NEW CONTROL AND DATA ACQUISITION SYSTEM IN THE ADVANCED LIGO PROJECT*

R.Bork, M. Aronsson, D. Barker, J.Batch, J. Heefner, A. Ivanov, R. McCarthy, V. Sandberg, K. Thorne, LIGO Project, California Institute of Technology, Pasadena, CA, USA.

Abstract

A new control and data acquisition system architecture is being implemented as part of the Advanced Laser Interferometer Gravitational-wave Observatory (aLIGO) project. This system employs a number of multi-core processor-based computers to perform real-time control, with connection to PCI Express Input/Output devices via fiber optic links. Requirements on the real-time control algorithms include servo loop rates of up to 65KHz and synchronous, deterministic operation to within a few microseconds. To attain this real-time performance, a patch has been developed to the Linux operating system that allows detachment of a processor core from the Linux scheduler for the exclusive use of an assigned real-time task. An overview of the real-time software design, which takes advantage of this "core locking", and the particulars of the Linux patch are described in this paper.

OVERVIEW

The two Laser Interferometer Gravitational-wave Observatory (LIGO observatories are presently undergoing an upgrade process to increase the overall sensitivity of the instruments. As part of this Advanced LIGO (aLIGO) upgrade, the Control and Data acquisition System (CDS) is being updated and expanded. On the hardware side:

- VME I/O crates are being replaced with PCI express (PCIe) I/O modules and links
- VME processors are being replaced by standard, rack mount server class computers

On the software side,

- Commercial real-time operating systems have been replaced by Gentoo Linux, with a LIGO custom patch to provide hard (plus or minus a few microseconds) real-time performance
- A software package has been developed, Real-time Code Generator (RCG), to allow system scientists and engineers to develop control algorithms via a Graphical User Interface (GUI).

HARDWARE

For aLIGO, PCI Express (PCIe) I/O modules replace VME modules. These are primarily 16 and 18 bit, multi-channel, simultaneous sampling, Analog to Digital Convertor (ADC) and Digital to Analog Convertor (DAC) modules. These modules are housed in a custom I/O expansion Chassis (IOC), that contains:

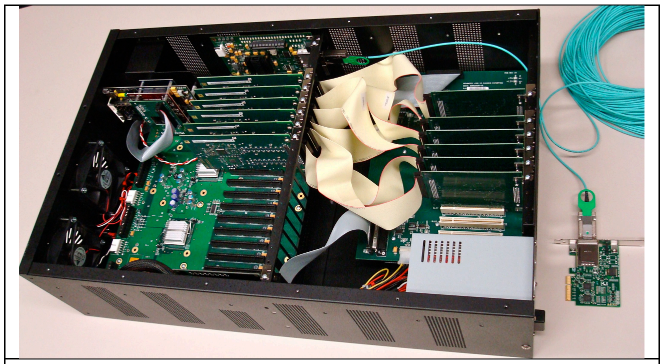
Figure 1: aLIGO IOC and PCIe Fiber Link

- A commercial 17 slot PCIe backplane.
- Commercial fiber optic PCIe uplink to the real-time control computer, that is remotely located up to 300m away.
- LIGO custom designed timing module that provides accurate ADC/DAC triggering at 65536Hz. This unit derives its time from the new timing distribution system, which is locked to the Global Positioning System (GPS) time.
- Custom timing and interface backplane.

The IOCs are mounted in electronics racks, adjacent to the control electronics that provide the control and data acquisition analog signals.

The control computers are located in a separate computer room, interfacing to the IOC via the PCIe fiber optic link. These computers are standard, commercial rack mount units. Each computer contains one or two, 6 core processors and 6GByte, or more, of RAM, depending on the application needs.

To run the LIGO real-time control software, care must be taken in the selection of the computer motherboard. First, the BIOS must be capable of mapping the required number of PCIe modules. Tests have shown that not all computers are created equal, in this sense.

As described later, our software requires that applications not be interrupted by the system. However, some motherboards have hidden System Management Interrupts (SMI), which could not be disabled. This introduced as much as 150µsec “glitches” in our code execution times. In the selection process, the Supermicro® X8DTU motherboards best fit our application requirements.

* LIGO was constructed by the California Institute of Technology and Massachusetts Institute of Technology with funding from the National Science Foundation and operates under cooperative agreement PHY-0107417.

