



A SKY Computers White Paper

Building HPEC Systems for Performance and Fault Resiliency with InfiniBand™

**SKY Computers, Inc.
8 Centennial Drive
Peabody, MA 01960
978 977 3000
www.skycomputers.com**

High Performance Embedded Computer (HPEC) systems are required to quickly process large amounts of data in real-time and are required to move that data throughout the system, from the sensor(s), to processing nodes, to output devices (including other HPEC systems) so that the data can be processed in real-time.

Over the years, a series of different interconnect strategies were developed to provide this required performance. Such interconnects include simple bus based interconnect such as the VME and PCI busses, and more complex switched interconnects such as SKYchannel and Raceway. These switched interconnects have proven to be well suited for the HPEC domain because they provide high point-to-point throughput and multiple pathways between endpoints that deliver an even higher aggregate throughput.

New interconnects that are being considered for deployment in the next generation of HPEC systems include but are not limited to the InfiniBand™ and RapidIO interconnects. Initially as HPEC vendors considered new interconnect choices their selection criteria was based primarily on performance considerations, namely the aggregate throughput. However, as anyone who has ever developed a system based on switched interconnects knows, performance is not the only aspect of interconnects that is important. The ability to handle faults is also very important. This includes handling faults during the development cycle (faster time to market considerations) as well as during deployment.

Dealing with interconnect and software faults requires both hardware support as well as software support. In general, software support can be added after the interconnect has been developed. However, unless the proper hardware support is available in the interconnect hardware, the degree of effectiveness of the software solution will be limited. Thus it is important that the interconnect itself provide the proper foundation upon which to build a fault resilient system.

The InfiniBand Architecture: Performance and Fault Support

InfiniBand is one of the leading contenders as the next generation standard interconnect for HPEC systems. InfiniBand is interesting in that it was not initially designed for HPEC systems but rather it was designed to be deployed in data centers. There were two aspects of this domain that were important to the design of the InfiniBand Architecture (IBA), namely high throughput and fault tolerance. The design of IBA reflects these two concerns. It is because the IBA does address these two aspects that it makes an excellent choice for an interconnect to be used in HPEC systems.

It is interesting to note that faults that can occur are not unique to HPEC systems but can occur in any type of system. This observation is important for it means that we can employ solutions that have been developed to deal with faults in other domains to deal with interconnect faults in the HPEC domain. In the remainder of this article, we will use the capabilities of the IBA to prevent faults from occurring, or at a minimum, isolate the damage that can occur due to these faults.

Defining Faults

When talking about faults, we are talking primarily about three different types of faults. The first is a fault in the integrity of the data that is being transferred via the interconnect. For example, if we transfer a data value from one CPU to another, we expect that it will arrive at its destination intact. There have been numerous mechanisms developed to effectively deal with this type of fault including using error correction codes to recover the original value of the data that has been transferred.

The second type of fault occurs due to errors in the software, either the operating system software, the middleware, or the customer's software. These “software” faults are most likely to occur during development and are extremely hard to identify and correct. The reason for this difficulty is that the results that are produced by these types of faults generally do not have any direct connection to the cause of the fault. For example, a process running on a CPU may crash due to its’ address space being corrupted. This corruption may be caused by some process running on some other CPU that inadvertently sent data to the wrong destination, i.e., the memory used by the process that crashed. The relative independence between the fault and the result make tracking these types of faults very difficult.

The third type of fault is a failure in the transmission of the data such that it never arrives at its destination. Shown in Figure 1 below these “interconnect” faults can occur in one of three places:

1. At the point the data is first placed onto the interconnect, e.g., in the interface chip that connects the system bus with the interconnect
2. At the point the data comes off the interconnect
3. Someplace within the interconnect, either at a switch or a link.

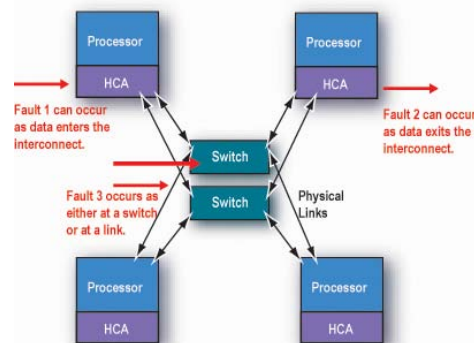


Figure 1: In the InfiniBand fabric there are only three possible interconnect fault points.

These types of faults don't normally show up during development but are more likely to show up during deployment when the system is in use for long amounts of time.

Preventing and Isolating Software Faults

One of the primary features of the IBA is its use of Work Queues. These queues basically hold command packets that are used to specify what data transfers are required. Work queues are connected such that data is transferred between queues. The exact way in which data is transferred is dependent on the type of transfer.

