EE Times

August 17, 2011

Design Article

Using PCIe & intelligent DMA to achieve blazing data rates in real-time recording instruments

Chris Tojeira

Chris Tojeira of Pentek describes how the use of PCIe, intelligent DMA and a real time front end as well as an efficient API messaging structure resulted in the design of a real-time data recorder capable of sustained data to disk rates of 2 GB/sec and higher.

In today's world of high-speed A/D converters, operating in the Gigahertz range and beyond, real-time signal recording has become a challenging feat that requires specialized hardware and intelligent application software. To build a real-time recorder capable of streaming sustained data to disk at rates of 2 GB/sec and higher, the system developer must be able to overcome the limitations presented by the recorder's I/O interfaces, memory, operating and file system as well as consider the optimal use of disk drive technology, RAID controller technology and user-friendly control tools for data capture and analysis.

Meeting real-time requirements

For a recording system to be considered "real-time" it is required that the recorder captures every sample of data that the front-end provides with absolutely no loss. This must happen consistently to provide confidence that the system will run in the most mission critical situations. It is also important to consider the ongoing requirement to move data faster.

With this in mind, the commercial PC server is an excellent platform choice as microprocessor technology is constantly advancing in both processing speed and architecture: memory interfaces are capable of streaming data at rates of 10 GB/sec and higher; serial fabric PCI Express Gen 2 x16 is capable of maximum data rates of 8 GB/sec; and SATA II provides disk storage rates of 3 GB/sec to a single drive.

The commercial PC motherboard's roadmap includes transitions to PCI Express Gen 3, capable of doubling the current data rates. SATA III is beginning to hit the commercial PC market, providing storage rates of 6 GB/sec to a single drive.

The latest Intel processors are approaching 4 GHz rates clock rates with hex and octal cores, while memory continues to get smaller, cheaper and faster. This provides engineers with a solid foundation to build upon; one that can grow as new high-speed front-ends are introduced.

Keeping Up with Real-Time Streams

Once the foundation of the recorder is established, it is essential to provide a front-end I/O device that is capable of streaming data in real-time. This front-end device typically consists of one or more high-speed A/D converters or digital interfaces that acquire data at a constant rate, and a set of DMA engines that move data off of the device.

In a real-time system, these DMA engines are the most critical feature of the I/O device, as the design that is implemented on the internal hardware will dictate how well the instrument can stream data and maintain its real-time performance requirement.

The latest high-performance I/O devices should provide Intelligent DMA chaining that give the system developer the ability to optimize data flow and maximize performance.

The Intelligent DMA engine permits the user to chain many large buffers of data in a link list. Buffering data in such a way will allow the system to absorb any momentary latency hits, which can be caused by a number of external factors, assuring the instrument meets its real-time requirements.

While the DMA engines are responsible for moving data off of the device, it is the PCI Express (PCIe) engine, inherent within the device, that supplies the path to the server PC's system memory.

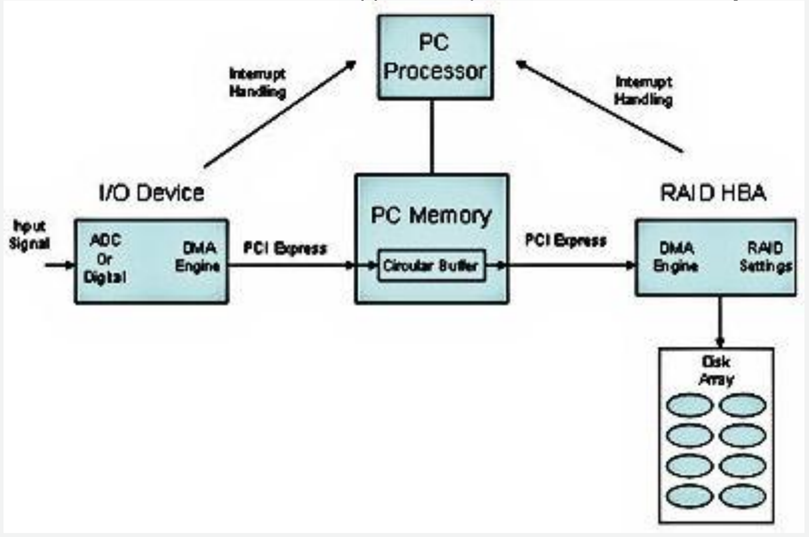


Figure 1 – Real-Time Recording Instrument System Level Block Diagram

It is essential that the I/O device supply a sufficiently fast enough PCIe interface, to provide engineers with the bandwidth to maintain the data rates required by the front-end. Buffered data must be sent to disk at data rates that can match those performed by the front end I/O device. **Figure 1 above** shows the block diagram for one typical real-time recording Instrument.

