

# Control System Integration

Tom Shea  
ORNL

# Accelerator Control System

- Convey all monitor, control, model-based, and computed data from all accelerator, facility, safety, and operations subsystems to accomplish supervisory control, automation, and operational analysis.
- Scope extends from the interface of the equipment being controlled through to the designers and operators of the accelerator facility. Sometimes including experimenters
- Includes all hardware and software between these bounds: computer systems, networking, hardware interfaces, and programmable logic controllers
- Not restricted to closed loop control systems (a primary focus in this school). In fact, at some accelerator facilities, these systems are treated as black boxes developed by the technical groups.

# Responsiveness

Configuration  
Archiving  
Analysis...

Control Room  
Physics Apps...

Digital Signal  
Processing  
Subsystem

Office PC  
Network

Supervisory Control  
System (SCADA)

Fly by  
Wire

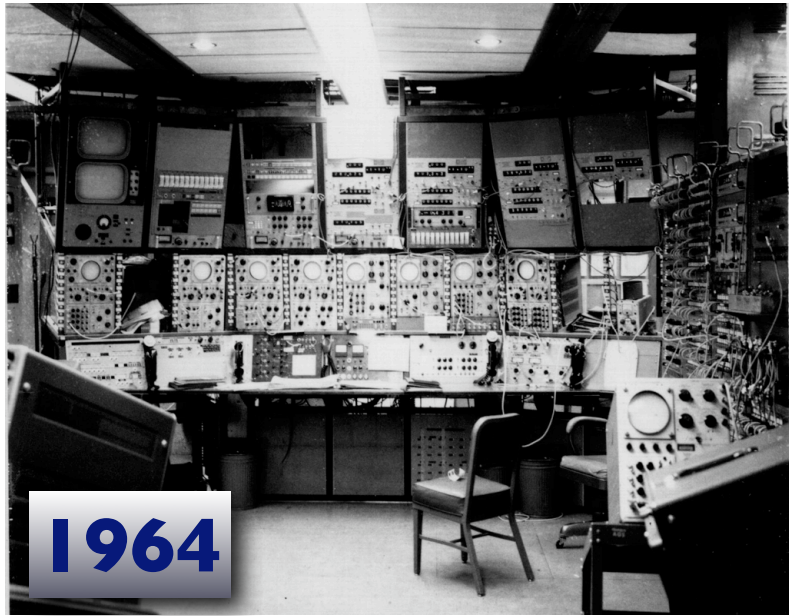
Nodes are responsive  
Network is responsive



Nodes are real-time  
Network is responsive



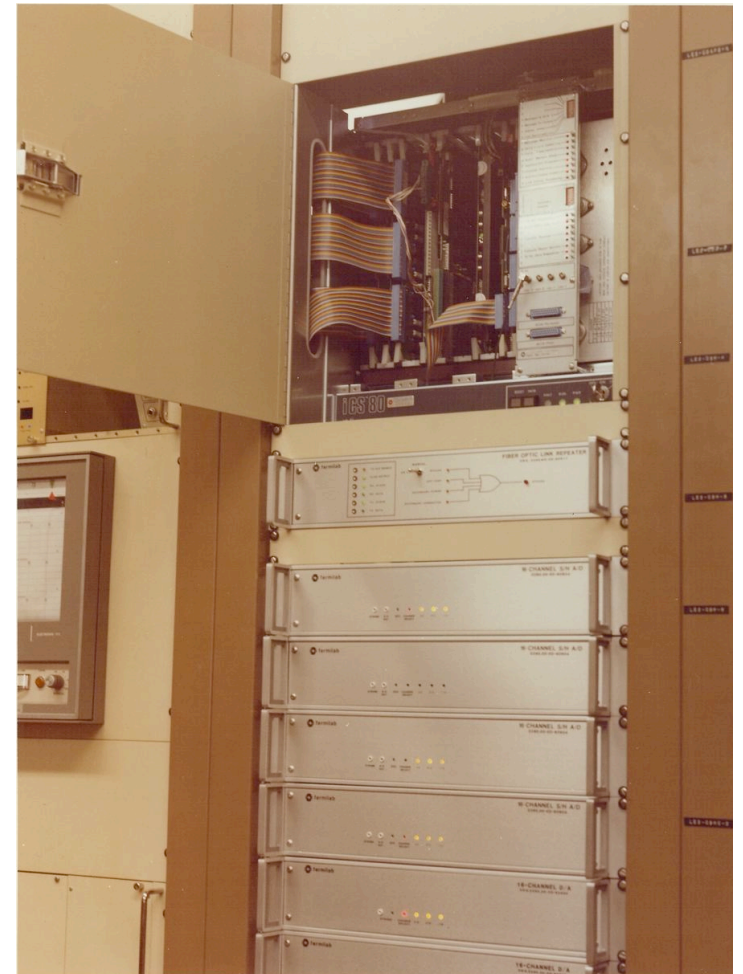
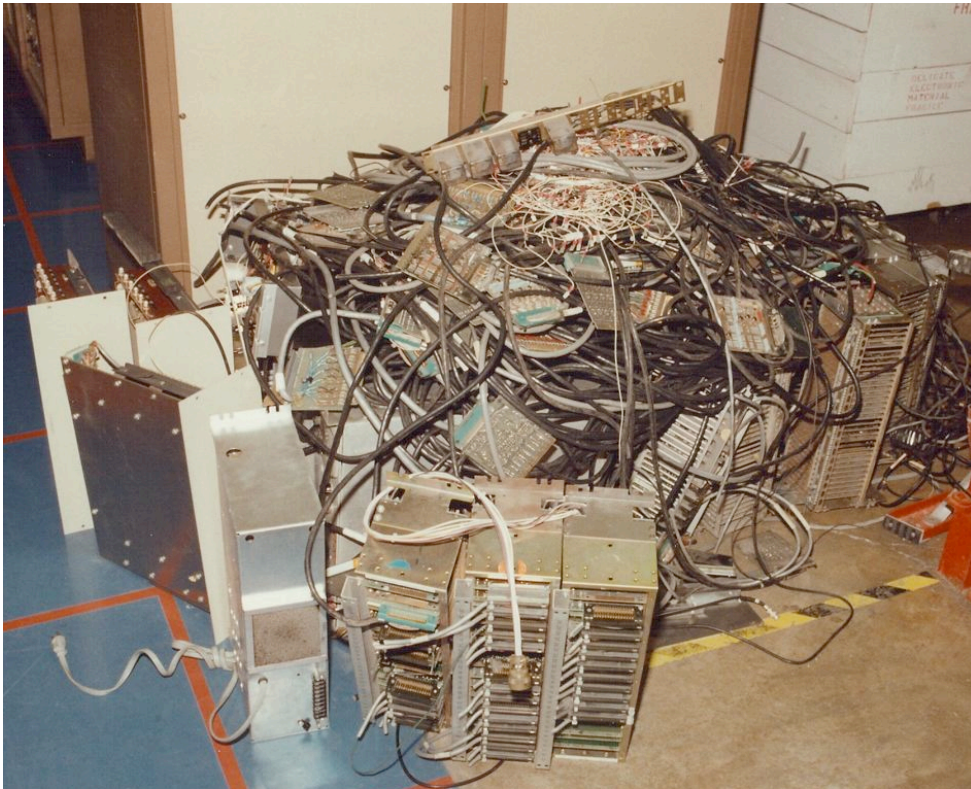
Nodes are real-time  
Network is real-time



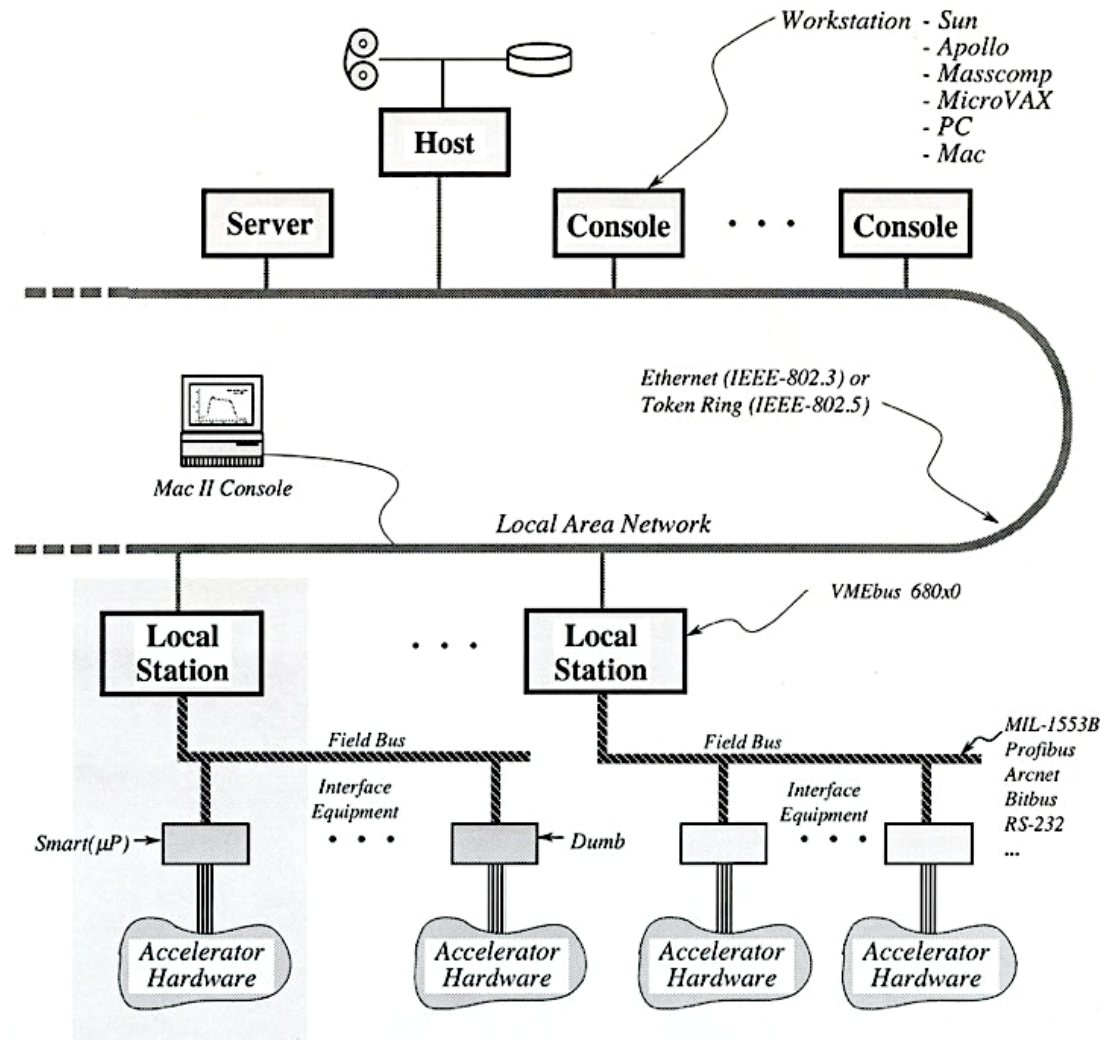
re



# Out With the Old, In With the New, circa 82

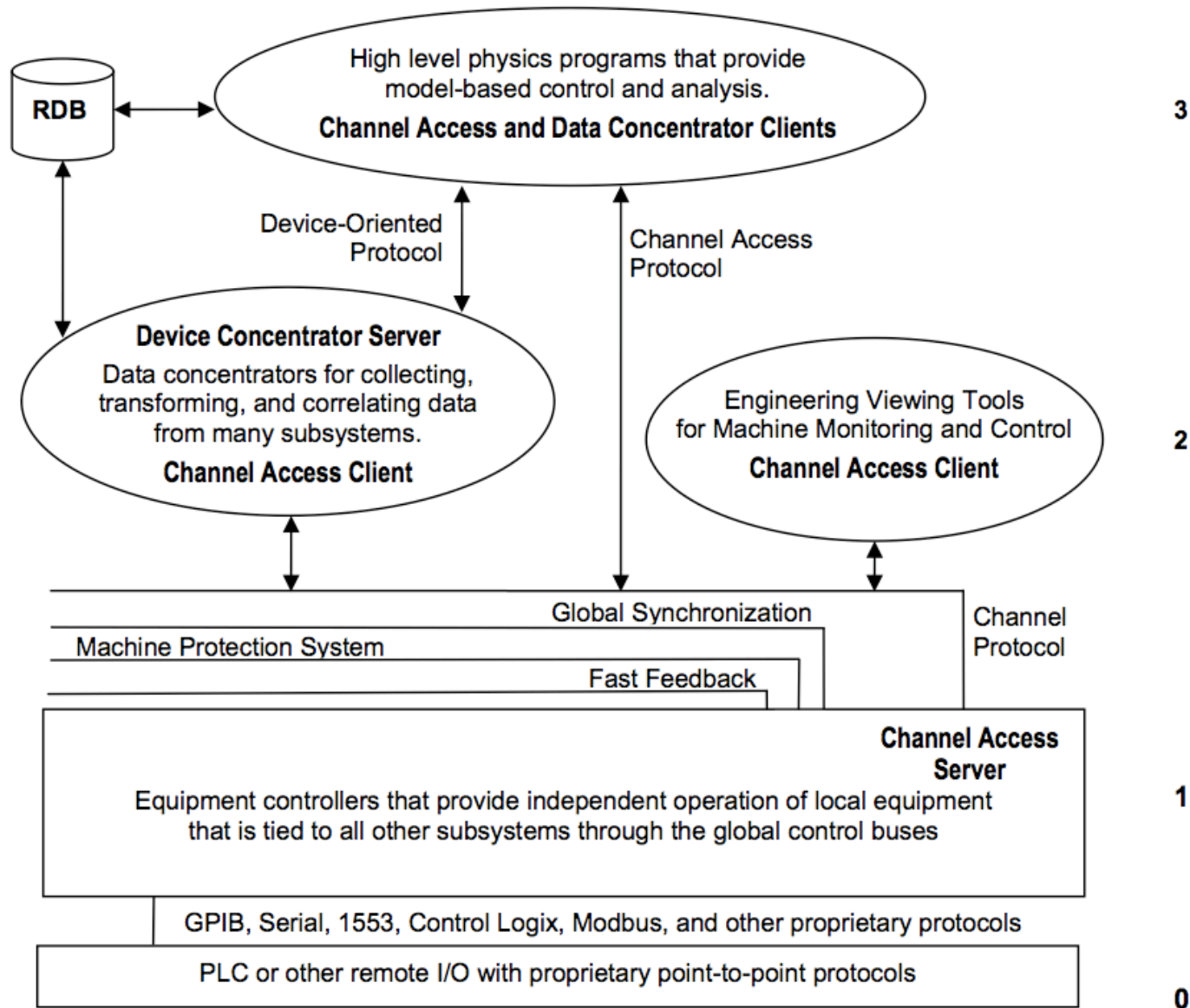


# The "Standard Model"



Generic Distributed Control System Architecture

# Recent Example



# Evolution of Distributed Systems

- mainframe/VAX
  - limited I/O drops lead to long analog signal runs
  - Centralized computing resources
- Standard Model
  - distributed I/O controllers
  - shorter analog runs
- Refinements to standard model
  - NAD: finer grained distribution
  - Multi-Tier/Middleware
  - New platforms: ATCA, etc...



