ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA027

EVALUATION OF TIMING AND SYNCHRONIZATION TECHNIQUES ON NI CompactRIO PLATFORMS

O. O. Andreassen, C. Charrondiere, K. Develle, R. Rossel, T. Zilliox CERN, Geneva, Switzerland

Abstract

Proper clock synchronization between systems is key to successfully integrate data acquisition and control systems in the CERN accelerator domain. This applies to everything from simple diagnostics- to mission critical systems. The internal clock of a National Instruments based CompactRIO (NI-cRIO) system has an accuracy of 40 ppm at 25 □. In addition, the NI-cRIO onboard FPGA has its own 40 MHz clock with an accuracy of 100 ppm. By default, the NI-cRIO FPGA clock is not synchronized with the controller. For short measurements, this drift is usually negligible, but for continuous data acquisition systems running 24/7, the accumulated error has to be compensated. In this article we show how to correct time drift using protocols such as Network Time Protocol (NTP), Precision Time Protocol (PTP) and White Rabbit (WR) in combination with NI's FPGA TimeKeeper library.

BACKGROUND

Precise timing is essential to successfully operate a large-scale control system. At CERN, the accelerators are synchronized to nanosecond accuracy with the in-house developed, General Machine Timing (GMT) system. GMT electronic boards has been developed for several bus systems such as VME and cPCI, however at its deployment (2004), the cRIO platform was not mature enough and therefore not considered at the time. However, during the last 15 years, the cRIO platform has grown considerably. Its FPGA based backend, multitude of measurement modules and rapid development cycle has made it a mature platform which has become increasingly popular for test and monitoring systems at CERN. This has led us to start evaluating what is needed to fully integrate the platform in the CERN accelerator domain [1].

The successor of the GMT system, White Rabbit (WR), features improved timing accuracy and solves most of the GMT's shortcomings, and is actively being developed at CERN. An IEEE standardisation committee has been set up to incorporate it into a new IEEE-1588 standard. This and the availability of the WR design on the Open Hardware (OH) portal has made it possible to integrate the CERN timing on the cRIO platform at the hardware level [2].

CHALLENGE

A cRIO system consists of a Real Time (RT) controller with a standard Intel or ARM based processor and a user-programmable FPGA that is populated with one or more conditioned I/O modules. These modules provide direct sensor connectivity and specialty functions [3].

In cRIO systems, there are up to three timed components: the FPGA, the real-time processor, and hardware timed IO modules such as the GPS or WR module. Each component has a different clock that needs to be synchronized [3].

A typical RT controller such as the NI cRIO-9035 has a real-time drift of ±40 ppm at 25 °C and the FPGA has a drift of ±100 ppm, which will cause time drifts up to several seconds per day if not synchronized [4, 5].

Choosing synchronization technologies depends on synchronization accuracy requirements, distance between nodes, platform support, technology availability, and more.

In general, there are two types of synchronization: time or signal based

- Time-based: All components have a common time reference. Events, triggers, and clocks can be generated based on this time.
- Signal-based: Clocks and triggers are physically connected between systems.

In this paper we will focus on Time based synchronization and how White Rabbit can be used to accurately align clocks on both cRIO controllers and FPGA's.

WHITE RABBIT

White Rabbit is the name of a collaborative project aimed at developing a fully deterministic Ethernet-based network for general purpose data transfer and sub-nanosecond accuracy time synchronization. The hardware designs as well as the source code are publicly available.

White Rabbit provides sub-nanosecond synchronization accuracy, which formerly required dedicated hard-wired timing systems, with the flexibility and modularity of real-time Ethernet networks. A White Rabbit network may be used solely to provide timing and synchronization to a distributed electronic system, or be used to provide both timing and real-time data transfer [2].

RT CONTROLLER TIME

The cRIO RT controller clock can be synchronized similar to regular computers, using either a software or hardware-based timing system such as Simple Network Time Protocol (SNTP), Precision Time Protocol (PTP), Global Positioning System (GPS) and White Rabbit (WR).

(S)NTP

The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks (Figure 1). In operation since before 1985, NTP is one of the oldest Internet protocols in current use and has an accuracy of ~1 ms [6].

The Simple Network Time Protocol (SNTP) is a less complex implementation of NTP, using the same protocol

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA027

but without requiring the storage of state over extended periods of time. It is used in some embedded systems and in applications where full NTP capability is not required [4].

