TRIGGER AND RF DISTRIBUTION USING WHITE RABBIT

T. Włostowski, J. Serrano, G. Daniluk, M. Lipiński, CERN, Geneva, Switzerland
F. Vaga, University of Pavia

Abstract

White Rabbit is an extension of Ethernet which allows remote synchronization of nodes with jitters of around 10ps. The technology can be used for a variety of purposes. This paper presents a fixed-latency trigger distribution system for the study of instabilities in the LHC. Fixed latency is achieved by precisely time-stamping incoming triggers, notifying other nodes via an Ethernet broadcast containing these time stamps and having these nodes produce pulses at well-defined time offsets. The same system is used to distribute the 89us LHC revolution tick. This paper also describes current efforts for distributing multiple RF signals over a WR network, using a Distributed DDS (Direct Digital Synthesis) paradigm.

SYNCHRONIZATION IN WHITE RABBIT

Both described systems require a precise time and frequency reference, provided by a WR network and locked to a GPS receiver or an atomic clock. WR achieves its sub-nanosecond accuracy and low jitter by employing multiple techniques:

- Distribution of frequency reference encoded in the physical data stream (Synchronous Ethernet).
- Coarse clock offset measurement by timestamping Ethernet frames using the Precision Time Protocol (PTP, IEEE1588).
- Fine offset compensation by tracking the phase shift between the outgoing and incoming clock/data stream.

Further details on the synchronization algorithms used in WR can be found in [1].

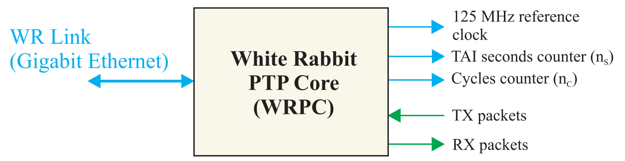


Figure 1: The White Rabbit PTP Core (WRPC).

The RF and Trigger distribution systems are interfaced with the WR network by the WR PTP Core (WRPC) [2], a standardized HDL block implementing the full WR synchronization stack. The WRPC is integrated in the FPGA firmware and delivers a 125 MHz reference frequency and time-of-day (counters of seconds n_S and 125 MHz clock cycles n_C). Furthermore, the WRPC can be used as general-purpose Ethernet Media Access Controller (MAC), sending and receiving Ethernet frames coming from the user's FPGA design (see Figure 1).

RF DISTRIBUTION

Introduction

The RF distribution system is part of a larger project which aims to merge the two historically separate timing systems:

- General purpose timing, usually referenced to the Universal Coordinated Time (UTC) timescale, which usually drives beam injection/extraction, synchronizes currents in the magnets and provides UTC-traceable timestamps for any sort of events that may occur in the machine.
- Beam-synchronous timing, using the bunch or RF frequency as the reference. Its natural applications are driving the RF cavities, synchronizing data acquisition from beam instrumentation or providing precise collision timestamps for event reconstruction (e.g. in the LHC Experiments).

The classic way of distributing a frequency reference in a timing system is to encode it in the data stream (using Manchester or 8B10B encoding) at the master node and recover it in the slave nodes using a Clock-Data Recovery (CDR) PLL. Examples of such timing systems can be found in [3] and [4]. This approach, however, can provide only one reference frequency per each physical link, necessitating the use of two separate timing networks. In many cases the machine's equipment needs simultaneous access to both timing systems, resulting in duplication of cabling and electronics.
Furthermore, many traditional systems experience troubles tracking large frequency changes (e.g. ramping of the RF in low energy or ion machines), due to bandwidth limitations of the CDR PLLs in the deserializer chips.
The method we developed does not rely on physical frequency encoding, but uses the common notion of time and frequency provided by WR to drive RF synthesis in each node of the system, as depicted in Figure 2. The master node keeps is local DDS phase-locked to the RF reference input and broadcasts the DDS tuning values calculated by its PLL over the WR network. The slaves simply feed the received data to their local DDS synthesizers. Since the reference clocks are identical, the DDS in the slave node produces an exact copy of the RF clock coming to the master node.

