FPGA Implementation of a NCO based CDR for the JUNO Front-End Electronics

F. Marini, M. Bellato, A. Bergnoli, R. Brugnera, F. dal Corso, D. Corti, J. Dong, A. Garfagnini, A. Giaz, G. Gong, J. Hu, R. Isocrate, X. Jiang, I. Lippi, K. von Sturm, S. Aiello,G. Andronico, V. Antonelli, W. Bandini, D. Basilico, A. Brigatti, A. Barresi, A. Budano, R. Bruno, B. Caccianiga, A. Cammi, R. Caruso, D. Chiesa, C. Clementi, S. Costa, X. Ding, S. Dusini, A. Fabbri, M. Fargetta, R. Ford, A. Formozov, M. Giammarchi, M. Grassi, C. Landini, P. Lombardi, C. Lombardo, F. Mantovani, S. M. Mari, C. Martellini, A. Martini, E. Meroni, M. Mezzetto, L. Miramonti, P. Montini, M. Montuschi, M. Nastasi, F. Ortica, A. Paoloni, S. Parmeggiano, N. Pelliccia, E. Previtali, G. Ranucci, D. Riondino, A. C. Re, B. Ricci, A. Romani, P. Saggese, A. Serafini, C. Sirignano, M. Sisti, L. Stanco, V. Strati, M. Torri, C. Tuvé, G. Verde, L. Votano

***Abstract*—This paper describes a design of an FPGA implementation of a Clock and Data Recovery (CDR) system. The core will be integrated in the FPGA configuration for the front-end electronics board of the Jiangmen Underground Neutrino Observatory (JUNO) experiment. The front-end will be placed on the main detector, underground and underwater, making the electronics not accessible after installation. The timing and trigger system relies on a synchronous link connection over CAT5e cable (up to 100 meters long) between the front-end and the back-end electronics, where a twisted-pair is dedicated to clock-forwarding. The robustness of the recovery clock system is essential for the stability of the FPGA firmware. The proposed project is intended to improve the clock recovery operation by increasing the immunity of the link to sudden electromagnetic interference. On top of this, the core allows to free a twisted-pair in the link, since the clock can be recovered from the data and there is no more need for a clock-dedicated transmission. This will optimize the link granting the possibility to implement other features. The design is based on two components: a Numerically-Controlled Oscillator (NCO), in order to create a controlled frequency clock signal, and a digital Phase Detector (PD) to match the clock frequency with the data rate. NCOs are often coupled with a Digital to Analog Converter (DAC) to create Direct Digital Synthesizers (DDS), which are able to produce analog waveforms of any desired frequency. In the presented case instead, the NCO generates a digital clock signal of an arbitrary frequency, while the PD manages this frequency by intercepting any shifting on the relative phase between the clock and the data. A Phase Aligner (PA) module guarantees that data is sampled in the middle of the eye pattern, which represents the optimal sampling point. The paper presents an overview of the NCO-based CDR design and implementation, together with some tests and results in order to verify the clock and data recovery reliability. Moreover, in the last section some other possible applications of the core are illustrated.**

***Index Terms*—CDR, NCO, phase detector, phase aligner, FPGA, eye pattern**

F. Marini, R. Brugnera, A. Garfagnini, A. Giaz, K. von Sturm, M. Grassi and C. Sirignano are with the Department of Physics and Astronomy, Padova University, Padua, Italy and INFN Padova, Padua, Italy (e-mail: filippo.marini@pd.infn.it)

M. Bellato, A. Bergnoli, F. dal Corso, D. Corti, R. Isocrate, I. Lippi, S. Dusini, M. Mezzetto and L. Stanco are with INFN Padova, Padua, Italy

J. Dong and G. Gong are with Tsinghua University, Beijing, China

J. Hu and X. Jiang are with the Institute of High Energy Physics, Beijing, China

S. Aiello, G. Andronico and R. Bruno are with INFN Catania, Catania, Italy

R. Caruso, M. Fargetta, C. Lombardo, C. Tuvé and G. Verde are with the Department of Physics and Astronomy, Catania University, Catania, Italy and INFN Catania, Catania, Italy

S. Costa is with Polytechnic University of Milan, Milan, Italy and INFN Catania, Catania, Italy

V. Antonelli, D. Basilico, B. Caccianiga, X. Ding, A. Formozov, M. Giammarchi, C. Landini, P. Lombardi, E. Meroni, L. Miramonti, S. Parmeggiano, G. Ranucci, A. C. Re, P. Saggese and M. Torri are with the Department of Physics, Milano University, Milan, Italy and INFN Milano, Milan, Italy

R. Ford is with the Department of Physics, Milano University, Milan, Italy, INFN Milano, Milan, Italy and SNOLAB, Lively, Ontario, Canada

W. Bandini, F. Mantovani, M. Montuschi, B. Ricci, A. Serafini and V. Strati are with the Department of Physics and Earth Science, Ferrara University, Ferrara, Italy and INFN Ferrara, Ferrara, Italy

A. Barresi, D. Chiesa, M. Nastasi, E. Previtali and M. Sisti are with the Department of Physics, Milano Bicocca University, Milan, Italy and INFN Milano Bicocca, Milan, Italy

A. Brigatti is with the Department of Physics, Milano Bicocca University, Milan, Italy

A. Cammi is with the Polytechnic University of Milan, Milan, Italy and with INFN Milano Bicocca, Milan, Italy

C. Clementi, F. Ortica, N. Pelliccia and A. Romani are with the Department of Chemistry, Biology and Biotechnology, Perugia University, Perugia, Italy and INFN Perugia, Perugia, Italy

A. Budano, A. Fabbri, S. M. Mari, C. Martellini, P. Montini and D. Riondino are with the Department of Mathematics and Physics, Roma Tre University, Rome, Italy and INFN Roma Tre, Rome, Italy

A. Martini, A. Paoloni and L. Votano are with INFN Frascati National Laboratories, Frascati, Italy

I. Introduction

The capability to extract timing information out of a serial data stream to recover incoming data has become a very common requirement, due to the proliferation of serial communication systems. Present readout systems in physics experiments usually rely on FPGAs to receive and transmit data at high rate to high capacity DAQ systems; exploiting FPGAs to regenerate the bit stream is therefore beneficial for a number of reasons, including power consumption and cost reduction. Some FPGA embedded mechanisms are exploited to only recover the incoming messages without recovering the clock as well [1] with many of them using oversampling techniques [2] [3]; other mechanisms rely on clock-centric serial links where synchronous data is embedded into a clock signal by modulation techniques [4]. While this would allow to send both the clock and the data in a single channel, it does not introduce improvements in the recovered clock stability

compared to a link where the clock has its own dedicated transmission lines.

