Generating a timing information (1-PPS) from a software defined radio decoding of GPS signals

D. Rabus, G. Goavec-Merou, G. Cabodevila[1], F. Meyer[2], J.-M Friedt

***Abstract*—We complement a secure, Software Defined Radio (SDR) implementation of Global Navigation Satellite System (GNSS) reception system with the 1-Pulse Per Second (PPS) output to steer the oscillator driving the analog to digital converter, the only reliable timing information in a SDR reception chain. We demonstrate long-term 1-PPS output and the derived Allan deviation consistent with the 10^{-8} at 1 s decreasing to 10^{-12} at 10^4 s performance met with most single frequency GNSS receivers.**

I. Introduction

Software Defined Radio aims at replacing as many hardware parts from a radiofrequency reception chain and replace them with a software implementation handing the output of the analog to digital converter. As such, this signal processing approach blurs the classical abstraction layers between hardware and software protocols. In the context of Global Navigation Satellite System (GNSS) secure data reception [1], spoofing identification and mitigation has been addressed at the plane wave physical characteristics low level approach by analyzing the phase of the signal received by multiple antennas, and possibly steering the null of the controlled radiation pattern antenna array (CRPA) towards the spoofing or jamming source (Fig. 1). The application is demonstrated on embedded systems exhibiting sufficient computational power to run `gnss-sdr` thanks to the efficient use of GNU Radio and its VOLK SIMD instruction set [2], [3]: all data displayed in this document have been collected either on Raspberry Pi4 or its System on Module (SoM) version the Compute Module 4, both based on the Broadcom BCM2711 quad core Cortex-A72 64-bit central processing unit clocked at 1.5 GHz in "performance" power settings. Opensource software is a core characteristics of the development capability throughout this document.

Despite such an approach demonstrating genuine positioning retrieval even under spoofing and jamming attacks [4], users of GNSS for timing capability need a hardware output of the 1-Pulse Per Second (1-PPS) and a typical 10-MHz frequency output. We demonstrate in this paper how a useful 1-PPS is generated and steered using the solution of the free, opensource implementation of GNSS decoder provided as `gnss-sdr` [5] supporting multiple constellations and frequency bands [6]. This solution is compatible with embedded applications since it has been demonstrated to run on single board computers [7] such as the Raspberry Pi4.

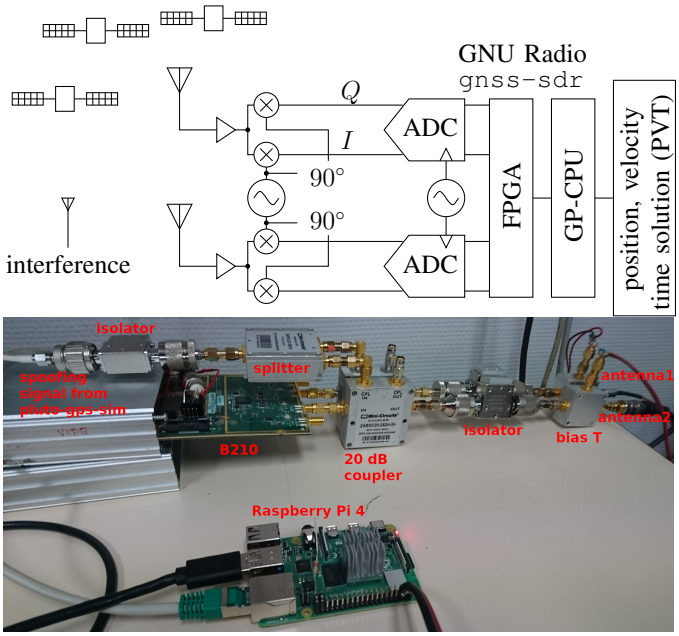

Fig. 1. Top: principle of SDR implementation of a secure GNSS receiver analyzing Direction of Arrival of the signals from the various satellites. In the SDR approach, as much hardware as possible is replaced with software running on the Field Programmable Gate Array (FPGA) or General-Purpose Central Processing Unit (GP-CPU), here with only the radiofrequency frontend amplification and frequency transposition still implemented as hardware before the analog to digital conversion. The external bias-T powering the preamplifier in the active antenna are not shown. Bottom: experimental setup, with a B210 SDR receiver fed by two bias-T-polarized active GNSS antenna connected to a Raspberry Pi 4 single board computer.

