High-Speed, Fixed-Latency Serial Links With FPGAs for Synchronous Transfers

Alberto Aloisio, Francesco Cevenini, Raffaele Giordano, and Vincenzo Izzo

***Abstract*—Fixed-latency serial links find application in trigger and data acquisition systems of High Energy Physics (HEP) experiments requiring a predictable data transfer timing. In some architectures, there is the need to clock the data in and out from the link synchronously with a system clock (i.e., synchronous transfers) instead of using the clock recovered from the serial stream.**

In this work, we present a synchronous link architecture based on high-speed transceivers embedded in latest generation Field Programmable Gate Arrays (FPGAs). These transceivers are typically designed for applications that tolerate latency variations. However, we have developed two configurations and a clocking scheme to implement fixed-latency operation. The latency is constant during the transfer, after a loss of lock or a power cycle. Once locked, the link can be considered as a synchronous pipeline. The configurations do not depend on a particular serial encoding, the encoder/decoder being external to the transceiver. We discuss the latency performance for each configuration and show an implementation of the architecture we propose. We also present experimental results showing the stability of the latency of the link.

***Index Terms*—Data acquisition, FPGAs, latency, serial links, synchronous transfers.**

I. Introduction

HIGH-SPEED transceivers embedded in FPGAs are typically designed for applications that do not require a deterministic latency with an accuracy at the level of nanosecond. Nevertheless, within the HEP community, some attempts have been made in order to take advantage of these devices in the design of fixed-latency links, due to the obvious benefits in terms of system integration, power dissipation and cost. We now briefly summarize and comment some results obtained in the field of FPGA-based serial links for application to HEP.

A study [1] have been conducted about the possibility to implement serial links of Large Hadron Collider (LHC) experiments by means of FPGA-embedded serial transceivers. The study focused on Multi-Gigabit Transceivers (MGTs) [2] embedded in Xilinx Virtex-II Pro devices and led to the implementation of a variable latency link compatible with a radiation-hard serializer [3] developed at CERN. In the framework of the A Large Ion Collider Experiment (ALICE), a Conditional Inversion Master Transition (CIMT) [4] compliant deserializer [5] based on Xilinx and Altera embedded SerDes has been designed. The Altera version has been experimentally tested, while the Xilinx one has only been simulated. Both solutions have been discarded because the latency of link was too high for the application. Moreover, the latency of the deserializer resulted to be variable. Another CIMT receiver has been implement by means of latest Xilinx transceivers for a portable test system [6] of a sub-detector of the A Toroidal LHC AapparatuS (ATLAS) experiment. No details about the latency of the implementation are provided and it is unclear whether or not the transfer is synchronous. The Compact Muon Solenoid (CMS) experiment's Global Calorimeter Trigger (GCT) includes some serial links [7] based on Xilinx MGTs. The links perform fixed-latency data transfers, unfortunately the authors do not provide details of the architecture and do not specify if the latency is always the same at each power-up of the link. The Level-0 muon trigger of the Large Hadron Collider beauty (LHCb) experiment [8] strongly relies on multi-Gigabit 8b10b serial links implemented with Altera embedded SerDes. Unfortunately, no details about the latency of the links are available. The Gigabit Transceiver project [9], currently under development at CERN, aims at providing a radiation-tolerant SerDes for trigger and data acquisition applications to super Large Hadron Collider (sLHC) experiments. The project foresees the deployment of FPGA-embedded SerDes on the off-detector part of the link. Trigger transfers will require a fixed-latency configuration of the SerDes.

All the cited papers mainly focus on the application of the developed links to the pertaining experiment but do not provide details on the system architectures. In order to bridge this gap, we present here a fixed-latency architecture based on a configurable SerDes embedded in a latest-generation FPGA device. We studied its architecture in order to find a configuration and a clocking scheme to achieve a deterministic transmission delay in synchronous transfers. As a case study for the discussion of our architecture, we choose the implementation of a serial link to be deployed in the ATLAS experiment.

In the barrel section of the ATLAS muon spectrometer [10], Level 1 (L1) trigger data is transferred across optical fibers from the on-detector electronics to the readout system, which is hosted in a counting room, at about a hundred meters from the detector. The transmitter electronics consists of custom boards, while the receiving ones lie in VME crates. The whole data path is implemented as a synchronous pipeline clocked by the Timing, Trigger and Control (TTC) system [11]. It distributes timing information such as the bunch crossing clock

Manuscript received December 22, 2008; revised April 09, 2009. Current version published October 07, 2009. This work was supported in part as a Programma di Ricerca di rilevante Interesse Nazionale (PRIN) by the Italian Ministero dell'Istruzione, Università e Ricerca Scientifica.

The authors are with the INFN Sezione di Napoli and Università degli Studi di Napoli "Federico II", Dipartimento di Scienze Fisiche, 80126 Napoli, Italy (e-mail: aloisio@na.infn.it; cevenini@na.infn.it; rgiordano@na.infn.it; izzo@na.infn.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2009.2027236

0018-9499/$26.00 © 2009 IEEE

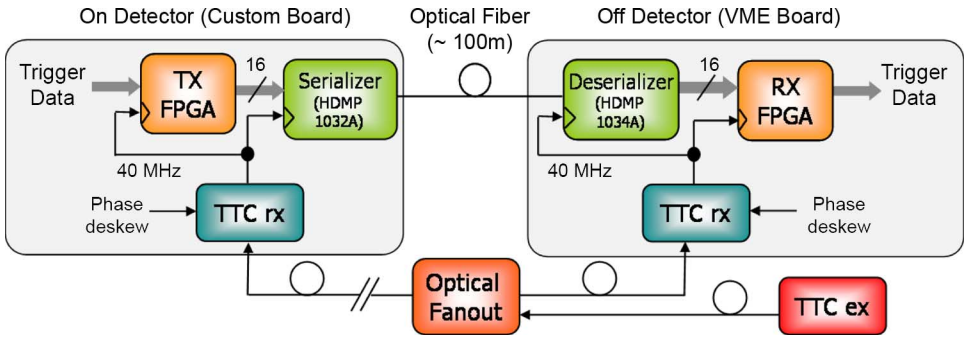

Fig. 1. Simplified block diagram of the serial link setup in L1 ATLAS barrel muon trigger.

(at about 40 MHz), the L1 trigger, the bunch crossing counter, the event counter and their reset signals. A TTC transmitter (TTCex) sends that information to TTC receivers (TTCrx), one per destination, by means of a tree of optical fibers. Each receiver extracts the TTC signals from the received data and, in particular, provides the bunch crossing clock, whose phase can be finely tuned with a de-skew control (Fig. 1).

On detector, an FPGA processes trigger data and transfers it to a high-speed serializer, which encodes the data and transmits it over an optical fiber. The transmitter FPGA and the serializer share the bunch crossing clock from the same TTCrx module. On the far end of the fiber, a deserializer restores the parallel form of the data and transfers it toward an FPGA for further processing. On this side of the link as well, the FPGA and the deserializer share the bunch crossing clock from a TTCrx module. In general, there is a non-zero, constant phase difference between the transmitter and the receiver clocks, but they are both copies of the LHC bunch crossing clock, so they have the same frequency.

