Software Architecture Development and Implementation of a Synchrophasor-based Real-Time Oscillation Damping Control System

Eldrich Rebello
Kungliga Tekniska högskolan
Stockholm, Sweden
Email: rebello@kth.se

Luigi Vanfretti
Statnett SF
Oslo, Norway
Email: luigi.vanfretti@statnett.no

Md. Shoaib Almas
Kungliga Tekniska högskolan
Stockholm, Sweden
Email: msalmas@kth.se

***Abstract*—Low-frequency, electromechanically induced, inter-area oscillations are of concern in the continued stability of inter-connected power systems. Wide Area Monitoring, Protection and Control (WAMPAC) systems based on wide-area measurements such as synchrophasor (C37.118) data can be exploited to address the inter-area oscillation problem. This work develops a hardware prototype of a synchrophasor-based oscillation damping control system. A Compact Reconfigurable Input Output (cRIO) controller from National Instruments is used to implement the real-time prototype. This paper presents the design process followed for the development of the software architecture. The design method followed a three step process of design proposal, design refinement and finally attempted implementation. The goals of the design, the challenges faced and the refinements necessary are presented. The design implemented is tested and validated on OPAL RT's eMEGASIM real-time simulation platform and a brief discussion of the experimental results is included.**

I. Introduction

The growth of power system interconnections between previously unconnected areas has given rise to the phenomenon of inter-area oscillations. These are low frequency (0.1-2 Hz.) oscillations where the generators of one synchronous area oscillate against those of another area. Damping for these oscillations is generally poor and if they are allowed to grow, they can lead to disconnection of the ties or a collapse of the power system. A famous example of the latter was the August 1996 blackout of the WSCC system in the USA [1]. 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.

A. Previous Experiences

Modern solutions to this problem involve using Power System Stabilisers (PSS) with locally available signals. These are effective at damping intra-area modes with good observability but may not be as effective at damping inter-area modes [7] [8]. A theoretical analysis of the advantages of using wide-area signals as a damping input is presented in [16]. Field-test experiences with WAPOD controllers from Norway and China are presented in [4] and [5] respectively. Although these initial field tests show promising results, the wide-area control systems tested so far have been implemented by extending the installed control system of an existing device (e.g. an SVC, see [4]) to receive synchrophasor data, process it and to then feed the control algorithm. To the knowledge of the authors, there has not been any reported attempt at designing a general purpose, wire-area control system, starting from specifications and considering different hardware and software constraints. Such an approach is attractive as it can easily be adapted to different controllable elements thereby reducing implementation costs and facilitating straightforward development.

B. Contributions

The goal of this paper is to document the software architecture development process and challenges faced in the real-time implementation of a wide-area control system. Ängquist and Gama's [2] Phasor Power Oscillation (Phasor POD) algorithm was chosen based on the operational requirements and control system design constraints. The developed prototype is termed a Wide Area Power Oscillation Damper [1] (WAPOD). For purposes of comparison, the real-time SIMULINK implementation of the Phasor POD algorithm by Almas and Vanfretti [9] is used as a benchmark. The performance of the hardware prototype developed will be tested and compared to the performance of this implementation of the algorithm. The two-area four-machine model developed by Klein, Rogers and Kundur [3] is used as a test-case and is modified to fit the requirements of experimental testing. A Hardware-in-the-loop experiment is constructed around the eMEGASIM real-time simulator from OPAL-RT [10] with the two-area model running in real-time, physical PMUs and a synchrophasor-based Phasor-POD algorithm running on a cRIO (Figure 1).

[1]Historically, damping stabilizers have been termed WAPOD where the P represents a measurement of active power through the line. In such context, active power is 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.