# Functional Interfaces

## Communications

- Backplane (Internal)
- Network
- Real-time data\*
- Event\*
- Reference Clock\*\*
- Machine protect\*
- Utility\*
- Beamline Device I/O\*\*\*
- Power
- Environmental

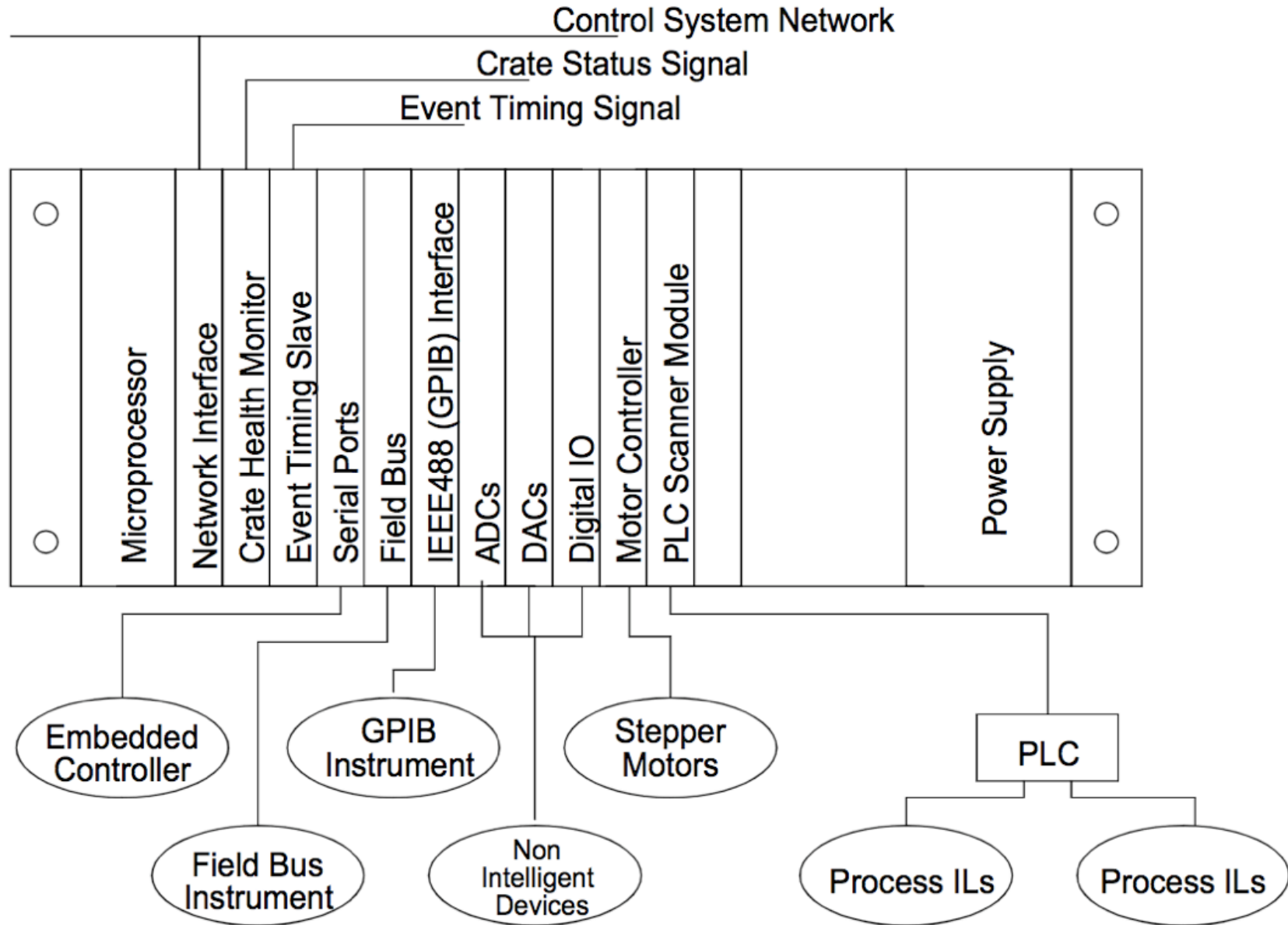
Typically specific to:

\* facility

\*\* machine

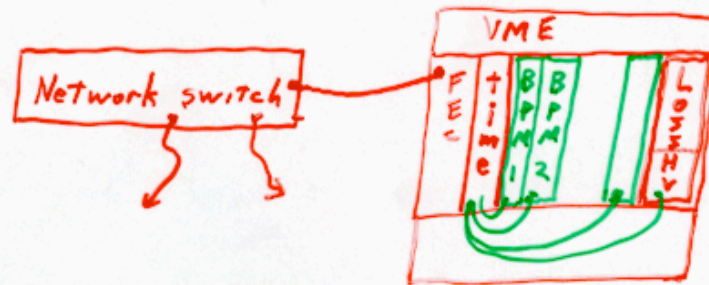
\*\*\* device

# Typical Crate

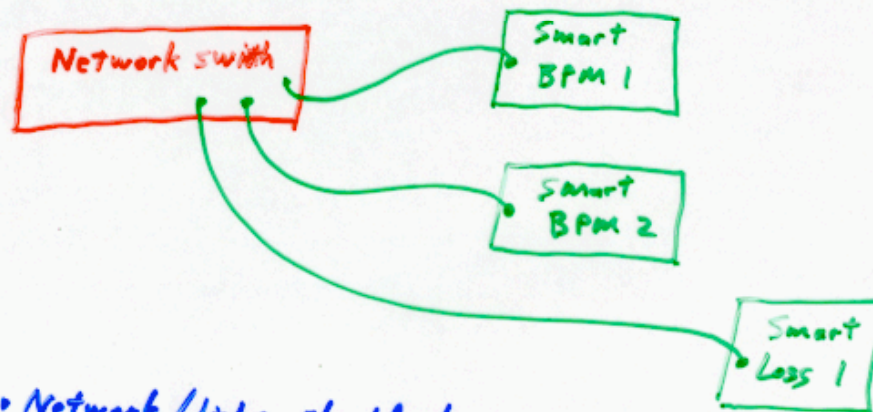


# Toward Network Attached Devices

Old:



New:



Once upon a time,  
there were  
transparencies...

- Network/links should become redundant
- smart boxes should share common platform

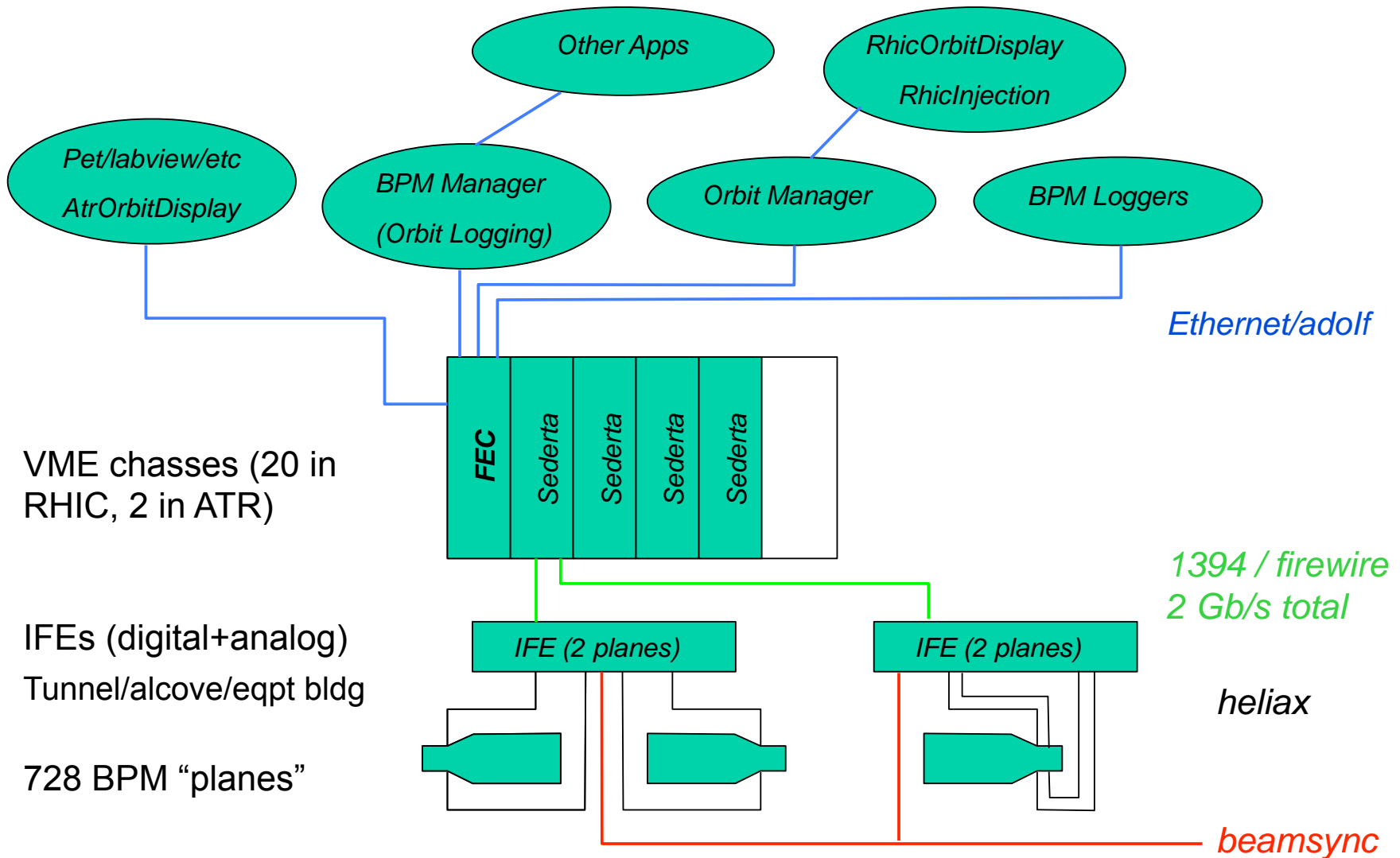
# Motivation for NAD

- **Shared resources** - tightly coupled
  - Tempting to make efficient use of modular infrastructure: pack with channels or share with multiple system types
  - System integration becomes more complex
- **NAD** - loosely coupled
  - limit unnecessary coupling between systems
  - vertically test individual units
  - integrate single units with focussed, well defined interfaces
  - Rapid SNS integration demonstrated success
- Even a subset of the concept can be helpful:  
i.e. get the mission critical vacuum interlock boards out of the experimental diagnostic crate, or integrate timing decoder on acquisition board in ATCA module

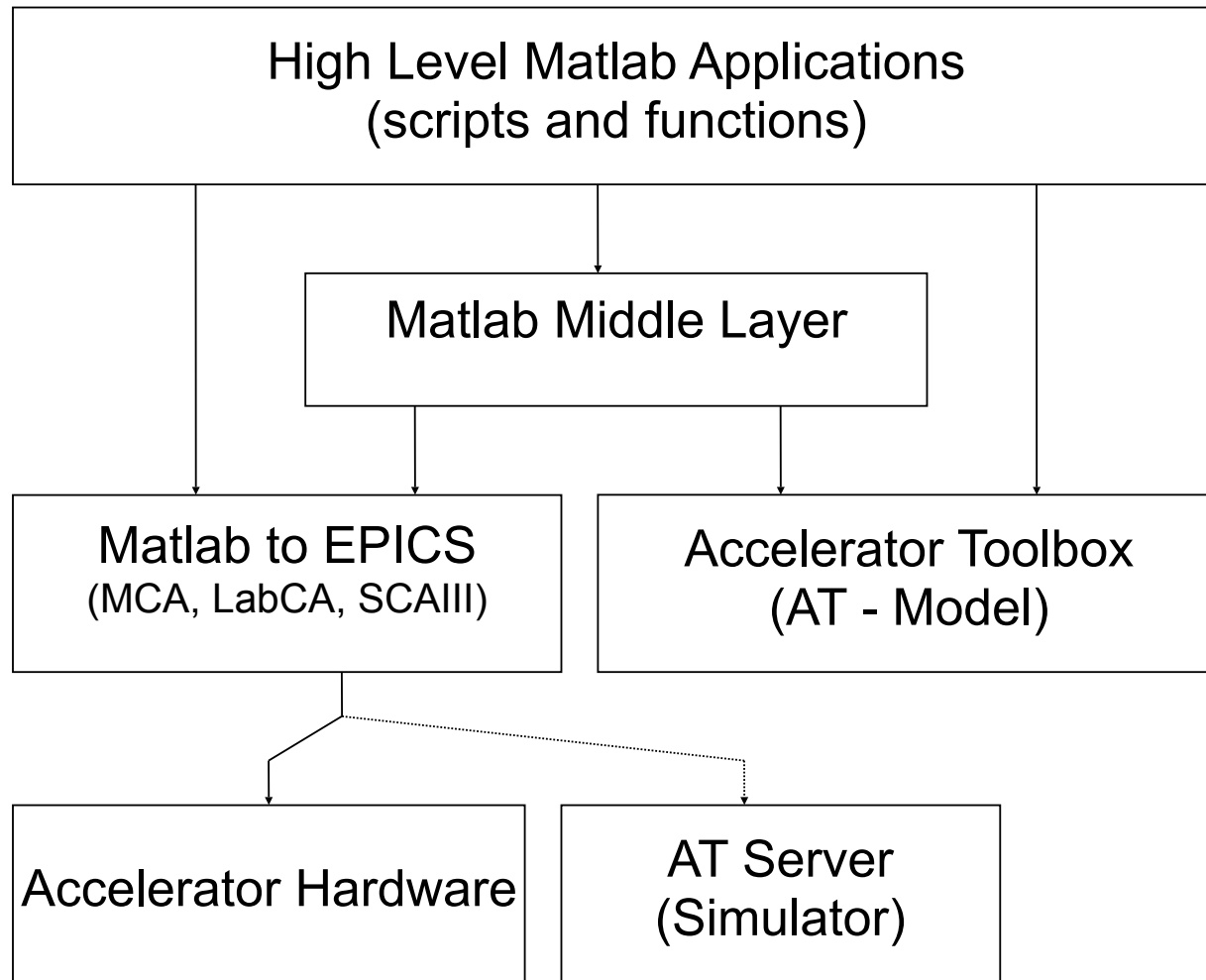
# Middleware

- Glue
- From component/device view to physics view
- Uniform access to multiple information sources (front end computers, online model servers, databases...).
- Could be soft real-time
- Complete solutions used heavily in the financial industry, less so in accelerators (but watch LHC)
- But many purpose-built implementations in use:
  - adapters
  - aggregators of requests (multiple clients) or responses (device collections particularly to support distributed high channel count systems)
  - correlators

