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 Linu

# 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*

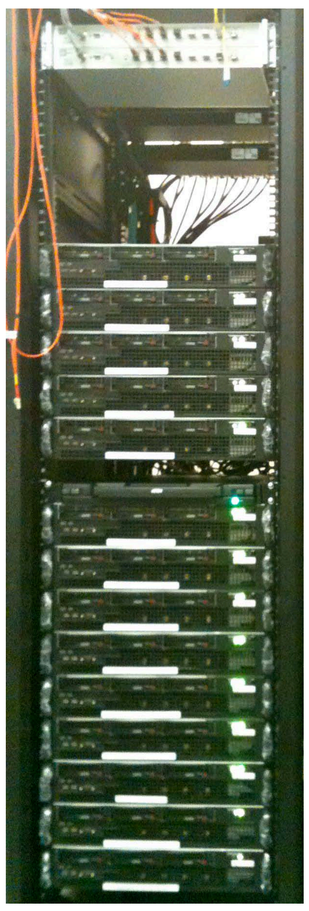
A new control and data acquisition system architecture is being implemented as part of the Advanced Laser Interferometer Gravitational-wave Observatory (aLIGO) project. This system employs a number of multi-core processor-based computers to perform real-time control, with connection to PCI Express Input/Output 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.

## *PCIe I/O replaces VME*

- LIGO custom chassis
- 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 that is locked to Global Positioning System (GPS) time.
- Custom timing and interface backplane.
- Commercial PCIe ADC, DAC and Digital I/O Cards

## *Rack mount servers replace VME processors*

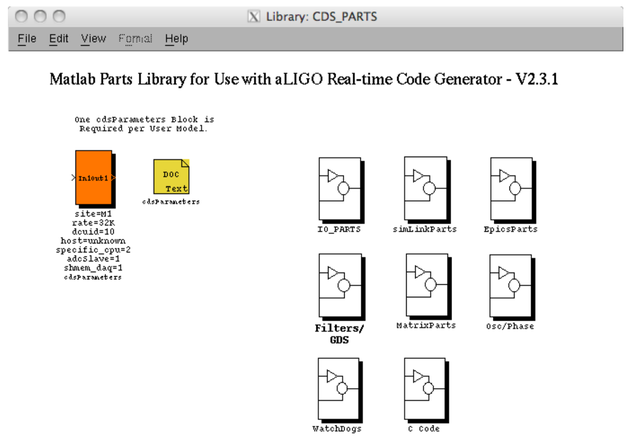
- 2U, rack mount, server class machines with up to 2 CPUs with 6 cores each (12 core ea. w/hyperthreading)
- PCIe interfaces to:
  - IOC, via PCIe fiber link
  - IRIG-B time code receiver for accurate time stamping.
  - PCIe real-time network (to 300m)
  - Reflected memory real-time network (long distance ie 4km)
- Two GigE network interfaces
  - Control LAN
  - Data AcQuisition (DAQ) network



* 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.

