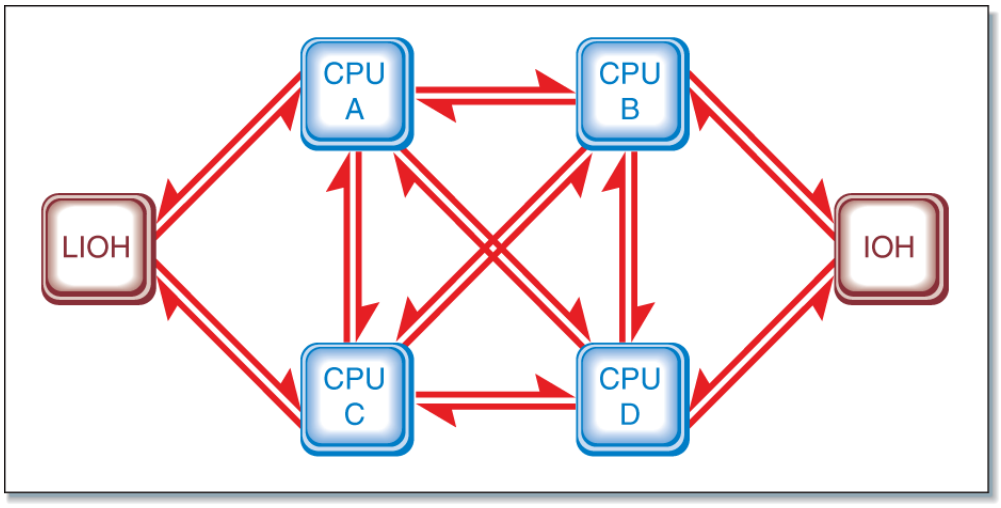


Figure 2 Four Processor, Dual IOH Topology

In both of these platform examples, the processor hardware configures itself so that initial code fetches will be successful. The hardware initializes itself since the boot firmware is not executing yet. Booting in this manner is termed the Intel QPI

link boot mode since the firmware is located across an Intel QPI link relative to the processor.

Other platform-specific boot modes may include a firmware storage device connected directly to the processor socket. Another option is to rely on an external system service processor that has the ability to independently initialize internal Control and Status Registers (CSRs) of all devices and to control the reset and bring-up of the platform. In the rest of this article, we focus only on the Intel QPI link boot mode because this is the most common method for small and medium-scale systems as shown in Figures 1 and 2.

Bringing Up the Link

Before any software can execute, a connection pathway to the memory device that contains the boot firmware code must be established. Since platform configurations may vary, the hardware itself establishes this pathway so that the boot firmware can then proceed. Note that in some cases the “hardware” could also encompass microcode or other unique initialization functions that may exist as part of the overall processor device. The processor hardware and any low level microcode work together to create the connection used during the rest of the boot firmware startup process.

One step in this process is to initialize the Intel QPI link(s) that are in the path between the processor and the boot firmware storage device. Various platforms may make different provisions in this regard. Simple configurations may allow only a specific pathway while more complex platform topologies may allow multiple options for making this connection. In any event, the link(s) are brought up to the L0 operational state. Once this has been done, the processor core can issue instruction fetches which flow across the interconnect to the hardware controlling the firmware storage device.

In addition to the link initialization, various control and status registers are set to default values so that basic functionality of the device is accomplished. Part of this is noting any functions that are not operational. For example, any Intel QPI links that fail to train to the L0 state are noted. This is the case if a processor socket is not populated. Any other processors note that the links going to that unpopulated socket are not operational.

Setting the Route

Recall that transactions on the Intel QPI fabric are routed according to a destination Node ID. Every function therefore is assigned a unique Node ID and the routing tables that define the rules for routing the transactions are initialized. Smaller-scale platforms may make some simplifying assumptions in order to facilitate this process. Larger-scale configurations generally allow more flexibility. In either case, in this link boot mode, the processor device typically must

establish these Node IDs and routing tables without the use of boot firmware, at least until an initial pathway has been created and the boot firmware can begin execution.