# RHIC BPM System Overview



# Matlab MiddleLayer



# Physical Packaging

- Field replaceable units can be:
  - Modular
  - Self contained - i.e. rackmount, wall mount
  - Integrated - i.e. PS controller

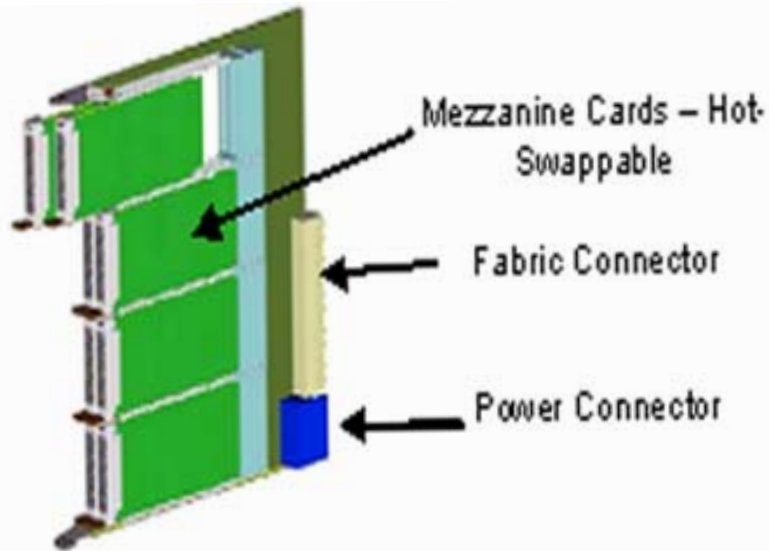
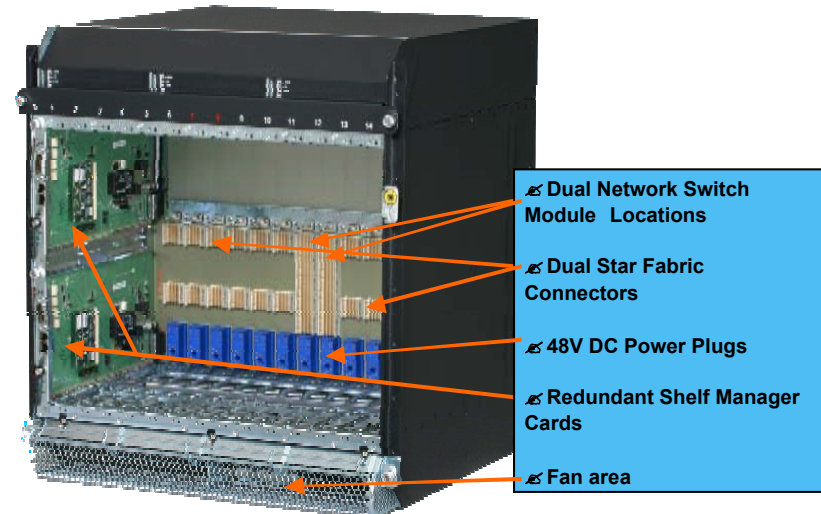


# Packaging issues

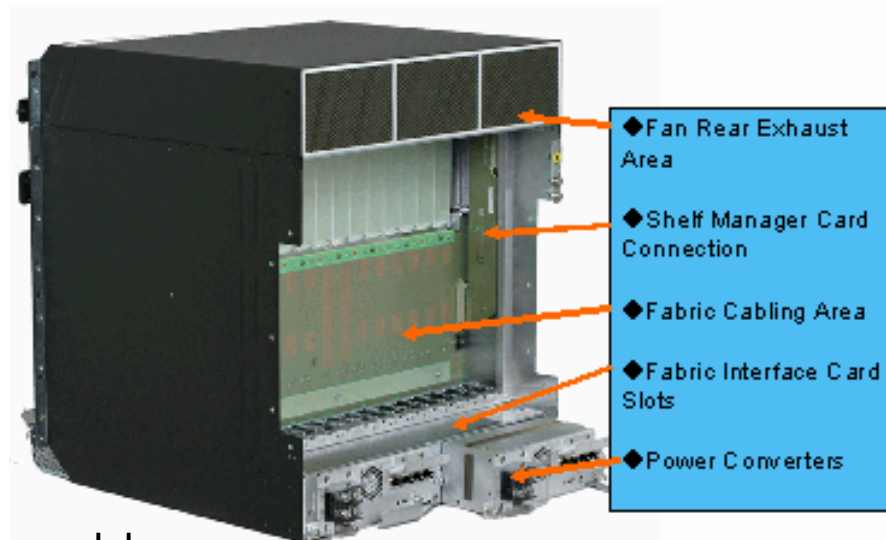
- Power
- Cooling
- Shielding
- Connections
- Partitioning
- Standardization
- Availability/MTTR

# Modular Packaging

- VME/PCI/cPCI
- VXI/PXI
- ATCA (shown)



Best MTTR with I/O on rear transition module



# Self contained

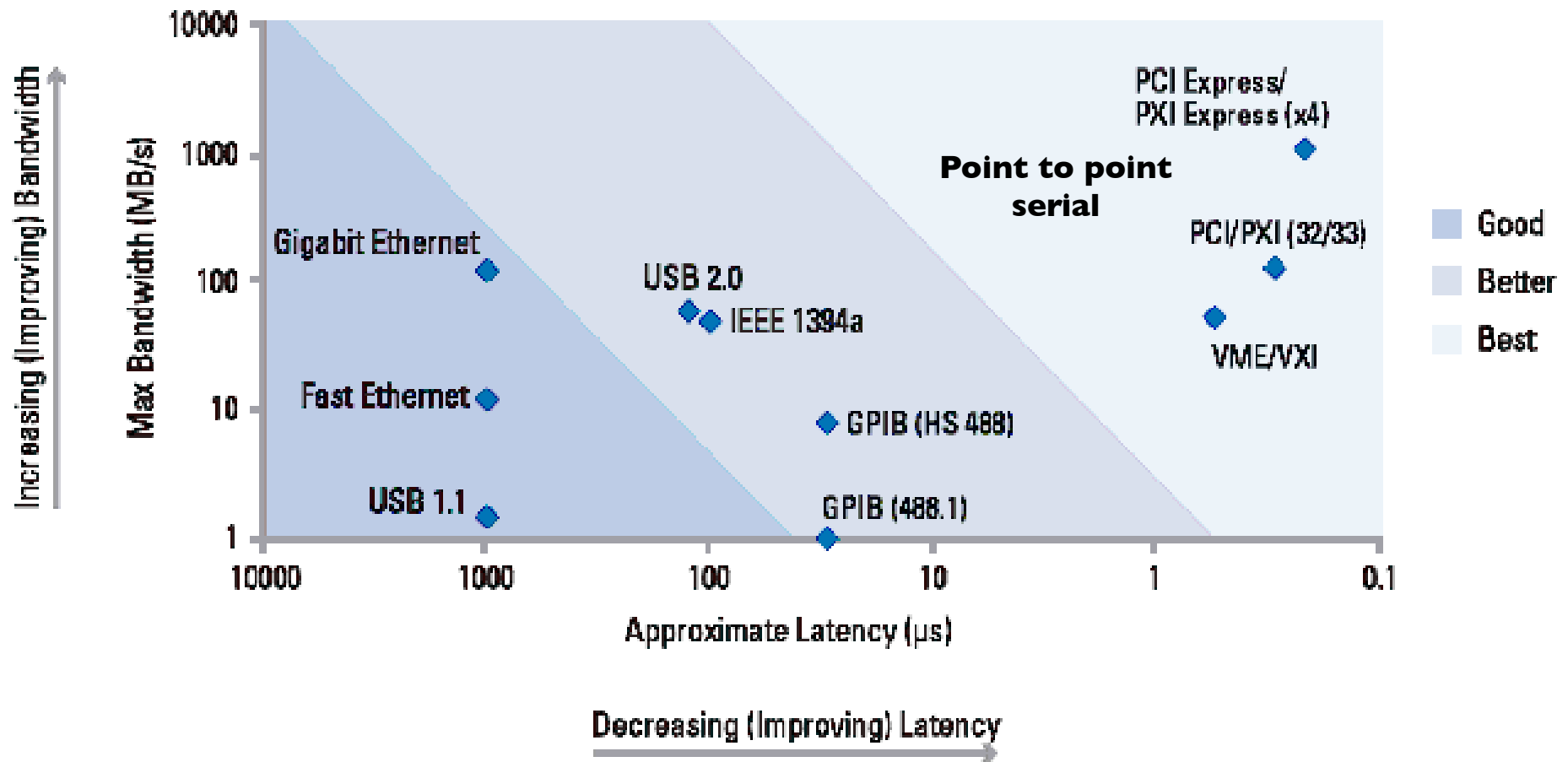
- Rack mount industrial PCs acting as NADs
  - could choose to stock prebuilt units as spares
  - changing I/O boards could lead to higher MTTR
- Chassis containing more integrated electronics
  - RHIC BPM (19" rackmount - in house)
  - Libera (19" rackmount - vendor supplied)

# Communications

# Communications Trends at Accelerator Facilities

- End of traditional parallel backplane bus paradigm (Announced every year since ~1989)
  - VME-PCI still there
  - watch PCI Express, RapidIO, ATCA
- Commercial networking products
  - DAQ 94' Conference: ATM, DS-Link, FibreChannel, SCI
  - Today: Gigabit Ethernet (1, 10, 30 GB/s)
- The ideal processing / memory / IO bandwidth device
  - The past: Transputers, DSPs
  - Today: FPGAs - Integrates receiver links, PPC, links, PPC, DSPs, memory.....
- Point-to-point link technology
  - The old style: Parallel Copper –Serial Optical
  - The modern style: Serial Copper –Parallel Optics, >3Gb/s today, 10Gb/s in demonstration

# Latency and Throughput

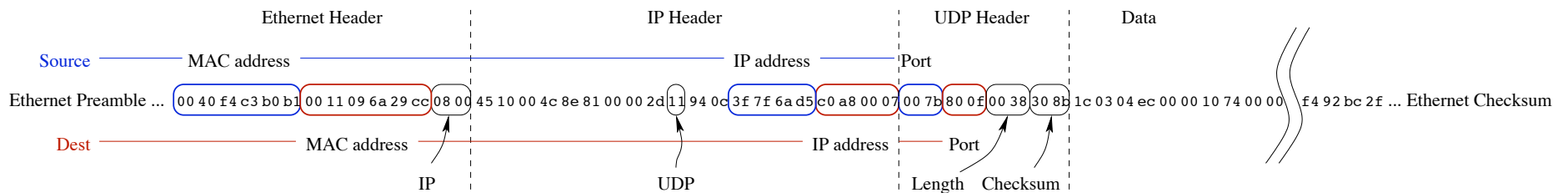


# Intra-chassis Standards

- CAMAC
- VME/VXI
- PCI/PXI
- IP, PMC
- PCIe
  - multilane serial
- ATCA
  - serial fabric
  - PCIe, ethernet, etc

	ATCA	PCI (long)	VME 6U
Board Area cm <sup>2</sup>	995	316	373
Power Watts	200	10/25	30
Bandwidth I/O Gb/s	20 full duplex	4.3 66MHz 64 bits	2.4 VME 2eSST
Front panel H * W cm	30 * 2	8 * 1.2	21.5 * 2
Component Height mm	21.33	14.48	13.72

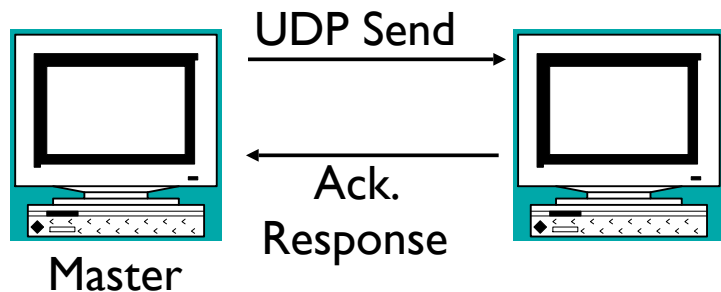
# UDP over Ethernet



- TCP: http, EPICS Channel Access
- UDP: deterministic communication, streaming multicast
- Either one: COTS routers, diagnostic tools
- intended for implementation with processor, but simple UDP demonstrated in hardware



