Real Time PMU Assisted Wide Area Oscillation Damping using Compact Reconfigurable Controllers

May 05, 2014

ELDRICH REBELLO

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden 2014

September 2013 to April 2014

Real Time PMU Assisted Wide Area Oscillation Damping using Compact Reconfigurable Controllers

— Master Thesis —

Eldrich Rebello[1]

Electrical Power Systems Division
School of Electrical Engineering, KTH Royal Institute of Technology, Sweden

Supervisor and Examiner
Prof. Dr.-Ing. Luigi Vanfretti
KTH Stockholm

Supervisors
Mohammed Shoaib Almas &
Maxime Baudette
KTH Stockholm

Stockholm, May 05, 2014

[1] eldrich.rebello@aalto.fi

Nada me separará
do amore de Deus

AYRTON SENNA DA SILVA
Died 01.05.1994, Tamburello Corner, Imola,
Italy

Abstract

The modern power grid in increasingly being used under operating conditions of increasing stress, for which it was not designed. The increasing levels of variable sources such as wind and other other renewables also present stability problems. One of the these stability issues is the phenomenon of low frequency, electro-mechanically induced, inter-area oscillations. Research has been carried out into the area of wide area oscillation damping to study and damp these inter-area modes. This thesis implements oscillation damping by modulating the excitation system input of a FACTS device with a supplementary control signal.Phasor Measurement Units are becoming increasingly common and provide remote, real-time access to power system measurement data. This thesis takes an established Phasor based oscillation damping method and combines it with modern PMU measurements to produce a hardware prototype of a real-time oscillation damping control system using remote PMU signals sent over a communications network. The developed prototype is tested with various inputs to demonstrate the flexibility and advantages of using wide area measurements in power system control.

Preface

While I have already expressed my thanks, in person, to all the people involved directly or indirectly with this thesis, I think it appropriate to thank them here, once more, for posterity.

A special thanks to Dr. Luigi Vanfretti for giving a random email a chance and allowing the person behind it the opportunity to do his thesis at KTH, Stockholm. Mh. Shoaib Almas also deserves special thanks for putting up with and showing great patience in answering my persistent, irritating and sometimes meandering questions. Another round of thanks is in order for Mh. Shoaib Almas for showing me the ropes of the SmartTS lab which is admittedly the most complicated set of hardware that I have ever used. Maxime Baudette was the other target of some of my most frequent yet completely random questions. Going against the stereotype, he rarely went on strike but patiently answered them all. To him I say 'Merci'.

The National Instruments Support Engineers also deserve a mention for the inputs they provided.

The glorious invention of Larry Page & Sergey Brin has made many previously unfamiliar terms and pieces of information easy to find. Also, that repository of mankind's collective knowledge, Wikipedia, has served ever so often as a good starting point for several topics covered in this thesis.

I express sincere gratitude to the people concerned for three free meals, one concert and much other entertainment. You know that the gratitude of a hungry and poor student comes from the heart.

To Suomi & Sverige; Helsinki & Stockholm; Snow & Sauna: Kippis & Skål !

And last, but the most important, to my Parents and Fish, for support in all ways possible.

Östermalm & Otaniemi, 30.05.2014

Eldrich Rebello

Acknowledgements

This thesis has been carried out at the SmarTS Lab at KTH, Stockholm from 03.09.2013 to 30.04.2014.

This work was partly supported by the following institutions and funding bodies, their support is sincerely acknowledged:

1. The EIT KIC InnoEnergy “Smart Power" project under Action 2.6: PMU-Based Power System Operation Tools

2. The STRONgrid project - funded by Nordic Energy Research through the Sustainable Energy Systems 2050 research funding program

3. Statnett SF, the Norwegian Transmission System Operator and

4. The STandUP for Energy collaboration initiative

Contents

Symbols and abbreviations

Abbreviations

P	Active Power
Q	Reactive Power
PSAT	Power System Analysis Toolbox
VI	Virtual Instrument (LabVIEW ™Code)
AVR	Automatic Voltage Regulator
TG	Turbine Governor
cRIO	Compact Reconfigurable Input / Output
Q	Reactive Power
GUI	Graphic User Interface
DC	Direct Current
AC	Alternating Current
PSS	Phasor System Stabiliser
POD	Power Oscillation Damper
OSS	Open Source Software
PMU	Phasor Measurement Unit
PDC	Phasor Data Concentrator
TCSC	Thyristor Controlled Series Capacitor

1 Introduction

As modern power systems grow in size, both in terms of power transfer capacity and geographic spread, they are increasingly being used for purposes that the power system was not designed for. Examples of these 'new'uses include conditions of increasing stress such as power trading between countries. These interconnections, which link synchronous generators, often separated by vast physical distances, create conditions where small disturbances can excite oscillations that may or may not settle. When the generators of one area oscillate at a low frequency (typically 0.2–2.5Hz) against the generators of another interconnected, but distinct area, 'inter-area'oscillations may start. These oscillations have been known about since the earliest days of interconnected, synchronous power systems [1] and the solutions have ranged from simple control parameter tuning to the worst-case scenario where the interconnection fails altogether. A famous example of system collapse due to undamped inter-area oscillations was the August 1996 blackout of the WSCC system in the USA [2].

Although the purpose of system interconnection was to increase stability, the present situation of the power system incorporates renewable energy sources and power trading corridors, both of which impact system stability. More modern solutions to the problems of inter area and intra oscillations use Power System Stabilizers (PSS) [3]. While a PSS provides excellent damping to intra area modes with good local observability, its performance with intra-area modes may not be satisfactory [5]. A solution to this problem is to use already installed FACTS devices as stabilizers [4]. This can be achieved by an additional control input to the FACTS device which is only activated when damping behaviour is required. This allows the device to perform its designed function during periods of normal operation while providing damping support under stressed operating conditions.

The device needed to generate this additional control signal will need inputs from several locations in the power network. These inputs can be wired, local analogue signals or can be signals sent over a communications network from several, remote locations. The latter provides more data but is dependent on the communication network for its operation. A Phasor Measurement Unit (PMU) that measures different power system data and transmits this data over a communication network is a presently available solution for acquiring remote-location power system measurements in real-time.

Goals

This thesis proposes to develop a Wide Area Power Oscillation Damper (WAPOD)[1] prototype using commercially available micro-controller hardware, based entirely on PMU measurements received over a TCP/IP network. A previously developed damping algorithm (Phasor Estimation [6]) will be implemented on a general purpose micro-controller and will be run in real –time. The inputs to the controller will come from one or multiple PMU's, each monitoring data at different points in the power system. The power system model used in this thesis is the two-area four machine model, originally proposed by Klein, Rogers and Kundur [10]. To prove the real-world applicability of the developed controller, all tests are carried out in real-time, with conditions such as noise and network transport delay present.

The power system model will be run in real-time, with a 50μs time step. Amplified, analogue measurements will be sent to the PMU's which will generate a IEEE C37.118 compliant data stream. The analogue measurement will then be extracted from these data streams. Once raw phasor data is available (currents, voltages and their respective angles), different variables will be computed. The desired parameter will then be streamed to the controller, over the network, to generate an analogue damping signal. This damping signal will be reinserted in the network model running in a real-time target.

Note

The term "real-time" refers to a process and control system where data is processed and made available as feedback within a deterministic time-frame. The meaning of real-time as used in this thesis is similar to that used in the field of embedded systems. Strict deadlines are enforced and the reaction time of the system from an event to system response is guaranteed and fixed. If the time constraints (or deadlines) are not met, the controller is deemed to have failed. Network simulations performed here are done in real-time. Though an actual power network was not used, the clock on the simulated network runs as fast as a real clock. The controller designed is thus able to provide feedback control to the power network so as to affect the network at that time.

Thesis Outline

Chapter 1 deals with the basic ideas used throughout this thesis viz. synchrophasors, PMU networks, the SmarTS Lab setup and the National Instruments cRIO platform.

[1] Historically, damping stabilizers have been termed WAPOD where the P represents a measurement of active power through the line. Active power here would be used as a controller input signal. Although this term is not accurate when other quantities are used as control inputs or feedback signals, the term is used here to maintain consistency with existing literature.

Chapter 2 examines the origin of power system oscillations, oscillation damping using FACTS devices and introduced the two area test network, the model used throughout this thesis as a test network. The process of readying the model for real-time simulation and the problems faced and their solutions are also covered.

Chapter 3 Covers the software architecture of the controller and the code development in LabVIEW. Various problems faced are documented along with their solutions.

Chapter 4 presents the entire Hardware-In-the-Loop (HIL) test setup together with the various tests conducted to verify the working of the controller. The controller developed was tested with various input signals derived from PMU measurements and the results are analysed. An analysis of the signal propagation delay is also presented in brief.

Finally, conclusions are drawn in **Chapter 5** together with a short listing of possible future work.

Note

Since this thesis uses several pieces of software & hardware concurrently (SIMULINK ®[28], LABVIEW™, Compact RIO ™hardware, OPAL RT's real-time targets and simulation software[29], PMU Unwrapping Software etc) the details of all of these are not covered. Also, the language and style used are not unduly complex or flowery when not required. One aim of this thesis to to communicate the central ideas concisely and as simply as possible. Specific technical terms are used and explained where necessary.

The appendices are not merely extra content but include material essential to this thesis. They have been separated into appendices merely to maintain continuity in the main text and also to document all issues faced at various stages of this thesis.

1.1 Introduction to Phasors & Synchrophasors

A sinusoidal, periodic waveform can be be expressed in the time domain as

$$x(t) = X_m cos(\omega t + \phi) \tag{1}$$

Here, X_m is the peak value of the sinusoid, ω its angular speed and ϕ it's phase angle relative to a fixed reference. The phasor representation (polar) at the angular frequency ω will be [16]:

$$X = \frac{X_m}{\sqrt{2}} e^{j\phi} \tag{2}$$

The phasor representation captures the r.m.s value of the sinusoid and it's phase angle relative to a reference.

Graphically, Figure 1 shows a comparison between the time domain and phasor representations of the same sinusoidal quantities.

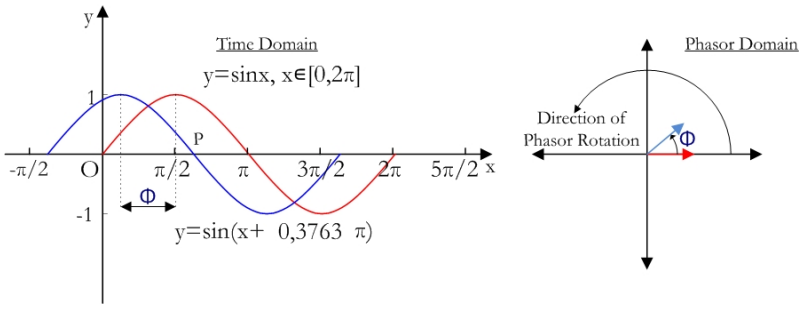

Figure 1: (L) Time Domain Sinusoid Representation & (R) Equivalent Phasor

Synchrophasors

The IEEE Standard C37.118.1-2011 defines a synchrophasor as "the value X in Equation 2 where the instantaneous value of the phase angle ϕ is calculated relative to a cosine function, at the nominal system frequency and synchronised to a timing source such as UTC" [16]. Three phase quantities can be represented by a phasor for each of the phases with a magnitude and phase angle. The phase angle of each phase will be calculated relative to the same, fixed reference and will hence change continuously in time. In a balanced, three-phase system, the relative angles between the three phase quantities will remain constant at 120 electrical degrees. It is a common convention to present synchrophasor angle data as a number between -180 degrees and +180 degrees however representation between 0 and 360 degrees is also possible [16]. The phase angle measurements wrap around at the ends of these limits. For the calculation & comparison of the phase angle of different quantities, a precise and reliable value of time, such as from a GPS timing source is a necessity [16]. The synchronised nature of these measurements allows for real-time network information to be extracted.

1.2 Introduction to PMU's & Synchrophasor Data

IEEE Standard C37.118.2-2011 defines a Phasor Measurement Unit as "a device that produces synchronized phasor, frequency, and rate of change of frequency (ROCOF) estimates from voltage and/or current signals and a time synchronizing signal." [17]

In other words, a Phasor Measurement Unit (PMU) measures and records instantaneous, time stamped values of current and voltage phasors. GPS signals are used to time-stamp measurements on each device and to synchronize measurements between different PMU devices. Using the GPS time reference, a PMU can maintain an accuracy of 1μs which corresponds to an angular accuracy of 0.018 degrees for a 50Hz system [18]. In essence, a PMU samples input waveforms, estimates the synchrophasor equivalents, formats this data in frames according to a synchrophasor standard such as IEEE C37.118 and then transmits this formatted data over a communications network [19]. A simple PMU is shown in Figure 2. Several PMU measurements can be collected, time aligned and retransmitted by a computer system called a Phasor Data Concentrator (PDC).

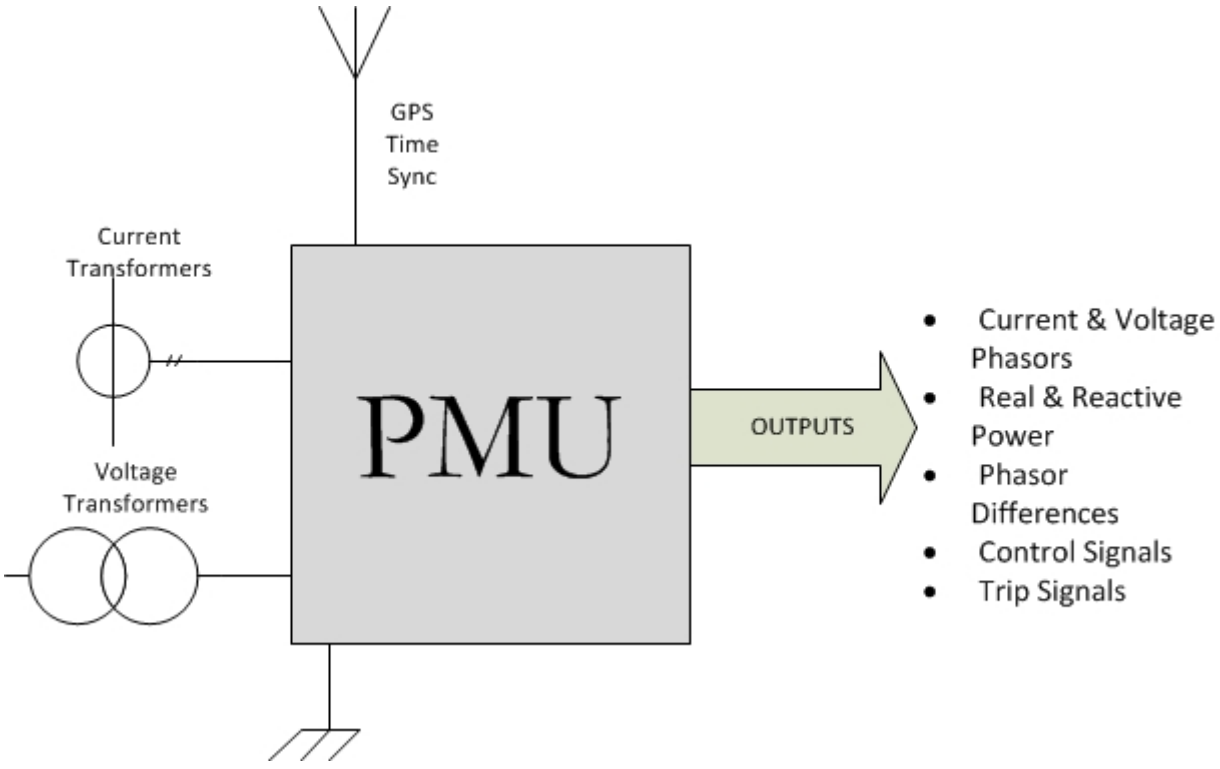

Figure 2: A Simple PMU

A PMU may be known by another name or, as is increasingly becoming common, be part of another device such as a protection relay. The PMU may also perform other functions such as monitoring the status of a circuit breaker[17].

The IEEE C37.118 standard (Parts 1 and 2) for Power System Synchrophasor

Data is widely adopted and several device manufacturers provide equipment conforming to this standard. This section provides a *very* brief overview of the data frames defined in this standard. For a more detailed & complete description, the reader is referred to [17]

1. **Data Frame** : Transmitted by PMU/PDC. Contains real-time measurement data in binary
2. **Configuration Frame** : Transmitted by PMU/PDC. Machine readable, binary data with data stream processing parameters. Contents have been expanded with successive standards
3. **Header Frame** : Transmitted by PMU/PDC. Contains information about scaling, algorithms, filtering etc. ASCII formatted, human readable.
4. **Command Frame** : Always received by the PMU/PDC. Contains commands for actions such as turning on or off transmission, requesting for configuration frames etc.

It is worth noting that the C37.118.1 standard does not specify either hardware or software requirements for a PMU. There are no constraints on whether a PMU is to be a stand-alone device or a part of another device such as a protection relay. There are also no suggestions or recommendations for algorithms to be used to compute phasors or other parameters such as the rate of change of frequency (ROCOF). Also, the standard merely specifies a data format for synchrophasor data and not a transport mechanism. The fact that all the synchrophasor data exchange in this thesis is done using TCP/IP does not mean that this the only method to transmit this data. In fact, efforts are ongoing to extend the IEC61850 GOOSE protocol to be able to transmit C37.118 formatted synchrophasor data (see IEC61850-90-5).

1.3 Synchrophasor Infrastructure

A PMU performs measurements and generates a synchrophasor data stream. The data from this stream, while useful on its own, can be put to better use when analysed with data from another PMU [20]. The simplest such 'synchrophasor network' will consist of one PMU streaming data to a Phasor Data Concentrator (PDC) [17]. This is a device that assembles data streams coming from one or more PMU's, performs a function on the data (archiving, conditioning, analysis) and streams the data to another location or application. A PDC also aligns different data streams using the included GPS timestamps. This is an important function as the network delay depends on the geographical distance as well as other factors[20]. Using this analogy in a typical power network, the PMU will be located at the substation level, measuring network data at different locations in the network and the PDC will be located at the control or monitoring centre [20]. A typical PMU can produce multiple streams allowing for one PMU to stream to multiple, redundant PDC's. At a central

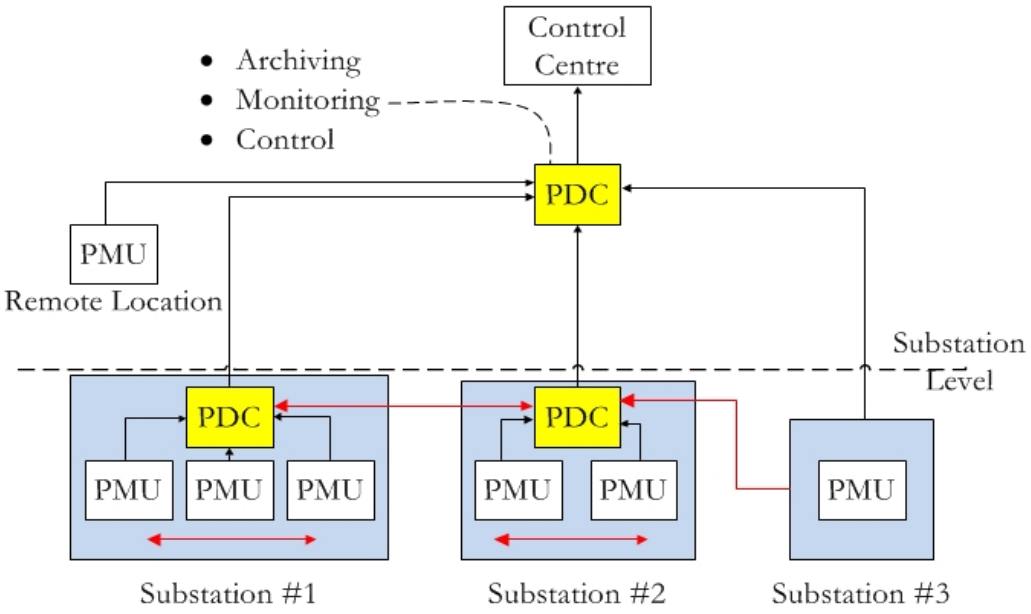

Figure 3: Example Synchrophasor Based Data Network

control level, streams from several PDC's can be assembled. The entire network can me made more reliable by exploiting the ability of PMU's to stream data to other PDC's thus allowing the data path to be changed if certain communication channels fail. An example of this type of synchrophasor data network is shown in Figure 3

PMU and PDC streams can also contain subsets of data or data streams at slower rates for certain applications. Functions such as archiving can be fed with streams at the full data rate while monitoring applications can receive a stream at a slower data rate [20].

A synchrophasor-based monitoring and control system can be constructed using multiple PMU's. The other components of such a system would be a Phasor Data Concentrator (PDC) to concentrate these PMU measurements, the monitoring or control tool and the communications network itself. Besides monitoring and control, functions typically performed by local devices, such as protection relays, can also be integrated into such a system. Such a system is not restricted by the physical locations of measuring or controlling devices and is called a Wide Area Monitoring, Protection and Control System (WAMPAC) system [20].

A communication network can be built to exchange synchrophasor data between devices (mainly PDC's) and also to a central processing unit (which triggers control action or can record data). This introduces a level of redundancy. If the communication means between a particular device and the central hub becomes damaged or is broken, its data can be rerouted through other devices. This provides more flexibility when compared to older communication standards such as SCADA where devices communicate individually to a central device. This redundancy also serves

as a data validator. If a state estimator (switch status open or closed) is used, intelligent methods can also be implemented to detect and correct data that is in error. This is all possible using synchrophasor-based state estimation.

1.4 Introduction to SmarTS Lab

The SmarTS Lab at KTH was set up with the aim of developing wide area monitoring, protection and control (WAMPAC) schemes for the power grid. Much of the infrastructure and activities involve PMU data and the associated communication and computer systems [21]. The lab is equipped with facilities for real-time (RT) simulations and also RT Hardware-in-the-loop (HIL) tests. A reduced schematic is shown in Figure 4

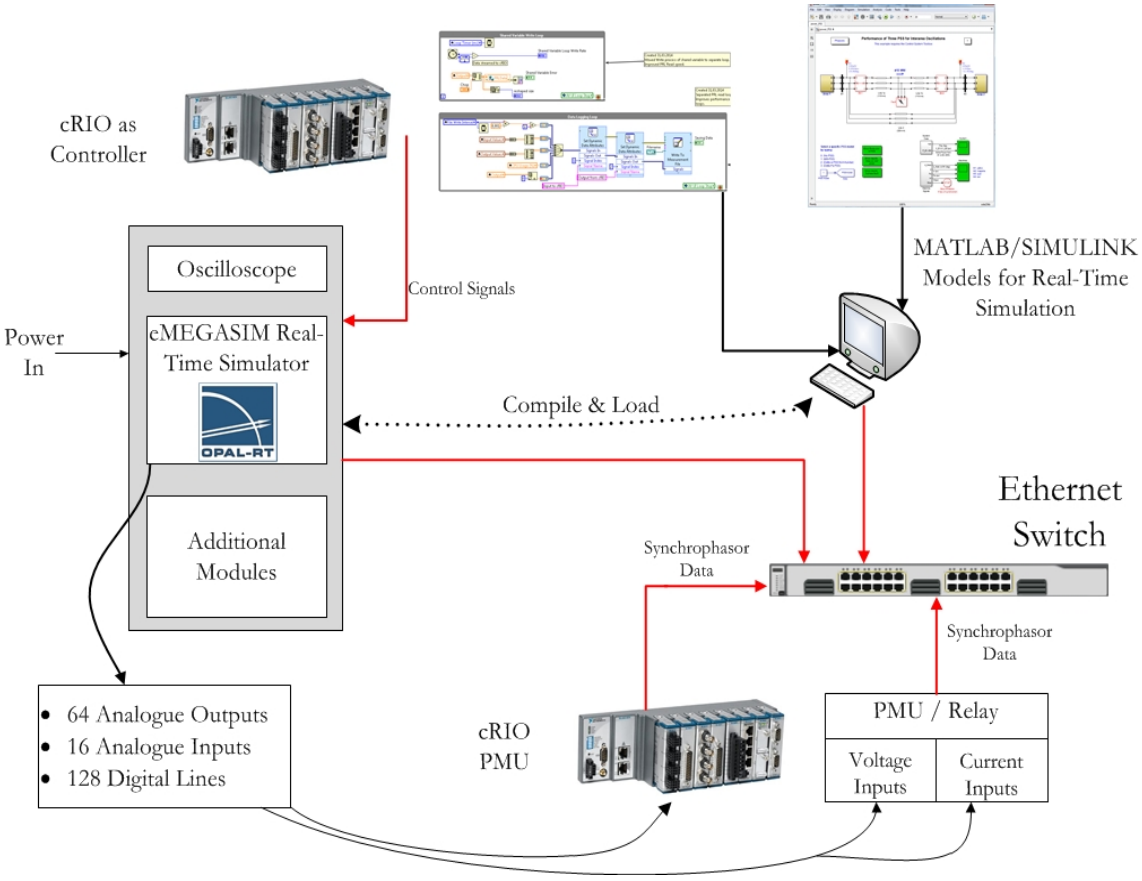

Figure 4: Outline of SmarTS Lab at KTH

The core of the setup is the eMEGASIM Real-time simulator from OPAL RT [29]. Two 'targets' are available, each running a 12-core 3.3Ghz Intel i7 processor [21]. This allows the running of models created in MATLAB/Simulink in real-time. These simulations can interact with external devices through the simulator's low-power analogue outputs and inputs or with data streamed over TCP/IP, UDP etc

[21]. In this thesis, the power system under test is simulated on this platform.

