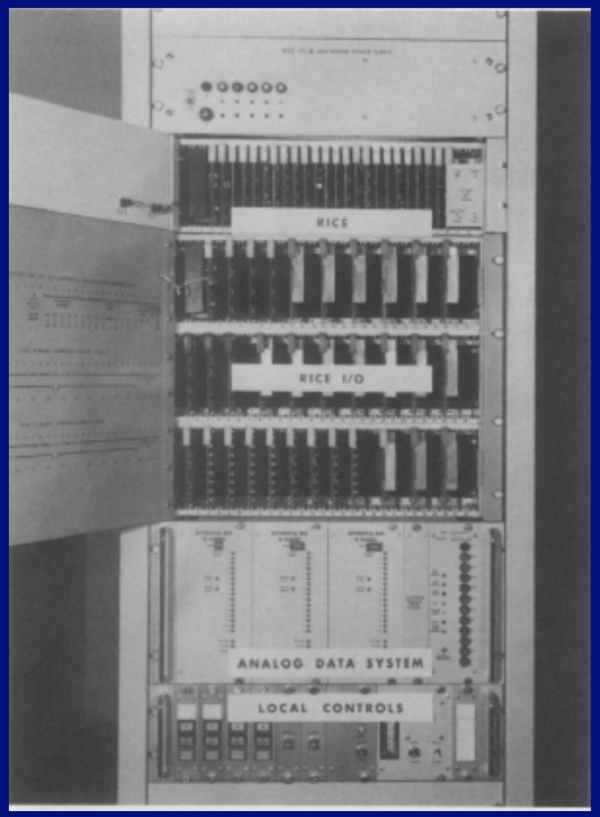
# LANSCE migration to an all-EPICS control system as enabled by Data Access EPICS, Industrial I/O, and IvPortDriver

S. A. Baily

April 27th 2023

LA-UR-23-24184

# LANSCE reached 50 years of operation last year




- Unique capability to deliver beams to 5 different areas, on a pulse-by-pulse basis
  - Linac pulsed at 120 Hz.
  - H+ and H- simultaneously to different areas.
  - Beam currents and energies can be varied depending on the delivery area.
- Original control system RICE.
  - Remote Instrumentation and Control Equipment.
  - Designed in the late 1960s (control computer upgraded twice since then).
  - Triggered ADCs that would schedule trigger time in cycle and pulse based on which data are requested.
  - Took synchronized data from 66 locations
- EPICS IOCs introduced in the early 1990s

# Hardware form factors to replace RICE


- NI compactRIO (~240 IOCs)
  - 9024 controller (Power PC) running VxWorks with Virtex 5 (12 years ago) (~150)
  - 9038 or 9048 controller (Intel Atom) running LinuxRT with Kintex 7 (latest) (~90)
  - Takes up to 8 C Series Modules.

- VPX/cPCI (~150 IOCs)
  - VPX Crate with one or more Bittware cards and a PCI Express to PCI bridge to a cPCI crate
  - Bittware card includes an FMC connector and a Stratix IV FPGA with a NIOS-II softcore processor.
  - Cyclone III FPGA provides an independent reset.
  - 125 MHz 14-bit 16-channel FMC ADC or  201.25 MHz 4-Channel FMC ADC
  - cPCI crate with analog conditioning cards and a Micro Research Finland Event Receiver
    - Controlled by the VPX card in the PCI-Express root port.

# Software Applications


- **Industrial I/O** (cRIO) – slow controls and monitoring
  - EPICS device support talks to the FPGA via NI's C Application Programming Interface (no LabVIEW RT application).
  - Generic I/O with traditional EPICS device support.
  - LabVIEW FPGA code supports many module types.

- Interceptive Beam Diagnostics (cRIO)
  - LabVIEW RT code starts an EPICS IOC built as a shared library
  - **IvPortDriver** (asynPortDriver-based) device support to interface with LabVIEW RT code.
  - LabVIEW RT and FPGA code are customized for the type of measurement (wirescanner, emittance, etc.).

- Non-interceptive Diagnostics (VPX/cPCI hybrid)
  - Waveform digitizers
  - IOC running **Data Access** EPICS (3.15 with Jeff Hill's special modifications)
  - Monitor beam current, spill, RF signals, beam position, and beam phase.