# Latency test for user mode process



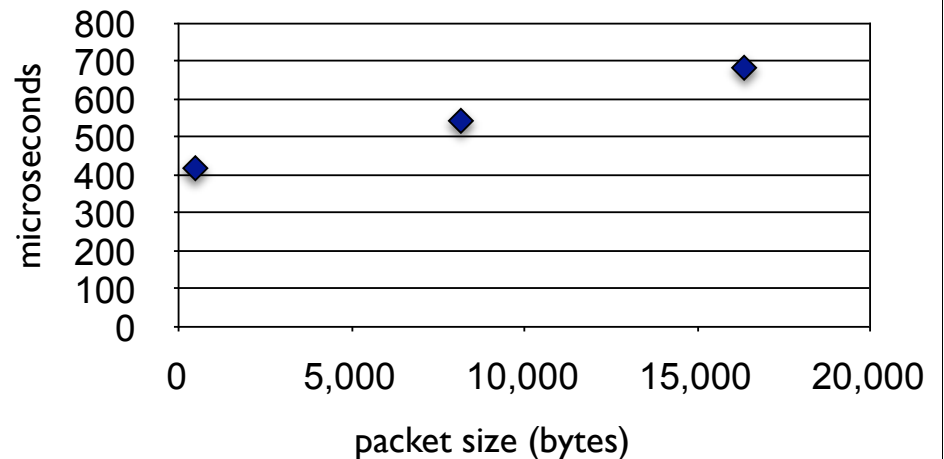
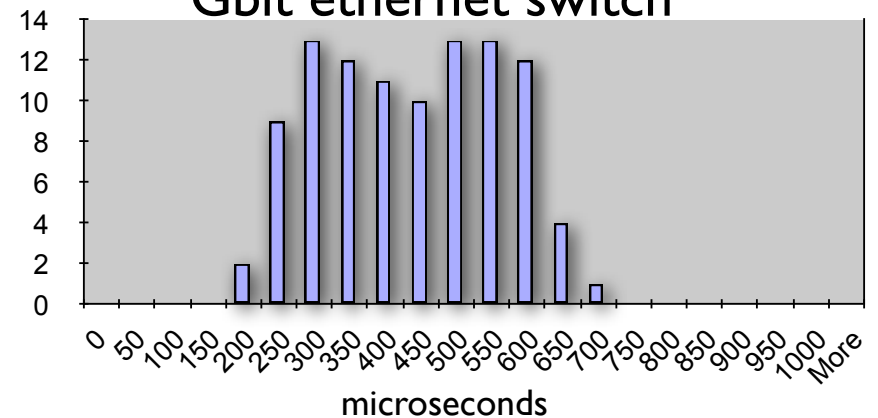
Measure time difference  
UDP Send – Ack Response  
At Master using high res. timer

300usec round trip latency.  
one hit at 1.7msec

Point to point, no switch

2002 Vintage Dell running  
Standard Windows 2000

Round Trip with 2005 vintage Tyan,  
Gbit ethernet switch

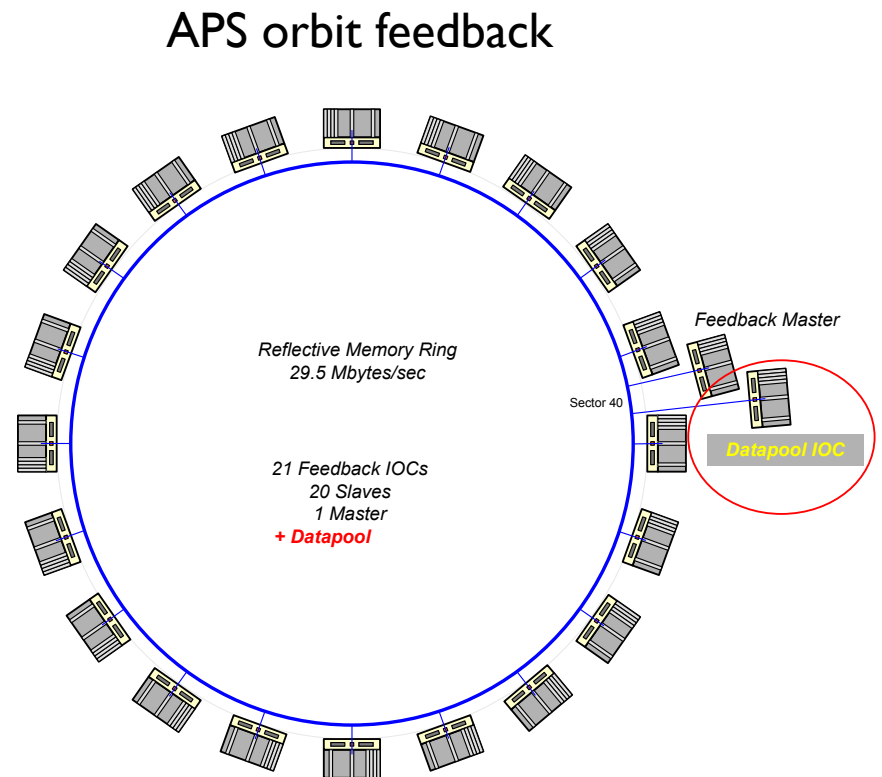


# Response Time

- What affects response time?
  - Network collisions (use switches/synchronization)
  - Thread Quantum (improved with preemptive interrupt capable OS: RTOS, Solaris, Windows, and new Linux Kernels  $\geq 2.6$ )
  - Context Switching times (10usec-100usec)
  - Packet size (Minor for small packets)
- Example using RTOS:
  - ALS orbit feedback system demonstrates negligible effect of UDP packet jitter @ 100Mb/s. Currently 1.1kHz rate. Expect to achieve 5 kHz rate @ Gb/s

# Reflective Memory Example

- Also known as Replicated Memory
- Each participant in the reflective memory network has a reflective memory module.
- Each module can be written and read as simple memory
- Property:
  - Anything written to a location in one reflective memory module appears (after a loop transit time) in the same location in all attached modules.
  - Transfer rate 29.5 Mbytes/second
- All Nodes see the same memory image
- Designed for Real-time Performance – Latency is minimized
- Typical implementation: Commercial PMC card on VME carrier or VME CPU board
- Calculated loop settling time = 21.4 usecs (22 nodes and 1200 meters of fiber)



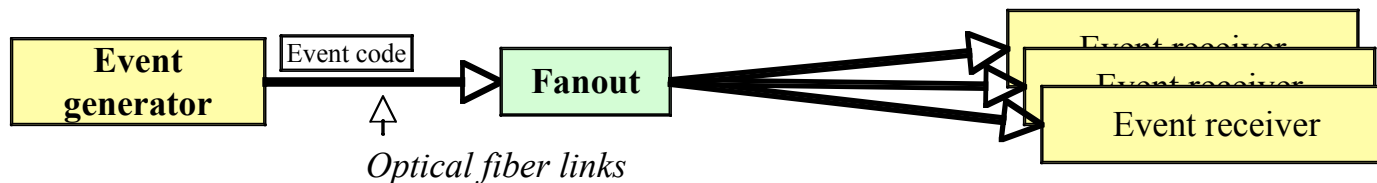
# Timing/Event Systems

- Functions:
  - Coordination of actions
  - Correlation of data
- Other aspects to consider:
  - Multiuser operation
  - Frequency change if beam synchronous
- Scalability: information should reach ALL devices
- Implementations
  - FNAL/AGS/RHIC/SNS separated function
    - Event Link, TClock
    - RTDL, MDat
- SLS, Diamond: Events/info on Ethernet PHY
- Other options
  - UDP broadcast
  - Direct fiber
  - “tagged” RF reference clock

# Typical Event System

The system provides:

- global distribution of events to all systems that have a receiver
- trigger and gate signals to hardware
- synchronized timestamp facility
- software sequencing by triggering channels to process from events
- software can be used to send events



The stimulus to send an event can be:

- pulse on a hardware input
- software event (write to a register)
- an entry in an event playback RAM .

When an event code is received the receiver can:

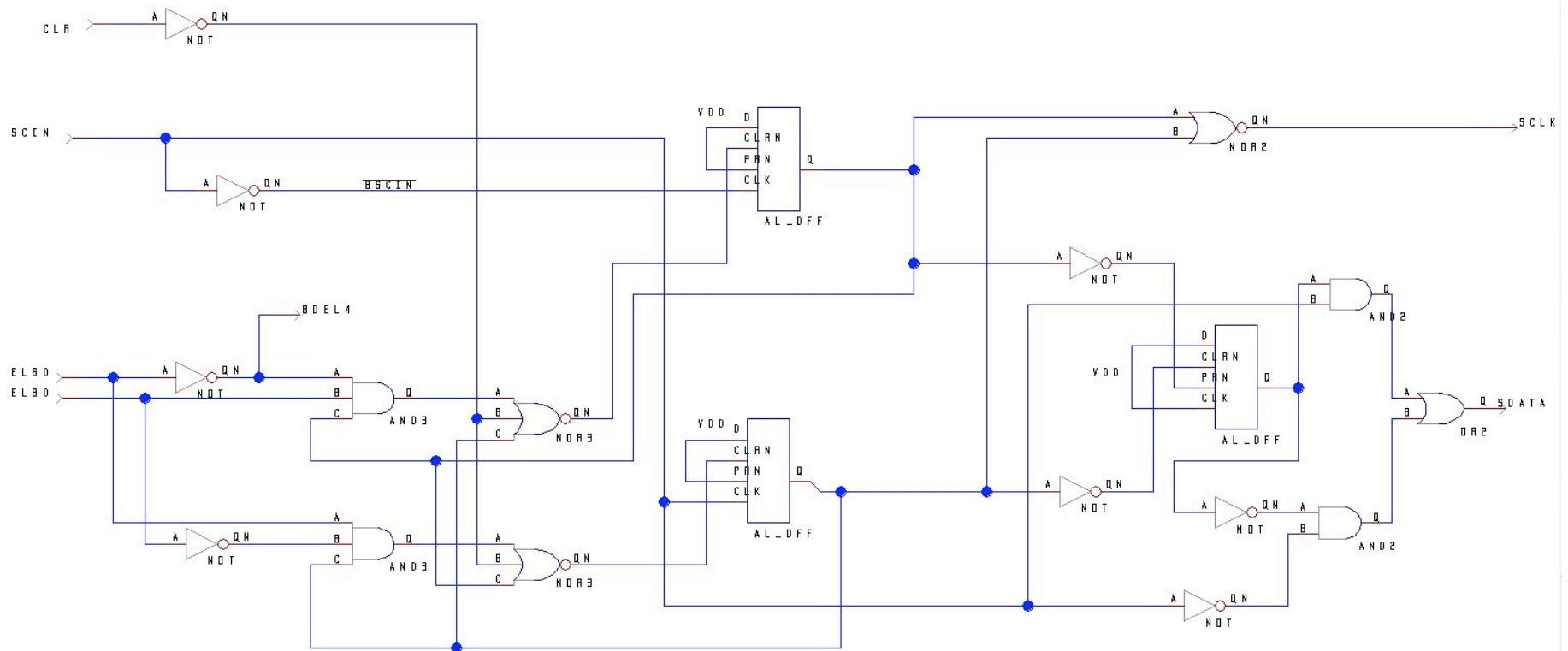
- output a pulse, of specified delay and width
- trigger a software action (process an EPICS record)

Each event receiver can be programmed to respond in a different way to the same event code.

# Event Timeline

- Coles/Shea diagram

# Example receiver



For  $\ll 1\%$  utilization of modern FPGA, get events within signal processing system.

This example is in-house standard. Trend is to adopt standard PHY.

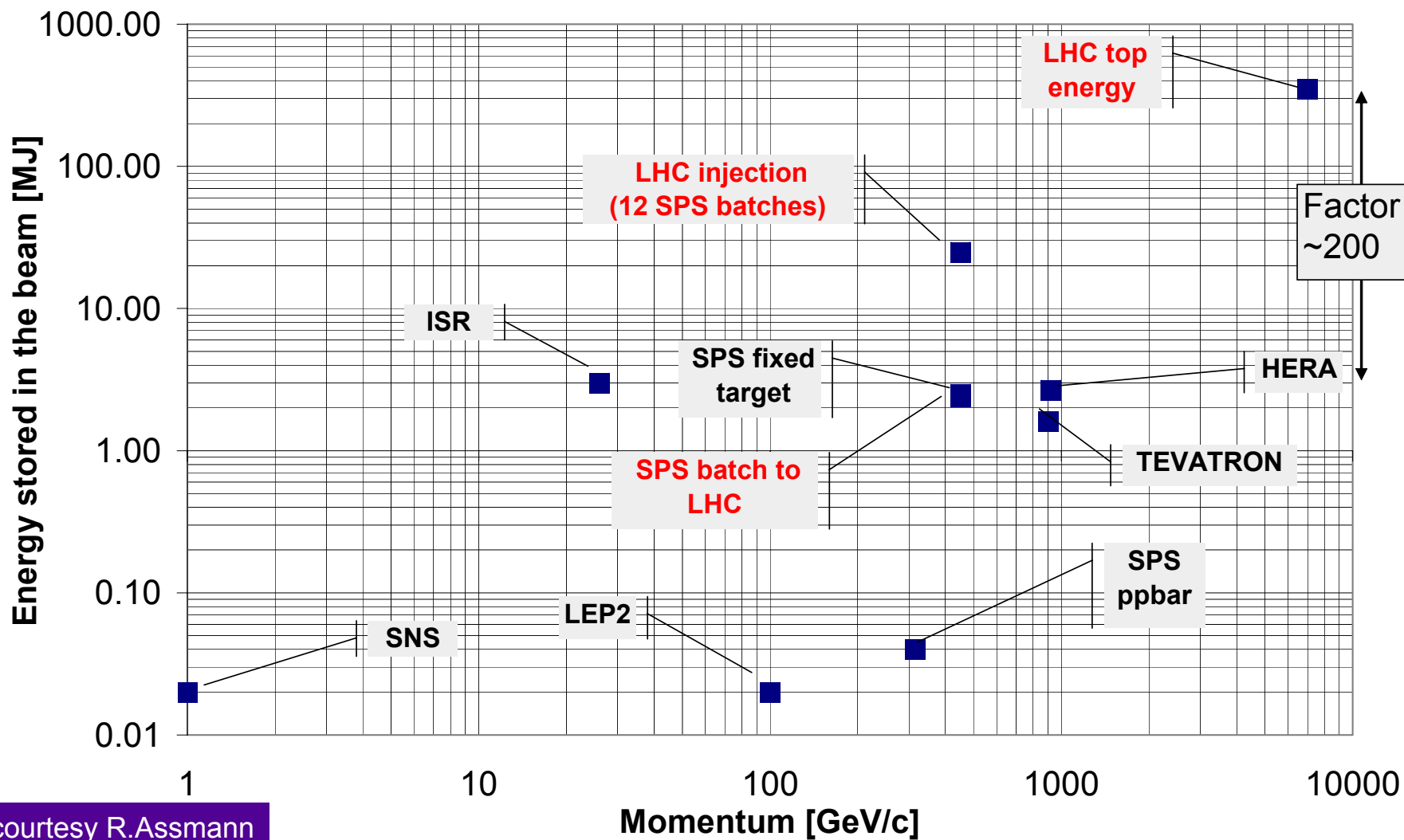
# Multimode Operation

- AKA: Multimode, Multiuser, Pulse to Pulse Modulation, Flavors, ...
- Different Beam Parameters
- Different feedforward tables
- Different results updated



