Resilient Information Architecture Platform for the Smart Grid (RIAPS): A Novel Open-Source Platform for Microgrid Control

Hao Tu, *Student Member, IEEE,* Yuhua Du, *Student Member, IEEE,* Hui Yu, *Student Member, IEEE,* Abhishek Dubey, *Senior Member, IEEE,* Srdjan Lukic, *Senior Member, IEEE,* and Gabor Karsai, *Senior Member, IEEE*

***Abstract*—Microgrids are seen as an effective way to achieve reliable, resilient, and efficient operation of the power distribution system. Core functions of the microgrid control system are defined by the IEEE standard 2030.7; however, the algorithms that realize these functions are not standardized, and are a topic of research. Furthermore, the corresponding controller hardware, operating system, and communication system to implement these functions vary significantly from one implementation to the next. In this paper, we introduce an open-source platform, Resilient Information Architecture Platform for the Smart Grid (RIAPS), ideally suited for implementing and deploying distributed microgrid control algorithms. RIAPS provides a design-time tool suite for development and deployment of distributed microgrid control algorithms. With support from a number of run-time platform services, developed algorithms can be easily implemented and deployed into real microgrids. To demonstrate the unique features of RIAPS, we propose and implement a distributed microgrid secondary control algorithm capable of synchronized and proportional compensation of voltage unbalance using distributed generators. Test results show the effectiveness of the proposed control and the salient features of the RIAPS platform.**

***Index Terms*—Distributed control, microgrid, microgrid control system.**

I. Introduction

THE ever increasing number of distributed generators (DGs) located deep in the distribution system makes conventional centralized control unsustainable. To address this, there is a need for various customer and utility owned resources to communicate and interact intelligently with each other with minimal data exchange to a centralized supervisory control and data acquisition (SCADA) system. Forming microgrids provides a natural way for these resources to interact while offering various services to the utility. Therefore, microgrids can help achieve stable and economical operation of the future grid while improving system resiliency and flexibility.

Per IEEE standard 2030.7 [1], a microgrid control system must provide two core functions: (1) the *dispatch function*, which dispatches individual devices in given operating modes and with specified setpoints; and (2) the *transition function*, which supervises the transitions between grid-connected and islanded states, and ensures that the dispatch is appropriate for the given state. The secondary and tertiary levels of hierarchical control try to optimize the dispatch functions to utilize the available resources based on optimization criteria. The transition function manages the primary-level controller where the DG operation modes or component states change (e.g. transition from grid forming to grid following mode for an inverter, or breaker turn-off as a part of an emergency dispatch order).

To design, implement, and deploy a microgrid control system, there is a need for a high fidelity testbed to emulate the microgrid power system, as well as the control system on which the microgrid controller resides. The testbed must be flexible enough to allow for modeling of a diverse set of microgrid topologies while being detailed enough to accurately represent the real-world implementation complexities. If the system is to be deployed on an actual microgrid, the developed control system should be able to interface with commercial devices and therefore support the native protocols used by the hardware controllers on the market today.

There are well-established methods of emulating microgrids and DG intensive distribution grids through the use of real-time digital simulators [2]–[6] (e.g. OPAL-RT, Typhoon, dSPACE, and etc.), or through scaled down [7]–[17] or full-scale [18]–[21] hardware implementations. Additionally, multiple tools exist that enable modeling of the communication networks (e.g. OPNET, NS-2, OMNET++, and etc.), and these tools have been successfully integrated into various testbeds [2], [3], [22], [23].

What has not been reported in the literature, according to our knowledge, is an open-source platform that allows for the seamless development, testing, and deployment of a microgrid control system. Such capability is important to understanding the interactions between different control levels in a practical system and exploring the complexities of implementing analytically developed algorithms in the real world, e.g., realizing computationally complex algorithms on nodes with limited computational capability. Importantly, a platform should provide all the necessary tools to deploy the system into the field. Prototyping platforms exist today, but most of them lack critical advanced functions, such as time synchronization and real-time capabilities, that are emerging in proprietary solutions.

In this paper we review the microgrid control platforms that have been proposed in literature and point out their advantages and shortcomings. We then present a new communication and computation platform, called Resilient Information Architecture Platform for the Smart Grid (RIAPS), which is designed to operate at the edge of the power grid. To demonstrate the unique features of the platform, we propose and implement a

This work was supported in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000666.

distributed microgrid secondary control algorithm capable of synchronized and proportional compensation of system voltage unbalance caused by unbalanced loads. The application demonstrates how the platform achieves multi-agent system (MAS)-type functionality through the implementation of a distributed control algorithm. Additionally, it demonstrates the time and functionality coordination between the system primary-level and secondary-level controllers, to establish a globally synchronized reference frame in which the DGs can proportionally compensate system unbalance at a selected microgrid bus.

The main contributions of this paper include:

- Comprehensive review of the state of the art microgrid control platforms used in the literature and their features and limitations.
- Complete description of the RIAPS platform features including the unique development tool suite and run-time services while considering system requirements such as scalability over a large network, real-time response to events, and time synchronization among nodes.
- Implementation of a novel microgrid control algorithm that achieves synchronized and proportional compensation of system voltage unbalance. The application demonstrates the key features of the platform used for microgrid control including node time synchronization, implementation of a distributed consensus algorithm, communication protocol support, etc.
- Characterization of RIAPS platform performance in terms of communication delay, packet loss rate, and synchronization error in a close-to-real environment, providing a reference for control algorithm developers.

The RIAPS platform is open-source and available to the community; it can be freely used for both lab prototyping and field implementation. The platform is ideally suited for implementing distributed control algorithms and for performance benchmarking of other microgrid control systems.

II. Related Work

Microgrid control systems are often implemented and tested in either in-house developed custom platforms or rely on proprietary solutions. Typically, little thought is given to how the system could migrate into the field.

In [2], OPAL-RT real-time simulator is used for grid simulation while OPNET emulates the behavior of the communication network. Similar approach with RTDS (Real Time Digital Simulator) as grid simulator and OMNET++ as network simulator can be found in [3]. Communication-based control algorithms are implemented in the grid simulator along with the low level power electronics control. A testbed with a grid simulator and network simulator allows for flexible simulation of different microgrid topologies and communication network configurations; however, with the control algorithms implemented and executed in the simulator, controller computational limitations, controller interactions, effects of computational time delays, and controller synchronization become impossible to evaluate, and are overlooked. In addition, porting the algorithms into the real microgrid control system could potentially require a complete system redesign, since there could be little similarity between the simulation system and actual hardware in the microgrid in terms of computation capability, communication method, and implementation details.

A more advanced approach, which provides a more realistic test environment for distributed microgrid control algorithms, uses dedicated hardware, separate from the power system simulator, with its own computation and communication capabilities [4]–[9]. In [4], a Mamba single board computer is interfaced with RTDS to collect simulated measurement data and implement communication-based control algorithms such as load balancing, fault detection, and fault isolation. In [6], a Zigbee-based MAS enables communication and coordination between different controllers. In [7], Raspberry PI single board computers are interfaced to different local control units to enable microgrid real-time state estimation, and the communication between Raspberry PIs is based on TCP/IP. In [8], [9], a system on chip (SoC) with two cores are used for microgrid control. While one core controls the inverter hardware, the second core is dedicated to communication using TCP/IP protocol. In [4]–[9], dedicated controller hardware enables a close-to-reality emulation of communication-based control implementation; however, the proposed platforms do not provide a systematic approach to deliver critical services that are orthogonal to the actual application logic including time synchronization, messaging middleware, consensus and coordination mechanisms, discovery and deployment mechanisms, fault-detection and recovery mechanisms, and distributed security mechanisms. Moreover, the code portability is another issue as the control algorithms are implemented assuming a specific controller hardware platform.