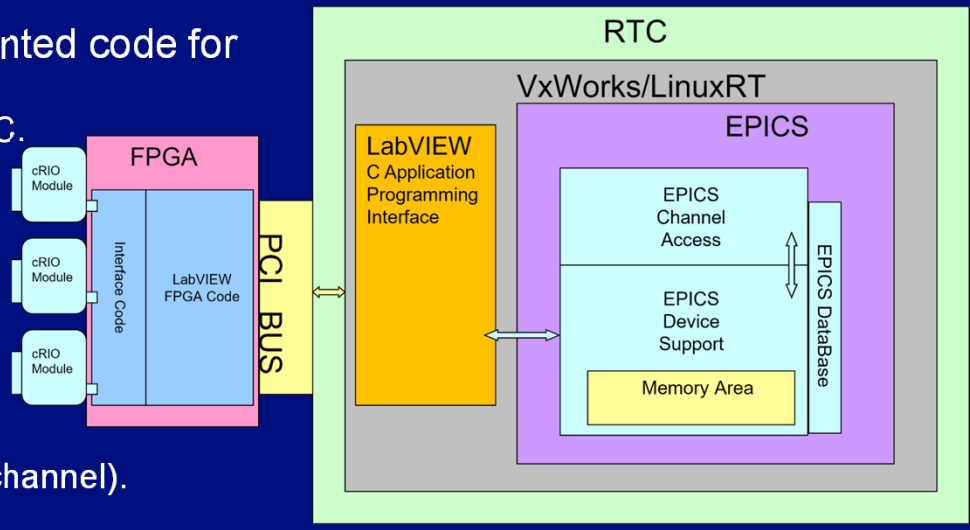
# Slow Controls and Monitoring - Industrial I/O

- Eric Björklund talked about this at previous collaboration meetings.

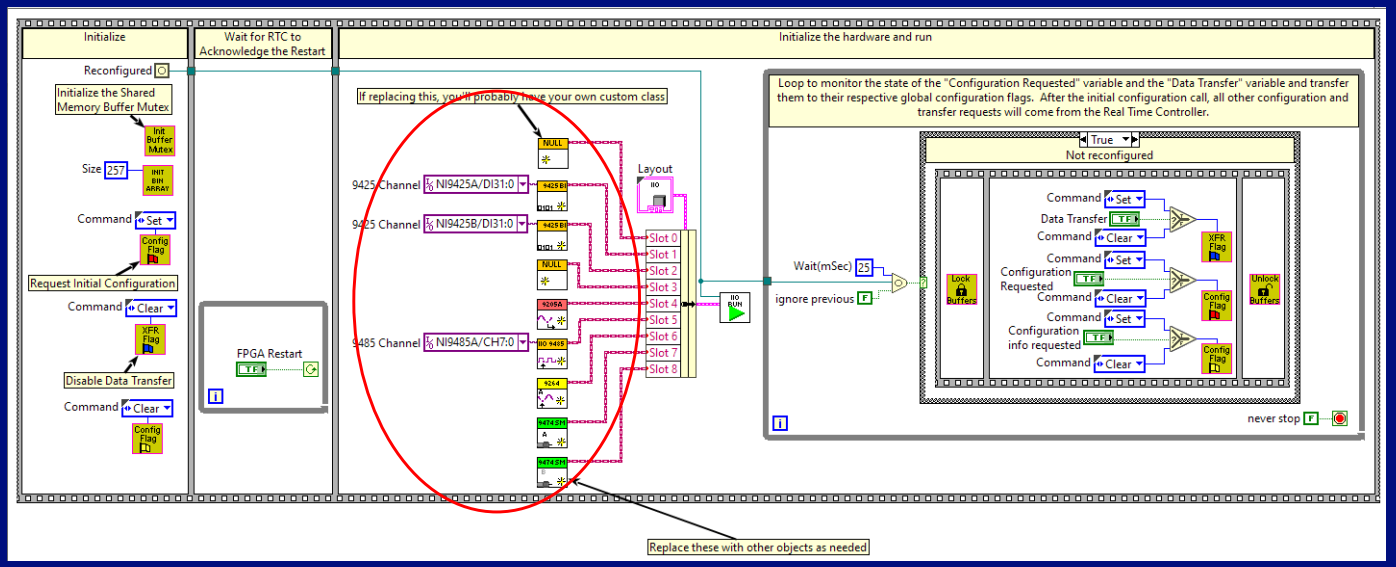
- EPICS device support talks to the FPGA via NI's C Application Programming Interface (no LabVIEW RT application).
  - Generic I/O with traditional EPICS device support and uses VME_IO addressing (card and signal)
  - Customization is in the EPICS database.
  - Inputs have some options for behavior that can be configured in the FPGA design at runtime.