RF Encoding and Broadcasting

The RF encoding done by the master node is illustrated in Figure 3. The central element of the system is the DDS synthesizer, which produces a sinusoidal signal of arbitrarily controlled frequency and phase. The synthesizer we used is a custom-designed FPGA core attached to a high speed DAC (Digital to Analog Converter). It employs the classical frequency synthesis approach, using a phase accumulator

ISBN 978-3-95450-148-9

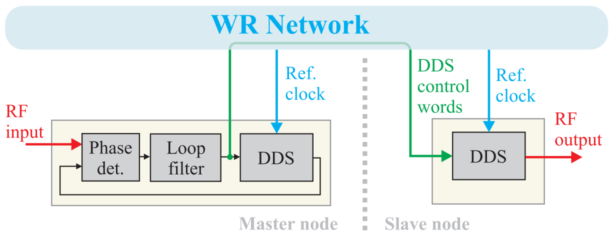


Figure 2: Idea of the RF distribution over White Rabbit.

and a sine lookup table, as described in [5]. We decided to implement a custom synthesizer since in-phase RF reconstruction requires taking snapshots of the DDS phase accumulator at fixed moments in WR time, a feature not available in low-cost integrated DDS+DAC chips.

The core runs synchronously to the 125 MHz WR reference clock F_{ref} and at each clock cycle produces four samples (phase shifted by one-fourth of the F_{ref} period), forming a 500 MSPS data stream for the DAC. In order to improve the dynamic range and spectral purity of the output signal, we used linear interpolation of the sine lookup table values and dithering of the sine signal with noise coming from a pseudo-random number generator.

The output of the DAC passes through an anti-aliasing low-pass filter, which limits the bandwidth of the system to approximately 70 MHz, followed by a zero-crossing detector which produces a square-wave clock signal of frequency f_{DDS}.

The DDS is programmed to a user-specified center frequency F_C, which is continuously adjusted to keep a fixed phase with respect to the varying input RF reference F_{in}. The input can be divided by a user-programmable integer value, extending the frequency range above the bandwidth limit of the DDS.

The adjustment is performed using a typical PLL scheme: the clocks are compared by a Phase-Frequency Detector (PFD) and the resulting phase error is fed to a PI (Proportional-Integral) regulator. The PI outputs the frequency adjustment (tune) of the DDS f_{tune}, closing the feedback loop. In our implementation the PLL consists of a discrete analog PFD followed by an Analog-to-Digital converter (ADC) which drives a digital PI controller. The analog PFD has been chosen to minimize the jitter.

As the RF frequency does not change very rapidly, the controller can sample the phase error and produce a new tune value with a relatively slow period T_S (up to several kHz). The PLL works synchronously to the beginning of the current second (e.g. the first tune update is done when $n_C = 0$, the second one when $n_C = 1 \cdot T_S/8$ ns and so on).

The values produced by the feedback loop are encapsulated into Etherbone [6] packets, comprising:

- The current tune value and the timestamp it was taken at. To simplify calculations, we send the current second and the index i_S of the tune sample with respect to the beginning of the current second.
- The value of the DDS accumulator ϕ_M at the beginning of the current second (when $n_C = 0$), allowing the slave node to compensate for the phase drift of the RF clock caused by the latency of the Ethernet link.
- Configuration parameters: T_S, f_C and the identifier of the master node to auto-configure the slave and allow for multiplexing different RF signals within the same network.

These messages provide all the information necessary for the slave nodes to reproduce the RF signal.

Reception and Reconstruction

RF reconstruction is done at the slave side using the following algorithm:

1. Set a fixed reconstruction latency N_{REC}, which defines by how many samples of f_{tune} the slave's RF output is delayed with respect to the master RF signal. The latency must be obviously larger than the signal processing and packet transmission/reception delays. In the test system, we used $N_{REC} = 3$ tune samples, corresponding to a 300 μs delay.

2. Filter the incoming packet stream to select only the messages containing information about the RF signal the slave is subscribed to.

3. Calculate the initial DDS accumulator value to align the phase of the slave's RF output with the master. Since phase is the integral of frequency, the slave's approximate output phase after N_{REC} tune samples can be derived from the last N_{REC} tune values (assuming that the distributed frequency remains stable over the period of N_{REC} samples):

$$\phi_S = \phi_M + \sum_{i=1}^{N_{REC}} \frac{T_S}{f_{tune}[i]} \qquad (1)$$

4. Apply the phase correction by overwriting the DDS accumulator with the value of ϕ_s.

5. Continuously feed the slave's DDS with the received tune values after delaying them by N_{REC} samples. For example, the first sample produced by the master (having $i_S = 0$) is written to the slave's DDS when $n_C = \frac{N_{REC} \cdot T_S}{8ns}$.