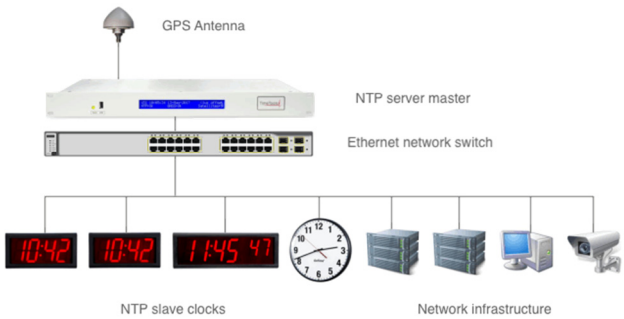

Figure 1: SNTP synchronization.

The cRIO controller can be configured to synchronize with both SNTP and NTP through its standard driver set, depending on the controller type. At CERN where the NTP master clocks are in fixed locations and the network routing can be fixed, we have achieved a stable timing accuracy better than 250 µs [6].

IEEE 1588

IEEE 1588, also known as the Precision Time Protocol (PTP), is an Ethernet-based synchronization method designed for cabled, local networks. IEEE 1588 allows you to synchronize targets over a standard Ethernet based network. It works by determining a master clock automatically from the network and having all other clocks slave to this master clock. IEEE 1588 is also a relatively fault tolerant synchronization system. If the master clock is disconnected from the network, a new master is dynamically determined. All cRIO controllers support a software implementation of IEEE 1588. The typical accuracy on a distributed network with TSN compatible switches is about 1 µs [7].

FPGA TIME

Any hardware-based timing on the cRIO platform has to pass through the FPGA. Whether it is a hardware trigger, a GPS signal or CERN's White Rabbit timing, it all ends up on the FPGA as a pulse or timestamp. The cRIO FPGA does not have a built in Phase Lock Loop (PLL) however, there are many examples and existing algorithms that can be used to implement one.

GPS

GPS determines locations on earth by receiving precisely time-stamped signals from multiple satellites, determining distance from each satellite, and then triangulating to determine position. Because the GPS satellites provide precise timing information, this infrastructure can be used to synchronize systems. GPS is supported on cRIO using a dedicated module which has a synchronization accuracy of 100 ns [8].

White Rabbit on the NI-cRIO

The White Rabbit Timing receiver is an open hardware design, available on CERN's open hardware repository. Engineers at the University of Zürich have used the WR design to build a C-series module for the cRIO platform, using the Module Development Kit of National Instruments (Figure 2). They adapted the FPGA code and wrote a driver in LabVIEW FPGA for the control and readout of the WR time stamp [2, 9].

Figure 2: cRIO-WR module.

As with the GPS, the cRIO-WR module does not automatically synchronize the FPGA clock, but it returns a WR timestamp.

FPGA Time Keeper

Since the FPGA interface on the cRIO platform doesn't have a built-in way to correct or synchronize the FPGA clock, the developer has to calculate any clock drift manually. For this purpose, National Instruments has developed a library called FPGA Timekeeper (Figure 3) which takes a timestamp from, for instance, a GPS or WR module as an input and calculates the real time drift. The library then returns a corrected time for the FPGA which can be used to timestamp data or align triggers and events [10].

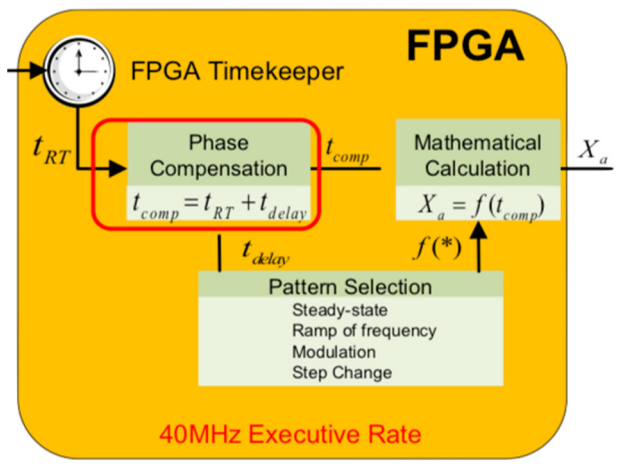

Figure 3: FPGA Timekeeper principle.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA027