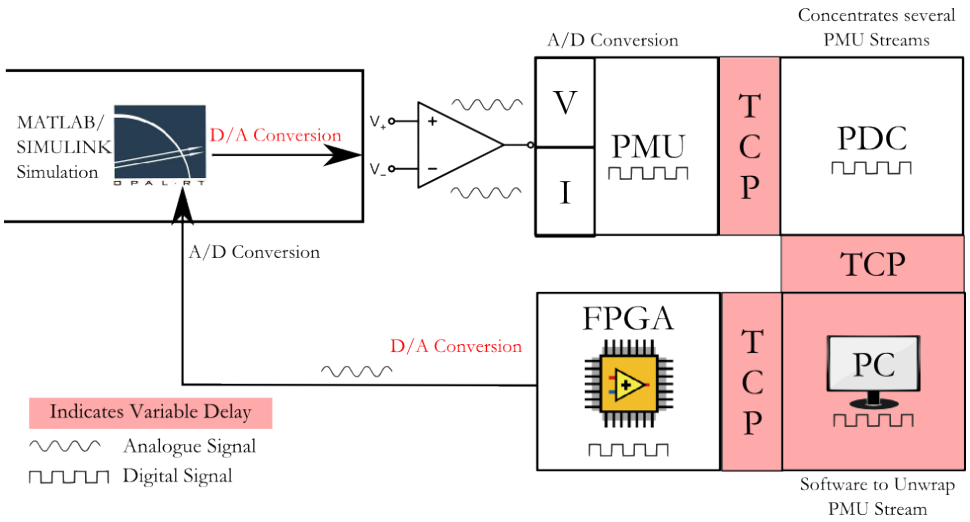

Fig. 1. Data flow with digital and analogue components indicated

C. Paper Outline

This paper is organised as follows. Section II outlines the hardware and software components used together with an overview of the Phasor POD algorithm implemented. Section III examines the details of the architecture developed for this controller and the various stages of refinement and changes made to it. Section V presents the results of the Hardware-in-the-loop experiments performed and conclusions are drawn in Section VI.

II. Background

A. Phasor POD Algorithm

The Phasor-POD algorithm was developed by Ängquist and Gama [2] and forms the core of this work. The algorithm was selected due to its wide applicability and the fact that it does not depend on the network model or updated knowledge of the network topology. Typical model-linearisation based damping algorithms rely on computer-intensive calculations and are often valid for a particular network operating point. The Phasor-POD algorithm only requires knowledge of the inter-area oscillation frequency, which is generally known from system studies or can be determined from synchrophasor measurements. In the two-area network used here, the inter-area oscillation frequency is known to be 0.64Hz. Consider 1 which is a representation of a general signal $s(t)$, as an average valued component and an oscillating component.

$$s(t) = s_{avg} + \mathrm{Re}\left\{ \vec{s}_{ph} \cdot e^{j^{wt}} \right\} \tag{1}$$

where $\vec{s}_{ph}$ is a complex phasor, rotating at the frequency ω [2]. The Phasor-POD algorithm uses the known oscillation frequency to set up a co-orindate system rotating at this frequency and continuously extracts a phasor representing the oscillation [2]. This extracted signal can subsequently be used as a modulating input to a controllable device such as the exciter of a FACTS device. The implementation of the algorithm used here is based on Almas and Vanfretti's Simulink implementation [9] and uses three inputs : the search frequency, ω_{cs1}, the sampling time T_s, the phase correction $alpha$ in addition to a signal scaling factor.

B. Hardware and Software

Real-Time Controller: The real-time implementation of the Phasor-POD algorithm in this work is based on the Compact Reconfigurable Input Output (cRIO) from National Instruments. The cRIO 9081 is used to implement the algorithm. It has an onboard Field Programmable Gate Array (FPGA) in addition to a 1.06 GHz. Intel Celeron processor for real-time control applications [17].

Phasor Measurement Units: The inputs to the real-time controller were taken from an IEEEC37.118-compliant synchrophasor data stream. Measurement data for this stream was generated by two PMU's which monitor three-phase currents and voltages at different points on the power network. In this work, two cRIO9076 real-time controllers [18] were deployed as PMUs [13].