- I recently upgraded the LabVIEW FPGA code to object-oriented code for better maintainability.
  - Once the C-API was released I rewrote the LabVIEW RT code in C.

- Currently have support for a variety of modules
  - Binary I/O: 9375, 9425, 9477, 9485
  - Counters: 9401, 9403
  - Analog Input: 9205, 9208, 9211, 9213, 9217, 9237
  - Stepper Motor: 9474
  - Analog Output: 9264, 9265
  - Array Analog output: 9262 - 1 MHz arbitrary waveform output (6 channel).



# IIO Object-Oriented LabVIEW FPGA Code



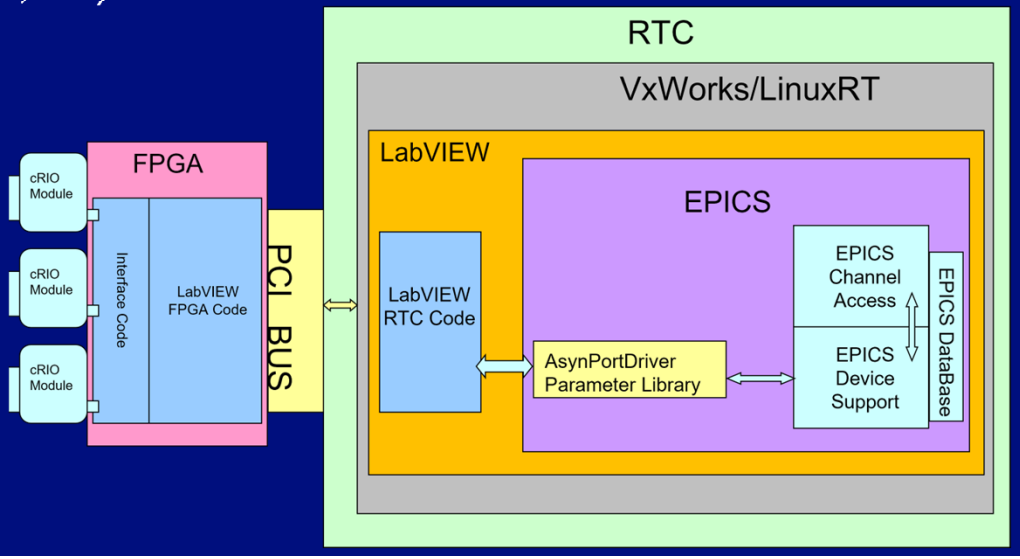

- Replace the circled objects and constants with the appropriate modules for the IOCs, then compile to a bitfile.
- Otherwise writing LabVIEW code or EPICS device support is only needed when adding support for new module types.
- The EPICS IOC starts as usual and loads the appropriate FPGA bitfile (based the contents of the st.cmd file).
- Behaves just like any other EPICS IOC.

# Interceptive Diagnostics - lvPortDriver

- I talked about this at the September 2016 collaboration meeting.
  - Available from https://github.com/lanl/lvPortDriver

- lvPortDriver is an AsynPortDriver class for talking to LabVIEW programs.
  - Instead of writing your own constructor, you create the parameter library at runtime in your LabVIEW code.
  - LabVIEW program starts the EPICS IOC (LabVIEW RT can't be built as a shared library).
  - Then it behaves like a normal IOC, except the IOC interacts with the LabVIEW software rather than directly with the hardware. Essentially this means you write your hardware support driver in LabVIEW instead of C++.
  - dbior reports all parameters in the parameter libraries.

- For developers that prefer to write their own FPGA and RT code to control the hardware in LabVIEW (e.g., wirescanners, emittance measurements, etc).

