PlasticNet+: Extending multi-FPGA interconnect architecture via Gigabit transceivers

Carlos Salazar-García*, Ronny García-Ramírez‡, Renato Rímolo-Donadío‡,
Christos Strydis§, and Alfonso Chacón-Rodríguez‡
*Mechatronics, ‡ Electronics Engineering, Instituto Tecnológico de Costa Rica, Cartago, Costa Rica
§Dept. of Neuroscience, Erasmus Medical Center, Rotterdam, The Netherlands
Email: {csalazar, rgarcia, rrimolo, alchacon}@tec.ac.cr, {c.strydis}@erasmusmc.nl

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401058

***Abstract*—This paper addresses the communication challenges posed in multi-FPGA systems, by improving a custom FPGA interconnect architecture via the high-speed transceivers available in modern FPGA development boards. The proposed network interconnection, built upon the PlasticNet architecture, is evaluated using the high-speed serial transceiver in Zynq ZC706 FPGA boards. Results show a best-case latency of only 300 ns, demonstrating equivalent results in terms of latency on a par with the known BlueLink framework, but with the plus of having total re-configurability across the different layers of its network interconnection model. This makes the current proposal a competitive option for the development of distributed, heterogeneous multi-FPGA processing systems.**

***Index Terms*—Custom FPGA networks, HLS, AXI4, interconnect architecture, inter-FPGA communication, multi-FPGA distributed processing.**

I. Introduction

The use of FPGAs has become widespread in distributed processing systems due to their high-logic capacity, integration of standard processing cores, floating-point capabilities, flexibility, power efficiency, embedded high-speed transceivers, and the availability of development tools such as High-Level Synthesis (HLS) for the rapid prototyping of complex designs (see [1]–[3]). But as the size and complexity of modern data processing grows, more and more applications hit the maximum logic capacity when ported even to high-end FPGAs, resulting in the need of the partitioning of designs among several FPGAs. Applications such as the simulation of biologically accurate neural networks [4]–[6], stencil computation [7], and hardware accelerators in common cluster infrastructures [8], [9] are current examples.

Distributing processing across different FPGAs entails efficient communication among the data-processing nodes, especially in terms of effective data-transfer rates and their interconnect flexibility. A proposal is given in [10] to tackle both issues: PlasticNet, a custom FPGA interconnected architecture aimed at distributed applications requiring massive data processing. PlasticNet enables the interconnection of different processing nodes (PNs) located both within the same FPGA and in neighboring FPGAs, through a reliable, flexible, and efficient custom protocol. PlasticNet is compatible with different types of PNs ranging from a simple module implemented in some hardware description language (HDL), to a complex block implemented using high-level synthesis (HLS), or a hard/soft embedded FPGA Processor system communicating through a DMA interface.

PlasticNet's performance has been presented in [10], but its physical interface was restricted to single-ended signaling and low-voltage differential signaling (LVDS), with its associated limited bandwidth. Latest generation FPGAs, nonetheless, includes faster, multi-Gigabit transceivers that easily surpass the attainable bandwidth provided by LVDS interconnects.

This paper presents the integration of high-speed transceivers within PlasticNet's interconnection architecture. Its performance is evaluated on a multi-FPGA system composed of a stack of Zynq ZC706 FPGA boards, with results compared against other interconnection alternatives found in literature. The paper is organized as follows: section II gives a literature review. Section III glances over PlasticNet's architecture. Section IV details the design flow proposed for handling the multi-gigabit transceivers embedded in modern FPGA boards. Section V evaluates the performance of the improved interconnection network architecture. Lastly, section VI gives the main conclusions and discusses future work.

II. Review on multi-FPGA systems

A general classification of multi-FPGA-based systems is provided in [11]: Hardwired Off-the-Shelf, Custom and Cabling. The first option comprises ready-made multi-FPGA boards. The second group gathers systems consisting of build-your-own multi-FPGA platforms, where physical interconnections among the different FPGAs are through printed hardwiring. The last category takes advantage of modern FPGA evaluation boards and proposes the interconnection of several single FPGA boards using external cables. After reviewing the characteristics of these three types of systems, authors of [10] have argued that cabling-type systems are not only more affordable, but are easier to deploy too, coming with the adaptability for varying processing needs as those required for instance for flexible interconnect frameworks such as PlasticNet.

