An Instant-Startup Jitter-Tolerant Manchester-Encoding Serializer/Deserializer Scheme for Event-Driven Bit-Serial LVDS Interchip AER Links

Carlos Zamarreño-Ramos, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco, *Fellow, IEEE*

***Abstract*—This paper presents a serializer/deserializer scheme for asynchronous address event representation (AER) bit-serial interchip communications. Each serial AER (sAER) link uses four wires: a micro strip pair for low voltage differential signaling (LVDS) and two handshaking lines. Each event is represented by a 32-bit word. Two extra preamble bits are used for alignment. Transmission clock is embedded in the data using Manchester encoding. As opposed to conventional LVDS links, the presented approach allows to stop physical communication between data events, so that no "comma" characters need to be transmitted during these pauses. As soon as a new event needs to be transmitted, the link recovers immediately thanks to a built-in control voltage memorization circuit. As a result, power consumption of the serializer and deserializer circuits is proportional to data event rate. The approach is also highly tolerant to clock jitter, due to the asynchronous nature and the Manchester encoding. A chip test prototype has been fabricated in standard 0.35 μm CMOS including a pair of Serializer and Deserializer circuits. Maximum measured event transmission rate is 15 Meps (mega events per second) for 32-bit events, with a maximum bit transmission speed of 670 Mbps (mega bits per second).**

***Index Terms*—Address event representation (AER), asynchronous circuits, asynchronous communications, clock data recovery (CDR), event-driven processing, low voltage differential signaling (LVDS), Manchester encoding, neuromorphic circuits and systems, serial AER, serial interchip communication.**

I. Introduction

ADDRESS EVENT representation (AER) is a well-established technology among neuromorphic engineers, proposed initially 20 years ago [1], [2]. AER exploits asynchronous principles [7], [8], and has been used fundamentally in vision (retina) sensors, for purposes such as simple light intensity to frequency transformations [12], [13], time-to-first-spike coding [14], [15], foveated sensors [16], spatial contrast [17]–[20], temporal contrast [12], [21]–[24], motion sensing and computation [5], and combined spatial and temporal contrast sensing [25], [26]. AER has also been used for auditory systems [3], [4], [27]–[29], competition and winner-takes-all networks [30], [31], and even for systems distributed over wireless networks [32]. But AER has also been employed for post-sensing event-driven processing, emulating biological cortical structures. Fixed-kernel [33], [34] and programmable-kernel [35]–[37] 2-D convolution chips have been reported, as well as reconfigurable multimodule AER processing systems [30], [38], [39]. The Appendix briefly explains the operation of a typical AER sensor or processor.

AER systems are clearly growing in complexity, and neuromorphic researchers are reporting work towards hierarchically structured multimodule AER systems [30], [38]–[42], using either a globally shared AER bus [30], [39], a global bus configured as a grid of physical links [43], many independent point-to-point plugged-in AER links between modules [38], or proposing multimodule multilink AER systems with local smart routing schemes [40], [41], [44]. This latter approach shows great potential for high degree of topological reconfigurability. The tendency seems to be towards individual AER modules, each with a relatively high number of AER links and with some local intelligence to reroute on-transit events, process incoming events, and send out newly produced events. Fig. 1 illustrates this philosophy. Fig. 1(a) shows a target PCB hosting 120 AER processing modules. Fig. 1(b) depicts an individual module communicating bidirectionally with other four neighboring modules, resulting in eight point-to-point links. This multichip assembly philosophy requires to minimize power and pin-count per module and link. For example, the solution proposed in the present paper uses 4 lines per link (a 2-line high-frequency differential micro strip and two lower frequency handshaking lines), and power consumption scales down with event rate.

Also, as overall system complexity grows, the amount of information to transmit per event also grows. In general, an event may carry information about local coordinates within a module, ID of sending and/or receiving module, as well as extra parameters or commands [40], [41], [44]. Thus, AER links should be capable to transmit large number of bits, and in practice it would be very desirable to have this number of bits per event configurable for each link. Present-day AER sensors transmit at the most 20 bit per event [3]–[6], [9]–[29]. For the example PCB in Fig. 1(a) we would need 7 extra bits to code an ID for each module, plus some extra bits for optional parameters. Consequently, a target event bit number of 32 seems quite realistic for a multimodule AER system. Most of reported AER sensor and processing chips use parallel AER interchip communication ports with a fixed number of bits. However, using parallel AER communication for a module as in Fig. 1 results in an excessive number of pins and a huge power consumption. An alternative solution is to transmit "*event subwords*" serially through

This work was supported by the EU under Grant 216777 (NABAB), Spanish grants (with support from the European Regional Development Fund) TEC2006-11730-C03-01 (SAMANTA2), TEC2009-106039-C04-01 (VULCANO), and Andalusian grant P06TIC01417 (Brain System). The work of C. Zamarreño-Ramos was supported by an FPU scholarship. This paper was recommended by Associate Editor V. Gaudet.