Selecting Processors for Specific Tasks

Each processor socket in these platforms typically contains multiple CPU cores that can each execute their own instruction stream independently. At system initialization, however, some steps must be performed in a very controlled manner. This requires that certain functions be limited to only execute on a subset of the total number of processor cores. That subset could be as small as one in some cases. Generally speaking, any of the available cores can be selected to perform these functions.

Processor Bootstrap Processor (PBSP)
This is the logical processor (a single core) responsible for initialization and operation of the socket in which the PBSP resides.

System Bootstrap Processor (SBSP)
This is the logical processor (a single core) responsible for the bulk of system-level initialization. This encompasses multiple processor sockets, multiple IOH devices, and other system resources such as memory, which typically are initialized prior to handoff to the operating system.

Application Processor (AP)
Application processors (APs) are those processor cores which do not participate in the system initialization process unless they are directed to do so by the SBSP.

What Is Out There? Topology Discovery

When the system first comes out of reset and the boot firmware starts executing, one of the primary tasks is to discover what resources are available and how they are interconnected. The various hardware functions play a role in this, as do the PBSPs while they configure the processor sockets individually. It is the SBSP however that then examines the information it has available, and goes through a methodical process to determine what resources exist, which ones are functional, and then configures those resources in preparation for the operating system. This process is referred to as *topology discovery.*

Intel® QuickPath Interconnect Layer Setup

After determining the system topology and the range of available resources, the SBSP has the task of configuring specific functions of each of the Intel QPI layers. Some default settings are established by the hardware and these are sufficient for

at least a subset of the links to be used for initial boot firmware functions. Beyond that it is up to the boot firmware itself, running on the PBSP, to configure the various control registers and to judiciously control the reset of various layers in order for new configuration settings to take effect.

Link Speed Transition

The physical characteristics of the interconnect generally require specific equalization settings to be established on a per-link basis before full speed operation can commence. Therefore, the Intel QPI links by default will come up in an initial slow mode, which is a 66.66-megatransfers/second (MT/s) rate. At the slow mode transfer rate the links will still train and operate in the L0 state, but there is no need for any platform-specific tuning. Part of the boot process then is to determine the desired full operational speed of the links, configure the necessary resources that control that speed, and then perform a controlled physical layer reset to bring the new speed into effect.

Rest of Boot Firmware

Once the system initialization process has completed all of the steps outlined above, all of the Intel QPI–specific items will have been configured and are operational. The communication fabric can then be used by the boot firmware in order to perform the remaining initialization functions that are not directly involved with the Intel QPI fabric. A big part of the remaining work is the task of memory system initialization and configuration. The final step is to synchronize all processors and turn over control to the operating system, optionally through the extensible firmware interface (EFI) layer. Such tasks are well understood in the context of multiple processor systems and they are not specific to Intel QPI. As such they will not be discussed in the rest of this article.

Actions Prior to Boot Firmware Execution

The primary action to be taken at this point is to establish the path to the device that contains the boot firmware. As shown in Figure 1 and 2, a boot firmware storage device (FWH) is typically connected to the ICH device that in turn is connected to the legacy IOH (LIOH). Other system configurations are possible, especially for large-scale servers, but for this discussion we shall consider only the Intel QPI link boot case. The issue then is how to configure the processor's system interface logic along with the LIOH logic so that the core's code fetch operation succeeds and the boot firmware can continue with platform setup.

Multiple Reset Domains

The first step is to come out of reset so that the hardware can begin executing. Because certain features require the ability to independently control reset to different hardware portions of the machine, Intel QPI devices are designed to accommodate multiple forms of reset.

Cold or Power-Good Reset
This type of reset affects all hardware elements of the platform. As the name implies, Power-Good reset occurs with the de-assertion of a hardware signal to the device when the power is stable. All hardware elements revert to their default state and no information is preserved from any previous power on state.

Device Level Reset
In this form of reset, the power supplies to all devices have remained steady, so the state can be preserved to some extent across this reset event. This type of reset allows system firmware to perform some amount of component initialization that is then “sticky” across the device reset.

Device reset can be caused by hardware signals if the platform is designed to manipulate the reset signals externally. The most common form of a device reset would be triggered by the boot firmware writing to a control register in the LIOH (or possibly elsewhere). This action then causes an external signal to toggle and resets the device.

