Towards a Reconfigurable Distributed Testbed to Enable Advanced Research and Development of Timing and Synchronization in Cyber-Physical Systems

Hugo A. Andrade ¶, Patricia Derler ¶, John C. Eidson ‖, Ya-Shian Li-Baboud‡,
Aviral Shrivastava*, Kevin Stanton†, Marc Weiss §
¶ National Instruments, Berkeley, CA
‖ University of California, Berkeley
‡ NIST Software and Systems Division, Gaithersburg, MD
* Arizona State University
† Intel Corporation, Hillsboro, OR
§ NIST Time and Frequency Division, Boulder, CO

***Abstract*—Timing and synchronization play a key role in cyber-physical systems (CPS). Precise timing, as often required in safety-critical CPS, depends on hardware support for enforcement of periodic measure, compute, and actuate cycles. For general CPS, designers use a combination of application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs) and conventional microprocessors. Microprocessors as well as commonly used computer languages and operating systems are essentially devoid of any explicit support for precise timing and synchronization. Modern computer science and microprocessor design has effectively removed time from the abstractions used by designers with the result that time is regarded as a performance metric rather than a correctness specification or criterion.**

There are interesting proposals and avenues of research to correct this situation, but the barrier is quite high for conducting proof of concept studies or collaborative research and development. This paper proposes a conceptual design and use model for a reconfigurable testbed designed specifically to support exploratory research, proof of concept, and collaborative work to introduce explicit support for time and synchronization in microprocessors, reconfigurable fabrics, language and design system architecture for time-sensitive CPS.

Reconfigurable computing is used throughout the system in several roles: as part of the prototyping platform infrastructure, the measurement and control system, and the application system under test.

***Index Terms*—Cyber-Physical Systems, Timing and Synchronization, Reconfigurable computing, Testbed, Correct-by-construction**

I. Motivation

Timing and synchronization play a key role in cyber-physical systems (CPS). The typical CPS implements a measure, compute, and actuate cycle. To ensure stability of the resulting control loop, a fundamental requirement is to control the loop time. As CPS become more complex with multiple sensors driving the control function and possibly with multiple actuators involved, the timing becomes even more critical in ensuring precise coordination and control. For example, each application will specify the temporal relationships between the sensor data, usually requiring simultaneous sampling within some tolerance. Likewise, any resulting actuation will have similar temporal constraints.

Traditionally, sensors and actuators communicate directly with the CPS controller via analog lines, point-to-point digital links, or via a specialized data bus such as controller area network (CAN) or highway addressable remote transducer (HART) protocols. In modern designs, the spatial extent or the need for greater bandwidth has resulted in CPS with multiple controllers, involving networks for communication. The control of timing in such systems is much more difficult than in the earlier, compact systems.

For safety-critical systems, time-triggered techniques are often used and depend on hardware support for enforcement of periodic measure, compute, and actuate cycles. Time-triggered architectures are relatively inflexible and do not scale well. In addition the model is a poor match for systems with asynchronous sporadic inputs. For general CPS, designers must use a combination of ASICs or FPGAs and conventional microprocessors.

Unfortunately, modern computer science and microprocessor design have effectively removed explicit time from microprocessor hardware, operating systems and languages with the result that time is essentially a performance metric rather than a correctness specification or criterion [1]. Without the semantics and standard interfaces to specify timing requirements, it becomes costly and difficult to build a CPS with robust timing leading to a methodology where system timeliness issues are corrected through test and adjustment. This methodology results in customized, application specific designs which are expensive to maintain and commission. Furthermore, system components are less likely to be interoperable and adding or swapping components can result in

costly re-validation of timing requirements. Users of major or critical CPS systems often purchase a lifetime supply of all components since the replacement of a hardware component, e.g. a faster microprocessor, or a change in code or firmware typically results in expensive re-qualification of the system.

In a time-triggered architecture, it is necessary to ensure that the computations can be completed in the allotted time. In a CPS utilizing general purpose hardware and software, the task of ensuring correct timing is even more difficult. What is clearly needed is a systematic design methodology where the designer can explicitly specify system timing and given a target computer, software, and network environment, be able to determine whether the proposed design can actually be executed in this environment with the proposed timing. Finally if the answer is yes, then the designer should be able to compile the design into an executable form with the assurance that during execution the system timing will agree to the designed timing within specified error bounds. This is simply not possible today using general purpose software development tools and hardware platforms.