Real-Time Simulator: The eMEGASIM real-time simulator from OPAL RT [6] was used for simulating the two-area network in real-time. This simulator allowed for hardware-in-the-loop tests to be carried out using its analogue input and output terminals. Voltage and current signals were extracted from the simulator's analogue outputs and fed to analogue amplifiers. These amplified signals were then wired to the current and voltage inputs of PMU's. Similarly, the damping signal generated by the Phasor-POD algorithm was then wired back to the simulator's analogue input terminals for use in the simulation. Figure 1 presents the entire signal path including the external, closed-loop controller.

Software: The test-case network model used here was a version of the Klein-Rogers-Kundur two-area system implemented in a SIMULINK demo available with SimPowerSystems [23]. This was modified to include an average-valued SVC model, identical to that used in [9]. The SIMULINK

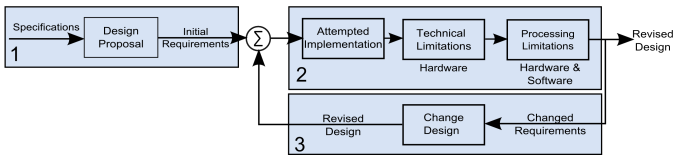

Fig. 2. General Iterative Revision Flow Diagram

model was modified and prepared for real-time simulation. The software on the cRIO was written using LabView's Real-Time modules. OPAL RT's eMEGASIM platform uses MATLAB code and SIMULINK models. Software development was done on a workstation computer. Code was loaded on the various devices (cRIO contollers, RT Simulator) over a TCP network.

III. Architecture Development Process

Before beginning the development process, a software development methodology was adopted with the aim of streamlining the process[2]. The approach followed here is based on the Waterfall method detailed in [11]. The approach broadly includes the four steps from [11]

1) Initial Investigation
2) Requirements Definition
3) Architecture Design
4) Coding and Implementation
5) Experimental Testing to validate requirements

An additional fifth step could be avoided thanks to the availability and use of HIL testing. The linear waterfall method was modified to be iterative so as to account for various constraints and to also account for revisions in the initial definition caused by these constraints. This iterative process is illustrated in Figure 2 and is realised for the specific application here, as illustrated in Figure 3. Several reasons were behind the choice of an iterative design approach. Chief among them was the cRIO hardware itself. As an implementation on this hardware had not been attempted earlier, the limitations of the hardware were unknown. Based on the limitations faced during an implementation attempt, the design and features incorporated would be revised to fit within the limitations of the hardware.

A. Constraints

Before defining requirements at each stage of the software implementation, the possible constraints at each stage of the test set-up were considered. The test set-up shown in Figure 1 is used to illustrate the constraints faced.

Differing Loop Rates: The power system simulation running on the real-time simulator was run at a loop rate of 50μs. This meant that the simulator generated new values for currents and voltages every 50μs. and would also expect data from the HIL system every 50μs. The constraint here was the data reporting rate of the PMU's used which was a maximum of 50 samples per second or one sample every 20ms. This was much slower that the 50μs. loop rate of the real-time simulator. A solution was required to address this issue.

FPGA Accuracy: While the exact role of the FPGA in this implementation was not known in advance, it brought with it some limitations. Chief among these was the limitation on data accuracy due to the fact that all calculations together with circuit logic would be implemented in hardware.

IV. Architecture Implementation and Refinement

Figure 2 shows the 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 implementation needed to be 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.

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 receive synchrophasor data (in the IEEE C37.118 format) directly over the communication network (ethernet) and extract measurement data. The architecture block diagram is shown in Figure 3. 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 the phasor-based oscillation damping control algorithm [2]. 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 chosen 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.