The latency through both FPGAs is constant, therefore in order to be able to predict the overall trigger processing time, the serializer and the de-serializer must have a fixed latency as well. In the ATLAS experiment the G-Link chip-set [12] (Agilent HDMP-1032A/1034A [13]) has been deployed. The chips transfer data with a fixed and deterministic latency. In order to recover the clock from the serial data stream and to perform word alignment and error detection on the receiver, data is encoded according to CIMT protocol. Despite G-Link's unique timing features, its production discontinued and users are now looking for compatible replacements. The transmission side of the link is on-detector and it would be very complicated to substitute, however it is still possible to upgrade the receiver board. Thus, we implemented a fixed-latency, synchronous CIMT serializer/deserializer by means of FPGAs.

II. Fixed Latency in Synchronous Systems

In a synchronous system, the latency through a path, from a node A to a node B, is the time interval needed for data to go from A and arrive (possibly processed) on B following that path. Let us consider the system represented in Fig. 2 and let us assume that the serial link transfers data with a fixed latency (i.e., it could be considered just as a constant delay between the source and the destination). We purposely insert a tunable delay

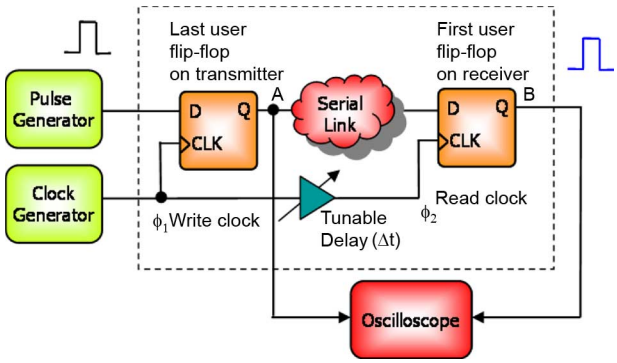

Fig. 2. Synchronous serial link with skew on clock distribution.

(Δt) between the clocks of the last user flip-flop before entering the link and the first user flip-flop on the output of the link. Let $T = 25$ ns be the clock period. We take the clock on the first flip-flop as our reference phase $(\phi_1 = 0)$, while on the second one it has a phase

$$\phi_2 = 2\pi \frac{\Delta t \bmod T}{T}. \tag{1}$$

At the beginning, we set Δt (and thus ϕ_2) to zero and the latency from A to B is L. We transmit a pulse with the first flip-flop and monitor the latency from A to B with an oscilloscope for five different values of Δt (Fig. 3). Each time that we increase Δt by a certain amount, the latency grows of the same amount. However, if we theoretically would choose a $\Delta t = T - \epsilon$ (with ϵ small as one wishes) and then increase it to T, ϕ_2 would go back to its initial value and so would do the latency, decreasing discontinuously from $L + T - \epsilon$ to L. We are neglecting setup and hold constraints since they are not essential to the discussion.

If we experimentally move Δt within a range around T, we find an interval for Δt such that the latency of the system can be either L or $L + T$. The amplitude of this range depends on the implementation of the system. It is very instructive to note that a synchronous pipeline can vary its latency due to skew on the clock distribution [14]. This effect can be avoided only if it is possible to control the clock skew. Unfortunately, this is not the case for a synchronous serial link when data is transmitted on a clock phase and received on an arbitrary different one (e.g., ATLAS level-one muon trigger serial links).

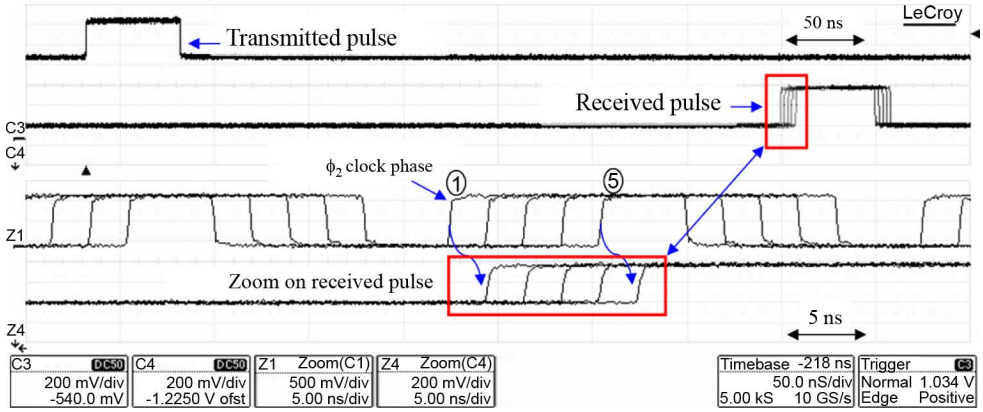

Fig. 3. Latency between transmitted and received pulse for different clock skews.

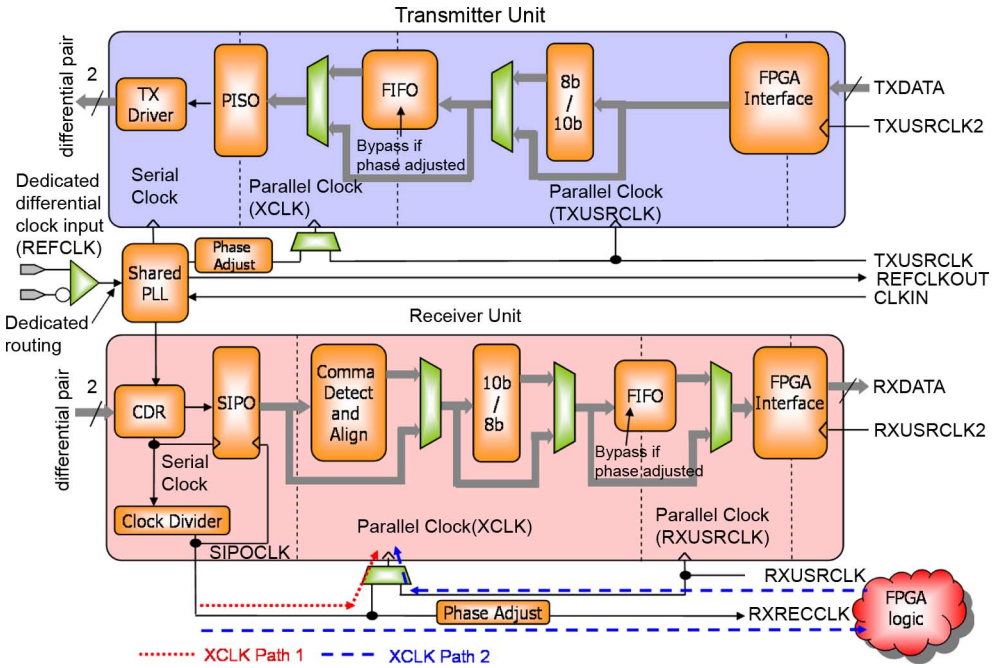

Fig. 4. Simplified block diagram of Xilinx GTP transceivers. Half a tile is shown.

