A Serial Communication Infrastructure for Multi-Chip Address Event Systems

Daniel B. Fasnacht, Adrian M. Whatley, Giacomo Indiveri
Institute of Neuroinformatics, UZH – ETH Zurich, Switzerland

***Abstract*— In recent years there have been an increasing number of research groups that have begun to develop multi-chip address-event systems. The communication protocol used to transmit signals between these systems' components is based on the Address-Event Representation (AER). It is therefore important to have access to robust and reliable AER communication infrastructures for streamlining the systems' development and prototyping stages.**

We propose an AER communication infrastructure that can be easily interfaced to workstations or laptops during a prototyping phase, and that can be embedded into compact and low-cost systems in the application phase. The infrastructure proposed uses a novel serial AER interface with flow-control, overcomes many of the drawbacks observed with previous solutions, and can achieve event rates of up to 78.125MHz for 32bit AEs.

I. Introduction

In recent years a new class of distributed multi-chip neuromorphic systems have emerged, e.g. [1]–[4]. These systems are typically composed of one or more neuromorphic sensors (e.g. [5], [6]), of additional VLSI chips that implement general-purpose computational architectures, often based on networks of silicon neurons and synapses e.g. [7], and potentially of interfaces to robotic actuators for implementing real-time sensory-processing behaving systems.

A. The Address-Event Representation

Multiple research groups are developing a wide variety of multi-chip neuromorphic systems in parallel. The characteristic that all these systems have in common is the data representation and the communication protocol used. Each component in these systems can receive and transmit information using the *Address-Event Representation* (AER) [8], [9] communication protocol. In this representation, input and output signals are real-time digital events that carry analog information in their temporal relationships (inter-spike intervals). Each event is represented by a binary word encoding the address of the sending node. Output signals of sending elements are converted into streams of *Address-Events* (*e.g.* using pulse-frequency modulation in the case of silicon neurons), and multiplexed onto an asynchronous digital bus.

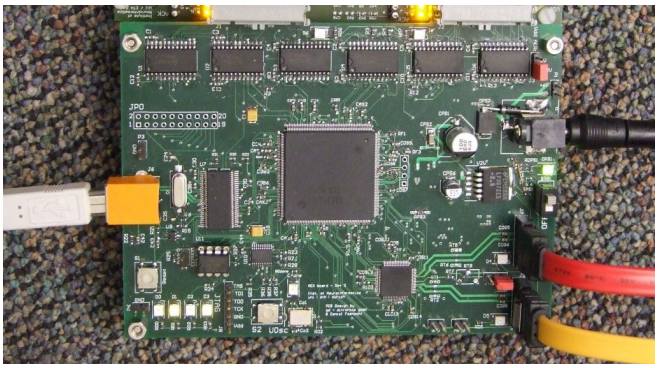

Fig. 1. The AEX Board

These multiplexing strategies are very efficient because only the addresses of active elements are transmitted (as opposed to conventional scanning techniques that allocate the same bandwidth for all the pixels, independent of their activity). The source address-events (AEs) being transmitted on the digital bus can be translated, converted or remapped to multiple destinations using conventional logic and memory elements. AER infrastructures therefore allow us to construct large multi-chip networks with arbitrary connectivity, and to seamlessly reconfigure the network topology.

As the trend to develop complex AER multi-chip experimental setups is increasing, there is a strong need for robust and reliable AER communication infrastructures, that can be easily interfaced to workstations or laptops during a prototyping phase, and that can be embedded into compact and low-cost systems in the application phase.

B. Existing AER Infrastructure and Approaches

Conventional approaches that use general purpose hardware in multi-chip AER systems involve logic-analyzers or general purpose digital data acquisition systems, but these approaches usually suffer drawbacks regarding asynchronous communication or on-line analysis of the acquired data [3]. This requires the design of special purpose hardware for building and debugging multi-chip AER systems.