The analogue outputs and their ratings are listed below:

- **Analogue Outputs** : 32 (+/-16V and +/-10mA)
- **Analogue Inputs** : 128 (+/-100V and +/-10mA)

The full listing of the simulator's capabilities and interfaces is covered in [21]. Only necessary and relevant details are covered here.

The WAMPAC platform includes a Phasor Data Concentrator (PDC) and its associated software from Schweitzer Engineering Laboratories (SEL). Other devices are interfaced with the PDC such as protection relays with embedded PMU functionality, line differential protection relays (ABB), Compact RIO micro-controllers (National Instruments) and analogue signal amplifiers (Megger)[21]. The hardware list here is incomplete and other devices are also used such as a GPS receiver, a relay current and voltage injection kit etc.

1.5 Introduction to LabVIEW & NI cRIO Platform

LabVIEW is an Object Oriented, graphical programming language [33], developed and maintained by National Instruments [30]. Compared to conventional text based languages like C or Java, LabVIEW uses graphical programming where icons and wires are used to represent sections of code and data respectively [33]. LabVIEW is used in a variety of applications requiring hardware interfaces such as instrument and control systems, data acquisition etc. Numerous add-on modules exist which expand the functionality of the base system to other areas [30]. This thesis uses the Real-Time Module and the LabVIEW FPGA Modules. LabVIEW programs use the 'data-flow' programming concept [33], similar to that used in Simulink where the execution sequence is determined by when all inputs to a block of code become available [32]. LabVIEW programs are referred to as VI's (Virtual Instruments) where each VI has a 'Front Panel' which serves as the user's interface point, allowing the input of data, display of data and monitoring [32], [30].

The Compact Reconfigurable Input Output (cRIO) platform from National Instruments is a reconfigurable, deterministic embedded controller [31]. It features a user programmable, real time controller along with an embedded Xilinx FPGA controller. The chassis features standardized slots to which different modules can be connected depending on the functionality desired [31]. Modules such as analogue and digital voltage outputs, analogue current inputs, GPS receivers, accelerometers, relays etc are readily available. Depending on the model selected, the chassis can accommodate from 4 to 16 expansion modules. On both models of the cRIO used in this thesis, the FPGA runs at 400MHz allowing for a time step as small as 25ns. This means that the outputs of the device can be updated as fast as every 25ns.

The essential characteristic of real-time systems is that they are capable of responding to external events in real-time. This means that they can analyse and respond to an external event while it in progress. The term 'jitter' is used to describe how much a given section of code deviates from its allowed execution time. Typically, jitter is low for a real-time device [33] as the time taken for code to execute is critical. Commercial, desktop computer operating systems such as Windows do not execute code in real-time as the code execution time is dependent on several external factors and can vary.

The NI cRIO platform also includes an embedded FPGA controller. An FPGA has a large number of unconnected logic gates that can be reconfigured (reconnected) depending on the functionality desired [33]. Code execution on an FPGA is truly parallel as different sections of code execute on different hardware locations. Additionally, all the FPGA hardware is deterministic meaning that code execution speed remains constant. This said, an FPGA is not suited for all types of computation tasks but is more useful for repetitive tasks that would otherwise be processor-intensive [33]. The LabVIEW FPGA module provides a subset of the full set of LabVIEW tools that are tailor made for execution on the limited resources of an FPGA [33]. LabVIEW allows for rapid development of FPGA applications as the user does not require knowledge of low-level languages such as VHDL.

A total of three cRIO's were used in this thesis, two function as Phasor Measurement Units and one on which the damping control system was implemented. Each cRIO chassis also had modules connected for the input and output interfaces. The hardware details are presented in brief below. For a full listing see the references after each description.

1. **cRIO to Run POD Algorithm**
    - cRIO (Compact Reconfigurable Input Output) Controller 9081[2]
    - 1.06 GHz dual-core Intel Celeron processor (RT)
    - Spartan-6 LX45 FPGA
    - 25ns FPGA execution rate (fastest possible)
    - **NI 9264 Analogue Voltage Output Module**
        - 16 channel analogue voltage out, 10mV accuracy
2. **cRIO to Run NI PMU Software**
    - cRIO (Compact Reconfigurable Input Output) Controller 9076[3]
    - 400MHz RT CPU
    - Spartan-6 LX45 FPGA

[2] http://sine.ni.com/nips/cds/view/p/lang/en/nid/210000
[3] http://sine.ni.com/nips/cds/view/p/lang/en/nid/209758

- 25ns FPGA execution rate (fastest possible)
- **NI 9225 4 Channel Analogue Voltage Input**
  - 50 kS/s/ch simultaneous inputs
  - Built-in anti alias filters
  - 300 Vrms measurement range
- **NI 9227 4 Channel Analogue Current Input**
  - 5 Arms measurement (14 A peak)
  - 50 kS/s/ch simultaneous inputs
  - Built-in anti alias filters

For detailed specifications and operating instructions refer to the operating manuals, linked in [35], for the cRIO9076 and in [34] for the cRIO9081.

2 Background

Introduction

This chapter provides an outline of the various components of this thesis. Several ideas are covered, but none in great detail. For a detailed treatment of each of these topics, the reader is pointed to the appropriate references in each section.

2.1 Origin of Power System Oscillations

Synchronous power generators produce electric power at a fixed frequency. The frequency can be raised or lowered by a small amount depending on the variations in the connected load. If a sudden increase in load occurs, the turbine controller may not be able to respond fast enough and increase the turbine power. In this case, some energy is taken from the kinetic energy of the generator-turbine system and is converted into power. This will reduce the rotational speed and consequently the electric frequency [1]. When groups of generators are operated together (synchronously) in a large power system, the response of each of them to load changes will not be the same. Some may slow down more than others. This causes groups of generators to oscillate against one another or for some generators to oscillate against the rest of the system. These small oscillations, if undamped or under-damped, can cause a loss of synchronisation of several machines that can lead to a failure of the power system.

Inter-Area Modes

Oscillations in a power system can broadly be divided into two types:

1. Local oscillations (or modes) : 1-3Hz
2. Wide Area oscillations (or modes) : 0.1-1Hz

Oscillations are primarily caused by electromechanical oscillations of the generator rotors which are poorly damped [11]. These oscillations can be due to faults or sudden load changes. The effect of a fault can be different depending on the distance of a generator to a fault. Generators close to the fault will see greater power swings that those further away. This difference results in groups of generators swinging against each other, giving rise to wide area oscillations. The natural frequency and damping of these oscillations depends on the power transfer between two areas, the aggregated inertia of each group of generators and also on the impedance of the connecting tie lines [12].

The tendency of generators to oscillate can be explained using a P-δ curve [10]. When a load change or fault occurs, the generators operating point is disturbed from the steady state, where the electrical power matches the mechanical power. Due to this change, a new steady state operating point will have to be established. In moving to a new operating point, an imbalance will be created between the mechanical

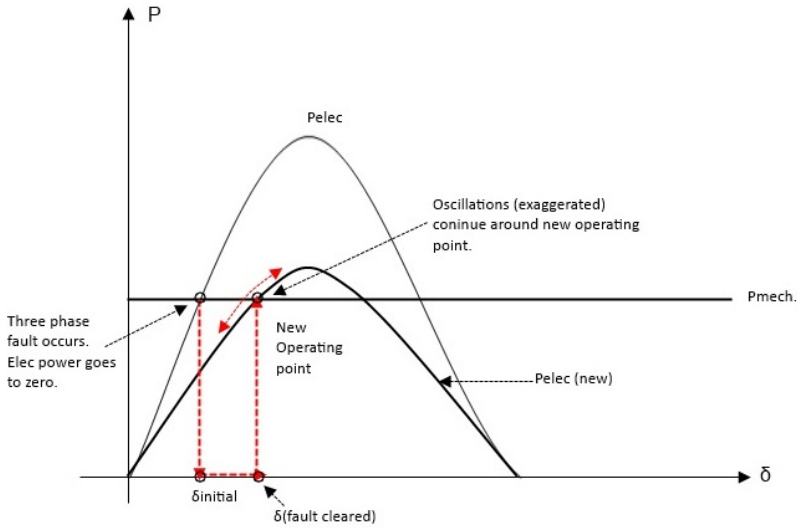

Figure 5: P-δ Curve Showing Origin of Power Oscillations

and electrical powers. This will give rise to an accelerating or decelerating torque. These torques are poorly damped and will lead to low frequency oscillations about the new operating point. This is shown in Figure 5. Sets of generators will then oscillate against each other, giving rise to local modes. A similar reasoning follows for the oscillations between interconnected groups of generators, which gives rise to inter-area oscillations.

Inter-area oscillations tend to cause large power swings between areas but have rarely led to complete system failures. The usual result is a single power system becoming separated into two or more areas, which is known as islanding. This has occurred in several cases such as the in the Southern Brazil network and between the Detroit-Edison and Hydro-Quebec networks, both in the 80's [1]

2.2 Controller Topology Selection

Before testing or implementing a hardware based controller, a model for the architecture topology will have to be selected. This architecture will need to meet real-time constraints. Issues such as the computational requirements have to be considered. Two major methods have been used to date to design stabilisers:

1. Model Linearization
2. Phasor Based Design

Model Linearisation

Traditional controllers for FACTS devices depend on accurate systems models at the operating condition. In the case of large and interconnected systems, data about

all parts of the systems may not always be available. Large systems also tend to be dynamic and change their topology often. POD design is based on small signal and linear analysis techniques. For example, the residue method which is based on these linear techniques requires a linear model of the system. These types of models are difficult to derive accurately for large and inter connected power systems [13]. Linearised models are also only valid for small deviations from the linearisation point (see Figure 6).

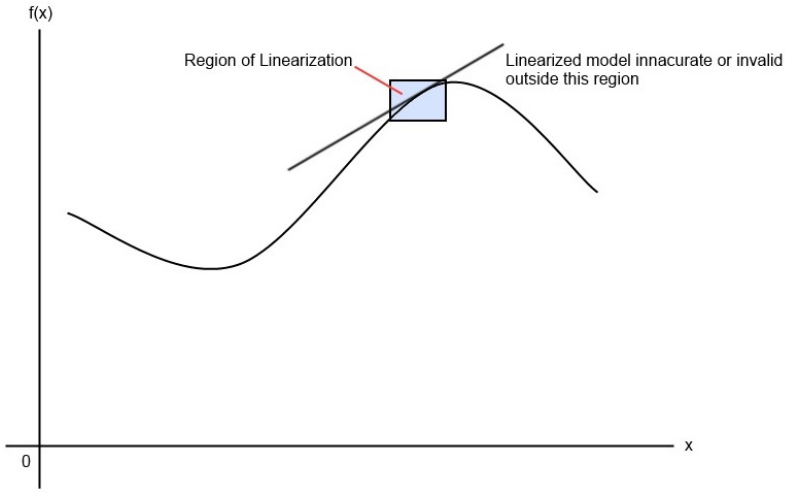

Figure 6: Region of Validity of Linearised Model

Linear model based controllers work by manipulating the eigenvalues of the system state matrix. Eigenvalues with a negative complex part (in the s-plane) indicate a stable system behaviour. For certain modes, the eigenvalue may lie in the right half of the s-plane, indicating a poorly damped and potentially unstable response. These controllers will attempt to compensate the system so that all eigenvalues are in the left half of the s-plane, resulting in a stable system.

The state matrix of the system is commonly derived from a linear model of the system. Further, a system model is linearised for a given system configuration. This refers to a particular state for each system component such as circuit breakers or transformer tap changers. Any change in this configuration, for example a reconfiguration caused due to fault clearing, will also change the model or the operating point. This new configuration will then have to be linearised again. Linearising models of complex systems is computationally intensive. This cannot be performed in real-time, given the limited resources available.

Phasor Based Design

Some of the problems of the model linearisation approach are addressed by phasor based oscillation damping algorithms. These work by extracting the oscillatory part

of a phasor.

The measured signal can be represented as a space-phasor:

$$(t) = s_{avg} + \mathrm{Re}\left\{\vec{s}_{ph} \cdot e^{jwt}\right\} [14] \tag{3}$$

This presents an average value and the associated oscillatory part, in a stationary reference frame. The oscillating part can then be used to generate a control signal for the FACTS (or other controllable device) using a control algorithm. This method is independent of the system state or configuration and is not computationally intensive. Controllers based on this approach also incorporate a degree of error checking and phasor estimation.

One of the reasons why this design has not been widely adopted is the fact that such controllers tend to be highly non-linear and complex [14]. This makes modelling them in system stability analysis studies difficult as commonly used tools in power system simulation are unfit for modelling such controls [23].

2.3 Oscillation Damping using FACTS Devices

Flexible AC Transmission Systems (FACTS) incorporate several static devices for reactive power and hence voltage control such as a TSR[4], SVC[5], TCSC etc. Such devices may be installed for different purposes (e.g. improving a line's voltage profile) but can also be put to use for oscillation damping [8]. This thesis has implemented a model of an SVC hence in the rest of this section, the term SVC will be used. However, this is can be equally applicable to any other FACTS device.

Typically, a device such as an SVC will have a local controller designed to monitor and control a system variable such as voltage. To achieve this control, the SVC absorbs or generates reactive power [7]. The SVC essentially consists of a TCR and a TSC [6]. A TSC is a bank of capacitors that can be connected or disconnected from the system in steps. The capacitors in an SVC supply reactive power to the network while the inductors (TSR) absorb reactive power. The thyristor firing angles can be changed depending on the amount of reactive power required by the network [7]. Changing the amount of reactive power in the network is the fundamental damping action of these devices. A supplementary control signal can be added to the SVC to achieve this damping action but care must be taken that it does not interfere with the device's normal operation [8].

With power system oscillation damping in mind, several other factors also come into play such as the location of the damping device (in this case, the SVC) and

[4]Thyristor-Switched Reactor
[5]Static VAR Compensator
[6]Thyristor Switched Capacitor

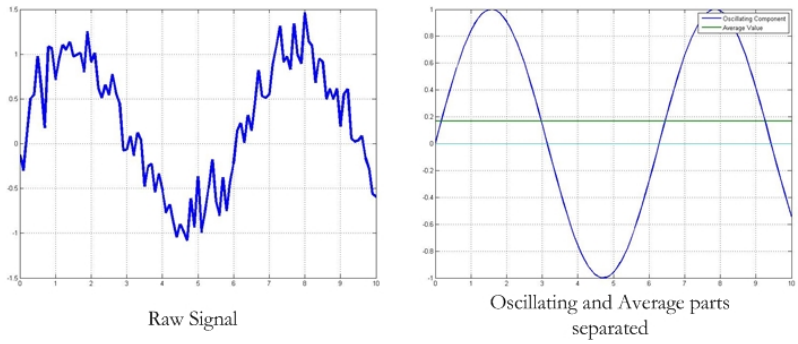

Figure 7: Separating a Signal (L) into oscillating and average parts (R)

the input signal selected [8] to generate the supplementary damping signal. Chow and Larsen list in [8] possible inputs signals along with their related considerations. Among the signals examined in [8] frequency has not been examined in this thesis however, the voltage angle which has been used, is related to frequency.

2.4 Phasor POD

The phasor based damping algorithm was originally proposed by Lennart Ängquist and Carlos Gama [6]. It works by separating the input signal (power, as described in [6]) into an oscillating component and an average valued component (see equation 3 and Figure 7). It takes advantage of the fact that the oscillation frequency for a given network configuration is usually known. Using this known frequency value, a co-ordinate system, rotating at this known frequency, is set up where the oscillating component is continuously extracted as a phasor [6]. In steady state, when the oscillation magnitude is zero, the extracted phasor will also have zero magnitude. In the event of a disturbance such as the loss of a generator or a fault, oscillations begin and the magnitude of the extracted phasor also increases. The method described in [6] then uses this extracted phasor data to generate a damping signal for a FACTS device such as a TCSC [7] simply by adding a phase shift. The phase correction will serve to bring the damping signal into phase-opposition with the original signal.

This method also incorporates a phase correction factor into the output signal to provide an arbitrary phase correction [6]. This is exploited in this thesis to allow the Phasor POD algorithm to work with different input signals.

2.5 Wide Area v/s Local Controllers

Traditionally, power system oscillation damping has been achieved through locally available signals using local controllers such as a PSS [5]. With the development of more modern communication networks and control algorithms, the reasons for preferring local signals as mentioned in [8] (communication delays and communication network reliability) become less of an impediment to adopt wide-area signals as

[7] Thyristor Controlled Series Capacitor

the preferred choice. A brief set of advantages of wide-area controllers is presented below [4] - [10].

1. Better Observability of Inter-area modes
2. Ability to provide damping signals to several points on the network
3. More input signals (number, type) available to the controller, increasing damping effectiveness and redundancy
4. Best observation point for a particular mode and best control point are often not the same

2.6 Two Area Test Network

A test model was selected with the aim of observing inter-area oscillations. This model was the Two-Area Four-Machine model, originally developed by Klein, Rogers and Kundur [10] (Figure 8). The model was selected as a study model because it includes both inter- and intra-area modes. The model essentially consists of two, distinct generations areas connected by a weak, heavily loaded line (double circuit). The two areas are identical in all respects except for the generator inertias. For detailed parameter listings see Appendix B. This model is similar to the one in [10]

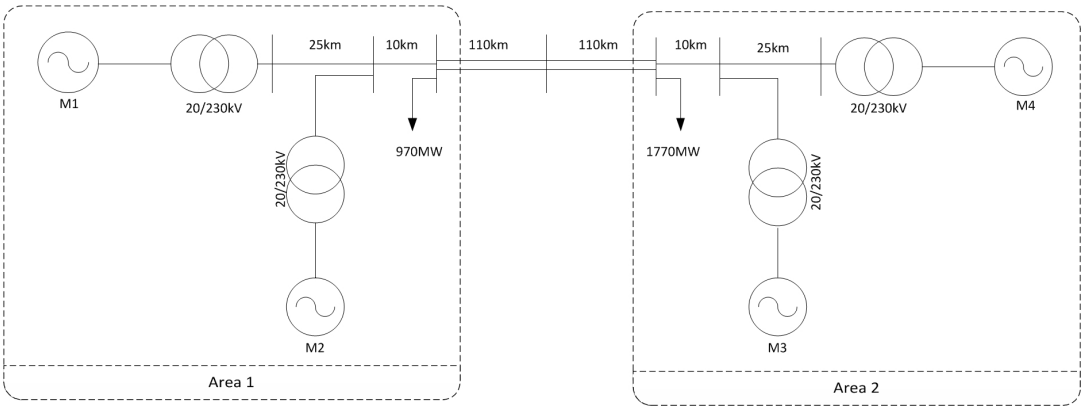

Figure 8: Single Line Diagram of Two Area Four-machine test system

Both machines in Area 1 have an inertia constant of 6.5s while both machines in Area 2 have an inertia constant of 6.175s. This difference is, in part, what will give rise to inter area oscillations in the event of a system disturbance such as a fault. The inertias influence the rate at which generator angular velocities, and hence frequencies, change when subjected to disturbances. This difference in response speed between two or more generators is one of the reasons for oscillations. Local loads are dispatched in both areas such that there is a net power flow of 413MW from Area 1 to Area 2. The Simulink Model used in this thesis is based on the model developed for PSS performance studies by Kamwa [15]. This can be accessed on most MATLAB versions (with access to SimPower Systems) simply by typing `power_PSS`

at the command line.

An analysis of the oscillation modes of this simple model reveals that three modes are prominent:

- Local Mode: Area 1 : Both Machines against each other : 1.12Hz
- Local Mode: Area 2 : Both Machines against each other : 1.16Hz
- Inter-Area Mode: Area 1 Machines against Area 2 Machines : 0.64Hz

This system is inherently unstable and will rapidly collapse if no stabilizing elements are included. The original SIMULINK model was developed to study Power System Stabilizer (PSS) performamce and includes a PSS at each of the generators. With this additional stabilization, the system can be made stable.

2.7 Model Preparation for Simulation

The Simulink model is configured to use a continuous, 60Hz phasor solver. In order to prepare the model for simulation on the OPAL RT platform, the model will have to be converted to 50Hz, discretized and simulated at a fixed step rate of 50μs. This is fairly straightforward and consists of replacing the phasor measurement blocks with discrete versions as shown in Figure 9. The system frequency also needs to be changed in the simulation parameters and also in all the filter and line elements. The simulation did not envisage a system with circuit breakers and hence these were discarded from the final model.

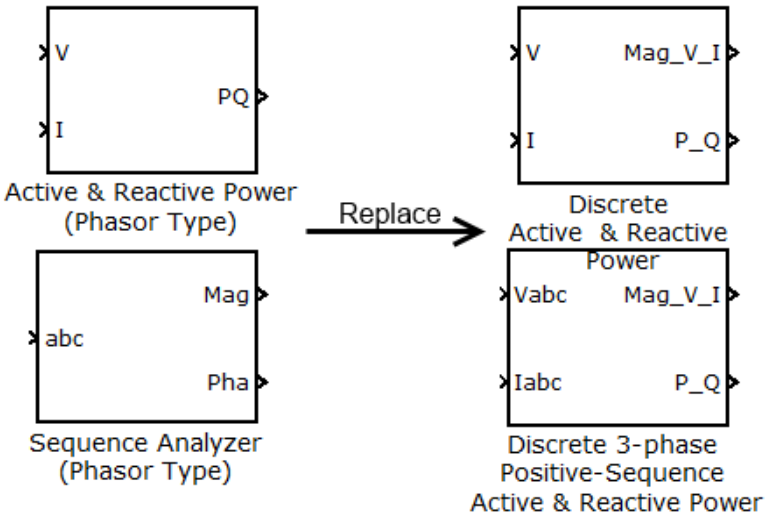

Figure 9: Illustration of phasor to discrete model conversion

The only major problem encountered was the initialization of the generators when using the discrete models. With the default settings, numerical oscillations

were produced each time the model was initialized and prevented the simulation from running. The solution to this is documented in the MATLAB help-files and consists of adding a parasitic resistive load of about 2.5% of the nominal machine power when using a 25μs time step. The machine ratings in this case are 900MVA and the parasitic load come out to 22.5MW for each machine, considering a 50μs step size. For both machines in each area, this is 45MW. This amount of load is subtracted from the local loads in each of the areas to keep the power flow solution consistent.