II. Timing consideration

The only known timing information in a SDR implementation is at the analog to digital converter (ADC) level under the assumption of a discontinuous datastream sampled periodically. All subsequent data transfers are asynchronous, whether between the buffer – most commonly in the FPGA handling the huge datarate from the ADC – and the general purpose processing unit, or in our case between the SDR receiver (Ettus Research B210 and its Analog Devices AD9361 radiofrequency frontend including low noise amplifier, tunable local oscillator, {I,Q} detector and ADCs) and the computer running GNU Radio communicating through a USB link. As such, the clock controlling the ADC will be steered by the Position, Velocity and Time (PVT) solution from the GNSS

FEMTO-ST Institute, Time and Frequency Department, Besançon, France.
[1] G.C. is currently Délégué Régional adjoint à la Recherche et à la Technologie but contributed significantly to this work
[2] F.M. is with OSU THETA, Besançon, France
Reference author JMF, e-mail: jmfriedt@femto-st.fr. Home page https://github.com/oscimp/gnss-sdr-1pps.

processing software, and this same clock must be used to control the 1-PPS output (Fig. 2).

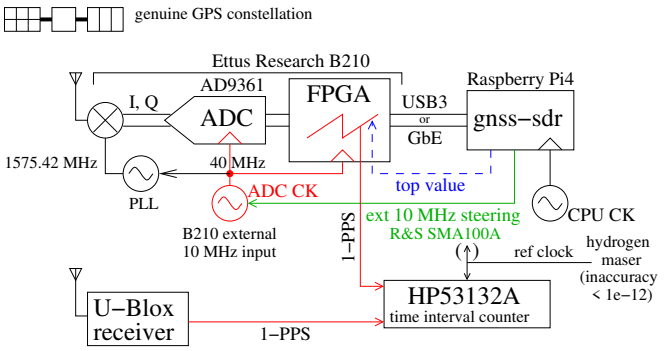

Fig. 2. 1-PPS synthesis architecture and analysis in the framework of an SDR receiver. The only sample timing information known to the user is the analog to digital converter since all subsequent data transfer is asynchronous. The aim is to steer the ADC and its FPGA controller clock with the GNSS PVT solution.

In this figure, the clock feeding the FPGA and the 1-PPS counter in the FPGA is indicated as controlled by a Hydrogen Maser (HM): this step is for preliminary assessment of the proper operation of the counter and its comparison with the hardware 1-PPS from the NEO-M8P U-Blox receiver, but the HM-controlled Rohde & Schwarz SMA100A synthesizer (Fig. 3) is later replaced with a Temperature Compensated Crystal Oscillator (TCXO) controlled Direct Digital Synthesizer (DDS).

The PVT solver analyzes the time offset between the three local copies of the pseudo-random Gold code sequence of each satellite – prompt, early and late – and deduces a global time offset between the local clock and the GNSS clock. This information is used to steer the clock controlling the FPGA driving the ADC clock.

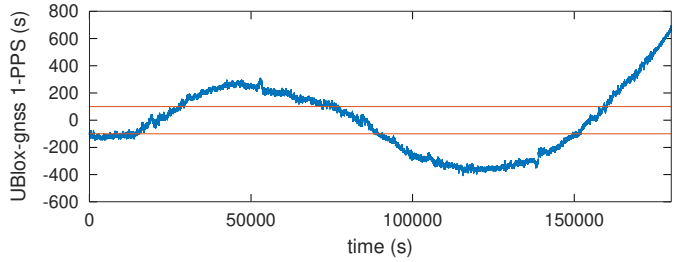

Fig. 3. Evolution over 50 hours of the time difference between a U-Blox NEO-M8P receiver and the 1-PPS counter implemented in the FPGA acquiring data to be processed by `gnss-sdr`, clocked by a *free running* SMA100A synthesizer set to a nominal frequency of 10 MHz. A drift of +11.1 ns/s=ppb has been removed and only the residue is displayed. Horizontal lines indicate the $\pm$100 ns targeted by controlling the oscillator on the PVT solution.