III. The GTP Transceiver

In this work we present a fixed-latency synchronous architecture based on the so called GTP transceiver [15] of the Xilinx Virtex 5 FPGA family. Inside the FPGA, GTPs are available as configurable hard-macros (or "tiles"). Each tile includes a pair of transceivers, which share some basic components, like a Phase Locked Loop (PLL) and the reset logic. Fig. 4 shows the architecture of the transmitter (Tx) and the receiver (Rx) included in each transceiver. We will now concisely present the features of the GTP essential to our work. More details can be found in the user guide.

The Tx consists of a serial and a parallel section, the first one works in a high-speed clock domain while the second one includes three clock domains. Two of these are input clocks (TXUSRCLK and TXUSRCLK2), while the third (XCLK) is internal. The PLL requires a reference clock (CLKIN), whose frequency is a sub-multiple of the bit-rate (or "line-rate").

The GTP supports 8,10,16 and 20-bit wide input words, but the following discussion focuses on the operation with 20-bit symbols. The FPGA Interface logic reads data from the fabric on the TXUSRCLK2 clock edges and outputs it synchronously with the TXUSRCLK clock. The interface splits input data in 10-bit words, hence the TXUSRCLK frequency must be twice of the TXUSRCLK2 frequency. The output from the FPGA interface is 8b/10b encoded, if needed, and it is transferred to a First In First Out (FIFO) buffer. The latter allows safe data transfers between the TXUSRCLK domain and the XCLK domain. In some configurations XCLK and TXUSRCLK have the same frequency and a constant phase offset. The FIFO can be bypassed if the offset is sufficiently small and in fact the device offers a phase alignment circuit in order to minimize it. In the XCLK domain, data is serialized by the Parallel In to Serial Out (PISO) block, whose output is synchronous with the high-speed serial clock.

On the GTP receiver unit, the serial stream from dedicated FPGA pins is received by the Clock and Data Recovery (CDR)

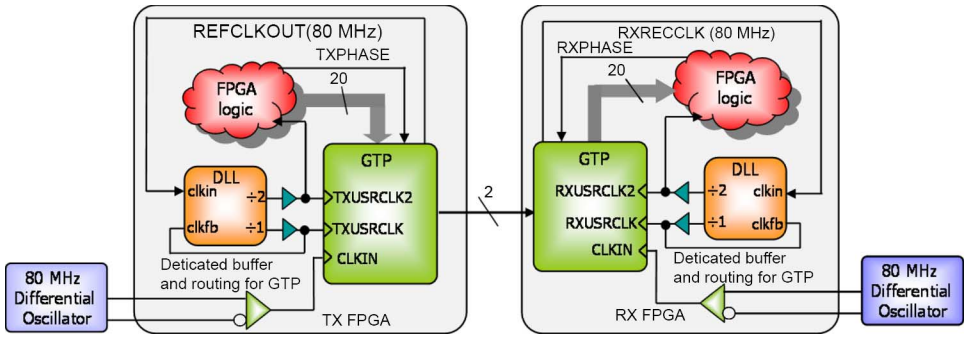

Fig. 5. Recommended GTP clocking scheme when using phase alignment circuits.

circuit, which extracts a clock and uses it to sample the data. The extracted clock is divided to generate a recovered clock for the Serial In to Parallel Output (SIPO), SIPOCLK and one for the parallel section (RXRECCLK), which can be used to clock the receiving FPGA logic. The SIPO block de-serializes data into 10-bit words synchronously with SIPOCLK, which can have a phase difference with respect to RXRECCLK. The next block in the data-path works synchronously with XCLK, which can be driven by SIPOCLK (XCLK path 1 in Fig. 4) or RXUSRCLK. In the latter case, there is a clock domain crossing from SIPOCLK to RXUSRCLK and it must be resolved. If RXUSRCLK is derived from RXRECCLK (XCLK path 2 in Fig. 4), there is in general a phase offset due to the path in FPGA fabric clocking elements (buffers, routing, PLLs, etc.). In this case, it is possible to minimize the offset between SIPOCLK and RXUSRCLK, by means of a dedicated phase alignment circuit. The Comma Detect and Align block following the SIPO can be programmed to search for and align to a comma. If alignment is performed outside of the GTP, this block can be skipped. Data is 10b to 8b decoded, if needed, and transferred to a FIFO in order to enter the RXUSRCLK domain (when XCLK is driven by SIPOCLK, otherwise data is already in the RXUSRCLK domain). The FPGA interface combines 10-bit words from the RXUSRCLK domain and outputs 20-bit words in the RXUSRCLK2 domain at half of the frequency.

IV. Fixed-Latency GTP Configurations

Unfortunately, standard GTP configurations do not have fixed latency. Moreover, we would like to keep the latency of the link as low as possible. We begin by discussing the configuration suggested by the vendor and analyzing its weaknesses. Afterward, we introduce our solution.

Let us suppose we need to implement an 800 Mb/s transmission of 20-bit frames. The minimum clock frequency required by the GTP for the reference clock is 60 MHz, so we choose 80-MHz clock sources. Since we want to achieve a latency as low as possible we bypass the GTP Tx and Rx FIFO and use their phase align circuits instead. The FPGA logic on both the link sides is in charge of activating the phase alignment circuits, by means of the TXPHASE and RXPHASE signals, and checking if the alignment has been achieved. Under these conditions, the clocking scheme suggested by the vendor is the following (Fig. 5).

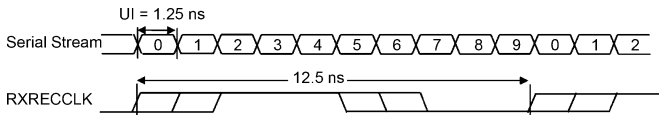

Fig. 6. Timing diagram of the generation of the recovered clock.

On both the Tx and Rx, the CLKIN input of the GTP receives an 80 MHz clock by means of the dedicated GTP differential clock buffer and routing.

On the Tx side, a Delay Locked Loop (DLL) divides the REFCLKOUT (at 80 MHz) from the GTP to generate a 40 MHz clock for the TXUSRCLK2 input and the FPGA logic. The undivided output of the DLL ("÷1") is used to feed the TXUSRCLK input of the GTP. According to the User Guide, TXUSRCLK and TXUSRCLK2 must be positive edge aligned with as low skew as possible. The clock buffer delay on the DLL "÷1" output is compensated since it is inserted in the feedback loop. The other clock buffer can be considered compensated as long as buffer delays can be considered identical. Under this approximation, TXUSRCLK and TXUSRCLK2 are positive-edge aligned with no skew.

On the Rx side, a DLL divides the clock recovered from the serial stream (RXRECCLK at 80 MHz) to generate the RXUSRCLK2 clock (at 40 MHz) for the GTP and the FPGA logic. The undivided output from the DLL is used to produce RXUSRCLK with no skew with respect to RXUSRCLK2, as required by the GTP specifications.