Subsystem Grouping

A Simulink model has to be grouped into distinct subsystems before it can be run on the eMEGASIM simulator. Every model must have a 'Master' subsystem which contains the computational sections of the model [22]. Other 'Slave' subsystems can also be used to group additional computational elements of the model. While every model must have one and only one 'Master' subsystem, 'Slave' subsystems can number from none to several depending on the computation complexity involved [22]. Each subsystem is assigned to one processor core on the real-time simulator. A 'Console' subsystem can also be included for interaction with the running model or for monitoring signals [24].

The two-area test case used in this thesis was grouped according to this philosophy, with one master subsystem that included the Area-1 elements, two slave subsystems that included Area-2 elements in one and a measuring and monitoring subsystem for line parameters. The measurement and monitoring slave subsystem also included the SVC and hardware interface blocks which are described later in this thesis. This basic outline is shown in Figure 10. Both Slave subsystems are shown in grey while the Master subsystem is shown in green.

The model was configured to execute at a time-step size of 50μs. This meant that all computations, writing to output values and reading input values would have to be completed in this 50μs interval. This was also the motivation behind grouping the two areas into two distinct subsystems as each subsystem would be assigned to a different logical core on the processor.

Additional Elements

Since this model was needed to test the performance of a hardware based controller, besides grouping the model into sybsystems, additional elements were necessary. The most important would be the Static VAR Compensator (SVC) and it's associated control system. Additionally, the POD algorithm implemented in hardware needed to be interfaced with the model implemented in Simulink for the purposes

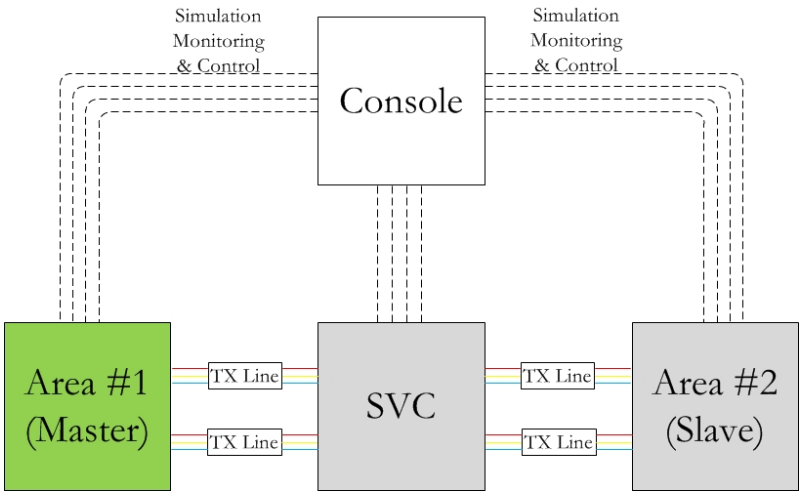

Figure 10: Two-Area System Regrouped for Real-Time Simulation

of comparison.

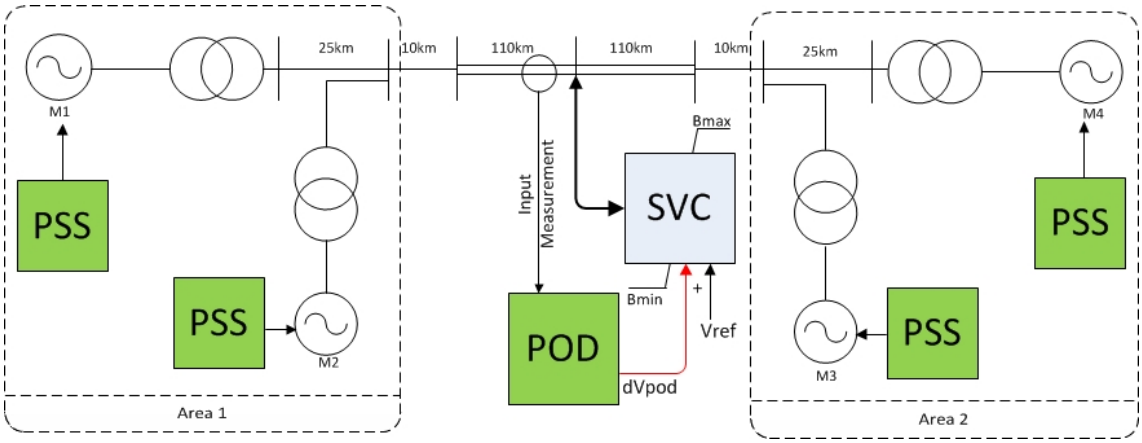

Figure 11: Two Area Model with SVC & POD

The SVC model included was an average value model (identical to that used in [7]) with a PID controller. The location for this SVC is at the mid-point of the line connecting the two areas. This is the point where voltage swings will be the greatest and also where the SVC can be most effective and damping power swings [8]. The Phasor POD model was developed by M.S. Almas and L.Vanfretti [7] as a Simulink model (see Page 16) with associated MATLAB script files.

The POD takes a particular input and generates a damping signal based on the Phasor POD algorithm. This damping signal is scaled, suitably phase shifted (depending on the input) and then incorporated into the control system of the SVC as an additional damping signal. With this implementation, the SVC functions as expected during normal system operation. At such times, the supplementary signal

generated by the POD is close to zero. Once oscillations start and damping is required, the supplementary signal generated by the POD grows in magnitude.

2.8 Hardware Interfacing

The Simulink model running in the real-time simulator needs to be interfaced with hardware elements. To do this, data was collected from different locations of the model and interfaced with the simulators analogue outputs. In this thesis, three-phase voltage and current data variables were interfaced. These were measured from the ends of each area, just before the transmission lines. These points are indicated in red in Figure 12.

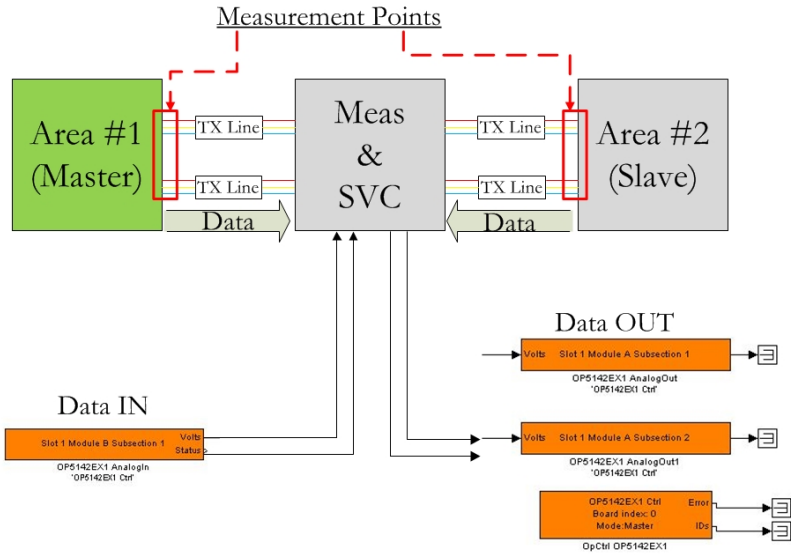

Figure 12: Hardware Interface of Simulink Model Showing Measurement Points

Figure 12 also shows the hardware interface point for the input damping signal. Essentially, this system replaces an entire section of the SIMULINK model with a hardware model that is interfaced through analogue signals. Since the SIMULINK model accepts discrete values, A/D[8] and D/A[9] conversion is performed at the inputs and outputs of the simulator respectively.

[8]Analogue to Digital
[9]Digital to Analogue

3 Code Development

This chapter covers the development and testing of the various components of the controller software in LabVIEW for implementation on the compact RIO (cRIO) platform. Aspects of the development process for the FPGA sections and the Real-Time sections are presented. No software development process is ever linear, straightforward or bug-free and neither was this. While the final code took several iterations, numerous trials, failure after failure and hours of waiting (while code compiled) much of these are not covered here. This section only focuses on the final, working code, it's development and working.

3.1 POD Algorithm Conversion

The purpose of this hardware controller was to replicate the behaviour and function of an Oscillation Damper previously implemented in SIMULINK. The basic outline of this algorithm is shown in Figure 13. Figure 13 shows two parallel POD implementations, both of which are identical. Each POD accepts inputs such as the search frequency, ω_{cs1}, the sampling time T_s, the phase correction *alpha* and the signal gain elements at the end. Each POD has a ramp integrator section followed by a Recursive Least Squares based Phasor estimator. The function blocks at the end implement a co-ordinate transformation. For complete details of the algorithm and its working, the reader is referred to [6].

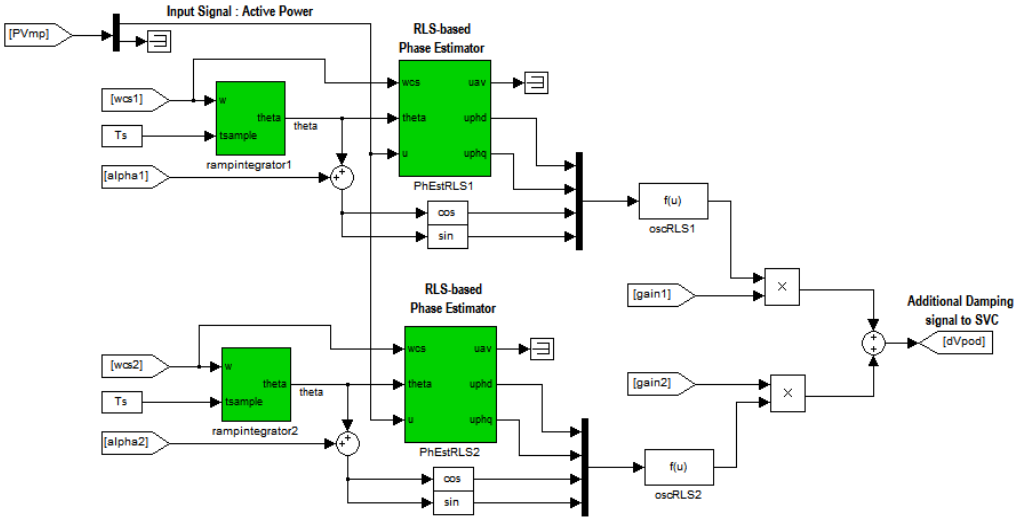

Figure 13: POD Implementation in SIMULINK

This functionality was selected for replication on the FPGA. The implementation of the POD algorithm on the FPGA followed the SIMULINK interpretation exactly. A section of the POD code is shown in Figure 14. The 'Control Configuration' block is used to pass parameters to the POD algorithm, equivalent to the Simulink version.

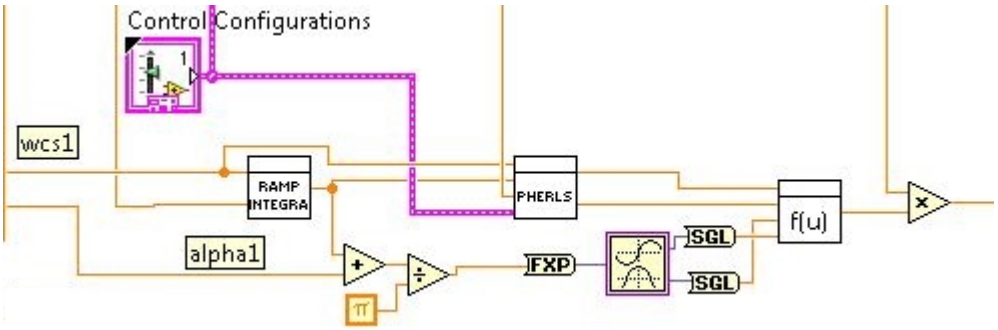

Figure 14: POD Implementation in LabVIEW

3.2 Controller Hardware Outline

Details of the hardware used in this thesis are documented on Page 10 but are repeated here for continuity within this chapter.

The cRIO platform from National Instruments is a reconfigurable controller with an embedded FPGA and has reconfigurable modules within a robust chassis. The core of the controller is a real-time controller that can be programmed using LabVIEW. The cRIO used here was configured with an analogue voltage output module. The POD algorithm runs on the embedded FPGA which is interfaced to the analogue voltage output module. The real-time controller functions as an interface between the workstation computer and the FPGA. It (the RT controller) receives input data and configuration parameters from the workstation computer in a LabVIEW interface and sends these to the POD algorithm on the FPGA. It also logs data produced by the FPGA algorithm along with any errors.

For the Phasor Oscillation Damping (POD) Algorithm, the cRIO9081 [34] was chosen. This has a Spartan 6 LX75 FPGA. After initial testing on the cRIO9076 [35] (identical FPGA model, lower resources), it was determined that more FPGA resources were needed for the POD algorithm. Although the code can be optimised to run on the cRIO9076, further development will be limited due to the constraint on the FPGA resources.

Note

Section 2.7 on Page 18 covers the code development process in Simulink. However, since a starting model was available [7] and the model only had to be 'made compatible' for real-time simulation, this process is not covered in detail. If needed, a basic guide that covers the essentials can be found at [22]. The actual process of readying the model for simulation had several more steps than covered in Section 2.7 but these are omitted from this thesis for the sake of brevity.

Also, this sections makes frequent reference to terminology specific to the LabVIEW environment [30] such as Virtual Instruments (VI's), shared variables, and the different components of the NI cRIO platform. For more details, the reader is referred to the online documentation of these, available on the National Instruments website [30].

3.3 Architecture Outline

The prime aim of this design was modularity. If a certain section of code was required to be changed, the process of change should be as simple as replacing the section of code with another. With this in mind, the schematic in Figure 15 was prepared. This shows an independent, network connected POD. However, since no PMU data-extraction software was capable of running independently on the RT controller, an alternate design for development was chosen. This software (identical to that used in [9]) would run on a workstation computer, extract data from the PMU stream and send this extracted data to a LabVIEW program running on the same computer [9]. Besides this difference, the only other modification made was that a local control function is not implemented on the cRIO alongside the POD function. A local control function such as a PSS (Power System Stabiliser) was implemented and tested but those results are not presented here. In practice, this could be implemented alongside the POD function on the same device but additional FPGA resources will be required.

The code design implemented here achieves this aim of modularity in several ways. The POD algorithm is independent of other sections of code. The only parameter required here is periodic data input. The controller damping parameters have default values and can be updated when required. The Real-time(RT) section of the code in this thesis serves as an interface between the workstation computer running the application interface and the POD algorithm running on the FPGA. Since this section of the controller has a network interface, the eventual aim is to unwrap a PMU stream on the controller thus removing the need for an additional computer for this purpose. Also, various other functions such as parameter setting and input signal selection can be automated and performed independently on the controller itself. The RT and FPGA controller can then function as an autonomous damping unit, with manual intervention used when needed.

Note

The architecture shown in Figure 15 was the initial, fully independent design goal. Not all aims of this design were met and several changes were made. The architecture implemented on the controller differs significantly from this initial design. For a detailed description of the initial architecture, together with justifications for design choices and revisions, see Appendix D.

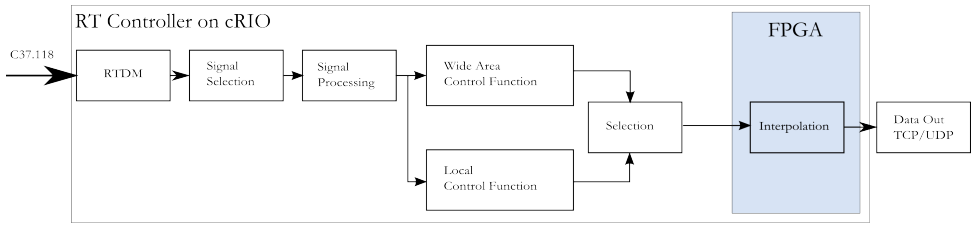

Figure 15: Modular Architecture of the Oscillation Damper

3.4 Architecture Description

The hardware implementation of the POD was based on the Compact Reconfigurable Input / Output (cRIO) 9081 from National Instruments. This controller is equipped with an on-board Field Programmable Gate Array (FPGA) running at 400MHz. For details of this device see the listing on Page 10. A three level design was chosen to be implemented on this controller. Three, independent levels of code were written and run independently on the FPGA, the RT Controller and the workstation computer. Figure 16 shows the various code sections together with their execution locations. Elements in red indicate VI's or LabVIEW code and blue elements show their execution locations (both connected by arrows). Note that in this particular case, the cRIO requires that the workstation computer be running in order to receive data from the PMU streams.

To implement the three-level program structure described before, the entire LabVIEW project was based on a template called 'FPGA Control on Compact RIO' [25]. This implements the three level control structure described above. The three sections are outlined below and then presented in detail.

- Core FPGA Software (FPGA Main.vi) : Interacts with hardware terminals for I/O and runs POD algorithm
- Real Time (RT) Software (RT Main.vi) : Manages network communication to remote terminal and also generates performance monitoring data
- Remote VI (UI Main.vi): Runs on workstation computer: Used to update algorithm parameters and monitor data & performance. This layer is non deterministic.

3.4.1 FPGA VI

The FPGA VI consists of the Phasor Oscillation Damper (POD) algorithm. It also includes a 'watchdog' to ensure that communication is maintained between the RT section of the code and the FPGA section. This also allows for operation control. The POD algorithm can be switched on or off or test values can be written to the analogue outputs, bypassing the POD algorithm completely. This is the only section of code that directly communicates with analogue hardware. The FPGA loop rate is set to 50μs to match the loop rate of the Real Time simulator. This

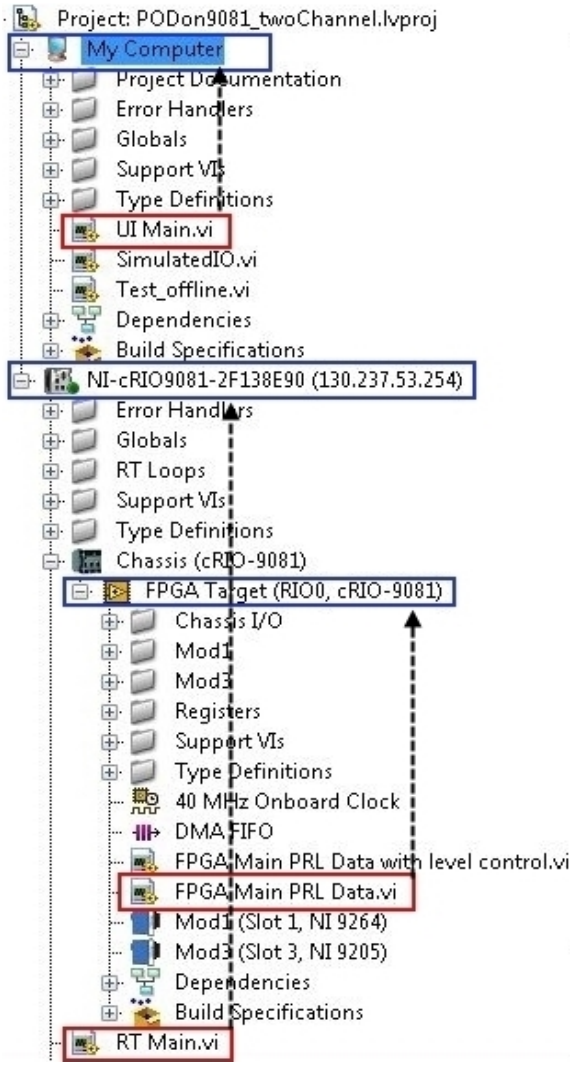

Figure 16: LabVIEW Project Browser

section is flexible and can incorporate parallel POD algorithms, each responding to a different oscillation frequency. The limiting factor here is that the FPGA resources are finite. The POD algorithm was implemented on the FPGA as it is computationally intensive but concurrent, a task that the FPGA is ideally suited for. This would free up the RT controller to handle other tasks such as network communication and data logging.

3.4.2 Real Time VI

The Real Time (RT) VI runs on the Real-time processor of the cRIO. It provides an interface between the FPGA and the workstation computer. This VI handles user input and network communication between the FPGA and the user. This VI performs functions such as passing parameter values to the FPGA, monitoring the inputs to and the outputs from the FPGA algorithm, transferring these values to

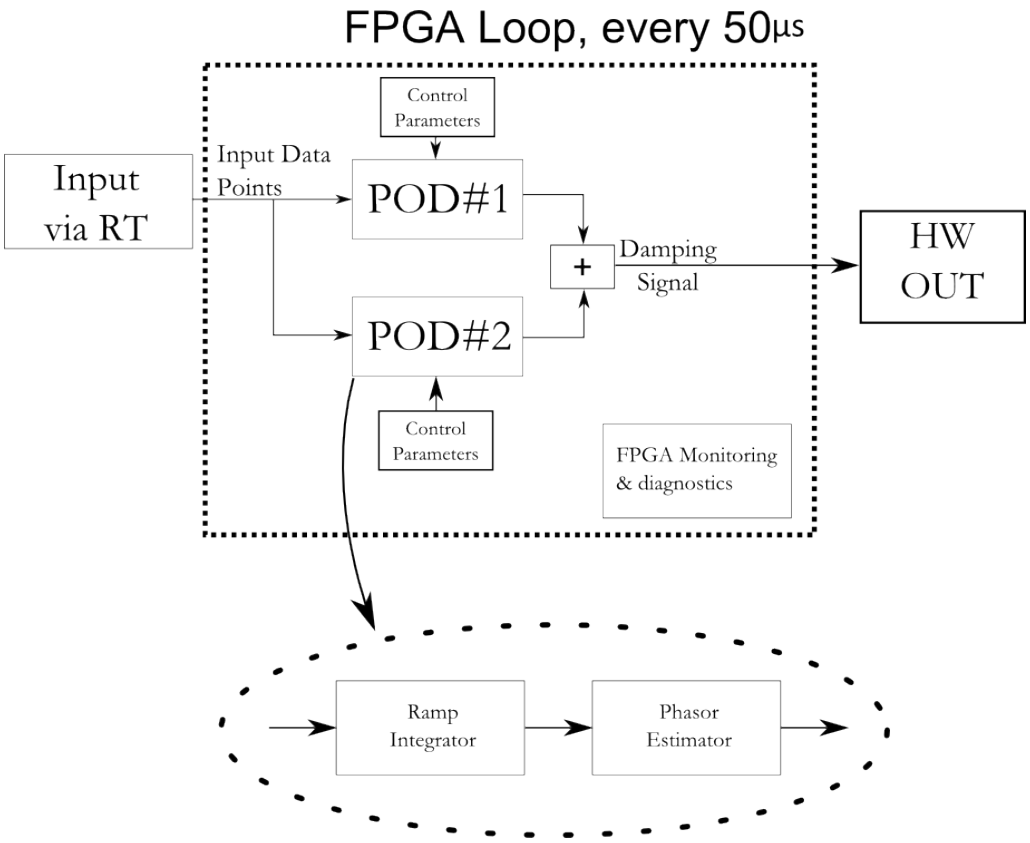

Figure 17: FPGA VI

the workstation computer for logging and other functions such as error handling. This VI runs at loop rate of 20ms[10]. Not all data that the FPGA generates is sent back for monitoring. Instead, the RT processor periodically samples the outputs of the FPGA and returns only those values for monitoring.