As the result, the slave's DDS produces an in-phase copy of the master RF clock. Note that the current algorithm compensates the phase shift only once, so the phase will drift over time due to changes of the RF frequency. A continuous phase compensation mechanism is currently being developed.

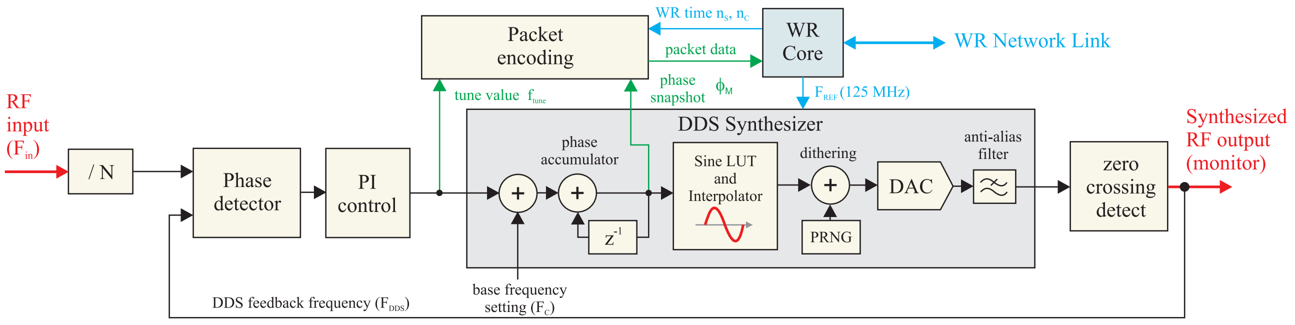


Figure 3: Architecture of the DDS master node.

TRIGGER DISTRIBUTION

The WR Trigger Distribution (WRTD) system was designed to provide transport of trigger pulses between the devices involved in the LHC beam instability diagnostics. The WRTD receives a trigger from a “cloud” of devices (e.g. the Base Band Tune Monitor or the Transverse Damper) upon the onset of an instability and distributes it to all relevant devices (e.g. the Beam Pickup oscilloscopes) to freeze their acquisition buffers, with low and fixed latency. The assignment of trigger outputs to trigger inputs as well as the latency is configurable by the user.

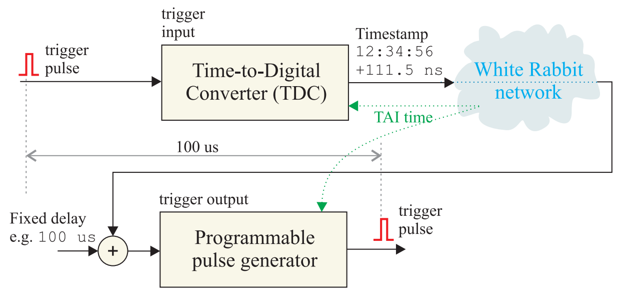


Figure 4: Trigger distribution over White Rabbit.

The concept of WRTD is shown in Figure 4:

- A pulse comes from a source device and is time tagged using a Time-to- Digital Converter (TDC).
- The timestamp produced by the TDC is broadcast over the WR network, with a user- assigned identifier, allowing to uniquely identify the device the trigger came from.
- A node interested in that trigger takes its origin timestamp, adds a certain (fixed) latency and produces a pulse at the calculated moment.

Since all the nodes in the network are synchronized by WR, the measured delay between input and output pulses is equal to the value added to the origin timestamp.

Functionality

The current set of features comprises:

- Nodes in VME64x format, each with 5 inputs and 4 outputs.
- An output can be simultaneously assigned to 128 trigger inputs, each with independently programmable delay.
- Single-shot and continuous triggering.
- Programmable input and output dead time.
- All parameters of triggering can be changed during run-time.
- Extensive diagnostic features: detailed status of each channel, injection of software trigger messages and logging of all trigger events (including triggers missed due to excessive latency).