MAS software platforms provide an agent-based programming paradigm with various platform services. The JADE (Java Agent Development Framework) platform, as one of the most widely used MAS platforms, is employed in [13], [14] to implement microgrid control algorithms. The controller hardware for the JADE agent can be any computer that runs Java virtual machine. In [15], the platform is augmented with support for IEC 61850 and DDS (Data Distribution Service). Despite the significant convenience provided by the JADE platform in terms of agent deployment and communication, it lacks support for applications specific to the power domain. For example, the JADE platform does not explicitly support many communication protocols needed to interact with practical power system devices. Individual extensions are needed to support each protocol as described in [15]. Also, the JADE platform is not suitable for time-sensitive control tasks due to the lack of real-time functionality in the platform service. Furthermore, the absence of synchronization among agents makes the platform unsuitable for applications where actions need to be taken by different agents simultaneously. Additionally, JADE is designed for implementing MAS algorithms. If a different form of interactions between the nodes is required, the platform is not well-suited for the task.

In addition to the open-source microgrid control platforms, commercial platforms are widely utilized for microgrid control. In [10]–[12], dSPACE simulator is used as hardware and Matlab/Simulink as software platform to implement both power electronics control and communication-based control

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2952803, IEEE Transactions on Industrial Electronics

TABLE I: Summary of the RIAPS Features

	RIAPS features	Description
Tool Suite	Component Framework	Reuse code for different apps
	Deployment Service	Deploy apps to hardware
Runtime Services	Comm. Framework	Support various comm. patterns including pub-sub, req-rep
	Hard Real-Time	Enable real-time operation
	Time Synchronization	Enable synchronous operation
	Device Interface	Support various comm. protocols including Modbus, C37.118, etc.
	Fault Management	Improve resiliency and reliability
	Cybersecurity	Protect apps from cyber attacks

algorithms for an experimental microgrid. Similar implementation using CompactRIO and LabVIEW form National Instruments can be found in [16], [17], [20]. Although commercialized solutions provide a mature development environment with enhanced capabilities such as support for remote deployment and various communication protocols, one major issue is that they are cost-prohibitive in many applications. Further, the proprietary nature of the solutions limits the ability of the end user to add functionality to the platform, and complicates the interoperability between components from different vendors.

The RIAPS platform, proposed in this paper, aims to address the shortcomings identified in the literature. RIAPS is an open-source software platform that helps developers implement, deploy, and execute microgrid control algorithms in hardware controllers. The hardware controller can be any processor capable of running Linux. In this work, we use the Bealgebone Black (BBB) single board computer as the hardware controller due to its low cost and open-source nature. The RIAPS platform, on one hand, can be combined with any real time simulators such as OPAL-RT or RTDS to fast prototype control algorithms with real hardware and network traffic. On the other hand, it can be integrated with grid devices such as inverters and relays for deploying a microgrid controller in the field.

III. The RIAPS Platform

This section presents an open-source software platform, RIAPS. The RIAPS platform features a tool suite and runtime services, which are summarized in Table I. Each feature is introduced in detail in the following subsections.

A. Reusable Component Framework

The RIAPS platform has a component-based design where an application is formed by components that can be independently designed and tested, integrated with other components to form applications, deployed, and executed. A RIAPS component is the most fundamental unit that implements user-defined functions by storing its local states and by exchanging information with other components. Fig. 1 shows a general component model with five different kinds of ports: subscribe-port, publish-port, request-port, reply-port and timer-port. The subscribe-port and publish-port are used in a publish-subscribe communication pattern, while request-port and reply-port facilitate request-reply communication pattern. Timer-ports are dynamically programmed to periodically execute at a given rate or singularly at a point in time in the future, thus allowing for scheduling events like planned microgrid islanding. A component can have multiple ports of each kind.

The subscribe-port, reply-port and timer-port have associated callback functions that are triggered when the subscribed

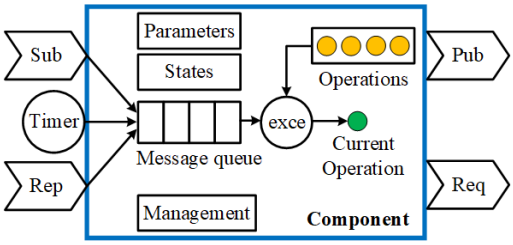


Fig. 1: RIAPS component model.

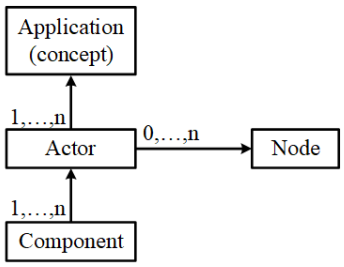


Fig. 2: RIAPS application/actor/component relationship.

message is received, the reply is received and the pre-set timer expires, respectively. Event-driven operations (e.g. power system protection) and time-driven operations (e.g. discrete control with a fixed time step) can be conveniently implemented in the callback functions that make use of RIAPS hard real-time features, is introduced in subsection III-D.

A RIAPS component is single-threaded: at most one operation (i.e. callback function) is executed at any time. This simplifies the component development, as the designer would not need to deal with complex synchronization issues – it is handled through message interactions among RIAPS components. While single-threaded components are easy to understand and implement, they are not suitable for implementing communication protocols that require asynchronous interactions with the physical world, i.e. power system devices. For this reason, the RIAPS platform supports a special class of component called "device component" that is multi-threaded and interacts with power system devices. Such components are expected to be developed with attention paid to concurrency issues. Device components are introduced in detail in subsection III-F.

RIAPS components that serve the same function are grouped and defined as a RIAPS actor for three reasons:

- An actor provides its components with access to the RIAPS platform services including time synchronization, resource discovery, and distributed coordination.
- Remote deployment (installation, start, stop, and removal) of components is done through actors. Instead of individual components, actors are deployed to the physical nodes on which the RIAPS platform resides.
- Each actor is an operating system process, which improves the run-time efficiency as it allows several components to run within the memory space of a single process. This is important as the hardware controller for RIAPS nodes may be single board computers with limited computational capability, e.g. the BBBs used in this work. Note that components within one actor are executed concurrently, i.e. an actor is multi-threaded if it contains more than one component.

One RIAPS application can contain multiple actors on dif-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2952803, IEEE Transactions on Industrial Electronics

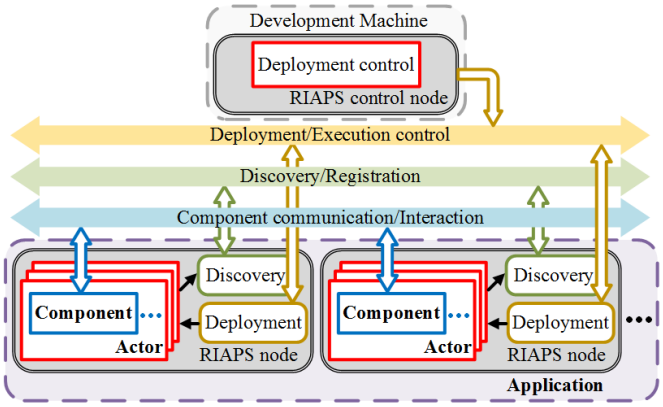


Fig. 3: RIAPS platform architecture overview.

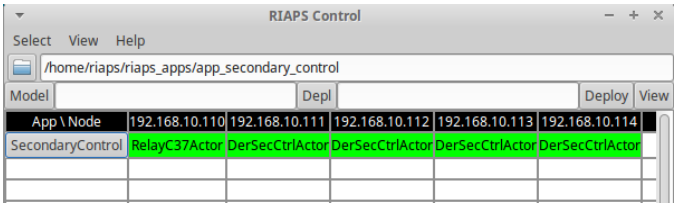


Fig. 4: RIAPS deployment service GUI deploying a secondary control application to five nodes. Relay C37 actor is deployed to the RIAPS node with internal IP 192.168.10.110. DER secondary control actor is deployed to four RIAPS nodes with internal IP from 192.168.10.111 to 192.168.10.114.

ferent nodes with one common objective. The relationship between RIAPS application, actor, and component is shown in Fig. 2. A component can be reused in several applications, while being packaged into various actors. Furthermore, the same actor can be deployed to multiple RIAPS nodes. This makes the RIAPS platform ideal for distributed control algorithms, as the controllers on different agents in a distributed control algorithm may be similar.