[2]There are many software development methods that could be adopted. However, for this specific application (i.e a PMU-based WAPOD), none of them have been reported and used in available literature. The process here was favoured over the others due to the availability and limitations of the hardware-in-the-loop experimental testing facilities available at the SmarTS Lab [14]

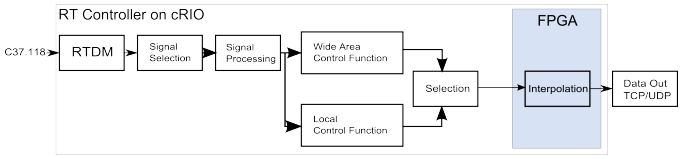

Fig. 3. Initial Software Architecture

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. Data would thus be generated faster than the RT controller was able to process. The second problem was producing control output at the rate expected by the real-time simulator. The speed constraint of the RT controller meant that it would not be possible to use it (the RT controller) to generate this control signal. 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 would be interpolated using a suitable interpolation algorithm. Data generated by the FPGA would 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 receive a synchrophasor stream and extract measurement data on the RT controller. This process had to be performed on a desktop computer running a real-time data mediator (Statnett's Synchrophasor Software Development Toolkit, S^3DK [20]) and LabVIEW. Once measurement data from the synchrophasor stream was available, it could be streamed to the real-time controller over the TCP/IP network using LabView's Shared Network Variables.

Two, data generated by the FPGA could not be sent to the real-time simulator directly over the communication network. Though theoretically possible using a FIFO[3] buffer, further work is required. Also, for each succesive data point received by the real-time controller, the FPGA would generate 400, interpolated data points. Synchronising the process of generating the control signal and using the generated data on the real-time simulator will be a complex task.

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. However, it is simpler than implementing the damping control algorithm on the FPGA. At this stage, FPGA limitations were the principal factor behind chosing to implement the Phasor-POD algorithm on the RT controller.

Four, an algorithm for automated signal selection based on observability indices had not been developed. In principle, this too, is possible and can be implemented on the real-time controller in the future when such an algorithm is developed and verified. The Phasor POD algorithm input paramaters also change depending on the input used. An approach to determine these parameters iteratively or calculate them from input data is required. This only adds to the complexity.

2) Development and Revision: With the limitations of the initial architecture in mind, revisions to the design were made. 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 included on the RT controller. This revision is shown in Figure 4. Here, the complexity of implementing an interpolation algorithm on the FPGA was examined in detail. An implementation was also attempted. It was determined that a simple linear interpolation algorithm would not be sufficiently accurate. 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 least (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 loop rate that could be achieved was always 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. Analogue output signals of the FPGA were instead hardwired to the real-time simulator's analogue inputs.

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 RT controller would only perform the task of receiving measurement data continuously over the network. Phasor POD algorithm parameters and monitoring data would be sent periodically. The local control function was also discarded as it was found to reduce the response speed of the RT controller. If needed and if space permits, the local control function can be implemented on the FPGA. The FPGA is suited to tasks that are repetitive in nature as it has a fast and predictable response time. The desired 50μs. data output rate could also be maintained as the FPGA was capable of response times of this order. The problem of differing data and loop rates was solved by implementing a basic sample-and-