The design presented in this paper allows to recover both the data and the clock from a serial Low-Voltage Differential Signaling (LVDS) data stream after traveling for 100 meters in a CAT5e cable, by emulating a Voltage-Controlled Crystal Oscillator (VCXO) with an NCO which only uses a SerDes and a clock management tile to obtain a dynamically controlled frequency clock signal.

This core is intended to be used in the context of the Jiangmen Underground Neutrino Observatory (JUNO) [5], a neutrino physics experiment featuring a 20 kton liquid scintillator detector surrounded by 18000 20-inch PhotoMultipliers Tubes (PMTs) aiming to recover the energy of incident anti-electron nuclear-reactors neutrinos with an unprecedented energy resolution of 3% at 1 MeV [6]. The Front-End Electronics (FEE) [7] principal component, the so called Global Control Unit (GCU) features an FPGA that manages the data readout, data buffering, slow control and monitoring, trigger and synchronization with the Back-End Electronics (BEE) [8]. To maintain the timing synchronization over time, it is essential that both the FEE and BEE internal clocks are synchronized within the same clock domain, to avoid any clock drifting over time. By recovering the clock signal from the data stream exchanged from the BEE to the FEE, the proposed CDR implementation is suitable for the application. While the data readout and slow control is managed through a 1000BASE-T Ethernet asynchronous link over a CAT5e Unshielded Twisted Pairs (UTP) cable, for the timing and trigger system a custom synchronous protocol running over up to 100 meters of CAT5e Foiled Twisted Pairs (FTP) cable has been developed. The CDR system proposed in this paper would be exploited to recover both the clock and the data exchanged by this synchronous link.

A. The Synchronous Link protocol

The synchronous link protocol developed for JUNO provides all signals necessary for the timing synchronization, trigger and arbitrary control data. JUNO employs this link with a star topology, where the hub, consisting of a Back-End Card (BEC), connects to 48 GCUs. The physical link consists of a standard CAT5e Foil Twisted Pair (FTP) cable, up to about 100 meters long.

The four twisted pairs transport

- 125 Mbps Trigger requests (GCU to BEC)
- 125 Mbps Synchronous messages (Bidirectional)
- 62.5 MHz Digital Clock (BEC to GCU)

An adaptive cable equalizer chip is employed at the receiver side of the cable. Since this chip requires no DC component on the transmission, the synchronous messages are Manchester encoded (plus Hamming encoded for 1 bit error correction), while the trigger requests use a scrambler mechanism, to avoid any bandwidth waste. The digital clock sent from the BEC, is used directly as input to two FPGA's Phase Locked Loops (PLL), the first set as a jitter cleaner, the second used to derive all the needed frequencies.

Even though the design is proved to work in controlled environments, the JUNO laboratory is still under construction and therefore the level of ElectroMagnetic Interference (EMI) is unknown; for example, possible noise and interference can be caused by the complex calibration systems that are currently being developed [9]. This interference may affect the link, in particular resulting in loss of clock locking by the GCU's PLL. The stability of the recovered clock from the BEC is an essential requirement for the front-end electronics operations, especially concerning the timing synchronization with the back-end. If the PLL in the GCU's FPGA lose the locking condition, the synchronization is lost and the whole firmware gets reset. Therefore a CDR can be employed to recover the 62.5 MHz digital clock (125 Mbps) to increase the recovered clock immunity to sudden EMI. Moreover, a CDR would allow the clock to be recovered directly from the 125 Mbps Manchester encoded stream, freeing up a twisted pair for extra-features. The proposed design, its FPGA implementation and some test results will be shown in the next sections.

II. CDR Digital Design Overview

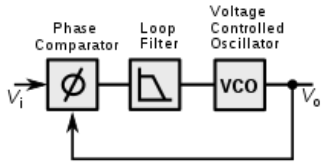

Fig. 1. The standard PLL architecture.

Usually A CDR architecture is similar to the PLL model (fig. 1), where the phase of a reference signal is compared to the phase of an adjustable feedback signal, generally provided by a controlled oscillator, like a Voltage Controlled Oscillator (VCO). The output of the Phase Detector (PD) is filtered and used to drive the VCO frequency. When the phase comparison is in steady state, e.g. the phase and frequency of the reference signal is equal to the phase and frequency of the feedback signal, we say that the PLL is locked. In the case of a CDR, the steady state is reached when the VCO clock frequency matches the reference signal's data rate.

The design is implemented in a Xilinx KC705 evaluation board [10] featuring a Xilinx Kintex-7 FPGA. An overview of the proposed CDR architecture is given in fig. 3. The design mimics the PLL architecture, consisting of a Numerically Controlled Oscillator (NCO) which is used to create a frequency controlled clock, a Phase and Frequency Detector (PFD) monitoring the NCO's clock frequency to match it with the data rate, and a Phase Aligner (PA) that, together with the Xilinx 7 Series MMCME2_ADV tile [11], dynamically adjusts residual clock drifting and have a deterministic phase relationship with the incoming data stream. As shown in the figure, many control signals are delivered between different modules to dynamically control phase and frequency. These logic signals may intersect Clock Domain Crossing (CDC) boundaries; for example, the *PFD unit* is in the NCO's clock domain, while the *Frequency Manager*, as well as the NCO

itself, are synchronized with the system clock. To comply with the CDC boundary crossing, the control signals are stretched and sent together with a CDC signal to avoid erroneous sampling [12]. A timing diagram example is shown in fig. 2.

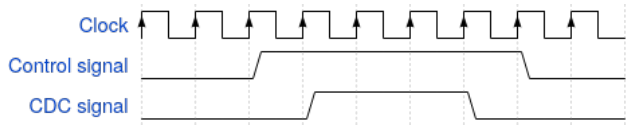

Fig. 2. Timing diagram example to comply with CDC crossing.