There are interesting proposals and avenues of research to correct this situation, but the barrier is quite high for conducting proof of concept studies or to conduct collaborative research and development on this subject.

In this paper, we describe our vision for a testbed and usage model designed specifically to support research, proof of concept, and collaborative work to introduce explicit support for time in microprocessor, reconfigurable fabrics, language and design system architecture for time-sensitive CPS.

The ultimate outcome of the research enabled by the proposed testbed should be the development of modular, interoperable system components including novel microprocessor architectures, software design, communication interfaces, compilers and semantics where timing correctness can be explicitly described and verified. Explicit time specifications can be enunciated, incorporated into application and software/firmware design and executed in such a way that the timing in the executing system matches the time specifications of the designer to within the accuracy of the CPS real-time clock. In other words, the ultimate goal is to enable *correct-by-construction* CPS system timing.

II. Testbed research challenge areas

One of the primary impediment to a designer's ability to have correct-by-construction timing is the availability of explicit support for time in the code stack from microprocessor to hardware, e.g., to support design requirements such as "raise on pin 3, a signal x microseconds following the time a signal was raised on pin 2 to within the precision of the local clock where x is a value determined at run time by system state". Timing specific instructions cannot be specified such that it would be reliably executed today except by custom design in ASICs or FPGAs. If explicit time was appropriately supported, then it would be possible to design true, portable real-time operating systems (OS), programming languages would emerge to build on this capability, and time-sensitive application and communication design practices would follow.

One of the key research challenges is enabling bounded timing support in the code stack. Of course there have been prior efforts in this direction [2]. Another example is the Programming Temporally Integrated Distributed Embedded Systems (PTIDES) model developed at the University of California Berkeley [3]. More research challenges and potential solutions have also been discussed in the National Institute of Standards and Technology (NIST) Cyber-Physical Systems Public Working Group [4]. Stemming from the challenge of enabling hardware agnostic timing support in the code stack is the ability to define common semantics and interfaces to enable correct-by-construction on a variety of microprocessors. Another research challenge is the ability to measure and verify that the system can achieve worst-case bounded timing or be able to handle graceful degradation if timing requirements are not achieved.

There are of course other issues such as providing synchronized clocks in each node and ensuring that network communications realize timing specifications. However the state of the art in clock synchronization is quite good using available technologies such as Global Navigation Satellite Systems (GNSS), Network Time Protocol (NTP) [5], IEEE 1588 Precision Time Protocol [6], or Conseil Europen pour la Recherche Nuclaire's (CERN) White Rabbit [7][8]. The situation with networks is less satisfactory but there is a concerted ongoing effort in the IEEE 802 community to enable time-sensitive networking (TSN) [9].

We envision a distributed testbed bringing together a community of multi-disciplinary experts to enable exploration of time aware interfaces, methodologies, and measurement capabilities to the simple CPS architecture diagrammed in Figure 1. Illustrated are four CPS nodes communicating via a fabric and with each node interacting with the external physical world to be measured and/or controlled.

As illustrated, each node consists of a system stack with custom hardware at the bottom, interacting with the external physical world and the communication fabric. This layer is typically a combination of an FPGA, digital to analog converters (DACs), analog to digital converters (ADCs), communication physical layer (PHYs), input/output (I/O) etc., with standard computer interfaces to the operating system of the microprocessor. The designer creates code running on the microprocessor and a closely connected or integrated reconfigurable fabric, which in conjunction with the underlying hardware, realizes the functional and timing specifications of the CPS.

The following sections describe a conceptual design and use model for a testbed designed specifically to support research, proof of concept, and collaborative work to introduce explicit support for time in microprocessor, reconfigurable fabrics, language and design system architecture for time-sensitive CPS. Because the testbed includes a physical monitoring and control scenario, it will be possible to quantitatively compare different approaches by measuring the timing performance of