IMPLEMENTATION

Hardware

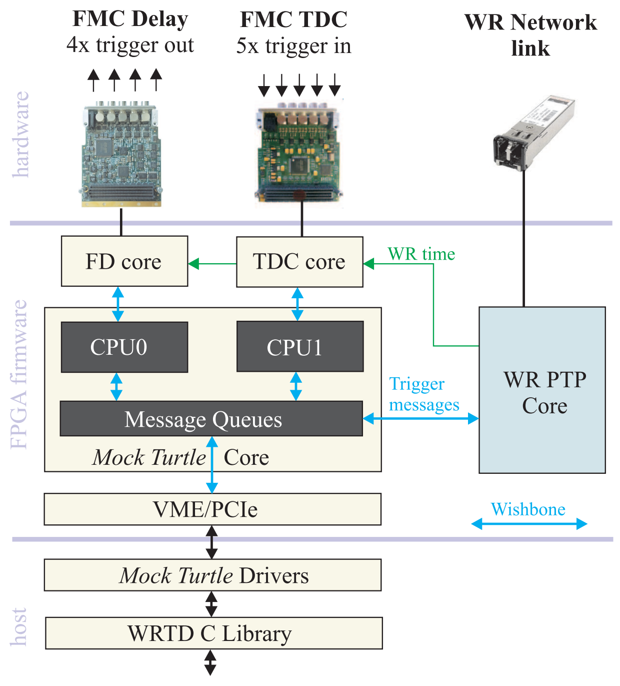
Both presented systems are based on the CERN BE-CO-HT’s Standard Hardware Kit [7] - a collection of carriers and FPGA Mezzanine (FMC) boards. We chose the Simple VME64x FMC Carrier board (SVEC, [8]), hosting the FPGA that runs the DDS/WRTD firmware and the three mezzanines, interfacing the FPGA with the physical signals:

- DDS FMC [9], with a 500 MSPS DAC, a phase detector and clock distribution and conditioning circuits. This card provides the hardware part of the RF Distribution system.
- TDC FMC [10], a 5-channel TTL input Time-To-Digital converter with 81ps resolution, used by the WRTD trigger inputs.
- Fine Delay FMC [11], used as a programmable 4-channel TTL pulse generator with 10 ps resolution, outputting triggers in the WRTD.

FPGA AND SOFTWARE

The FPGA architecture is based on a mix of dedicated VHDL IP cores and embedded soft CPUs, provided by the *Mock Turtle* framework [12] and interconnected through the Wishbone bus [13].

The trigger distribution FPGA firmware was built using unmodified Fine Delay and TDC Cores [11] [10]. Processing of the network messages, configuration and delay adjustment is handled by the soft CPUs. One processor is responsible for the trigger inputs and the other for the trigger outputs, as shown in Figure 5. No additional VHDL development was necessary except for connecting the blocks together.



Client application (LHC Instabilities, OASIS, etc...)

Figure 5: Architecture of the WRTD Hardware, FPGA and software.

In the case of the RF distribution system, the VHDL core provides only the DDS synthesizer and interfaces to the clock distribution circuit and the phase detector. The complex part: the entire feedback loop, packet encoding and phase compensation is incorporated in the embedded software running on the CPU, as its processing power and determinism are sufficient for the required latencies (300 μs) and deadlines (PLL update every 100 μs).

The carrier-mezzanine abstraction model we employed made the designs independent from the host platform. Initially developed for VME64x, both systems can be easily ported to other platforms (PCI Express or PXIe) by simply changing a carrier-dependent VHDL template module.

The software stack consists of bare-metal (i.e. without an operating system) real time applications that run on the embedded CPUs, and user-space libraries providing the APIs for trigger management (WRTD) and DDS configuration (RF distribution). Both of them are built on top of the *Mock Turtle* software, which provides a real-time development environment, as well as a generic Linux device driver/library to communicate between the host system and the real-time applications.

PERFORMANCE

RF Distribution

Preliminary measurements were performed for f_{in} = 352 MHz, f_{DDS} = 44 MHz (input divided by 8) and an update period T_S = 100 μs (10 kHz sampling rate). The system provided phase alignment of the slave node better than 1 nanosecond and rms jitter (100 Hz - 10 MHz) of 20 picoseconds. The largest contribution to the jitter comes from sub-optimal design of the WR reference clock generation circuit on the DDS mezzanine, which will be improved in the next version of the hardware, with an expected jitter of 3 ps rms.