# Machine Protection

# Motivation

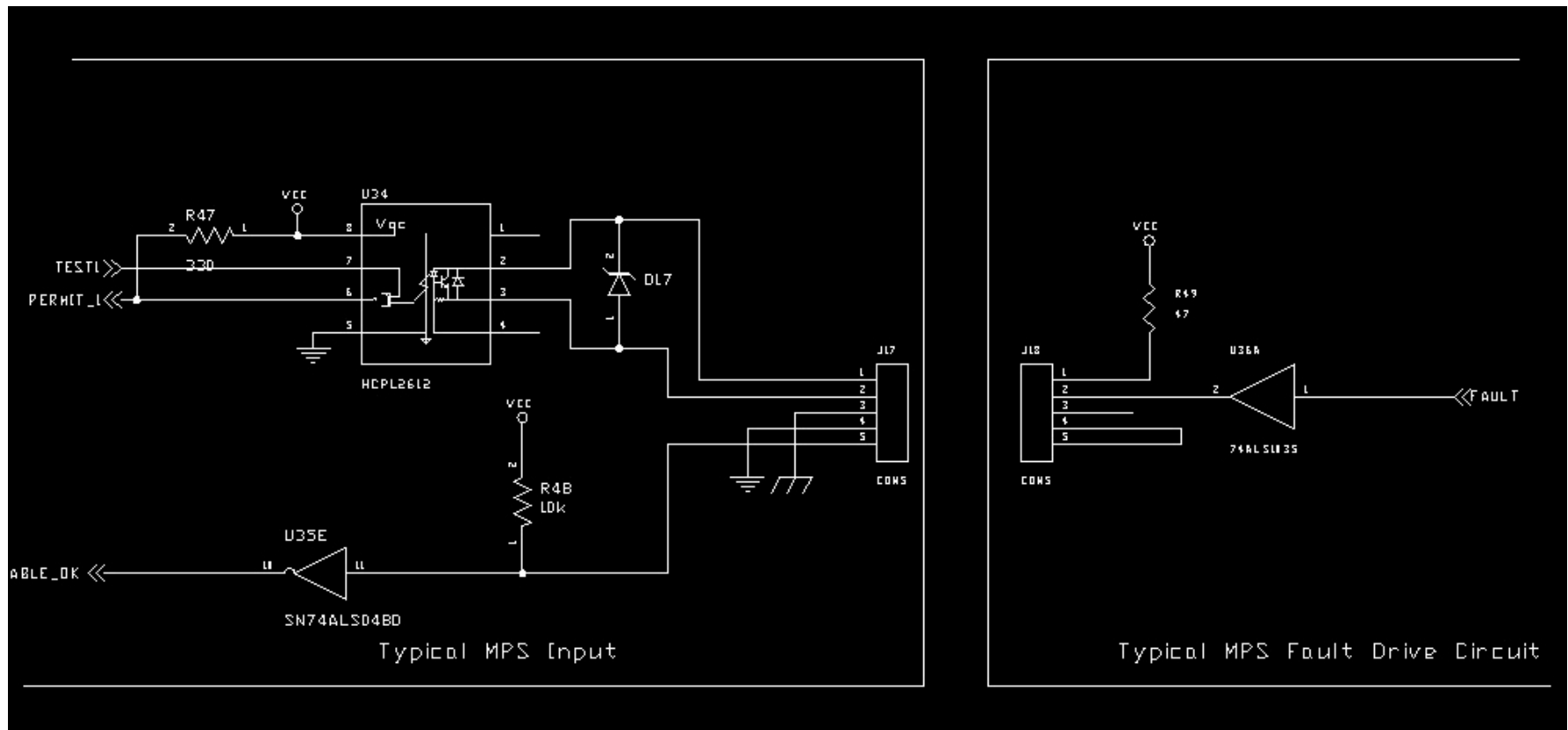


courtesy R.Assmann

# Issues

- Subsystem drives MPS input
  - DSP detects loss of control
  - Differential current measurement detects beam loss
- Subsystem responds to MPS trip
  - Communicated on timing or real-time data broadcast system
  - Circular buffers

# Interface to MPS

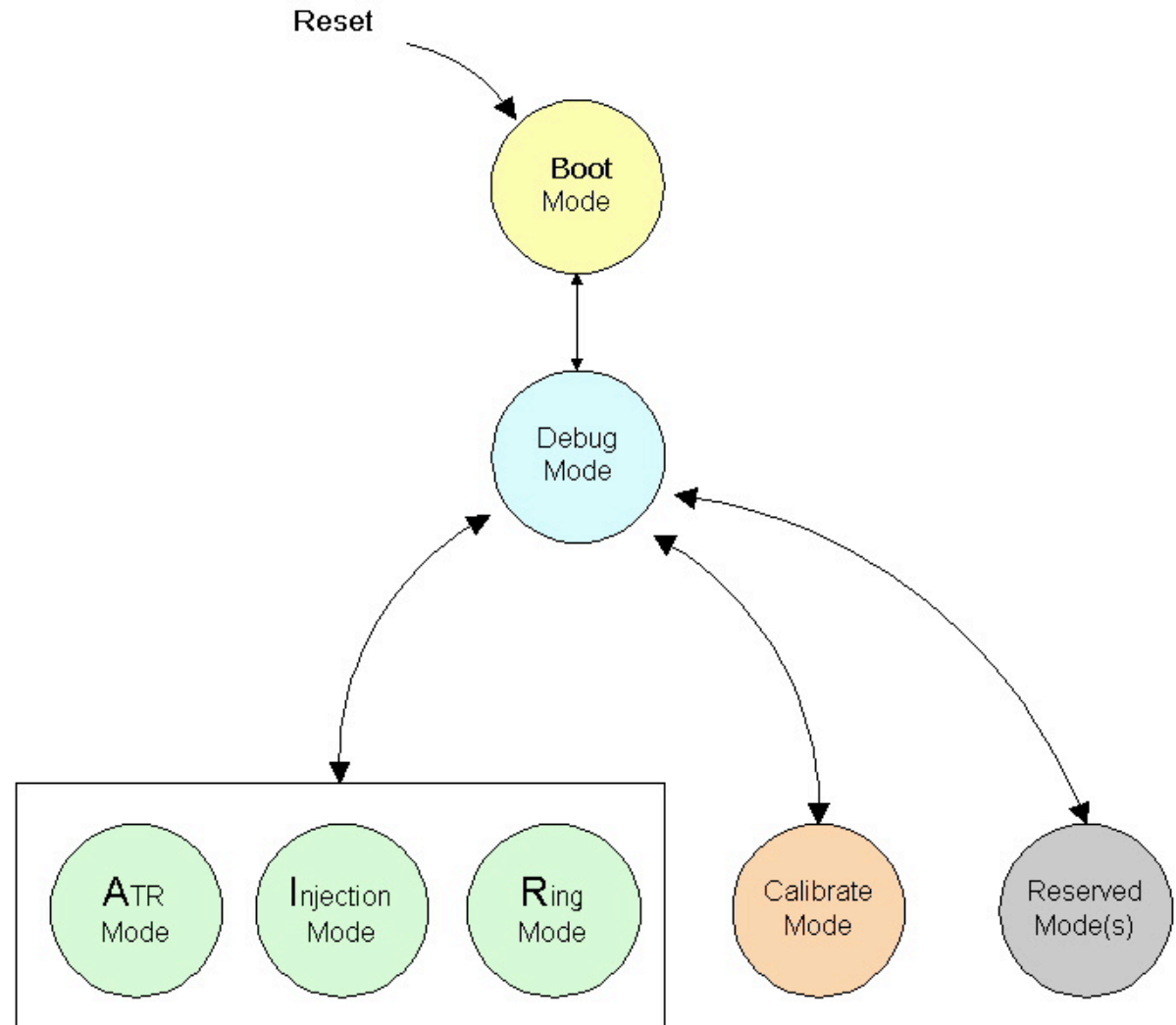


# Utility

- Temperature monitor
  - for fault monitoring (filter change time)
  - for correction of thermal effects
- Fan Speed
- Power supply current and voltage
- Heartbeat
  - monitored
  - optional watchdog timer for automatic reset
- Remote Power and Reset
  - Personnel access restrictions
  - Facility Size
    - MTTR

# Safe Reconfiguration

- avoiding the “turn antenna away from earth” command.
- protected boot memory containing communication code
- permanent communication subsystem (tiny IOC on a card, hard core on FPGA,...)
- Out of band communication as an option - but may add interconnect or cable in some cases

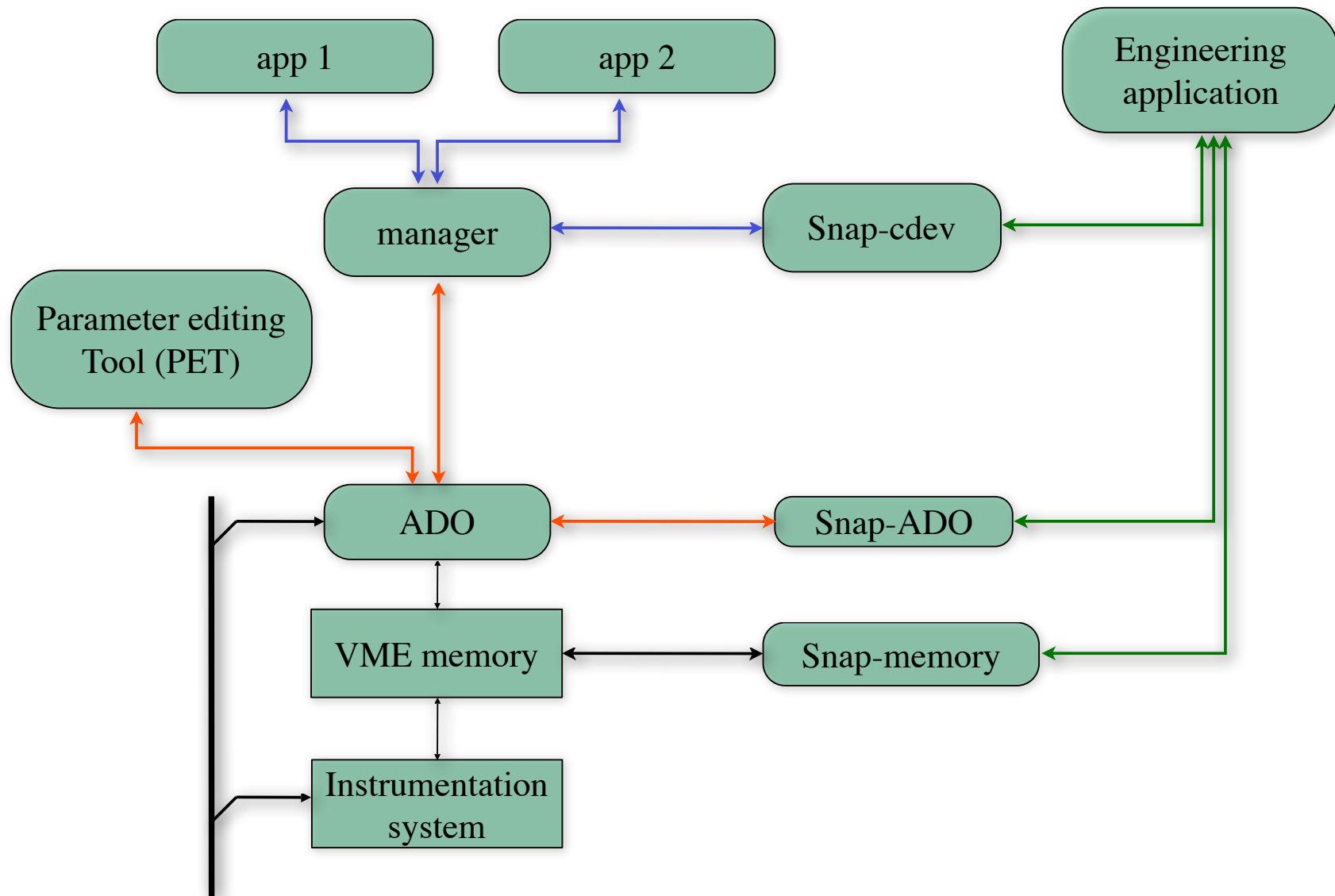


# Remote Debugging

by example

- RHIC BPM: Overview
  - Most electronics in radiation area
- SNS LLRF: In depth
  - SNS Equipment gallery put under access restrictions during commissioning

# RHIC Debugging Tools



Event links and RTDL



# Interface to LLRF Module

- Register based
  - Random read and write access to ease debugging
    - No "don't write during ..." conditions that require tricky timing.
    - No write-only registers that one cannot verify.
- Message based interfaces were avoided
  - Because they are harder to debug.
  - Also harder to implement in an FPGA, requiring parser.
  - One can easily trigger VME/VXI/PCI backplane analyzer on register access. Messages are harder.

# Register Documentation

## Register† LLRF\_DDS\_FREQ

The offset frequency for the on-board Direct Digital Synthesizer (DDS). This is a signed 16-bit number that adjusts the frequency of cavity operation, relative to the Master Oscillator, in the range of  $\pm 625$  kHz. One bit corresponds to 19.073486328125 Hz exactly (assuming the RF sampling clock is a perfect 40 MHz).

## Register LLRF\_STATUS

Read-only bit map of module status.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	-	CLP	FLT	LRC2	LRC1	PLL	IL	RF_ON	KCM

CLP: Latched indicator of occurrence of any premature termination of RF due to output magnitude clipping fault. Cleared at beginning of each pulse.

FLT: Latched indicator of occurrence of any premature termination of RF due to faults within the pulse. Cleared at beginning of each pulse.

LRC2,LRC1: FCM-specific readout of the configuration switches that define its position. 0=Left, 1=Center, 2=Right, 3=Invalid. Non-FCM systems read 0.