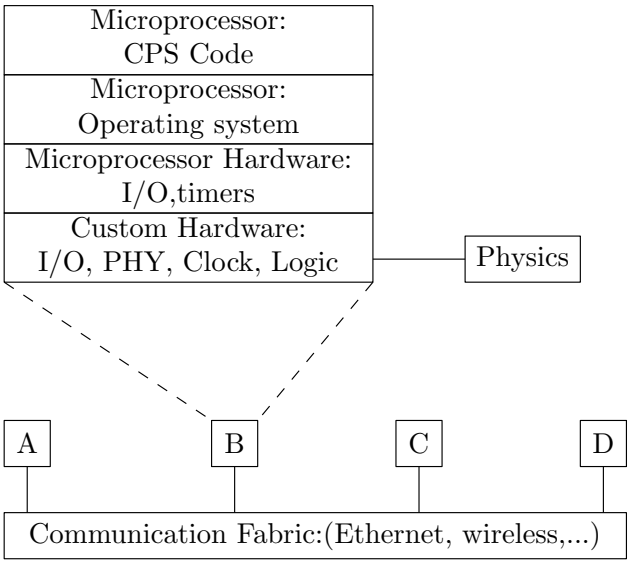

Fig. 1. Simple CPS

two or more proposed designs. One of the objectives of the testbed architecture is to support the exploration of application designs and practices in explicit timing support in hardware, operating system and code design stack of underlying real-time applications. For distributed systems, the proposed testbed could be expanded to enable research on time-sensitive network components and practices.

III. The testbed architecture

The proposed testbed architecture is illustrated in Figure 2. The key elements are as follows:

- Four CPS nodes described in section III-A. These are connected to a communication fabric and interface to testbed physics.
- Communication fabric: Standard gigabit Ethernet implemented with an IEEE 1588 bridge to enable precise clock synchronization among the distributed nodes. Synchronization is needed to ensure timing requirements can be estimated and measured in a distributed system.
- Physics: A selection of devices to be used in simple CPS applications. See section III-B.
- Physics Monitoring and Control: Instrumentation to monitor, configure and control the experiment physics to provide ground truth timing measurements for comparison with the specifications of the CPS system being tested. See section III-C.
- Testbed Site Computer: The computer has three main functions:
  - Communication Monitor and Device Configuration: This interface monitors traffic on the testbed communication fabric, e.g., with Wireshark. It also interfaces with the Physics Monitoring and Control instrumentation. It provides the interface to the four CPS nodes for downloading node FPGA and software.
  - Local Testbed Management: Manages the operation of the testbed. Included is user session management, loading of default CPS node designs, etc. A repository of testbed site approved introductory code samples similar to "hello world" would be maintained here. The repository of code contributed will enable the community to share code and evolve the algorithms and software implementations to ensure hardware portability and system scalability as each user may apply improve upon the code to fit their use case and system. The repository will also aid in growing the ontology for enabling explicit timing support by exploring frequently used semantics needed to describe the timing requirements. Through exposure to a wide variety of CPS use cases as well as experimental algorithms and methodologies, the distributed testbed will enable the community to ensure semantics for explicit timing support are adequately captured and provide the flexibility needed to meet a wide range of CPS timing requirements. Flexibility in application requirements and hardware relies on a high quality, stable and complete ontology to describe the features that are pertinent to reliable system timing. See section III-D.
  - Interface to Remote Users: Provides an interface to remote user sites to permit download of CPS node design, monitoring of the network, CPS node performance, and physics monitoring and control. It provides security and login functions.
- Internet: The public Internet is used for communications between remote user sites and testbed sites.
- User Site Computer: The user site computer has the following functions:
  - Interface to Remote Testbed: Provides an interface to remote user sites to permit download of CPS node design, network monitoring, CPS node performance, and physics monitoring and control. It also provides security and login functions.
  - User Interface, Design and Configuration Tools: The interface enables exploration of explicit timing support methodologies, such as PTIDES, to allow a remote user to generate FPGA design, to program and compile software for the microprocessor, to download FPGA designs and code executables, and to monitor and configure the CPS. FPGA design interfaces and code compilers could execute on the client side or done remotely on tools executing at the remote testbed site. The tools and the testbed would enable experimenting with different:
    - designs for hardware support of explicit time,
    - designs for true real-time operating systems,
    - languages, compilers, and other software development infrastructure

