Multiplexing AER Asynchronous Channels over LVDS Links with Flow-Control and Clock-Correction for Scalable Neuromorphic Systems

A. Yousefzadeh[1], M. Jabłoński[2], T. Iakymchuk[3], A. Linares-Barranco[4], A. Rosado[3], L. A. Plana[5], T. Serrano-Gotarredona[1], S. Furber[5], and B. Linares-Barranco[1]

[1]Instituto de Microelectrónica de Sevilla (CSIC and Univ. de Sevilla), Sevilla, Spain {reza, bernabe}@imse-cnm.csic.es
[2]AGH University of Science and Technology, Krakow, Poland
[3]School of Engineering (Univ. of Valencia), Spain
[4]Dept. of Computer Architectures (Univ. of Sevilla), Spain
[5]Dept. Comp. Science, University of Manchester, UK

Abstract— **Address-Event-Representation (AER) is a widely extended asynchronous technique for interchanging "neural spikes" among different hardware elements in Neuromorphic Systems. Conventional AER links use parallel physical wires together with a pair of handshaking signals (Request and Acknowledge). Here we present a fully serial implementation using bidirectional SATA connectors with a pair of LVDS (low voltage differential signaling) wires for each direction. The proposed implementation can multiplex a number of conventional parallel AER links per LVDS physical connection. It uses flow control, clock correction, and byte alignment techniques to transmit 32-bit address events reliably over multiplexed serial connections. The setup has been tested using commercial Spartan6 FPGAs reaching a maximum event transmission speed of 75Meps (Mega Events per second) for 32-bit events at 3.0Gbps line data rate.**

Keywords— ***Neuromorphic Systems, Virtual Wiring, AER (Address Event Representation), Scalable Neuromorphic Systems.***

I. Introduction

Address Event Representation (AER) is a popular "virtual wiring" technique used by many neuromorphic hardware engineers to interconnect spiking neuromorphic systems. As neuromorphic systems have been scaling up in size and complexity over the years, researchers have developed more complex and smarter AER "variations" to improve efficiency, reconfigurability and reliability. Since very early [1] it became apparent that the original parallel-AER (pAER) bus bulkiness would seriously limit the scalability of AER systems to arbitrary size, and researchers started to look at serial connectivity options. Fully bit-serial Low-Voltage-Differential-Signaling (LVDS) allow for multi-gigabit-per-second communications with only a pair of wires. However, with only two uni-directional differential wires it is not obvious to implement a handshaking protocol per event transmission, nor a flow-control scheme to signal data congestion at the receiver side. Berge et al. [2] experimented with the 2.5Gbps (Giga-bit-per-second) LVDS IP block available in the VirtexII-Pro Xilinx FPGAs, achieving 41.66Meps (Mega-events-per-second) for 16-bit AEs (Address Events).

However, there was no hand-shaking nor flow-control mechanism to avoid data loss in case the receiver side would be temporarily slower than the transmitter side. Fasnacht et al. [3] reported a bit-serial interface based on off-the-shelf 16-bit Serializer/Deserializer commercial components (TLK 2501/3101) connected to a Spartan3E parallel event AER processor, using a second backwards LVDS link for flow control signaling. The bit-serial link could operate at 2.5Gbps (TLK 2501) or 3.125Gbps (TLK3101) line speed and also used 8b/10b encoding to allow for idle commas. On the LVDS backward link the receiver puts a square wave whose frequency signals whether to stop or resume event transmission from the sender. Using this setup the link could transmit 32-bit events at a maximum rate of 62.5Meps (for 2.5Gbps) or 78.125Meps (for 3.125Gbps), at the cost of sacrificing one LVDS backward link for flow control. Zamarreño et al. [4] developed a bi-directional LVDS link using Virtex6 FPGA Rocket-I/O IPs. In this scheme two 2.5Gbps LVDS links are used, each for communicating 32-bit events in each direction with 8b/10b encoding. Flow-control in each direction is implemented by special control symbols

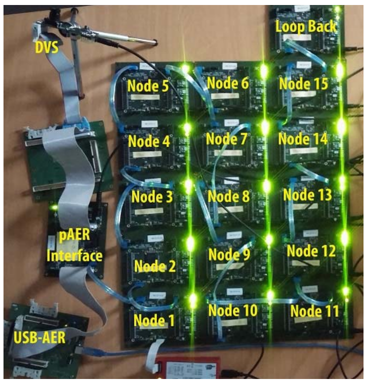

Fig. 1: Example setup of seventeen Spartan6 PCBs communicating through bidirectional LVDS SATA orthogonal links, plus a DVS cameras and USB-AER board communicating through parallel AER buses with one of the Spartan6 PCBs.