Since the B210 can be fed by an external 10-MHz reference, our implementation uses a radiofrequency synthesizer controlled to tune its output frequency used as external reference and track GNSS frequency. Because the 1-PPS phase is the integral of the frequency, the resulting 1-PPS counter will be synchronous with a hardware 1-PPS from a U-Blox receiver, within a constant offset.

III. Implementation

Thanks to the opensource software used to generate the FPGA bitstream running in the Ettus Research B210 Universal Software Radio Peripheral as provided at https://github.com/EttusResearch/fpga.git, the counter and control logic driving the 1-PPS generation have been added next to the existing dataflow handling logic as described at https://github.com/oscimp/gnss-sdr-1pps. As such, the 1-PPS logic shares the same clock than the ADC and hence both radiofrequency samples and 1-PPS front are driven with the same signal, controlled with the PVT time offset. Thanks to the opensource software `gnss-sdr` found at https://github.com/gnss-sdr/gnss-sdr and documented at https://gnss-sdr.org/ used to process the GNSS datastreams, the PVT time offset solution is acquired and processed in a Proportional, Integral (PI) control loop generating the signal driving the Rohde & Schwarz SMA100A synthesizer used to drive the 10-MHz nominal input of the B210. Notice that this 10 MHz reference signal is multiplied on the B210 board by 4 to generate the 40 MHz signal clocking the AD9361: while any sampling rate can be used to configure the AD9361, only integer divisions to 40 MHz will guarantee that the sampling rate is actually the expected value. If this condition is not met, `gnss-sdr` believes the sampling rate is the nominal value while the fractional PLL only reaches the closest achievable frequency, inducing a drift of the 1-PPS. As examples of usable sampling rates, 1.25 MHz (integer division by 32) or 2 MHz (integer division by 20) but even 1.125 MHz (fractional division by 320/9) have led to successful 1-PPS implementations as will be characterized below, while the theoretical 1.023 MHz sampling rate leads to a drifting 1-PPS output.

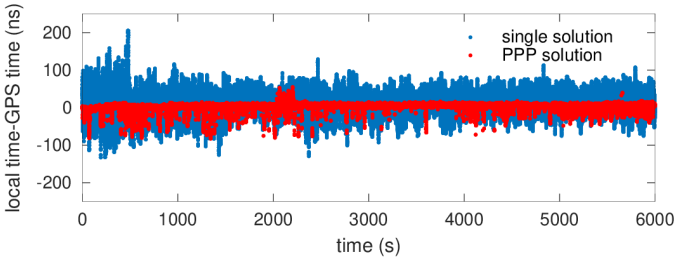

Fig. 4. Time offset evolution of the local time error deduced from the PVT solution with respect to GPS time. The comparison between the "single solution" and the PPP solution emphasizes the lower fluctuation of the latter. Notice that `gnss-sdr` provides the PPP time solution as range (i.e. multiplied by the speed of light in vacuum) while the Single solution is in second. The sampling period is 1-chip or 20 ms.

As a demonstration of the preliminary assessment of the proper operation of the 1-PPS implementation in the FPGA, the SMA100A synthesizer is clocked by the HM. In this case, the `gnss-sdr` generated 1-PPS only drifts over time with respect to GPS-time by the relative frequency offset of the HM, here $1.3{\cdot}10^{-12}$, as shown by the PVT solution corrected 1-PPS interval displayed in Fig. 4 with the drift hardly visible on such a short ($\simeq$ 2-h) acquisition. The PPP solution exhibits a lower fluctuation than the single solution. This analysis demonstrates that clocking the B210 with a high stability 10 MHz reference leads to a consistent and constant time offset of the local clock

with GPS time. We have also verified that the physical 1-PPS does not drift with respect to the hardware 1-PPS generated by the U-Blox receiver in this condition (data not shown).