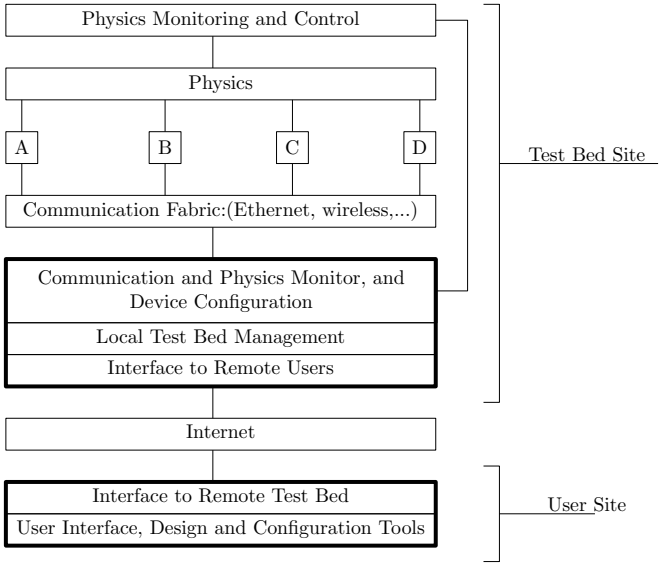

Fig. 2. Testbed Architecture

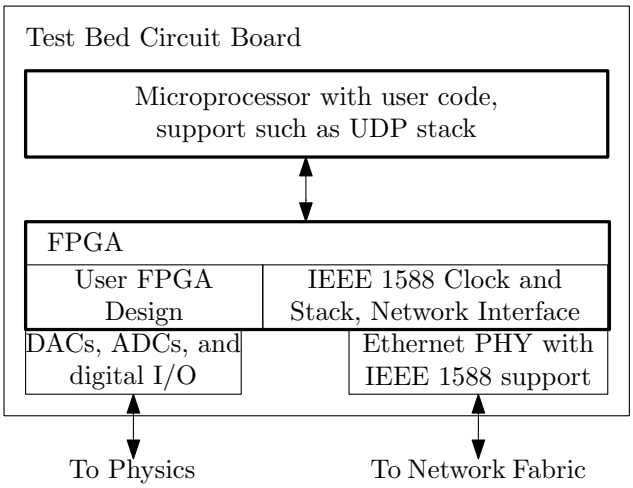

Fig. 3. Testbed Node Architecture

- techniques for exploiting explicit time in applications both within a single node and in a distributed CPS system.

A. Testbed CPS Node Architecture

The architecture of each of the four CPS testbed nodes is illustrated in Figure 3 and consists of a circuit board with the following components:

- An FPGA: A fairly large FPGA to give users plenty of room to try out designs. Several options will be supported: the FPGA can have resident microprocessors either for user code or to implement all or part of the network or clock synchronization stacks, or the FPGA will serve as an interface to a separate chip level implementation of the microprocessor design. In any case, the FPGA will contain IEEE 1588 hardware clock to provide synchronized clock service among the peers in the CPS. It also contains any needed network interface. The clock will interface to user FPGA designs and/or to the microprocessor. It should be noted that it is our intent to support new and evolving standards such as the IEEE TSN [9].
- A microprocessor: The specific microprocessor, associated memory and support are to be determined as well as whether a different microprocessor should be on each board to allow investigation/proof that a explicit timing support methodology, such as correct-by-construction, can be invariant to microprocessor speed, cache size, etc. The testbed intends to include a wide range of hardware designs for supporting explicit time. Some of the hardware currently shown in the FPGA block may reside in the microprocessor.
- Ethernet PHY with IEEE 1588 support: These PHYs are readily available and simplify the implementation of the IEEE 1588 stack. The PHY will also provide data connectivity to the gigabit network fabric.
- Physics interface: A selection of DACs, ADCs and digital I/O will be provided. The specifics depend on the details of the experimental physics section of the testbed.

B. Testbed CPS Physics

The testbed physics enables testing of time sensitive designs applied to realistic CPS applications and the comparison of alternative designs. The selection of components should be simple at first but at a minimum should provide devices suitable for analog, digital and frequency dependent applications. Examples might include one or more of the following:

- Two small laboratory bench size motor-generator sets, mock transmission lines, capability to measure waveforms or phase, and capability to connect/disconnect from a load. This could be used to mimic power system applications such as synchronizing two generators prior to connecting to a load.
- A digital pattern generator and capture device to allow testing of stimulus response applications with sporadic or patterned signals.
- Two or more vibration sensors, perhaps mounted on the motor-generators to allow testing of machine condition monitoring style applications where frequency control of ADCs is important.
- The physics could be implemented in a Hardware-in-the-Loop (HIL) simulation manner, in which a powerful computer is running a model of the physics in real-time.

C. Testbed CPS Physics Monitoring

The testbed monitoring capabilities will depend on the specifics of the testbed physics. The monitoring will generate a variety of time series data which could be fused together from local or remote locations. Ideally the physics and monitoring should be designed together and in many cases can be

implemented using standard small scale laboratory equipment available from several manufacturers. In some cases this equipment can be synchronized to the IEEE 1588 timescale to simplify comparison of ground truth monitoring with the results obtained from the CPS nodes. In the case where the physics are done via HIL simulation, physics monitoring would be integrated into the model and connected directly to the rest of the system.