B. Deployment Service

RIAPS applications will have actors running across multiple RIAPS nodes that can be geographically distant from each other. As such, a robust application deployment and management service is crucial. The goal of the deployment service is to provide an interface to manage applications on the network. The assumption is that each RIAPS node is accessible from a control node (e.g. a node in the control room) via an IP network as shown in Fig. 3. The control node is responsible for the installation, start, stop, and removal of applications on the RIAPS nodes.

The RIAPS platform provides a graphical user interface (GUI) to interact with the deployment service. An example of the GUI is shown in Fig. 4, which deploys an application called SecondaryControl to five RIAPS nodes.

C. Discovery Service and Communication Framework

Components and thus actors can communicate with each other through their publish-ports, subscribe-ports, request-ports and reply-ports. However, in distributed software frameworks like RIAPS, it is possible that the publish, subscribe, request and reply ports do not know the identity or location of the other party, yet they have to communicate via point-to-point protocols like TCP/IP. The discovery service assists with setting up the network connections among actors, as shown in Fig. 3. The main use cases of the discovery service are:

- **Actors in one application:** When actors in one application are deployed and started, they contact the discovery service and inform it of (1) what message types they publish and/or subscribe to, and (2) what services they offer and require. The discovery service keeps track of this information in its internal database. The discovery service also informs the actors about potential matches and establishes the connections between actors.
- **Actors in multiple applications:** In some cases, different applications may need to communicate directly. Data are shared among applications using the publish-subscribe communication pattern. The discovery service facilitates the setting up of connections between actors that belong to different applications.
- **Fault tolerance:** The discovery service detects whether an actor is alive or not and informs other interested actors. The discovery service itself is fault-tolerant since it exists in multiple copies on the network; if one or more nodes in the network go down, the discovery service is still functional. When a discovery service restarts, it may retrieve its state from persistent storage and/or the discovery service community.

Ports dictate the interaction patterns (publish-subscribe and request-reply) that may be used in applications. However, ports alone do not determine the communication protocols that are used to implement these interaction patterns. TCP/IP protocol is used as the backbone for the RIAPS communication framework, since it is widely accepted and available for most intelligent electronic devices (IEDs) and grid devices. Built upon TCP/IP, many protocols such as DDS, MQTT, and ZeroMQ are available as messaging middleware. ZeroMQ is used by the RIAPS platform as the messaging middleware, because it is message based and has embedded peer-to-peer communication, security and discovery features with good community support [24].

D. Hard Real-Time Features

A key requirement for a microgrid control system is the ability to respond to time sensitive events such as responding to incoming requests or dispatching an action when an issue is found in the power grid. Commonly, hardware systems are able to provide the required accuracy and response time, since the processors are fast enough and the timers have acceptable precision. The missing real-time behavior is software related: the code execution needs to be scheduled with high accuracy. This requires the ability to minimize the scheduling interrupts that can occur within the kernel code. The RIAPS platform uses a well established and trusted technology, Embedded Linux, as its operating system. The Linux kernel is configured with the PREEMPT_RT feature to make it preemptible, achieving real-time behaviors by reducing the amount of time during which high-priority operations are blocked.

E. High-Precision Time Synchronization

The primary function of the RIAPS time synchronization service is to align the system clock of every RIAPS node to a common reference clock with minimum jitter and latency. The time synchronization architecture for the RIAPS platform

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2952803, IEEE Transactions on Industrial Electronics

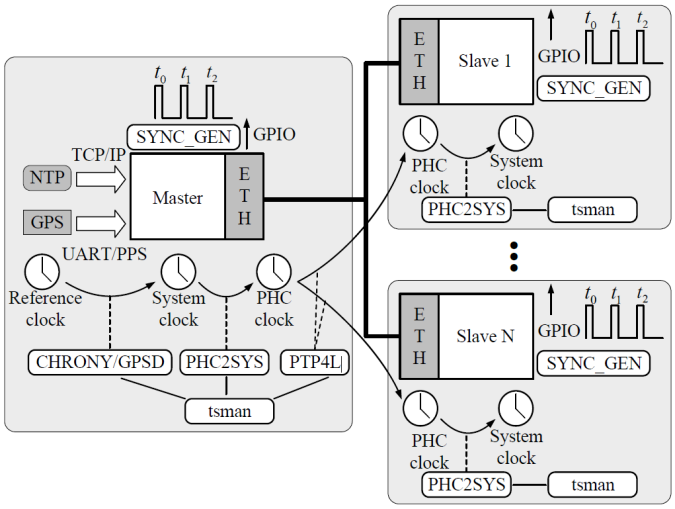


Fig. 5: Synchronization architecture of the RIAPS platform.

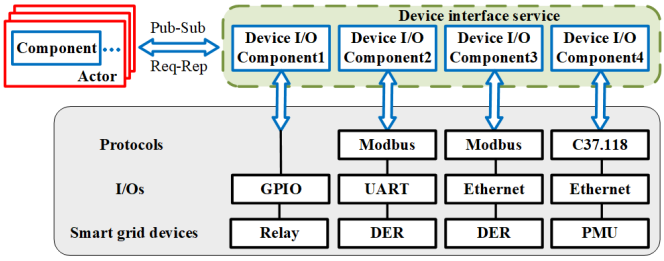


Fig. 6: An example of four different device I/O components: component 1 can interact with relay through a GPIO channel; component 2 can interact with DER through Modbus RTU protocol; component 3 can interact with DER through Modbus TCP/IP protocol; component 4 can interact with PMU through C37.118 protocol. Depending on the devices the node interacts with, one or more device I/O components can be integrated into the application.

is shown in Fig. 5. The synchronization service uses a master-slave approach where a master node is synchronized to the reference clock – a GPS receiver, network time, or the master's own system clock. The slave nodes synchronize to the master node by Precision Time Protocol (PTP) and a RIAPS-specific configuration management tool (tsman).

The secondary objective of the RIAPS time synchronization service is to provide easy-to-integrate and reliable time synchronization services for external hardware. A hardware-specific tool, `sync_gen`, was developed on top of the RIAPS time synchronization service to generate accurate synchronization pulses on the selected BBB general purpose input output (GPIO) pin in a wide-range of frequencies. The pulses are aligned to the globally established time-scale; thus, they are generated at the same time instant on all nodes. To minimize the time jitter and bias from the RIAPS node's system clock to its GPIO assertion, various measures are taken:

- Instead of relying on user-space libraries (e.g. *libsoc*) and kernel services (*sysfs*), the service uses direct memory-mapped I/O access for driving the GPIO pins.
- The `sync_gen` service is scheduled with real-time policy (`SCHED_FIFO`) at the highest priority.
- The timing of the pulse signal does not rely on the sleep services (e.g. *clock_nanosleep*). Instead, well before the pulse signal is due, the process wakes up and uses a busy-wait loop (continuously checking the system time) to find the best moment to assert the pulse signal.

F. Device Interface Service

Various microgrid components such as DGs, relays, PMUs, etc. use a wide variety of communication protocols and physical links to communicate with a supervisory controller. For example, IEEE Std. 1547 [25] defines Modbus and DNP3 as standard communication protocols for inverters while C37.118 is a common protocol for PMUs [26]. Additionally, IEC 61850 defines a number of protocols (e.g. GOOSE) for communication with IEDs such as relays [27]. The fact that there being no standard protocol for microgrid devices necessitates the RIAPS device interface service.

The aim of the RIAPS device interface service is to encapsulate specific I/O devices so that they are accessible to application components using a unified interface. In other words, the device interface service encapsulates the grid devices into a special class of RIAPS components called device I/O components. The specific ports and interfaces implemented by these device I/O components are specific to the connected grid device, the protocol used, and the physical link used. The I/O components of many devices are available in the RIAPS platform as library components. They can be configured and deployed to communicate with grid devices directly without concerning the implementation details of the underlying communication protocols. While device I/O components exchange information with grid devices, other application components communicate with device I/O components using the standard RIAPS interaction patterns such as publish-subscribe and request-reply. This approach allows the developers to focus on the applications rather than on defining interfaces between the RIAPS platform and the grid devices. An example of four different types of device I/O components is shown in Fig. 6.