The authors are with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC). 41092 Sevilla, Spain (e-mail: bernabe@imse-cnm.csic.es).

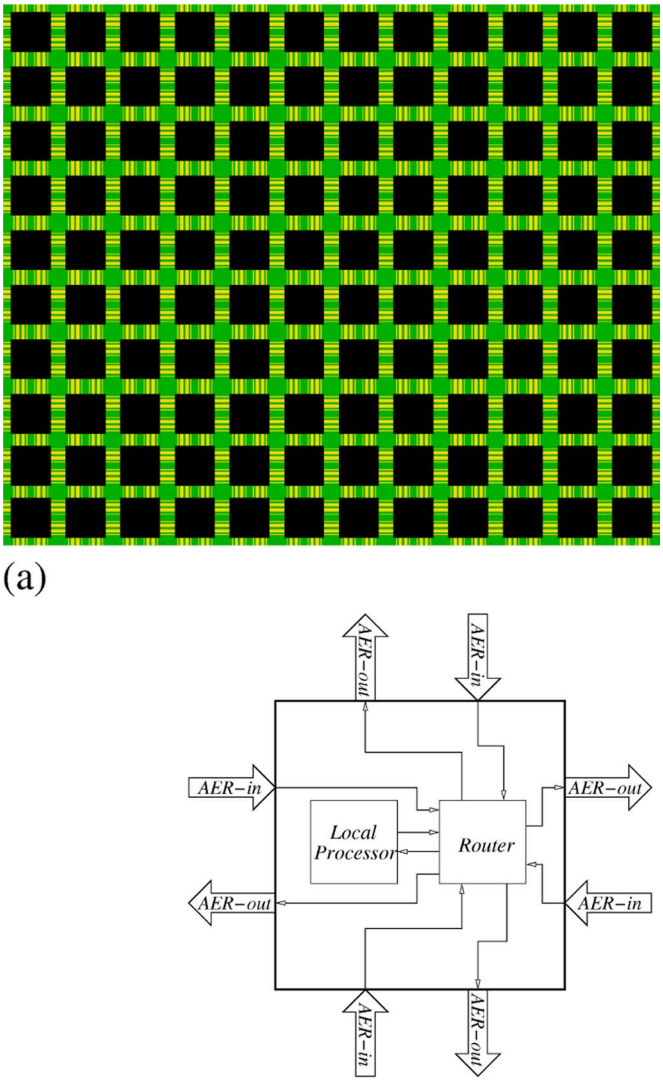


Fig. 1. (a) Example of multimodule multilink AER system PCB Concept. (b) Example AER module.

smaller parallel ports [35], [45]–[47], so that when consecutive events share some of the "*event subwords*" they are not transmitted again, saving time and power. One very efficient solution in terms of both speed and power is to use low voltage differential signaling (LVDS) [64], [65] technology and transmit event information in fully bit-serial format. LVDS allows for higher speed at a reasonable power budget. Although some researchers are combining chips that have traditional parallel AER ports with commercial serializer/deserializer custom chips [48]–[50], the ultimate solution needs to embed the parallel to-and-from serial conversion on-chip. This way, a VLSI AER module, as the one shown in Fig. 1, can provide reasonable pin count and affordable power consumption, making it possible to assemble many of them (like several hundreds) on a single PCB.

In AER processing systems activity tends to be sparse. In general, event rate tends to be maximum at the sensor output, but as processing is performed, the event rate at the output of each module tends to decrease as more processing stages are crossed [38], [42]. Reported AER vision sensors can provide peak rates reaching 10–20 Meps (mega events per second) [12], [19]–[24]. However, when observing life scenes, average event rates are in the order of 100 keps at the sensor output. In a multimodule AER (vision) processing system, maximum event rate is observed at the sensor or first processing stages, and decreases rapidly for subsequent stages [30], [36], [38], [39], [42]. Consequently, it is very sensible to develop serial AER links that can be turned off during event traffic pauses, and turn them on when an event needs to be transmitted. This way communication power consumption can be reduced proportionally to average event rate.