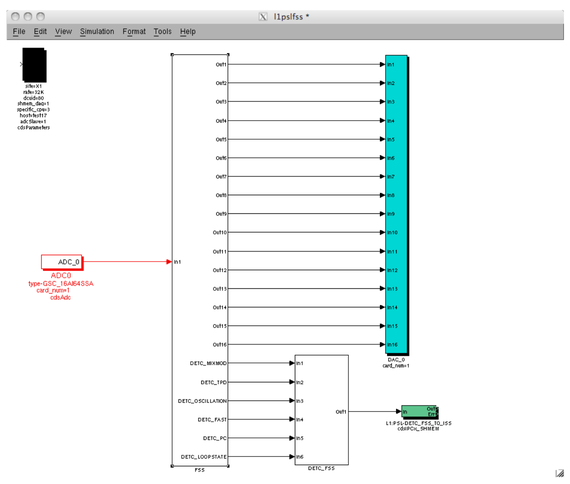
# *Real-time Code Generator*

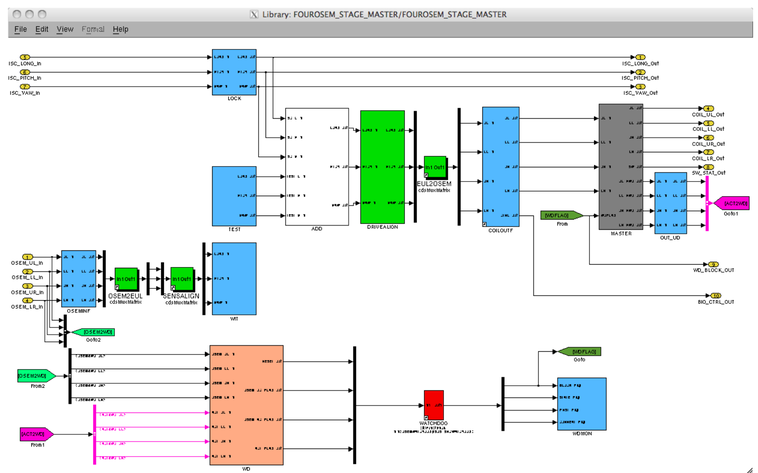
- Allows system scientists and engineers to develop real-time control applications via familiar tool
- Matlab® Simulink® Graphical User Interface (GUI)
- RCG library of supported components
  - I/O Parts Library
  - Digital Filters
    - IIR
    - Polyphase FIR
  - EPICS Record communication
  - Math functions
  - Inter-Process Communications (IPC)

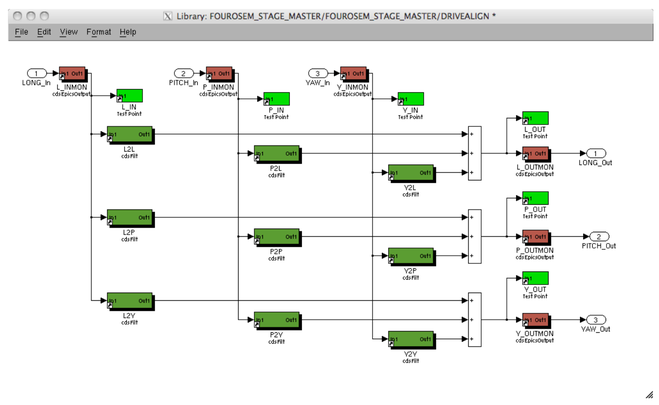



RCG Main Menu

# Example Control Model




# *RCG Compilation*

• RCG Perl scripts parse the model file, developing a parts and code sequence list.

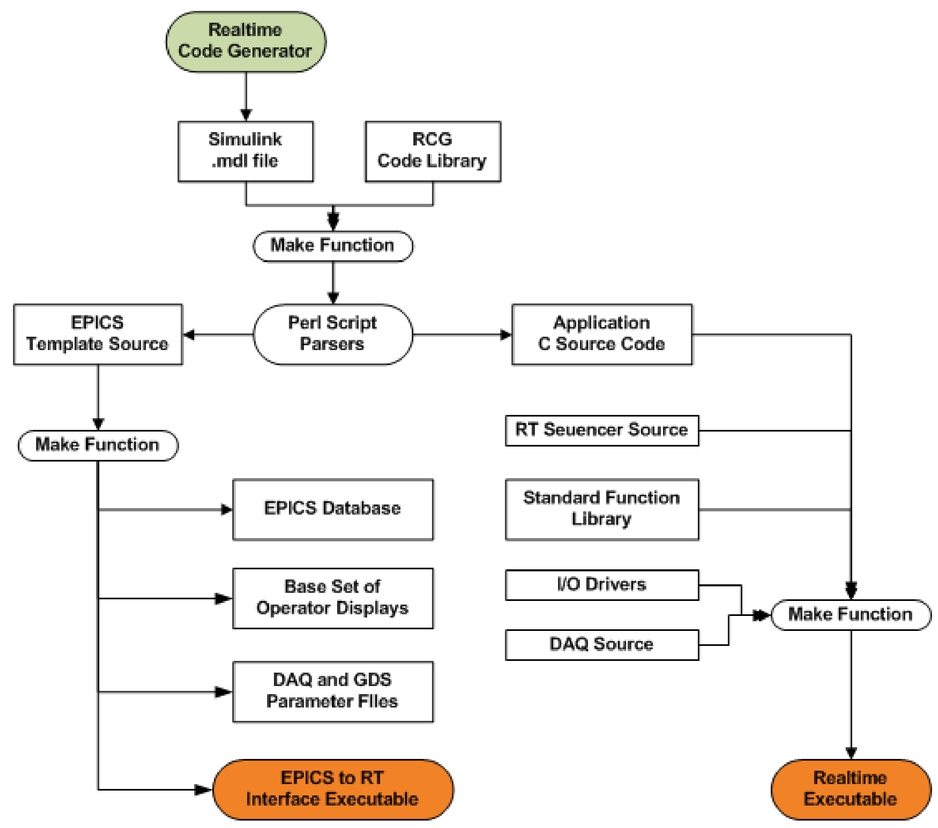
•Perl script produces the real-time C code source file.

