General-Purpose Data Recording

Mark Stanovich, James Langston, Michael Sloderbeck, Matthew Bosworth, Karl Schoder, Mischa Steurer

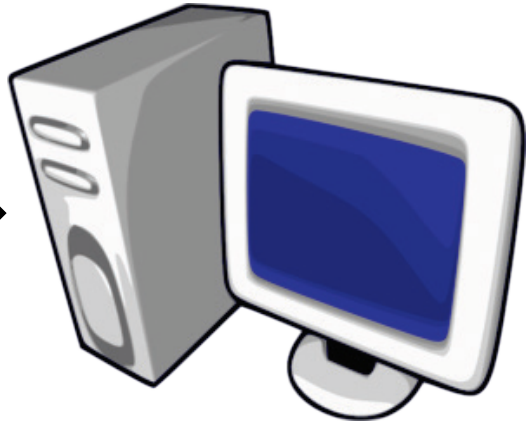
Center for Advanced Power Systems
Florida State University

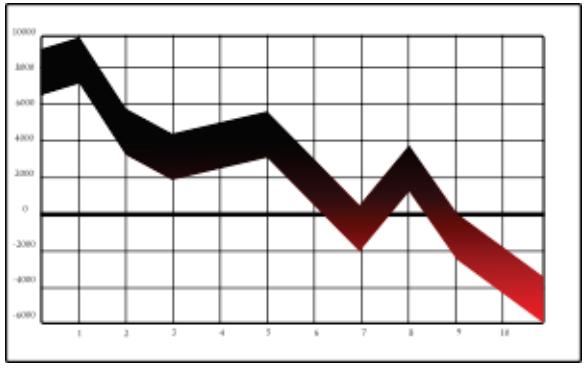
Partially supported through ONR under grant N000141410198 (ESRDC)

Overview

Simulated Power System

Data



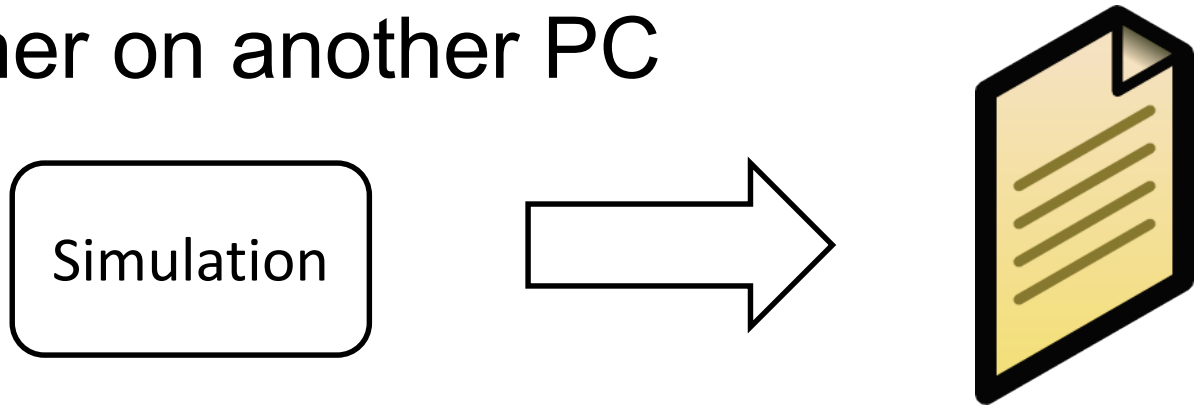


Data Recording at CAPS

- Model and Device Characterization
- Demonstration
- Fault recording
- Contexts
  - RTDS only simulation
  - Co-simulation
  - PHIL
  - CHIL

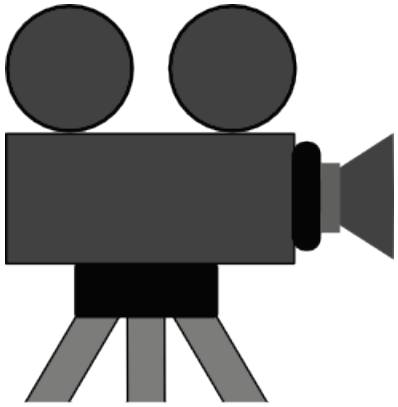
Implementations

- RSCAD plot captures
- RSCAD script reading and recording runtime meters
- External recording scopes (data acquisition system)
- GTNET stream data to server
- Listener on another PC