The expected recovered clock frequency of 125 MHz allows the use of the high range (HR) general purpose I/O pin of the FPGA rather than dedicated transceivers, resulting in a reduced power consumption and a more straightforward design. Moreover, the code is easily portable to a different FPGA architecture, as the VHDL core is generic and no proprietary Xilinx Intellectual Properties (IP) cores are used. Nevertheless, particular attention should be paid for the porting of I/O logic resources (SerDes) and clock management tiles, which must be features supported by the FPGA foreseen to employ this solution.

On the following sections a description of the implemented hardware is provided, together with some details of the VHDL code used to generate it.

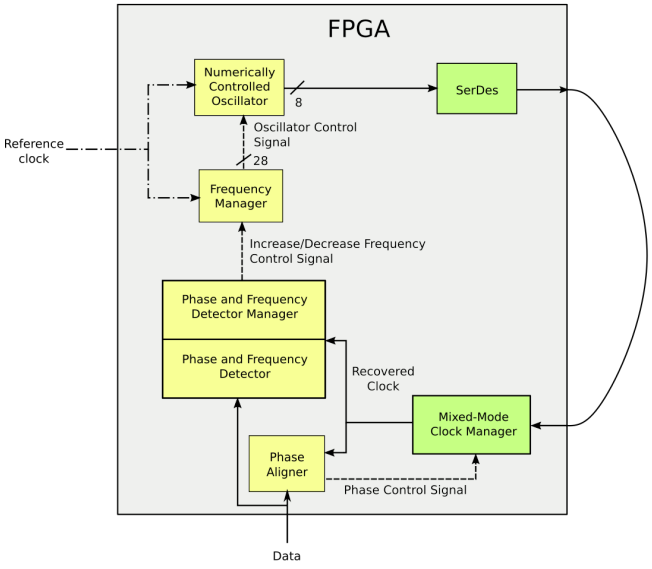

Fig. 3. Block diagram for the proposed CDR design. Yellow blocks represents custom VHDL modules, green blocks are used to represent FPGA proprietary tiles. Dashed lines are used for control signals.

A. Numerically Controlled Oscillator

The NCO [13] design consists of two parts:

- A Phase Accumulator (PACC), which is basically a counter driven by a reference clock incremented by a user-defined value related to the nominal data stream rate.
- A phase-to-amplitude converter, which uses the PACC output as an index to a Look-Up Table (LUT)

To better understand the mechanism, we can think of a phase-wheel (fig. 4). This phase-wheel is equally divided in a certain number of sections, bounded by phase-points (the PACC output) and for each phase-point we associate the corresponding sine value (this association process is done by the LUT). As a vector rotates around the wheel, by taking these correlated sine values a digital sine waveform is generated. A complete revolution around the phase-circle corresponds to a complete period of the sine wave. Imagining now that the vector skips

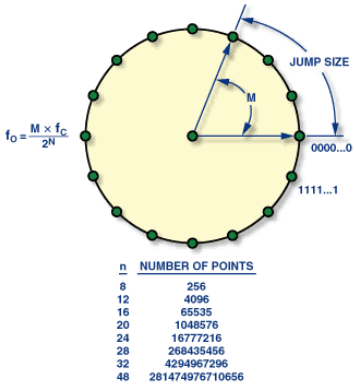

Fig. 4. The phase-wheel.

a few (fixed) points each jump, the revolution is completed in a much shorter time. As a result, the frequency of the output waveform has increased. The correlation between the jump size, the reference clock and the output waveform frequency is

$$f_{OUT} = \frac{M \times f_C}{2^N} \tag{1}$$

where:

- M is the jump size
- f_{OUT} is the NCO output waveform frequency
- f_C is the reference clock frequency, generated by a local oscillator on board
- N is the number of bits dedicated to the PACC counter

To retrieve a digital clock signal, the LUT design is simple: half of the circle is associated to the digital value 0, while the other half to the digital value 1. However, the design presents a limitation in the phase resolution. Since the output signal is digital, the time domain is discrete and it corresponds to the reference clock period. This implies that the positive (and negative) fraction of the output clock signal can only be a multiple of this time domain resolution, making the output frequency only on average determined by the jump size of the accumulator.

In the next chapter, a way to increase the phase resolution is illustrated.

1) The SerDes Technique: The most straightforward way to improve the phase resolution is to increase the reference clock frequency. Graphically, going back to the phase-circle example, this would result in the vector taking shorter pauses between jumps, thus increasing the time (phase) resolution of the resulted waveform. Even though this approach is correct, the reference clock frequency must comply with the FPGA

architecture, and can not be increased over a certain threshold to avoid timing errors.

A different approach relies on the parallelism capability of the FPGA. Basically, instead of decreasing the jump's time interval by increasing the reference clock frequency, we generate multiple phase-wheels with equal jump-size and for each one we compute a different phase-point at the same rising edge of the reference clock. The phase-points offset between each phase-wheel is determined by

$$offset = \frac{M}{PW} \tag{2}$$

where PW is the number of phase-wheels generated and M is the jump size (Note that M must be equal or greater than PW. If the result is fractional, *offset* is approximated to the closest integer). The PW number of waveforms generated will be merged, to retrieve a single, higher resolution, waveform. To clarify, fig. 5 presents an example where two phase-wheels are being generated. For graphical reason, the output will be represented as a sine waveform instead of a digital 50% duty cycle clock.