D. Testbed Hello World Examples

As with any set of tools, users will experience a learning curve. A proven way to shorten the learning curve is to provide hello world examples that illustrate how to formulate the FPGA and code designs, load them into the testbed and observe the results. The current testbed design is comprised of:

- A simple time-triggered application: The measure, compute, and actuate cycle could be implemented in the FPGA in combination with microprocessor code. This would be suitable for implementing the power generation test example mentioned in section III-B. This technique is discussed in [10].
- A PTIDES-based application: The PTIDES model [3] could be implemented for a simple control or measurement system. The power systems example would be appropriate as would monitoring and response to sporadic signals. Such a test application is described in [11].
- A National Instrument (NI) Reconfigurable I/O (RIO) system, where the application under test could be developed in LabVIEW using both Real-Time and FPGA modules, implemented on a CompactRIO controller, and the physics could either be a physical experiment and NI-PXI equipment would be used as measurement and control, or the Physics could be an HIL simulation implemented in NI PXI equipment using real-time and FPGA RIO hardware and software, and the measurement and control sub-system would be integrated with the physics. [12]

IV. The testbed usage model

Initially we envision separate testbeds at the location of the core group of participants. This will make it easier to converge on a suitable and robust design for the testbed. As the number of users grows we will want to replicate all or part of the testbed in their own facilities. The testbed hardware and software tools would be open source.

- The designs, code and parts lists will be made available to the technical community.
- Once the testbed model is deployed, the idea would be to expand towards a modular, distributed testbed to enable remote access to one or more locations that would be available to researchers and to encourage collaborative efforts. *One of our goals in presenting this paper at this conference is to solicit open feedback and start building a community around this effort.*

Ideally researchers in remote locations could reserve time on a testbed. Remote access to the testbed would provide monitoring of usage, agreement on testbed policy, etc. They would use the provided design tools to implement their designs which would then be loaded onto the testbed. Execution would be monitored by the users to verify performance and correctness of the implementation.

After some experience with the testbed, it might be appropriate to provide additional open source examples. *The repository is critical to publicizing successful developments arising out of the use of the testbed.* What is needed is the ability of interested researchers and potential industrial users to leverage the proposed solutions and to experiment with the code without having to recreate the entire system in their own facility.

Provision would be made for multidisciplinary researchers to collaborate enabling systems, compilers and networking researchers, among others, can evolve their technologies realize correct-by-construction.

Finally it is critical to develop clear, well documented examples and tutorials to encourage and educate system designers and developers. Depending on the selection of the initial example this work should be done by people with experience with the example techniques and applications.

V. Benefits of the testbed

Aside from the principal benefit of enabling individual and collaborative research on explicit time support in the node stack, other possible uses and benefits include:

- Allowing semiconductor manufacturers to explore tradeoffs on the distribution and form of hardware timing support in microprocessor, the development of time-explicit tool chains and the like. This capability should shorten their learning curve, allow beta demonstrations to gather customer feedback and make it much easier for manufactures to confidently incorporate the advances arising out of the use of the testbed.
- Plugfest activity: The testbed should be relatively portable which should enable its use for conducting tests and demonstrating the new technologies. The presence of the testbed should encourage expansion of the ideas explored.
- University teaching projects: Many universities include projects for classes in embedded system design, control and related subjects. A distributed testbed enables students to develop code on a variety of platforms for different CPS scenarios, without having to acquire potentially costly equipment.
- Developing a cadre of researchers, students, and others who are familiar with the developed techniques would be of great benefit to societal innovation. One of the barriers to changing system design and programming methodologies is the lack of an experience base and thorough validation. The emerging cadre of experienced students by learning through community and online examples would facilitate adoption and encourage the idea to proliferate in a variety of domains and platforms.

The testbed concept would be an ideal way for industry to rapidly gain experience and innovate safety-critical systems more rapidly.

- As a way of promoting the principles and technologies for time-explicit design, the testbed would be an ideal demonstration platform easily used by application engineers. Introductory, open-source examples would be a boon to proliferating the understanding, adoption and expansion of the ideas.
- The proposed testbed would also benefit standards development and other industry efforts to enable explicit timing support in dealing with CPS.
  The testbed can be particularly useful in allowing research groups and industry to rapidly evaluate academic work in the area of CPS timing and to promote collaborative development of standard interfaces among consortia, government, and academic researchers. As a result, an important aspect of the testbed is to make the interfaces open, generic and interoperable, so that some other equipment or processor can be used to build the system, and evaluated. Such organizations include The AVNU Alliance, http://avnu.org/, the Industrial Internet Consortium (IIC), http://www.industrialinternetconsortium.org/, and the IEEE 802.1 TSN http://www.ieee802.org/1/pages/tsn.html working group.