Desirable Functionality of Data Recorder

- Continuously record data over reasonably long time intervals (hours)
- Vary sampling rates (down to every time step)
  - per signal
  - Within single capture
- Efficiently analyze recorded data
  - Loading time for post processing analysis (e.g., Matlab)
  - Online analysis
- Trigger
  - Manual
  - Fault events
  - Pretrigger
  - Continuous

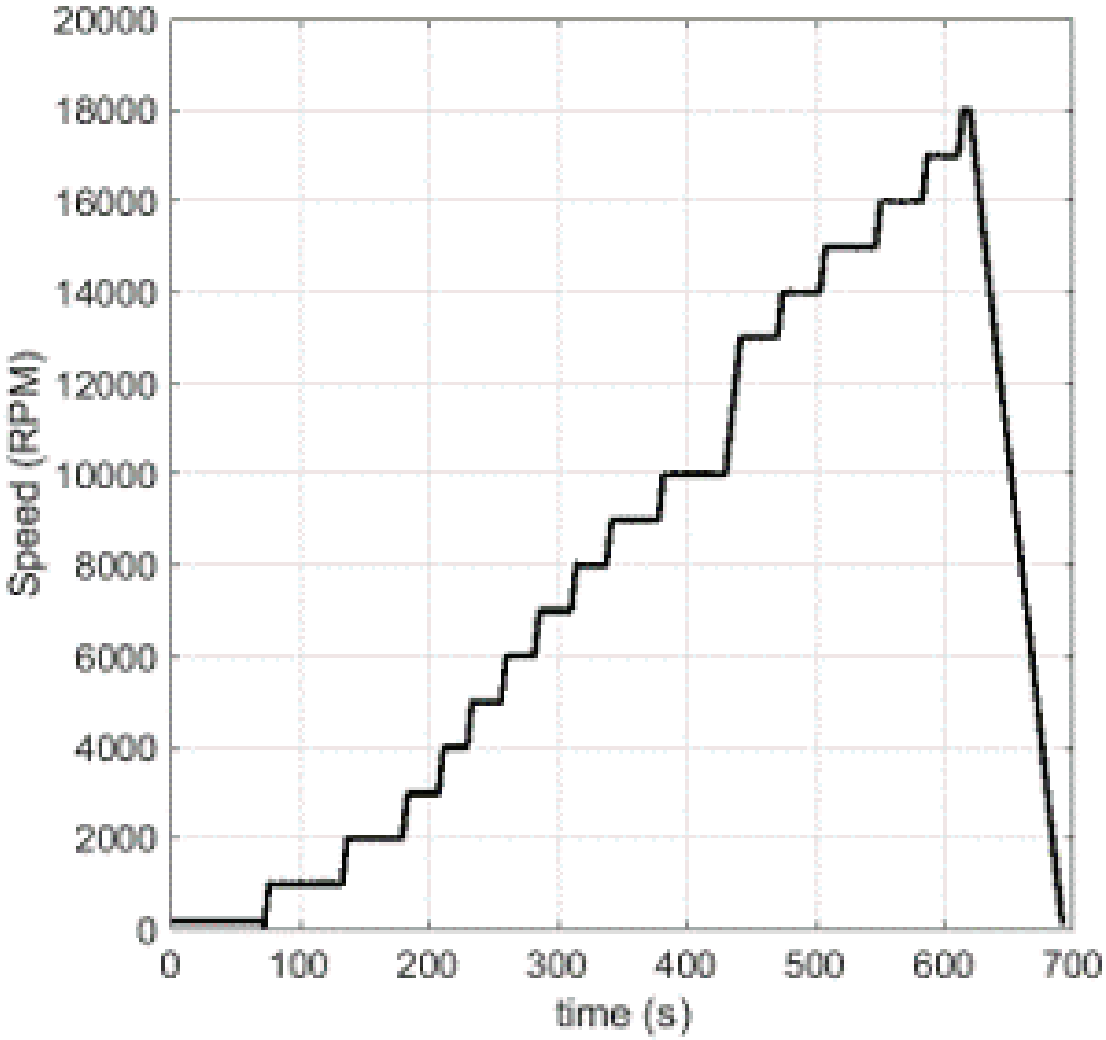


Desirable Functionality of Data Recorder (cont'd)