For example a generic AER interfacing solution implemented using special purpose hardware is the PCI-AER board [3]. It consists of a custom made PCI card and a daughter board which are connected by a ribbon cable. The daughter board has parallel AER interface connectors and supports up to four input channels and four output channels. The PCI board consists of multiple FPGAs, FIFOs, SRAM and a PCI interface controller chip. The PCI board can monitor[0] incoming AE streams and then send the timestamped AEs via PCI to a program running on the computer. It can also do the reverse: sequence[0] timestamped data provided to it over the PCI bus out on any or all of the output channels. The FPGAs on the PCI-AER board also implement a one to many mapper that can be reconfigured via the PCI interface.

Recent boards for interfacing AER to PC were also implemented using USB instead of PCI, e.g. [10].

Similarly, recent serial AER communication schemes were proposed in [11].

Other groups building multi-chip AER systems tend not to use generic AER infrastructure, but build special purpose PCBs on a per project basis e.g. [2], [12], or analogous solutions that are not as flexible, or powerful, as the system described here.

Here we propose a general purpose serial AER infrastructure that can be reused in multiple projects or experimental setups.

[0]There are two different types of AEs. They can either have explicit timestamps attached, or the event-time can be implicit, simply when an address is communicated. Of course only timestamped AEs are suitable for packetized transmission or storage. Attaching a timestamp to an event is called *monitoring*, sending out an event timed according to its timestamp is called *sequencing*.

978-1-4244-1684-4/08/$25.00 ©2008 IEEE

II. Parallel versus Serial AER

With the speeds that AER chips and systems have recently reached, the parallel AER approach in board to board communication has become a limiting factor at the system level potentially causing unreliable behavior.

With the frequencies on parallel AER in the order of tens of megahertz, the wavelength of those frequencies has shrunk to about the order of magnitude of the lengths involved in our experimental setups, or only slightly larger.

One rule of thumb in electrical engineering says that if the signal wavelength is not at least one to two orders of magnitude greater than the physical size of the system, then the RF properties of the signals have to be taken into account: wires can no longer be assumed to be perfect conductances with the same potential at every point, but have to be treated as transmission lines.

If these problems are not taken into account, issues such as RF sensitivity, cross-talk and ground-bounce arrise, especially in parallel AER links using ribbon cables. These issues can best be solved by resorting to serial differential signaling.

A. General Trend towards Serial Differential Signaling

1) Single-Ended Signaling → Differential Signaling: The issues referred to above with the parallel approach have also played a major role in industrial and consumer electronics in general. The solution has been to use even faster, but differential links, and to carefully control the line impedance at every point between the sender and receiver.

In such a differential signaling scheme there is always a pair of wires that carry signals of opposite sense. The absolute value of the voltages on the signal wires does not have any meaning, only the voltage difference between the two wires of the pair has.

These so called differential pairs are then usually shielded, thus avoiding the problems of RF sensitivity and cross-talk to other signal wires.

Because of the differential signaling, the ground-bounce problem is also solved. A differential driver always pushes as much charge into one wire as it pulls from the other. Thus the net charge flow is always zero.

2) Parallel → Serial: The data rates that can be achieved using differential signaling are orders of magnitude higher than with traditional single-ended signaling. Therefore less (but better) wires are nowadays used to achieve the same or better bandwidth than with the many parallel wires in traditional bus links.

For example IDE / parallel ATA can achieve up to 1Gbit/s using 16 single-ended data signals, but only in one direction at a time (half-duplex) [13].

Serial ATA has 2 differential pairs (and thus four signal wires), one pair to send, and one to receive [14]. Each pair can transmit up to 3Gbit/s.

The AER communication infrastructure we implemented uses serial differential signaling for inter-board communication, following an approach similar to the one proposed in [11].

III. The *AEX* Board

The printed circuit board that implements the serial communication infrastructure, called the *AEX* board, is shown in Fig. 1. As can be seen on the AEX block diagram (Fig. 2) the board consists of three interface sections:

Parallel AER – for connecting the neuromorphic chips
USB2.0 – for Monitoring & Sequencing on a PC
Serial AER (SAER) – to interface to other AEX boards or other boards with a Serial AER interface

and an FPGA used to route data between those interfaces.