The described clocking strategy is iso-synchronous, because the GTPs generate the clocks for the transmitting and receiving FPGA logic. On the Tx side, data enters the GTP synchronously with a TXUSRCLK2 generated from the REFCLKOUT clock, which may have a skew with respect to the board clock driving the GTP PLL (CLKIN). On the Rx, data exits from the GTP synchronously with the RXUSRCLK2 clock generated from the recovered clock (RXRECCLK), whose phase and frequency depend on the incoming serial stream and not on the board clock. This clocking strategy does not allow us to clock data in and out from the link synchronously with the board clocks.

Let us now focus on the latency of the transmission. Edges in the serial stream can be divided in 10 classes, each labeled by a bit position in a 10-bit symbol (Fig. 6) (#0, #1…, #9). At each power-up, the GTP can randomly produce a recovered clock aligned with edges belonging to one of the 10 classes.

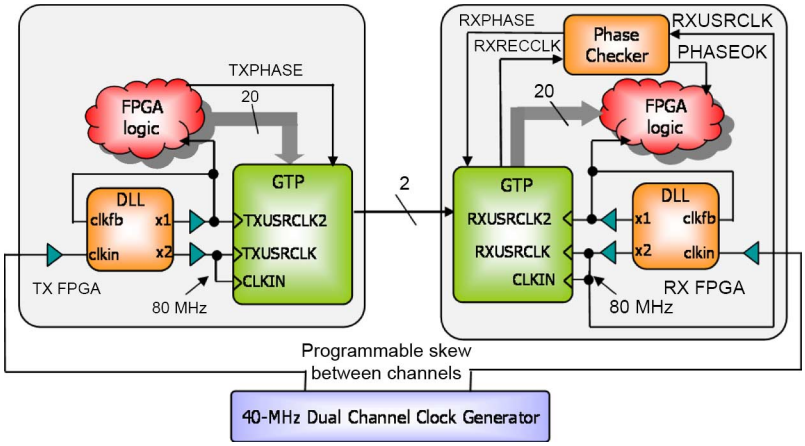

Fig. 7. Adopted clocking scheme for the GTP transmitter and receiver for synchronous operation.

Thus, we have a latency variable up to 10 Unit Intervals (UI). Even if we assume that the 80-MHz recovered clock is generated always with the same phase offset with respect to the serial stream, the division of the clock to 40-MHz is uncorrelated with respect to the stream. In fact, the recovered clock rising edges are synchronous with the stream and can be divided in two classes: the ones synchronous with Most Significant Byte (MSB) of the 20-bit symbol and the ones synchronous with the Least Significant Byte (LSB). At each power-up of the link, the external DLL could lock on the MSB edge or the LSB edge of the 80-MHz clock and this is unpredictable. This uncertainty of the RXUSRCLK2 phase with respect to serial stream makes the latency variable of one recovered clock period (T_{rec}). A designer may think to configure the GTP to work with a 10-bit word width clocked at 80 MHz and then perform the division by 2 of the clock frequency in such a way to distinguish between LSB and MSB cycles, for instance by means of a 1-bit counter which is reset always on the LSB edge of RXUSRCLK. Unfortunately, this would misalign the RXUSRCLK clock edge with respect to its divided version and additional logic would be needed to work this problem around. The presented effects produce a variation of the latency between two subsequent resets of $n\mathrm{UI} + mT_{rec}$, where n is an integer number in the range from 0 to 9 and m can be either 0 or 1.

As an example to discuss our architecture, we have selected the development of a fixed-latency link compatible with the ATLAS L1 Muon trigger links. We have a common 40-MHz clock distributed at the transmitter board and to the receiver board with some skew. The previously described architecture does not satisfy our requirements: it is not synchronous with the board clocks and it transfers data with a variable latency. In order to overcome these limitations, we adopted a different clocking scheme. On the Tx, the 80 MHz CLKIN clock for the GTP is obtained by multiplying the 40 MHz board clock with a DLL. Since the clock signal is not differential and requires a frequency multiplication before entering the GTP, we can not use the dedicated routing and differential buffer. In this original clocking scheme, we use the same DLL output ("x2") to provide the TXUSRCLK clock to the GTP (Fig. 7), while we use the unmultiplied output ("x1") of the DLL to clock the FPGA logic and the TXUSRCLK2 GTP input. The buffer on "x1" output is phase compensated, since it is in the feed-back loop of the DLL. Hence, the skew between the board clock and the FPGA logic clock is due to the input buffer delay on the DLL clkin pin and the pertaining routing from the FPGA input pad ($\sim$ 1 ns). On the Rx, we deploy the same clocking architecture, where TXUSRCLK and TXUSRCLK2 are substituted respectively by RXUSRCLK and RXUSRCLK2. We do not use the clock recovered from the serial stream (RXRECCLK). In a full-duplex communication, this clocking scheme has also the advantage of requiring just one clock multiplication per transceiver. In fact the "2x" output from the DLL can be used to feed both the TXUSRCLK and RXUSRCLK inputs of the GTP, while the "1x" can be used for the TXUSRCLK2 and RXUSRCLK2 inputs.

In our tests, we clock the receiver and the transmitter with a pulse generator providing two 40-MHz clocks with a tunable delay. The delay can go from 0 ns to 600 ns with steps of 200 fs; we used this capability to set a skew between the receiver and transmitter clocks. The generator has an rms period jitter smaller than 3 ps.

If we now consider the overall link, we can note that data is transmitted synchronously with Tx board clock and received synchronously with the Rx board clock. We have experimentally found out that when the reference clocks of the transmitter and the receiver are generated by the same source, the recovered clock is always generated with a fixed phase relationship with respect to the serial stream. In our setup, it means that the recovered clock has always the same phase difference with respect to RXUSRCLK, even after a power-cycle. This is a necessary condition to achieve fixed latency.

Our clocking strategy raises a problem. In this configuration, XCLK is driven by SIPOCLK and we should use the phase align circuit to resolve phase differences between RXUSRCLK and SIPOCLK. The circuit is able to generate RXRECCLK by selecting one among 20 equally spaced phases, spanning one clock period. When RXUSRCLK is derived from RXRECCLK, the circuit minimizes the skew between SIPOCLK and RXUSRCLK domains, by choosing a suitable phase for RXRECCLK. However, in our clocking scheme RXUSRCLK is not derived