The reference clock has a frequency of 250 MHz, therefore every phase-wheel evaluates a phase-point every 4 ns (Note that the green and blue points are computed at the same time) and applying eq. 1, the resulted waveform's frequency is 62.5 MHz. According to eq. 2, the second phase-wheel vector presents a offset of 1 phase-point compared to the first one ($PW = 2$, $M = 2$). Merging the two plot (*i.e.*, serializing the points) cuts in half the period between the points, increasing the waveform time resolution. The translation of

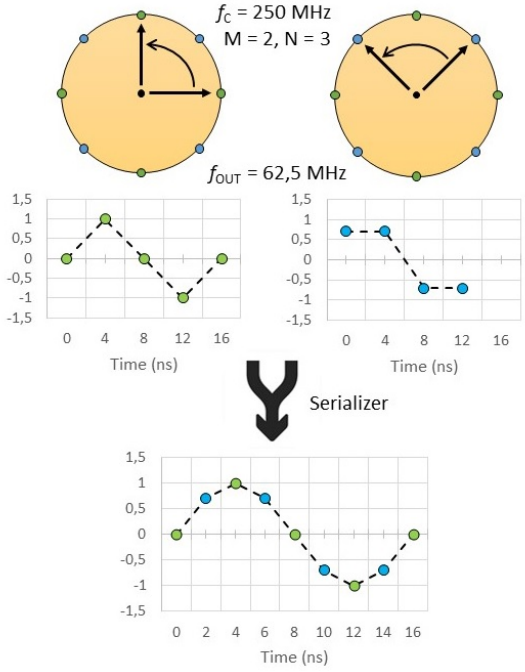

Fig. 5. Example of the phase-wheel parallelization technique.

this design to a Register-Transfer Level (RTL) architecture is made through the use of an FPGA I/O resource called SerDes (Serializer/Deserializer), which is a dedicated parallel to serial converter for the implementation of high-speed synchronous interfaces, present in the Xilinx 7 series devices under the name of OSERDESE2 [14].

In the actual implementation the reference clock is 125 MHz and eight PACC are present, which means that 8 phase-points are translated to either 0 or 1 by the LUT every 8 ns. These are fed to the OSERDESE2 resource which serializes them using a 500 MHz clock in Double Data Rate (DDR) mode. The digital output results in a digital clock signal with a phase resolution of 1 ns (By evaluating one point on the phase wheel every nanosecond, the maximum frequency obtainable by the NCO is 500 MHz. Of course such a high frequency is not feasible, as it would generate timing errors on the following CDR modules, such as the *Phase and Frequency Detector*). Since the OSERDESE2 output can only be connected to an Input/Output Block (IOB), a loop-back in and out of the FPGA is a precondition to be foreseen when designing the Printed Circuit Board (PCB). By setting a top-level generic value called *g_multiplication_factor*, the user is able to generate output frequencies which are higher than the single phase-wheel maximum output frequency (which is half of the system clock due to the Nyquist theorem). The equation for the overall NCO output frequency is no more Eq. 1 but it is substituted by

$$f_{OUT} = \frac{M \times f_C}{2^N} * 2^{g\_multiplication\_factor-1} \tag{3}$$

When setting the generic, the user must keep in mind that the following rule must apply:

$$f_{OUT}/2^{g\_multiplication\_factor-1} < f_C/2 \tag{4}$$

B. Phase and Frequency Detector

To mimic the PLL architecture, a PFD [15] is needed to compare the NCO output clock frequency to the data rate. Denoting with f_d the data frequency and with f_{NCO} the clock frequency, we have that:

$$f_d = (\phi_d(t_1) - \phi_d(t_0))/(t_1 - t_0) \tag{5}$$

$$f_{NCO} = (\phi_{NCO}(t_1) - \phi_{NCO}(t_0))/(t_1 - t_0) \tag{6}$$

where $\phi_d(t)$ and $\phi_{NCO}(t)$ represents the data and clock phase respectively at the time t. To detect a frequency difference $f_d - f_{NCO}$, the times t_1 and t_0 shall be provided by the NCO itself, which is the only time-base available in the CDR. In fact, t_1 and t_0 shall come from the NCO dedicated waveforms (clocks). The frequency difference is then given by:

$$f_d - f_{NCO} = \frac{[(\phi_d(t_1) - \phi_{NCO}(t_1)) - (\phi_d(t_0) - \phi_{NCO}(t_0))]}{(t_1 - t_0)} \tag{7}$$

The two phase differences in the numerator at the right hand side of the equation are the output of the phase detector, which compares the data transitions with the NCO clock transitions at the instances t_1 and t_0. These phase differences will vary with time (in case of frequency mismatch), making a frequency difference detection possible.

In the implemented design, the frequency detection capability relies on the use of two clock signals, with 50% duty cycle and orthogonal with each other ($\pi/2$ of phase difference). This

allows the division of the entire 360 degrees clock period into four quadrants, as shown in fig. 6.

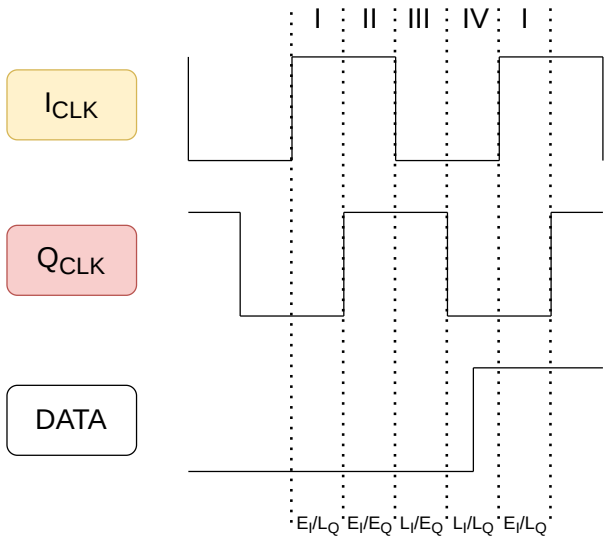

Fig. 6. Two synchronous clocks with 50% duty cycle, I_{CLK} (In-phase Clock) and Q_{CLK} (Quadrature Clock), divide the period T in four quadrants identified by their positive and negative transitions. Q_{CLK} leads I_{CLK} by 90 degrees. Thanks to the Early/Late (E/L) identification by the Phase Detector Unit, it is possible to identify where the *DATA* transitions reside, the fourth quadrant in the example.

The *Phase and Frequency Detector* module (fig. 7) features a *Phase Detector Unit* which exploits the use of a couple of PD whose output is forwarded to the *Frequency Detector Unit* to determine whether the data transition edges are shifting up or down in the clock quadrants. This information is passed to the *Phase and Frequency Detector Manager* which acts as a low pass filter for the frequency shifting requests (filtering is mandatory due to meta-stability and wrong sampling when data edges are close to clock edges) as well as a controller for the CDR lock flag.