G. Fault Management and Resiliency

As a software platform, the RIAPS platform resiliency is defined as the built-in fault tolerance capability of the RIAPS *software*. Resiliency may have a different definition for power system but it is referred to as *software* resiliency in this paper. The RIAPS platform fault management operates at two levels: 1) platform service level: the fault management at this level is responsible for fault detection and recovery for the discovery, deployment and time synchronization services. The platform service failures can be detected and corresponding services can be restarted based on the rules specified by the application developer; 2) application level: the application level fault management is present both inside and outside the actor. For internal fault management, the actor is responsible for detecting exceptions generated by the component's code. Messages about the exceptions are sent through a fault message bus and the developer is able to define handlers for such messages. The fault management outside the actor is responsible for detecting the crash of the actor process and also restarting the actor process if so specified by the developer. For example, if a communication link among components fails, the platform will detect the failure and try to reestablish the link. At the same time, the running application is notified and the handler defined for communication failure in the code is activated. Further, the hardware resource usage such as CPU, memory,

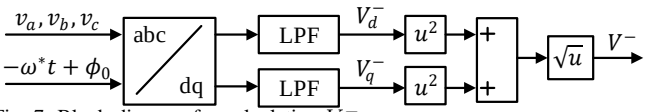


Fig. 7: Block diagram for calculating V^-.

and network bandwidth utilization is monitored. Violations are detected and reported to the running application.

H. Cybersecurity

The RIAPS platform addresses cybersecurity from several aspects. First, application packages are encrypted and cryptographically signed when deployed to the hardware controller from the development machine. Second, the service registry is protected by elliptic curve encryption with dynamically generated keys. Third, all internal communications within an application are protected by the same method. Additionally, application processes run under their unique, dynamically generated user ids, within their own separated and isolated directories. Application processes can communicate only with authorized nodes and they are strictly controlled using a non-bypassable Mandatory Access Control mechanism.

IV. RIAPS Microgrid Control Application

In this section, we present a distributed microgrid secondary control algorithm capable of synchronized and proportional compensation of system unbalance using coordinated DGs. The purpose of presenting this application is to showcase how the RIAPS platform features, which were described in the previous section, allow for the implementation of distributed, time-sensitive algorithms on multiple hardware nodes.

A. Problem Statement

In an islanded AC microgrid, droop control (1) is commonly used as primary control for dispatchable DGs to share microgrid load without the need for communication:

$$\omega_i = \omega^* - m_i(P_i - P_i^*) \tag{1a}$$

$$E_i = E^* - n_i(Q_i - Q_i^*) \tag{1b}$$

where ω^* and E^* are the nominal frequency and voltage magnitude, respectively; m_i and n_i are the frequency and voltage droop gains of DGi, respectively; P_i and Q_i are the active and reactive power output, respectively; P_i^* and Q_i^* are the active and reactive power setpoint, respectively; ω_i and E_i are the frequency and voltage magnitude reference for capacitor voltage, respectively.

Droop control stabilizes the system voltage and frequency; however, it introduces steady state frequency and voltage deviations if the setpoints P_i^* and Q_i^* are not equal to the actual power output P_i and Q_i, respectively. To eliminate such deviations, secondary control is used to adjust the setpoints:

$$\omega_i = \omega^* - m_i(P_i - P_i^*) + \Omega_i \tag{2a}$$

$$E_i = E^* - n_i(Q_i - Q_i^*) + e_i \tag{2b}$$

where Ω_i and e_i are the frequency and voltage secondary control variables, respectively.

If unbalanced loads exist in the microgrid, the voltage in the microgrid can be unbalanced. The degree of unbalance is measured by voltage unbalance factor (VUF):

$$\text{VUF} = V^-/V^+ \tag{3}$$

where V^- is the magnitude of fundamental negative sequence voltage and V^+ is the magnitude of fundamental positive sequence voltage. The fundamental negative sequence voltage magnitude can be calculated in any dq reference frame rotating at $-\omega^*$ as shown in Fig. 7. Depending on its phase lock loop angle, different DGs' local reference frames can have different initial angles ϕ_0. Thus, given the same input v_a, v_b, v_c, each DG could come out with different V_d^- and V_q^- if using local reference frames, while the magnitude V^- remains the same.

To compensate the voltage unbalance and thus reduce the VUF in the microgrid, the DGs can actively inject negative sequence current [28], [29]. While the magnitude of the injected negative sequence current I_i^- is a measure of the DGi's compensation effort for voltage unbalance, the d and q components of the injected negative sequence current $i_{d,i}^-$ and $i_{q,i}^-$ depend on the phase angle of the negative sequence voltage of the selected compensated bus, $\theta_{i,\text{BUS}n}^-$. The injected negative sequence current can be calculated as,

$$i_{d,i}^- = I_i^- \cos\theta_{i,\text{BUS}n}^- \ , \qquad i_{q,i}^- = I_i^- \sin\theta_{i,\text{BUS}n}^- \tag{4a}$$

$$\theta_{i,\text{BUS}n}^- = \arccos \frac{V_{d,i,\text{BUS}n}^-}{\sqrt{\left(V_{d,i,\text{BUS}n}^-\right)^2 + \left(V_{q,i,\text{BUS}n}^-\right)^2}} \tag{4b}$$

where $V_{d,i,\text{BUS}n}^-$ and $V_{q,i,\text{BUS}n}^-$ are the d and q component of negative sequence voltage of the compensated bus in DGi's local dq reference frame, respectively.

Commonly, a DG only has voltage sensors installed locally and does not have access to the voltage measurement of the external bus that needs unbalance compensation. Furthermore, the communication bandwidth is usually not high enough to stream the measured instantaneous three-phase bus voltage to the DG. For these reasons, the control strategies in [28] and [29] use the DG's local negative sequence capacitor voltage as the reference vector to calculate the injected negative sequence current. However, if the compensated bus is electrically remote from the DG (i.e. there is a large impedance in between), the negative sequence capacitor voltage and the negative sequence bus voltage can be very different, resulting in an unsatisfactory compensation performance for the voltage unbalance at the remote bus. Thus, there is a need for a globally synchronized dq reference frame. With that, the negative sequence voltages at the remote bus can be measured as DC quantities in the globally synchronized dq reference frame and transmitted to the DG using low speed communication channels. After converting the negative sequence voltages from the globally synchronized dq reference frame to the DG's local reference frame, (4) is used to calculate the injected negative sequence current and improve the unbalance compensation performance. Subsection V-C describes how such a globally synchronized dq reference frame is created using the RIAPS platform.

B. Proposed Algorithm

To demonstrate the distributed control capability of the RIAPS platform, we use a distributed secondary controller to recover the system frequency and voltage to their nominal values and regulate VUF at a remote bus, while sharing the power and unbalance compensation effort proportionally

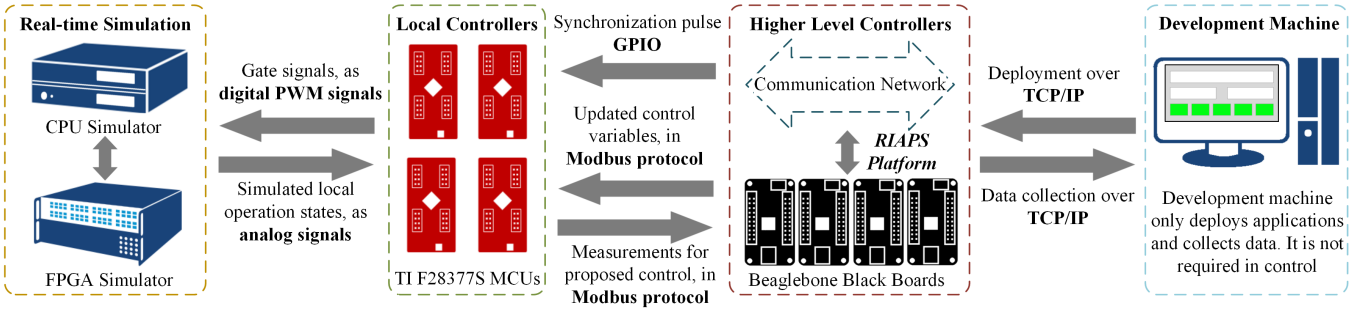


Fig. 8: Hardware-in-the-loop testbed setup.