- Ability to interact with runtime while recording especially important during PHIL experiments
- Low network congestion
  - Computer executing the runtime has a slow/congested network connection
  - High data rates may cause congestion
- Vary capture rates
  - Per signal
  - Within single capture
- Commodity hardware and software

Example Use Case

5 MW Dynamometer and Gear Box



Length of time between pulses

GTFPGA
Pulse Counter

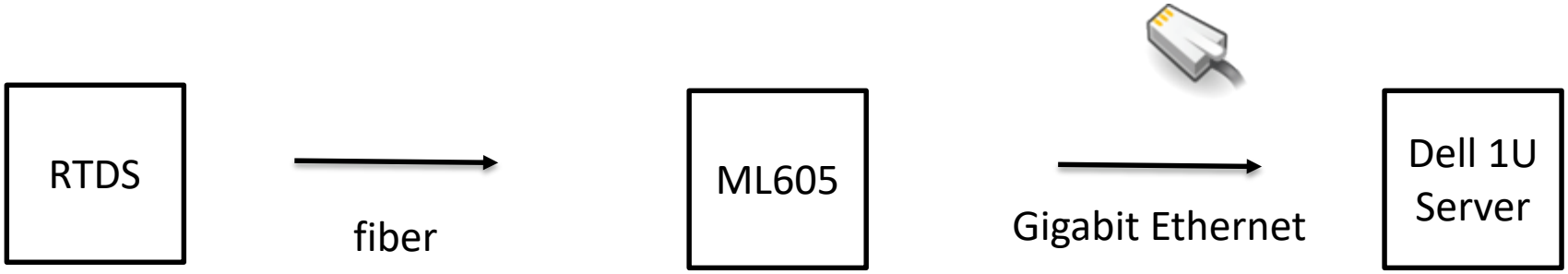
Signals

GTFPGA
Logger

LOGGER IMPLEMENTATION

Logger Implementation

- ML605 and GTFPGA
- Computer running Linux

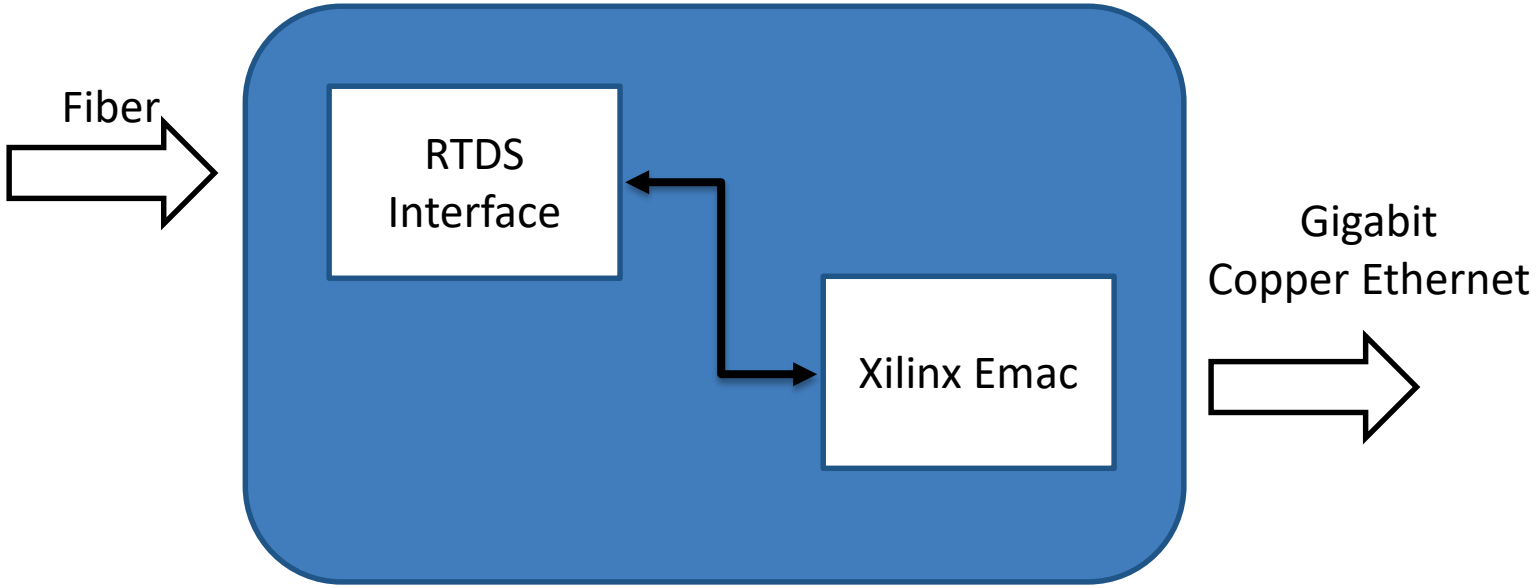


Software and Hardware Overview

- ML605 Xilinx FPGA development board
  - GTFPGA
  - Xilinx Emac coregen
  - VHDL
- Dell 1U rack server
  - Linux with PREEMPT_RT patches
  - Python/Cython, C++
  - SSH

Fiber to Ethernet

- RTDS protocol over fiber to *frame*-level Ethernet (1 gigabit)
  - Less protocol overhead
  - *Not* TCP/IP, so not routable but can still be switched
- ML605 - high-speed transceivers
- RTDS interface block
- Xilinx emac coregen



Communication Protocol Between FPGA and Server

- Logger initiates communication by sending data to FPGA
- GTFPGA responds with a frame containing the values from latest time step
- Send a frame every time step and process the received frame (~50 microseconds)
- Vanilla implementation has large variability (10s of microseconds to milliseconds)

Tasks

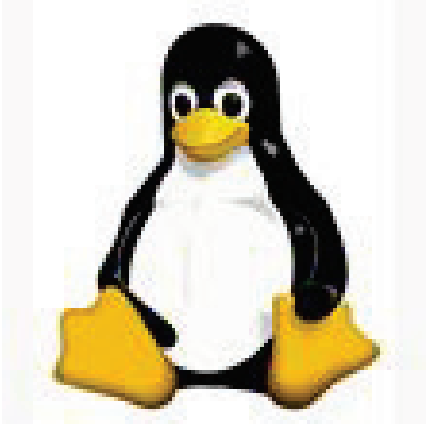
- Ethernet communication with short deadline
- Process received frames
  - Check if data should be logged
  - Check for missed time steps
  - Format data for storage
- Write data to disk