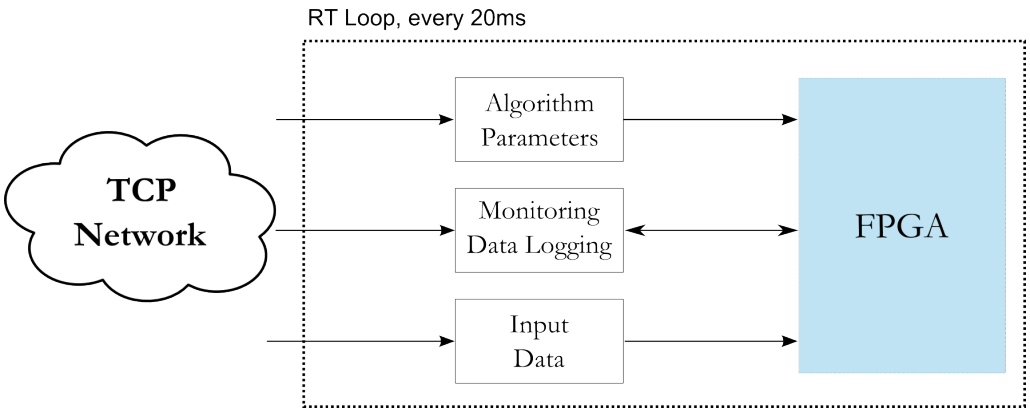

Figure 18: RT VI Block Diagram

[10]After several tests, this was the fastest that this code could run

3.4.3 UI Main VI

This VI runs on the workstation computer and acts as the user interface for the RT controller. It accepts user inputs for the various controller parameters and sends them over the network to the RT controller. Also, it allows for monitoring of the inputs and outputs of the FPGA. In this thesis, this VI performs another essential function. The PMU streams are unwrapped to extract the current and voltage phasors. To achieve this, special software is used. In this case, Statnett's Synchrophasor Software Development Kit (S^3DK) was used to unwrap the PMU streams and extract phasor data [9]. This allowed for data to be extracted and used directly in the LabVIEW environment. Since the PMU data rate was 50 messages a second, new data was available every 20ms. The loop rate used by the S^3DK was hence 20ms. The selection of a input for the controller and the computation are performed in this VI. For example, if active power is to be used as a POD input, voltage and current values must be multiplied to obtain the active power. If the voltage angle difference is to be used as an input, the required calculations are performed in this VI.

Once data was extracted from the PMU stream and the required computations were performed, this data was sent to the POD (on the cRIO) over the network. To achieve this, LabVIEW provides a functionality called a Shared Variable [27]. A shared variable can be used to exchange data between LabVIEW applications running on different machines. In this case, both the cRIO and the workstation computer are connected to the same network and so exchanging data using a shared variable is straightforward. This is similar to a variable in any other programming language except that the value of the variable is published to the network and can be read to or written from different locations. This process adds delay and non-determinism to the design.

3.4.4 cRIO Deployed as PMU

Two cRIO's were deployed as Phasor Measurement Units (PMU's) to receive three-phase analogue current and voltage measurements and to generate a synchrophasor data stream. For this, two identical cRIO9076 controllers (hardware details on Page 10) were used. National Instruments provides PMU software[11] for the FPGA and the RT controllers. This includes a web interface from where the user can configure various parameters such as the data rate and the instrument transformer ratios. For this case, the data rate selected was 50 messages a second, or one message every 20ms. This meant that the input voltage and current values were reported every 20ms. The reporting rate was not a constraint to observing the inter-area oscillation as the frequency was 0.64Hz, which could easily be observed at a reporting rate of 50s/s. The two PMU streams were sent to a Phasor Data Concentrator, to be combined into a single stream. This single output stream from the PDC was configured to include data from both PMU's. The PDC also logged the input data that the PMU's generated.

[11] http://sine.ni.com/nips/cds/view/p/lang/en/nid/211676

3.5 Implementation Challenges

The development of this controller architecture and the associated code was not without several challenges. While several problems were faced, only the most important are covered here along with the solutions.

3.5.1 Necessity of Workstation Computer

The architecture chosen was designed to be 'headless'. This means that the controller software running on the cRIO should be able to function independently. The workstation computer is only required to monitor the controller's performance, log data and to take manual control if required. Ideally, the RT controller should be able to connect to a PMU/PDC stream, unwrap and extract the phasor data and perform the required computations to obtain the desired data. Presently, the software required to do this on the RT controller is not available. The RT controller has the needed resources to perform the computations required in the selected 20ms cycle time. However, since the software required to unwrap the PMU/PDC data stream is presently only able to run on a desktop computer, the workstation computer is a necessity.

3.5.2 Different Loop Rates

This was one of the most significant challenges in the development of this controller. As mentioned in previous sections, each component of the simulation and the controller ran at a different loop rate. These are summarised in Table 1.

Element	**Loop Rate**	**Mode**
OPAL RT Simulator	50μs	Real Time
cRIO PMU	20msec	Real Time
Workstation Computer	20msec	Not Real Time
POD RT	20msec	Real Time
POD FPGA	50μs	Real Time

Table 1: Comparison of Loop Rates of Different Components

Note that the section running on the workstation computer, the software to extract phasor data from the PMU stream, does not run in real-time. This introduces a variable source of delay in the control loop. The FPGA is run at the same rate as the OPAL RT simulator so that data is always available at the input of the simulator.

These different loop rates presented a problem to the execution of the code. The FPGA would expect data every 50μs or 2000 samples per second. The PMU's were

not capable of such a high data rate as they were limited to a maximum of 50 samples per second. An additional limitation was that the fastest speed at which the RT section of the POD controller (on the cRIO) could execute was 1ms. One solution to this problem would be to **upsample** the data from 50 samples per second to 2000 samples per second required for the FPGA. This would have to be performed on the RT controller. A FIFO[12] buffer would have to be used to buffer the data generated by the up-sampling as it was gradually consumed by the FPGA. The major problem with this method was that the up-sampling process on the RT controller is computationally intensive and would not run at the required 20ms loop rate.

The alternative solution would be to implement a **sample and hold** algorithm on the FPGA. As data was extracted every 20ms, the RT controller would receive this data and send it to the FPGA. This value would then be held constant till the next data point arrives. This was implemented in a very simple way, using a temporary variable. The shared variable used to stream data from the workstation computer to the RT controller has an important limitation that it implements an internal FIFO buffer. This is implemented to account for TCP transport delays and lost packets [27]. As a consequence of this, data that is written to this shared variable can only be read once. Subsequent reads will either return invalid values or zeros. To overcome this, the value read from the shared variable was stored in a local variable on the RT controller. This functions similar to a variable in languages such as C and C++ in that reading the variable repeatedly will produce the same value each time. This continues until a new value is assigned to the variable.

This implementation resulted in the real-time simulation and the hardware controller running at the same time step. The only difference between the performance of simulated POD (in Simulink) and that running on the cRIO was the data resolution of their inputs. The simulated POD received new data every 50μs while the cRIO-based POD received data every 20ms.

3.5.3 Analogue Limits

The original POD (Phasor Oscillation Damper) algorithm was developed and simulated in an ideal, noise-free environment with zero delay. More importantly, no limits are imposed on the magnitude of either the controller's inputs or outputs when it is simulated. In contrast, a hardware-based implementation that uses analogue signals faces several challenges, the most significant of which is the analogue signal magnitude limits.

Consider the analogue outputs of the OPAL RT simulator listed on Page 9. These are low level outputs and can be directly connected to the PMU inputs. However, the inputs modules of the PMU's are rated for 0-300V and a 0-16V analogue signal will not use a significant portion of this range. Additionally, a signal of such a small

[12]First In First Out

magnitude will also be contaminated by noise and will consequently have a poor Signal to Noise ratio. A similar argument can be made for the current outputs.

The output of the simulated POD can vary over several orders of magnitude, ranging from 10^5 at times of peak damping to as small as 10^{-3} once the oscillation magnitude has become small. It is not possible to recreate analogue signals with such vast ranges. The voltage output module used with the cRIO here had a 24-bit resolution and was limited to 10V in magnitude. Any values generated by the POD algorithm that were greater in magnitude that 10V would cause output saturation. All these issues meant that signal magnitudes had to be amplified in certain cases to use the full measurement ranges or had to be limited in other cases, so as to capture variations without saturation.

3.5.4 FPGA Accuracy and Data Formats

While the FPGA is a fast, deterministic and reliable computational device, it brings with it a set of limitations. Most of these arise from the fact that an FPGA has no operating system as such and all circuit logic is directly implemented in hardware. All computation is performed at the bit level and hence can become very complex. This limits the amount and complexity of computations that can be performed with the FPGA. Functions such as division or multiplication consume significant space on the FPGA[33] as do arrays and large amounts of data. The FPGA also implements a unique numeric representation called Fixed Point[30]. Here, the number of bits assigned to represent the integer and fractional part of a number is fixed before code execution[33]. For example, if four bits are used to represent the integer part of a number, then the maximum binary number that can be represented is [1111] or 15 in decimal. This representation stores integer numbers as bits corresponding to increasing powers of 2, identical to binary. The same logic is used to represent the fractional part of numbers, however decreasing powers of 2 are used. Typical floating point calculations, although possible, consume significant space on the FPGA and are typically slower than corresponding fixed-point calculations[33]. Trigonometric functions such as a sine or cosine can be implemented using specifically designed code that takes several clock cycles to execute. A trade-off has to be made between code execution speed and accuracy.

The Phasor POD algorithm implemented in this thesis uses floating point calculations, multiplication, division operations and trigonometric operations. Due to the FPGA design, not all these calculations are performed in the same data format. The FPGA-specific data format, the Fixed Point representation (see above), is used to optimise complex computations such as those required in trigonometric and Fourier functions. The drawback of this representation is that a trade-off must be made between accuracy and the range of values that can be represented. Keeping in mind the fact that the input values to the POD algorithm are not necessarily limited in magnitude, the POD algorithm is implemented using Floating-point numbers. Cer-

tain functions used in the algorithm, in particular the trigonometric functions, use FPGA-optimised code and require input and output in the fixed-point representation. Conversion between these two formats (Fixed and Floating point) sometimes results in errors. The floating-point format includes a representation for calculations that result in infinite values or complex valued results, called **NaN** (Not a Number)[33]. This representation is not available in the fixed-point format and conversion results in errors. The most common result is that a conversion from a floating-point NaN results in a fixed-point number where all the bits are 1. This produces a finite number and is incorrect.

A further constraint is the requirement that the each iteration of the POD algorithm must complete within 50μs. This limited the accuracy of the trigonometric functions used. Due to all these factors and the limited FPGA resources, only a single POD algorithm could be implemented on the FPGA instead of the two parallel implementations.

3.5.5 Signal Delay

It was mentioned in previous sections that the FPGA algorithm was set to run at an interval of 50μs. While this is identical to the Real-Time simulator step size, this also represents a delay between the FPGA receiving data and producing an output. In this case, the delay remained constant at 50μs. Each stage of the process represents a delay which when combined amounted to a significant value. Compared to the simulated POD which receives data directly from other SIMULINK blocks, the cRIO-based POD's data must go through PMU's, a PDC, the TCP network, an unwrapping software and sections of LabVIEW code before it is finally received at the input of the FPGA. This total delay is a significant factor to be considered when implementing a control system as it limits the damping that this system can achieve.

3.5.6 FPGA Code Optimisation

The FPGA on the cRIO has limited resources available for computation. Optimising code for space and speed were two important goals during this implementation. The Phasor POD algorithm, as implemented in SIMULINK, allows for code to be grouped into 'sub systems'. LabVIEW implements a similar functionality as sections of code that are used often can be grouped into 'Sub VI's'. This concept is similar to the concept of a function in other programming languages. Passing parameters to a SubVI (function) is optional as is the SubVI (function) returning a value. With the aim of modularity in mind, sections of the POD algorithm were grouped into SubVI's, such as the RLS (Recursive Least Squares) phase estimator and the co-ordinate transformation function ($\alpha\beta$ to dq). This modular code design would allow these functions to be reused if multiple, parallel POD algorithms are included in the future. If two POD implementations are included on the FPGA, each will have identical RLS phase estimators and co-ordinate conversion sections.

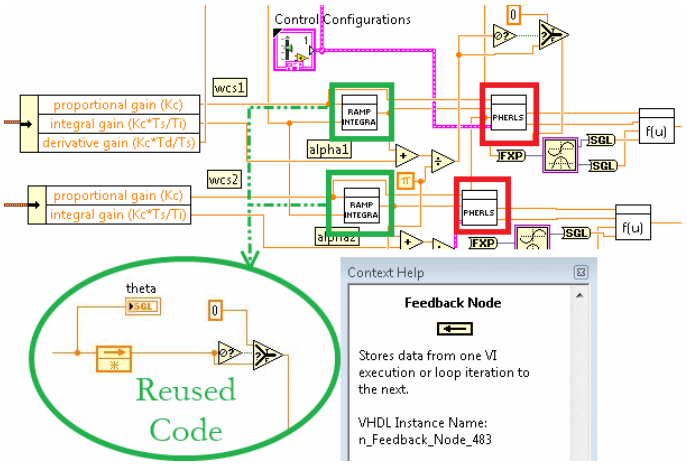

Figure 19: Parallel Code Implementation Showing Code Reuse Issues

Code for these sections can be shared as described below.

To optimise FPGA code even further, LabVIEW provides the option to make SubVI's (functions) 're-entrant'[33]. This means that only one instance of the SubVI actually exists on the FPGA hardware and calls are made to this, same VI. One limitation of this is that simultaneous calls to the same section of code have to be queued and cannot be executed in parallel. The alternative approach is to create an independent section on the FPGA hardware for each instance of the SubVI. This means that if two parallel POD implementations are made on the FPGA, each will have a different section of FPGA hardware devoted to each SubVI. The RLS phase estimator of POD-1 and that of POD-2 will be different and independent. This allows for true parallel execution of code, however, the trade-off is that more FPGA resources are used.

Initially, the method chosen was to make each SubVI, re-entrant. This meant that parallel POD implementations shared the same section of physical FPGA hardware. When tested, this implementation was found to produce unexpected results at the output. On analysis, the culprit was the memory elements implemented in the SubVI's. The memory elements were used for numerical implementations of functions such as integration. This means that the value generated in the previous cycle of code execution is stored and used in the present cycle. This created a problem when parallel POD algorithms are executed as the data from one algorithm gets used in the other (see Figure 19). This is because the actual SubVI implementation is on the same, physical FPGA hardware. This method was ultimately discarded for all sections of code that used memory elements. This limits the number of parallel algorithms that can be executed on the FPGA.

4 Test Setup, Development & Results

The power oscillation damper (POD) developed for the cRIO was tested using four different input signals. The test method consisted of two steps:

1. Offline simulation to verify that the POD parameters were correct and the algorithm was functioning properly
2. Real-time HIL (Hardware In-the-Loop) test

This two step method was followed for only two of the inputs, active power and current magnitude. Only real-time simulations were performed for the other two inputs, voltage magnitude difference and voltage angle difference. The Phasor POD design separates the oscillatory part from the average value of the input signal and thus it can be used with different input signals. Using a PMU/PDC data stream provides flexibility in selection of input signals. The different signals available on a typical PMU stream are:

1. Line Voltages
2. Phase Voltages
3. Phase Currents
4. Current and voltage angle differences
5. Active Power
6. Reactive Power
7. Apparent Power
8. Frequency

By combining data from multiple PMU's, differences can be computed between variables at two different network locations. This fact is exploited when using voltage magnitude and angle differences as inputs to the POD. In the case of this particular thesis, where an SVC is used to control the line voltage, the only quantity that cannot be used as a control input is the line voltage magnitude at the SVC location. If this voltage is to be used as an input signal to the POD, the controller design will have to be modified to include a feed-forward element as well. This thesis investigates the use of the active power, positive sequence current, voltage magnitude difference and voltage angle differences as inputs to the POD.

4.1 Active Power as Input

The original work (reference) of the Phasor POD development used active power as an input. In this case, the POD settings used in the SIMULINK model are listed in Table 2. For the purposes of comparison, a simulation was run in SIMULINK with only the POD providing damping. The test was identical to that used in all the other tests (a 5% change in the voltage reference point of machine M1). The result is shown in Figure 20. It can be observed that the power signal does not show only the 0.64Hz mode but also other modes. The system in 11 has a total of five stabilizing elements: four Power System Stabilisers (PSS, one for each machine) and the POD attached to the SVC (all shown in green). This simulation used only the Phasor POD algorithm to damp the oscillations caused due to a disturbance. All other damping elements (PSS at each machine) were disabled. For comparison purposes, the scenario with only the PSS's damping the oscillations (and no POD) is presented in Appendix A.1.

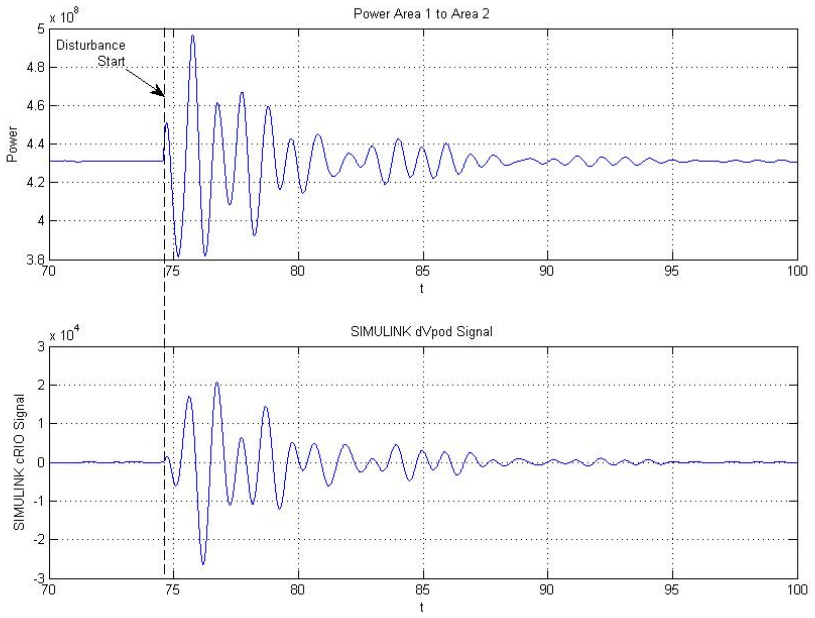

Figure 20: Damping achieved in SIMULINK using a simulated POD

Table 2: POD Parameters used in SIMULINK : Active Power Input

Search Frequency	ω_{CS}	$2\pi 0.64$
Phase Correction	α_1	$\frac{\pi}{2}$
Gain	-	0.0005

Simulations were carried out subsequently to verify if the POD implementation on the cRIO is able to damp power system oscillations in a manner similar to the simulated POD. As can be seen from Figure 20, the simulated POD algorithm is indeed effective at damping oscillations. Similar behaviour is to be expected from the POD implementation on the cRIO. Note, also, that the POD alone, acting as a damping element, is significantly less effective than the combination of local PSS's installed at each of the four machines.

4.2 HIL Verification using Active Power as Input

To verify the implementation of the cRIO POD, a Hardware in-the-loop setup was constructed with the cRIO POD receiving power system data through a PMU. The power system model (Two Area Test Model) was executed in real-time on the OPAL RT simulator. Analogue currents and voltages were then be sent to the PMU's which generated synchrophasor data streams. Relevant data was then extracted from this stream and sent to the POD algorithm running on the cRIO. The entire data path is shown in Figure 21.

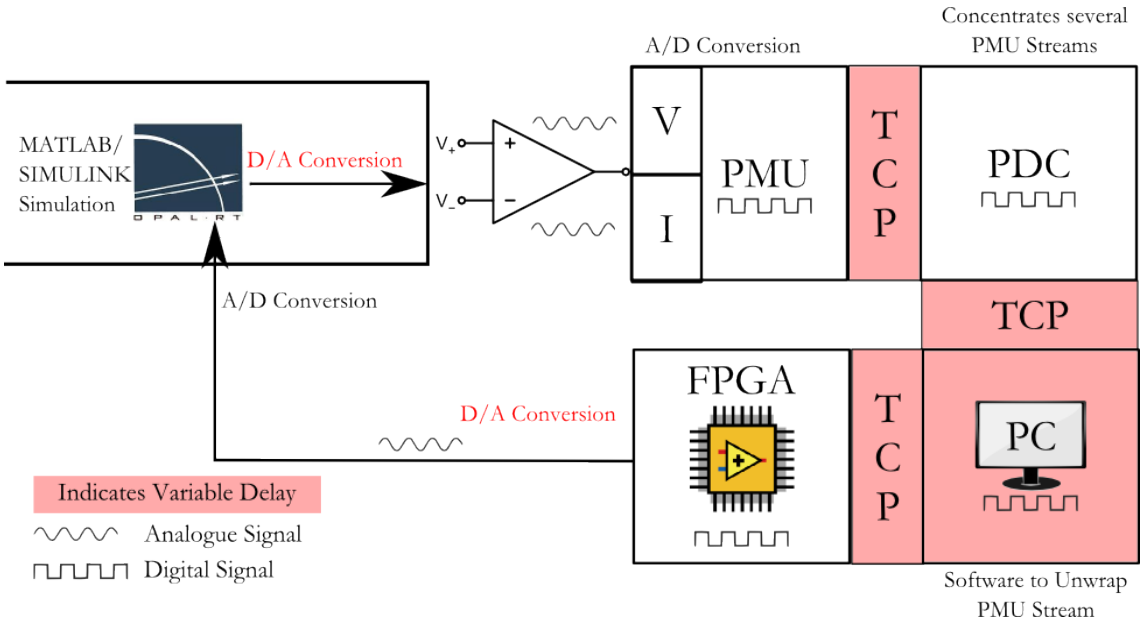

Figure 21: Real-time Data Flow during HIL Test

Due to the time delay involved in getting data to the hardware-based POD, the value of the phase correction used needed to be changed. This was done iteratively by observing the point at which the hardware POD signal was able to start damping the system's oscillations. This method is not ideal as the data path includes elements that do not run in real-time such as the network communication (TCP) and the computer used to extract data from the PMU stream.

An additional difference compared to the Simulink approach was that the gain

needed to be changed. The limits imposed by the analogue outputs of the RT simulator and also the limits of the cRIO output modules constrained the signal magnitude to within these limits at both these stages. The real-time simulator was not capable of producing an output greater than 16V in magnitude. The simplest solution was to use p.u. values of currents and voltages and to scale them up (or down) as needed. The analogue output module on the cRIO was also limited to a signal magnitude of 10V. This required scaling down the signal (or up, depending on input) so that all possible values fell within the 10V range.

The values used in the cRIO POD are listed in Table 3. The large amplitude of oscillations from the power signal are scaled down with a small gain. Also, note that the phase correction is different from the $\pi/2$ used in the SIMULINK implementation.

Table 3: POD Parameters used in cRIO

Search Frequency	ω_{CS}	$2\pi \times 0.64$
Phase Correction	α_1	2.32
Gain	-	0.00045

Due to the scaling factor used on the cRIO, the signal generated was limited to +/-10V. To account for this, the signal was multiplied by a scaling factor in SIMULINK before being fed to the SVC (see Figure 22). The scaling factor used was **1700** when using active power as an input signal. This value was determined by observing the magnitude of the damping signal generated by the SIMULINK POD and calculating the scaling factor required to bring the cRIO-POD's damping signal to the same magnitude.

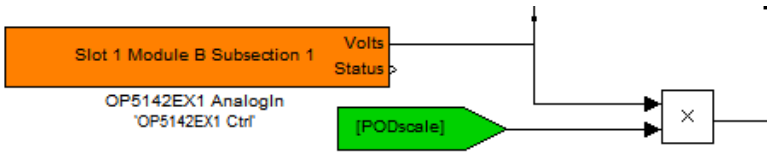

Figure 22: POD Signal Scaling in SIMULINK

The results obtained are shown in Figure 23.

Note that the time delay between the onset of the disturbance and the cRIO output is clearly visible. Damping is limited using this method, due to the fact that as the oscillation magnitude decreases, the POD reacts to noise. This signal gets added to the SVC's reference control signal and will result in a small but random valued & persistent disturbance. This can be seen towards the end of Figure 23.