Notice that the `gnss-sdr` configuration must *not* include a `resampler` step if the input and output frequencies are equal. Indeed, we have noticed that under such conditions, a single sample is dropped every 2^{32} samples. At a sampling rate of 1.125 MS/s, this occurrence happens once every $2^{32}/(1.125 \cdot 10^6) = 3817$ s or about 1 h, with a sudden shift by $1/1.125 = 0.888$ μs (Fig. 5, demonstrating the same argumentation when sampling at 2 MS/s). The solution is to replace the `resampler` with a `PassThrough` processing block in `gnss-sdr`.

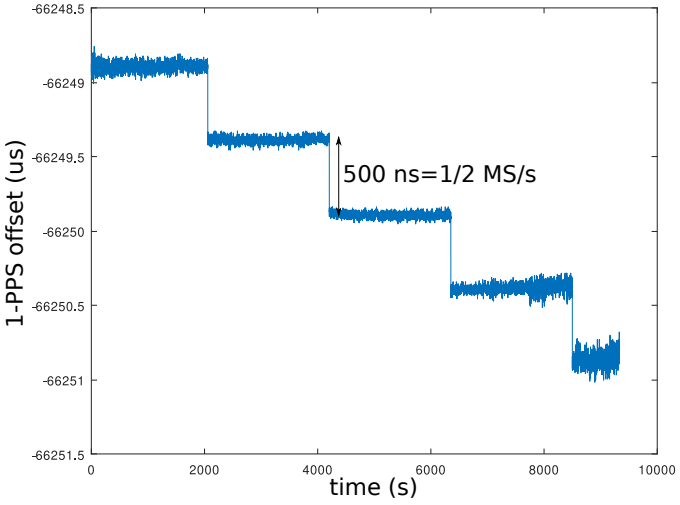

Fig. 5. Evolution of the time offset between a reference U-Blox receiver and the 1-PPS output deduced from processing `gnss-sdr` time difference information when including a `resampler` with the same input and output frequency. The time-delay jumps are induced by our erroneous result of the resampling block when input and output sampling rates are the same, as a sample is dropped every 2^{32}. At a sampling rate of 2 MS/s, this sample drop occurs every $2^{32}/(2 \cdot 10^6) = 2147.5$ s and the 1-PPS jump is equal to the inverse of the sampling rate or 500 ns in this demonstration.

IV. Results

A 39-hours long record of 1-PPS output delay comparison with a U-Blox NEO-M8P hardware receiver used as reference is displayed on Fig. 6. Not only does this chart demonstrate long term stability of the proposed solution after fine tuning the `gnss-sdr` parameters, but the frequency drift classically observed for even an excellent synthesizer as the Rohde & Schwarz SMA100A has been cancelled, and the short term stability in the 10^{-8} range is consistent with classical figures from hardware receivers. Here the control loop acts every `PVT.display_rate_ms=500` ms when the PVT solution is displayed, although the solution itself is updated every `PVT.output_rate_ms=20` ms, but the Hewlett Packard HP53132A time interval counter is set to integrate on the 1-s interval of the 1-PPS, hence providing one output every second. The HP53132A counter reference clock is a HM known to exhibit a much lower Allan deviation (below 10^{-12} over the integration times considered here) on this time interval.

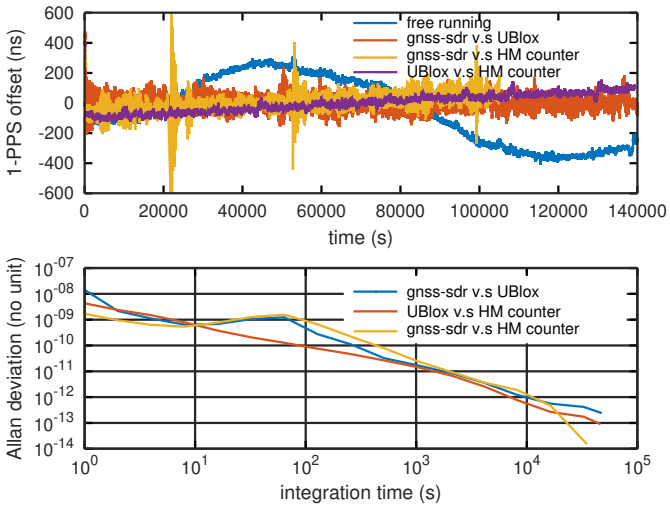

Fig. 6. Top: 39 h long record of the time interval between the 1-PPS output of a U-Blox Neo-M8P receiver, our SDR implementation of the 1-PPS output driven by the PVT solution extracted from `gnss-sdr` and a counter clocked by a Hydrogen Maser (HM) drifting with respect to GPS time. The free running oscillator chart is the same as shown on Fig. 3 for comparison (different scale). Bottom: resulting Modified Allan deviation of the various 1-PPS comparisons.