- Write messages to screen
- Server for user to retrieve recorded data

Meeting Deadlines

- Rewrite portions in C++
- Multiple threads of execution
  - C++ thread communicating with ML605
  - Python thread processing frames
    - Check if data should be logged
    - Send data to another thread to write to disk
  - C++ thread logging messages to console

Linux Configuration

- Vanilla Linux with PREEMPT_RT patches
  - Better preemptibility and thread prioritization
  - More deterministic time to execute an operation (at the expense of lower average-case performance)
- Isolate cores and pin threads to separate cores
  - Efficient cache usage
  - Minimize context switches
- Disable hyperthreaded (virtual) cores



Meeting Deadlines

- *Lockless* communicating between threads
  - Use processor instructions for synchronization rather than spinning or OS
  - Single reader, single writer queue
- Assign kernel thread priorities
  - I/O also needs processor time
  - File system
  - Network
- Don't go to sleep

Usage by User

- User creates *signal list file* specifying signals to be captured
- RSCAD script
  - Reads signal list file
  - Sets draft variables for GTFPGA configuration of user selected signals
  - Places a copy of the user's signal list file in shared location for use by the logger
- Logger generates a *csv file* with the recorded data, with signal names in header (facilitates associating data with the signal name when loaded)
- User controls starting and stopping recording of data through an RTDS signal (e.g. can be a runtime switch)
- *Config file* exists on shared file server
  - Sets sampling rate
  - Other parameters for future use