There are two primary types of transfers that can be initiated using these queues, both of which have aspects that help prevent software faults from occurring. There are sub classifications of these transfers, such as whether an acknowledgment is required, that even though are very important to the basic operation of the IBA, will not be discussed here since they are not directly pertinent to the fault tolerance.

The first type of transfer is known as the basic send operation. In this type of operation, not only does the sender have to specify where the data to be sent can be found, but also, the receiver has to specify where the data is to be written. Requiring the receiver to specify where the data is to be written prevents a sender process from accidentally writing data to an invalid location in the receiver's address space.

Of course, this type of protection doesn't prevent the receiver from specifying an invalid address to which the data is written. However, tracking down the root cause of this type of fault is far easier than tracking down an errant remote process. The reason is that we would know that the receiver died because of something it did and because of some action taken by some other process on some other node (which we would have to determine in order to fix the problem).

The second type of data transfer is known as the remote DMA operation. In this type of operation, the receiver specifies a priori where the data is to be written. Specifying where the data is to be written involves the generation of a memory key. The receiver is responsible for generating the memory key and sending it to the sender. The sender includes this key in the remote DMA command packet. As with the send operation, this type of handshake prevents the sender from writing to an invalid memory location.

The previous two mechanisms, supported by the IBA, deal primarily with making sure the sender does not have direct access into the receivers memory space, i.e., prevents the sender from accidentally overwriting a portion of the receiver's memory. However, the receiver can inadvertently specify an invalid address. In order to prevent this type of error from occurring, the receiver must register the memory that the IBA can use to write the data into prior to accepting

any data. This registration of memory with the IBA is how the memory keys that are required for the remote DMA operation and the receiving addresses specified in the send operation are generated.

In addition to providing protection for individual data transfers, the IBA provides a more global capability that can help isolate software faults. Simply put, the IBA supports the ability to partition the nodes in the network into separate domains with each node in the network belonging to at least one domain. When the work queues are connected, their domains are checked to make sure they exist in the same domain. Only queues that belong to the same domain can be connected. Properly used, these domains can help isolate faults to a small number of nodes, making identification of the root cause easier.

Automatically Solving Interconnect Faults

The previous section identified some of the capabilities of the IBA that help prevent and isolate faults. This section explains some of the additional capabilities of the IBA that help deal with interconnect faults.

In the IBA, there are three points at which a hardware failure can occur. These are the channel adapter, the links, and switches. Depending on the actual configuration of the specific IBA network, different degrees of fault handling will be possible. For example, if there is only a single switch in the network, and that switch fails, the entire network will fail. Thus, if fault handling is to be required, the actual configuration of the network must be an integral part of the product design.

Assuming that fault handling is required, the network must be designed such that there is at least two distinct paths between any two adapters. Such a configuration will allow the IBA to automatically handle failure within the network.

Part of the process of connecting work queues involves specifying the route, i.e., the links and switches that will use to transfer data. In addition to specifying this primary route, a secondary route can also be specified. If an error should be detected, e.g., a timeout occurs without the message having been sent, the IBA starts a negotiation phase that results in the secondary route being promoted to the primary route.

As shown in Figures 2 and 3, should a switch that is used in the primary route fail during the transmission of some data, that data will be resent using the secondary path. In addition, all subsequent data will be sent using this secondary path.

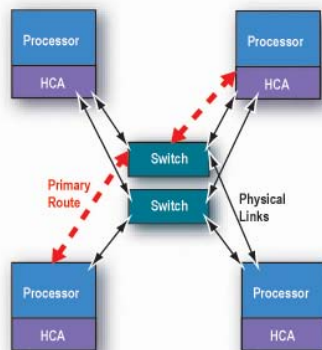


Figure 2: Work Queues specify the primary route that will be used to transfer data.

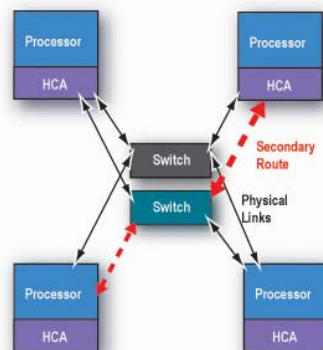


Figure 3: When a switch used in the primary route of InfiniBand fails, the intelligence of the fabric automatically reroutes the data to a secondary path allowing the system to complete the task.

Conclusion

As we've shown using the InfiniBand Architecture there are several mechanisms that can be employed in order to help deal with faults and isolate their root causes. With its performance and fault resiliency InfiniBand can address a wide range of HPEC applications. SKY Computers will use InfiniBand throughout the architecture of its next generation of embedded boards and systems. SKY Computers has collected relevant white papers on the use of InfiniBand for HPEC systems at http://www.skycomputers.com/news/article_archive.cfm. For a full discussion of additional capabilities of the IBA, download the IBA specification at <http://www.infinibandta.org>.