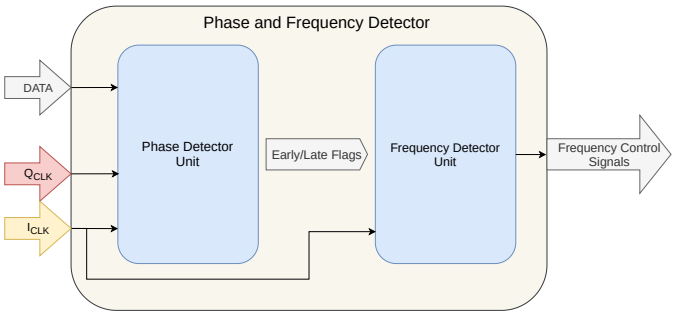

Fig. 7. Block diagram of the Phase and Frequency Detector module. The *Frequency Control Signals* identify whether the data edge is shifting up or down the four quadrants defined by the two clocks in quadrature I_{CLK} and Q_{CLK}.

1) Phase Detector Unit: As mentioned in Section II-B, the phase and frequency detection capability of the PFD relies on two PD and a first stage of low-pass filtering. The implemented PD are Alexander type Bang-Bang Phase Detectors [16], which is strongly non-linear since it is only able to detect the sign of the phase difference, without collecting any information about the amount of the phase difference it detects. The early/late output pulses are filtered by the *Phase Detector Filter* Master and Slave couple. The filtering is based on an early/late counter inside a predefined time length window given by the Master. To asses a transition, the counter, which increases its value when the PD outputs a 'late' pulse and decreases it when 'early', must exceed a certain threshold. Moreover, a minimum number of transitions must be detected for an early/late decision to be approved. Fig. 8 shows the block diagram of the PD unit.

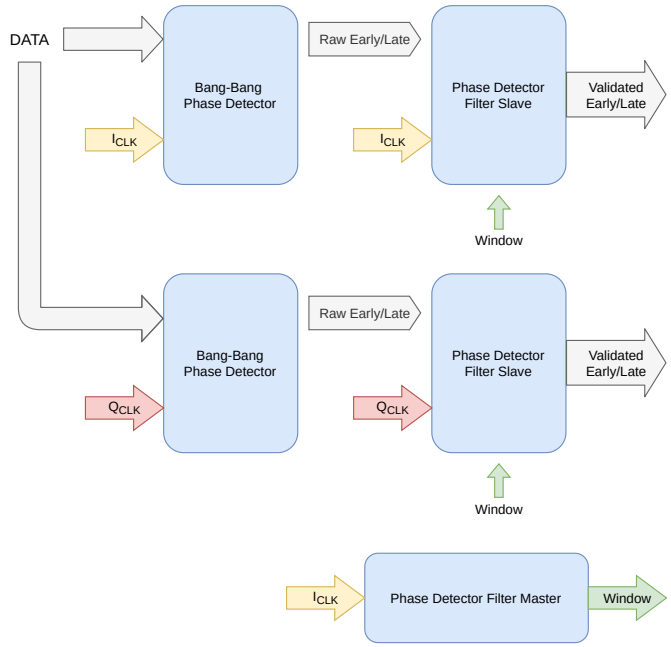

Fig. 8. Block diagram for the PD unit design.

2) Frequency Detector Unit: The Frequency Detector (FD) unit consists of two modules: a *Quadrant Detector* and a *Quadrant Shifting Detector*. Basically, the first is able to detect the current quadrant where the data edges resides, based on the early/late information coming from the PD unit, while the second stores and analyzes this information to determine whether this current data edges quadrants are drifting up ($f_{NCO} > f_d$) or down ($f_{NCO} < f_d$).

A block diagram is shown in fig. 9.

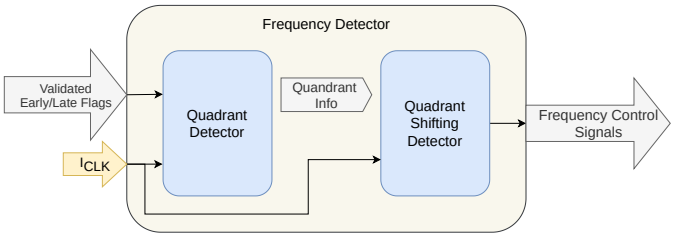

Fig. 9. Block diagram for the FD unit design.

C. Phase and Frequency Detector Manager

The *PFD Manager* purpose is to make sure the NCO's clock frequency matches as close as possible the data rate.

When this condition is met, the CDR locked flag is asserted High, otherwise, the locked flag is de-asserted if the input data rate mismatches the NCO clock frequency. Anyway, the CDR always tries to follow slow data rate fluctuations.

The CDR locked condition is not straightforward to declare. An ideal behavior would consist in a decrease of the quadrant shifting detection rate as the clock frequency approaches the data rate, getting to zero when locked (or oscillating between up and down shiftings, since the possible NCO frequencies are discrete). Unfortunately, this is not the observed behavior in practical implementations: real-world clocks always contain some jitter, plus the correct sampling is not guaranteed when the Setup/Hold time requirements are violated. This results in glitches around quadrant transitions. To comply with this un-predictable scenario, the *PFD Manager* evaluates the frequency shifting requests and the lock condition based on a counter mechanism: A Finite State Machine (FSM) counts the number of frequency increase (+1) and decrease (-1) requests inside a defined window and different ranges defines different behaviors:

- $\pm$ *lock threshold* ($\sim$ 10% of the maximum value): inside this range the CDR is locked.
- $\pm$ *activate threshold* ($\sim$ 50%): when locked, outside this range a frequency change request is forwarded to the NCO.
- $\pm$ *unlock threshold* ($\sim$ 90%): if exceeded, the CDR lock flag gets de-asserted.

By employing this mechanism, the CDR takes several seconds to lock, which is compatible with the JUNO requirements since it is expected to only lock once during the whole data taking phase ($\sim$ 6 years). Actually the locking time is strongly dependent on the maximum possible value of the counter, which is defined by the duration of the window. The longer the window, the harder it is to unlock the CDR once it is locked (*i.e.* the CDR can sustain longer interference on the data channel), but the slower it is to follow possible source clock drifting and the longer it takes to lock. So a trade-off here is necessary.