Data File Format

- Text format (csv file)
  - Human readable
  - Large files are unmanageable (10GB cannot be read by Matlab)
- Binary (Matlab)
  - Difficult to incrementally append
  - Much quicker loading time

Future

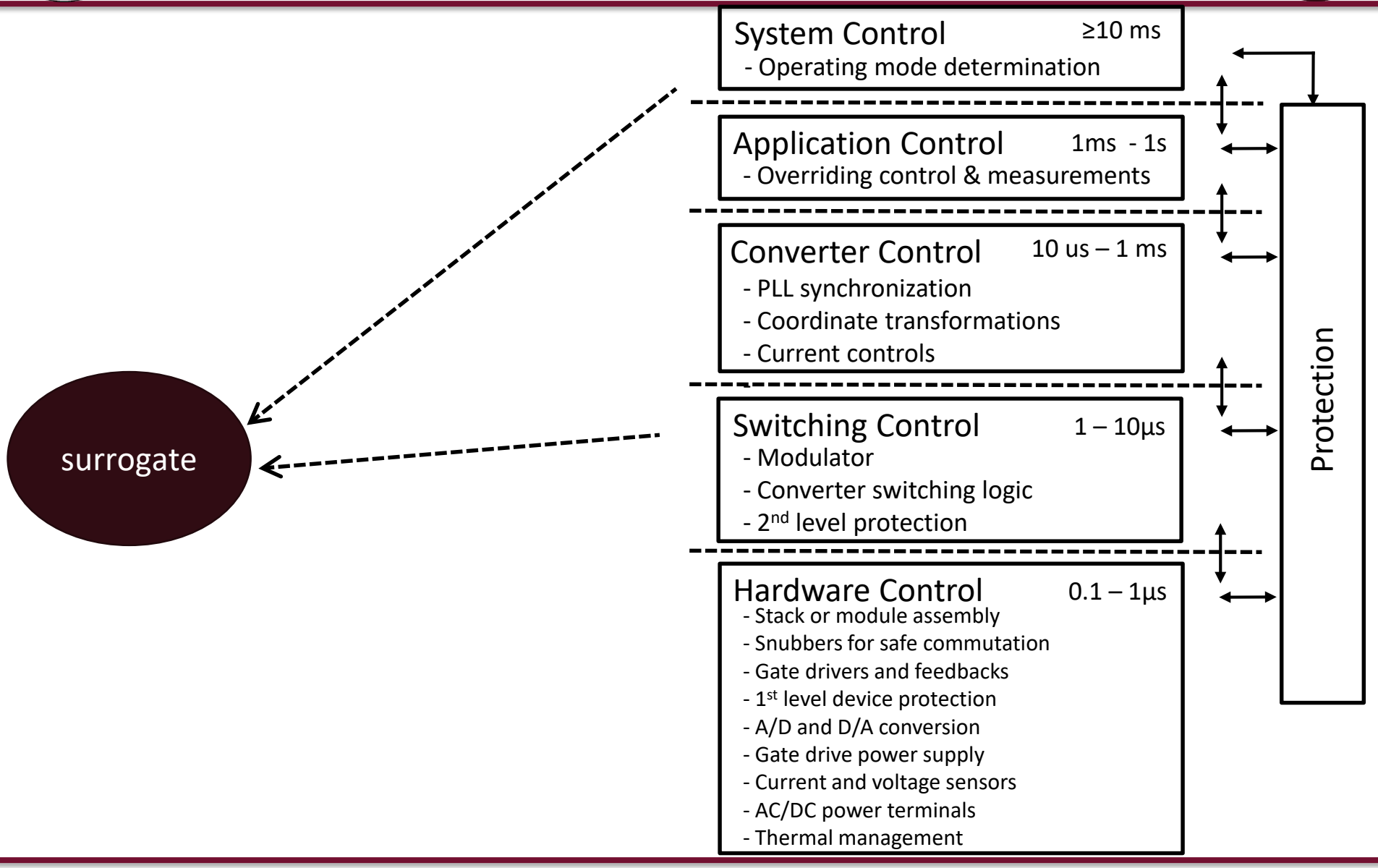
- Logging to handle concurrent RTDS racks and chassis
  - Need to update configuration mechanism
  - RTDS to single PC via switch
- Online analysis and visualization
- Modular solution to use with GTNET, etc.
- *Bi-directional support* between simulation and *surrogate* HW