Commercial LVDS custom chips/FPGAs need uninterrupted data transmission as well as very low jitter clocks for proper clock data recovery (CDR) at the receiver. Here we propose a serializing/deserializing scheme for event-driven asynchronous AER links that can be turned off during interevent pauses [67], turn back on quickly, and does not require a low-jitter clock. This way, it results in a compact but robust circuit, very appropriate for embedding many of them in full custom VLSI low-cost AER processing modules, similar to the one shown in Fig. 1. A chip with a pair of full sender/receiver test prototypes has been fabricated and tested in a low-cost 0.35 μm CMOS process achieving peak rates of up to 600 Mbps. Section II reviews reported serial LVDS approaches compatible with event-driven data transmission and why we seek a new approach. Section III summarizes the developed serial link, while Section IV describes in detail the operation of the serial transmitter and Section V describes in detail the operation of the serial receiver. Section VI shows experimental results from the fabricated prototypes, and finally Section VII draws conclusions and points to future work.

II. Overview of Clock-Data-Recovery (CDR) Schemes

In general, commercial high-speed serial links require a continuous data flow between transmitter and receiver in order to keep the link synchronized. When there are no user data to be transmitted, an idle comma character is sent and the receiver uses it to identify a pause and discard the received data. In this case, architectures use a PLL or a DLL [51] to lock in frequency and phase with the transmission clock. If no data edges are received, the PLL cannot lock with the transmission clock, and data synchronization is lost. As a PLL or DLL is bandwidth limited for stability or jitter issues, a lost of synchronization leads to hundreds of clock cycles of recovery [52]. This extra latency introduced by the PLL relocking is not tolerable in event-driven asynchronous AER systems, or systems that require to transmit data in bursts and stop the transmission during data pauses.

There are several known solutions for burst-mode (or event-driven) CDR implementations [53]. They have a data-independent loop that tunes the transmission frequency, which keeps it tuned also in absence of data. The phase information is extracted from the data stream, but since there is no feedback with the frequency, proper phase alignment is achieved after a few clock cycles. There are three main types of burst-mode CDRs.

a) *Oversampling CDRs* [54]–[57]: a PLL generates several phases of the transmission clock and the input stream is sampled at these instants through a bank of multiphase samplers. A high-speed digital circuit chooses the received bit value analyzing all the samples. The major drawbacks of this architecture are that it requires many parallel blocks, each being very high frequency (faster than the transmission frequency), and the jitter budget

of the link is very stringent for keeping all the phases properly tuned.

b) *Gated-oscillator CDRs* [58], [59]: the synchronous clock is derived from a gated oscillator triggered from pulses generated from the data stream edges. These pulses cause the VCO to start its oscillation with the initial phase given by the data, not requiring any loop to tune the delay.

c) *High-Q Bandpass Filter CDRs* [60], [61]: the gated oscillator can be substituted by a high-quality factor bandpass filter that recovers the clock from the edges detected in the data stream.

In both solutions b) and c), there is no phase tracking between the receiving and transmitting clocks, requiring the use of low-jitter circuits to generate both frequencies with a very low offset between them. This phase shift also limits the maximum number of consecutive ones or zeros that can be transmitted without any data edge. Moreover, in b) a replica of the VCO for the frequency tuning is needed. This results in high sensitivity to process, temperature, voltage variations and mismatch. The edge pulse generation and the gated-oscillator are also very demanding because they must be much faster than the transmission frequency. In c) a monolithic integration is not possible due to the limitations of CMOS technologies implementing high-quality factor resonators.

Here we seek a low cost and simple solution for burst mode serial interfaces. The solution should overcome the limitations mentioned for existing burst-mode CDR architectures without increasing the system complexity or power consumption. The design goals for our target serial interface are:

- **Low complexity.** External components or huge on-chip devices such as inductors are not desirable, as we want to integrate several of them per chip.
- **Low latency.** After a data pause, the link must recover the transmission clock on the fly when a new event is received.
- **Jitter robustness.** The link will be part of noisy environments where a low-jitter clock and CDR implementations could be very demanding. If the solution is very robust to timing variations caused by jitter, the clock generation complexity, the CDR, and the system level design can be made more simple and with lower power budget.
- **Low power.** Reducing the power consumption, specially in pauses, of the serializer/deserializer circuits is essential to integrate many of them in the same AER system.
- **Arbitrarily long pauses capability.** AER data rates can vary several orders of magnitude. The link should be able to keep the system synchronized in all cases, allowing arbitrarily long pauses in the data stream.

Section III describes the proposed approach, which satisfies all these requirements. But first, let us compare with some reported VLSI CDR implementations, as well as with some non-VLSI serial AER realizations.