•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. The result is a Linux kernel object.

•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.




RCG Compilation Flow

# *Operating System*

▪ Operating system for control computers is GPL Linux, with LIGO custom 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.

▪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).

▪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.

- ▪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.

# *Real-time Execution*

▪ Executes on multi-CPU, multi-core computer systems

▪ CPU core 0 reserved for GPL Linux OS and non-real-time critical applications.

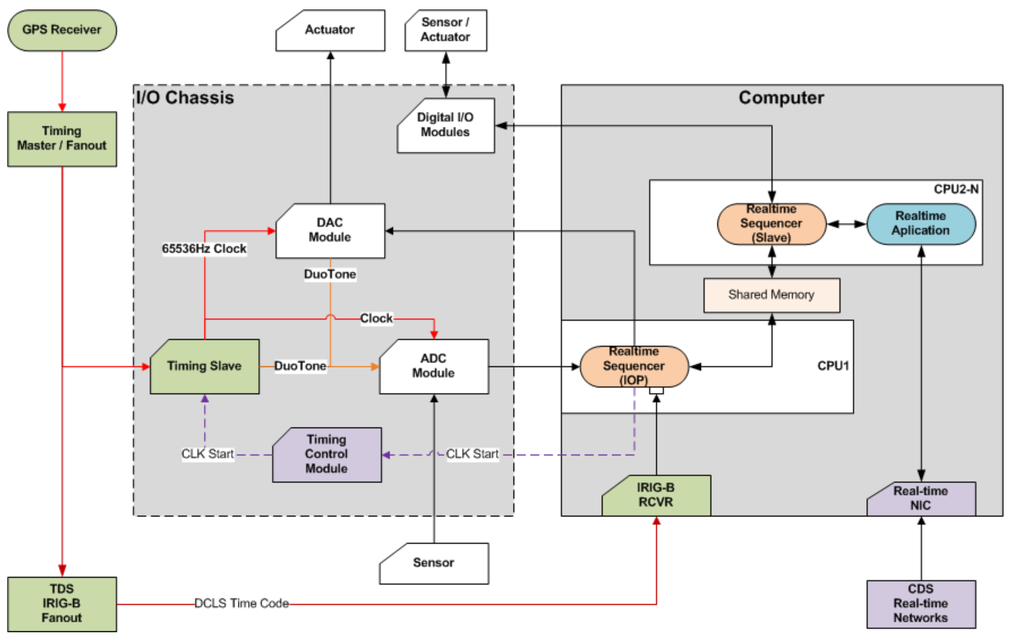
▪ CPU core 1 reserved for I/O Processor (IOP)

- ▪ Initializes and controls ADC/DAC I/O
- ▪ Synchronously triggered by ADC trigger, which is triggered by LIGO timing system at 65536 Hz.
- ▪ Synchronously triggers remaining user control applications.

▪ CPU core 2 to n

- ▪ One control application per core
- ▪ Applications operate at 2K to 64K samples/sec
- ▪ Applications may communicate in real-time via:
  - ▪ Shared Memory
  - ▪ PCIe network (1µsec latency)
  - ▪ Long range reflected memory (15 µsec latency over 4km range)