Conclusions

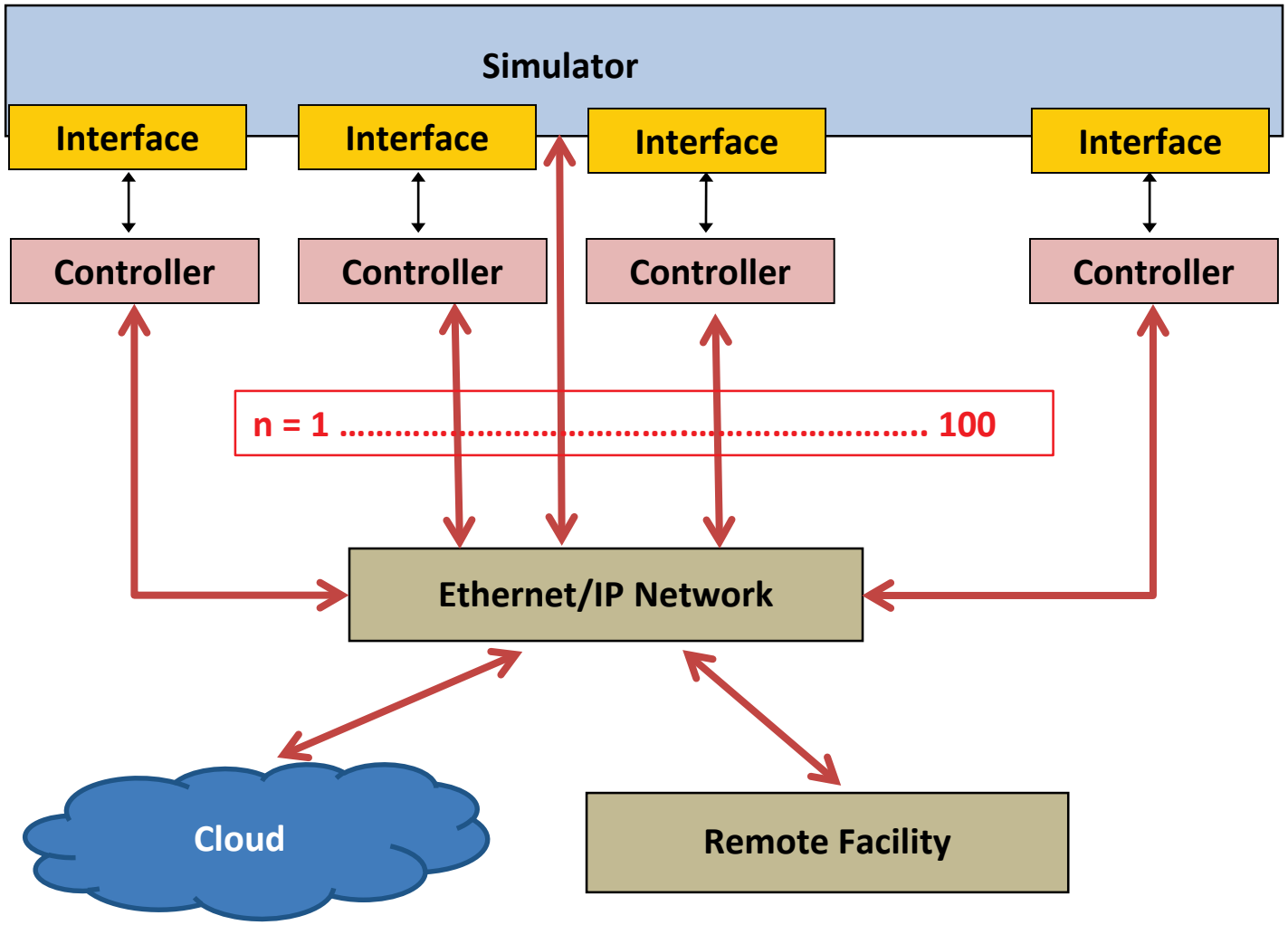
- Commodity HW/SW
  - Solution can more easily take advantage of HW upgrades
  - Most computers/devices have Ethernet ports
- Programming can be cumbersome
  - High-level languages (e.g., Python) can be inefficient for short deadlines
  - Limited tools and languages for targeting FPGA (e.g., Matlab/Simulink)
- Ethernet provides more flexibility than PCI-e implementation
  - Same/similar driver for each OS (and version)
  - Latency is sufficient and high throughput of PCIe not generally needed