Also, as expected, both the power signal and the generated damping signal show modes other than the 0.64Hz inter-area mode. This behaviour is identical to that observed in the off-line simulation depicted in Figure 20.

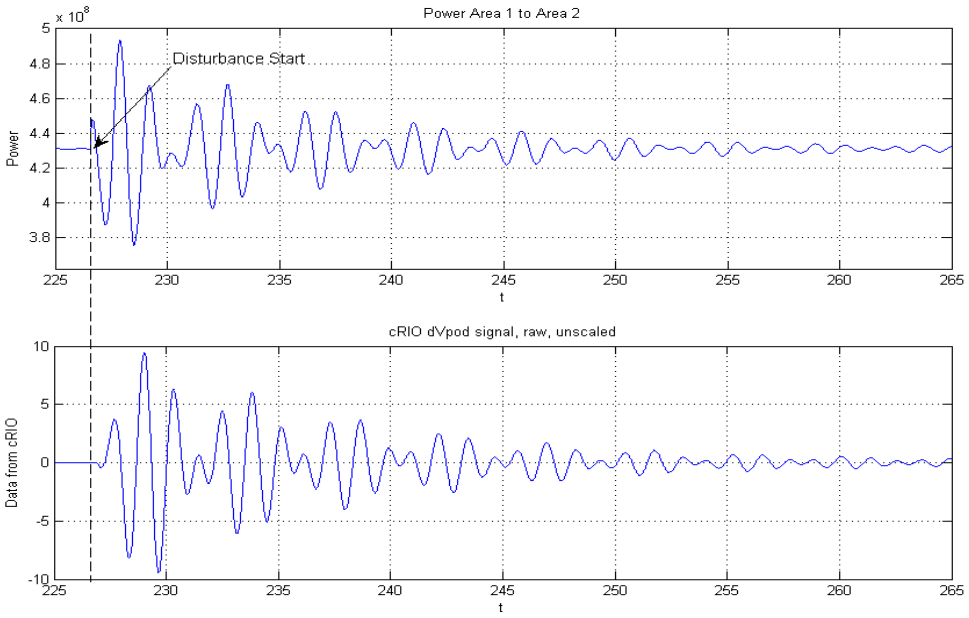

Figure 23: Damping with HIL POD (using active power)

4.3 Current Magnitude as Input

The POD in the SIMULINK model was modified to operate on the positive sequence current magnitude as an input. Since the magnitude of oscillations in the current signal is smaller in magnitude than in the power signal, the parameters of the POD algorithm were changed. The gain, in particular, needs to be much higher than the gain used with the active power signal. The phase compensation needed was computed simply by monitoring the phase angle between the current and the original power signal. The gain required was determined iteratively, by searching for when an increased value of gain would lead to instability instead of damping. The parameters determined from this exercise are listed in Table 4.

Table 4: POD Parameters used in SIMULINK (Posotive Sequence Current as Input)

Search Frequency	ω_{CS}	$2\pi \times 0.64$
Phase Correction	α_1	$\frac{\pi}{2}$
Gain	-	75

To verify the working of the modified POD and to generate a reference scenario, an off-line simulation was conducted. As with the case of active power, all stabilization elements were removed from the system except the POD. In this case, the effect that the POD alone has on system performance is studied. The results are shown in Figure 24.

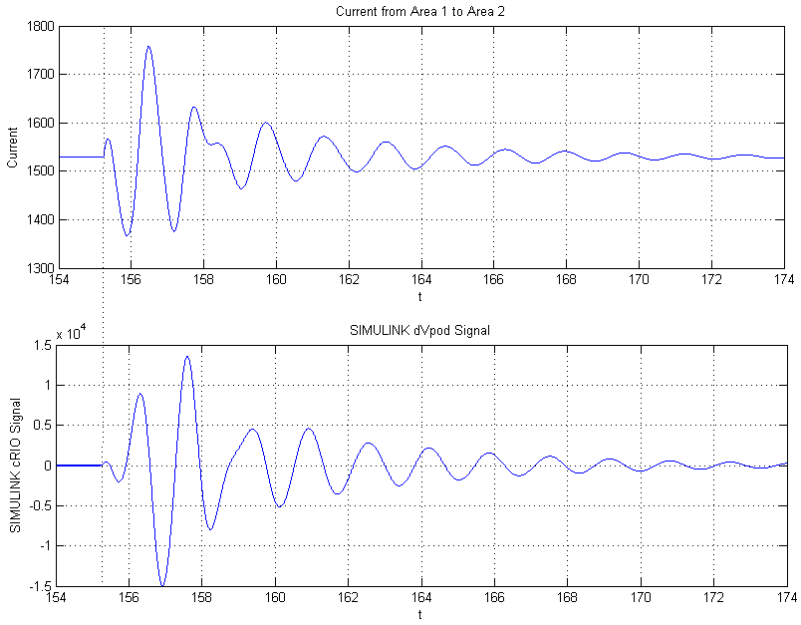

Figure 24: Ideal damping scenario with Current Magnitude as input (SIMULINK)

It is interesting to note that the local modes are not as prominent when using the current signal as they are when using active power. As with the previous case of active power, these exact parameters cannot be used with the cRIO POD. Modifications were needed in the gain and phase parameters. The phase correction was determined iteratively. The gain values were selected by considering the analogue limits imposed by the hardware used. The parameters used in the cRIO POD are listed in Table 4.3.

Table 5: POD Parameters used on cRIO (Positive Sequence Current as Input)

Search Frequency	ω_{CS}	$2\pi \times 0.64$
Phase Correction	α_1	2.35
Gain	-	0.025

As explained previously, the signal generated by the cRIO was limited to 10V in magnitude. Before applying this signal as a supplementary input to the SVC model in the real-time simulation, it was scaled up. Since the oscillation magnitude in the

current signal was smaller than in the active power signal, the cRIO's signal was scaled up by a factor of **1880** (see Figure 22). Note that though the ideal phase shift required with both the active power and current is $\pi/2$, the actual values used on both the cRIO implementation are almost identical (2.32 & 2.35) but neither is exactly equal to $\pi/2$. This can be justified on the basis of time delay. The results of this test are shown in Figure 25. From Figure 25, it is also interesting to note that the overshoot of the response is lower than in the case when active power was used as an input. Also, the perturbations introduced by the noise in the controller's input signal are smaller (with current input) and allow for the system to reach a more stable point than possible when using active power as the input. This could be justified in one way by thinking of the active power as being the product of current and voltage and hence the product of two noise sources. Current, on the other hand is only one noise source and produces a more stable signal with less noise.

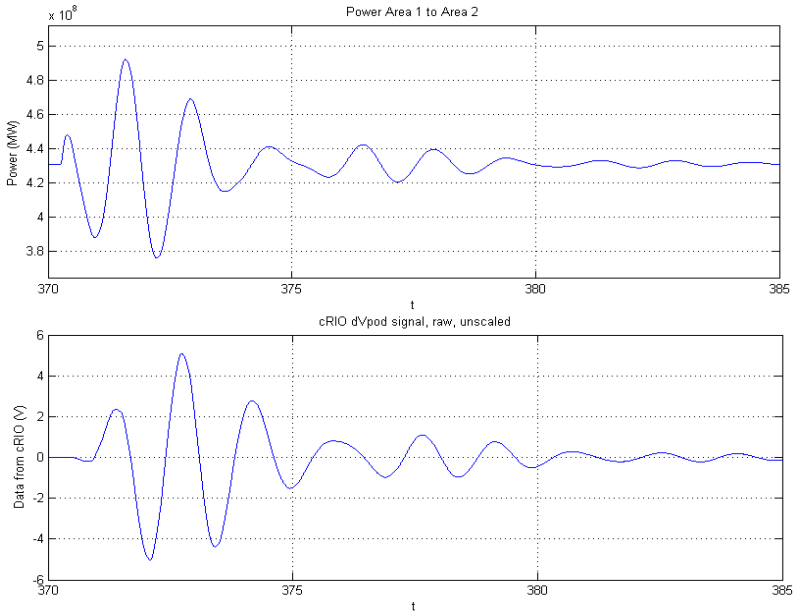

Figure 25: HIL Test : Damping Achieved with cRIO

4.4 Other Inputs

Using data from a PMU stream as a controller input provides flexibility in selecting an input signal. Even greater flexibility can be achieved by using data from another location in the power network. Besides active power and current magnitude, two other inputs were tested with the developed controller, viz. Voltage Phasor Magnitude Difference and Voltage Phasor Angle Difference. While active power and current magnitude require only one PMU for data, computing a difference requires a second PMU's data. The second PMU's measurement point was the end of Area 2. Both measurement points are marked in Figure 12. Both tests were carried out solely with the hardware controller to illustrate its flexibility. No comparisons are

made with SIMULINK simulations as these were not carried out.

4.4.1 Voltage Phasor Angle Difference

The voltage angle difference between the two areas was selected as a possible input for the POD. In the off-line simulations performed, some observations were made.

The first major problem was that angle measurements are limited to 0-2π radians or $-\pi$ to $+\pi$ radians. When reporting the value of voltage angle difference, values outside this range are wrapped around to the other end. This causes sudden jumps in the measured angle difference. A simple solution to this problem is to unwrap the phase, ensuring continuity when such jumps occur.
A more serious problem was with the amplitude of oscillations that could be observed in the voltage angle difference signal. With an instantaneous oscillation amplitude of 30MW in the active power signal, the oscillation magnitude in the voltage angle difference signal was 0.002 radians (see Figure 26). Amplifying such a small signal presents new problems such as amplification of noise.

However, contrary to what was expected, the performance of the controller using voltage angle difference as a damping input was better than expected. It must be noted that the gains required here were significantly higher than in the other cases, however, this did not lead to amplification of noise. Since the signal generated by the cRIO POD was very small in magnitude, the gain used in Simulink, before inserting this signal into the SVC, was very large. Compared to the 1700 used with active power, the gain (scaling factor) used in this case was **4500**. The other parameters used were determined iteratively and are listed in Table 6

Table 6: POD Parameters used on cRIO (Voltage Angle Phasor Difference as Input)

Search Frequency	ω_{CS}	$2\pi \times 0.64$
Phase Correction	α_1	0.4
Gain	-	75

From Figure 27, it is obvious that the dominant mode is the 0.64Hz inter-area mode. Other modes are not as prominent as when using the active power and current signals as inputs and are barely visible. From a preliminary observation, this could be one of the most significant reasons for the performance with this input being better than all the others. This confirms the mathematical analysis presented in [12]. Also evident from Figure 27 is that the power signal (upper plot) moves steadily towards stability and does not show the effects of noise visible when using other inputs. When allowed to run for extended periods, the performance of the controller in this case produces a steady state very close to that achievable when

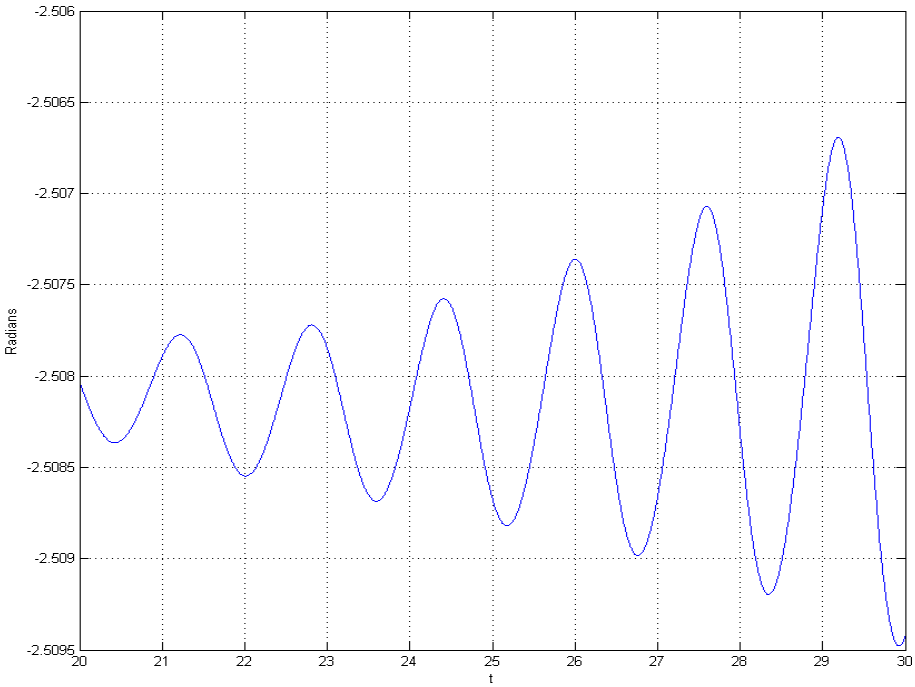

Figure 26: Oscillation observable in Voltage Angle Difference Signal

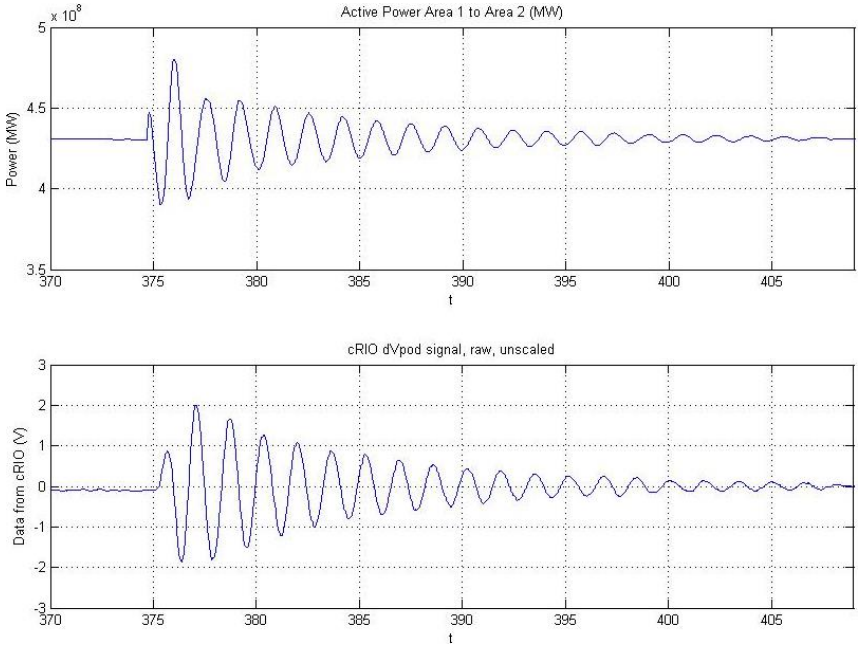

Figure 27: Damping Achieved using Voltage Angle Difference as HIL-POD Input

using a POD running entirely in Simulink.

As evidence that the cRIO-based POD was actually able to generate a damping signal in real-time, Figure 28 shows the voltage output of the cRIO-POD, as measured on an oscilloscope. This is the damping signal generated when using the

voltage angle difference as the POD's input. This damping signal is fed back to the SVC model, running in the real-time simulator, as a supplementary control signal. It can be clearly observed that the damping signal is decreasing in magnitude with time, indicating that the cRIO-based POD is indeed effective at replicating the functional behaviour of the simulated POD. Also, identical to the results presented previously, the dominant mode is the 0.64Hz inter-area mode.

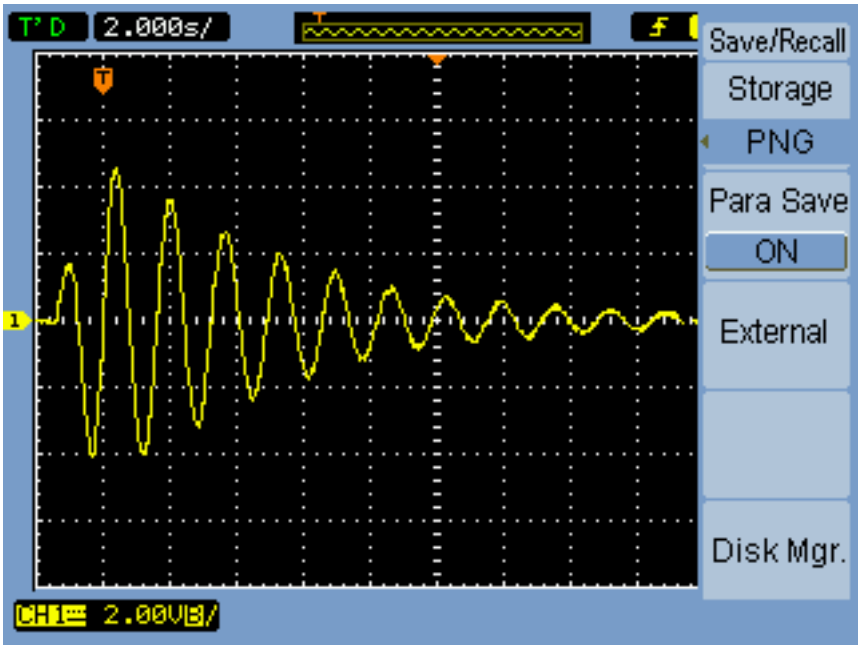

Figure 28: Damping Signal Generated by cRIO as Measured on an Oscilloscope

4.4.2 Voltage Magnitude Difference

The magnitude of the voltage difference between Area 1 and Area 2 could also be used as a damping input to the POD. Since this will also require a second set of PMU measurements, two measurement points were selected. These are identical to those used in the previous case and are as marked in Figure 12. Similar to the previous section, the parameters used on the cRIO are listed in Table 7. The value of the gain used in SIMULUNK, to scale the cRIO generated signal up was **2000**. This is smaller than the value used with the voltage angle difference as input.

The results obtained with this input are shown in Figure 29. It can be observed that the dominant mode has a frequency of 0.64Hz and similar to the previous case with Voltage Angle difference, other modes are not very visible. Also, the effect of noise on the signal is not as great as when using active power as an input. As with the previous case, the active power plot (upper plot) is able to move towards a steady state where the effects of noise are not very prominent. The one significant difference between this and the previous case (with Voltage Angle Difference) is the

Table 7: POD Parameters used on cRIO (Voltage Magnitude Difference as Input)

Search Frequency	ω_{CS}	$2\pi \times 0.64$
Phase Correction	α_1	π
Gain	-	1

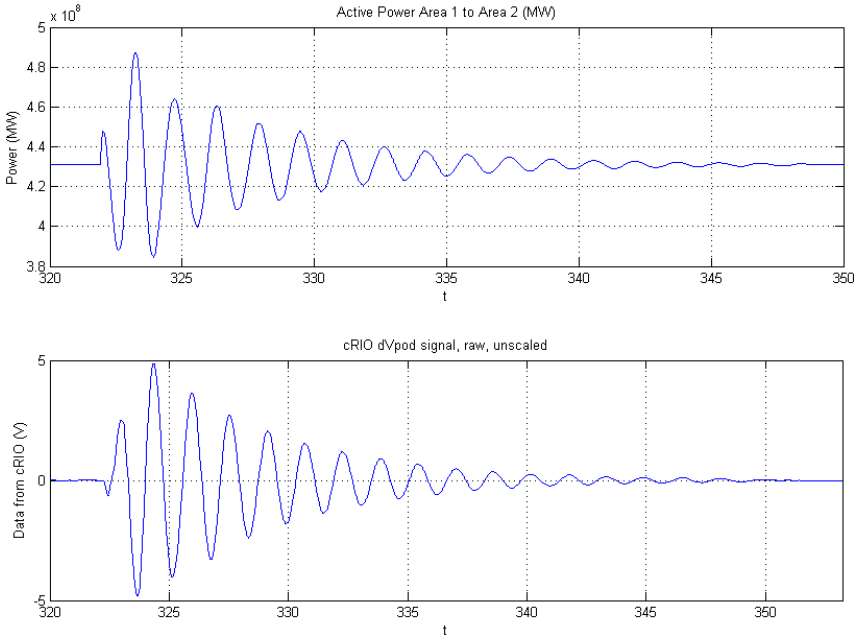

Figure 29: Damping Achieved using Voltage Magnitude Difference as HIL-POD Input

magnitude of the overshoot. This is visible in the response of the POD algorithm whose peak is 4.9V in this case versus 2.03V in the case with Voltage Angle Difference. Further analysis is required to understand this behaviour.

4.5 Comparison between HIL Tests Using Active Power & Current

Off-line analysis indicated that though the active power signal could achieve damping, it led to multiple modes appearing in the power signal. The current magnitude provided damping performance comparable to that achieved with the active power as an input. However, using the current as an input produced secondary modes that were smaller in magnitude. With HIL (Hardware in the loop) tests, this was confirmed. Better damping was achieved using current magnitude as a POD input compared to the active power. The time required for oscillation magnitude to be limited to 5% of the final value was less when using the current magnitude as an

input. Also, the oscillations magnitude in the damped signal were smaller in magnitude. Most importantly, although off-line simulations indicated that both active power and current magnitude were able to stabilise the system, with no oscillations visible in the active power signal, this was not the case when the HIL results were analysed. This behaviour can be attributed to the effect of noise. When using active power as an input, noise in both the analogue current and analogue voltage measurements affect the computed power signal. The current magnitude contains only one noise source. This was verified by computing the SINAD (signal to noise and distortion) ratios in both cases (active power & current). The computations were performed in LabVIEW by taking the last 100 data points from the POD's response plot. The SINAD is generally defined as [26] :

$$SINAD = \frac{P_{Signal} + P_{Noise} + P_{Distortion}}{P_{Noise} + P_{Distortion}} \tag{4}$$

From this definition, a higher value of the SINAD indicates less noise. The results of this are below.

Input	SINAD (dB)
Active Power	25.5643
Current	34.7579

A comparison of the controller's performance with the four parameters used (active power, current magnitude, voltage magnitude difference and voltage angle difference) is presented in Appendix A.3.

4.6 Tests with different FPGA Cycle Times

The OPAL RT simulator reads data at its analogue input terminals every 50μs. Data is generated by the cRIO at this same interval. Precise synchronization is not required because the cRIO will hold the analogue output value until the next value is generated. The cycle time (loop rate) of the POD algorithm on the FPGA could be changed by a simple software setting. Operation of the algorithm was checked for three different values of the loop rate: (i) 15μs (ii) 25μs and (iii) 50μs.

The result with a 15μs loop rate is shown in Figure 30. Since the FPGA update rate and the OPAL RT read intervals are not synchronized nor multiples of each other, transitional values are repeatedly read which distorts the signal. The performance of such a controller cannot be directly compared with that of a simulation as the time step sizes are not the same.

Figure 31 shows the result with a 25μs loop rate. Here, signal distortion is not as large as with the 15μs step size. This is because 25μs is a multiple the OPAL RT sampling interval (50μs). However, signal accuracy will not exceed that of the 50μs step size as every second value is lost. The fact that this figure shows the oscillation magnitude is simply due to poor tuning. A controller running at this time step can

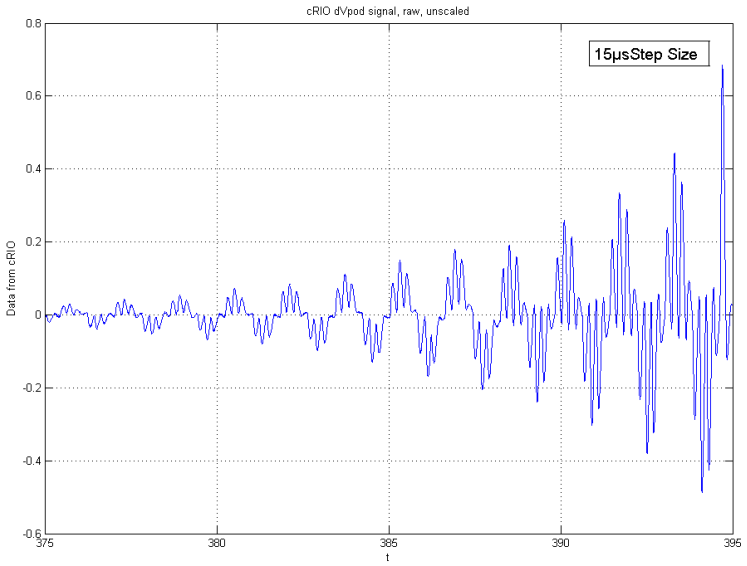

Figure 30: Operation with 15μs FPGA step size

adequately perform by proper tuning the algorithm's parameters. This argument is valid for time step sizes that are multiples of the simulation time step. A hardware controller running at a 100μs time step will also function properly.