PLL: Instantaneous readout of the same signal described in the PLL bit of the

# "Single Source"

- Perl scripts convert Verilog register definitions into C++ include file, EPICS Database config, ...

```
register-map
SET_I      14'b00000000100000
SET_Q      14'b00000000100001
DDS_FREQ   14'b00000000100010
CONFIG     14'b00000000100011

llrf_defines.h
/* automatically generated by intercon.pl ../source/register
#define LLRF_CONFIG_ROM      (0x0000)
#define LLRF_ERRORS          (0x0010)
#define LLRF_STATUS          (0x0011)
#define LLRF_UCAPE           (0x0012)
#define LLRF_DCAPE           (0x0013)
#define LLRF_FCAPE           (0x0014)
#define LLRF_INT_STATUS      (0x0015)
#define LLRF_TTLTRG_MON      (0x0016)
#define LLRF_PEAK_ERR        (0x0017)
#define LLRF_PEAK_OUT        (0x0018)
#define LLRF_TOTALIZER       (0x001c)
#define LLRF_SHADOW          (0x0020)
#define LLRF_SET_I           (0x0020)
#define LLRF_SET_Q           (0x0021)
#define LLRF_DDS_FREQ        (0x0022)
#define LLRF_CONFIG          (0x0023)
```

# Debugging Registers: Console

- Via serial line or "telnet"

```
test-oper@rftf-test-opi-cow4:~
test-llrf-ioc-vxi> vxiHelp
VXI device support, version 2004-09-23 w/ write checks.

VXI helper routines:

NOTE: Test mode, A24 access actually
uses local RAM when the variable use_A24_TEST is set to 1.
Right now, use_A24_TEST is set to 0

vxiHelp
vxiInfo(int la) - check given LA
vxiRegister(const char *name, int la) - register LA (for dbior)
vxiReport(int level) - report on registered LAs
vxiSearch() - Tests all LAs
Word *vxiBoardAddr(int la) - Get A16 base as seen by CPU
void *vxiTestA24(int la) - Check read access to A24 mem
Bool vxiPeek(int la, int reg) - Read A16 register
Bool vxiPoke(int la, int reg, Word value) - Write A16 register
void vxiMapA24(int la, unsigned long start) - Assign & enable A24 base addr.
Word *vxiA24Addr(int la) - Get A24 base as seen by CPU
Bool vxiPeekA24(int la, int word) - Read A24 register
Bool vxiPokeA24(int la, int word, Word value) - Write A24 register
worddump(Word *addr, int count) - Dump memory by using 16-bit access
value = 83 = 0x53 = 'S'
test-llrf-ioc-vxi>
test-llrf-ioc-vxi> vxiPeekA24(0xD4, 0x22)
LA 0xD4, A24 word 0x22 = 0x0000
value = 1 = 0x1
test-llrf-ioc-vxi> █
```

# Debugging Registers:

- Note:** Our hardware returns value **0xBAD** for undecoded registers (pullups & downs on data lines in case FPGA doesn't drive them)

The screenshot shows the 'VXI Registers' application window. The title bar reads 'VXI Registers'. The main content area is titled 'VXI Registers, SCL\_LLRF:IOC01a'. It is divided into sections for A16 and A24 read/write operations, each with input fields for LA, Register, and Value, and corresponding readback values and bit patterns.

**A16 Read:**

LA (hex)	Register(hex, dec)	Value (hex)	(ASCII)	(Bits)
0xd4	0 0	0x4fa0	O_	[Bit pattern]
0xd4	0x1 1	0x1f4d	_M	[Bit pattern]

**A16 Write:**

Register(hex, dec)	Value (hex)	Readback
0 0	0	0x4fa0
0 0	0	0xbad

**A24 Read:**

Register(hex, dec)	Value (hex)	Readback
0xd4 0x22 34	0	0

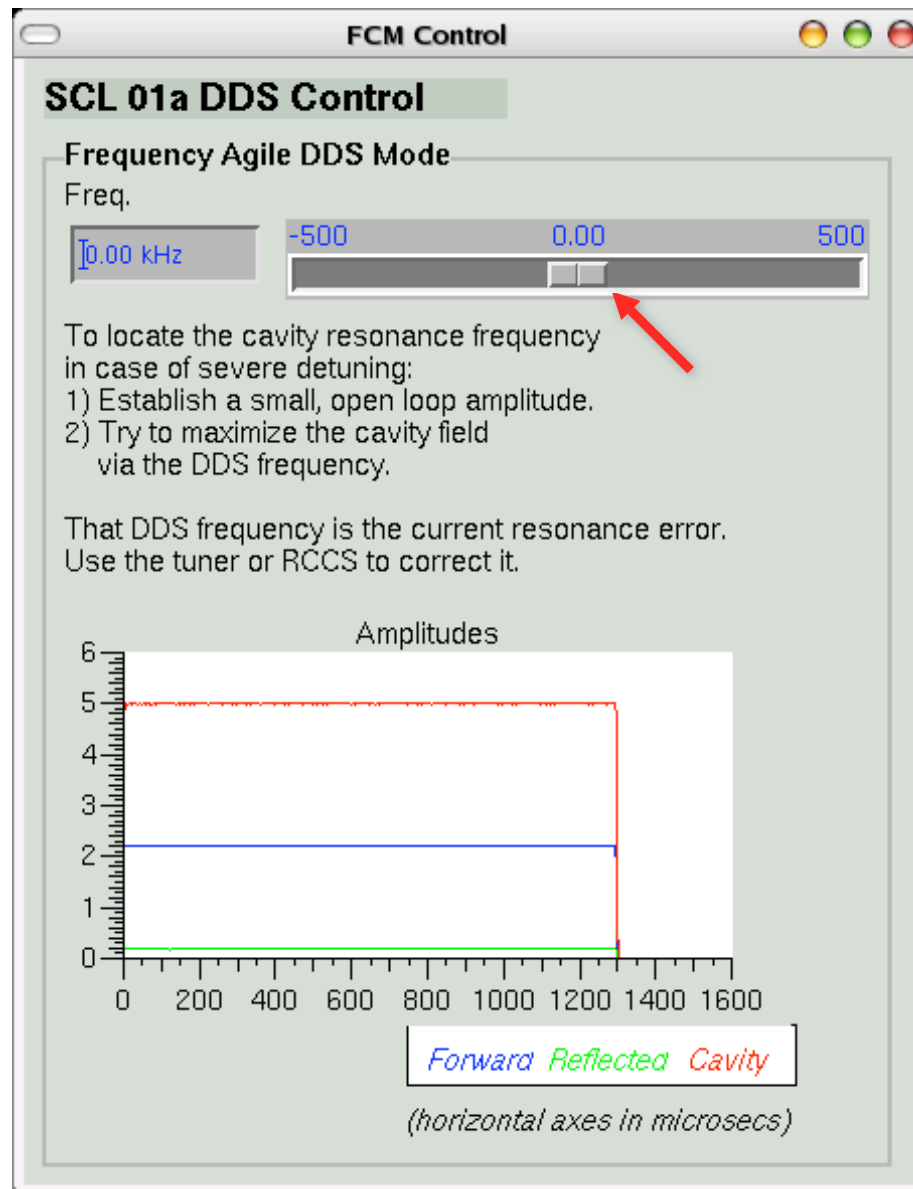
**A24 Write:**

Register(hex, dec)	Value (hex)	Readback
0xd4 0x22 34	0	0

Use WORD offsets (2-byte) f. Registers, press Return to send.

A red arrow points to the '34' register value in the A24 Read section. A blue arrow points from the '0xbad' readback value in the A16 Write section to the bit pattern display in the A24 Read section.

# End-User Register Access



Title\_date

# Other Debug Tools

- **EPICS Sequencer and Database**
  - Online, remote access to internal state
  - View the "raw" value behind some displayed data
  - Timestamps
  - Anything can be displayed on "Strip chart", or added to an archive tool for later analysis - good for infrequent events and unanticipated correlations
- **Custom C/C++ Code**
  - Access to internal data must be specifically added to the code
    - "Report" methods
    - Debug flags
    - Time stamps

# Source Code Control



# Version Control Benefits

- **Serves as repository**
  - Develop some bugfix on laptop, merge that into copy on the main development machine
  - Deploy "latest" version onto linac server
- **"Roll back to state of January 18, 2006"**
- **View history of changes, compare different versions**
  - "When did we add this behavior?"
  - "How was this handled 2 years ago?"

# CVS - Concurrent

- **A free, open source version control system**
  - Available for every operating system
  - Stable
  - Command-line, Emacs, Eclipse, ...
- **Handles text very well**
  - Plain ASCII or LaTeX documents
  - EPICS Sequencer and Database sources
  - C/C++
  - Verilog/VHDL
  - Front-end computer startup files
- **For binary files, only date & comments available, no comparisons possible**
  - Images, FPGA bitfiles, LabVIEW, ...
- **Old**
  - "subversion" might be better at handling directories

# CVS under Eclipse: Verilog

The screenshot displays the Eclipse IDE interface for comparing two versions of a Verilog file, `fdbk_loop.v`. The window title is "C/C++ - Compare fdbk\_loop.v 1.12 and 1.11 - Eclipse SDK".

The left sidebar shows a project tree with various Verilog files. The main editor area is split into two panes, each showing a different revision of the file:

- Repository file: fdbk\_loop.v 1.12** (left pane):

```
Repository file: fdbk_loop.v 1.12
(e4[12:0], 3'b000) : {e4[15:10], 5'b000000} : {e4[15:10]
wire [15:0] e4_ls5 = ((e4[15:10]==6'b000000) | (e4[15:10]
(e4[10:0], 5'b000000) : {e4[15:10]
// wire [15:0] e4_int = int_scale_sc ? e4_shift
wire [15:0] e4_shift = int_scale_sc[1] ? e4_ls5
wire [15:0] e4_int = int_scale_sc[0] ? e4_shift
wire [13:0] fdbk_err_wide = {integrate_out[20], in
// arithmetic saturation from 14 bits down to 13 b
wire [11:0] fdbk_err = ((fdbk_err_wide[13:12]==2'b
fdbk_err_wide[12:1] : {fdbk_err_wide[13],{11{
always @(posedge clk40) fdbk_err_out <= fdbk_err;

wire [11:0] peak_err_narrow;
trip trip(
.clk(clk40), .inval(e2[10:2]), .trip_thresh(tr
.gate(trip_gate), .reset( trip_reset ),
.tripped(err_tripped), .peak_val(peak_err_narr
);
assign peak_err = {peak_err_narrow, 4'b0};

// gauging for output magnitude
wire [11:0] peak_out_narrow;
trip clip(
.clk(clk40), .inval(fdbk_err[11:3]), .trip
.gate(feedback_on), .reset( trip_reset ),
.tripped(out_clipped), .peak_val(peak_out
);
assign peak_out = {peak_out_narrow, 4'b0};

endmodule
```
- Repository file: fdbk\_loop.v 1.11** (right pane):

```
Repository file: fdbk_loop.v 1.11
// ff_pipe <= feedforward_data;
ff_pipe <= ff_data_sat; // hma:
integrate_input <= {ff_pipe,5'b0} + (corrupte
end
// Combine low-latency linear feedback with the i
// Saturation and sign extension are claimed vali
// wire [13:0] fdbk_err_wide = {integrate_out[20]
wire [15:0] e4_ls3 = ((e4[15:12]==4'b0000) | (e4
(e4[12:0], 3'b000) : {e4[15:10], 5'b000000} : {e4[15:10]
wire [15:0] e4_ls5 = ((e4[15:10]==6'b000000) | (
(e4[10:0], 5'b000000) : {e4[15:10]
// wire [15:0] e4_int = int_scale_sc ? e4_shi
wire [15:0] e4_shift = int_scale_sc[1] ? e4_ls5
wire [15:0] e4_int = int_scale_sc[0] ? e4_shi
wire [13:0] fdbk_err_wide = {integrate_out[20], i
// arithmetic saturation from 14 bits down to 13 b
wire [11:0] fdbk_err = ((fdbk_err_wide[13:12]==2'b
fdbk_err_wide[12:1] : {fdbk_err_wide[13],{11{
always @(posedge clk40) fdbk_err_out <= fdbk_err;

wire [11:0] peak_err_narrow;
trip trip(
.clk(clk40), .inval(e2[10:2]), .trip_thresh(tr
.gate(trip_gate), .reset( trip_reset ),
.tripped(err_tripped), .peak_val(peak_err_narr
);
assign peak_err = {peak_err_narrow, 4'b0};

endmodule
```

At the bottom, the CVS History table shows the following entries:

Revision	Tags	Revision Time	Comment
Previous			
*1.12		2/14/07 12:13 PM	firmware with output clipping
1.11		06-10-05 10:06:06 AM	get new firmware

# FPGA Firmware Handling

- **Verilog sources are in CVS**
  - Full benefit of version comparisons
- **Bitfiles also in CVS as 'binary'**
  - No insight into changes, but since each "place-and-route" creates different bitfile, it's good to keep a copy of the specific bitfile
- **Front-end computer programs FPGA**
  - Loads bitfile via network

*(For machine protection related FPGA, bitfile is in local EEPROM)*

# Configuration Control

# SNS Device Database

Active configuration file stored here.  
Supported by rest of purple tables which include historic.

Diagnostics IOC Active Configuration

IOC Config Type Id (FK)	VARCHAR(40)	NOT NULL
Config File Nm	VARCHAR(255)	NOT NULL
Config File Loc	VARCHAR(128)	NULL
Config File Nr	NUMERIC(9,0)	NULL
Contents	LONG VARCHAR	NULL
Modify Date	DATE	NOT NULL
Mod By Usr	VARCHAR(64)	NOT NULL
Editable Indicator	CHAR(1)	NOT NULL

Diagnostics IOC Configuration History

IOC Config Type Id (FK)	VARCHAR(40)	NOT NULL
Config File Nm	VARCHAR(255)	NOT NULL
Config File Loc	VARCHAR(128)	NULL
Config File Nr	NUMERIC(9,0)	NULL
Contents	LONG VARCHAR	NULL
Modify Date	DATE	NOT NULL
Mod By Usr	VARCHAR(64)	NOT NULL

Diagnostics IOC Parameter Group

IOC Config Type Id (FK)	VARCHAR(40)	NOT NULL
Param Grp Nm	VARCHAR(25)	NULL
Comment	VARCHAR(2000)	NULL

Diag IOC Config Maj Ver

IOC Config Type Id (FK)	VARCHAR(40)	NOT NULL
Major Ver Nbr	NUMERIC(3,0)	NOT NULL
Minor Ver Nbr	NUMERIC(3,0)	NOT NULL
Comment	VARCHAR(2000)	NULL
Modify Date	DATE	NOT NULL
Mod By Usr	VARCHAR(64)	NOT NULL

Diagnostics IOC Parameter

IOC Config Type Id (FK)	VARCHAR(40)	NOT NULL
Param Nm	VARCHAR(50)	NOT NULL
Param Type Id (FK)	VARCHAR(20)	NOT NULL

Parameter Datatype

Param Type Id	VARCHAR(20)	NOT NULL
Datatype Pkcsid	NUMERIC(2,0)	NULL

Diagnostics IOC Parameter Value

```

BEGIN
  SELECT config_file_nm, config_file_loc
  INTO v_config_file_nm, v_config_file_loc
  FROM epics.ioc_config_type
  WHERE ioc_config_type_id = (SELECT ioc_config_type_id
                              FROM epics.ioc_dvc
                              WHERE dvc_id = p_dvc_id);

  EXCEPTION
  WHEN NO_DATA_FOUND
  THEN
    RAISE;
END;

SELECT maj_ver_nbr, min_ver_nbr, ioc_config_type_id
INTO v_maj_ver_nbr, v_min_ver_nbr, v_ioc_config_t
FROM diag_ioc_act_config
WHERE dvc_id = p_dvc_id;
EXCEPTION
WHEN NO_DATA_FOUND
THEN
  ops@oracle.global_utils.v.errornum := SQLCODE;
  
```

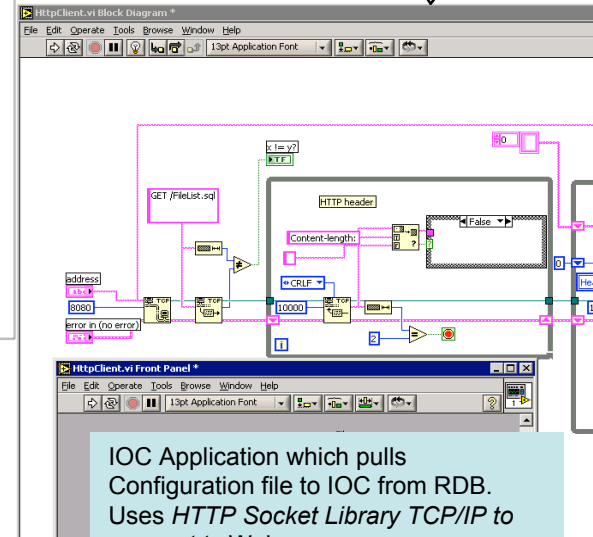
Portion of stored procedure used when IOC Configuration File generation UI creates a new configuration file for a given IOC. These selects confirm active versions which then dictate data to be used to create the file stored for use.