The frequency change requests are passed to the *Frequency Manager* module, which will increase or decrease the jump size accordingly. As mentioned in Section II-B1, the PD is not able to measure the extent of the phase difference, thus each request will change the NCO's jump size by a pre-defined fixed value, independently from the magnitude of the frequency-data rate mismatch.

D. Phase Aligner

The task of matching perfectly the NCO's frequency with the data rate as well as providing a deterministic phase relationship is taken care by the *Phase Aligner* module. Once the PFD is locked, a Bang-Bang PD dynamically operates directly on the MMCM, to generate an extra clock signal phase aligned with the data rate. This is made possible thanks to the phase shifting capability of the Xilinx 7 Series MMCME2_ADV.

The phase alignment step is mandatory in this CDR design to eliminate any residual drifting of the clock respect to the data stream and to make sure that data sampling is performed in the middle of the eye pattern, which is the optimal sampling point.

III. CDR Testing

The testing of the FPGA-implemented CDR is intended to give some quantitative indications on the data transmission reliability of the JUNO's synchronous link design. Following, a description of the test performed and their results.

A. Test Setup

To evaluate the design, the proposed CDR has to recover the clock and the data from a 125 Mbps Pseudo-Random Binary Sequence (PRBS) generated from a different board. To better emulate the JUNO conditions, the physical medium which transmits the sequence is a 100 meters long CAT5e Foiled Twisted Pair (FTP) cable.

The block design for the setup is illustrated in fig. 10 and 11. A first GCU board generates the PRBS sequence and exploits the on-board differential cable buffers to forward it to the cable where, reaching the end, a second GCU recovers the signal through the cable equalizer chip and redirects it to a Micro-Miniature Coaxial (MMCX) connector and to a differential GPIO pin pair. No logic operations are performed inside this second FPGA. From the GCU's MMCX connector and through an FMC interconnection board, the PRBS sequence arrives at the Xilinx KC705's FPGA, where the actual CDR operation takes place. After the clock is recovered, this is used to sample the incoming data and, thanks to a PRBS checker module [17], monitor any error in the reconstructed sequence.

The recovered clock from the CDR is also used to trigger an oscilloscope, where the second GCU's differential pair signal is monitored thanks to a differential probe. By setting an infinite persistence in the oscilloscope's display, the eye pattern of the PRBS sequence is visible.

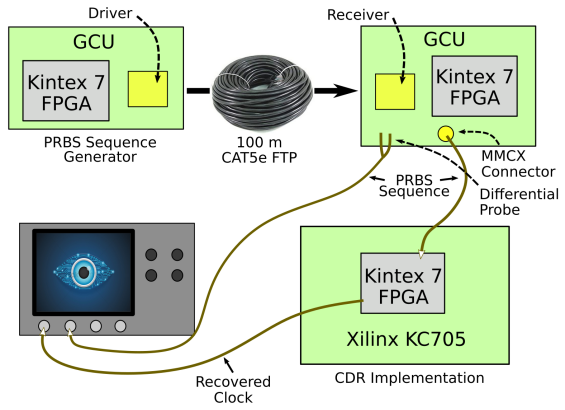

Fig. 10. Block diagram for the CDR testing setup.

B. Test Results

As previously stated, the design is evaluated by

- Analyzing the eye pattern of the PRBS digital signal after 100 meters of CAT5e cable
- Monitoring the PRBS error counter in order to derive the Bit Error Ratio (BER)

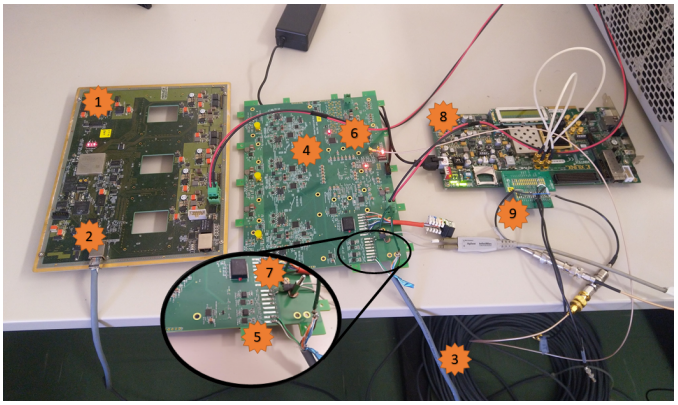

Fig. 11. Picture of the CDR testing setup. · 1: GCU version 1.0 · 2: RJ45 connector · 3: 100m CAT5e FTP cable · 4: GCU version 2.2 · 5: RJ45 soldering pads · 6: MMCX connector · 7: Differential probe · 8: Xilinx KC705 evaluation board · 9: FMC interconnection board.

1) Eye Pattern: Fig. 12 displays the measured eye diagram of the PRBS data after the cable transmission. The pattern is obtained by capturing in the oscilloscope the data after the receiver chip, triggering with the CDR recovered clock. The infinite persistence display option is used to create the characteristic eye figure. The vertical width of the eye has been

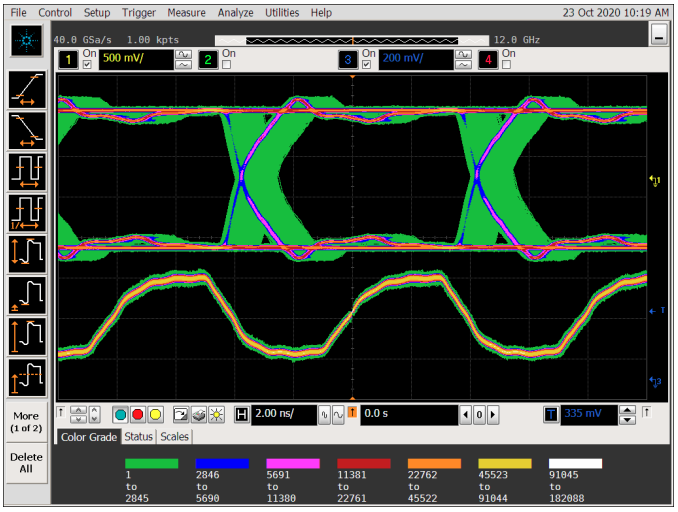

Fig. 12. Measured eye diagram of the PRBS data at 125 Mbps after 100 m of CAT5e cable (top) together with the CDR recovered clock (bottom).