QUESTIONS?

IEEE 1676-2010, Guide for Control Architecture for High Power Electronics
(1 MW and Greater) Used in Electric Power Transmission and Distribution Systems,
Recommended architecture for power electronics applications (Fig. 1)

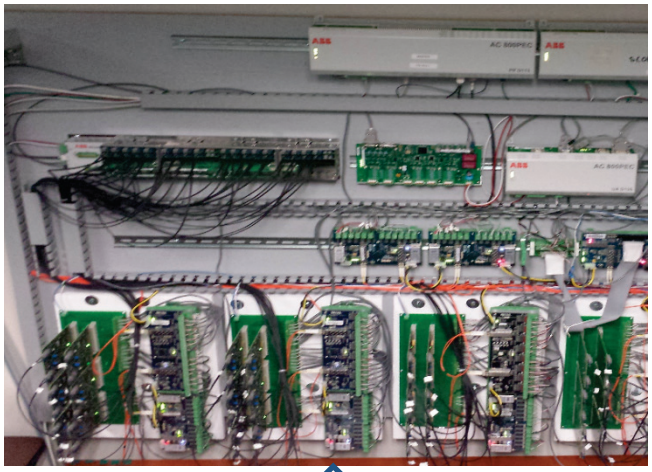


Large-Scale CHIL



Different Flavors of CHIL

Traditional



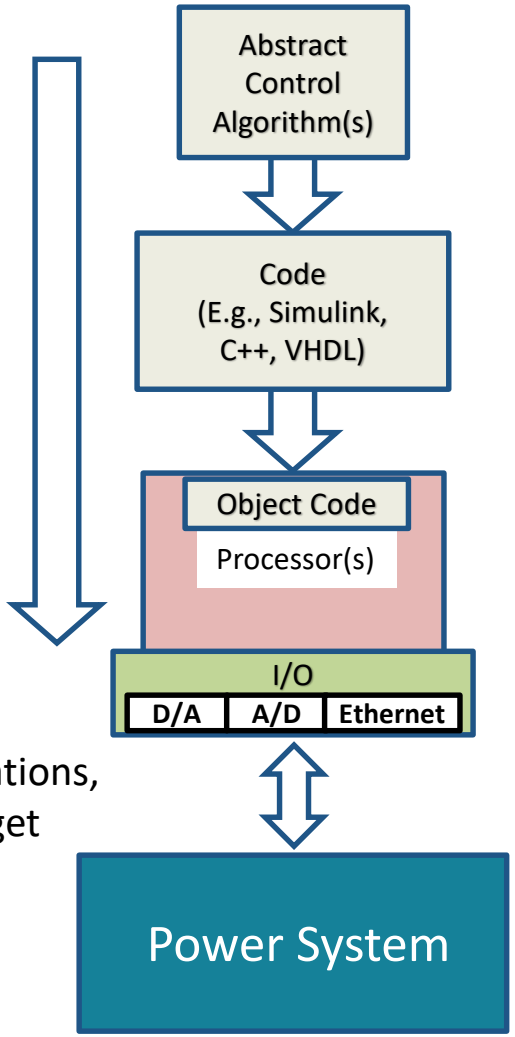
Power System

Focus: control algorithm on field deployed target platform

Surrogate

Representative host
More flexible, reuseable
Open platforms

Implementation and evaluation levels of interest



Focus: control algorithms implementations, distributed, networked, multi-use target platforms, large scale