Device	VARCHAR(40)	NOT NULL
Device Id	VARCHAR(10)	NULL
System Id (FK)	VARCHAR(10)	NULL
Device Type Id (FK)	VARCHAR(25)	NULL
Device Inst	VARCHAR(25)	NULL
Parent Device Id (FK)	VARCHAR(40)	NULL
Device Desc	VARCHAR(255)	NULL
Bl Device Ind	CHAR(1)	NOT NULL
Virtual Device Ind	CHAR(1)	NOT NULL
Physical Ind	CHAR(1)	NOT NULL
Device Id Alias	VARCHAR(75)	NULL
Act Device Ind	CHAR(1)	NULL
MPS Device Ind	CHAR(1)	NOT NULL
Device Name	VARCHAR(80)	NULL
Device Id	VARCHAR(40)	NULL
Mod By Usr	VARCHAR(64)	NULL
Mod Date	DATE	NULL
Owner Cat Id	VARCHAR(10)	NULL

IOC Configuration File	VARCHAR(40)	NOT NULL
Config File Nm	VARCHAR(255)	NOT NULL
Config File Loc	VARCHAR(128)	NULL
Config File Nr	NUMERIC(9,0)	NULL
Contents	LONG VARCHAR	NULL
Modify Date	DATE	NOT NULL
Mod By Usr	VARCHAR(64)	NOT NULL
Editable Indicator	CHAR(1)	NOT NULL

IOC Config File Hist	VARCHAR(40)	NOT NULL
Config File Nm	VARCHAR(100)	NULL
Modify Date	DATE	NOT NULL
Config File Loc	VARCHAR(128)	NULL
Config File Cont	TEXT	NULL
Mod By Usr	VARCHAR(64)	NOT NULL

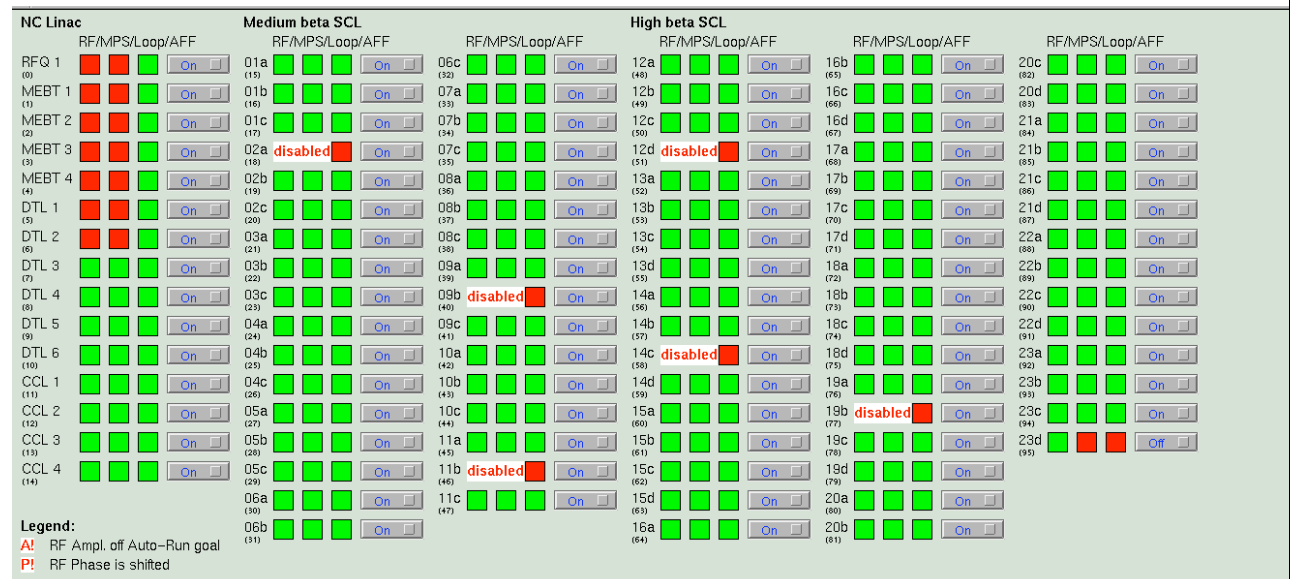
Files pulled from here.



IOC Application which pulls Configuration file to IOC from RDB. Uses HTTP Socket Library TCP/IP to connect to Web server. Web Server uses standard RDB connectivity: PHP, JSP, ASP

# Example: LLRF Multiplicity

- Almost 100 SNS LLRF Systems, handled by ~50 front-ends
  - As different as warm vs. super-conducting cavities
- One source base for all of them
  - Differences handled by configuration settings
  - If possible, startup files and overview displays script-generated from central system info.

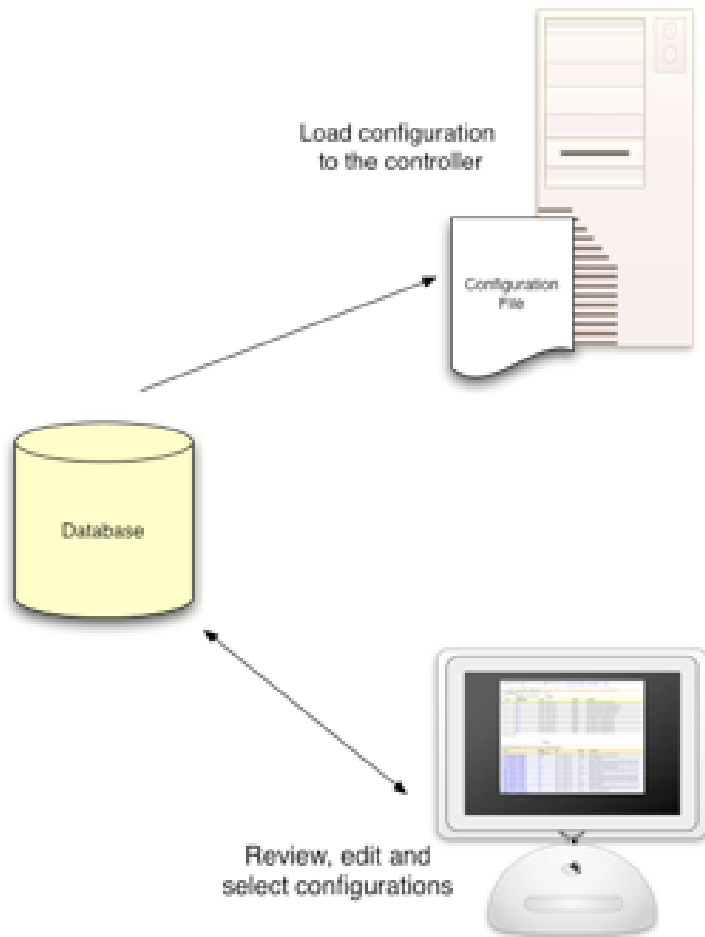


# Configuration File Strategy

- **Track changes to configuration files**
  - Who made the change
  - When was the change made
  - Why was the change made
- **Restore past configuration files when necessary**
- **Configuration consists of **structure** and **data****
  - **Structure (collection of properties that describe the device) is typically common across devices of a specific type**
  - **Data typically varies for each device and represents the values for a device's properties**
  - **Structure is associated with a configuration's major version number and data is associated with the minor version**



# Configuration File Storage/Retrieval



Configuration Type:  Device Overview | [Configuration Templates](#) | [Batch Import](#) | [Logout](#)

[Close](#) DTL\_Diag:IOC\_BCM622 *Warning: This is not the default device for your IP Address!*

25 Configurations Display  items Page  of 3

no file selected

Active	Configuration	Date	Author	Comment
<input checked="" type="radio"/>	<a href="#">17.22</a>	May 01, 2007 16:14	900688	Batch upload of configuration files.
<input type="radio"/>	<a href="#">17.21</a>	May 01, 2007 15:48	900688	Demo Batch upload of configuration files.
<input type="radio"/>	<a href="#">17.20</a>	May 01, 2007 11:36	900688	Test batch upload of configuration files.
<input type="radio"/>	<a href="#">17.19</a>	May 01, 2007 11:36	900688	Test batch upload of configuration files.
<input type="radio"/>	<a href="#">17.18</a>	May 01, 2007 10:46	900688	Batch upload of configuration files.
<input type="radio"/>	<a href="#">17.17</a>	May 01, 2007 10:44	900688	Batch upload of configuration files.
<input type="radio"/>	<a href="#">17.16</a>	May 01, 2007 10:43	900688	Batch upload of configuration files.
<input type="radio"/>	<a href="#">17.15</a>	May 01, 2007 10:42	900688	Batch upload of configuration files.
<input type="radio"/>	<a href="#">17.14</a>	May 01, 2007 10:41	900688	Batch upload of configuration files.
<input type="radio"/>	<a href="#">17.13</a>	May 01, 2007 10:40	900688	Batch upload of configuration files.

Activation notes:

BCM Diagnostic Device Active Configurations:

Device	Active Configuration	Date	Editor	Comment
<a href="#">CCL_Diag:IOC_BCM102</a>	<a href="#">14.22</a>	May 01, 2007 16:14	900688	Batch upload of configuration files.
<a href="#">DTL_Diag:IOC_BCM400</a>	<a href="#">11.1</a>	Nov 13, 2006 14:51	900688	Use the INI file.
<a href="#">DTL_Diag:IOC_BCM428</a>	<a href="#">2.1</a>	May 31, 2006 14:28	9PJ	Original Configuration taken during initial insert May 2006
<a href="#">DTL_Diag:IOC_BCM600</a>	<a href="#">2.1</a>	May 31, 2006 14:28	9PJ	Original Configuration taken during initial insert May 2006
<a href="#">DTL_Diag:IOC_BCM622</a>	<a href="#">17.22</a>	May 01, 2007 16:14	900688	Batch upload of configuration files.
<a href="#">EDmp_Diag:IOC_BCM02</a>	<a href="#">4.1</a>	May 31, 2006 14:28	9PJ	Original Configuration taken during initial insert May 2006
<a href="#">HEBT_Diag:IOC_BCM01</a>	<a href="#">12.1</a>	Nov 13, 2006 14:52	900688	Use the INI file.
<a href="#">HEBT_Diag:IOC_BCM09</a>	<a href="#">2.1</a>	May 31, 2006 14:28	9PJ	Original Configuration taken during initial insert May 2006
<a href="#">HEBT_Diag:IOC_BCM20</a>	<a href="#">2.1</a>	May 31, 2006 14:28	9PJ	Original Configuration taken during initial insert May 2006
<a href="#">HEBT_Diag:IOC_BCM32</a>	<a href="#">2.1</a>	May 31, 2006 14:28	9PJ	Original Configuration taken during initial insert May 2006
<a href="#">IDmp_Diag:IOC_BCM01</a>	<a href="#">11.1</a>	Nov 16, 2006 12:55	900688	Use the INI file.
<a href="#">LDmp_Diag:IOC_BCM05</a>	<a href="#">2.1</a>	May 31, 2006 14:28	9PJ	Original Configuration taken during initial insert May 2006
<a href="#">LEBT_Diag:IOC_BCM114</a>	<a href="#">11.1</a>	May 01, 2007 11:44	900688	Test new activation code.

# Implementation Choices

Flexibility  
Rapid development



Performance

- Human
- Commercial/Scripted High Level
- High Level Application (Multiuser OS)
- Low Level Application (RTOS target)
- Embedded (DSP, limited OS)
- FPGA
- ASIC
- Analog

“Some folk built like this, some folk built like that  
But the way I'm built, you shouldn't call me fat  
Because I'm built for comfort, I ain't built for speed...”

- Willie Dixon

# Flexibility or Performance?

- Flexibility in the form of
  - Rapid development, independent testing, remote access, online changes, rich set of debug tools



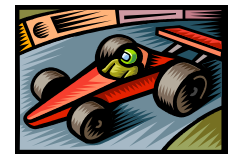
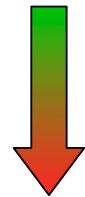
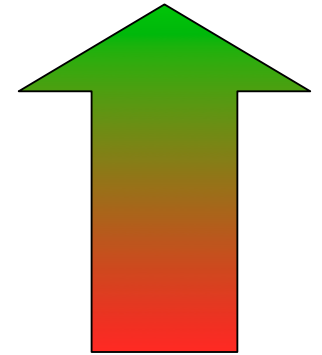
... often differs from ...

- Performance
  - Fast startup times, short response times, deterministic "real time" behavior.