among all DGs. The proposed secondary control is based on consensus algorithm and it is expressed as:

$$k_i \frac{d\Omega_i}{dt} = -\alpha_i(\omega_i - \omega^*) - \sum_{j=1}^{n} a_{ij}\left(\frac{P_i}{P_i^r} - \frac{P_j}{P_j^r}\right) \tag{5a}$$

$$\kappa_i \frac{de_i}{dt} = -\beta_i(\bar{E} - E^*) - \sum_{j=1}^{n} b_{ij}\left(\frac{Q_i}{Q_i^r} - \frac{Q_j}{Q_j^r}\right) \tag{5b}$$

$$\eta_i \frac{dI_i^-}{dt} = -\gamma_i(\mathrm{VUF}_{\mathrm{BUSn}} - \mathrm{VUF}^*) - \sum_{j=1}^{n} c_{ij}\left(\frac{I_i^-}{I_i^{-r}} - \frac{I_j^-}{I_j^{-r}}\right) \tag{5c}$$

where k_i, κ_i and η_i are the frequency, voltage and unbalance compensation secondary control gain, respectively; α_i, β_i and γ are the frequency, voltage and VUF regulation gain, respectively; E^* is the microgrid rated voltage and $\bar{E}$ is the microgrid average voltage whose value can be estimated using a distributed method such as the one from [30]. P_i^r and Q_i^r are the rated active and reactive power for DGi, respectively. $\mathrm{VUF}_{\mathrm{BUSn}}$ and VUF^* are the measured VUF at compensated bus and the required VUF, respectively. I_i^{-r} is the desired share of unbalance compensation effort for DGi. Adjacency matrix $A = \{a_{ij}\}$, $B = \{b_{ij}\}$ and $C = \{c_{ij}\}$ depend on the microgrid communication topology: $a_{ij} = b_{ij} = c_{ij} = 1$ if there is a valid communication link between DG i and DG j, otherwise $a_{ij} = b_{ij} = c_{ij} = 0$.

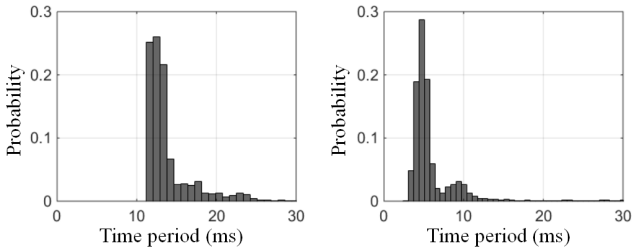
It is worth mentioning that the frequency and voltage secondary controller in (5a) and (5b) do not require synchronization between different nodes. However, synchronization is necessary for creating a globally synchronized dq reference frame for measuring and compensating the voltage unbalance at a remote location.

V. Experiment Results

To demonstrate the salient features of the RIAPS platform, we present a hardware-in-the-loop (HIL) testbed using industry-grade hardware and implement the proposed microgrid control using the RIAPS platform. The performance of the RIAPS platform and the effectiveness of the proposed microgrid control is validated through experiments.

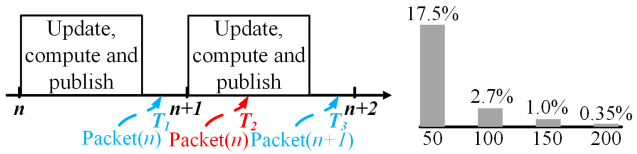
A. Hardware-in-the-loop Testbed

Fig. 8 presents an overview of the HIL testbed. The testbed uses an OPAL-RT real-time simulator to model the microgrid and and its components including DGs, loads, relays, and etc. The switching power electronics converters are modeled in OPAL-RT FPGA-based simulator with 500 ns simulation time step, which allows capture of fast dynamics of switching converters in real-time. The non-switching components are modeled in the CPU-based simulator. The converter models in the simulator are interfaced with TMS320F28377S microcontroller units (MCUs) from Texas Instruments. The simulated states like current and voltage are output as analog signals and scaled down to a suitable level for the analog-to-digital converters (ADCs) of the MCUs. The power electronics converters' local control algorithms are implemented in the MCUs and they output pulse-width modulated (PWM) signals as the gate signals for the simulated converters.



(a) Modbus Protocol (b) RIAPS platform

Fig. 9: Distribution of different delay components.



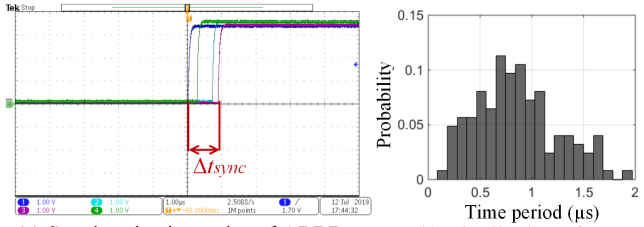
(a) Algorithm execution process (b) Packet loss rate VS time step (ms)

Fig. 10: Packet loss rate measurement

The RIAPS platform is deployed as a high level control platform in the testbed. Each MCU is associated with a RIAPS node whose hardware is BBB. With RIAPS device interface service, each BBB can interact with its MCU using Modbus protocol to access measurement data as well as issue commands to the MCU. Further, a GPIO channel between each BBB and its MCU enables synchronizing different MCUs in the testbed.

The HIL testbed is developed to evaluate the performance of the RIAPS platform and microgrid control algorithm. Besides OPAL-RT real-time simulator, the RIAPS platform can also be integrated with other real-time simulators such as RTDS, as well as real power stage hardware such as inverters. It allows a rapid control prototyping as the developed control algorithm are tested on hardware controllers. Moreover, it can be easily extended to test the performance of other components. For example, an inverter and its RIAPS node can be integrated into the testbed using the method in [31]. Also, the commercialized

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2952803, IEEE Transactions on Industrial Electronics



(a) Synchronization pulse of 4 BBBs (b) Distribution of Δt_{sync}

Fig. 11: Synchronization error and its distribution.

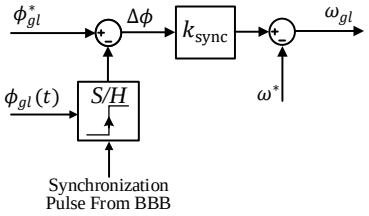


Fig. 12: Block diagram for tuning ω_{gl} in synchronization ISR.

relays can communicate with a RIAPS node using C37.118 protocol and be tested in the system.

B. RIAPS Platform Performance

In this subsection, the performance of the RIAPS platform is measured in terms of communication delay, communication packet loss rate and synchronization error.

1) Communication delay: Fig. 9 shows the distribution of communication delays measured in the testbed. The average delay for Modbus communication between the BBB and DSP is 14 ms while the average communication delay between two BBBs is 7 ms. The communication delay between two BBB may increase as the number of the RIAPS nodes becomes larger due to increased network traffic.

2) Packet loss rate: In each time step, the BBB first updates the measurement, then computes the algorithm and lastly publishes the calculated results. If the time step is not large enough, the published results may not be received by other BBBs before the next time step. This process is shown in Fig. 10(a). If the packet for time step n arrives at T_1, it is used for calculation in the next time step. However, if it arrives at T_2, it is not used and the information in this packet is lost. Fig. 10(b) shows the packet loss rate measured using different time steps. While a large time step gives a lower packet loss rate, it may degrade the controller performance. As a trade-off, 100 ms time step is selected in the following experiment.

3) Synchronization error: Fig. 11(a) shows the rising edge of the synchronization pulses generated by four BBBs. The synchronization error is marked as Δt_{sync}. The test is repeated 120 times and the distribution of Δt_{sync} is shown in Fig. 11(b). The average synchronization error is 0.85 μs and the maximum synchronization error is 1.96 μs.