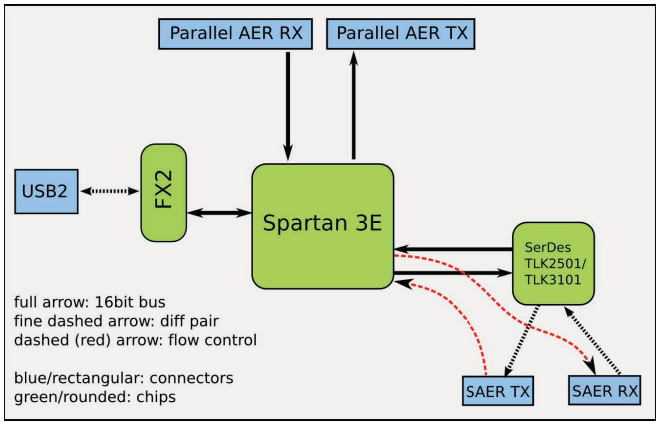

Fig. 2. AEX – Block Diagram

A. Parallel AER interface

With this interface many common parallel AER devices, i.e. neuromorphic chips, can be attached to the AEX board.

The connectors are designed in a way that allows the AEX board to be plugged directly to a chip carrier board, without the need for ribbon cables between the AEX and the target chip. This allows the parallel AER interface to be used at higher speeds.

B. USB2.0 interface

The USB2.0 interface is used to transfer timestamped AE data back and forth between the FPGA and a PC. USB2.0 was chosen for it's good performance and ubiquity (e.g. in laptops).

The interface uses the well known FX2 chip from Cypress. As the firmware programmable 8051 core of this chip does not need to manipulate the USB data-stream, the full bandwidth of the 480Mbit/s highspeed mode of the USB2.0 specification can be achieved.

We developed the firmware for the FX2 from scratch and compiled it using the open-source compiler [15].

We also developed a high-performance driver for Linux as part of this project with which we achieve actual bandwidths of 40MByte/s from AEX to software. The limiting factor here is the USB host-controller of the computer.

C. Serial AER interface

Our serial AER approach differs from previously proposed solutions (e.g. in [11]) in several aspects:

Instead of using a high-end FPGA natively supporting serial IO standards, we are using a low cost Xilinx Spartan FPGA plus a dedicated *SerDes*[1] chip. The usage of such a SerDes chip allows us to get higher event rates at significant lower silicon cost. The FPGA and SerDes we use cost about $40, a about third of the cost for the cheapest *Xilinx Virtex-II Pro* series FPGA necessary for implementing a system as in [11]. Using this hardware we currently achieve event rates that are about three to four times faster than in [11].

[1]Such a *Serializer-Deserializer* locally receives data on a parallel bus and then sends it over a serial output at a multiple of the parallel interface speed and vice versa for the serial receive path. The parallel interface is usually used for on-board, the serial for off-board communication.

In the approach described in [11], the receiver simply drops events if it is not ready to receive them. We implemented a *ow-control* scheme that ensures that all events reach its destination. In case the receiver is currently unable to receive an event because it does not have the necessary receive buffer space available, it can tell the sender to stop until space is available. The FPGA package type chosen allow for in-house assembly and repair as opposed to the ball-grid-array package used in [11].

1) SerDes - TI TLK2501 / TLK3101: The SerDes we can use on our system is either the TLK2501 or the TLK3101 from Texas Instruments. The TLK2501 supports up to 2.5Gbit/s, the TLK3101 supports up to 3.125Gbit/s, and has on-chip termination resistors. As terminating the differential traces correctly is not a trivial layout task, it is easier to achieve working PCB layouts with the TLK3101. Our system both supports the TLK2501 and the TLK3101 as an assembly option. We also successfully achieved mixed setups where TLK2501 and TLK3101 are communicating with each other at 2.5Gbit/s.

On the parallel side of the SerDes these chips have a 16bit transmit and a 16bit receive bus. They use 8bit/10bit coding and are also otherwise very similar to the Rocket IOs used in [11]. With the 16bit word length and the 8bit/10bit coding the SerDes parallel interfaces run at $^1/_{20}$ of the serial speed.