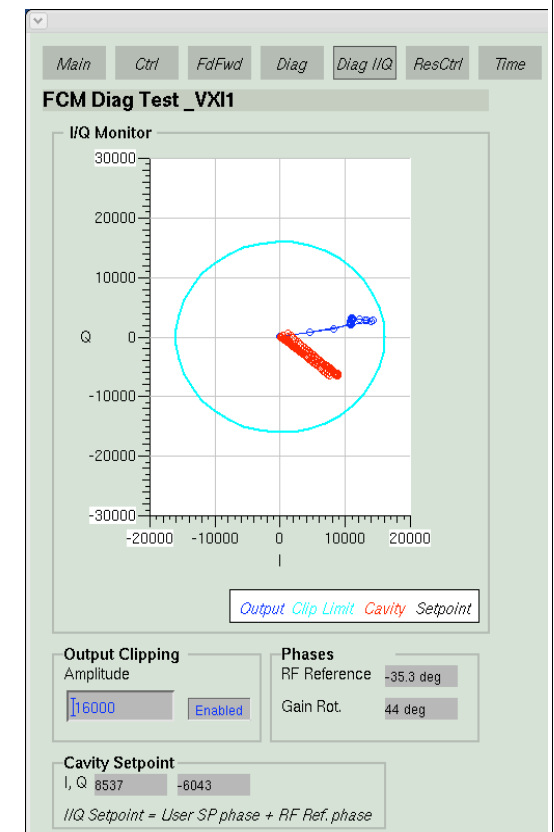
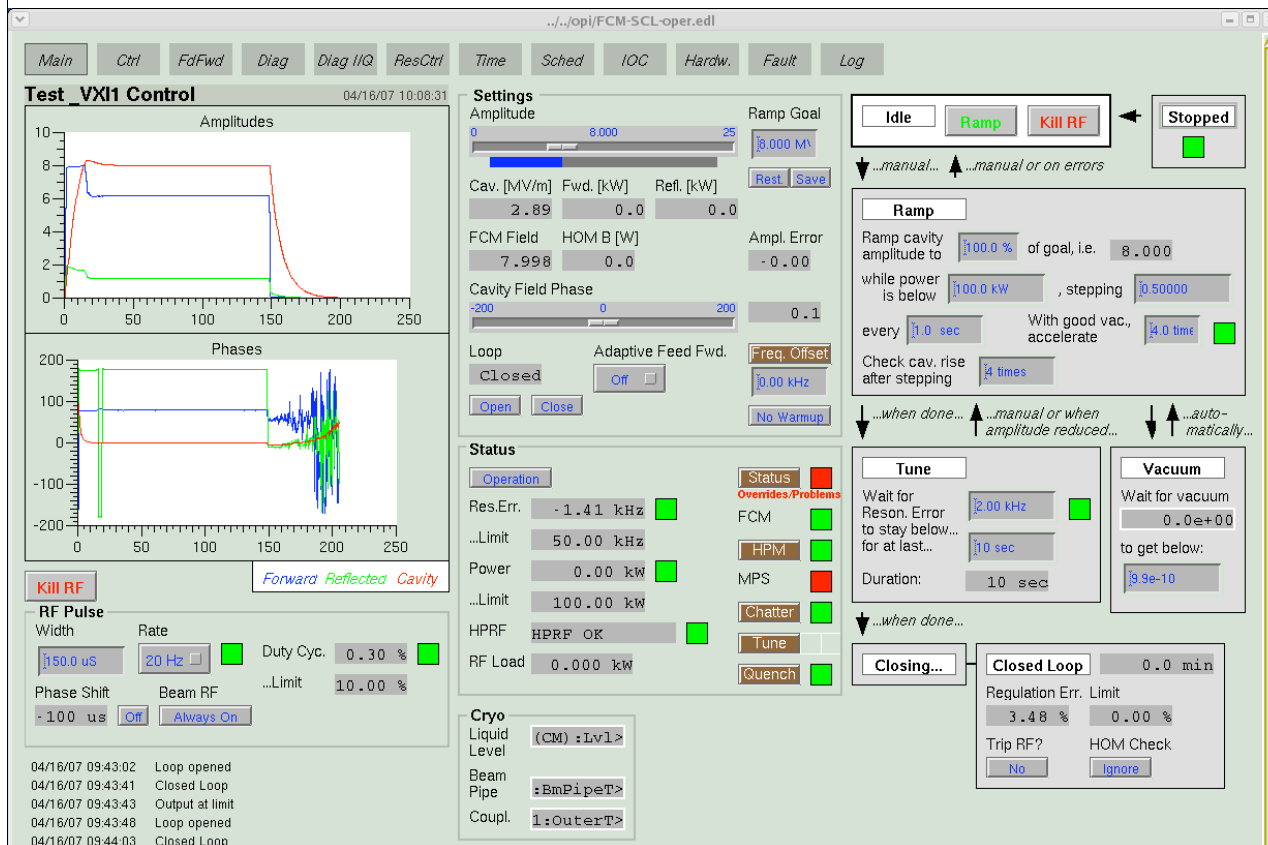
# SNS LLRF Choices

- Software based a control system **framework**  
(Experimental Physics and Industrial Control System, EPICS)
  - Matlab scripts for test & development of algorithms
  - Front-End computer uses EPICS **State Machine Tool** for automation, and "runtime **Database**" for data flow.
  - C/C++ driver code
- 
- **Fast** Feedback (~40MHz) and interlocks in **Verilog**, VHDL, AHDL. Several Iterations
  - Hardware as simple as possible: Analog filtering, ADCs/DACs, then **FPGA**



# Operator Interface: Display Manager

- Drawing package for
  - Placing labels, text-monitors, meters, ... on a screen
  - Connecting them to online Process Variables
  - Display panel **Configuration** instead of coding

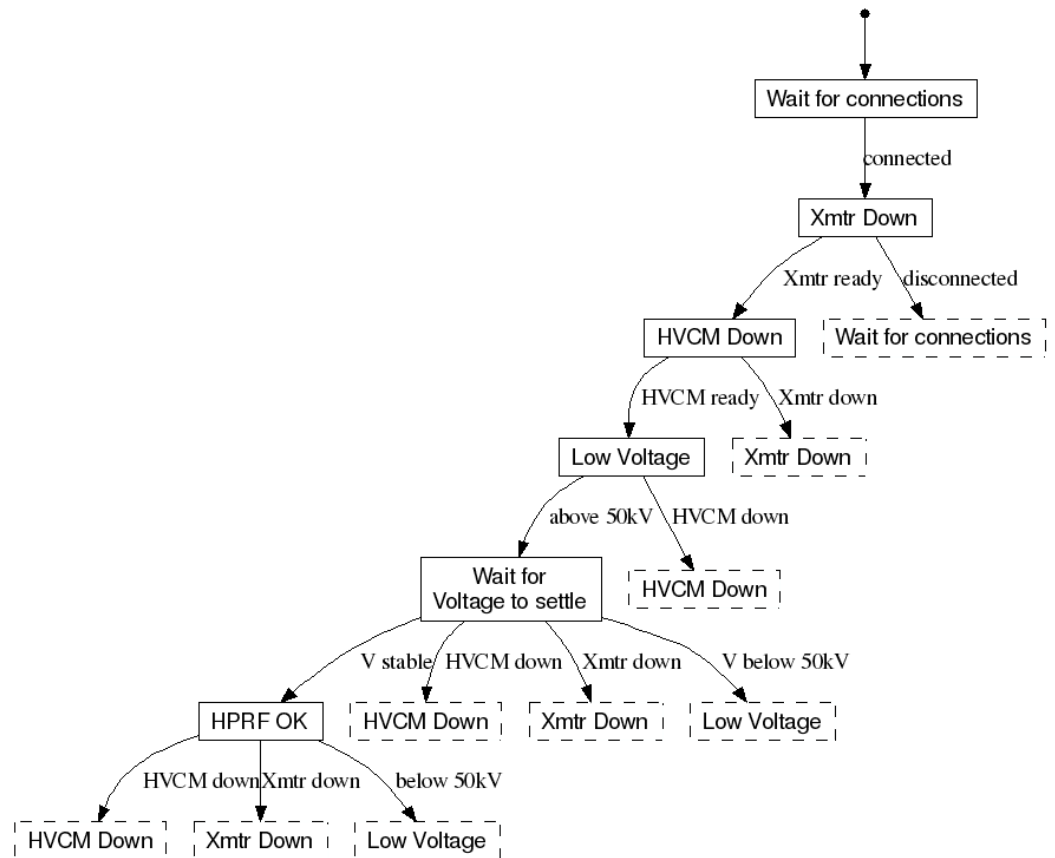


# Overall SNS LLRF

- Requirements change, so **Flexibility** is key.
- Resonance Error computation, feedback loop setup, ...:
  - If possible, first developed in Matlab
  - Then implemented on Front-End as State Machine or EPICS Database
  - If necessary, later moved into custom C++ driver code, or even FPGA

# State Machine (EPICS "Sequencer")

- Used for automation whenever possible
- "On Demand" tasks, response times of .1 to 1 sec
- Safest and most flexible tool
  - Runs on host as well as front-end
  - Start/Stop/Update without front-end reboot



# EPICS "Database"

- Used for steady-state control, data flow.
- **Full remote access** to any detail.
- Limited online changes.
- "Records", building blocks
  - Read input, computation, write output, ...
- Database Engine handles
  - Periodic or event-driven scanning
  - Time stamps(!)
  - Check of alarm limits
  - Publication of data in "Process Variables"
- Response times of millisecs possible, or more than 10000 records per front-end computer.

```
record(ai, "temp")
{
    SCAN "1 second"
    INP      "#C2"
    S3"
    EGU     "deg C"
    HIGH    "40.0"
    ...
}
```

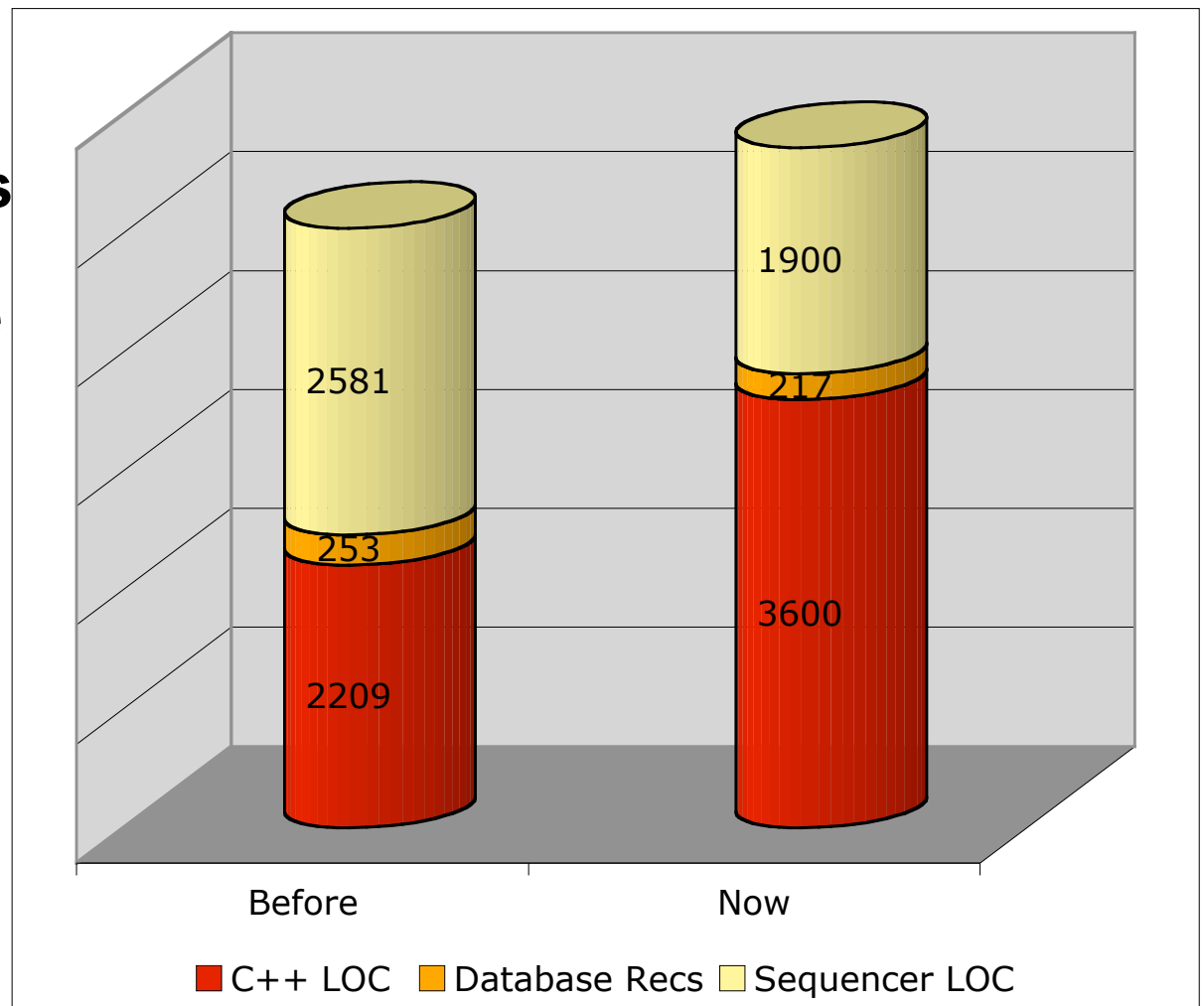


# Custom Low Level Code

- Custom C/C++
  - When required for higher **performance**, or interrupt service routines, low-level access to custom hardware
  - Usually **no** online changes.
  - It's really **hard** to understand, extend, debug somebody else's custom code
  - Debug tools vary with operating system
    - SNS, using vxWorks5 with Linux hosts has currently no online source-level debugger....
- **FPGA**
  - Same **problems** as custom C/C++ code
    - Probably even more so, since HDL is a "code", but often not implemented by software engineers.
  - Good simulation and offline analysis tools but online debugging limited to scope, signal analyzer.

# SNS LLRF Changes in early 2007

- Improve performance of tested algorithms by converting Sequencer (State Machine) code and Database Records to C++



# Alternative DSP Implementations

- High level environment
  - Commercial
  - Physics application framework
- Within control system toolkit
- Vertically integrated commercial products

# Matlab Scripting Example

## Orbit correction

```
% Get the vertical orbit
Y = getam('BPMY');

% Get the Vertical response matrix from the model
Ry = getrespmat('BPMY', 'VCM');    % 120x70 matrix

% Computes the SVD of the response matrix
lvec = 1:48;
[U, S, V] = svd(Ry, 0);

% Find the corrector changes use 48 singular values
DeltaAmps = -V(:,lvec) * S(lvec,lvec)^-1 * U(:,lvec)' * Y;

% Changes the corrector strengths
stepsp('VCM', DeltaAmps);
```

# LabVIEW FPGA with EPICS