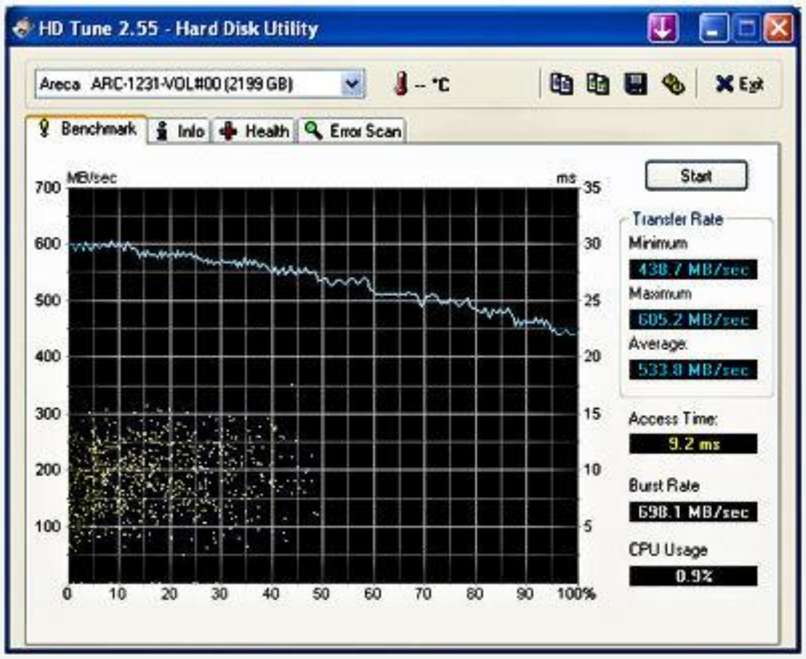
RAID considerations

The challenge in selecting the best RAID controller lies in finding one that reliably meets the sustained streaming write and read requirements of the system. Developers can capitalize on the off-the-shelf RAID host bus adaptors' (HBA) inherent features including selectable RAID level control and automated disk recovery facilities.

Additionally, these high-performance RAIDs provide user-selectable parameters, such as cache size, caching method, stripe size, command queuing and other settings that allow the user to maximize the performance of the recorder.

In multi-channel recorders, it is often beneficial to create multiple RAID drives, consisting of different disks for each channel, but controlled by the same RAID controller. By assigning independent logical drives to the recorder's channels, the RAID is able to write to contiguous disk space as much as possible.

It is also important to recognize the non-linearity of hard disk drives (HDD) when developing a recorder. Since the density of an HDD remains constant through the disk and the rotation speed remains constant, the read and write rates of a disk fall as HDD accesses move from the outer edge of the disk to the inner edge. This can be seen in **Figure 2 below**, since disks present their logical addressing from the outer edge to the inner edge, disk read and write rates fall as a disk fills up.



Transfer Rate
Minimum: 438.7 MB/sec
Maximum: 605.2 MB/sec
Average: 533.8 MB/sec
Access Time: 9.2 ms
Burst Rate: 698.1 MB/sec
CPU Usage: 0.9%

Figure 2 - HD Tune Plot showing the disk data rate, as it changes throughout the physical drive.

RAIDs, built on several disks provide similar non-linearity in their data rates. Because of this, it is imperative that the system developer either provide enough drives in the RAID to meet the maximum data rate requirement for the entire volume, or limit the size of the drive volume to the percentage of disk space that will meet the system's data rate requirements.

In the case of a RAID with the performance shown in Figure 2 above, if the system developer was required to build a recorder that could maintain 500 MB/sec, the drive volume should be formatted to use about fifty percent of the total disk space. This is done simply by formatting the drive to the appropriate size within the operating system.

Operating system tasks

When dealing with non-real-time operating systems like Windows and Linux, it is important to minimize the operating system's impact on the recording application. The amount of processor intervention in the recording software can be minimized by dedicating the data transfers to the DMA controllers and simply leaving the processor to manage the data flow. The developer should also elevate the priorities of real-time tasks within the application and keep background tasks at lower priorities to avoid impacting the recorder's performance.

GUI control is a must

When designing a real-time recorder, it is important to provide the user a control interface that is intuitive and easy to use. The easiest way to achieve this is through a graphical user interface (GUI.)

A GUI provides the user with the ability to control the recorder by pushing virtual buttons with a mouse or via a touch screen, providing a simple and satisfying out-of-the-box experience. The GUI should allow the user to, not only control the recorder, but also provide: ongoing status updates to the user; facilities to manage the data files; and utilities to monitor and analyze the signals being recorded.