C. Globally Synchronized dq Reference Frame

The proposed unbalance compensation control requires the knowledge of negative sequence voltage at the compensated bus which poses challenges as the DGs' controllers do not have a common clock reference. To overcome this issue we create a globally synchronized dq reference frame at all nodes using RIAPS' high accuracy time synchronization service.

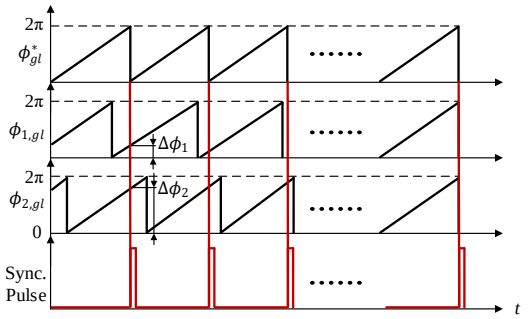


Fig. 13: Two MCUs' global dq reference frame synchronizing gradually.

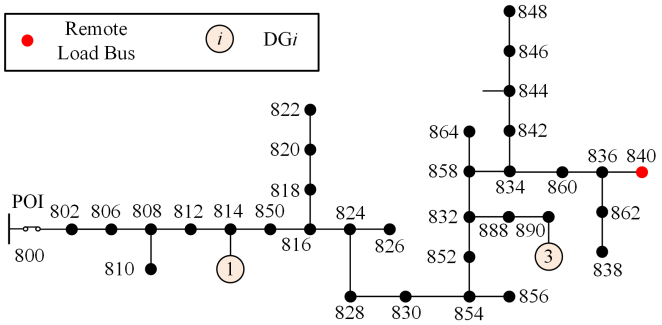


Fig. 14: Simulated IEEE 34-bus system for proposed secondary control.

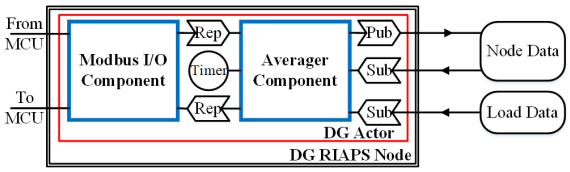


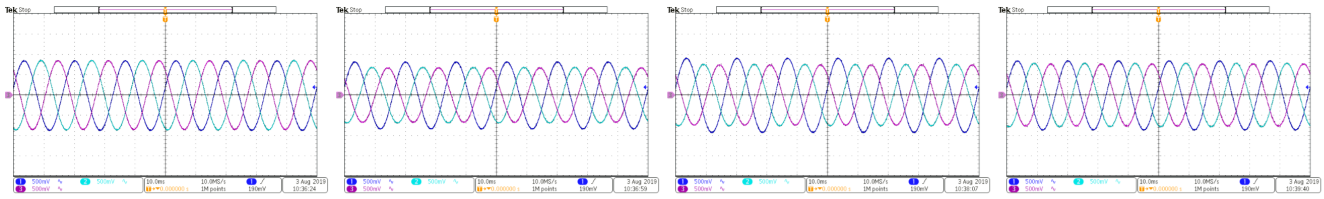
Fig. 15: RIAPS application for distributed microgrid secondary control.

The goal of globally synchronized dq reference frame is to have the same global reference frame phase ϕ_{gl} on all the MCUs while maintaining ω^* rotating frequency. This is accomplished by two steps. First, in the main PWM interrupt service routine (ISR), global reference frame phase ϕ_{gl} is calculated by integrating global reference frame frequency ω_{gl} over one switching period T_{sw},

$$\phi_{gl}(n+1) = \phi_{gl}(n) + \omega_{gl} T_{sw} \tag{6}$$

The second step is to tune ω_{gl} such that all the MCUs can have the same phase ϕ_{gl}. A GPIO channel sends synchronized pulse from BBB to its MCU. The synchronized pulse triggers a synchronization ISR in each MCU. A phase controller shown in Fig. 12 is implemented in the ISR. As the synchronization pulse is generated by all BBBs at the same time instant (with μs tolerance), all MCUs enter the ISR at the same time and the sampled global reference frame phase ϕ_{gl} should be the same on all MCUs. If not, a proportional controller is used to tune the frequency slightly around its nominal value. Therefore, an MCU with leading phase will have a smaller ω_{gl} while an MCU with lagging phase will have a larger ω_{gl} until the next synchronization pulse. Thus, the phase difference can be eliminated gradually. Fig. 13 shows the synchronization process of two MCUs' global reference frame. The synchronization ISR has a higher priority than the main ISR to ensure accurate sampling of the global reference frame phase ϕ_{gl}.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2952803, IEEE Transactions on Industrial Electronics



(a) Grid-connected operation ($t < t_2$) (b) Islanded operation without secondary control ($t_2 < t < t_3$) (c) Islanded operation with frequency and voltage secondary control ($t_3 < t < t_4$) (d) Islanded operation with frequency, voltage and unbalance secondary control ($t_4 < t$)

Fig. 16: Test results for distributed microgrid secondary control: three-phase voltage waveform at Bus 840

TABLE II: Microgrid Parameters

DG Parameters		
$DG_{1,2,3,4}$	$m = 3.1416 \times 10^{-5}$	$n = 1 \times 10^{-4}$
	$f_{sw} = 5000$ Hz	$k_{sync} = 5$
Secondary Controller Parameters		
Frequency control	$k = 0.16$	$\alpha = 0.12$
Voltage control	$\kappa = 0.016$	$\beta = 0.028$
Unbalance control	$\eta = 0.02$	$\gamma = 0.35$
Time step	$\Delta t = 100$ ms	
Synchronization Parameters		
Sync. pulse frequency	$f_{sync} = 60$ Hz	

D. Case Study

To verify the effectiveness of the proposed control algorithm and the performance of the RIAPS platform as a microgrid controller, we adopt IEEE 34-bus system as a single microgrid and implement three changes from the nominal system. First, four DGs are introduced at bus 814, 822, 890, 844. Second, all the single phase loads are replaced with three phase balanced load with a floating neutral to avoid zero sequence current. Third, the phase-to-phase load in IEEE 34-bus system will cause voltage unbalance. The remote Bus 840 is expected to have critical load and its VUF should be low. The controller parameters are shown in Table II.

Each simulated DG is controlled by an MCU in the testbed. The control implemented in the MCU includes an inner current loop to regulate the inverter current, an outer voltage loop to regulate the filter capacitor voltage, and a droop loop (2). The negative sequence current injection (4) is also implemented in the MCU. An additional MCU measures the negative sequence voltage using the globally synchronized dq reference frame and VUF at Bus 840. This information is shared with other DGs for unbalance compensation control.

The RIAPS application for distributed microgrid secondary control is shown in Fig. 15. The DG actor is deployed to four DG nodes. A Modbus I/O component sends and receives Modbus messages from the MCU. The secondary controllers (5) are implemented in component **Averager**. **Averager** has a timer-port that triggers its callback functions every time step Δt. In the callback function, (5) is discretized and the control variables Ω_i, e_i and I_i^- are calculated. Local information such as P_i/P_i^r, Q_i/Q_i^r and I_i^-/I_i^{-r} is packed and published to the RIAPS communication network under the topic **Node Data**. By default, the RIAPS platform assumes a full communication network, i.e any DG RIAPS node can receive **Node Data** from any other DG RIAPS nodes. If a controller's performance under a sparse communication network is to be verified, the communication network can be reconfigured. All the DG nodes also subscribe to the message **Load Data** from the

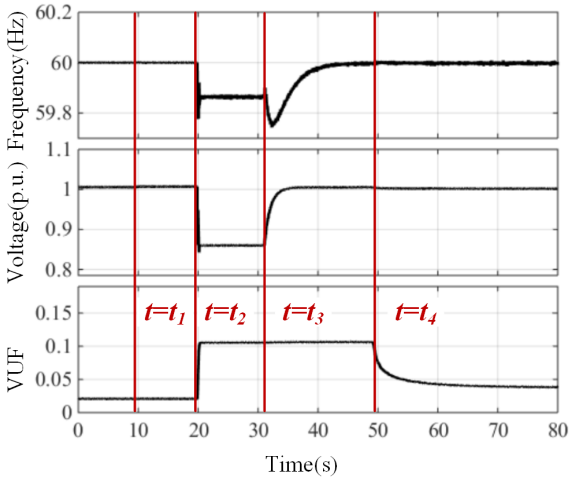


Fig. 17: Test results for distributed microgrid secondary control: frequency, voltage and VUF measured at Bus 840.