2) Cables & Connector Pin-Out: We are using Serial ATA connectors and cables to create Serial AER connections between our boards in multi-chip experimental setups. The connectors have seven pins, two differential pairs and three ground pins. With a SATA cable connecting boards A and B, we use the first differential pair of the cable to transmit serial AER data from the SerDes on A to the SerDes on B. The second differential pair is used to feed back a flow-control signal from the FPGA on B to the FPGA on A.

On the connector pins 2/3 are SerialAER+/-, pins 5/6 are FlowControl+/-. The remaining pins are the shielding, which we simply left unconnected on both sides, thus having a floating shield.

3) AC Coupling: We decided to used AC coupled instead of the simpler DC coupled serial links. With AC coupled links there is no common ground reference over all the boards in a system. This eliminates board-to-board ground-bounce problems, and also reduces line frequency injection.

4) Flow-Control Scheme: The flow control signal has to fulfill the following requirements:

it has to be transmitted over a differential pair;
for AC coupling it has to be DC free;
it has to represent two states, receiver busy or ready.

We chose the flow control signal to be a square-wave because it is DC free and can easily be generated by clocked digital logic. The part of the FPGA which interfaces to the SerDes and performs the flow control is running at the same clock-speed as the parallel SerDes interfaces, e.g. 125MHz for a 2.5Gbit/s link. The receiver FPGA signals that it is ready to receive by generating a square-wave at half its clock frequency, i.e. 62.5MHz. If the receiver is running out of FIFO space it signals the sender to stop by generating a square-wave at an eighth of the clock frequency.

These signals can be easily decoded by the sender FPGA even though they are not synchronous to any of the sender FPGA's clock signals. It does so by counting the number of clock cycles the flow control signal keeps the same value. If this counter is one to three the sender keeps sending, if it counts to four or more the sender has to stop.

We have to know at what receiver FIFO fill-level we have to signal a stop condition to the sender. It is the sum of the forward channel and the back channel latency. According to [16] the SerDes has a total link latency of $38 + 107 = 145$ bit times, giving 7.25 clock cycles, plus the line delay of the cable.

The flow-control back channel has a latency equal to the line delay plus two cycles for the synchronizer registers, plus 4 to 5 cycles to detect the stop state. This adds up to 14.25 cycles plus two line delays.

At a $2m$ maximum cable length this is $2 \times 2m \,/\, 0.5c = 26.6\,ns$ which is 3.3 cycles.[2] Thus the total delay should be less than 18 cycles. The latest time to dispatch the flow control stop signal is thus when we have 18 words of the 16bit receiver FIFO remaining free.

5) 32bit word synchronization: When using 32bit addresses, two 16bit words have to be transferred per address. In order to detect the 32bit word boundary we define that the two 16bit words have to be sent back-to-back, with no IDLE characters in between. Once an IDLE character is seen, the receiver knows the 32bit word boundary. This allows 32bit words to also be sent back-to-back, once the receiver has seen a single IDLE character, thus the full bandwidth available can be used for address data.

D. FPGA implementation

We are using a *Xilinx Spartan 3E* series FPGA on the AEX board to link the three interface sections together. The PQ208 package chosen has a sufficient pin count for this system, while still allowing in-house assembly without reflow soldering.

Fig. 3 shows the FPGA-internal block diagram. The three interfaces, serial AER, parallel AER and USB are drawn in orange. The USB interface, as opposed to the other interfaces, is handling explicitly timestamped addresses. Thus we need monitoring and sequencing units (green) between the two domains. The *routing fabric* between the three interface blocks allows AEs to be selectively routed between the three interfaces. It also contains simple mapping and filtering units.

The mapping units can add a configurable offset to an AE stream, so that different address spaces can be made non-overlapping. The filtering units allow to select which events are routed to which destination.

All these functional units are interconnected using FIFOs (blue, striped).

IV. On the Importance of Flow-Control

Here we compare the statistics of a Serial AER implementation with flow control and one that simply drops events.