Figure 32 shows operation with a step size of 50μs. Here, damping of the power

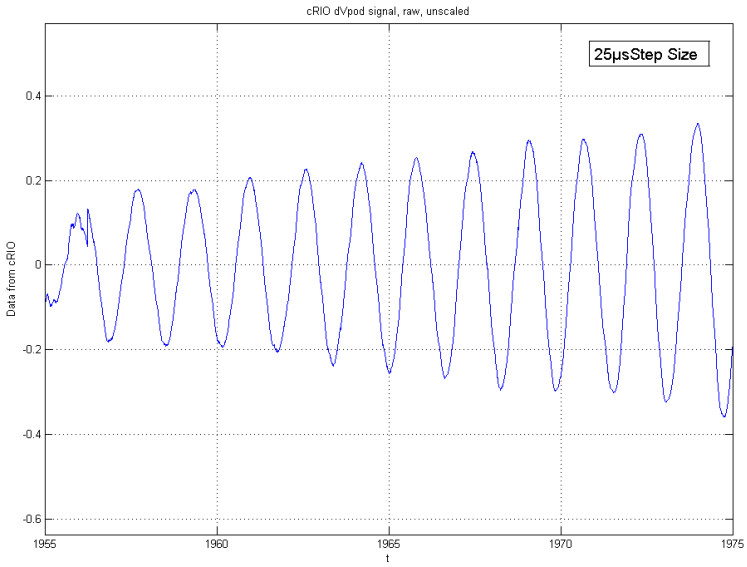

Figure 31: Operation with 25μs FPGA step size

signal is obtained which is why the signal magnitude progressively decreases. The fact that the controller is able to damp the oscillations in this case is not due to the fact that the simulation and the hardware controller are running at the same time step.

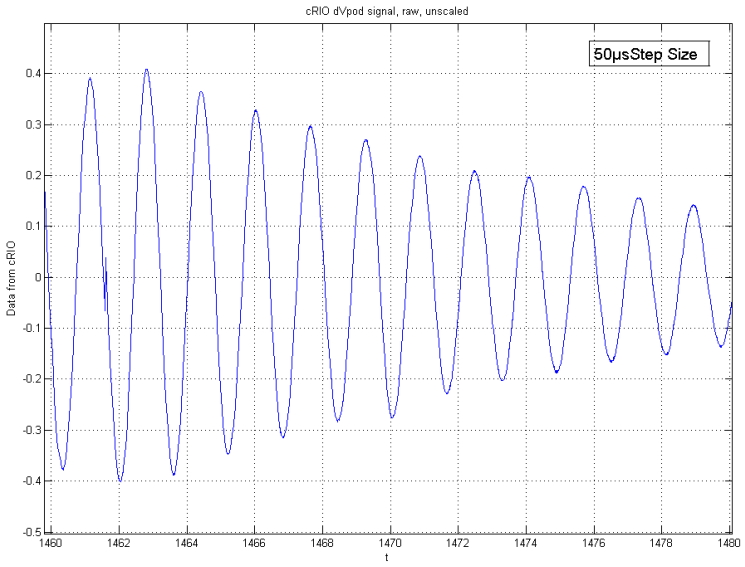

Figure 32: Operation with 50μs FPGA step size

4.7 Controller Time Delay Analysis

The response of the hardware based POD differs from that of the simulated POD in several ways. Among these are noise, harmonic content and scale. The time taken for the signal to travel from the simulator to the POD is also significant. For each step that the signal must pass through, a time delay is introduced. The delay introduced at each stage can be deterministic or stochastic.

The real-time data path is shown in Figure 21.

Certain sections of this process have fixed signal propagation delays, for example:

- D/A and A/D conversion at the OPAL simulator interfaces
- Signal Amplifiers
- PMU (Sampling analogue voltages and generating a digital stream)
- PDC combining PMU streams and generating output stream
- FPGA execution speed (50μs, known and constant)

The data path also includes several stages where data is transferred over a conventional TCP/IP network. The network delay forms a variable component of the total delay. Another significant source of delay is the computer used to unwrap the PMU data stream and extract the raw measurement data. Depending on the software used, this delay can vary from a few milliseconds to several tens of milliseconds.

The reason for this variable delay is due to the non-real time design of a typical PC OS.

In this particular controller design, LabVIEW code is used to stream data extracted from the PMU stream to the cRIO POD. This consists of a loop taking extracted data and then publishing it to a NI Shared Variable[27]. The function calls to the shared variable engine take different amounts of time to execute depending on the size of the data being published. The design of the shared variable also includes a 10msec delay [27] in addition to the network transport delay. The sum total of the delay on the workstation computer thus consists of three factors:

1. Time taken to extract data from PMU stream
2. Time taken for LabVIEW code to process extracted data
3. Time taken to publish data to the communication network (shared variable)

Since individual delays cannot be measured, the total delay for the entire communication process was measured. This was done by logging data at the OPAL RT simulator. The measurement point used was the application of a 5% voltage disturbance. This produced a sudden jump in the active power and POD damping signals. For a detailed explanation of this process, see Appendix A.2

To measure the total delay introduced by the HIL setup, timestamped data would be required at all steps of the process. However, since the data path includes a workstation PC which is not running in real-time and has no GPS synchronisation, time-stamping data passing through the PC would not be possible. The PMU's and the PDC both record time stamped data. Unfortunately, these were the only two points in the entire loop where data was time stamped. This posed a problem to accurately measure the signal delay introduced by each component of the test set-up.

Soultion

A method was devised to get a reasonable sense of the time taken by data to flow from the simulator, through the entire HIL set-up and back. This involved sending a periodic 'heartbeat' signal through the entire set-up (see Figure 34). This signal was sent as an analogue output from the OPAL RT simulator and then travelled through the entire HIL setup before being recorded at the input of the OPAL RT simulator. Data at the OPAL RT simulator could be captured, both at the inputs and the outputs. By measuring the delay between the start of the pulse at the output of the simulator and the start of the same pulse at the input, the total delay could be estimated. This method allows for estimation of the total signal delay without the need for timestamped data.

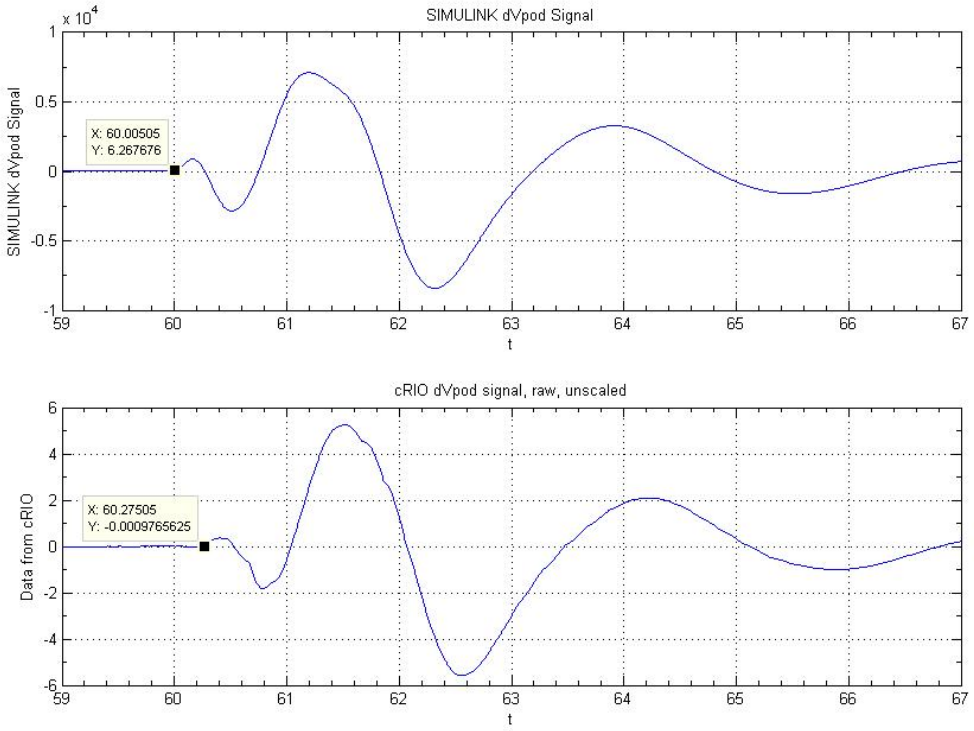

Figure 33: Measuring the total signal delay

The PMU posed a problem as the heartbeat signal could not be used as is. The PMU would detect the signal an one (or more) parts of a three phase signal and generate a phasor accordingly. The phasor generated would not resemble the input to the PMU. Further, when this phasor is received by the POD, its response would further modify the signal. As a result of these two processes, the signal read at the inputs of the OPAL RT simulator would bear little resemblance to the original signal.

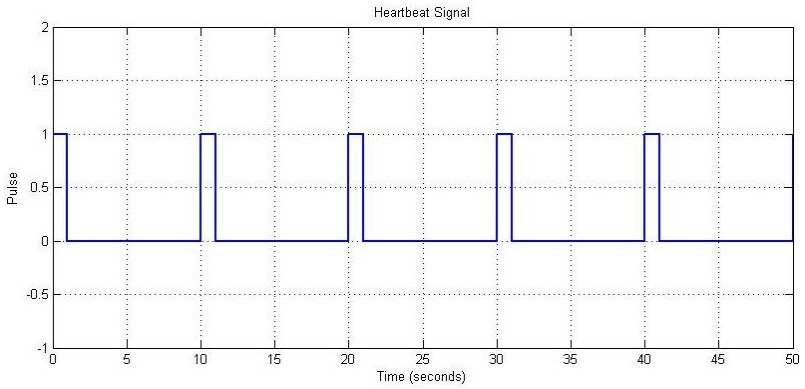

Figure 34: Periodic 'Heartbeat' signal

This method was modified so that the 'heartbeat' signal was added as a disturbance in one of the machines in the two area system. If the voltage reference set point of the generator excitation system is changed, the generator active power output changes accordingly. The heartbeat signal was added as a 10% voltage reference change to the excitation system reference of machine M1 (in Figure 11). The disturbance in the active power would produce a response from the POD running in SIMULINK (i.e. on the simulator). Since the HIL implementation of the POD is identical to that in SIMULINK, the POD running on the cRIO would also produce an identical response. A delay will exist in the starting points of these two responses. This difference will be equal to the signal propagation delay through the entire HIL network.

Figure 33 shows an illustration of the time delay between the onset of a damping signal on the simulated POD (in SIMULINK) and that from the cRIO. This was for the onset of one of the 'heartbeat' pulses described above. Multiple pulses (8 in total) were applied and the delays calculated for each of them. This data is listed in Table 8. Maximum and minimum delay values are indicated in **bold**.

Table 8: Signal Propagation Delay Calculation

Pulse Number	**Simulated POD Response Start** (s)	**cRIO POD Response Start** (s)	**Time Delay** (ms)
1	60.01	60.275	265
2	70.04	70.35	310
3	80.02	80.265	245
4	90.04	90.37	330
5	100.04	100.35	310
6	110.03	110.285	255
7	120.05	120.26	**210**
8	130.02	130.36	**340**

From this data, it is obvious that the start point of each of the pulses was almost exactly 10 seconds apart. The small differences in the start times of the SIMULINK POD's response is due to measurement errors. However, the start point of the cRIO POD's response differs significantly from one pulse to the next. The difference between the cRIO POD's start time and the start time of the SIMULINK POD's response gives a good indication of the total time delay taken for signals to travel through the entire HIL setup. The average signal propagation delay was calculated to be **283.1ms** based on this data. The standard deviation was **0.046s**. The stan-

dard deviation is not zero. This can be justified by the fact that the workstation computer which performs the function of unwrapping the PMU data and passing it to LabVIEW, is not running a real-time operating system. The delay introduced by this computer is variable. The actual delay depends on the processor load as the operating system is designed to be multi-tasking. There is also the fact that TCP network communication introduces a variable delay however this variation is not as significant as that introduced by the computer.

This amount of delay requires a phase shift in the controllers output. As mentioned before, this phase shift has to be calculated using the 0.64Hz oscillation (or damping signal). Here, 0.64Hz, or a cycle time of 1.5625s will correspond to a phase of 2π. Using this, a delay of 283.1ms corresponds to a phase shift of 0.3623π radians. This is merely an averaged value and the actual phase compensation was closer to $\pi/2$ in most cases.

With the same magnitude and disturbance introduced in the test system, it can be observed that the hardware POD (on the cRIO) takes significantly longer than the simulated POD. The signal time delay of 283ms is the chief reason for this. This time delay corresponds to a little more than 14 cycles (at 50Hz) before damping action is initiated by the controller. During this time, the disturbance magnitude (and hence the oscillation magnitude) increases continuously. The simulated POD (in SIMULINK) on the other hand commenced damping action as soon as the disturbance is applied. The hardware-POD has to damp oscillations that are significantly higher in magnitude than the simulated POD has to.

See Appendix A for an illustration and a more detailed explanation.

4.8 Other Issues During Testing

Besides the problems faced during the development of the cRIO POD code, numerous other problems were faced. Three, along with their solutions are listed here.

4.8.1 Noise

The setup described above was initially planned to be used without any signal amplification. Low-level signals taken directly from the real-time simulator would be wired to the PMU's. However, the levels of noise in these signals prevented their use. This problem was further compounded by the fact that the PMU input modules were rated for 0-300V and 0-10A. Applying a voltage signal with a 10V peak amplitude to a measuring device rated for a peak of 300V would result in significant measurement errors (see Figure 35). A similar argument can be made for the current measurement inputs. When both inputs were combined to generate a power signal, the noise level was further increased. This prohibited the controller from operating. An argument can be made for using input modules with higher resolution and better

noise filtering. While this is true, typical power system instrument transformers (CT's[13] and VT's[14]) produce secondary signals that are of the 0-300V range for voltage and between 0-5A (or greater) for current. The argument for using input modules with these ranges is thus justified.

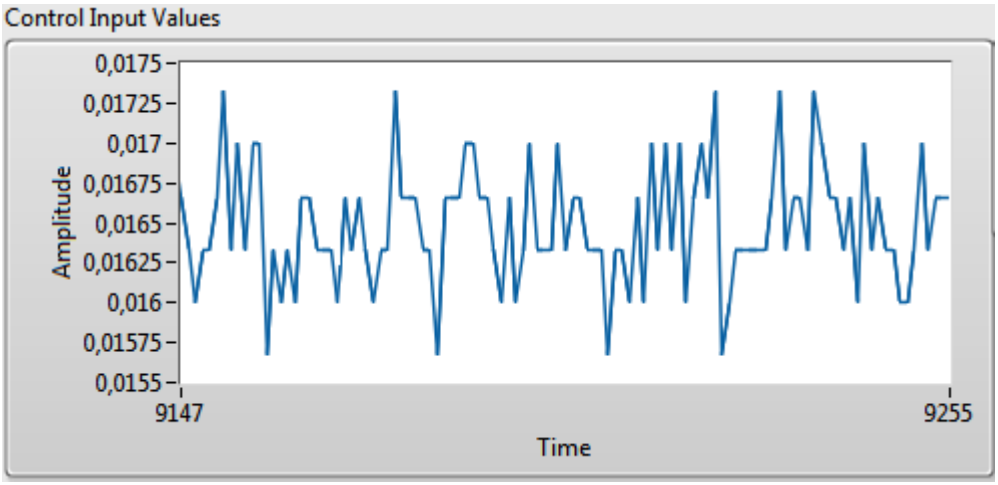

Figure 35: Noise Visible in Low-Level Signal With Average Value of 0V

4.8.2 Necessity for Signal Amplification

The conclusion of the argument made above is that signal amplification is necessary, for both the currents and the voltages. Using analogue signal amplifiers greatly improved the signal to noise ratio. However, this did not completely eliminate noise. As can be seen in Figure 36, the signal generated is not completely smooth and still includes effects of noise. Using signal amplifiers also adds to the overall signal delay however this component of the delay is constant and deterministic. This requirement of signal amplification arose as the outputs of the Real-Time simulator were low level signals. When a PMU is used in a real power system and is connected to the secondary of instrument transformers (CT's and PT's) signal amplification should not be necessary.

4.8.3 Variable Time Delays

The variable nature of the time delay was the most significant problem faced during the testing of the HIL controller. A constant time delay can be measured and compensated for using an appropriate phase shift. A delay that is variable can neither be measured nor compensated for effectively. Determining the effectiveness of the controller using different inputs became a tedious task as comparisons were not

[13] Current Transformer
[14] Voltage/Potential Transformer

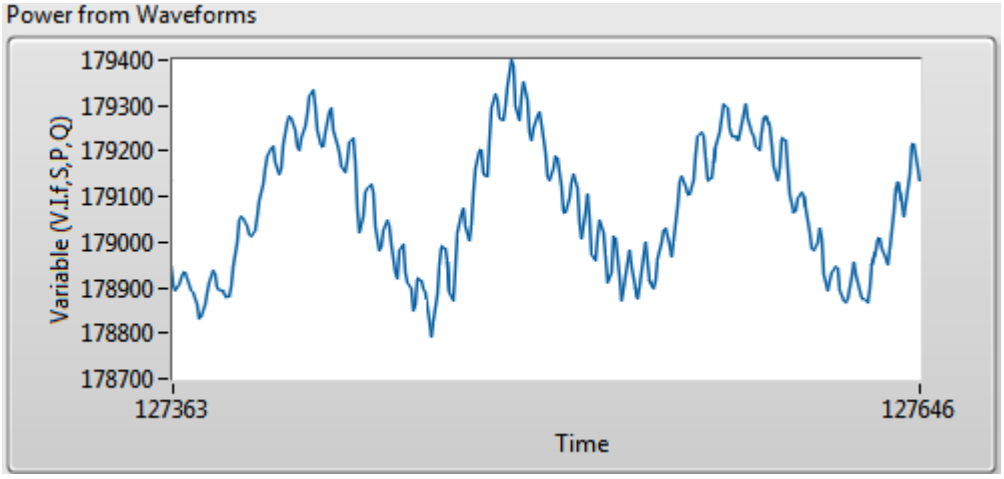

Figure 36: Power Signal Generated Using Amplified Data

straightforward or direct. A higher time delay would reduce the damping effectiveness of an input and contradict simulation results. To counteract this issue to some extent (although not completely) all inputs were tested with the controller during a single run. This was done under the assumption that the delay would not suffer large excursions during each experiment.

5 Summary & Conclusion

This thesis has analysed a Phasor based oscillation damper originally developed for SIMULINK. It has also successfully replicated the behaviour of the SIMULINK design using a hardware based (NI cRIO) prototype which receives input data via synchrophasors. This hardware based prototype was also tested using a real-time simulator running an unstable two-area network and was capable of stabilising the power system. The prototype developed also served to demonstrate that the damping algorithm is effective in a real-world scenario with noise and communication delays. Further, to demonstrate the flexibility of the synchrophasor based design, the prototype's operation was demonstrated using various inputs such as active power, current magnitude, voltage magnitude difference and voltage angle difference.

The phasor based algorithm also adds to flexibility as the only network-specific input needed is the oscillation frequency. Such a system can be easily deployed in a real-life power system in conjunction with a FACTS device.

Future Work

The flexibility that synchrophasor data brings to the POD application opens tremendous scope for improvement of this design.
The prototype demonstrated in this thesis relied on manual (operator) input to select an input signal. This process can be automated so as to select a signal with the highest observability of a given oscillation mode. The required computations for this can be performed locally, on the micro-controller. Further automation is also required with the phase compensation section of the Phasor POD algorithm. By using time stamped data at all states of the real-time data flow, the micro-controller can be used to monitor the signal propagation delay in real-time. Once this value of delay is known, it can be used to calculate the required phase compensation for a given input signal. This will also have the advantage of being able to switch to an alternate damping signal when the delay crosses unacceptable thresholds.

The present design incorporates a workstation computer in the loop which does not run a real-time operating system. The only purpose of this computer is to extract raw measurement data from the PMU synchrophasor stream. This can be performed more effectively in real-time on the micro-controller running the POD algorithm.

The most substantial cost of the present design are the hardware and software platforms. This prototype has used software and hardware from National Instruments, both of which are proprietary platforms. A cheaper and more standardized platform can be designed by porting the POD code to an open hardware platform such as the Arduino or the Raspberry PI.

References

[1] Rogers, G. Introduction, in, Power System Oscillations, Springer Science+Business Media LLC, New York, 2006 pp. 2-4.

[2] North American Electric Reliability Council, "Review of selected 1996 Electric System Disturbances in North America," August 2002. [Online]. Available: http://www.nerc.com/pa/rrm/ea/System%20Disturbance%20Reports20DL/1996SystemDisturbance.pdf

[3] Dmello, F. P. &. Concordia, C. "Concepts of Synchronous Machine Stability as Affected by Excitation Control," IEEE TRANSACTIONS ON POWER APPARATUS AND SYSTEMS, vol. PAS 88, no. 4, 1969.

[4] Larsen E. &. Swann D. A. , "APPLYING POWER SYSTEM STABILIZERS Part I II & III", IEEE Transactions on Power Apparatus and Systems, vol. PAS 100, no. 6, p. 3017, 1981.

[5] Aboul-ELa M. E., Sallam A. A., McCalley J. D., and Fouad A. A., "Damping Controller Design for Power System Oscillations Using Global Signals", IEEE trans. on Power Systems, Vol. 11, No. 2, May 1996, pp. 767-773

[6] Ängquist, Lennart and Gama Carlos C. G. , "Damping Algorithm based on Phasor Estimation," in Power Engineering Society Winter Meeting, 2001. IEEE, Volume 3, pp. 1160 - 1165

[7] M. Shoaib Almas and L. Vanfretti, "Implementation of Conventional PSS and Phasor Based POD for Power Stabilizing Controls for Real-Time Simulation", IEEE IES IECON14, 29 Oct ? 1 Nov, 2014, Dallas, USA.

[8] Larsen, E. V., and Chow, E. H., General Electric Company, NY, "Application of Static VAR Systems for System Dynamic Performance", 1987 IEEE pp. 43-46

[9] Vanfretti, Luigi and Aarstrand, Vemund H. and Almas, M. Shoaib and Perić, Vedran S. and Gjerde, Jan O. , "A Software Development Toolkit for Real-Time Synchrophasor Applications", IEEE PES Grenoble PowerTech, 2013

[10] Klein, M. Rogers, G. J. and Kundur P. "A fundamental study of inter-area oscillations in Power Systems," IEEE Trans, PWRS, no. 6, pp. 914-921, 1991.

[11] Rouco, L. "Eigenvalue-based Methods for Analysis and Control of Power System Oscillations," in Power System Dynamics Stabilisation (Digest No. 1998/196 and 1998/278), IEE Colloquium on, Coventry, 1998 pp. 3/1 - 3/6

[12] L. Vanfretti, Y. Chompoobutrgool, and J.H. Chow, "Chapter 10: Inter-Area Mode Analysis for Large Power Systems using Synchrophasor Data", Book Chapter, in Coherency and Model Reduction of Large Power Systems, Joe H. Chow (Ed.), Springer, 2013. Available Online at http://www.springer.com/energy/systems%2C+storage+and+harvesting/book/978-1-4614-1802-3

[13] Agee J.C. , Patterson S. ,Beaulieu R., Coultes M.,Grondin R. , Kamwa I., Trudel G., Godhwani A. , Bérubé R. , Hajagos L. , Malik O. , Murdoch A., Boukarim G. , Taborda J. , and Thornton-Jones R. "IEEE tutorial course power system stabilization via excitation control". Technical report, June 2007.