minimally impacted by the cable trip, thanks to the cable driver and receiver which are able to transmit and reconstruct the signal efficiently. The main issue that could potentially impact the data transmission quality is jitter, which is a combination of data and recovered clock jitter. This affects the horizontal opening of the eye, reducing its width; in our case the overall (PRBS data jitter after 100 m of CAT5e cable + recovered clock jitter) eye width is about 6.4 ns (500.000 recorded eye), guaranteeing a good tolerance for correct data sampling. Other measurements provided by the oscilloscope are:

- Eye jitter: $\sim$ 217 ps RMS.
- Q-Factor: $\sim$ 14.

2) BER Test: Sending a long sequence of bits through the system is needed to calculate the BER, which is equal to the number of incorrectly received bits divided by the total number of bits sent. By fixing the BER value at 10^{-12} (standard for data transmission over fiber optics and Ethernet channels) and the Confidence Level (CL) threshold at 95%, the number of bits required to be sent without any error is about $3{*}10^{12}$ [18]. With a data rate of 125 Mbps, this equals to 400 minutes of error-free data transmission.

To perform the test, a PRBS checker alongside an error counter, has been implemented in the CDR firmware. The monitoring of the PRBS errors is performed through the customizable Integrated Logic Analyzer (ILA) IP core by Xilinx. Both the PRBS generator board and the CDR implementation board has been kept at constant temperature (ambient temperature) throughout the whole test.

The test ran for 400 minutes with zero errors counted and without any CDR loss-of-lock, allowing to reject the possibility of a BER higher than 10^{-12} with a CL of 95%.

3) CDR Clock Immunity to EMI: To test the improvement of the recovered clock stability to sudden EMI (and environmental noise in general), a qualitative test has been set up using a neon tube lamp placed underneath the CAT5e coil, exploiting its "property" to emit EMI when turned on.

The first test has been performed with a JUNO-like architecture, where the first GCU board generates a 62.5 MHz clock, sending it through the CAT5e cable to the second GCU board, where it is recovered by the FPGA's PLL. The 'locked' flag of the PLL is continuously monitored though the Xilinx ILA IP Core. In this condition, when the neon tubes are turned on, the PLL loses the locking state for several clock cycles. In an actual JUNO-experiment scenario, this would cause the board to lose the timing synchronization with the BEC, therefore losing data from three PMTs until a new synchronization procedure would run.

The same setup has been used to test the stability of the CDR recovered clock, sending PRBS-7 data though the cable and connecting the Xilinx KC705 board to the second JUNO board. The Xilinx ILA IP Core has been used to check both the CDR locking condition as well as the PRBS error counter. The test shows that, even though the neon tubes induce errors on the recovered PRBS data stream, the CDR does not lose the lock and the recovered clock remains stable.

The proposed CDR is also insensitive to transition-free data pattern, not losing the lock condition keeping the recovered clock at a stable frequency.

C. Out of Specification Results

The main limitation on the CDR performances is given by the OSERDESE2 maximum data rate, which is 1250 Mbps for our target FPGA [19]. This limitation has an impact on the maximum recovered clock frequency and phase resolution, as the OSERDESE2 is a key component in the SerDes technique (see Sec. II-A1). Additional hardware constraints are given by the MMCM maximum VCO and input frequencies, 1866.00 and 933.00 MHz respectively in our case [19], but these are uninfluential compared to the SerDes restriction. If we try and push the SerDes data rate beyond the specifications, the CDR performances can improve drastically. For demonstration, a test has been performed using a system clock (f_C

in eq. 3) of 250 MHz, doubling the 125 MHz used for the main test. By using 8 phase wheels (PW in eq. 2) the OSERDESE2 works with a 1 GHz clock DDR. By setting the *g_multiplication_factor* to 3 (eq. 3) the output clock frequency of the CDR (f_{OUT}) is equal to 250 MHz. With this setup the CDR is able to decode data streams of 250 Mbps.

In fig. 13 the eye pattern of the 250 Mbps data sampled by the CDR recovered clock is presented. The physical transmission medium of the data has not changed, and the stream traveled for 100 meters in a CAT5e cable before being used as input by the CDR core. The eye width is approximately 3 ns. A BER test has also been performed and even in these conditions, the possibility of a BER higher than 10^{-12} is rejected with a CL of 95%.

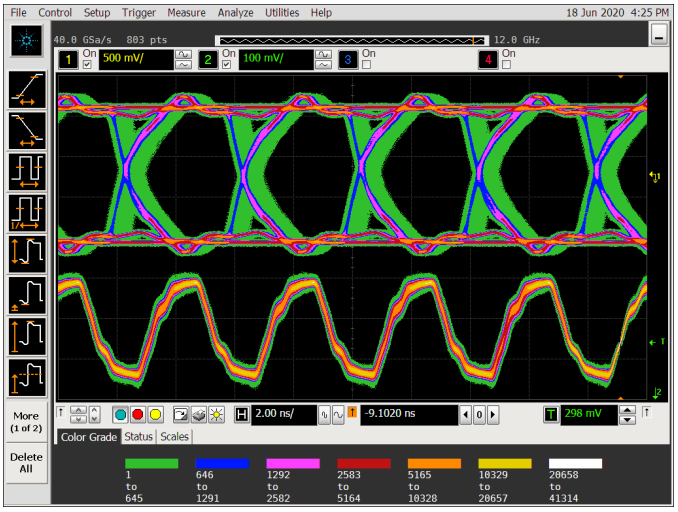

Fig. 13. Measured eye diagram of the PRBS data at 250 Mbps after 100 m of CAT5e cable (top) together with the CDR recovered clock (bottom).