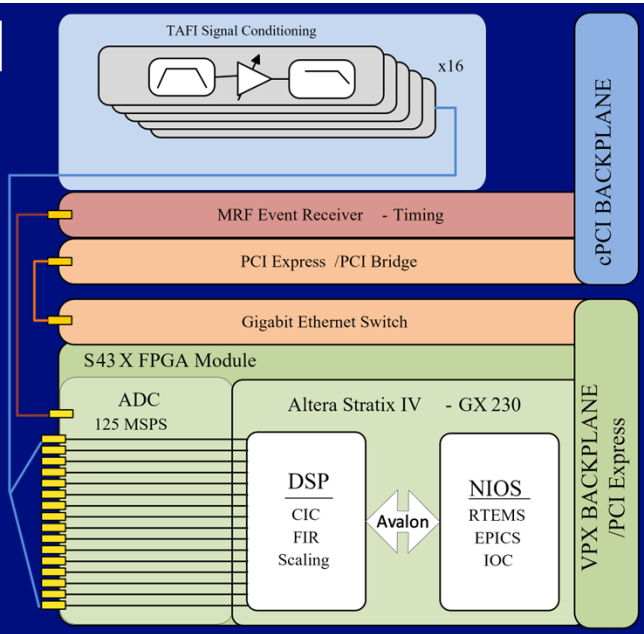


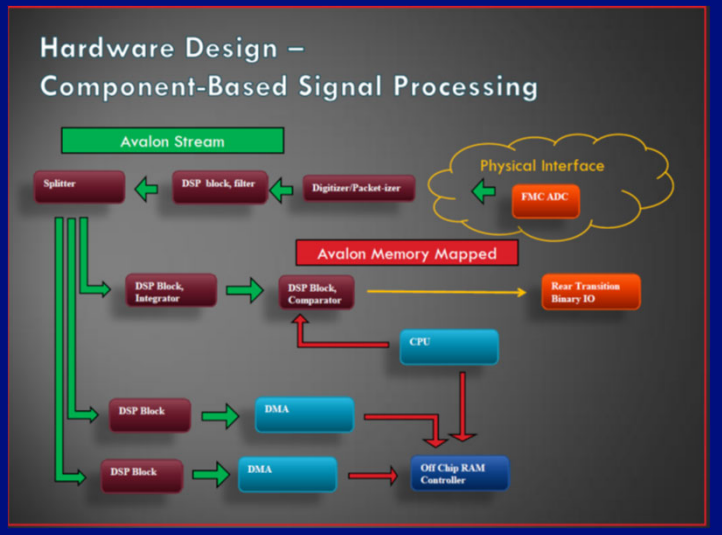
# lvPortDriver– For Interceptive Diagnostics

- The RICE system used to rely on the client application to coordinate all hardware, including the number of pulses sent to a stepper motor between datapoints.
- This new method allows the IOC to handle all the hardware details, the user application just configures parameters starts the scan and analyzes the data returned.
- Many developers are more comfortable working in LabVIEW than with state notation language, subroutine records, and sequence records or creating large custom programs as C++ device support.
- Having a real EPICS IOC is a huge advantage over having a LabVIEW program that reads and writes to a soft IOC to communicate with EPICS.
  - If a record doesn't update its address tells you which parameter its information is supposed to be coming from.
  - If a command record writes to an invalid parameter, then the record will be marked as invalid rather than, just no response because nothing monitors the PV.

# Non-interceptive Diagnostics – VPX/cPCI

- Jeff Hill has talked about the hardware architecture and the EPICS modifications at prior collaboration meetings.
- The most difficult RICE capability to replicate was the custom data collection. Using waveform digitizers on independent IOCs, we can **provide more data and improve performance**.
- The cPCI board allows the use of Event Receivers with more outputs and enables development of inexpensive analog signal conditioning cards.
- Our new beam position and phase monitors use the same design, but with a different FMC card.
- As with industrial I/O, this data is handled generically on a per channel basis.