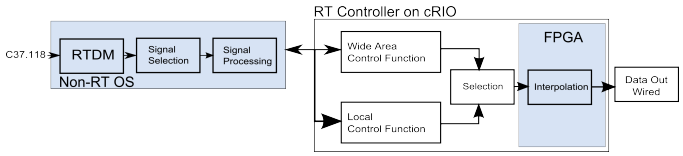

Fig. 4. Revised Architecture

[3]First In First Out

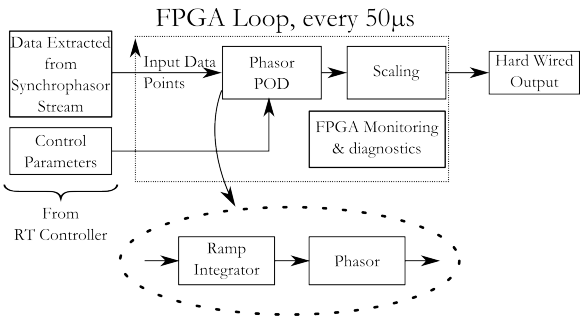

Fig. 6. FPGA Software block diagram

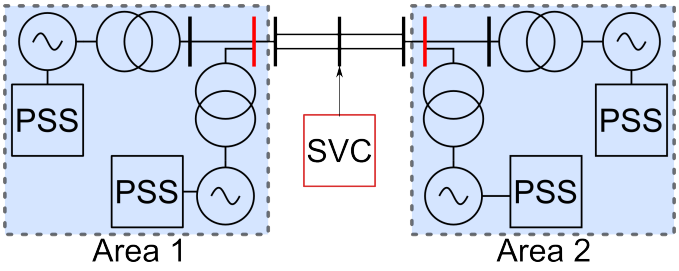

Fig. 7. Two Area Test network used. PMU measurements are taken from the buses indicated in red.

hold algorithm on the FPGA. Data would still be received from the RT controller every 20 ms. but the output would be held constant till the next data point was received and processed. With the Phasor POD algorithm implemented on the FPGA, resource utilization stood at 78%. Space was 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 (Figure 5) was the final architecture as implemented. An outline of the FPGA code as implemented are shown in Figure 6.

V. Hardware-in-the-Loop Test

A setup identical to that outlined in Figure 1 was constructed and used to verify the working of the real-time implementation of the Phasor-POD algorithm. The Phasor-POD algorithm had previously been tested with the two area system within a SIMULINK environment (no HIL execution). It was verified to be able to keep the system in steady state stability and to restore the system to stability after the application of a minor disturbance. The benchmark for the real-time implementation of the algorithm is thus established.

The single-line-diagram of the two-area test network used is shown in Figure 7 with PMU locations shown by the red buses. The signal generated from the real-time Phasor POD implementation is reinserted in the model at the control system of the SVC. This model was simulated in real-time.

The hardware implementation of the POD algorithm was verified to be able to keep the system stable in steady state. It was also tested with small perturbations applied at one generator to excite the inter-area mode and was verified to work satisfactorily. Figure 8 shows the performance of the

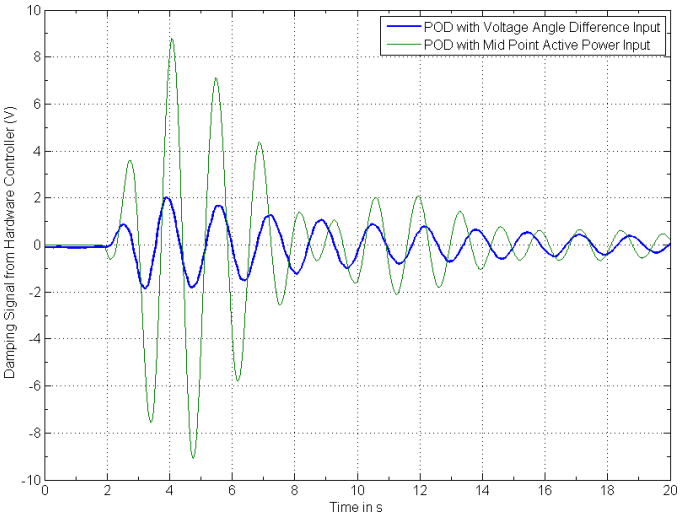

Fig. 8. Performance of Hardware controller with small disturbance. Note that the decreasing magnitude of the damping signal indicates that the oscillation magnitude is decreasing in correspondence.

controller with two different inputs. The green trace shows the damping performance with active power as a controller input while the blue trace shows damping performance with voltage angle difference as a controller input. It is immediately evident that using the voltage angle difference provides better performance. Also observed was the fact that stead state performance had less noise with using the voltage angle difference as a controller input. Though this performance is not identical to that observed in simulations, it indicated that the real-time implementation of the algorithm was successful. Also, the damping signal generated (in Volts) by the hardware controller was captured on an oscilloscope and is presented in Figure 9.