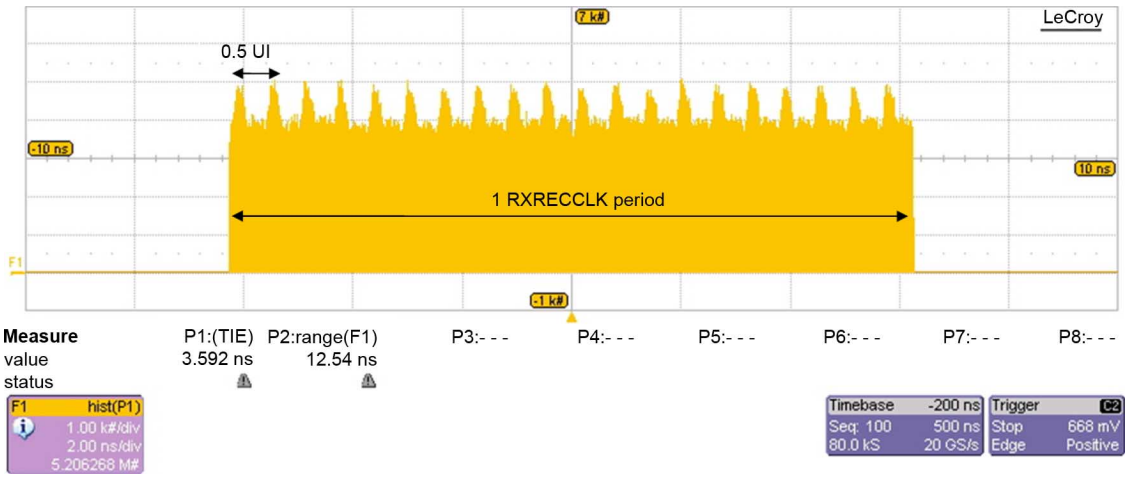

Fig. 8. Histogram showing the relative phase between RXRECCLK and the reference clock when $\Delta\phi$ does not satisfy the requirements of the phase align circuit.

from RXRECCLK, thus a phase shift on RXRECCLK has no effect on RXUSRCLK. Let $\Delta\phi$ be the phase difference between SIPOCLK and RXUSRCLK. If the phase align circuit is activated two situations can happen:

1) $\Delta\phi$ satisfies the requirements of the circuit and no phase shift on RXRECCLK is performed;
2) $\Delta\phi$ does not satisfy the requirements of the circuit and the phase of RXRECCLK is changed, but, as this do not affect the RXUSRCLK phase, the circuit keeps on changing the phase forever.

The histogram of the relative phase between RXRECCLK and the reference clock (Fig. 8) has been acquired in the second situation, and shows all the possible phase offsets.

Thus, a particular RXUSRCLK phase is suitable to read safely data from the SIPO, if the phase align circuit does not continuously shift RXRECCLK. We designed a very simple logic (the "Phase Checker" block in Fig. 7) which activates the phase alignment circuitry and checks whether or not the recovered clock is shifting with respect to RXUSRCLK. If RXRECCLK is not shifting, this means the RXUSRCLK phase satisfies the requirements of the phase align circuitry and the logic asserts the PHASEOK signal. The phase offset between the transmitter board clock (which is synchronous with RXRECCLK) and the receiver board clock (which is synchronous with RXUSRCLK) must be set, either inside the FPGA or externally, to a suitable value by checking the level of the PHASEOK signal. In synchronous applications where the relative clock phase can be tuned, this is not an issue. For instance in the ATLAS experiment, in order to find a suitable phase difference, the designer may include a dedicated logic in charge of checking the PHASEOK signal and, if needed, changing the clock phase by means of the dedicated TTCrx de-skew feature, which is accessible via a I2C interface.

The described configuration of the GTP, which we will further refer to as "Configuration One", has the minimum achievable latency with the GTP. Unfortunately, Configuration One can be used only to transfer 10-bit or 20-bit frames and not 8-bit or 16-bit frames, which instead are supported by the GTP when using the internal FIFO.

For applications where a phase tuning is not possible, we developed another configuration, which we will refer to as "Configuration Two". We kept the same configuration for the GTPs with the only difference that we enabled the FIFO on the receiver. The FIFO allows to compensate any phase difference between the transmission and reception clocks, and the use of the phase alignment circuit is not necessary. Data is written in the FIFO synchronously with SIPOCLK and read with RXUSRCLK, which are phase misaligned but synchronous. The GTP guarantees that, after a reset, the FIFO is always filled with the same number of words before starting to output them. This ensures a deterministic latency through the FIFO. We have set this value to the minimum (3 words), in order to have the lowest possible latency. Configuration Two solves the phase offset issues encountered with Configuration One and supports all the frame widths offered by the GTP. Unfortunately, the deployment of the FIFO increases the latency by 3 RXUSRCLK clock cycles.

The latencies of the transmitter and the receiver in Configuration One, estimated by means of the user guide, are given in Table I. For each component we report the latency in terms of RXUSRCLK periods and its absolute value in nanoseconds. We remark that some blocks (e.g., FIFOs) have an associated latency even if they are not used. The total latency of the transmitter and of the receiver are respectively 4.5 and 9.5 clock periods. In Configuration Two, the latency of the transmitter is still 4.5 clock periods, while latency of the receiver is 12.5 clock periods (due to the activation of the FIFO).

V. Applicability of the Proposed Architecture to Other Embedded Transceivers

Our experience is that the documentation provided by FPGA vendors is not sufficient to predict if fixed-latency operation can be achieved. We tested in the field the architecture we propose for the Xilinx GTP transceiver. Configurations of other transceivers should be experimentally tested as well. We now discuss the applicability of our configurations and clocking strategy to latest FPGA-embedded SerDes produced by Altera and Lattice. We warn the reader that the following discussion is only based on information found in the data-sheets.

TABLE I
LATENCY OF THE INTERNAL BLOCKS OF THE TRANSMITTER AND OF THE RECEIVER

The table reports latencies for Configuration One, when different, latencies for Configuration Two are reported in parenthesis.

	# of clock cycles	Block latency (ns)
Transmitter		
FPGA Interface	1.5	18.75
FIFO (skipped)	1	12.5
Serial Section	2	25
Total GTP Latency	4.5	56.25
Receiver		
Serial Section	1.5	18.75
Comma Detector (skipped)	3	37.5
FIFO (skipped)	2 (5)	25 (62.5)
FPGA Interface	3	37.5
Total GTP Latency	9.5 (12.5)	118.75
Total Link Latency	14 (17)	175 (212.5)

GX transceivers [16] embedded in the Altera Stratix IV FPGA family and flexi Physical Coding Sublayer (flexiPCS) transceivers [17] embedded in LatticeSC/M devices offer a simpler clocking strategy than the GTP. In fact, they only need the reference clock for the internal PLL plus a transmit clock and a receive clock to interface with the FPGA fabric. All the clocks for the internal domains are generated by a dedicated circuitry and are hidden to the user. On the contrary, the GTP requires the reference clock plus two transmit clocks (TXUSRCLK and TXUSRCLK2) and two receive clocks (RXUSRCLK and RXUSRCLK2).

For both the GX and the flexiPCS it is unspecified if the recovered clock has fixed phase relationship with respect to the serial stream. Both devices include FIFOs to resolve phase differences between the internal parallel clock domains and the external transmission and reception clock domains. They do not offer any alternative low-latency configuration, on the contrary of what Xilinx does with the phase alignment circuit. The Altera GX offer a number of options for skipping internal sub-blocks (FIFOs, 8b10b encoder, etc.) comparable to the Xilinx GTP. Configuration Two could in principle be implemented by skipping all the blocks in the data-path but the phase compensation FIFO. However, the latency through the FIFO can vary from 2 to 3 clock cycles in the "low-latency mode" and 4 to 5 clock cycles in the "high-latency mode". It is not specified if, after a reset, the FIFO is always filled with same number of words prior to start transferring them to the fabric. Configuration One should be supported by using the "PMA Direct Drive" mode of the GX, which allows to skip all the blocks in data-path, including the FIFO. With this device, Configuration One is the best candidate, among the two, in order to achieve a fixed latency, but it should be experimentally tested.