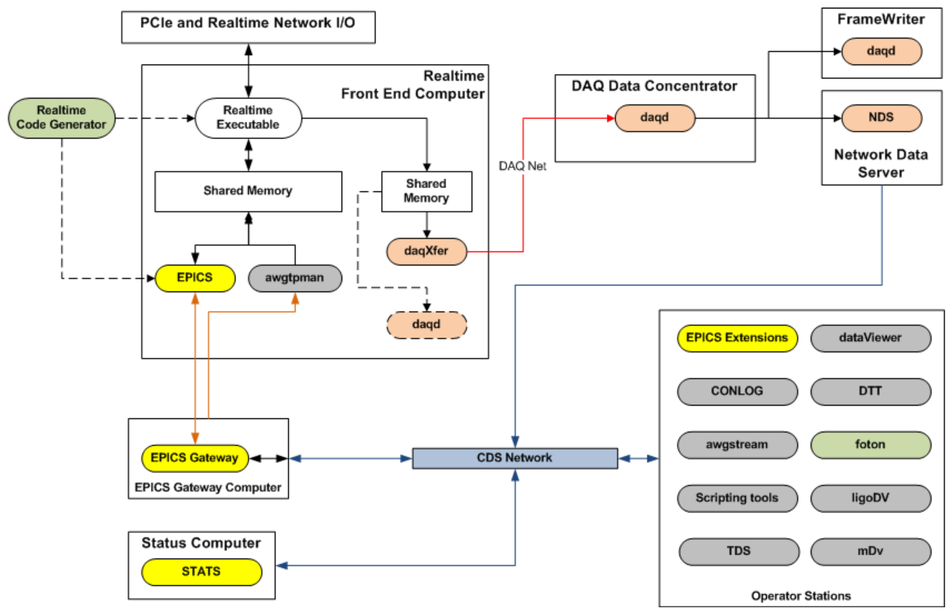
▪Built in Data Acquisition up to 4MByte/sec per application via dedicated network.