Table I compares the performance figures of several reported VLSI CDR circuits that have low acquisition times, making them potential candidates for event-driven AER systems. All of them are burst-mode CDRs, except the one with a broadband PLL [62], which we have included for comparison. Usually PLL based CDRs are slow and require long bit streams for locking. This particular design uses a broadband PLL making it faster, although it still requires 100 bits of acquisition time. The fastest ones present one bit equivalent delay of acquisition time, which means the first bit might be lost. On the contrary, the architecture presented in this paper does not loose the first bit, thus presenting an effective acquisition time of 0 bits. Transmission data rate is expressed in absolute values as well as normalized to the process cutoff frequency[1] f_t, for comparison. Similarly, circuit area is also given in absolute number and relative to λ (half of minimum feature size). Acquisition times and consecutive identical digits (CID) are expressed in equivalent bit times. As can be seen, for none of the reported circuits power scales down with data packet rate, and all of them have strong jitter requirements. As we will see later in experimental results, the proposed serializer/deserializer pair consumes about 73 mW at 10 Meps, which would scale down to 0.73 mW at an average event rate of 100 keps. Consequently, for the system in Fig. 1(a) with 120×4 Ser/Des pairs, power consumption would be about 350 mW @ 100 keps average link rate. However, for any of the solutions in Table I, power consumption of just 120×4 CDRs would range from 2.4 W to 261 W.

The jitter performance of the compared CDR implementations is shown in Table I. The jitter analysis has been divided in two parts. The first one (jitter tolerance) is related to the CDR immunity against the data pattern jitter. The second part (jitter reported) is related to the recovered clock jitter obtained for a safe data recovery. Our solution is immune to the recovered clock jitter because the clock is generated through edges present in the Manchester-encoded bit stream. Hence, data will be always sampled at proper instants. The CDR design allows for a 0.4 UI in terms of input clock tolerance. Only oversampling CDRs present larger input clock immunity at the cost of having a lot of low-jitter sampling phases for the high-speed data.

Other researchers have reported serial LVDS links developed specially for AER systems, but using either commercial serializer/deserializer circuits or FPGA built-in ones. Table II compares some of them against the VLSI ser/des pair reported in this paper. Serial bit rate ranges from 0.7 to 3.125 Gbps, with event sizes of 16, 24, or 32 bits. Table II also includes the "latency overhead" from the input of the serializer until the output of the deserializer, expressed in serial bit speed. Note that the total latency includes two components. The "data latency" to transmit the events bits serially at the available bit rate, plus the extra "overhead latency" (which includes, for example, error correction codes, framing and deframing, physical circuits delay, etc.). In general, commercial chips and FPGA built-in ser/des pairs add important overheads, resulting in significant extra latencies. Also, all of them require continuous uninterrupted serial transmission. This means they cannot be turned off and on quickly. Only one of them [48] allows for flow control, although it is implemented through an extra LVDS return path. All of them use embedded clock, except one [63] which requires an additional LVDS link for transmitting the clock (which is shared by 16 LVDS data links). This allows for burst mode operation and double data rate (DDR) achieving 1 Gbps bit rate with a 500 MHz clock. They also report an AER link between Virtex5 FPGAs using Rocket I/O, which is similar to the one by Berge [49] in Table II.

In summary, none of the reported LVDS links (either VLSI or non-VLSI) fulfills all of the requirements we need for a mul-

[1]None of the references in Table I gives the f_t of the technology used. This number has been estimated from other similar technologies.

TABLE I
LOW ACQUISITION TIME VLSI CDR CIRCUITS

	Gated Oscillator [58]	**Injection Locking [59]**	**Broadband PLL [62]**	**Blind Oversampling [56]**	**Semi-blind Oversampling [57]**	**Present Work**
Technology	250nm SiGe BiCMOS	90nm CMOS	180nm CMOS	90nm CMOS	110nm CMOS	350nm CMOS
Technology f_t	200 GHz	180 GHz	120 GHz	180 GHz	150 GHz	15 GHz
Bit Rate (absolute)	10.3 Gbps	5-10 Gbps	2.488 Gbps	2-3.5 Gbps	3.2 Gbps	0.67 Gbps
Bit Rate (relative to f_t)	0.052 f_t	0.028-0.055 f_t	0.021 f_t	0.011-0.019 f_t	0.023 f_t	0.045 f_t
Area (in mm^2)	3×3	0.55×0.55	1×1	0.13×0.13	0.44×0.34	0.9×0.38
Area (in λ^2)	24000×24000	12172×12172	11111×11111	2940×2949	8000×6182	5142×2171
Supply Voltage	3.3V (bipolar)/ 1.8V (CMOS)	1.2V	1.8V	1.2V	1.2V	3.3V
Power consumption	544mW	70mW	54mW	5mW@2.5Gbps	115mW	46mW@10Meps
Acquisition time	1 bit	8-15 bits	100 bits	1 bit	N/A	0 bit
CID tolerance	160 bits	N/A	infinite	infinite	N/A	infinite
Jitter Tolerance	0.27UIpp	N/A	N/A	93UI @ LF 0.68UI @ HF	200UI @ LF 0.4UI @ HF	0.4UI @ worst case
Jitter reported	$14.7ps_{pp}$/ $2.4ps_{rms}$ @ recov. clk	$15.5ps_{pp}$/ $2.2ps_{rms}$ @ recov. clk	-79dBc/Hz @ 1MHz	$142.2ps_{pp}$/ $25.94ps_{rms}$ @ recov. clk	N/A	Not affected by recov. clk jitter
Needs frequency tuning loop	Yes	No	Yes	Yes	No	No
Power scales with packet rate	No	No	No	No	No	Yes
Wide input frequency tolerance	No	No	No	No	Yes	Yes