With Flow-Control: Assume we have an event-consumer that can handle event rates up to $125MHz$. Thanks to the flow-control scheme, the consumer can block the producer as necessary. In this example we choose a fairly strict requirement that an event is delivered with a delay of more than $1\mu s$ at probability of less than 10^{-6}.

Given a Poisson distributed[3] producer, this means that the mean event rate of the producer can be up to 63.7% of the consumer event rate without violating our requirements.

Without Flow-Control: For comparison we assume a consumer that can handle event rates up to $125MHz$, but if two or more events arrive within an $8ns$ ($= 1/125MHz$) time-slot all except the first

[2] In this calculation the signal propagation speed for the SATA cables was assumed to be half the speed of light, a rather conservative estimate.

[3] A Poisson distribution is probably an unsuitable assumption when looking at a longer typical AE sequence. But what is critical is the performance in event bursts. We here take the Poisson distribution for looking at such bursts, typical for address event systems. The mean event rate should then be interpreted as the mean event rate in event bursts.

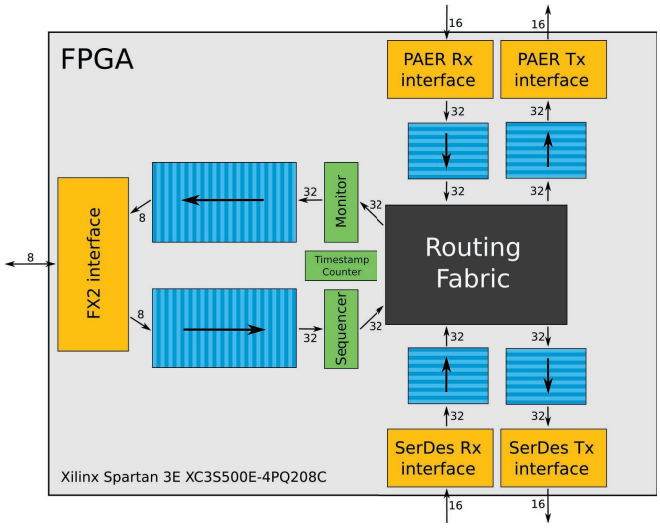

Fig. 3. AEX – FPGA block diagram

one are dropped. The probability that an event is dropped shall be no more than 10^{-3}. Under these circumstances a Poisson producer can then have a mean event rate of no more than 4.54% of the consumer rate.

Thus for our practical purposes flow-control gives us about one order of magnitude of actually usable event rate. In an experimental setup it also allows us to handle channel congestion either at the sender or the receiver side.

Further discussion of flow control in address event systems can be found in [9], [17], [18].

V. Results and Conclusion

We have developed an AER interfacing board part of a generic AER communication system suitable for building complex multi-chip AE based systems.

While using common parallel AER interfaces for connecting to the locally attached chip, we use a novel serial AER interface with flow-control. With this interface running at a bit clock of up to 3.125GHz we achieve event rates of up to 78.125MHz for 32bit AEs.

The parallel AER interface allows for event rates of up to 20 to 30MHz. This is in practice reduced by the signal propagation delays induced by the PCB traces and especially when used with ribbon cables.

For sending monitored AEs to a PC and reading AEs to be sequenced back from it we implemented a USB2.0 interface. Here we achieved bandwidths of 40MB/s, only limited by the USB host-controller on the computer itself. This allows for an event rate of 5MHz with 64bit timestamped AEs.

Given the filtering capabilities of the FPGA's routing fabric we can easily select parts of the address space we are interested in for monitoring, and because of the large buffers for monitored data on the FPGA itself we can compensate for the fact that the FPGA to PC interface is a lot slower than the parallel and serial AER interfaces.

The very high speeds of the serial AER interface allows us to have very low latency in serial AER links, and these links allow for the construction of very large multi-chip address event systems, e.g. by daisy-chaining multiple AEX boards.