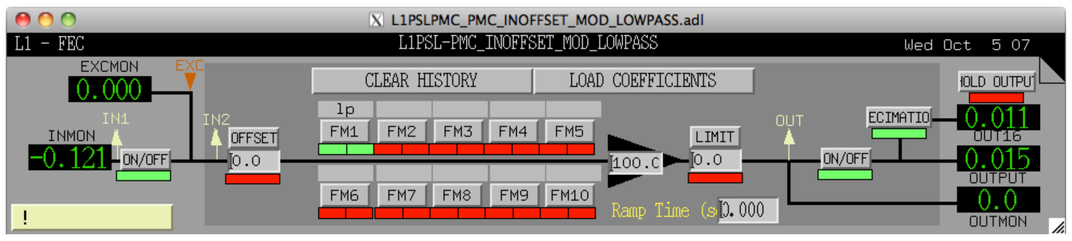



▪Real-time Software Execution Model

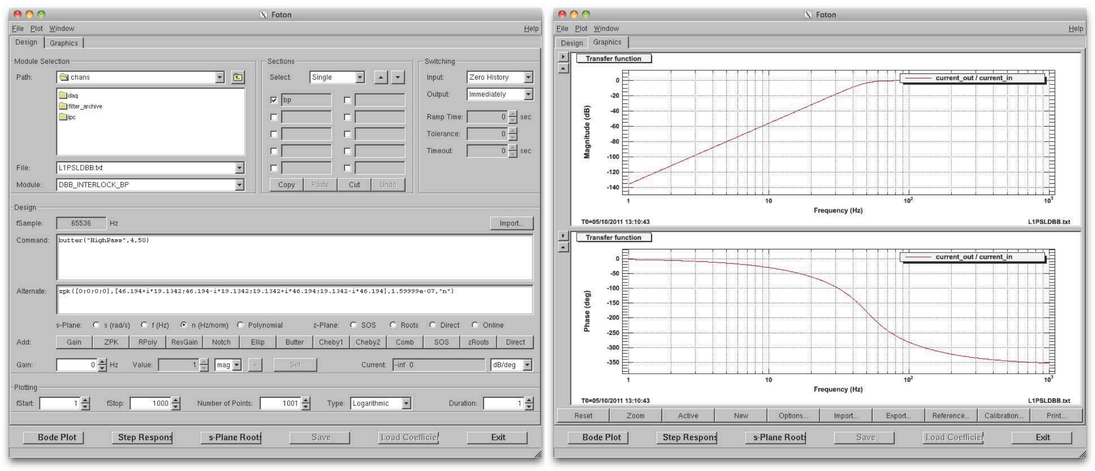
# *Additional Tools*

- A number of software tools were developed and/or adapted to support this system, some of which are listed below.
- Each real-time application is provided with a number of interfaces for monitoring, data acquisition and diagnostics
  - EPICS, via shared memory, allows use of EPICS Channel Access to communicate via EPICS standard tools.
  - Arbitrary Waveform Generator / Test Point Manager (awgtpman) for diagnostic software signal injection and transmission.
  - Dedicated DAQ network connection.
- DAQ System
  - Archive data to disk
    - Individual channel rates from 1 to 64K samples/sec.
    - Aggregate rates tested to 48MByte/sec continuous with over 100,000 channels.
  - Serves data via a Network Data Server (NDS)
    - Real-time data feed or from archive
    - Client software developed for various software packages.

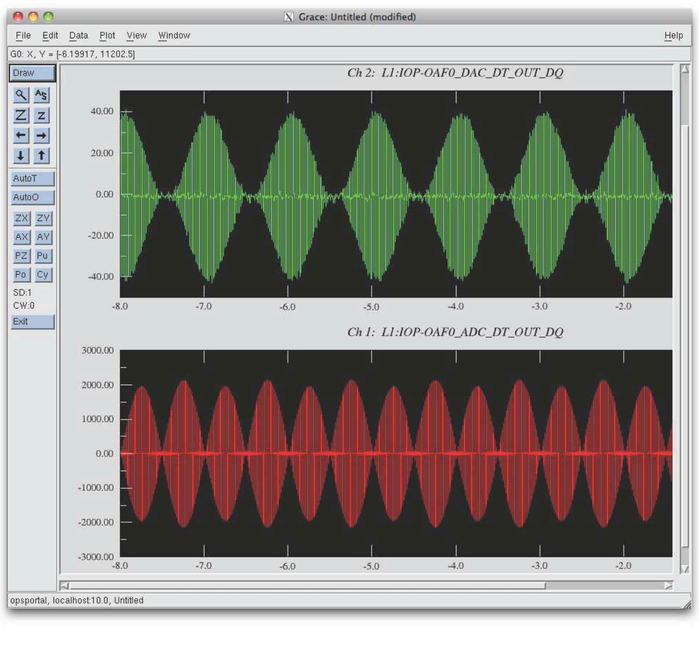


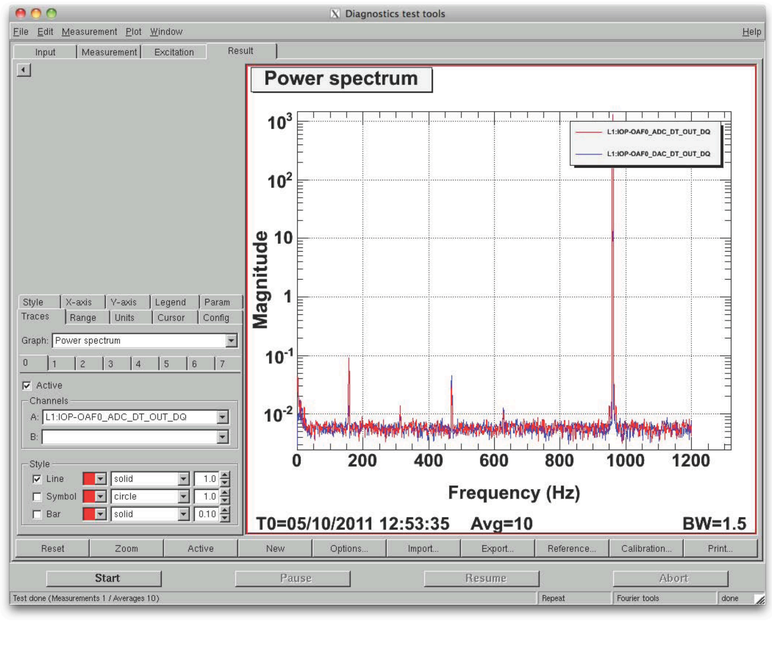

- IIR Filter EPICS MEDM Display




- Foton IIR Filter Design Tool




- Dataviewer
  - NDS client for plotting data, real-time or from archive using xmgrace.




- Diagnostic Test Tool (DTT)
  - On line data analysis package with GUI interface
  - Design and inject test signals via awgtpman
  - Receive data via NDS.