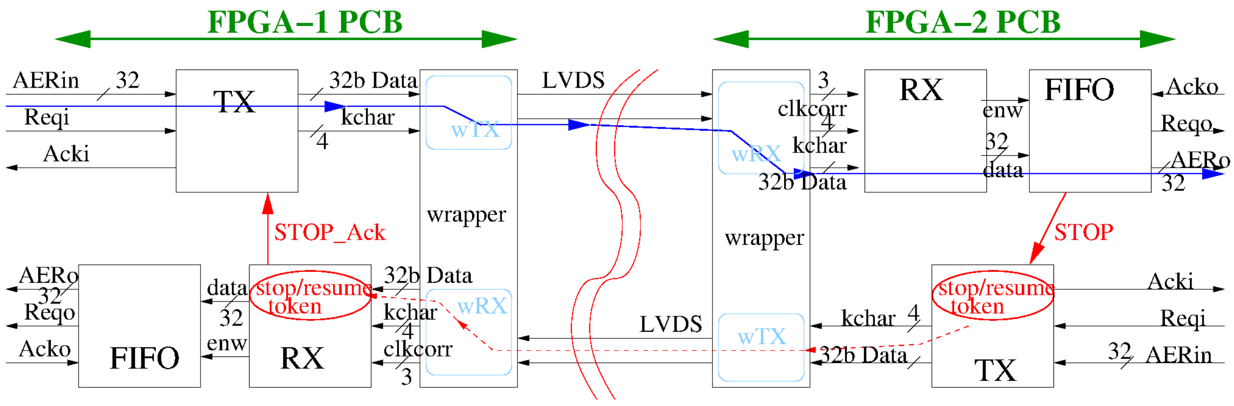

Fig. 2: Block Diagram of Bidirectional LVDS link communicating two parallel AER asynchronous links of opposite direction

sent in the opposite link direction to toggle between the stop and resume states. However, this design did not include any clock-correction support, which limits the approach to the case when all FPGAs use the same physical reference clock, thus hampering scalability.

All previous asynchronous pAER to bit-serial AER conversion schemes operate correctly if there is only one clock domain. However, this is not viable when one wants to interconnect multiple FPGAs, each with its own local-clock synchronous event processing subsystem, together with multiple bidirectional LVDS links per FPGA. In this case, each FPGA will have one or more local clock domains, which can be interfering with the clock domains of neighboring FPGAs. Under these circumstances, it is necessary to use some clock correction technique in order to compensate for clock frequency/phase drifts and avoid sudden byte misalignment problems and data loss. In a preliminary work [5] we exploited the use of elastic buffers available within Xilinx Rocket-I/O serial LVDS IPs, although it was used for fully synchronous handshake-less systems deployed over multiple FPGAs. Each link was unidirectional since the backward LVDS path was fully used for handling flow control. More recently, the SpiNNaker team has developed bidirectional bit-serial LVDS links [6] to bundle eight 2-of-7 AER multi-symbol inter-SpiNNaker-chip links [7] into one bit-serial SATA link. This scheme has been developed to interconnect multiple (up to 1200) 48-chip SpiNNaker Boards [8]. The scheme uses flow-control, clock correction, together with a complex framing protocol that samples the eight channels and performs CRC (Cyclic Redundancy Checks) to increase reliability. This introduces extra overheads, limiting the maximum throughput theoretically to 50Meps (maximum 6.25Meps per channel).

Here we present an extended version of the one reported earlier [5], but with fully bidirectional bit-serial LVDS communication capability, with token-based flow-control protocol together with clock correction capability, as well as a robust interface to conventional parallel AER ports (like those used in AER sensor chips) with 4-phase handshaking asynchronous communication. Using a 3.0Gbps LVDS line transmission rate it is possible to achieve 32-bit 75.0Meps sustained transmission in each direction of the link. Fig. 1 shows an example target setup consisting of an array of 17 AER-Node Boards [5] interconnected through SATA to their neighbors. Additionally, an AER retina sensor and a USB-AER board [13] are connected to the setup through parallel AER buses [9].

II. Bidirectional LVDS AER Bit-Serial Link with Flow-Control, Clock Correction and Byte-Alignment

8b/10b encoding [10] transforms 8-bit bytes into 10-bit words while dc balancing zeros and ones. Extra dc-balanced 10-bit words are available, which can be used by the user as “command” characters, also called “k-chars” or “commas”. Here we use some of these k-chars to perform flow control, byte alignments and clock corrections.