Acknowledgment

Some of the ideas presented in this work were inspired by discussions held at the Telluride Neuromorphic Engineering Workshop. Particularly helpful suggestions were provided by V. Dante, ISS Italy, and A. Linares-Barranco, Universidad de Sevilla, Spain. We wish to also thank N. Felber, D-ITET ETHZ for supporting us regarding design aspects and U. Breu, C. Flaig, D. Flatz & A. Lehmann for proofreading. This work was supported in part by the EU grant DAISY (FP6-2005-015803).

References

[1] U. Mallik, R. J. Vogelstein, E. Culurciello, R. Etienne-Cummings, and G. Cauwenberghs, "A real-time spike-domain sensory information processing system," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 3, 2005, pp. 1919–1922.

[2] T. Y. W. Choi, P. A. Merolla, J. V. Arthur, K. A. Boahen, and B. E. Shi, "Neuromorphic implementation of orientation hypercolumns," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 6, pp. 1049–1060, 2005.

[3] E. Chicca, A. M. Whatley, V. Dante, P. Lichtsteiner, T. Delbrück, P. Del Giudice, R. J. Douglas, and G. Indiveri, "A multi-chip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity," *IEEE Transactions on Circuits and Systems I, Regular Papers*, vol. 5, no. 54, pp. 981–993, 2007.

[4] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, H. Kolle Riis, T. Delbrück, S. C. Liu, S. Zahnd, A. M. Whatley, R. J. Douglas, P. Häfliger, G. Jimenez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco, "AER building blocks for multi-layer multi-chip neuromorphic vision systems," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds., vol. 15. MIT Press, Dec 2005.

[5] P. Lichtsteiner, T. Delbrück, and C. Posch, "A 100dB dynamic range high-speed dual-line optical transient sensor with asynchronous readout," in *Proceedings of IEEE International Symposium on Circuits and Systems*. IEEE, 2006.

[6] V. Chan, A. van Schaik, and S.-C. Liu, "Spike response properties of an aer ear," in *Proceedings of IEEE International Symposium on Circuits and Systems*. IEEE, 2006.

[7] G. Indiveri and S. Fusi, "Spike-based learning in VLSI networks of integrate-and-fire neurons," in *Proc. IEEE International Symposium on Circuits and Systems, ISCAS 2007*, 2007, pp. 3371–3374.

[8] M. Mahowald, "VLSI analogs of neuronal visual processing: a synthesis of form and function," Ph.D. dissertation, Department of Computation and Neural Systems, California Institute of Technology, Pasadena, CA., 1992.

[9] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, May 2000, http://www.neuroengineering.upenn.edu/publications/-papers/kwabena/BoahenAER2.pdf.

[10] R. Berner, T. Delbruck, A. Civit-Balcells, and A. Linares-Barranco, "A 5 Meps $100 USB2.0 Address-Event Monitor-Sequencer interface," in *IEEE International Symposium on Circuits and Systems*, 2007.

[11] H. K. O. Berge and P. Häfliger, "High-Speed Serial AER on FPGA," in *IEEE International Symposium on Circuits and Systems*, 2007.

[12] P. A. Merolla, J. V. Arthur, B. E. Shi, and K. A. Boahen, "Expandable networks for neuromorphic chips," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 54, no. 2, pp. 301–311, Feb. 2007.

[13] "PATA - Parallel ATA - ATA/ATAPI," http://www.t10.org/t13/project/-d1532v3r4a-ATA-ATAPI-7.pdf.

[14] "SATA - Serial ATA," http://www.sata-io.org/.

[15] "SDCC - Small Device C Compiler," http://sdcc.sourceforge.net/.

[16] Texas Instruments, "Ti TLK3101 data sheet," http://www.ti.com/.

[17] A. Abusland, T. S. Lande, and M. Hovin, "A vlsi communication architecture for stochastically pulse-encoded analog signals," in *IEEE International Symposium on Circuits and Systems*, vol. III. IEEE Press, May 1996, pp. 401–404.

[18] L. M. Reyneri, "A performance analysis of pulse stream neural and fuzzy computing systems," *IEEE Trans. Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 42, no. 10, pp. 642–660, October 1995.