PCIe would seem a straightforward solution, as proposed for instance in [12], [13], which presented multi-FPGA applications communicated through a PCIe switch. However, most FPGA boards only contain one PCIe port, limiting physical interconnection across different network topologies. Besides,

its implementation in hardware is not straightforward and requires either the use of some proprietary hardware block or an open-source high-performance PCIe streaming library, such as the one developed in [14].

An alternative is given in [5], where authors report BlueHive, a multi-FPGA system based on an FPGA-to-FPGA interconnect library called BlueLink, allowing to build clusters of FPGAs boards for massive data processing applications, through cheap interconnect fixtures. By incorporating a thoroughly defined network interconnection architecture and using of all the high-speed transceivers built into the FPGA evaluation boards, BlueLink proved, to be a good choice for those in search of flexible, scalable multi-processing using FPGAs.

PlasticNet builds on the same goals as BlueHive and its BlueLink interconnect framework (see [10]) proposing a custom FPGA interconnect architecture designed for distributed processing applications as well. But, unlike BlueLink, PlasticNet is completely re-configurable, therefore it makes no assumptions on the type of blocks being connected in the application layer. And though PlasticNet's preliminary results were not competitive against BlueHive (with an average latency 7.5 times slower: 16 μs against the 500 ns reported in [15]), these numbers were considered as promising, since PlasticNet's first version only supported a serial transmission physical layer operated using LVDS.

III. Overview of PlasticNet

Figure 1 shows the general PlasticNet architecture: Fig. 1(a) presents the hardware blocks used to build PlasticNet's communication infrastructure, while Fig. 1(b) depicts its protocol hierarchy.

At the highest level is the application layer, governed by the user-defined application, which in the case of this paper is made up of a set of hardware/software PNs, each of them implementing a part of an algorithm to be accelerated.

Each PN sends messages of up to 8 kB of data. In addition to the message, each PN places a 4-byte header, where the first byte is a global identifier of the transmitting node called `TX_UID`, the second byte is a global identifier of the receiving node called `RX_UID` and the remaining two bytes are reserved for adding extra functionalities, if required. Both `TX_UID` and `RX_UID` are unique for the entire network.

In the transport and network layer, each message is divided into packets of user-defined size, depending on each application, which varies in the range of 128 to 4096 bits. In each packet, an extra 4-byte header is added, where the first byte is a bus identifier for intra-FPGA data transmission called BS_ID. The second identifier is the identifier of the transmitting FPGA called FPGA_ID, which is used as an end-of-transmission mechanism when a node broadcasts packets. Finally, the last 2 bytes store the payload size, embedded in each packet. The transport layer is implemented within the unpacking/packaging unit while the network layer is implemented within the internal bus. The internal bus uses a daisy-chain topology (More details on this block are available in [16]). The communication between the two layers is done through FIFOs. For intra-FPGA communication, only the transport layer and the network layer are required; thus, latency is optimized. Figure 2 depicts PlasticNet's packet format.