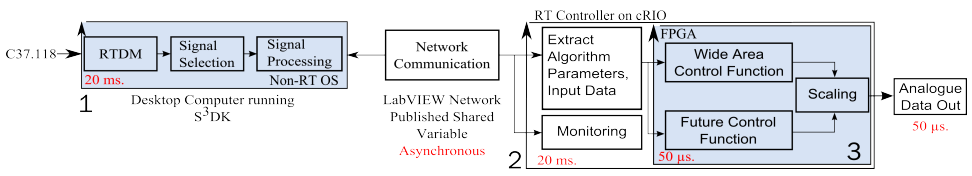

Fig. 5. Final Controller Architecture as Implemented

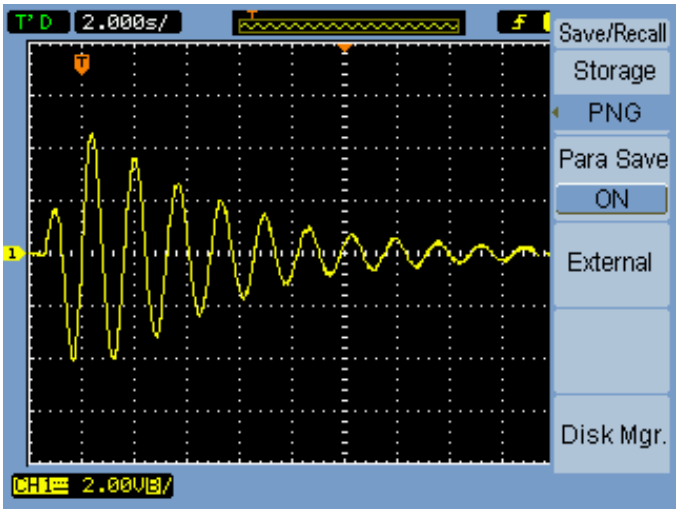

Fig. 9. Controller Response with Voltage Angle Difference input captured using an Oscilloscope

VI. Conclusion

A hardware prototype of a real-time power oscillation damping control system was developed and tested. The developed prototype uses a real-time implementation of a wide-area control system. Issues and challenges faced in this implementation are documented and examined. The architecture development & refinement process for this implementation is also examined in detail. The success of the tests performed indicates that PMU-based wide area controllers can be explouted for multiple control applications in power system control and monitoring.

Acknowledgment

The work of E. Rebello & M.S. Almas was supported by the STRONg^2rid project, funded by Nordic Energy Research. L. Vanfretti is supported by the STRONg^2rid project, funded by Nordic Energy Research, and by the STandUP for Energy collaboration initiative and by Statnett SF, the Norwegian transmission system operator.

References

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

[2] L. Ängquist and C. Gama *Damping Algorithm based on Phasor Estimation* in Power Engineering Society Winter Meeting, 2001. IEEE, Volume 3, pp. 1160 - 1165

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

[4] K. Uhlen, L. Vanfretti, M.M. De Oliveira, A.B. Leirbukt,V. H. Aarstrand and J. O. Gjerde *Wide-Area Power Oscillation Damper implementation and testing in the Norwegian transmission network* in Power and Energy Society General Meeting, 2012 IEEE pp. 1-7

[5] Li Peng and Wu Xiaochen and Lu Chao and Shi Jinghai and Hu Jiong and He Jingbo and Zhao Yong and Aidong Xu *Implementation of CSG's Wide-Area Damping Control System: Overview and experience* in Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES pp. 1-9

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

[7] F. P. Dmello, and C. Concordia *Concepts of Synchronous Machine Stability as Affected by Excitation Control* IEEE Transactions on Power Apparatus and Systems, vol. PAS 88, no. 4, 1969.

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

[9] 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.