Physical Layer Reset
As a general rule, Intel QPI devices have the ability to independently reset various layers of the interface. In particular the link and physical layers are able to be reset without affecting the state of the rest of the machine. A physical layer reset is used when the intent is to force that layer to reestablish proper low-level synchronization between the two components, possibly involving a change in some of the operating characteristics of that interface. Some typical uses of this form of reset include: a change in operational speed, recovery from severe bit error events, or recovery from loss of part of the physical connection. Reset of the physical layer has no impact on the actual information being conveyed across the link since all error checking, flow control, and data interpretation is done at the higher layers.

Physical layer reset can be caused by any of several possible actions. Writing to a physical layer control register is one common method used by the boot firmware. Losing the incoming forwarded clock is interpreted as an Inband reset and causes a physical layer reset. Internal mechanisms also exist that allow the link layer logic to force a physical layer reset, typically triggered if excessive CRC errors have been detected.

Link Layer Reset

Link layer reset is generally used whenever certain parameters of the link layer must be changed. After a Power-Good reset, the link layer capabilities are in their default state. In order to use certain functions such as low-power modes, non-default credit allocations, or Rolling CRC mode, the link layer is reconfigured and then reset. The new settings take effect after the reset.

Link layer reset is done by writing to a control register. Note that more care must be taken when resetting the link layer than with a physical layer reset. This is because the link layer does maintain information that is critical to the correct information exchange between agents. Improper use of this reset would cause incorrect system operation, so link layer reset is used judiciously.

Partition Level Reset

The final level of reset is a reset applied to an entire operating system partition. Intel QPI platforms can support running multiple instances of an operating system on different portions of the hardware. For example in Figure 2, two of the CPUs could be running one partition while the other two run a second partition. It is also possible that different operating systems coexist, each using its own resources. In such an environment the platform can provide mechanisms for resetting one of the partitions without affecting the remaining resources. From the Intel QPI fabric perspective, this ability relies upon the multiple reset domains described above. From the operating system perspective, there is obviously much more involved and that is outside the scope of this article.

Boot Modes

Processors based on Intel QPI technology are designed to handle various boot modes as dictated by the needs of the platform environment they are required to operate in. For example a small dual-processor socket system may only support the Intel QPI link boot mode as mentioned previously. Larger servers typically contain some sort of independent system service processor, or may have requirements that dictate the use of directly connected firmware. The component design teams take these requirements into account when creating the hardware for the processor and other devices.

Generally speaking the boot mode for a processor is either fixed by the hardware design, or has a platform-specific method of selecting which boot mode will be used. Intel processors designed for the configuration shown in Figure 1 for example may be designed to always (and only) use the Intel QPI link boot mode. Each processor “knows” to look for the boot firmware code by sending transactions down the Intel QPI link connected to the LIOH.

In a similar manner, processors designed to go into the configuration shown in Figure 2 typically would allow multiple boot modes. The boot mode actually used may be determined through the use of hardware strapping options on the processors. As the processor comes out of reset, the state of that strap is sensed and then the boot flow proceeds accordingly. In the Intel QPI link boot mode, each processor proceeds as in the two socket case, by sending transactions over the Intel QPI fabric to the LIOH. In this configuration however, there may be intermediate hops through an adjacent processor socket in order to access the LIOH. This limits the ability of the processors that are not connected directly to the LIOH. Specifically, those processors (CPU B and CPU D in Figure 2) cannot become the SBSP. In fact they typically wait for the SBSP to configure the various Intel QPI resources before they can begin execution; they will stay in a wait loop until the resource configuration has been completed.

Physical Layer Actions

In the Intel QPI link boot mode, the processor hardware typically triggers the initialization of the physical and link layers without waiting for any sort of boot firmware action. Links will train up to the L0 operational state if they are functioning correctly. Link layer initialization including the parameter exchange functions will also take place. In addition, certain address decoding functions are initialized so that the initial code fetches can be directed to the proper destination. The physical layer starts off in the slow mode transfer rate. The link can remain in this mode for as long as necessary although the expectation is that a transition to full-speed operation will be performed as soon as possible.