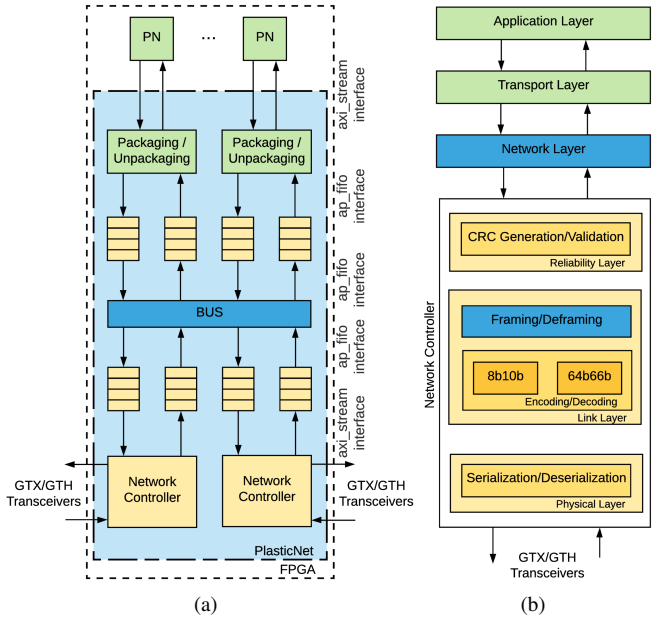

Figure 1. General architecture of PlasticNet: (a) hardware blocks and data flow, (b) layer hierarchy and Network Controller details. Blocks or layers implemented via HLS are green-colored. Verilog blocks are in blue (shading indicates the abstraction level). Physical layer IPs and FIFO blocks are yellow-colored and were generated using Xilinx's IP libraries.

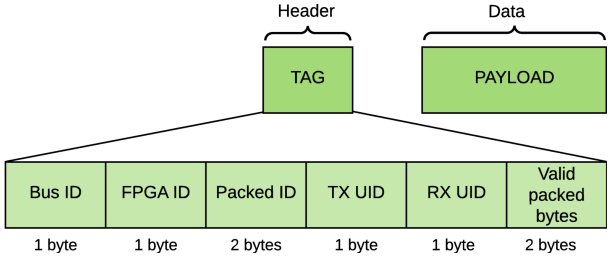

Figure 2. PlasticNet's packet format.

The Network Controller is the unit in charge of communicating between the different FPGAs, and it is the focus of this work. Its architecture is outlined in the next section.

IV. Network Controller

The Network Controller implements the reliability, link, and physical layers of PlasticNet, and is responsible for sending/receiving packets among FPGAs. Figure 3 illustrates its architecture: the yellow blocks are units implemented using Xilinx IPs and the blue ones for units using Verilog HDL.

High-speed serial transceivers are handled with the Xilinx®LogiCORE™ IP Aurora 8B/10B core, that provides the resources to manage the transceivers. Aurora can handle either GTP, GTX, or GTH transceivers, instantiating the different-lane logic modules of the FPGA chip. During channel initialization, verification phases and bonding are performed.

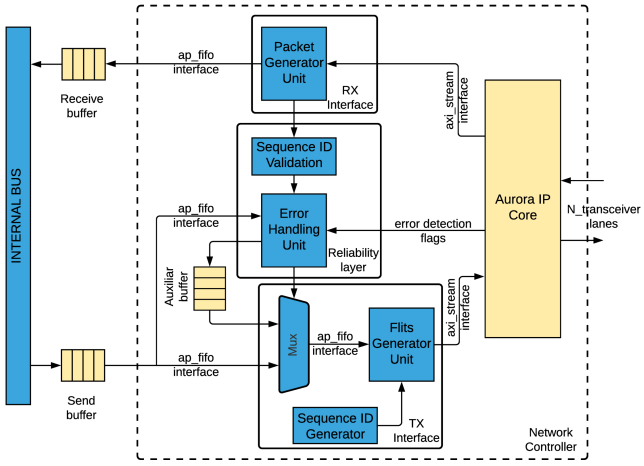

Figure 3. Network Controller block.

During operation, the different-lane logic blocks are analyzed for detecting errors and maintaining the channel operating properly. The encoding 8B/10B implemented in this Xilinx IP allows the board-to-board interconnection using external cables and backplanes. This lightweight, easy to scale IP Core provides high data-transfer rates with efficient channel management as reported in [17].

When Aurora 8B/10B is connected to an Aurora channel partner, it automatically initializes the channel. After initialization, data is sent through the channel using either streams or frames. PlasticNet uses frames, which provide better data transmission control than typical streaming formats, providing for extra channel reliability. Since frames in Aurora can be of any size, and considering that the Aurora core provides an AXI4-Stream user interface for performing flow-control functions and moving data from the channel to the application and vice versa, an AXI4-Stream interface is used to encapsulate each packet within each frame. The data port of each AXI4-Stream interface depends on the number of lanes selected and the lane width.

For the transmission of each packet, it must be divided into multiple flits whose number of bits must be equal to the size of the data port of the AXI4-Stream TX user interface. Each flit is sent to the Aurora IP Core with each positive edge of the user clock provided by this IP. The size of each frame is controlled by the `t_valid`, `t_last` and `t_keep` AXI4-Stream flags. These operations are completed in the block called Flits Generator Unit.