[14] Chaudhuri, N. S. Majumder R. and Chaudhuri B. , "Interaction between conventional and adaptive phasor power oscillation damping controllers," in Power and Energy Society General Meeting, 2010 IEEE Minneapolis, MN, 2012 pp. 1-7

[15] Kamwa I. (Hydro-Quebec) "Performance of Three PSS for Interarea Oscillations" Available Online at : http://www.mathworks.se/help/physmod/sps/examples_v2/performance-of-three-pss-for-interarea-oscillations.html

[16] IEEE Standard C37.118.1-2011 "IEEE Standard for Synchrophasor Measurements for Power Systems" pp. 5

[17] IEEE Standard C37.118.2-2011 "IEEE Standard for Synchrophasor Measurements for Power Systems" pp. 10

[18] "IEEE Standard for Synchrophasors for Power Systems", 1995 pp. 4-6

[19] Martin, K. E. "Synchrophasors in the IEEE C37.118 and IEC 61850" in Critical Infrastructure (CRIS), 2010 5th International Conference on, Sep 2010 pp. 1-8

[20] C.F. Magnus Danielson, L. Vanfretti, Y. Choompoobutrgool, and M. Shoaib Almas, "Analysis of Communication Network Challenges for Real-Time Synchrophasor-Based Wide-Area Control Applications", 2013 IREP Symposium ? Bulk Power System Dynamics and Control ? IX (IREP), August 25-30, 2013, Rethymnon, Crete, Greece.

[21] M.S. Almas, M. Baudette, L. Vanfretti, S. Løvlund and J.O. Gjerde, "Synchrophasor Network, Laboratory and Software Applications developed in the STRONg2rid project", IEEE PES General Meeting, 2014.

[22] "RT-Lab Solo - Getting Started User's Manual, Version 2.11", Centre for Intelligent Machines, McGill University, March 27, 2003 Available Online at http://www.cim.mcgill.ca/~ialab/members/usefuldoc/RT-Lab_Instructions_v2.11.pdf

[23] Vanfretti, L. Li, W. ; Bogodorova, T. ; Panciatici, P. "Unambiguous power system dynamic modeling and simulation using modelica tools" in Power and Energy Society General Meeting (PES), 2013 IEEE pp 1-5

[24] Documentation and Help Files Distributed with RT LAB OPAL-RT copyright: © 2007 Opal-RT Technologies Inc. All rights reserved for all countries

[25] "FPGA Control on Compact RIO Sample Project Documentation", Available Online at http://www.ni.com/white-paper/14137/en/

[26] "Analogue.com Understand SINAD, ENOB, SNR, THD, THD + N, and SFDR so You Don't Get Lost in the Noise Floor MT - 003 Tutorial" , P. Web Resource. Online Document. Last Updated on 10.08.2009. Retrieved On 24.03.2014. Available At: http://www.analog.com/static/imported-files/tutorials/MT-003.pdf

[27] National Instruments, "Sending Shared Variable Data over the Network Immediately," National Instuments, P. Web Resource. Online Document. Last Updated June 2011. Retreived On 13.03.2014. Available At: http://zone.ni.com/reference/en-XX/help/371361H-01/lvconcepts/sv_flush/

[28] "SimPowerSystems User's Guide" , Available online at: http://www.mathworks.com/help/

[29] "eMEGASIM Power Grid Real-Time Digital HArdware in the Loop Simulator" Available Online at : http://www.opal-rt.com/

[30] LabVIEW Online Help System Available Online at http://sine.ni.com/psp/app/doc/p/id/psp-357

[31] LabVIEW Online Documentation 'What is Compact RIO' available online at http://www.ni.com/compactrio/whatis/

[32] Wikipedia Entry for "LabVIEW" Available online at https://en.wikipedia.org/wiki/LabVIEW

[33] LabVIEW Course Manuals - Core 1, Real Time, FPGA, August 2012 Editions, Copyright 2012 National Instruments

[34] "Operating Instructions and Specifications Compact RIO NI cRIO-9081/9082", National Instruments, Available Online at http://www.ni.com/pdf/manuals/375714a.pdf

[35] "Operating Instructions and Specifications Compact RIO NI cRIO-9075/9076", National Instruments, Available Online at http://www.ni.com/pdf/manuals/375650b.pdf

A Additional Data Plots

A.1 Damping Performance with PSSs at Each Machine

Figure B1 shows the damping response with a PSS installed at each of the four machines in the Two Area Test System[15]. The disturbance is a 5% change in the voltage reference, for 200ms applied at Machine M1 (See Figure 8) in the Two Area Test System. This response depicts the power transfer from Area 1 to Area 2. The only damping elements were the PSS's attached to each machine. No Phasor POD or SVC are used in this model. This response displays two important characteristics. One, the overshoot at the point of application of the disturbance is the smallest of all the tests performed. Two, the time required for the response to settle to within 1% of the final value (433MW) is under 5s (approx 4.2s). This is the best damping performance that can be achieved with the PSS parameters used in this model.

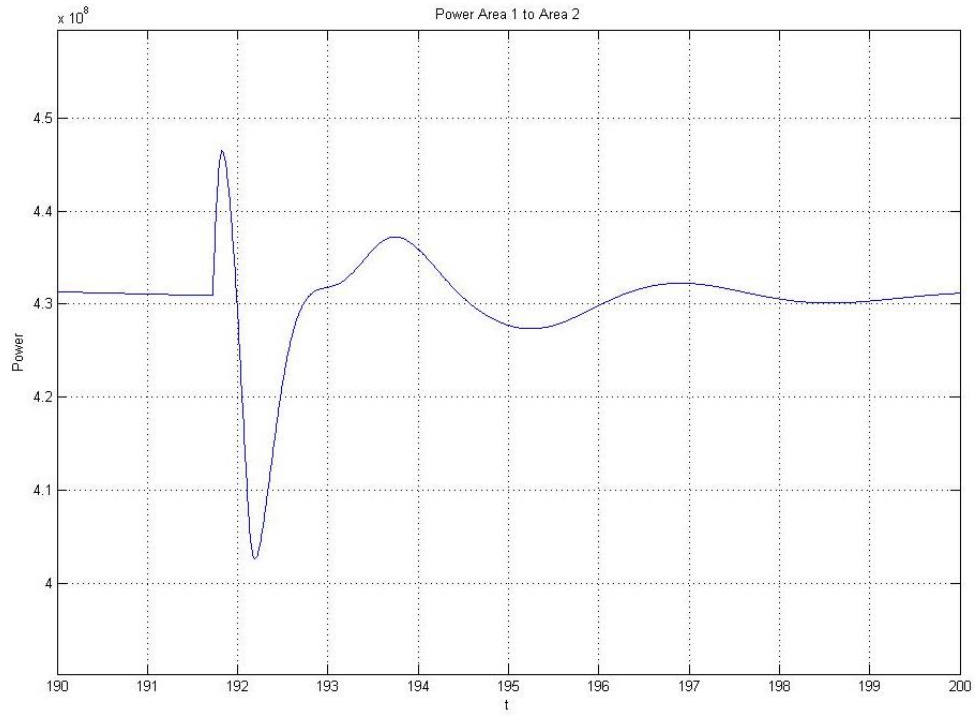

Figure B1: Damped Response to Disturbance with PSS at each machine

A.2 Time Delay in the HIL Test Set-up

Figure B2 shows (from top to bottom)

- Active Power signal with periodic disturbance
- Simulink POD's response to periodic disturbance
- HIL POD's response to periodic disturbance

This explanation follows from the section on Page 48. The disturbance in the reference voltage setting of Machine M1 causes a surge in active power flow. This is followed by a the reference being restored to the original value. The duration of the disturbance can be varied and 1s is used in this case. The disturbance pulse repeats every 10s. It can be observed that the damping elements in the system (PSS together with POD) are able to achieve significant damping in this 9s period and the system is close to stability. At the end of the 10s period, when the next disturbance pulse is applied, the system response is almost identical to the previous cycle, with a negligible increase in the disturbance magnitude.

A damping signal will be generated at two different locations. One is by the POD model running in the real-time Simulink simulation (middle plot). The other will be generated by the POD running on the cRIO, receiving data through the HIL set-up (bottom plot). Both responses will be identical in nature (not magnitude). The only difference will be that the HIL POD's response will be slightly delayed compared to the Simulink version. This delay can be measured by comparing the starting points of the two responses.

The HIL set-up is not completely deterministic and includes elements that introduce a variable time delay. The delay from one pulse to the next will never be constant and will keep varying depending on different factors explained on Page 48.

A.3 Controller Performance with Different Inputs

Figure B3 shows the damping signal generated as a response by the POD running on the cRIO. Each signal was generated using a different damping input. From top to bottom, in Figure B3 they are Active Power, Current Magnitude, Voltage Phasor Magnitude Difference and Voltage Phasor Angle Difference. For the purpose of comparison, the time taken by each of these responses to reduce to and be limited to within +/-1V is recorded in Table B1. From this data, it is obvious that the performance is best with the voltage angle difference as input. Also, from Figure B3 it can be observed that the overshoot when the disturbance is applied is the smallest when the voltage angle difference is used as an input. Also, the peak of the response progressively decreases, i.e. from active power to current magnitude to voltage magnitude to voltage angle difference. This supports the mathematical analysis presented in [12] An elementary analysis of the responses is presented in Figures B4, B5, B6 and B7, which depict the frequency spectra of each of the responses. Note that all responses except that with active power show only one dominant mode which is close to the expected 0.64Hz inter-area mode. The controller's response with active power (Figure B4) shows several modes each of which contribute to distort the generated damping signal.

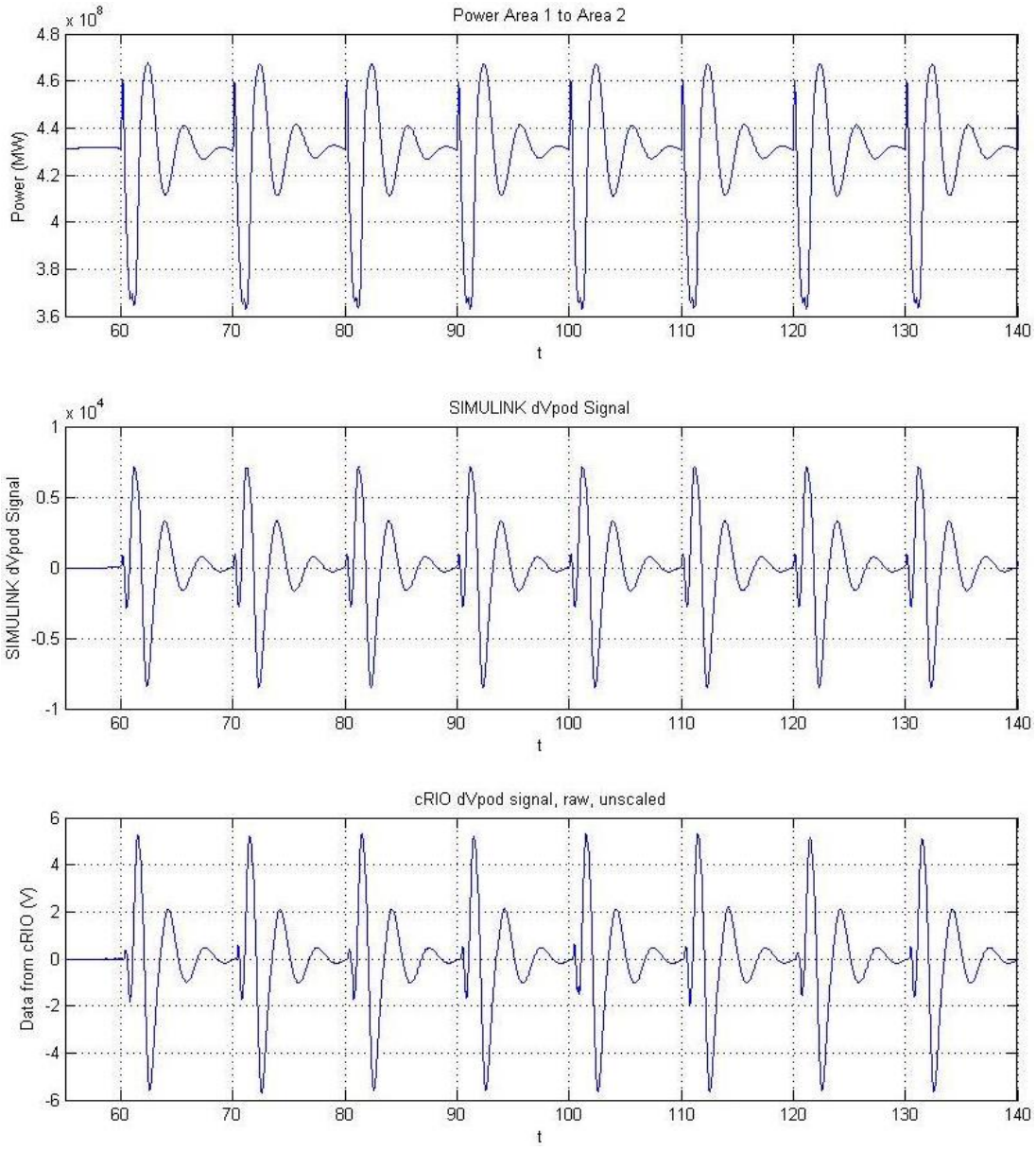

Figure B2: Periodic Disturbance to Calculate Signal Propagation Delay

Table B1: Response Settling Time Comparison

Input to POD	**Response Start Time (s)**	**Response Constrained to +/-1V (s)**	**Settling Time (s)**
Active Power	232.5	244.5	12
Current Magnitude	274	289	15
Voltage Magnitude Difference	322.4	333	10.6
Voltage Angle Difference	375.3	382.8	7.5

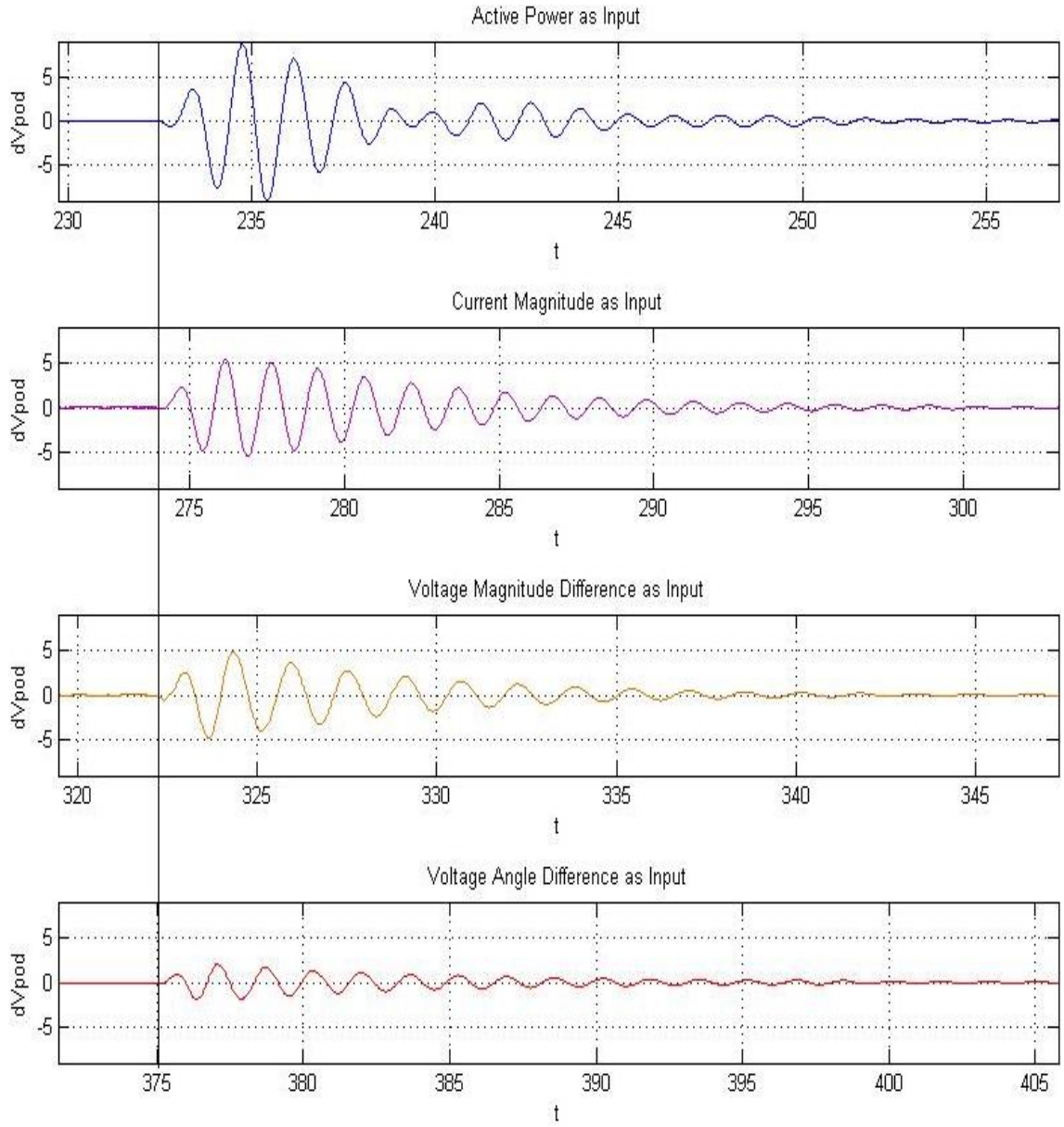

Figure B3: HIL POD Response to Different Inputs

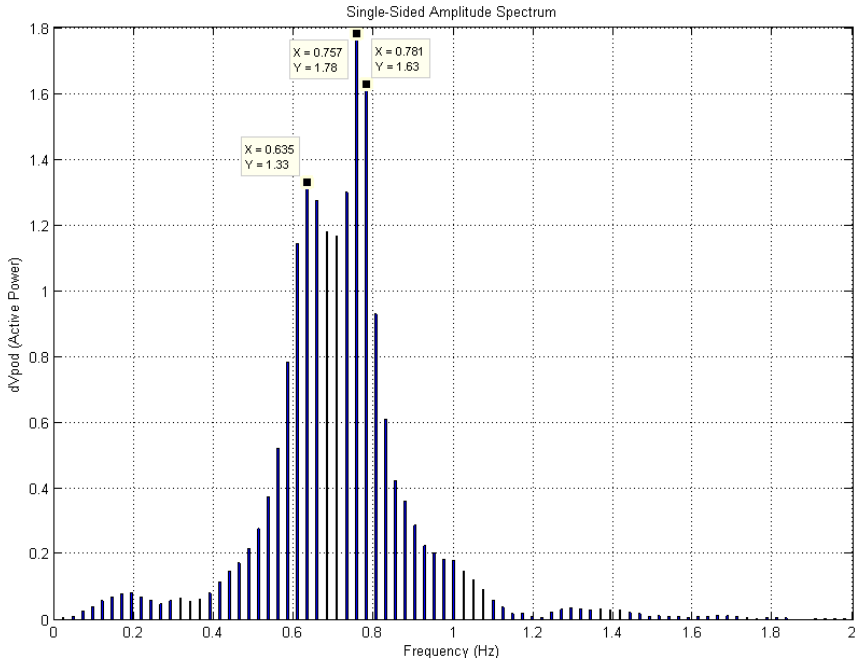

Figure B4: Fourier Analysis of Controller Response with Active Power Input

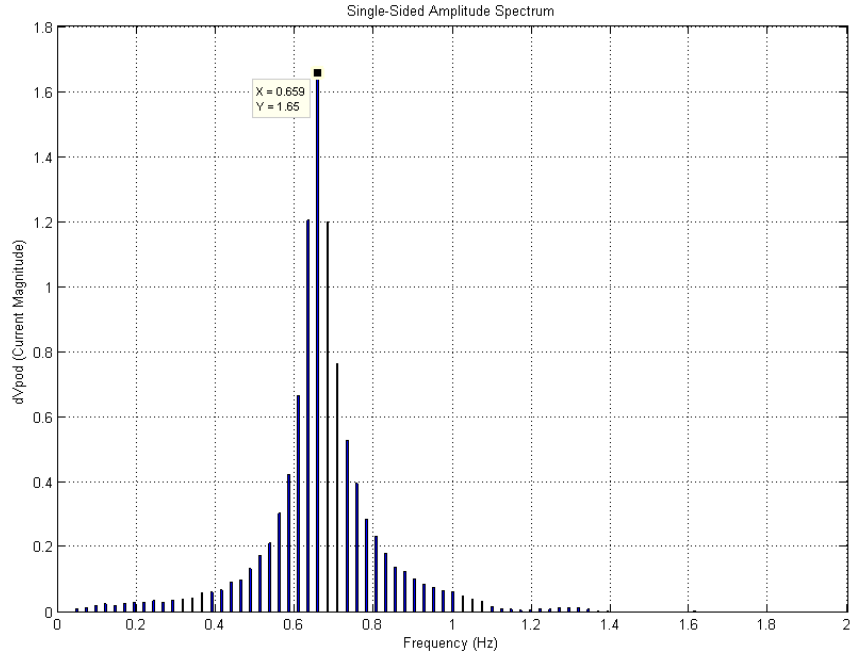

Figure B5: Fourier Analysis of Controller Response with Current Input

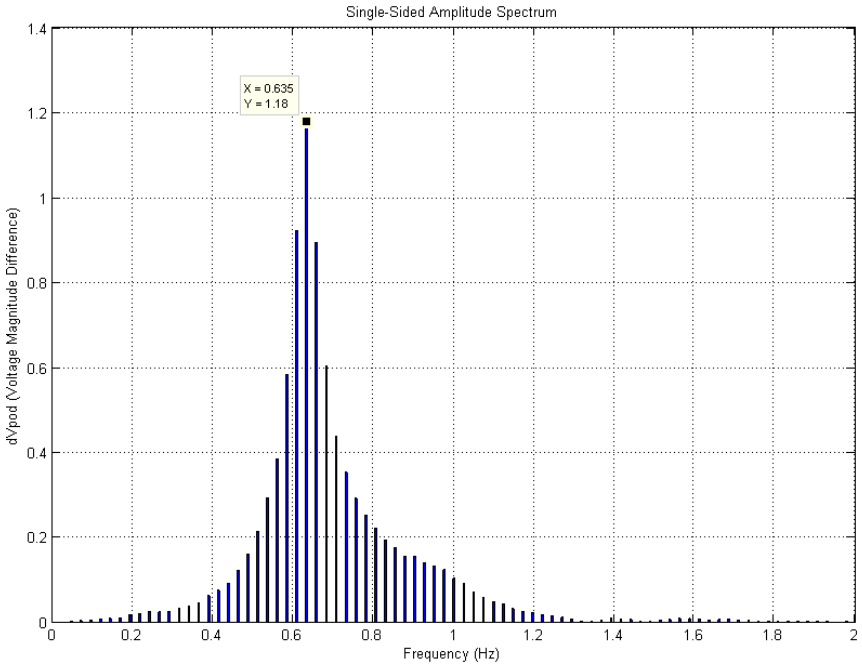

Figure B6: Fourier Analysis of Controller Response with Voltage Magnitude Difference Input

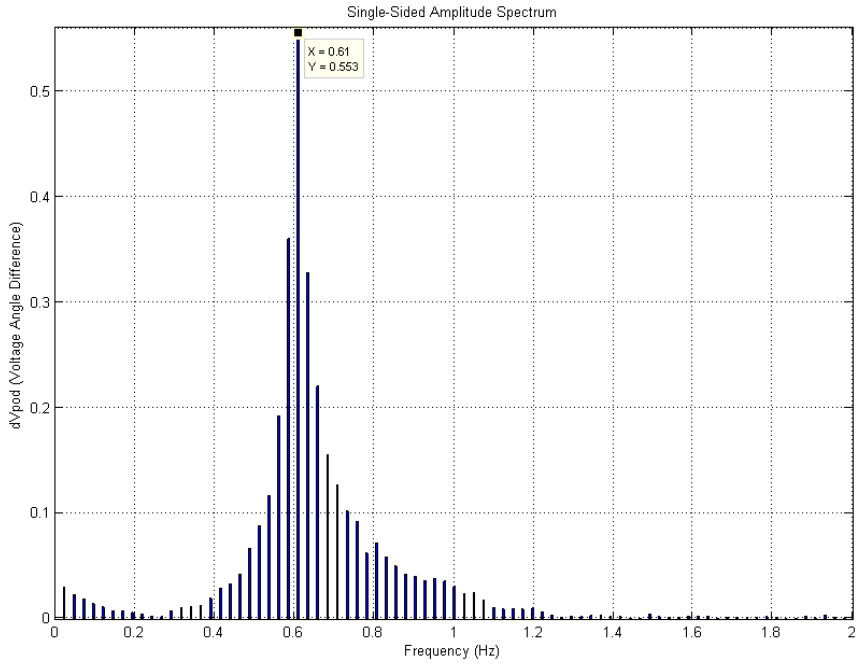

Figure B7: Fourier Analysis of Controller Response with Voltage Angle Difference Input

B Two Area Four-Machine Model Parameters

Common Generator Parameters used in Simulink Two Area Model (Similar to [10])

Table C1: Common Generator Parameters used in Simulink Two Area Model

X_d	1.8	X'_d	1.8	X''_d	0.25
X_q	1.7	X'_q	0.55	X''_q	0.25
X_l	0.2	R_a	0.0025	K_d	0
T'_{d0}	8s	T'_{q0}	0.4s	-	-
T''_d	0.03s	T''_q	0.05s	-	-
A_{SAT}	0.015	B_{SAT}	9.6	ψ_{T1}	0.9

N.B. All Parameters in p.u., on Generator Base

Step Up Transformer Impedance : $0 + j0.15$

Transmission line parameters:

$r = 0.0001$ pu/km	$x_l = 0.001$ pu/km	$b_c = 0.00175$ pu/km	Nominal Voltage : 230kV

Self-excited DC exciter Parameters (common to all generators):

$K_A = 20$	$T_A = 0.055$	$T_E = 0.36$	$K_F = 0.125$
$T_F = 1.8$	$A_{ex} = 0.0053$	$B_{ex} = 1.075$	$T_R = 0.05$

C Interface Screen-shots

Figure D1 shows the application interface running on the workstation computer. The 'Control Settings' Tab is shown. This is where the Phasor Power Oscillation Damper parameters such as the search frequency, the phase shift and the gain are entered. When these parameters are sent to the controller is determined by when the user presses the 'Update Control Configurations' button. The POD damping input selection is also visible on the bottom left and is set to Positive Sequence Voltage Difference in this case.