Fig. 2 shows a block diagram of a bidirectional AER link using two FPGA PCBs connected through a single SATA cable (containing two pairs of wires). Each FPGA connects to one hand-shaken 32-bit pAER sender and one pAER receiver. The “wrapper” block contains a wrapper transmitter sub-block wTX and a wrapper receiver sub-block wRX. wTX takes as input a 32-bit clock-synchronous “DATA” word and a 4-bit “k-char” word. Each of the four “k-char” bits indicate whether the corresponding four bytes in the 32-bit “DATA” word are either a control comma byte or a regular user data byte. In our case, LVDS line rates were 3.0Gbps, obtained from a low jitter differential 150MHz reference Xtal oscillator on the FPGA PCB. This means that the 32-bit event data, transformed into a 40-bit sequence by 8b/10b encoding, needs $40/(3 \times 10^9\,\text{Hz}) = 13.3\text{ns}$ to be transmitted. The wrapper provides two reference clocks for the user circuitry. One at frequency $f = 1/13.3\text{ns} = 75.0\text{MHz}$, at the rising edges of which the user circuitry has to provide the 32-bit parallel “DATA” word and the corresponding 4-bit “k-char” word. The other reference clock has frequency $4f = 300\text{MHz}$ (because the 32-bit “DATA” word contains 4 bytes) and is synchronized to clock f. Latency of transmission in our setup is 20 clock cycles

(266ns). The user designed circuitry within each FPGA in Fig. 2 comprises 3 blocks:

1) *Transmitter Block* (TX). This block handles the asynchronous handshaking with the input 32-bit AERin port, the synchronization between the asynchronous and synchronous domains and "Stop/Run" state for the transmission of data which is used by the flow control protocol. Also, this block provides for each clock cycle the 32-bit "DATA" word and the 4-bit "k-char" word required by the "wrapper" (wTX), generates startup byte alignment sequences, and inserts control symbols for flow control, clock correction, idle commas, and periodic commas for alignment.

2) *Receiver Block* (RX). This block receives and separates data and control commas. Data symbols are sent to the FIFO block, while control commas are interpreted and executed for proper flow control, word (re)alignment, and clock correction.

3) *FIFO block*. This block accumulates 32-bit synchronous data symbols from the RX block into a FIFO register, while it empties the FIFO by sending data out to the AERo port. This block handles the asynchronous handshaking with the output 32-bit AERo port and the synchronization with the synchronous data "DATA" clock domain. If the FIFO gets filled up above a threshold it will trigger the flow control mechanism.

Fig. 2 illustrates flow control mechanism. If FIFO block is close to getting full, it sends a 4-byte stop token control comma through the backward link to its corresponding TX block on the transmitter side, setting it into 'stop' mode. In this mode Ack will not be acknowledged and the parallel AER port stops taking new events. Once the FIFO block has enough free space, a 4-byte resume token is sent in the same manner to disable TX block stop mode and resume communication. For clock correction, periodic 4-byte control commas are inserted in the data flow by the transmitter. If the receiver clock is faster/slower it will erase/insert such commas through its elastic buffer, thus equalizing the effective data rate between transmitter and receiver side. Additionally, when no data are sent, 4-byte idle commas are inserted to keep the LVDS link running and aligned.

III. Multiple AER Channels Multiplexing

For multiple AER channels multiplexing the approach shown in Fig. 3 is used. The top q bits of the 32-bit data word are used to encode k AER channels ($2^q \geq k$). Each AER channel TX-FIFO requests access to an encoder in the CH-MUX block. Once access is granted, communication goes through the CH-MUX in a similar way as was described in the previous Section for single-channel communications. The 4-byte control commas are replaced by a combination of one data byte plus a 3-byte control comma. The data byte is then used to encode AER channel number.

IV. Experimental Results

Fig. 4 shows a typical setup where the proposed bidirectional serial link has been used. The figure shows an AER retina sensor, connected by a parallel AER connector to the AER-Node board [5]. The retina communicates with the Spartan6 FPGA on the AER-Node board, which communicates through a SATA cable to one of the Spartan6 in a 48-chip SpiNNaker board[1] [8]. The SpiNNaker board receives events, processes them and sends the resulting event flow back to the AER-Node board through the same SATA wire. The AER-Node board sends the results through another parallel AER port to one USBAERmini2 board [13] which communicates through USB with a host computer to display the results in real time.