Regarding reliability, the same strategy followed in [11] was adopted. On the transmitter side, a Cyclic Redundancy Check (CRC) is performed on each packet. Also, the `BS_ID` is replaced by a 8-bit packet sequencer generated by the unit called Sequence ID Generator. An auxiliary buffer stores a copy of each transmitted packet. On the receiver side, the Network Controller checks the CRC and verifies that the sequence identifier matches the expected value. If one of the two verification mechanisms detects an error, the receiver will request the transmitter to resend the packet. As soon as the transmitter receives this request, it needs only read the data stored in the auxiliary buffer, without resorting to higher layers to solve the problem. These functions are implementing in the Error-Handling Unit.

Table I
DEVICE UTILIZATION SUMMARY FOR THIS WORK (XILINX'S VIVADO 2018.3 POST-SYNTHESIS REPORT FOR A ZYNQ®-7000 SOC ZC706).

Number of PNs	Packet Size (bits)	128	256	512	1024	2048	4096
2	Slice Registers (%)	3.30	3.39	3.59	3.98	4.74	7.86
4		5.6	6.19	6.58	6.75	8.29	11.38
2	BRAMs (%)	1.28	2.56	5.12	10.24	20.48	40.96
4		1.92	3.84	7.68	15.36	30.72	61.44
2	LUTs (%)	3.39	3.45	3.70	4.18	5.74	7.07
4		5.44	5.66	6.10	6.97	8.69	12.16

The packet generator unit takes data from the channel through the AXI4-Stream RX user interface and reassembles the packets. Using a routing table, the `BS_ID` identifier is incorporated again in each packet. Finally, each packet is sent out of the Network Controller to the Receive Buffer, which connects the Internal Bus with the Network Controller.

Our Network Controller can thus be integrated into the PlasticNet framework. This permits the creation of a multi-FPGA network interconnection architecture which is capable of efficiently handling all the transceivers embedded in modern FPGA evaluation boards, with the additional advantage that the external interconnection topologies between FPGAs are re-configurable depending on the needs of each application.

V. EVALUATION

PlasticNet was evaluated using a stack of four Zynq®-7000 SoC ZC706 boards selected primarily because of their affordability and availability. The FPGA has 16 GTX transceivers, accessible on the evaluation board through the FMC (FPGA Mezzanine Card) connector, SMA connectors, small form-factor pluggable plus (SFP+) connectors and a PCI Express adapter. The four FPGA boards are interconnected for the tests reported here using a ring topology since: 1) evaluation boards have a limited number of high-speed serial links and 2) few applications exhibit an all-to-all interconnection architecture among all the PNs which compose the distributed processing system. Two Network Controllers were instantiated per FPGA, one in charge of transporting data to the board located to the left, and another in charge of moving data to the board located in the opposite direction, following a ring topology. In addition, the internal buffers were sized to avoid congestion, allowing for up to 1024 data packets. The clock frequency was fixed to 166.66 MHz and the line rate was set to 5.0 Gbps. SMA cables were used for the physical interconnection.

The area of the network interconnection architecture was evaluated while exploring two parameters: packet size and the number of PNs contained by each FPGA. For a more adequate evaluation, a situation where all the transceivers are enabled was assumed. As seen in table I, data packets smaller than 1024 bits seemed more suitable.

A pseudo-random sequence number generator (a Linear Feedback Shift Register (LFSR) written in C++ and synthe-

sized using Xilinx Vivado 2018.3 HLS) served as PN, in order to test communication and evaluate PlasticNet's performance, in terms of latency and channel overhead. A variable number of PNs is distributed both within the same FPGA as well as in neighboring FPGAs. Both inter-FPGA and intra-FPGA communication are managed by PlasticNet. Message sizes range from 1 kB to 8 kB of data, sent to the PNs using different packet sizes through the network infrastructure. Figure 4 shows the channel overhead in terms of the packet size, assuming a CRC32 for reliability checks. The results tend to the expected theoretical value (since bandwidth overhead is 20% in 8b/10b encoding), as the number of bits per packet increases.