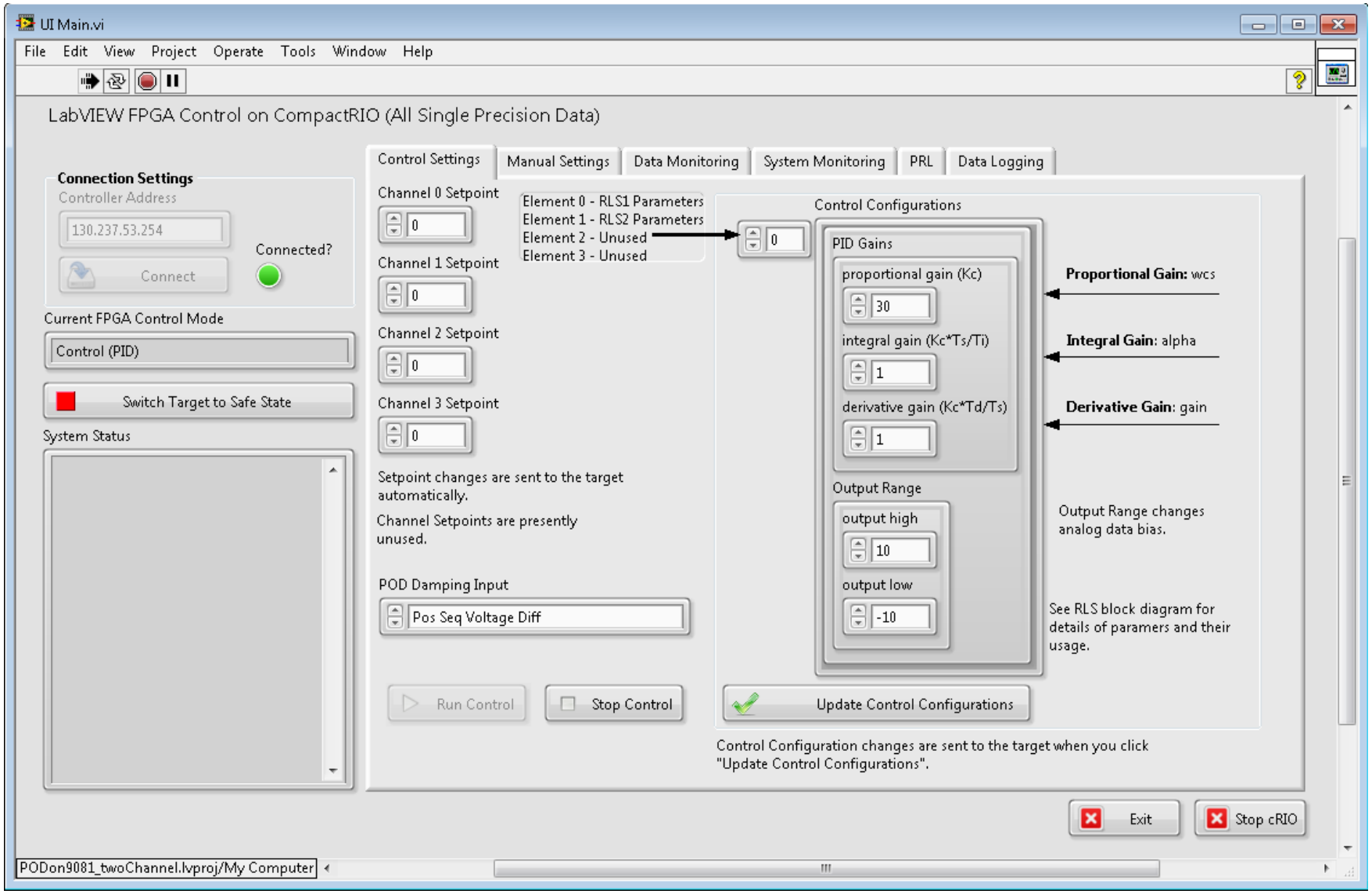

Figure D1: Interface Running on Workstation Computer

D Architecture Development Process

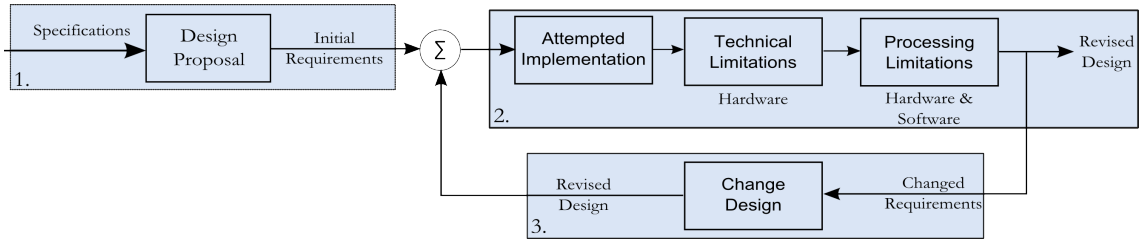

Figure D1: Architecture Review Work-Flow

Before beginning with code implementation on the cRIO, a software architecture was designed. This architecture was designed keeping in mind the functions desired and the computational resources needed. An initial architecture proposal was drawn-up indicating where control functions were located and also the functions of all other devices. This design was progressively changed depending on hardware limitations or processing needs. The final design implemented was arrived at iteratively, after examining the design at each stage. Changes were made to the initial design where required. A three-step process was followed with the three stages being design, development and revision and final implementation, as illustrated in Figure D1.

D.1 Design Process

Figure D1 shows a three stage design process with design proposal, implementation and revision. To begin with, only the controller specifications were available. These were used to draft a design proposal. An implementation attempt was made using this draft proposal. When the additional limits imposed by the software and hardware platforms were included, some goals of the original code would have to revised. With these limitations, the original design was modified to generate a revised design. An attempt was then made to implement this revised design. If further limitations were encountered, the design was further modified. This iterative process was repeated till a working implementation was reached.

D.1.1 Initial Design

Initially, the goal was to keep the controller (the cRIO) independent of any other devices. Such a controller would be able to take synchrophasor data (in the C37.118 format) directly from the network and extract measurement data. The architecture block diagram is shown in Figure D2. This controller was completely autonomous, with automated signal selection and processing. This design would be able to compute observability indices for the different input signals and then automatically select the one with the highest observability of a particular mode. This design also incorporated two control functions, a wide area control function and a local control function. The wide area control function selected for implementation was an

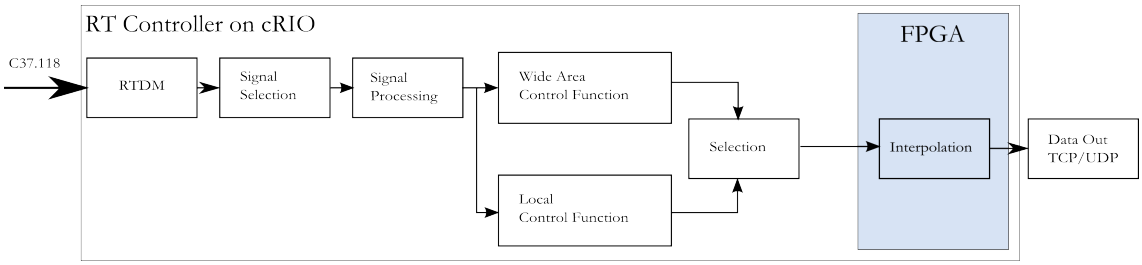

Figure D2: Initial Software Architecture

inter-area oscillation damping control algorithm. The local control function would use locally available data and implement the control function in a manner similar to a Power System Stabiliser (PSS). Automatic switching between the two functions was also considered. If the signal to noise ratio in the wide area signal deteriorated or if the signal delay became prohibitively high, the controller would switch to the local signal or a back-up wide area signal automatically. The principal consideration in this design was the limited resources available on the FPGA and the complexity of the algorithms to be implemented. This required implementation of all algorithms and computation sections on the RT section of the cRIO. In addition, the RT controller has tremendous flexibility as algorithms are implemented using software, rather than hardware as on the FPGA. The RT controller also has more storage space available to implement algorithms. The trade-off was computation speed.

This design was faced with a problem of differing loop rates. The real-time simulator runs at a 50μs time step. The RT controller on the cRIO is not capable of matching this speed. The fastest that it could run was 1ms. To address this issue, this architecture envisaged using the FPGA to perform interpolation between successive data points. The FPGA would run at a loop rate of 50 microseconds, to match the real-time simulator. The region between successive data points was to be interpolated using a suitable interpolation algorithm. Data generated by the FPGA was to be sent over the communication network back to the real-time simulator for use in the simulation.

This design was changed due to limitations of different components. One, no software was available to read a synchrophasor stream on the RT controller and extract measurement data from the stream. This process had to be performed on a desktop computer running a real-time data mediator and LabVIEW.

Two, data generated by the FPGA could not be sent to the real-time simulator directly over the communication network. Though possible, further work is required.

Three, the process of data interpolation on the FPGA is complex. To achieve results better than with a simple linear interpolation, a history of past data points is required. This process is in itself complex and also introduces further delay. How-

ever, it is simpler than implementing the damping control algorithm on the FPGA. At this stage, FPGA limitations were the principal factor behind implementing the oscillation damping algorithm on the RT controller.

D.1.2 Development and Revision

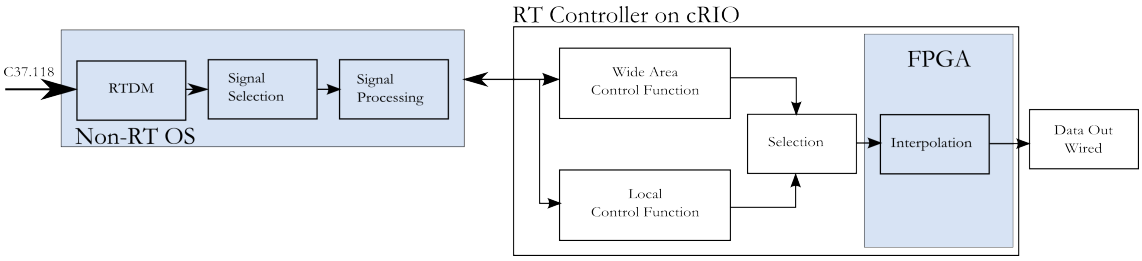

Figure D3: Revised Architecture

With the limitations of the initial architecture in mind, it was revised. In this architecture, the process of extracting data from the synchrophasor stream was shifted to a workstation computer. The process of signal selection and signal processing was also moved to this computer. Once data was available, it would be sent to the RT controller, over a communication network. The damping control was kept on the RT controller with the FPGA performing the interpolation required to match the read-interval of the real-time simulator. Besides the damping control algorithm, a local control function was maintained on the RT controller. This revision is shown in Figure D3.

Here, the complexity of implementing an interpolation algorithm on the FPGA was examined. It was realised that a simple linear interpolation algorithm would not be sufficient. Further, the FPGA output was limited by the speed of the RT controller as new data would only be available after a minimum of 1ms at the earliest (if the RT controller were to run at its maximum speed).

An implementation of the damping control algorithm on the RT controller was also tested. The fastest execution speed that could be achieved was in excess of 25ms. This was slower than the reporting rate of the PMUs.

Communication between the cRIO and the RT simulator using TCP/IP was also abandoned. Output values of the FPGA were instead hardwired to the real-time simulator's analogue inputs.

D.1.3 Final Implementation

At this stage, the damping control algorithm was moved to the FPGA with the RT controller being used only for network communication and data monitoring. The

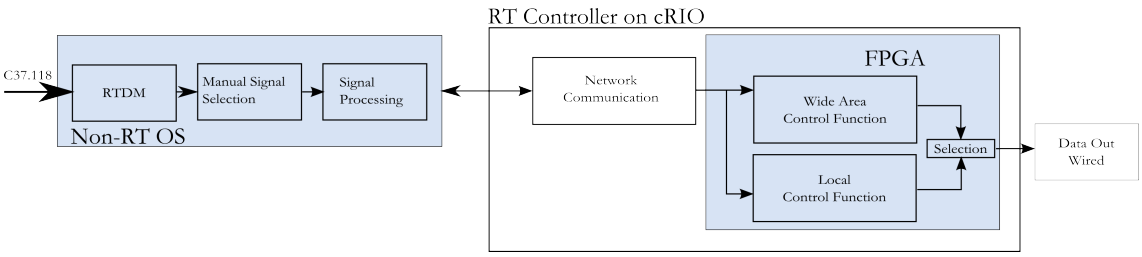

Figure D4: Final Software Architecture as Implemented

local control function was also discarded as it was found to reduce the response speed of the RT controller. If needed, the local control function can be implemented on the FPGA. The FPGA is suited to tasks that are repetitive in nature. The FPGA also has a fast and predictable response time. With the oscillation damping algorithm implemented on the FPGA, resource utilization stood at 78%. Space is still available if further functions need to be implemented on the FPGA. As with the previous design, the process of extracting data from the synchrophasor stream was done on a workstation computer. Signal selection and processing was also done on this computer. This was the final architecture as implemented.

E Setup Photos

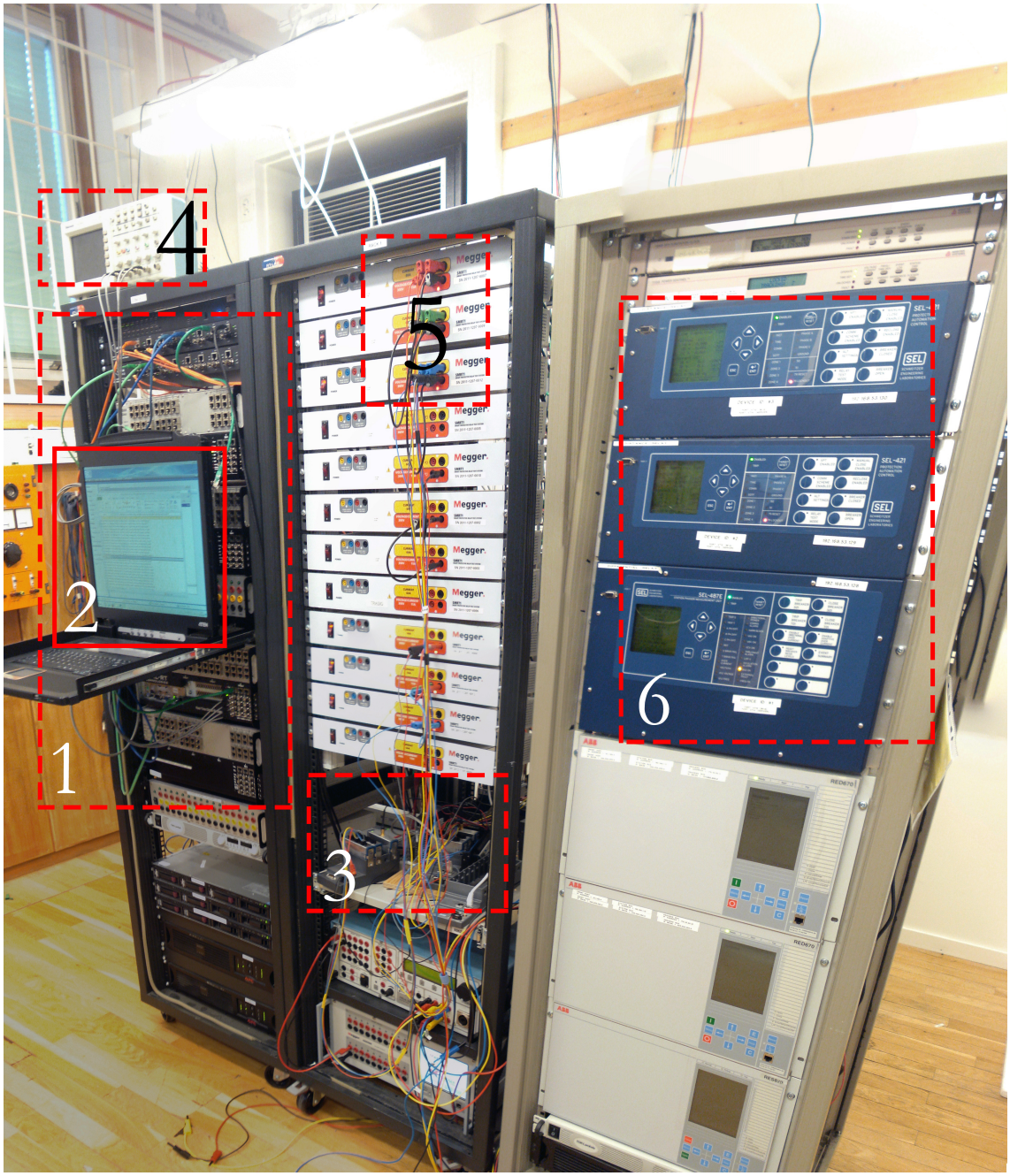

Figure E1: Outline of the SmarTS Lab. Corresponding to Numbers: 1: Real Time Simulator, 2: PDC Interface, 3: cRIO Tray, 4: Oscilloscope, 5: Analogue Signal Amplifiers, 6: SEL Protection Relays

Figure E2: cRIO 9081, With Voltage Module Mounted

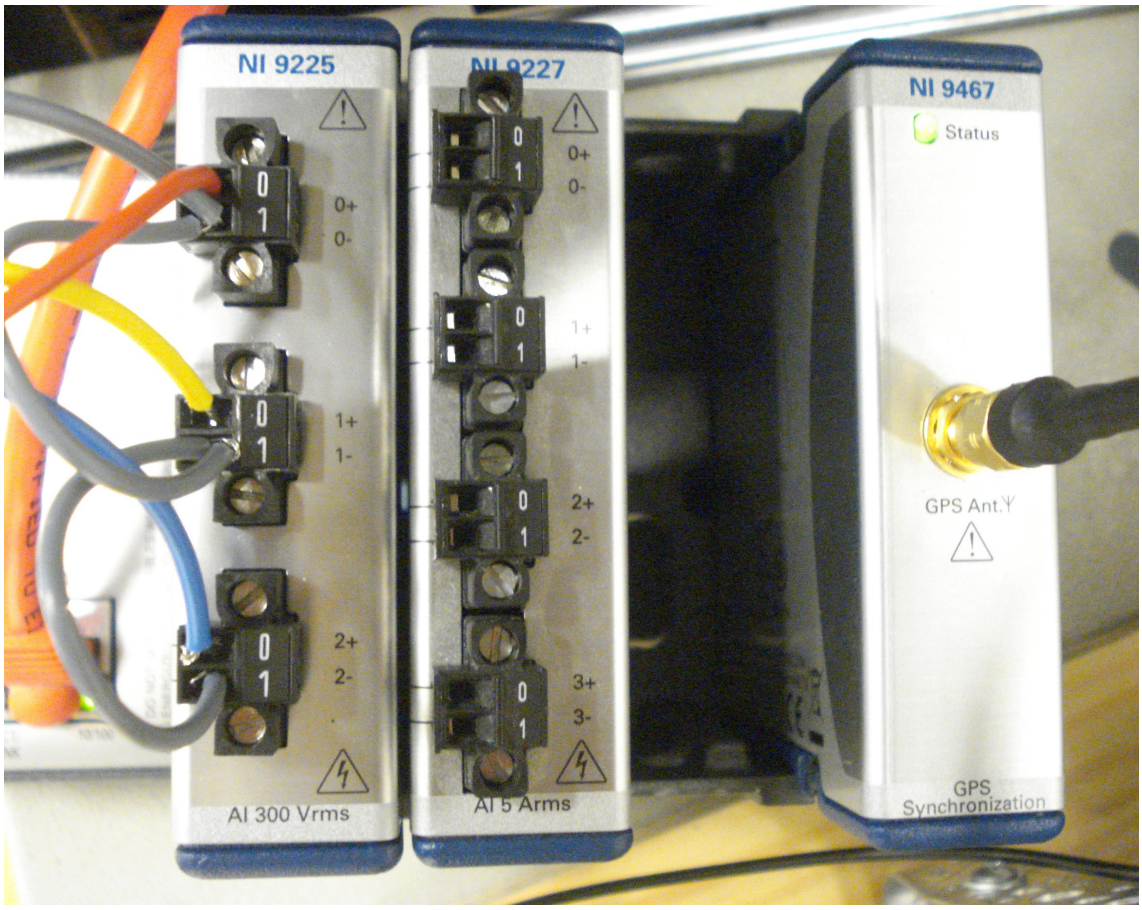

Figure E3: cRIO9076, Functioning as PMU. Shown, from L to R : Voltage Input Module, Current Input Module, GPS Module