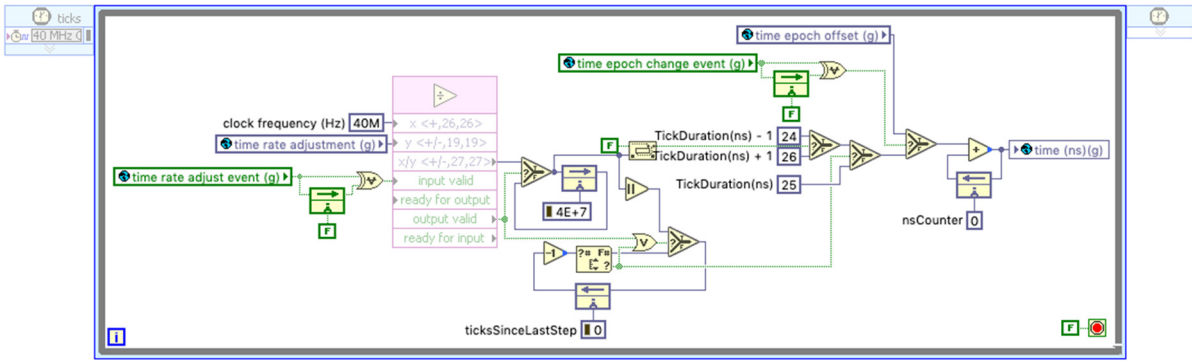

Figure 4: FPGA Timekeeper clock implementation.

The FPGA Timekeeper clock compares the clock frequency (40 MHz) to an internal increment and adjusts the counter giving you the correct time (Figure 4). Basically, the timekeeper library measures the time offset and slowly increments a counter by 1 ns until it reaches the setpoint time [11, 12].

EXAMPLE: SEISMIC MONITORING

In 2017, Three seismic measurement stations were installed to monitor the stability of LHC underground experiments. This was done to detect possible disturbances from civil engineering work affecting the accelerators. The stations send their data to a local data base at CERN and in parallel to the Swiss Seismological Service (SED) at the ETH Zürich. SED requires that the raw data arrive to their data server within 10 s, sampled at 250 Hz with a 1 ms accuracy on every data point, if not the data will be rejected. This means that both the FPGA and the RT controller has to operate with a timing accuracy of less than 1 ms [13, 14].

Controller Synchronization

To synchronize the cRIO controller, we are using NTP. The Linux NTP daemon is a learning process. The daemon will let the local clock drift (faster or slower than the NTP server time) then will apply a first correction to overcompensate the drift (Figure 5).

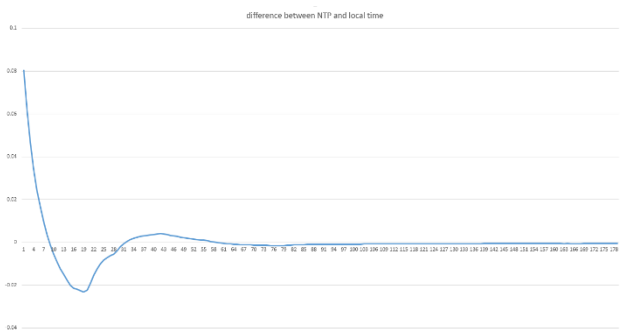

Figure 5: Difference between NTP and local time.

The vertical axis represents the difference between the local clock and NTP time. The horizontal axis represents the number of minutes after the start-up of the cRIO. As seen from Figure 5, it took 1 hour to have less than a 1 ms difference between the local clock and the NTP time. This process is slow because the NTP protocol is calculating the drift of the local clock and the network topology to be able to synchronize less often when it is correctly synchronized. The NTP request to the server is done every 2 s at start-up and every 1024 s when the system is stable [13].

FPGA Synchronization

To synchronize the FPGA, the strategy is a bit different. Two Single Cycle Time Loop (SCTL) [14] are used, were one of the loops is used to count the number of FPGA cycles, and raises a flag every iteration (40 MHz). The other loop (120 MHz) waits for the flag and then reads the time from the White Rabbit C-module. The WR module on the cRIO has a timing precision of ± 3 ns with respect to the WR atomic clock source, which is better than the FPGA drift of 100 ppm. This time is then used as a reference in the FPGA timekeeper library which calculates the FPGA clock drift and returns the corrected timestamp [11, 12].