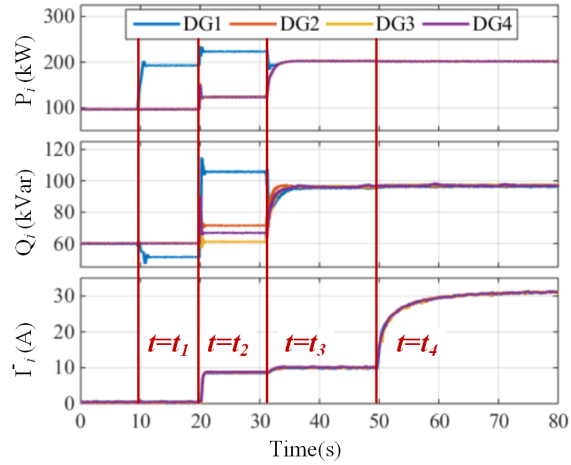


Fig. 18: Test results for distributed microgrid secondary control: active power, reactive power and negative sequence current of DGs.

remote bus node. The RIAPS remote bus node publishes VUF_{BUSn}, as well as $V_{d,gl,BUSn}^-$ and $V_{q,gl,BUSn}^-$ calculated in the globally synchronized dq reference frame.

The HIL test results are presented in Fig. 16, Fig. 17 and Fig. 18. Initially, the microgrid operates in grid-connected mode with the point of interconnection (POI) relay closed. All the DGs output 100 kW active power and 60 kvar reactive power. At $t = t_1$ a dispatch command is sent from the microgrid control system to DG1. Based on the command, the output of DG1 is changed to 200 kW active power and 50 kvar reactive power. The three phase voltage waveform at Bus 840 is shown in Fig. 16(a). As the grid is a stiff voltage source, the voltage waveform at Bus 840 shows little unbalance during

grid-connected operation.

At $t = t_2$, the microgrid experiences an unintentional islanding event and the POC relay opens. The microgrid loads are supplied by the DGs in the microgrid. During $t = t_2$ to $t = t_3$, the microgrid operates in islanded mode and the secondary control application is not activated. The system frequency and voltage deviation introduced by droop control can be observed: $|\Delta V_{SS}| = 0.14$ p.u. and $|\Delta f_{SS}| = 0.002$ p.u. The voltage waveform at Bus 840 in Fig. 16(b) shows a reduced voltage magnitude and significant voltage unbalance. The VUF at this stage is 0.11.

At $t = t_3$, the frequency and voltage secondary control (5a) and (5b) is enabled, while unbalance secondary control (5c) remains disabled. The microgrid frequency and magnitude deviation is eliminated gradually. At $t = 45$ s the microgrid frequency and magnitude are restored to their nominal values, respectively. However, the voltage at Bus 840 is still highly unbalanced as shown in Fig. 16(c) and the VUF is 0.11.

At $t = t_4$, the proposed unbalance secondary control (5c) is further enabled. The DGs start to compensate the voltage unbalance by injecting negative sequence current. After around 10 seconds, the VUF at Bus 840 is regulated to below 0.04. The voltage waveform with all controllers enabled is shown in Fig. 16(d). Fig. 18 shows the active power, reactive power and injected negative sequence current are shared equally by all DGs in steady state.

This application demonstrates that the RIAPS platform serving as a micorgrid control system can provide all the necessary functionalities for both microgrid grid-connected and islanding operation. It also supports advanced features such as distributed control and time synchronization.

VI. Conclusion

We presented an open-source software platform, RIAPS, for microgrid control. The platform provides a design-time toolsuite and run-time environment for the design, implementation, testing, and deployment of microgrid control systems with enhanced capabilities. It supports the development of distributed applications and provides time synchronization, real-time scheduling of operations, and fault tolerance. Further, we presented a distributed microgrid secondary control algorithm capable of synchronized and proportional compensation of system voltage unbalance. Besides its the MAS-type functionality, this application demonstrates the platform's synchronization functionality which is exploited to establish a globally synchronized reference frame.

References

[1] "IEEE standard for the specification of microgrid controllers," *IEEE Std 2030.7-2017*, pp. 1–43, April 2018.

[2] F. Guo, L. Herrera, R. Murawski, E. Inoa, C.-L. Wang, P. Beauchamp, E. Ekici, and J. Wang, "Comprehensive real-time simulation of the smart grid," *IEEE Transactions on Industry Applications*, vol. 49, no. 2, pp. 899–908, 2013.

[3] A. Nelson, S. Chakraborty, D. Wang, P. Singh, Q. Cui, L. Yang, and S. Suryanarayanan, "Cyber-physical test platform for microgrids: Combining hardware, hardware-in-the-loop, and network-simulator-in-the-loop," in *Power and Energy Society General Meeting (PESGM), 2016*. IEEE, 2016, pp. 1–5.

[4] M. J. Stanovich, I. Leonard, K. Sanjeev, M. Steurer, T. P. Roth, S. Jackson, and M. Bruce, "Development of a smart-grid cyber-physical systems testbed," in *Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES*. IEEE, 2013, pp. 1–6.

[5] I. Leonard, T. Baldwin, and M. Sloderbeck, "Accelerating the customer-driven microgrid through real-time digital simulation," in *Power & Energy Society General Meeting, 2009. PES'09. IEEE*. IEEE, 2009, pp. 1–3.

[6] C.-H. Yoo, W.-J. Choi, I.-Y. Chung, D.-J. Won, S.-S. Hong, and B.-J. Jang, "Hardware-in-the-loop simulation of dc microgrid with multi-agent system for emergency demand response," in *Power and Energy Society General Meeting, 2012 IEEE*. IEEE, 2012, pp. 1–6.

[7] P. García, P. Arboleya, B. Mohamed, and A. A. C. Vega, "Implementation of a hybrid distributed/centralized real-time monitoring system for a dc/ac microgrid with energy storage capabilities," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1900–1909, 2016.

[8] H. K. M. Paredes, J. P. Bonaldo, and J. A. Pomilio, "Centralized control center implementation for synergistic operation of distributed multifunctional single-phase grid-tie inverters in a microgrid," *IEEE Transactions on Industrial Electronics*, 2018.

[9] M. Castilla, A. Camacho, P. Martí, M. Velasco, and M. M. Ghahderijani, "Impact of clock drifts on communication-free secondary control schemes for inverter-based islanded microgrids," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 6, pp. 4739–4749, 2018.

[10] L. Meng, A. Luna, E. R. Díaz, B. Sun, T. Dragicevic, M. Savaghebi, J. C. Vasquez, J. M. Guerrero, M. Graells, and F. Andrade, "Flexible system integration and advanced hierarchical control architectures in the microgrid research laboratory of aalborg university," *IEEE Transactions on Industry Applications*, vol. 52, no. 2, pp. 1736–1749, 2016.

[11] I. U. Nutkani, P. C. Loh, P. Wang, and F. Blaabjerg, "Linear decentralized power sharing schemes for economic operation of ac microgrids." *IEEE Trans. Industrial Electronics*, vol. 63, no. 1, pp. 225–234, 2016.

[12] E. A. A. Coelho, D. Wu, J. M. Guerrero, J. C. Vasquez, T. Dragicevic, C. Stefanovic, and P. Popovski, "Small-signal analysis of the microgrid secondary control considering a communication time delay." *IEEE Trans. Industrial Electronics*, vol. 63, no. 10, pp. 6257–6269, 2016.