As far as it concerns the flexiPCS, the device offers several working modes, each suited to a different protocol. A special mode, named "SerDes Only" allows the use of an external encoder. The phase compensation FIFO can not be skipped and it could possibly prevent a fixed-latency operation. The data-sheet

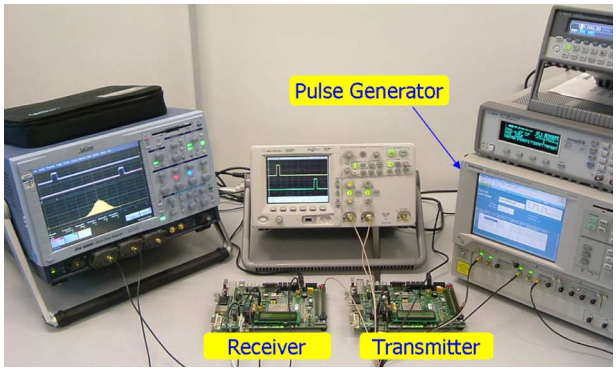

Fig. 9. Experimental setup for latency tests.

provides no information about the latency through the FIFO, but states that the latency through the receiver is 5 parallel clock cycles. In this device only Configuration Two could be implemented and it is not clear whether or not fixed-latency operation is achievable.

VI. G-LINK EMULATION

The configuration we developed are coding-independent, in fact no assumptions have been made on the serial symbols. Any encoding compatible with the CDR may be chosen, the only requirement is that coding, decoding and frame alignment are performed with a constant latency. Since we have chosen the ATLAS L1 Muon trigger links as a case study, we now briefly introduce the CIMT protocol, which is implemented by the G-Link chip-set.

A CIMT stream is a sequence of 20-bit words, each containing 16 data bits (D-Field) and 4 control bits (C-Field). The C-Field flags each word as a data word, a control word or an idle word. Idle words are used in order to synchronize the link at start-up and to keep it phase-locked when no data or control words are transmitted. The DC-balance of the link is ensured by sending inverted or unaltered words in such a way to minimize the bit disparity, defined as the difference between the total number of transmitted 1s and 0s. By reading the C-Field content, the receiver is able to determine if a word is inverted or not.

Let us now consider the link we designed around GTPs. We deployed two Virtex 5 LX50T FPGAs, which include GTP transceivers. We realized two designs, one implementing a link according to Configuration One and the other to Configuration Two. Since we implemented a CIMT transmission, we disabled 8b10b encoding-decoding. On the Tx side, some logic encodes 16-bit words incoming from a payload generator into 20-bit CIMT words and transfers them to the GTP. On the Rx side, some logic receives 20-bit symbols from the GTP and performs the CIMT decoding and the frame alignment. Our encoder and decoder support the simplex and enhanced CIMT modes of the HDMP-1032/34A chip-set, but not the 20/21-bit modes of the older HDMP-1022/24.

The latencies of the transmitter and the receiver, estimated by means of the user guide, are respectively 168.75 ns and 143.75 ns. Details about the contribution of internal blocks are given in Table II. Most of the latency of the transmitter is due to the fabric encoding logic (112.5 ns), while the GTP has a

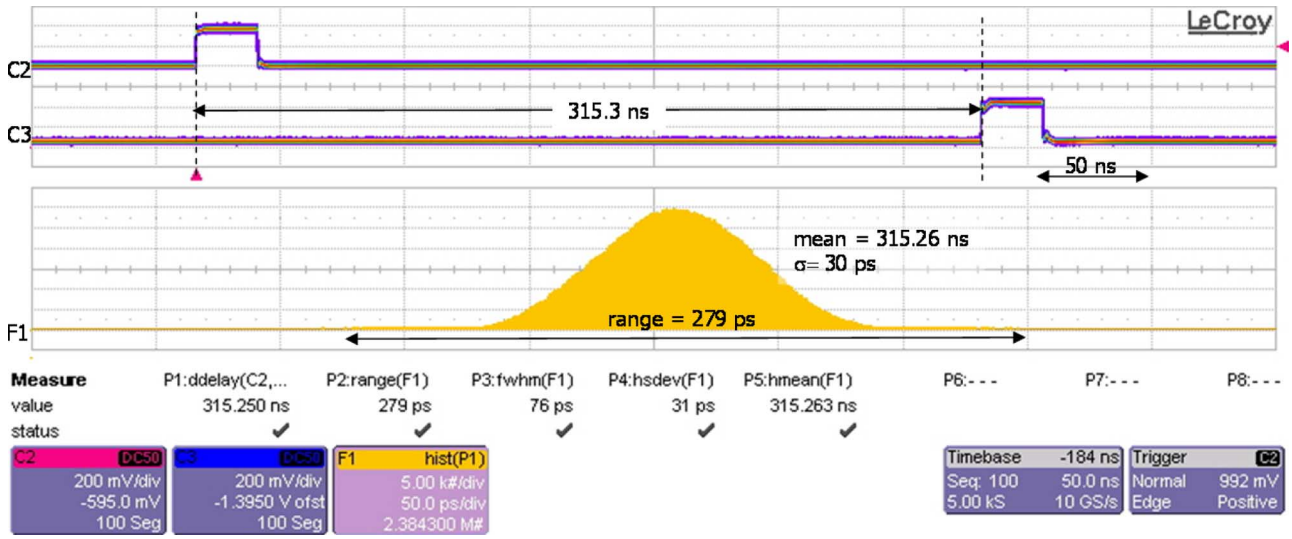

Fig. 10. Histogram of the latency of the link. Topmost trace: transmitted payload bit. Second trace: corresponding received bit. Down: histogram of the latency.

smaller latency (56.25 ns). On the receiver it is the converse, the GTP introduces more latency (118.75 ns) than the alignment and decoding logic (25 ns).

VII. Test Results

In order to test our link, we deployed two off-the-shelf boards (Xilinx ML-505 [18]). The boards route the serial I/O pins of one of the GTPs on the FPGA to SubMiniature version A (SMA) connectors. We connected Tx and Rx GTPs with a pair of 5.0 ± 0.1 ns, $50\ \Omega$ impedance coaxial cables. Transmitted and received payloads are available on SMA test-points and are monitored by an oscilloscope (Fig. 9). We checked that the transmission latency remained always the same during transfers and between subsequent power-ups of the system. For each configuration, we performed a 24-hour test, resetting the transmitter and the receiver every 3 seconds (i.e., simulating a power-cycle) and holding all the acquired waveforms on the scope screen, in order to record latency variations.