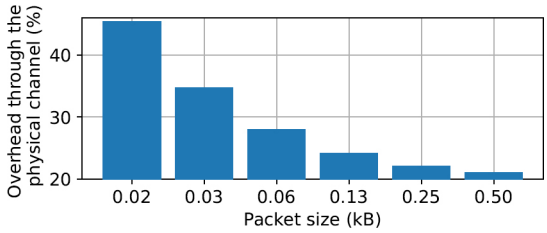

Figure 4. Channel overhead of the transmission channel after incorporating Aurora 8b10b Core to PlasticNet, as a function of the packet size.

To determine latency, a distributed-processing application with two PNs per FPGA is evaluated. Eight nodes were instantiated within the four FPGAs, two per FPGA. Each PN sends messages of varying size to its successor in the network, testing thus both the intra and inter FPGA communication links. Each message is transmitted using packets of different sizes. Figure 5(a) shows the average transmission latency of the messages between FPGAs, as a function of the packet size, where each curve represents a different message size in relation to the packet size. Figure 5(b) plots the PlasticNet's average latency considering different packet injection rates, starting at 10% and ending at 100%, using a FIFO with a depth of 1024 elements. Latency was also evaluated for different packet sizes: latency scales linearly regardless of the packet size as the injection rate increases.

Figures 5(a) and 5(b) suggest an optimal packet size of 64 B. Such packet size allows for an average latency of only 300 ns under lightly loaded conditions (input FIFOs usually near-empty). At half-occupancy, latency was of 200 μs, a more likely situation with FIFOs at half capacity (FIFOs in this case had a depth of 1024 elements), using a single serial link in the physical layer. Figure 6 shows the obtained latency in this work against BlueLink's, with 10G Ethernet used as a reference, under the same conditions as in [15]: a single lane at 10 Gbps, short physical links, same-size flits and identical workload conditions for the FIFOs. Note that the latency is comparable to that one reported by BlueHive, and surpasses Ethernet 10G's (consider as well that the latter is limited to one PHY controller per board, while PlasticNet, same as BlueHive, has more transceivers available). Besides, also as in BlueHive, as more transceivers are added in the physical layer, the probability that FIFOs operate under lightly loaded

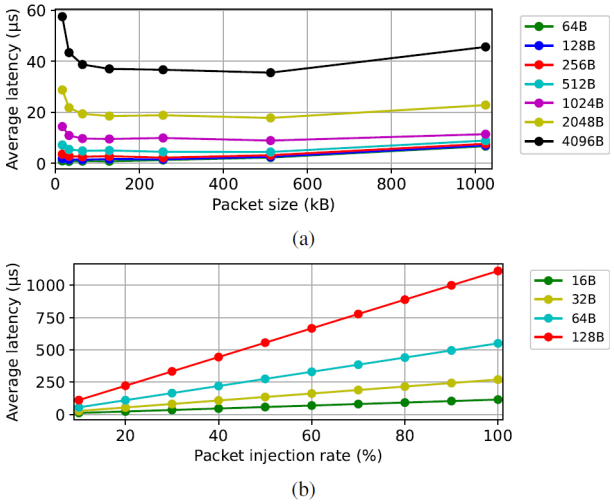

Figure 5. (a) Average latency when sending different size messages, varying the packet size. (b) Average latency as a function of the packet injection rate using FIFOs with a depth of 1024. Each curve depicts a different packet size.

conditions is higher, thus improving latency even more. All this without losing PlasticNet's higher degree of configurability over BlueLink, allowing for easier interconnection of heterogeneous processing nodes in the highest layer, and the use of alternative interconnection topologies (inter and intra-FPGA) in the lower layer, depending on the application.

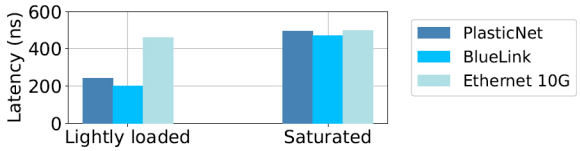

Figure 6. Latency comparison between the different network controllers of PlasticNet, BlueLink and Ethernet MAC, at 10 Gbps. The comparison was made under the same conditions.