VI. Role of reconfigurable computing (RC) and FPGA technology in the testbed

Reconfigurable computing and FPGA technology play a key role in this project. It will be used at least in the following subcomponents:

- *System interface for the testbed nodes.* RC/FPGAs will be used to implement the deterministic networking interface and the logic to interface to microprocessors and I/O.
- *Device under Test.* Since the researchers using this facility are mainly trying to test out new technologies, RC and FPGAs provide a great vehicle to test their designs. A shared or dedicated FPGA can used also be part of the testbed node.
- *Deterministic networking.* As part of the system interface, this component will rely on existing (IEEE 1588) or new (IEEE TSN) standards to provide a global notion of time and deterministic data transfer, respectively. Since these are going to be evolving standards, having the flexibility of the FPGA is very important.
- *Physics modeling.* The testbed user has an option of interfacing to the real physics associated with their system, or to simulate the physics with HIL infrastructure based on high performance instruction processors or RC/FPGAs.
- *Measurement infrastructure.* In order to accurately correlate results of the interaction of a CPS cyber-controller and the plant/physics, the measurement infrastructure would interface to the deterministic network using RC/FPGAs to enable measurement processing in real-time. In addition, the testbed would also support measurement capabilities for explicit timing support. Through experience, the efforts of the testbed can determine the pertinent metrics needed to ensure distributed, deterministic, and interoperable timing support.

VII. Conclusion

We have proposed a conceptual design and use model for a reconfigurable testbed designed specifically to support research, proof of concept, and collaborative work to introduce explicit support for time and synchronization in microprocessors, reconfigurable fabrics, language and design system architecture for time-sensitive CPS. Reconfigurable computing is used throughout the system in several roles: as part of the prototyping platform infrastructure, the measurement and control system, and the application system under test.

Disclaimer: Certain commercial entities, equipment, or materials are identified in this document in order to describe the experimental design or to illustrate concepts. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

References

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC),*. Orlando, Florida: IEEE, 2008, pp. 363 – 369. [Online]. Available: http://dx.doi.org/10.1109/ISORC.2008.25

[2] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert, "From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications," in *ACM Sigplan Notices*, vol. 38, no. 7. ACM, 2003, pp. 153–162.

[3] Y. Zhao, "On the design of concurrent, distributed real-time systems," PhD, University of California, Berkeley, 2009.

[4] NIST, "NIST Cyber-Physical Systems Public Working Group," 2014. [Online]. Available: http://www.nist.gov/cps/

[5] D. J. Mills, E. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," University of Delaware, Tech. Rep. RFC 5905, June 2010. [Online]. Available: http://www.hjp.at/doc/rfc/rfc5905.html

[6] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std. 1588-2008*.

[7] G. Gaderer, P. Loschmidt, E. G. Cota, J. H. Lewis, J. Serrano, M. Cattin, P. Alvarez, P. M. Oliveira Fernandes Moreira, T. Wlostowski, J. Dedic, C. Prados, M. Kreider, R.Baer, S.Rauch, and T.Fleck, "The white rabbit project," in *Int. Conf. on Accelerator and Large Experimental Physics Control Systems*, Kobe, Japan, 2009.

[8] M. Lipinski, T. Wlostowski, J. Serrano, P. Alvarez, J. Gonzalez Cobas, A. Rubini, and P. Moreira, "Performance results of the first white rabbit installation for cngs time transfer," in *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*. IEEE, 2012.

[9] Institute of Electrical and Electronics Engineers, "Time-Sensitive Networking Task Group," 2014. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[10] H. Kopetz, *Real-Time Systems : Design Principles for Distributed Embedded Applications*. Springer, 1997.

[11] P. Derler, J. C. Eidson, S. Goose, E. A. Lee, S. Matic, and M. Zimmer, "Using PTIDES and synchronized clocks to design distributed systems with deterministic system-wide timing," in *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2013.

[12] National Instruments, "LabVIEW RIO Architecture," 2015, http://www.ni.com/white-paper/10894/en/ [Online; accessed 4-August]. [Online]. Available: http://www.ni.com/white-paper/10894/en/