Let $\Delta\phi$ be the phase difference between the Tx clock and the Rx clock. As far as it concerns to Configuration One, proper operation can be only ensured for some ranges of $\Delta\phi$, which can be experimentally found by means of the Phase Checker we implemented in the receiver. In these ranges latency is fixed and resistant to power cycles (it keeps the same value after a reset of the system). The histogram in Fig. 10 shows the distribution of the link latency during subsequent resets.

We measured the latency of the transmitter and of the receiver (Fig. 11) with respect to the serial stream. In order to easily find the serial bit corresponding to a pulse on the parallel word, we sent a word sequence containing 1023 zero words and one marker word ($(0020)_{16}$), having all bits set to zero but the fifth from the least significant.

We measured the latency of the transmitter by probing the bit #5 of the payload on a test point and the serial output of the transceiver. We routed the signals to the oscilloscope by means of two cables with same propagation delay (within 0.1 ns). The delays through the cables canceled each other and did not bias the measurement. In our setup, the transmitter latency is $L_{tx} =$ 164.5 ± 0.5 ns. However, this measurement underestimates the latency by a quantity equal to the propagation delay of the FPGA Input/Output Block (IOB) and internal routing. This delay is unknown, but estimated to be at most 7 ns by the Xilinx timing analysis tool. This explains the difference between the measured and the estimated latency of Table II.

TABLE II
Latency of the Building Blocks of the Link

	# of clock cycles	Clock Period (ns)	Block latency (ns)
Transmitter			
CIMT Encoder	4	25	100
Register on falling TXUSRCLK edge	0.5	25	12.5
Total Encoding Latency (fabric)	4.5	25	112.5
Total GTP Latency	4.5	12.5	56.25
Total Transmitter Latency			168.75
Receiver			
Total GTP Latency	9.5	12.5	118.75
Total Decoding Latency (fabric)	1	25	25
Total Receiver Latency			143.75
Total Link Latency			312.5

As far as it concerns the receiver, we remark that the estimated and the measured latencies could differ up to one board clock period. This is due to registering the data on the receiver clock, which has an unknown phase offset with respect to transmitter clock and the serial stream. As an unambiguous figure, we provide the minimum latency obtainable by tuning the clock of the receiver.

We measured the latency of the receiver by probing one of the ends of the differential pair at the input of the transceiver and by measuring the received bit #5 of the payload. In this setup, the oscilloscope receives the stream at the same time of the GTP receiver but it receives the parallel payload bit with a delay equal to the sum of the propagation through the FPGA IOB (≤ 7 ns) and the connection cable (5 ns). Hence, the latency

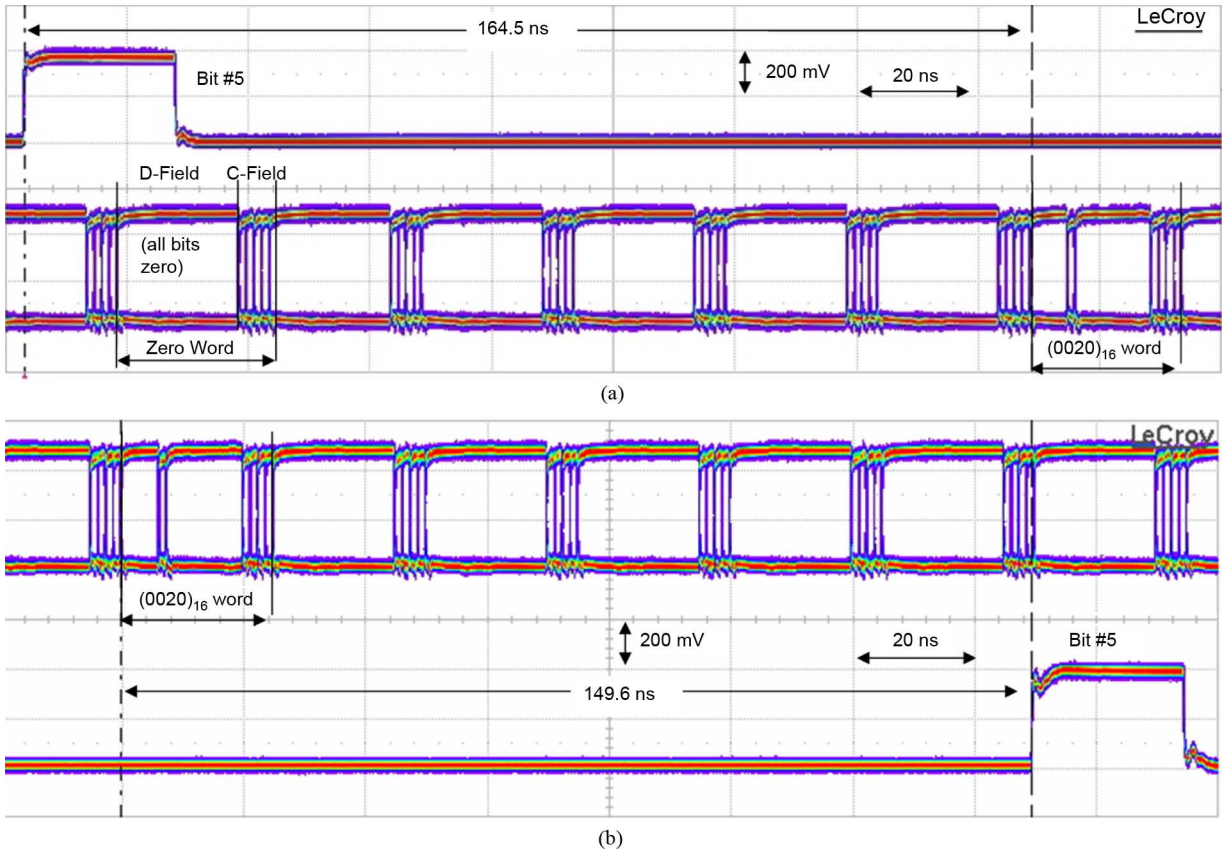

Fig. 11. Oscilloscope screen-shots showing the latencies of the transmitter and of the receiver. (a) Upper trace: transmitted payload bit. Lower Trace: CIMT encoded serial stream. (b) Upper trace: CIMT encoded serial stream. Lower Trace: received payload bit.

is overestimated. The measured latency is $L^1_{rx} = 144.6 \pm 0.6$ ns, we subtracted the cable delay but not the IOB delay which we only have an upper limit for.

As far as it concerns Configuration Two, the link transfers data with a deterministic latency, which as before keeps the same value after a power cycle. The latency of the transmitter is the same as in Configuration One. The latency of the receiver was measured to be $L^2_{rx} = 181.7 \pm 0.6$ ns (without the cable delay), thus the latency of the receiver increased by ~ 37 ns, due to activation of the FIFO. The increment is not exactly 37.5 ns because of internal routing changes between the two implementations. In this configuration, there are no limitations on $\Delta\phi$ but a very small forbidden interval $(90 \pm 10$ ps$)$, which leads to a change of latency of one board clock cycle between subsequent power-ups of the link. We measured this interval by shifting the board clocks with a step of 5 ps and checking whether or not for 20 subsequent resets the link latency remained the same. As discussed in Section II, this problem affects every synchronous system with an arbitrary skew on the clock distribution and it is not dependent on our design. For instance, the same effect occurs with the original G-Link chip-set and with the Configuration One.