[10] eMEGAsim PowerGrid Real-Time Digital Hardware in the Loop Simulator Opal RT, [Online]. Available: http://www.opal-rt.com/

[11] James Taylor *Managing Information Technology Projects : Applying Project Management Strategies to Software, Hardware, and Integration Initiatives*, AMACOM Books, September 2003 pp 237-239

[12] N.S. Chaudhuri, R. Majumder and B Chaudhuri *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

[13] M. Paolone, A. Borghetti, C. A. Nucci, *A Synchrophasor Estimation Algorithm for the Monitoring of Active Distribution Networks in Steady State and Transient Conditions*, Proc. of the 17th Power Systems Computation Conference (PSCC 2011), Stockholm, Sweden, Aug. 22-26, 2011

[14] L. Vanfretti.; M. Chenine; M.S. Almas; R. Leelaruji; L. Angquist; L. Nordstrom, "*SmarTS Lab A laboratory for developing applications for WAMPAC Systems*," Power and Energy Society General Meeting, 2012 IEEE , vol., no., pp.1,8, 22-26 July 2012

[15] E.V Larsen and E.H Chow, General Electric Company, NY, *SVC control design concepts for system dynamic performance*, in Application of Static Var Systems for System Dynamic Performance 1987, IEEE, pp. 36-53

[16] 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

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

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

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

[20] L. Vanfretti, V.H. Aarstrand, M. Shoaib Almas, V. S. Perić and J.O Gjerde *A Software Development Toolkit for Real-Time Synchrophasor Applications*, IEEE PES Grenoble PowerTech, 2013

[21] I Kamwa *Performance of Three PSS for Interarea Oscillations* SimPowerSystems Examples, MATLAB R2011a

[22] *Generic Power System Stabilizer* Documentation distributed with MATLAB R2014a Available online at http://www.mathworks.se/help/physmod/sps/powersys/ref/genericpowersystemstabilizer.html

[23] 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

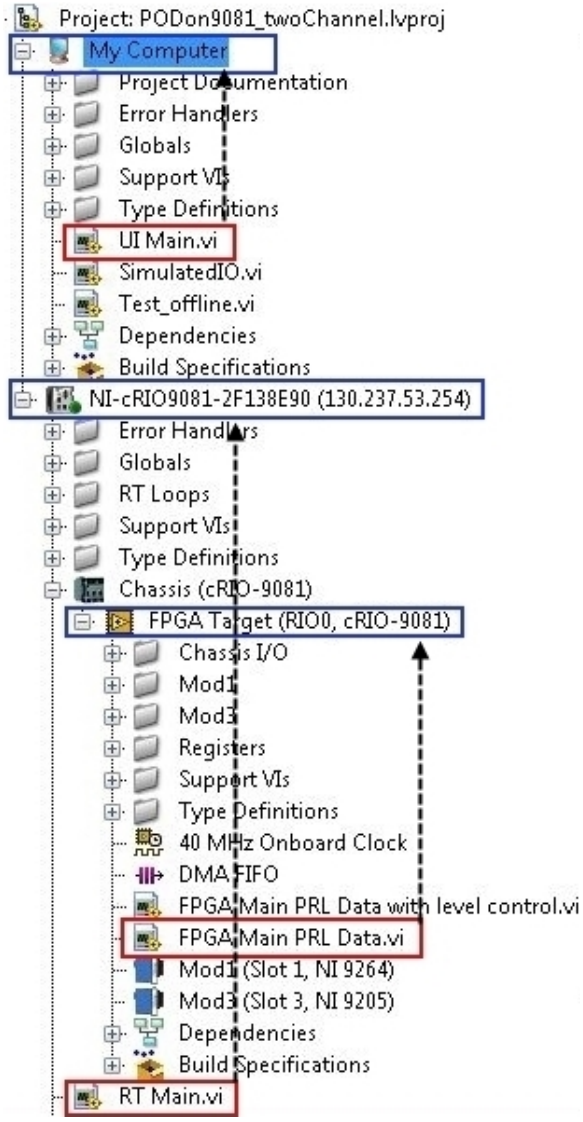

Fig. 10. LabView Project Explorer showing different sections of code and the devices on which they run. **This image is too tall to fit within the text. I wanted to include this as an appendix but the Latex file says that conference papers generally don't have appendices.**