SOFTWARE

Overview

The real-time software is designed as a complete package, from providing a Graphical User Interface (GUI) for code development, standard code library to develop source code, 'make' functions to compile the code, to final installation and execution of the software. For real-time code development and execution, the primary components of this package, discussed in further detail in this paper, are:

- The LIGO Real-time Code Generator (RCG).
- A custom patch to the General Public License (GPL) Linux operating system kernel to obtain real-time performance.

The system also provides additional software tools in support of the real-time software, listed here briefly, but not within the scope of this paper:

- Data AcQuisition (DAQ) System software. This software receives data from all LIGO real-time systems, combines and compresses the information into a standard file format, and archives the data to disk.
- Network Data Server (NDS): Software that provides data from the DAQ system, either real-time feeds or from disk/tape archive, to various data clients.
- Dataviewer: Provides graphical representation of data, from NDS, either in real-time or from archived data.
- Diagnostic Test Tools (DTT): Provides a number of standard diagnostic functions, including power spectrums and swept-sine injection and transfer function analysis capabilities. These diagnostics are presented via a GUI.
- Foton: A GUI tool for designing and loading IIR filter designs to the real-time software.

In addition, the LIGO control system uses the Experimental Physics and Industrial Control System (EPICS) base software and extensions to provide communications between real-time systems and operator interfaces.

Real-Time Code Generator (RCG)

In planning for aLIGO CDS, it was foreseen that continuing to write custom C code for control applications, even with standard library parts, would overburden the relatively small group of software developers. Not only did final systems for aLIGO need to be supported, but there were also many aLIGO subsystem test facilities that would require CDS support very early in the process. This led to the development of the RCG.

The RCG uses MATLAB® Simulink® as the GUI. A custom library was added, that includes CDS custom software modules and supported Simulink® parts. The code developer may then use these parts in an application model and interconnect the parts to describe desired software execution sequence. Some key elements of this library are:

- I/O: Parts are provided to describe and connect all of the I/O modules supported by the RCG.
- Inter-process communications (IPC): real-time communications between applications is provided via shared memory (applications running on the same computer), reflected memory networks (applications running on computers located 4km apart), or via a PCIe switched network fabric (computers located within 300m of each other). Latencies vary from a few nanoseconds, using shared memory, a microsecond, using PCIe, or 15usec via the 4km-long reflected memory runs.
- IIR Filter Module: This is the key module for developing the proper transfer functions for system control. Depending on control complexity, a control model may contain as many a few hundred of these modules. Each of these modules may contain up to 10 separate filters, along with individual, and group, on/off selections and gain and offsets, with ramping, at runtime. For diagnostics, and data acquisition, each module is provided with three output software test point locations and a software test signal injection location. The system also supports the reloading of filter coefficients during runtime as new filters are designed or updated.
- User defined part to support custom C code. This provides a mechanism to allow users to include their own custom C code into the control model, if desired.

Once the model has been built and saved, a standard "Makefile" is provided to produce the executable software. Invoking 'make' starts the build process:

- RCG Perl scripts parse the model file, developing a parts and code sequence list.
- Perl script produces the real-time C code source file.
- Compiler is invoked to produce the real-time executable. This includes the user-defined application, with standard RCG wrapper software. The latter provides for proper code timing and sequencing, standard set of diagnostics, connections to the data acquisition system and I/O drivers.
- An EPICS sequencer and database are built to support communications to/from the RCG real-time process and EPICS channel access. This allows the use of various EPICS extension software to provide operator interfaces via Ethernet.
- Various files are produced to describe all of the code data channels available to the data acquisition system and diagnostic test point information.
- Produces a basic set of EPICS display screens for use as operator interfaces.

Operating System

The software is designed to operate at sample rates from 2048 samples/sec to 65536 samples/sec. The real-time execution must be precise and repeatable to within a few microseconds. Execution must also occur synchronously with the 65536Hz clock provided by the LIGO timing system.

To achieve this performance, the GPL Linux operating system, used on LIGO real-time control machines, has been modified with a LIGO custom kernel patch. This patch was developed to isolate a given CPU core from the Linux system for the exclusive use of the real-time control program.

The Linux kernel comes with a built-in mechanism to isolate a CPU core from the rest of the Linux system, both Linux kernel and user space. This mechanism is called CPU hot-plug. This interface provides a function in the user space to off-line or shutdown CPU cores. This interface does not, however, provide a mechanism to load user specified software onto the core when it is set offline, as it loads a CPU idle function.