IV. Conclusion

A cost and power efficient FPGA implementation of a digital CDR was presented in this paper which successfully sustains data transmission rates of 125 Mbps over 100 meters of CAT5e cable (going up to 250 Mbps of stable clock and data recovery if components are used out of specifications). The design has been implemented in a Xilinx Kintex-7 FPGA, but it is easily portable to other FPGAs equipped with a SerDes and a clock manager tile. The main limitations of this approach are:

- The use of a SerDes running at a frequency up to 8 times higher than the data rate. This could limit the choice of FPGAs suitable for this design.
- the mandatory loop-back of the NCO's output clock, which is to be foreseen when designing the PCB.
- The clock recovered after 100 meters of CAT5e cable, presents a non-negligible jitter, especially if you need it to clock an Analog to Digital Converter (ADC). An external jitter cleaner chip is still recommended and it is foreseen to be used in the JUNO's GCU.
- The CDR takes several seconds to lock to the data rate.

If implemented in the GCU's FPGA configuration for the JUNO's experiment, almost all the limitations described above are tolerated. However, a small modification on the PCB would be mandatory, to comply with the required clock loop-back.

By using the proposed CDR core, the synchronous link would gain

- Robustness, in terms of avoiding loss of clock locking in the presence of short electromagnetic interference.
- Features, by freeing a twisted pair in the cable, previously dedicated to clock forwarding.

Furthermore, the proposed CDR implementation could find applications in several High Energy and Nuclear Physics experiments in which radiation exposure is a setup constraint, avoiding the acquisition of transceivers and radiation hard/tolerant CDR chips, thanks to

- the fact that the whole core is contained within the FPGA and does not rely on external components, except for a reference clock oscillator whose frequency stays within the accepted range of the FPGA's PLL/MMCM
- the portability of the presented core to other FPGA's family, including flash-based radiation tolerant models [20].
- the use of copper cables as transmission medium over optical fibers, which suffers from the Radiation Induced Attenuation (RIA) [21] effect.

Finally, if in a possible application the use of an external oscillator on board is forbidden, the core could be further developed by generating a delay-line clock signal directly within the FPGA, like the one in [22].

References

[1] Xilinx Application Note, *XAPP224 (v2.5)*. July 11. 2005. https://www.xilinx.com/support/documentation/application_notes/xapp224.pdf
[2] M. Kubek, Z. Kolka. *Blind Oversampling Data Recovery with Low Hardware Complexity*, Radioengineering, vol. 1, pp. 74-78, 2010.
[3] Li Pang, Houde Liu, Baohua Li and Bin Liang. *A Data Recovery Method for High Speed Serial Communication based on FPGA*, IEEE, pp. 664-667, 2010
[4] D. Calvet. *Clock-Centric Serial Links for the Synchronization of Distributed Readout Systems*, in IEEE Trans. Nucl. Sci., vol. 67, no. 8, pp. 1912-1919, Aug. 2020, doi: 10.1109/TNS.2020.3006698.
[5] F.P. An *et al.*. *Neutrino Physics with JUNO*. J. Phys. G 43 (2016) 030401. arXiv:1507.05613
[6] T. Adam *et al.*. *JUNO Conceptual Design Report*. arXiv:1508.07166
[7] M. Bellato *et al.*. *Embedded Readout Electronics R&D for the Large PMTs in the JUNO Experiment*. NIMA Volume 985, 1 January 2021, Article number 164600 arXiv:2003.08339
[8] D. Pedretti *et al.*. *Nanoseconds Timing System Based on IEEE 1588 FPGA Implementation*. IEEE Trans. Nucl. Sci. 66 2019 1151.
[9] Zhang, Qingmin and Guo, Yuhang. *The JUNO Calibration System*, PoS ICHEP2018 (2019), 811, https://doi.org/10.22323/1.340.0811
[10] Xilinx Kintex-7 FPGA KC705 Evaluation Kit. https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html
[11] Xilinx. *7 Series FPGAs Clocking Resources User Guide (UG472)* http://www.xilinx.com/support/documentation/user_guides/ug472_7Series_Clocking.pdf
[12] Clifford E. Cummings. *Clock Domain Crossing (CDC) Design & Verification Techniques Using SystemVerilog*. SNUG Boston 2008. Rev. 1.0
[13] Eva Murphy and Colm Slattery. *Ask The Application Engineer33: All About Direct Digital Synthesis*. Analog Dialogue vol.38. 2004. https://www.analog.com/en/analog-dialogue/articles/all-about-direct-digital-synthesis.html
[14] Xilinx. *7 Series FPGAs SelectIO Resources User Guide (UG471)* https://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf
[15] Clock and Data Recovery. https://en.wikibooks.org/wiki/Clock_and_Data_Recovery

[16] J.D.H. Alexander. *Clock Recovery from Random Binary Signals*. vol. 11. pp. 541 - 542. October 1975 in Electronics Letters. Institution of Electrical Engineers.

[17] Xilinx Application Note, *XAPP884 (v1.0)*. January 10. 2011. https://www.xilinx.com/support/documentation/application_notes/xapp884_PRBS_GeneratorChecker.pdf

[18] Dragan Miti, Aleksandar Lebl and Zarko Markov. *Calculating the required number of bits in the function of confidence level and error probability estimation*. Serbian Journal of Electrical Engineering. 9. 361-375. 2012. 10.2298/SJEE1203361M.

[19] Xilinx Kintex-7 FPGAs Data Sheet:DC and AC Switching Characteristics, *DS182 (v2.18)*. June 28. 2019. https://www.xilinx.com/support/documentation/data_sheets/ds182_Kintex_7_Data_Sheet.pdf

[20] A. Menicucci *et al. Flash-based FPGAs in space, design guidelines and trade-off for critical applications*. Proceedings of the European Conference on Radiation and its Effects on Components and Systems, RADECS. 10.1109/RADECS.2013.6937404, 2013.

[21] Sylvain Girard *et al.*. *Overview of radiation induced point defects in silica-based optical fibers*, Reviews in Physics, Volume 4, 2019, 100032, ISSN 2405-4283, https://doi.org/10.1016/j.revip.2019.100032.

[22] R. Giordano *et al.*, *High-Resolution Synthesizable Digitally-Controlled Delay Lines*, IEEE Trans. Nucl. Sci., vol. 62, no. 6, pp. 3163-3171, Dec. 2015, doi: 10.1109/TNS.2015.2497539.