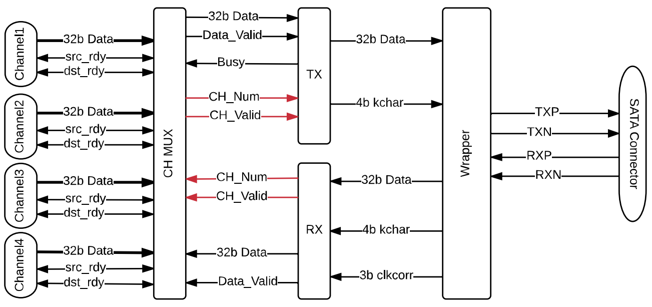

Fig. 3: Block Diagram of Multiple AER Channel Multiplexing Scheme

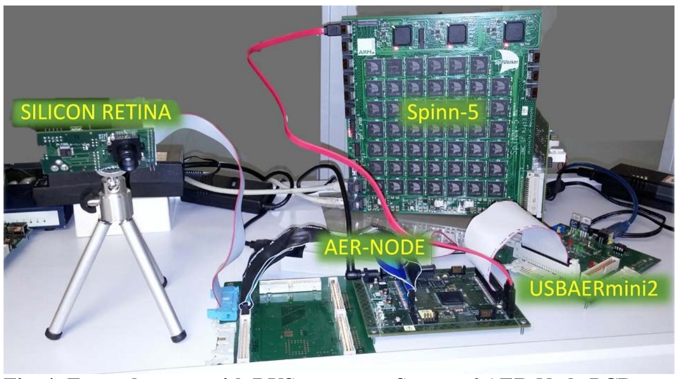

Fig. 4: Example setup with DVS camera, a Spartan6 AER-Node PCB, a 48-chip SpiNNaker PCB, and a USBAERmini2 computer communication PCB.

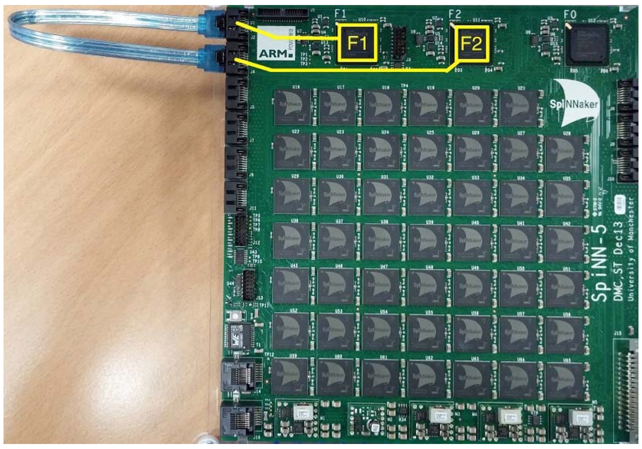

Fig. 5: Block Diagram of Experimental setup for testing link performance between two independent Spartan6 FPGA with separate clock signal oscillators in a 48-chip SpiNNaker PCB board

[1] The SpiNNaker Board FPGA communicates with the SpiNNaker chips through fast 2-of-7 predictive handshaking [11]. An improved version of it is now part of the official Spinn-Link release [12].

Experimental verification and characterization of the proposed communication scheme was performed using a pair of Spartan6 FPGAs located on a 48-chip SpiNNaker PCB. Each Spartan6 (XC6SLX45T-3) FPGA uses its own 150MHz Xtal oscillator. For testing the multiplexed link at maximum throughput, we used one GTP port of each FPGA. Each FPGA uses its local 75MHz reference clock. The setup was configured with 4 separate bidirectional AER Channels to be multiplexed over the SATA. For this, we used a pair of "Test Pattern Generator" (TPG) and "Test Pattern Checker" (TPC) inside FPGA and monitor them using the ChipScope analyzer tool from Xilinx. The TPG provides a known sequence of patterns, while the TPC checks and counts event errors in this sequence and computes the effective event rate received (excluding all control commas). In each FPGA, three TPG/TPC pairs where clocked with the same clock than the Transceiver/Multiplexing core discussed in Fig. 2 and Fig. 3. This is a clock at f = 75MHz derived from the external 150MHz Xtal reference oscillator. Therefore, each of these 3 synchronous TPGs can provide an event rate of up to 75Meps (one per clock cycle). The 4th TPG/TPC pair was clocked with an additional 67MHz to verify the performance of the synchronizers. LVDS line rate was set at 3.0Gbps. The setup was tested during 65 hours. None of the channels did detect a single error in the transmission. The link bandwidth (75Meps) was shared by the four Channels. The TPGs clocked at 75MHz try to deliver data at one event per clock cycle, but are slowed down by the corresponding encoder in CH-MUX block if SATA link bandwidth is reached. The total bandwidth shared among the channels was 74.93Meps (99.90% of link bandwidth). This means that the remaining 0.10% was used by commas.