The LIGO Linux patch is a modification of this function. Rather than having the core load an idle function as it goes offline, the patch software loads the desired control application process as an independent kernel object. The primary advantage of this method is that it provides total core isolation for the control process. It is now independent of the Linux scheduler and no other tasks and/or interrupt routines will be assigned to this core.

Now that the core and software have been isolated, a code scheduling mechanism must be provided. This is accomplished by a special RCG model known as the I/O Processor (IOP). The IOP, further described in the following section, provides sequencing and timing information to the user application real-time processes for synchronous operation.

If the control process needs to be removed, for example if new code is to be loaded, it is necessary to unload the control process and reinstate the core as a resource to the Linux operating system. To do this, the control program is issued a stop flag from user space. The code now exists and returns. The “cpu idle” routine is called, then the CPU core is brought back on line using the standard Linux kernel mechanism.

It should be noted that application of this patch does not turn the system into a traditional, full-featured real-time operating system. A pre-emptive scheduler is not provided, nor are interrupt capabilities implemented. So, there are some downsides for general use:

- User must provide a scheduling mechanism. For LIGO applications, the IOP provides for this function.
- Each application requires a separate CPU core to run on. This can be wasteful of resources for small applications and/or require more cores.
- Control applications run as a continuous loop, performing all calculations on each trigger. While the application could be designed to only run some calculations on each trigger and/or alternate what code gets processed on a given cycle, there is no mechanism, such as an interrupt, to asynchronously trigger code.

On the positive side, this method has a few advantages:

- No context switching and associated latencies. Once locked into a core and cache, the code timing is very precise.
- No chance of priority inversions.

Real-time Execution

As described above, each real-time application is loaded on to its own CPU core at run time. The first CPU core (0) is always reserved for Linux operating system use and running non-real-time critical applications. Applications in this category include:

- EPICS software to interface the real-time processors with the controls Ethernet links.
- Data Acquisition networking software, which transports data from the real-time processes to the data acquisition system, via dedicated Ethernet links. This data is sent in blocks at 16Hz.
- Arbitrary Waveform Generator: software used to derive test signals for injection into the real-time software.
- Test Point Manager: Conveys diagnostic test point selections to the real-time processes. The control process is built with a number of test points included, each of which may be selected on demand to provide data output to the data acquisition system, and, in turn, to operator station diagnostic software.

CPU core 1 is reserved for the IOP. The IOP is designed to run in a continuous loop at the highest supported rate of the system, i.e. 65536Hz.

The IOP provides a number of key functions for the system:

- Code synchronization. The IOP is triggered to run by detection of a new set of data from the IOC ADC modules. The ADC modules, in turn, are precisely clocked by the LIGO timing system at 65536Hz.
- Initializes and maps all I/O, including PCIe I/O modules.
- Signals the timing system to start sending clocks to the I/O modules coincident with the next GPS 1PPS time mark.
- Acts as the interface between ADC/DAC modules and user applications, such that data is clocked in/out on precise 65536Hz time marks.

Remaining CPU cores may now be loaded with user control applications. These applications run in a continuous loops at rates from 2048Hz to 65536Hz. The basic loop process is:

- Wait for a trigger from the IOP, received via shared memory.
- Get timestamp, in the form of GPS seconds, from the IRIG-B receiver module. This module receives its time from the LIGO timing system.
- Input and process ADC and IPC data.
- Run the user defined control application.
- Send outgoing DAC settings to the IOP.

- Send IPC information, via real-time networks or shared memory.
- Output data to be sent to the data acquisition system, via shared memory, to the data acquisition network software.
- Communicate data to/from the EPICS interface, via shared memory.
- Perform various housekeeping and self-diagnostic tests.
- Return to Wait State.

STATUS

The real-time software, and associated hardware, have been installed for use at a number of locations over the past several years in support of test facilities. Installations have ranged from "stand-alone" systems (real-time control and DAQ software on a single computer, with one or more IOC) for small LIGO test stands, to distributed systems for larger LIGO test facilities. Presently, control systems are being installed for aLIGO at the Hanford, WA and Livingston, LA sites. These systems consist of as many as 30 real-time control computers for distributed control.

The latest released version of software, designated V2.3, was distributed as the initial code install to both LIGO sites in June of this year. The next release is planned for November 2011, which incorporates a number of enhancements requested by the application developer community of scientists and engineers.

ACKNOWLEDGEMENTS

LIGO was constructed by the California Institute of Technology and Massachusetts Institute of Technology with funding from the National Science Foundation and operates under cooperative agreement PHY-0107417. This paper has LIGO Document Number LIGO-P1100052-v1.