Improvement on the control implementation is still needed as the bump on the Allan deviation between 10 and 100 s is attributed to the local oscillator steering control signal.

V. From OCXO to TXCO driven clock

The SMA100A is an excellent (short term phase noise and long term Allan deviation) clock source controlled by an Oven Controlled Crystal Oscillator (OCXO), well suited for a laboratory demonstration but hardly usable for a real work application where power consumption, size and cost are all issues to be tackled for a use beyond the laboratory.

Replacing the OXCO driven SMA100A with a TCXO driven 32-bit resolution Analog Devices AD9959 DDS stresses the control loop and degrades not only the short term performance but also the long term stability as shown in Fig. 7. The 18.6 mHz DDS frequency resolution when clocked by a 20 MHz TCXO internally multiplied by 4 is similar to the 10 mHz frequency resolution of the SMA100A synthesizer, hence hardly deteriorating the control capability. Clocking the DDS with the SMA100A allows us to validate that the source of the frequency fluctuations is indeed the TCXO and not the DDS since the initial performance is recovered in the latter case (data not shown).

One of the challenging issues we met was the slow convergence of the frequency control loop when starting too far from setpoint. Indeed, the TCXO exhibits very poor accuracy at startup, and only once the GNSS constellation signal has been locked and the time drift is identified can the DDS control word be tuned to correct for the TCXO offset. In order to avoid the slow convergence of the control loop, a preliminary openloop identification of the frequency offset is performed during the first few seconds after the GNSS signal has been decoded and its timing signal is being tracked, before setting the time offset setpoint and the frequency offset is roughly

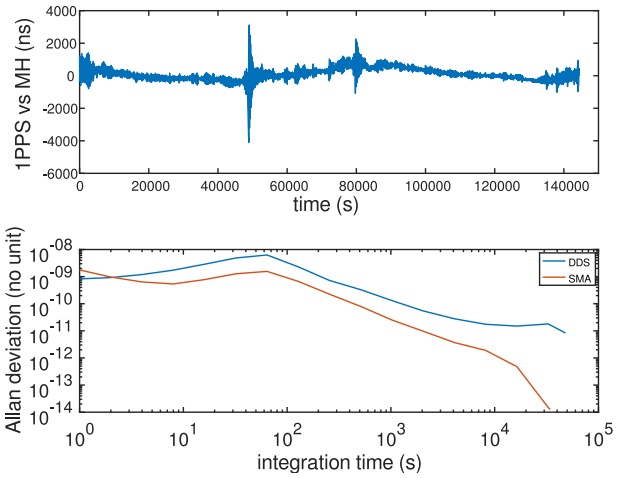

Fig. 7. Time-evolution of the 1-PPS interval (top) and resulting Allan deviation when clocking the B210 with the output of an AD9959 DDS clocked by a 20 MHz TCXO multiplied by 4 by the internal Phase Locked Loop. The SMA100A reference measurement is reproduced for comparison.

corrected prior to switching on the control loop. This sequence is experimentally demonstrated on Fig. 8.