Los Alamos NATIONAL LABORATORY

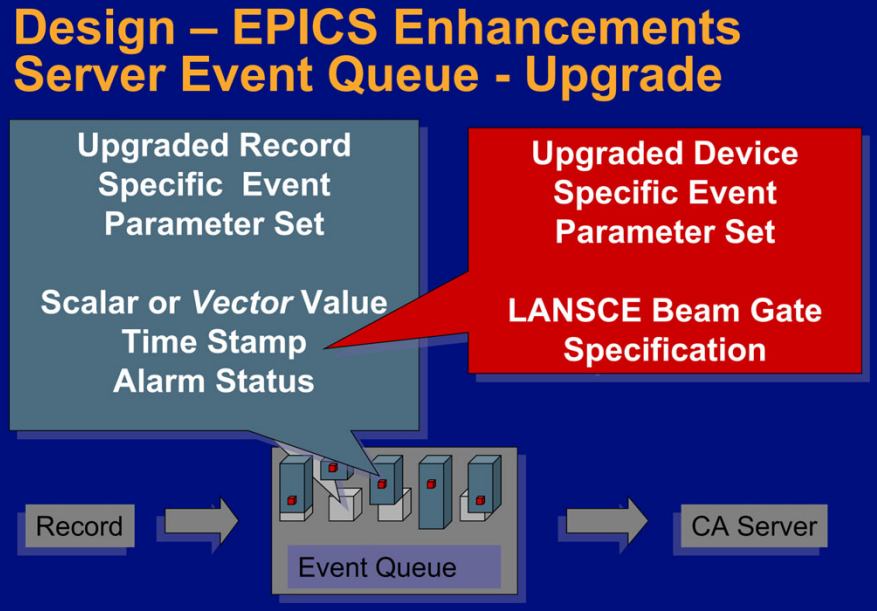
# Non-interceptive Diagnostics – Data Access

- Jeff's channel access server improvements allow filtering based on what data is of interest to the client.
  - Lua code filters can be applied.
  - We have implemented some LANSCE-specific filtering.
    - Flavor, i.e., beam delivery location(s) can be specified.
    - A time slice of the waveform base on timing system gate parameters can be specified.

- A Java library allows data from many IOCs to be correlated using a per beam pulse timestamp (distributed to each IOC via the Event Receiver).

- EPICS Server enhancements allow smart pointers to objects to be placed on the event queue, so timing information can be stored with the data, and no copies are made until the data is written to the network.

- A recent enhancement in the device support allows waveforms to be scaled in the FPGA using EPICS breaktables.
  - Table information is written to registers in the FPGA.



# Upgrade was successful

- New systems provide better data and are easier to maintain and upgrade
- Unfortunately, new systems are not expected to last for 50 years (neither was the RICE).
- FPGA enables rapid development and deployment of some things that were traditionally done with discrete hardware components, but the availability and support lifetime is significantly shorter.
- IIO and lvPortDriver are both good options for integrating EPICS with cRIO systems.
- The Data Access features Jeff Hill implemented allowed us to provide a modern system that incorporates the required functionality of the old system and improves upon it.

# Acknowledgements


- J. O. Hill and E. Björklund were instrumental in developing the software tools for RICE replacement.
- The entire AOT-IC (Accelerator Operations and Technology – Instrumentation and Controls) group led by M. Pieck and H. A. Watkins contributed to this effort.

# Hardware  Lifecycle: Vendor Supported vs Customizable

- NI cRIO:
  - Pro
    - Upgrade path – old code can usually just be rebuilt for new hardware
  - Con
    - Tied to the vendor
- S43X:
  - Pro
    - Very flexible hardware options
  - Con
    - Upgrade path may require significant development time if particularly if no succession hardware is available.

# Software

- NI cRIO:
  - Pro
    - Just write functional code
      - Timing constraints already written
      - Board Support Package already written
  - Con
    - Tied to the vendor
- S43X:
  - Pro
    - Completely customizable.  We are running RTEMS on a softcore processor, hardware instructions can be configured in the FPGA design
  - Con
    - Need to write timing constraints (FPGA)
    - Need to write and maintain RTEMS BSP for soft core processor

# Software lifecycle: biggest risk

- For both systems:
  - Pro
    - Upgrade path
  - Con
    - New tools leave old hardware behind
    - Ability to run old tools requires maintaining old development environment
- cRIO:
  - LabVIEW 2020 and newer no longer support VxWorks cRIOs
  - cRIOs with pre-2020 software require Silverlight to maintain.
  - ISE Xilinx compile tools require Windows 7 or RHEL6.
- S43X:
  - Altera Cyclone III not supported after version 13 of the Altera tools. RHEL8 is first listed for version 20 (Intel).
  - If the SGDMA IP (not recommended for new designs) becomes obsolete significant work will be required for both FPGA and BSP.