Another test was designed to study the effect of clock correction comma insertion period on the link. In this test only one channel with full speed was used. Because errors happen in bursts, error ratio in this case is not important. It is important to know how long it will take to lose alignment. As can be seen in Table 1, clock correction insertion period with less than 1.7ms (at frequency of 75MHz, it will be one comma per 2^{17} events) is enough to not experience errors.

COMMA Period	Time to first error
Infinitive	138ms
56ms	138ms
28ms	196ms
14ms	247ms
7ms	340ms
3.5ms	325s
1.7ms	Infinitive

Table 1 Experimentally Measured time to first error by sweeping the clock correction comma period in TX block

V. Acknowledgements

This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement n. 320689, from the UK Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/G015740/1, EU FP7 grant 604102 "The Human Brain Project" (HBP), EU H2020 grants 644096 "ECOMODE" and 687299 "NEURAM3", by Samsung Advanced Institute of Technology grant NPP, by the National Centre of Research and Development of Republic of Poland (NCBiR) under grant PBS3/A8/134/2015, by Spanish grants from the Ministry of Economy and Competitivity TEC2012-37868-C04-01 (BIOSENSE), TEC2015-63884- C2-1-P (COGNET), and TEC2016-77785-P (COFNET) (with support from the European Regional Development Fund), and by Andalusian grant TIC-6091 (NANO-NEURO). ARY is supported by a Spanish FPI Scholarship from the Ministry of Economy and Competitivity.

References

[1] P. O. Pouliquen and A. G. Andreou, "Bit-Serial Address-Event Representation," *Proceedings of the 33rd Annual Conference on Information Sciences and Systems*, Baltimore MD, USA, March 1999.

[2] H. Berge and P. Häfliger, "High-Speed Serial AER on FPGA," *Proc. IEEE Int. Symp. on Circ. and Syst.* (ISCAS), pp. 857–860, May 2007.

[3] D. B. Fasnacht, et al., "A Serial Communication Infrastructure for Multi-Chip Address Event Systems," *Proc. IEEE Int. Symp. on Circ. and Syst.* (ISCAS), pp. 648–651, May 2008.

[4] C. Zamarreño-Ramos, et al., "Multi-Casting Mesh AER: A Scalable Assembly Approach for Reconfigurable Neuromorphic Structured AER Systems. Application to ConvNets," *IEEE Trans. Biomedical Circ. and Syst.*, vol. 7, no. 1, pp. 82–102, Feb. 2013.

[5] T. Iakymchuk, et al., "An AER Handshake-Less Modular Infrastructure PCB with x8 2.5Gbps LVDS Serial Links," *Proc. of the IEEE Int. Symp. on Circuits and Systems*, pp. 1556–1559, 2014.

[6] L. Plana, J. Heathcote, J. Pepper, S. Davidson, J. Garside, S. Temple, and S. Furber, "pI/O: A library of FPGA designs and reusable modules for I/O in SpiNNaker systems," *Available on-line at*: http://dx.doi.org/10.5281/zenodo.51476, 2014.

[7] L. A. Plana, S. B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, and S. Yang, "A GALS Infrastructure for a Massively Parallel Multiprocessor," *IEEE Design and Test of Computers*, pp. 454–463, Sep-Oct 2007.

[8] S. Furber, et al., "Overview of the SpiNNaker System Architecture," *IEEE Trans. Computers*, vol. 62, no. 12, pp. 2454–2467, 2012.

[9] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128x128 1.5% Contrast Sensitivity 0.9% FPN 3us Latency 4mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Amplifiers," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013.

[10] P. A. Franaszek and A. X. Widmer, "Byte Oriented DC Balanced (0,4) 8b/10b Partitioned Block Transmission Code," *US Patent* 4,486,738, Dec. 4, 1984.

[11] A. R. Yousefzadeh, et al., "Fast Predictive Handshaking in Synchronous FPGAs for Fully Asynchronous Multi-Symbol Chip Links. Application to SpiNNaker 2-of-7 Links," *IEEE Trans. on Circuits and Systems, Part II*, vol. 63, No. 8, pp. 763-767, Aug. 2016

[12] https://github.com/SpiNNakerManchester/spio/blob/master/modules/spinnaker_link/spio_spinnaker_link_receiver.v

[13] R. Serrano-Gotarredona, et al., "CAVIAR: A 45k Neuron, 5M Synapse, 12G Connect/s AER Hardware Sensory-Processing-Learning-Actuating System for High Speed Visual Object Recognition and Tracking," *IEEE Trans. Neural Networks*, vol. 20, no. 9, pp. 1417–1438, Sept. 2009.