We also checked that our G-Link emulator is able to correctly transmit (receive) data toward (from) an Agilent G-Link receiver (transmitter) chip. In order to perform this test, we deployed a ML-505 board and a custom board hosting a G-Link transmitter and a receiver described in [19]. The test showed that the emulator correctly exchanges data with a G-Link chip in both the CIMT encoding modes supported by the HDMP-1032/1034A chip-set.

VIII. Conclusions

High-speed SerDes embedded in FPGAs are typically designed for variable-latency transfers. However, by suitably clocking and configuring two GTP transceivers embedded in Xilinx FPGAs, we have been able to develop a fixed-latency link for synchronous transfers. Our links requires a common reference clock to be distributed both at the transmitter and at the receiver, like the global clock distributed by the TTC system in LHC experiments. The latency of the link is constant during the transfer and even after a reset or a power-cycle of the system. We developed a clocking scheme and two configurations: one with minimum latency, which set some constraints on the skew between the transmitter and receiver board clocks and a second one with a higher latency but without any constraint on the skew. Our configurations support an encoding/decoding external to the transceiver, thus they can be used with any serial protocol. The only requirement is that the external encoding, coding and alignment is performed with a fixed latency. We provided guidelines to use our results also with Lattice and Altera embedded SerDes. However, the fixed-latency operation on those devices should be experimentally tested.

As an example application to discuss our architecture, we designed a serial link to be deployed in the ATLAS L1 barrel

muon trigger receiver boards compatible with the Agilent G-Link chip-set. The link we implemented is also of interest for the trigger system of the K Long Observation Experiment (KLOE) [20]. We successfully tested G-Link/GTPs hybrid configurations. We deployed DLLs to provide the reference clock on to the transceivers, but we have been allowed to do so thanks to the relatively low speed of the link (800 Mbps). For higher-speed links, the use of DLLs or PLLs should be avoided since their jitter performance may be insufficient to ensure a reliable operation of the transceiver. We remark that GTPs can achieve multi-Gigabit data-rates ($\sim$ 3.75 Gb/s) and therefore are appealing for deployment in super LHC experiments, where the requested bandwidth will increase with respect to the present.

ACKNOWLEDGMENT

The authors would like to thank G. Guasti and F. Contu of Xilinx Italy for their support and help in configuring the GTP transceiver.

REFERENCES

[1] Ph. Busson, L. Dobrzynski, A. Karar, T. Romanteau, P. Moreira, J. L. Brelet, J. R. Macé, M. Défossez, M. Crowley, and C. Gannon, "Embedding deserialisation of LHC experimental data inside field programmable gate arrays," in *Proc. 8th Workshop on Electronics for LHC Experiments*, 2002, Contribution ID A51.

[2] "Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, Module 2," Xilinx, 2007, pp. 10–17 [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf

[3] P. Moreira, T. Toifl, A. Kluge, G. Cervelli, F. Faccio, A. Marchioro, and J. Christiansen, "G-link and gigabit ethernet compliant serializer for LHC data transmission," in *Nucl. Sci. Symp. Conf. Rec.*, Oct. 15–20, 2000, vol. 2, pp. 9/6–9/9.

[4] C.-S. Yen, R. C. Walker, P. T. Petruno, C. Stout, B. W. H. Lai, and W. J. McFarland, "G-link: A chipset for gigabit-rate data communication," *Hewlett Packard J.*, vol. 43, no. 5, pp. 103–115, Oct. 1992.

[5] G. Aglieri Rinella, A. Klugea, and M. Krivdaab, "The level 0 pixel trigger system for the ALICE experiment," *J. Instrumen.*, vol. 1, no. 1, p. P01007, 2007.

[6] A. Hidvégi, D. Eriksson, and C. Bohm, "A small portable test system for the TileCal Digitizer System," in *Proc. Topical Workshop on Electronics for Particle Physics*, Naxos, Greece, Sep. 19, 2008, pp. 513–515.

[7] G. Iles *et al.*, "Performance and lessons of the CMS global calorimeter trigger," in *Proc. Topical Workshop on Electronics for Particle Physics*, Naxos, Greece, Sep. 15–19, 2008, pp. 129–132.

[8] E. Aslanides *et al.*, "The level-0 Muon Trigger for the LHCb experiment," *Nuclear Instruments and Methods in Physics Research, Section A*, vol. 579, no. 3, pp. 989–1004, 2007.

[9] P. Moreira and A. M. Kloukinas, "The GBT : A proposed architecure for multi-Gb/s data transmission in high energy physics," in *Proc. Topical Workshop on Electronics for Particle Physics*, Prague, Czech Republic, Sep. 3–7, 2007, pp. 332–336.

[10] "ATLAS Detector and Physics Performance, Technical Design Report Volume I," ATLAS Collaboration, 1999 [Online]. Available: http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/TDR/physics_tdr/printout/Volume_I.pdf

[11] B. G. Taylor, for the RD12 project collaboration, "TTC distribution for LHC detectors," *IEEE Trans. Nucl. Sci.* vol. 45, no. 3, pp. 821–828, Jun. 1998..

[12] R. C. Walker, C. L. Stout, J. Wu, B. Lai, C. Yen, T. Hornak, and P. T. Petruno, "A two-chip 1.5-GBd serial link interface," *IEEE J. Solid-State Circuits*, vol. 21, no. 12, pp. 1805–1811, Dec. 1992.

[13] "Agilent HDMP 1032-1034 Transmitter-Receiver Chipset Datasheet," Agilent, 2001 [Online]. Available: http://www.physics.ohio-state.edu/~cms/cfeb/datasheets/hdmp1032.pdf

[14] S. H. Unger and C. J. Tan, "Clocking schemes for high-speed digital systems," *IEEE Trans. Comput.*, vol. C-35, no. 10, pp. 880–895, Oct. 1986.

[15] "Virtex-5 FPGA RocketIO GTP Transceiver User Guide, UG196 (v1.7)," Xilinx, 2008 [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug196.pdf

[16] "Stratix IV Device Handbook," Altera, 2008 [Online]. Available: http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf

[17] "LatticeSC/M Family Data Sheet," Lattice Semiconductor Corp., 2008 [Online]. Available: http://www.latticesemi.com/documents/DS1005.pdf

[18] "ML505/ML506/ML507 Evaluation Platform User Guide, UG347 (v3.0.1)," Xilinx, 2008 [Online]. Available: http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf

[19] A. Aloisio, F. Cevenini, and V. Izzo, "Do's and don'ts with the Agilent's G-link chipset," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 3, pp. 795–800, Jun. 2006.

[20] M. Adinolfi *et al.*, "The trigger system of the KLOE experiment," *Nucl. Instr. Meth.*, vol. A492, pp. 134–146, 2002.