In our system, when configuring a cycle time of 500 µs the real cycle time of the acquisition loop is 500.0025 µs, giving us an error of 2.5 ns. This error can vary both from system to system depending on the FPGA clock, and is affected by external influences such as temperature and humidity, making it necessary to synchronize with an external clock source. After 2000 samples this has accumulated to a 5 µs error (Figure 6). To compensate for this, we have to add the 2.5 ns error to each data point (Figure 7), avoiding any gaps between consecutive data buffers.

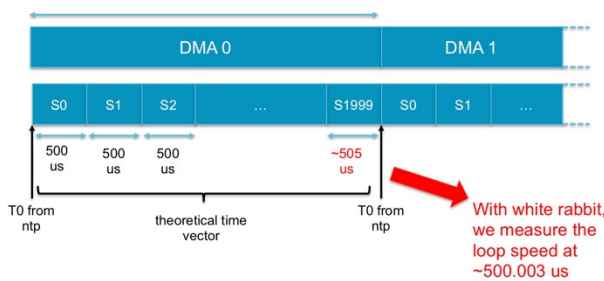

Figure 6: Accumulated timing error.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA027

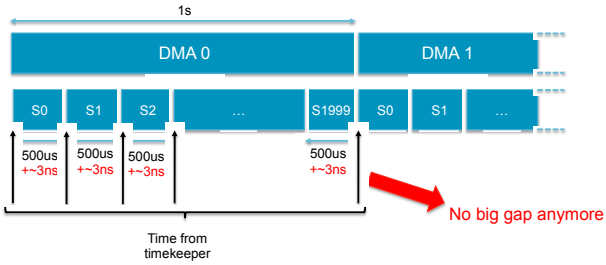

Figure 7: Apply error on each data sample.

CONCLUSION

When designing a measurement or control system, each individual clock in the system has to be considered. As shown, a controller clock with a 40 ppm accuracy can lead to several seconds of drift per day for continuously running applications.

By using tools such as the FPGA time keeper, the cRIO base WR module and NTP we have been able to synchronize and timestamp data on NI-cRIO based systems. This has then allowed successful integration in the CERN accelerator domain and the Swiss Seismological Service.

REFERENCES

[1] J. Serrano, P. Álvarez, D. Domínguez, and J. Lewis, "Nanosecond Level UTC Timing Generation and Stamping in CERN's LHC", in *Proc. 9th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'03)*, Gyeongju, Korea, Oct. 2003, paper MP533, pp. 119-121.

[2] http://www.ohwr.org/projects/wr-node-core/wiki

[3] Choosing a CompactRIO Synchronization Technology, https://www.ni.com/en-us/innovations/white-papers/16/choosing-a-compactrio-synchronization-technology.html

[4] NI cRIO-9035: Embedded CompactRIO Controller with Real-Time Processor andReconfigurable FPGA, http://www.ni.com/pdf/manuals/376935d_02.pdf

[5] CompactRIO Systems, http://www.ni.com/en-us/shop/compactrio.html

[6] The Difference Between NTP and SNTP, https://timetoolsltd.com/network-time-servers/the-difference-between-ntp-and-sntp/

[7] Precision Time Protocol, https://en.wikipedia.org/wiki/Precision_Time_Protocol

[8] Global Positioning System, https://no.wikipedia.org/wiki/Global_Positioning_System

[9] CompactRIO White Rabbit (CRIO-WR), https://www.ohwr.org/project/crio-wr/wikis/home

[10] FPGA Timekeeper and the NI 9467 GPS Module, https://knowledge.ni.com/KnowledgeArticleDtails?id=kA00Z000000P70iSAC&l=en-US

[11] Virtual-Instrumentation-Based PMU Calibratorfor IEEE C37.118.1-2011 Compliance Testing, http://cigre-usnc.org/wp-content/uploads/2015/06/Virtual-Instrumentation-Based-PMU-Calibrator-for-IEEE-C37.118.1-2011-Compliance-Testing.pdf

[12] NI LabVIEW High-Performance FPGA Developer's Guide, http://download.ni.com/pub/gdc/tut/labview_high-perf_fpga_v1.1.pdf

[13] C. Charrondière, M. Cabon, K. Develle, and M. Guinchard, "Ground vibration monitoring at CERN as part of the international seismic network", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 1695-1698. doi:10.18429/JACoW-ICALEPCS2017-THPHA134

[14] Swiss Seismological Service, http://www.seismo.ethz.ch/en/earthquakes/monitoring