Trigger Distribution

The number of trigger sources in the system and the trigger repetition rate is limited by the network bandwidth and maximum acceptable latency requirements. The system installed in the LHC is specified to transport 35000 trigger pulses per second, which occupies approximately 5% of the bandwidth of a Gigabit Ethernet link at the worst-case latency of 300 μs (out of which, 200 μs is allocated for the delays introduced by the fibers). This allows for distribution of the 89 μs revolution tick using the WRTD network, for the purpose of realigning asynchronously generated triggers with the machine's bunch crossing frequency.

PROJECT STATUS AND OUTLOOK

The RF distribution is part of a larger effort to design a unified timing system, incorporating the general purpose and beam synchronous timing in a single standardized network. We have already proven the feasibility of transferring an RF clock over an Ethernet link. In order to provide a complete timing system, we are currently developing the event distribution and reception infrastructure.

WRTD is a mature system, working in operational installations in the LHC and being integrated in the CERN Controls software stack. Furthermore, the OASIS project at CERN [14], which provides acquisition, correlation and visualization of analog signals (a "distributed oscilloscope") is developing a new WRTD-based infrastructure. WRTD will gradually replace the current trigger distribution system based on a central trigger multiplexer and discrete cabling.

The RF and trigger distribution are only two examples of applications of WR technology to solve real problems in distributed real-time controls and data acquisition. In our experience, the availability of a set of modular, reusable open source hardware building blocks greatly reduces development time and allows us to better fulfill the needs our users.

REFERENCES

[1] J. Serrano, M. Cattin, E. Gousiou, E. van der Bij, T. Włostowski, G. Daniluk, M. Lipiński, "The White Rabbit Project", IBIC2013, Oxford, UK (2013).

[2] G. Daniluk, “White Rabbit PTP Core: the sub-nanosecond time synchronization over Ethernet”, *M.Sc. thesis*, Warsaw University of Technology (2012), http://www.ohwr.org/documents/174

[3] Micro-Research Finland Oy website: http://www.mrf.fi/index.php/timing-system/70-timing-system-structure

[4] D. Domínguez, J.J. Gras, J. Lewis, J.J. Savioz, J. Serrano, F.J. Ballester, “An FPGA-based Multiprocessing CPU for Beam-Synchronous Timing in CERN’s SPS and LHC”, ICALEPCS2003, Gyeongju, Korea (2003).

[5] Analog Devices, “A Technical Tutorial on Digital Signal Synthesis” (2009), http://www.ieee.li/pdf/essay/dds.pdf

[6] M. Kreider, R. Baer, D. Beck, W. Terpstra, J. Davies, V. Grout, J. Lewis, J. Serrano, T. Włostowski, “Open borders for system-on-a-chip buses: A wire format for connecting large physics controls”, Phys. Rev. ST Accel. Beams, vol. 15 (2012).

[7] E. Van der Bij, M. Cattin, E. Gousiou, J. Serrano, T. Włostowski, “CERN’s FMC Kit”, ICALEPCS2013, San Francisco, USA (2013).

[8] Simple VME64x FMC Carrier project homepage: http://www.ohwr.org/projects/svec/

[9] FMC DDS project homepage: http://www.ohwr.org/projects/fmc-dac-600m-12b-1cha-dds/wiki

[10] FMC TDC project homepage: http://www.ohwr.org/projects/fmc-tdc-1ns-5cha-hw/wiki

[11] FMC Fine Delay project homepage: http://www.ohwr.org/projects/fmc-delay-1ns-8cha/wiki

[12] T. Włostowski, J. Serrano, F. Vaga, “Developing Distributed Hard-Real Time Software Systems Using FPGAs and Soft Cores”, THHA2I01, these proceedings, ICALEPCS’2015, Melbourne, Australia (2015).

[13] Wishbone bus specification, version B.4: cdn.opencores.org/downloads/wbspec_b4.pdf

[14] S. Deghaye, L. Bojtar, C. Charrondiere, Y. Georgievskiy, F. Peters, I. Zharinov, “OASIS Evolution”, ICALEPCS2007, Knoxville, USA (2007).