Link Layer Initialization

Link layer initialization takes place after the physical layer on the component has reached the L0 state. The link layer state machine proceeds through a set of handshake actions that eventually result in the link being ready for normal operation. A key function during this process is the exchange of certain parameters between the two agents. Each agent generates a set of parameters, sometimes called link exchange parameters (LEPs) and the receiving agent captures that information into certain control and status registers. In some cases these parameters directly control the operation of the link layer hardware. Standard or Rolling CRC mode selection is an example of this. In other cases the parameters provide useful information to the boot firmware so that the firmware can make the correct decisions when configuring the hardware resources for the platform.

The LEP values contain several items useful for system configuration as well as others that inform the system as to the link layer's capabilities. One of the primary uses of the LEP information is the *topology discovery* process, which is

described later in this article. The captured LEP values include:

- The sender's port number
- Multiple agent type indicators: home, caching, I/O proxy, or firmware agent are common types
- Portions of the sender's Node ID
- Status bits indicating support levels for low-power modes L0 and L1
- Status bits indicating which CRC modes are supported and preferred by the sender
- The size of the sender's Link Level Retry buffer

In addition to the captured LEP values, status registers exist for both the physical and link layers. The contents of these registers indicate the end result of the initialization process. Normal operation is the expected status, abnormal results are also captured in the status registers and can be used to record problems.

Node IDs

Determining the proper Node ID information to be used for Intel QPI transactions is an essential part of the configuration process. The correct settings for these values are determined by a combination of component design, hardware actions based on platform configuration, configuration strapping on the device, and possibly the firmware or service processor setting configuration register bits. Each component has particular methods that are suited to the needs of the platform that component was designed to service.

Small-Scale Platform Node ID Convention
While a small platform is not required to follow the convention described here, this method is how current Intel processors and IOH devices used in the configuration shown in Figure 1 perform their Node ID assignment.

The process relies on the LEP values as described above, in particular the sender's port number. Currently, each processor and IOH in this type of platform requires one Node ID. By default the LIOH Node ID is set to 0. During the LEP exchange, each processor captures the LEP values from the IOH and from the other processor. This allows the processor to determine which one of its links is connected to the IOH, and which one goes to the second processor. A Socket ID (and APIC ID) for each processor is then assigned based on the port number of the IOH that connects to that processor. Detecting an LEP value indicating port 0 implies that the Socket ID for that processor will become 0. Likewise, the processor connected to IOH port 1 will become Socket ID 1. The Node ID for each processor is then computed as:

Socket_ID + Node_ID_IOH +1.

Referring back again to Figure 1, assume that the IOH port on the left side is Port 0 and that the IOH Node ID is also 0. The Node ID assignment algorithm will result in the left CPU getting Node ID = 1 (0+0+1) and the right CPU getting Node ID = 2 (1+0+1). In this case, hardware takes care of setting this information; no boot firmware or user intervention is required. Note that this algorithm works regardless of which physical port on the processors or the IOH is actually used. The board layout is not constrained in terms of connecting specific ports together in this platform.

Medium-Scale Platform Node ID Convention
Again, while a medium-scale platform is not required to follow the convention described here, this method of assigning Node IDs is used by certain current Intel processors and IOH devices that have been designed for the configuration shown in Figure 2.

In this case, additional functions within the processor require more Node IDs per socket to be assigned. The lower order two bits of the Node IDs are determined by the hardware design and cannot be changed. Higher order bits are set by strapping options on the printed circuit board. When coming out of reset the processor captures these values into internal hardware. The processor then advertises the entire Node ID value as part of the LEP process. No computation algorithm is required but this method does require strapping pins on the processor to be set correctly. In this platform, the LIOH is given a Node ID of 0 and the second IOH would get a Node ID of 4. A total of eleven Node IDs are used to identify all functions in this platform configuration. Note that this means the processors each consume multiple Node IDs in order to identify their internal caching, home, and configuration agents.

PBSP Selection and CSR Access

After the Reset signal to a processor has been de-asserted, each processor socket will determine which of the cores will become the processor boot strap processor (PBSP) for that socket. This takes place before any boot firmware runs because the connection to the boot firmware storage device has not been established yet. Generally speaking, this task is performed by specialized microcode contained within the processor itself.