The GUI should have the ability to run either locally on the recorder or remotely on an independent PC or other device. Being able to run on a remote device allows the user to put the recorder in an environment that may not be appropriate for the operator. This allows the recorder to be close to the sensor or antenna source, placing the A/D or digital I/O interface near the signal of interest.

The best way to provide both local and remote control of the recorder is through a client-server architecture. This architecture provides a socket-based communication link between the client GUI and the server recording application, separating the real-time portion of the recorder (the server) from the non-real-time portion (the client.) The client can then connect to the server, whether the client sits on the server PC itself or on a PC that is connected to the server over Ethernet.

By sending messages to the server, the client GUI can control all aspects of the recorder. This includes the ability to start and stop recordings, to setup front-end hardware parameters, to monitor incoming signal information and to request status information from the server. The interface for this messaging structure should be defined in an Application Programming Interface (API).

Integration into a larger system

While the API would be used by the GUI to communicate with the server, the engineer can also utilize the API to allow the recorder to be integrated into a larger system.

By providing a user API, the recorder becomes more than a stand-alone instrument, serving as a user development platform as well. This allows different types of users the flexibility they may desire, while providing the out-of-the-box experience of an instrument all in one product.

In order to define an API that can be easily integrated into a larger application, it is important to define the API routines in a simple and straightforward manner. These routines should abstract the user from details of the message building, the socket interface and other intricacies of the recording architecture. Error checking should be included in the recording server, allowing the user to receive error codes in response to failed messages.

The API should not only contain routines that control the interface, but should contain routines that obtain general status information, perform built-in-test (BIT) facilities and allow the user to view snapshots of the data prior to and during recording. Combining these facilities provides the user with the ability to create a well-featured recorder application as part of a bigger system.

Post processing and analysis

One of the problems engineers often have to deal with after recording data in the field is the issue of offloading the mission data to a system that will perform the analysis, post-processing and archiving. This process should be made simple and quick, minimizing the down-time that field engineers have during the offload process.

The use of a non-proprietary file system (NTFS) to record data provides the ability to avoid the offload process, required by systems that use a proprietary file system. In this situation, the user can instantly access recordings as standard files on the recording device itself without the need for a lengthy conversion process.

In addition, by utilizing hot-swappable SATA drives, field engineers can simply swap out disks filled with mission data for fresh ones and transport only the data disks to their analysis system.

By adding a RAID controller to the analysis system data can be instantly accessible on the analysis machine, avoiding the offload process completely. The recorder is then immediately available to collect new data as soon as the disk swap is made, allowing for almost no down-time in the field.

Keep in mind the data files that are provided to the analysis system must contain critical information related to the recording event itself. This can be accomplished by adding a small header to the beginning of each data file.

The parameters stored in this header should be well-defined, containing all of the critical information about the recording, including time-stamping, I/O module settings and general information about the recording session itself. Additionally, user-settable fields should be reserved, allowing the user to add any information that he wishes to the file header.

The information stored in the file header can be used by system analysis and signal visualization tools. Integrating these tools into the recorder provides a robust product; one that allows field engineers the ability to verify their signal integrity prior to, during and after a recording session.

Example real-time control application

Combining all of the techniques described will help the system developer create a highly dependable real-time recorder, capable of recording and playing data at very high sustained data rates.

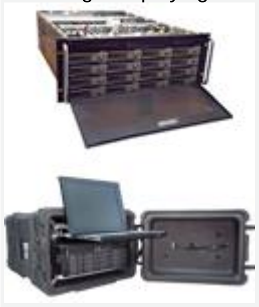


Figure 3 – Real-Time Recording and Playback Instrument (COTS and Rugged Versions)

An example control application is silicon lithography inspection, which uses a Model 2706 Real-Time recording instrument (**Figure 3 above**), along with its SystemFlow Recording Software, to inspect silicon wafers. The RTS 2706 recording instrument records silicon wafer data via a sensor and the data can be used for quality control and inspection purposes.

Chris Tojeira *is the Technical Director and Chief Architect of the Recording Systems Product Line at Pentek, Inc. Chris has over 15 years experience in the electronics industry and has held positions as a Strategic Account Engineer and Applications Engineering Manager for Pentek. He holds a BS degree in Electrical Engineering from Rutgers University in NJ.*

http://www.eetimes.com/design/embedded/4218871/Using-PCIe---intelligent-DMA-to-achieve-blazing-data-rates-in-real-time-recording-instruments