VI. Conclusions

This paper has presented the integration of high-speed transceivers into the PlasticNet framework. Evaluations in terms of area, channel overhead and latency show that the improved network architecture is feasible for distributed-processing applications in multi-FPGA systems, with a latency just over 300 ns for lightly loaded links for a ring-based multi-FPGA interconnect, half the latency of an Ethernet 10G link. PlasticNet's performance is practically equal to what has been reported in [15], but with the added advantages of its adaptability to a wider variety of networking topologies, and its capabilities of integrating modules written either in HDL, high level languages, or even hard macros such as integrated processors, due to its full compatibility with the AXI4 standard.

Future work includes: 1) developing PCB adapters for the transceivers embedded in the target boards' PCI and FMC connectors and, 2) creating a centralized router for testing other interconnection topologies.

REFERENCES

[1] K. D. Underwood *et al.*, "Closing the gap: CPU and FPGA trends in sustainable floating-point BLAS performance," in *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2004, pp. 219–228.

[2] K. D. Underwood, "FPGAs vs. CPUs: trends in peak floating-point performance," in *FPGA*, 2004.

[3] K. Alfaro-Badilla *et al.*, "Prototyping a Biologically Plausible Neuron Model on a Heterogeneous CPU-FPGA Board," in *2019 IEEE 10th Latin American Symposium on Circuits Systems (LASCAS)*, Feb 2019, pp. 5–8.

[4] A. Sripad *et al.*, "SNAVA—A real-time multi-FPGA multi-model spiking neural network simulation architecture," *Neural Networks*, vol. 97, pp. 28 – 45, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608017302150

[5] S. W. Moore *et al.*, "Bluehive - a field-programable custom computing machine for extreme-scale real-time neural network simulation," in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, 2012, pp. 133–140.

[6] A. Zjajo *et al.*, "A Real-Time Reconfigurable Multichip Architecture for Large-Scale Biophysically Accurate Neuron Simulation," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 2, pp. 326–337, 2018.

[7] K. Sano *et al.*, "Multi-FPGA Accelerator for Scalable Stencil Computation with Constant Memory Bandwidth," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 695–705, 2014.

[8] O. Knodel *et al.*, "Integration of a Highly Scalable, Multi-FPGA-Based Hardware Accelerator in Common Cluster Infrastructures," in *2013 42nd International Conference on Parallel Processing*, 2013, pp. 893–900.

[9] A. Putnam *et al.*, "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," *IEEE Micro*, vol. 35, no. 3, pp. 10–22, 2015.

[10] C. Salazar-García *et al.*, "Plasticnet: A low latency flexible network architecture for interconnected multi-FPGA systems," in *2020 IEEE 3rd Conference on PhD Research in Microelectronics and Electronics in Latin America (PRIME-LA)*, 2020, pp. 1–4.

[11] Q. Tang *et al.*, "Performance Comparison between Multi-FPGA Prototyping Platforms: Hardwired Off-the-Shelf, Cabling, and Custom," in *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*, May 2014, pp. 125–132.

[12] V. Viswanathan *et al.*, "Massively Parallel Dynamically Reconfigurable Multi-FPGA Computing System," in *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, 2015, pp. 165–165.

[13] S. Gao and J. Chritz, "Characterization of OpenCL on a scalable FPGA architecture," in *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, 2014, pp. 1–6.

[14] M. Vesper *et al.*, "Jetstream: An open-source high-performance PCI Express 3 streaming library for FPGA-to-Host and FPGA-to-FPGA communication," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–9.

[15] A. Theodore Markettos *et al.*, "Interconnect for commodity FPGA clusters: Standardized or customized?" in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2014, pp. 1–8.

[16] R. Garcia-Ramirez *et al.*, "Pre-Synthesis Evaluation of Digital Bus Micro-Architectures," in *2020 IEEE 3rd Conference on PhD Research in Microelectronics and Electronics in Latin America (PRIME-LA)*, 2020, pp. 1–4.

[17] Xilinx. (2016) Aurora 8B/10B v11.0 LogiCORE IP Product Guide. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/aurora_8b10b/v11_0/pg046-aurora-8b10b.pdf