[13] R. de Azevedo, M. H. Cintuglu, T. Ma, and O. A. Mohammed, "Multiagent-based optimal microgrid control using fully distributed diffusion strategy," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1997–2008, 2017.

[14] W. Liu, J.-M. Kim, C. Wang, W. Im, L. Liu, and H. Xu, "Power converters based advanced experimental platform for integrated study of power and controls," *IEEE Transactions on Industrial Informatics*, 2018.

[15] M. H. Cintuglu, T. Youssef, and O. A. Mohammed, "Development and application of a real-time testbed for multiagent system interoperability: A case study on hierarchical microgrid control," *IEEE Transactions on Smart Grid*, 2016.

[16] G. Turner, J. P. Kelley, C. L. Storm, D. A. Wetz, and W.-J. Lee, "Design and active control of a microgrid testbed," *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 73–81, 2015.

[17] L. Yang, Y. Ma, J. Wang, J. Wang, X. Zhang, L. M. Tolbert, F. Wang, and K. Tomsovic, "Development of converter based reconfigurable power grid emulator," in *Energy Conversion Congress and Exposition (ECCE), 2014 IEEE*. IEEE, 2014, pp. 3990–3997.

[18] R. H. Lasseter, J. H. Eto, B. Schenkman, J. Stevens, H. Vollkommer, D. Klapp, E. Linton, H. Hurtado, and J. Roy, "Certs microgrid laboratory test bed," *IEEE Transactions on Power Delivery*, vol. 26, no. 1, pp. 325–332, 2011.

[19] B. Zhao, X. Zhang, and J. Chen, "Integrated microgrid laboratory system," *IEEE Transactions on power systems*, vol. 27, no. 4, pp. 2175–2185, 2012.

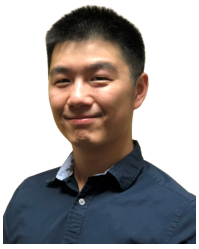
[20] M. Starke, B. Xiao, G. Liu, B. Ollis, P. Irminger, D. King, A. Herron, and Y. Xue, "Architecture and implementation of microgrid controller," in *Innovative Smart Grid Technologies Conference (ISGT), 2016 IEEE Power & Energy Society*. IEEE, 2016, pp. 1–5.

[21] C. Wang, X. Yang, Z. Wu, Y. Che, L. Guo, S. Zhang, and Y. Liu, "A highly integrated and reconfigurable microgrid testbed with hybrid distributed energy sources," *IEEE Transactions on Smart Grid*, vol. 7, no. 1, pp. 451–459, 2016.

[22] K. Zhu, M. Chenine, and L. Nordstrom, "Ict architecture impact on wide area monitoring and control systems' reliability," *IEEE transactions on power delivery*, vol. 26, no. 4, pp. 2801–2808, 2011.

[23] W. Li, M. Ferdowsi, M. Stevic, A. Monti, and F. Ponci, "Cosimulation for smart grid communications," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2374–2384, 2014.

[24] M. Starke, A. Herron, D. King, and Y. Xue, "Implementation of a publish-subscribe protocol in microgrid islanding and resynchronization with self-discovery," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 361–370, Jan 2019.
[25] "IEEE standard for interconnection and interoperability of distributed energy resources with associated electric power systems interfaces," *IEEE Std 1547-2018 (Revision of IEEE Std 1547-2003)*, pp. 1–138, April 2018.
[26] "Ieee standard for synchrophasor data transfer for power systems," *IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1–53, Dec 2011.
[27] R. E. Mackiewicz, "Overview of iec 61850 and benefits," in *2006 IEEE Power Engineering Society General Meeting*, June 2006, pp. 8 pp.–.
[28] P.-T. Cheng, C.-A. Chen, T.-L. Lee, and S.-Y. Kuo, "A cooperative imbalance compensation method for distributed-generation interface converters," *IEEE Transactions on Industry Applications*, vol. 45, no. 2, pp. 805–815, 2009.
[29] M. Savaghebi, A. Jalilian, J. C. Vasquez, and J. M. Guerrero, "Autonomous voltage unbalance compensation in an islanded droop-controlled microgrid," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 4, pp. 1390–1402, 2013.
[30] G. Lou, W. Gu, J. Wang, W. Sheng, and L. Sun, "Optimal design for distributed secondary voltage control in islanded microgrids: Communication topology and controller," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 968–981, 2019.
[31] P. C. Kotsampopoulos, F. Lehfuss, G. F. Lauss, B. Bletterie, and N. D. Hatziargyriou, "The limitations of digital simulation and the advantages of phil testing in studying distributed generation provision of ancillary services," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 9, pp. 5502–5515, Sep. 2015.



Hao Tu (S'17) received his bachelor and master degree from Xian Jiaotong University, China in 2012 and RWTH Aachen University, Germany in 2015, respectively, both in electrical engineering. He is currently working towards the Ph.D. degree with the Future Renewable Electric Energy Delivery and Management Systems Center, North Carolina State University, Raleigh, NC, USA.

His research interests include control of power electronics converters, energy storage systems, and microgrids.

Yuhua Du (S'17) received the B.S degree in electrical engineering from Xian Jiaotong University, China in 2013 and Ph.D. degree in electrical and computer engineering from North Carolina State University, USA in 2019. He was a Research Aide with Argonne National Laboratory in 2018.

His research interests include voltage source converter modeling and control, microgrid distributed secondary control and development of microgrid hardware-in-the-loop testbed.

Hui Yu (S'17) received the Bachelor and Master degrees in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2013 and 2016, respectively. He is currently working toward the Ph.D. degree at the Future Renewable Electric Energy Delivery and Management (FREEDM) Systems Center, North Carolina State University, Raleigh, NC, USA.

His research interests include power electronics converter modeling, design, and control in microgrids and energy storage systems. He was a recipient of the Outstanding Presentation Award of the 2018 IEEE Applied Power Electronics Conference.

Abhishek Dubey (SM'15) is a Senior Research Scientist at the Institute for Software Integrated Systems and an Assistant Professor in the Electrical Engineering and Computer Science department at Vanderbilt University. His research interests lies at the intersection of Distributed Systems, Big Data and Cyber Physical Systems. He is especially interested in resilient system design, performance management, and failure diagnostics of smart and connected community systems, with a focus on transportation networks and smart grid. Abhishek directs the SCOPE lab at Institute for Software Integrated Systems. The lab works very closely with Vanderbilt Initiative for Smart Cities Operation and Research (VISOR).

Abhishek completed his PhD in Electrical Engineering from Vanderbilt University in 2009, received his M.S. in Electrical Engineering from Vanderbilt University in August 2005 and completed his undergraduate studies in electrical engineering from the Indian Institute of Technology, Banaras Hindu University, India in May 2001. He is a senior member of IEEE.

Srdjan Lukic (SM'19) received the Ph.D. degree in electrical engineering from the Illinois Institute of Technology, Chicago, IL, USA, in 2008, and is a Professor with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA. He serves as the Deputy Director of the National Science Foundation Future Renewable Electric Energy Delivery and Management (FREEDM) Systems Engineering Research Center, headquartered at North Carolina State University.

His research interests include design and control of power electronic converters and electromagnetic energy conversion with application to microgrids, wireless power transfer, energy storage systems, and electric automotive systems. He was a Distinguished Lecturer with the IEEE Vehicular Technology Society from 2011 to 2015.

Gabor Karsai (SM'06) received the B.Sc., M.Sc., and Dr. Techn. degrees from the Technical University of Budapest, Budapest, Hungary, in 1982, 1984, and 1988, respectively, and the Ph.D. degree from Vanderbilt University, Nashville, TN, USA, in 1988.

He is Professor of Electrical Engineering and Computer Science at Vanderbilt University and Senior Research Scientist and Associate Director at the Institute for Software-Integrated Systems. He has over 25 years of experience in research on systems and software engineering. He conducts research in the design and implementation of embedded systems, in programming tools for visual programming environments, and in the theory and practice of model-integrated computing.