Control and Status Registers (CSRs) are the “knobs” that the boot firmware can set or examine in order to complete the task of bringing up the system. These CSRs are mapped into PCI configuration space in a manner that is implementation-dependent, but the end result is that the PBSP will be able to perform read and write accesses to these CSRs both within the local socket and within remote sockets such as another processor or an IOH.

Firmware Discovery

The final step before boot firmware execution can begin is to finish establishing the path to the boot code storage device. In the topology shown in Figure 1, each processor socket sends the code fetch commands down the Intel QPI link connected to the IOH. Since there is only one IOH in this case, it is the LIOH. In Figure 2 the issue is a little more complex: the commands must be sent only to the LIOH. The problem is how to establish this path in the Intel QPI Link boot mode where no software is able to execute and help out this process.

The solution lies again in the LEP values sent out by each device, and in specialized hardware and microcode functions in the processor sockets. An *Agent Type* field is communicated as part of the LEP information. One possible Agent Type is a *Firmware Agent.* Only the LIOH will identify itself as the Firmware Agent during the LEP process. Which IOH is actually the LIOH is determined by hardware strapping on the printed circuit board.

The processor hardware and microcode therefore have the information needed to determine which physical link actually goes directly to the LIOH. The processor uses this information to configure itself to be able to send the boot firmware code fetches down that Intel QPI link connected to the LIOH. The LIOH in turn performs the actions necessary to actually interact with the boot code storage device and return the instructions back to the processor(s). At this point, the LIOH will only return the opcode back on the same Intel QPI link on which the request was received.

Summary

Once all of these processes are complete, control is handed over to the boot firmware at the reset vector. Everything necessary for basic Intel QPI code fetches has been established. Links have trained and are in the L0 operational state. LEP parameters have been captured and are used for initial setup and they also contain information useful for later configuration. Node IDs, routing paths to the LIOH, local and remote CSR access, and rudimentary address decoding initialization have all taken place. Note that each PBSP will begin executing the same code at this point. In Figure 1 topology, this means two cores will be executing. In Figure 2 topology there are also only two cores executing since only the sockets directly connected to the LIOH can become PBSPs. What happens next is a function of how the boot firmware is written.

About the Authors

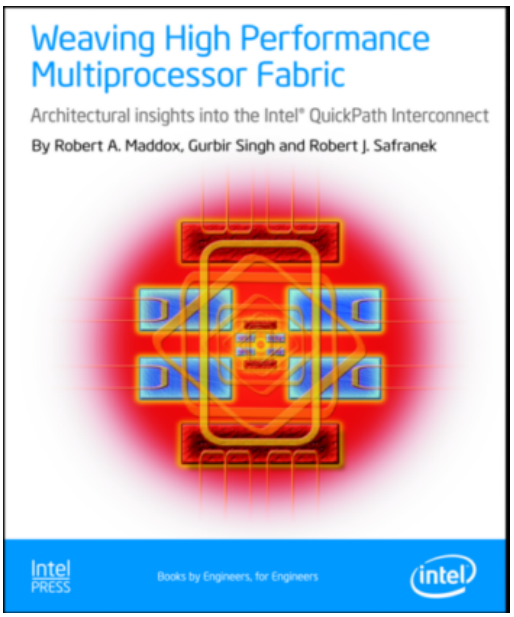
Robert A. Maddox joined Intel® Corporation in 1998. He is currently a Staff Technical Marketing Engineer in the Server Platforms Group, focused on working with the industry on enabling the use of the Intel® QuickPath Interconnect.

Gurbir Singh joined Intel® Corporation in 1984 and is now a Senior Principal Engineer in the Digital Enterprise Architecture and Planning group in Intel. Gurbir led the architecture team defining the Intel® QuickPath Interface. He holds thirty patents in the field of system interfaces and cache architecture.

Robert J. Safranek joined Intel® Corporation in 2000. He has been working on the definition of the Intel® QuickPath Interconnect since its inception and was the primary architect for the first IA-32 products based on it.

This article is based on material found in the book *Weaving High Performance Multiprocessor Fabric* by Robert A. Maddox, Gurbir Singh and Robert J. Safranek. Visit the Intel Press web site to learn more about this book:

http://www.intel.com/intelpress/sum_qpi.htm



E-mail: intelpress@intel.com 20090611