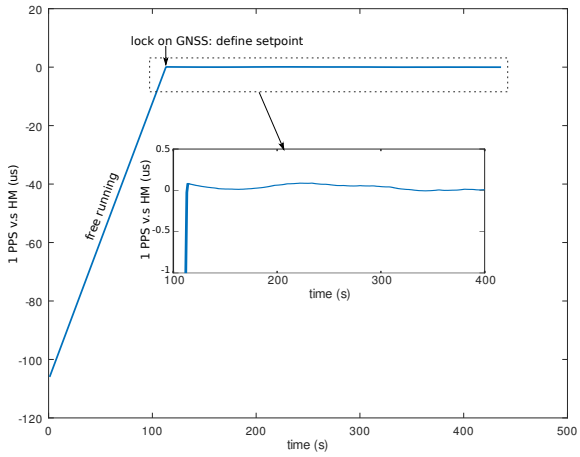

Fig. 8. TXCO controlled DDS generating the signal driving the B210 SDR receiver feeding `gnss-sdr`. Prior to GNSS constellation acquisition, the TCXO freely drifts until the GNSS signal is tracked: a few seconds of drift are observed to identify the frequency offset and correct digitally the AD9959 DDS output prior to setting the setpoint time offset to be tracked by the control loop.

Since the time offset information between the local replica of the clock and GPS time is provided within a 20 ms interval – 1-chip length – an uncertainty remains on the phase output of our 1-PPS multiple of 20 ms. Nevertheless, this output signal is useful for controlling the frequency output of an external oscillator despite the random phase of our current 1-PPS output implementation.

VI. Conclusion

This contribution bridges the virtual output of SDR and its application to GNSS decoding with a hardware 1-PPS output useful for driving external oscillators. Having identified the clock controlling both the radiofrequency frontend fitted with the analog to digital converters as well as the FPGA as the timing information, an external frequency source feeds this clock signal and is controlled with respect to the GNSS timing information for steering the frequency. The long term drift is hence cancelled and the GNSS performance of 10^{-12} and 10000 s integration time is recovered. This implementation solves the frequency transfer challenge while an unknown phase offset remains on the 1-PPS output preventing time transfer. This issue is now under investigation. The software implementing the 1-PPS output when fetching GNSS information from an Ettus Research B210 SDR receiver clocked by a Rohde & Schwarz SMA100A synthesizer is available at https://github.com/oscimp/gnss-sdr-1pps.

Acknowledgements

`gnss-sdr` is free, opensource software provided by the Centre Tecnològic Telecomunicacions Catalunya (CTTC, Spain): C. Fernández-Prades and J. Arribas are acknowledged for fruitful discussions. This investigation is motivated by the FAST-LAB joint laboratory between FEMTO-ST, the Besançon Observatory (OSU THETA) and the company Gorgy Timing. The infrastructures of the Oscillator Instability Measurement Platform (OscillatorIMP) provide the reference signals.

References

[1] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A software-defined GPS and Galileo receiver: a single-frequency approach*. Springer Science & Business Media, 2007.

[2] J.-M. Friedt, D. Rabus, and G. Goavec-Merou, "Software defined radio based global navigation satellite system real time spoofing detection and cancellation," in *GNU Radio Conference (GRCon)*, 2020, https://pubs.gnuradio.org/index.php/grcon/article/view/73.

[3] G. Goavec-Merou and J.-M. Friedt, "Never compile on the target ! GNURadio on embedded systems using Buildroot," in *Free Open Source Developer Meeting (FOSDEM)*, 2021, https://fosdem.org/2021/schedule/event/fsr_gnu_radio_on_embedded_using_buildroot/.

[4] W. Feng, J.-M. Friedt, G. Goavec-Merou, and F. Meyer, "Software defined radio implemented GPS spoofing and its computationally efficient detection and suppression," *IEEE Aerospace and Electronic Systems Magazine*, 2021.

[5] C. Fernández–Prades, J. Arribas, P. Closas, C. Avilés, and L. Esteve, "GNSS-SDR: An open source tool for researchers and developers," in *Proc. 24th Intl. Tech. Meeting Sat. Div. Inst. Navig.*, Portland, Oregon, Sept. 2011, pp. 780–794.

[6] C. Fernández-Prades, J. Arribas, L. Esteve, D. Pubill, and P. Closas, "An open source Galileo E1 software receiver," in *2012 6th ESA Workshop on Satellite Navigation Technologies (Navitec 2012) & European Workshop on GNSS Signals and Signal Processing*. IEEE, 2012, pp. 1–8.

[7] J.-M. Friedt, W. Feng, D. Rabus, and G. Goavec-Merou, "Real time GNSS spoofing detection and cancellation on embedded systems using software defined radio," in *EuCAP*, Düsseldorf, Germany, 2021.