TABLE II
SERIAL NON-VLSI AER REALIZATIONS

	Indiveri et al. [48]	**Miró et al. [50]**	**Hartmann et al. [63]**	**Berge et al. [49]**	**Present Work**
Chips	TI TLK3101	MAX9205/6 Spartan3	Virtex5 LX110T	Rocket I/O Virtex2 Pro	VLSI 350nm CMOS
Bit Rate	3.125 Gbps	0.792 Gbps	1 Gbps (DDR)	2.5 Gbps	0.67 Gbps
Event Size	32 bits	10 bits	24 bits (4 events per 128-bit packet)	16 bits	32 bits
Event Rate	78 Meps	66 Meps	31.25 Meps (per link)	41.66 Meps	15 Meps
Encoding	8B/10B	none	9 bit CRC	8B/10B	Manchester
Latency overhead	113 bits (w/o FC) 398 bits (with FC)	106 bits	N/A	304 bits	8 bits
Flow Control (FC)	Through extra LVDS return path	No	No	No	Yes
Burst-Mode	No	No	Yes	No	Yes
Extra clock path	No	No	Yes	No	No

tichip AER system with large number of chips. In the rest of the paper we describe a VLSI ser/des pair which does satisfy the requirements we seek.

III. HIGH-SPEED SERIAL AER LINK

Fig. 2 shows the block diagram of the proposed serial AER sender and receiver link. Parallel 32-bit input events $AERin$ are managed by the *Parallel to Serial AER Sender* system. This circuit receives the input request $reqIN$, generates the signals to latch the event data, and begins the serialization process. When the event data is captured, an acknowledge signal $ackIN$ is sent back to the parallel AER sender.

AER information is coded with the transmission clock using Manchester encoding [68]. This introduces extra edges at middle bit times. Manchester encoding reduces bit rate to half compared to nonreturn to zero encoding. However, it allows for very low cost, low power, jitter tolerant, and efficient CDR, thus being very suitable for asynchronous AER transmissions.

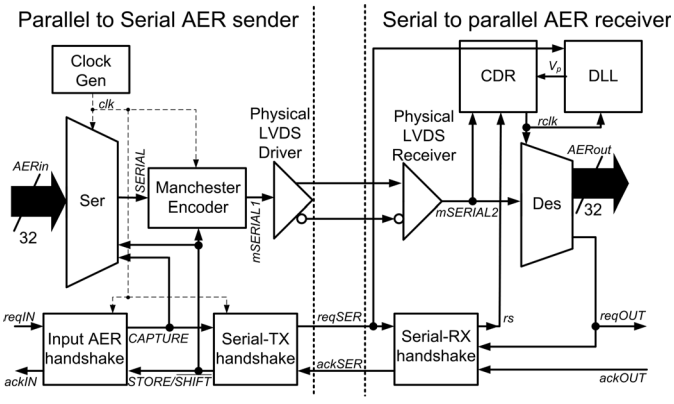


Fig. 2. Serial AER interface system level design.

For the "Physical Driver" and the "Physical Receiver" pads in Fig. 2 we use conventional LVDS [64] designs available for

our technology [66]. This signal format reduces the power consumption and filters the common mode noise coupled into the channel [65]. The standard specifies a 1.2 V common mode level and 350 mV of differential amplitude for the LVDS signals [64]–[66].

The *Serial to Parallel AER Receiver* system in Fig. 2 manages the serial stream of Manchester encoded data and converts it into the parallel output $AERout$ with its corresponding handshaking signals $reqOUT$ and $ackOUT$. Together with the serial transmission, a pair of request $reqSER$ and acknowledge $ackSER$ signals are used for flow control purposes. When a new event is ready to be transmitted serially, a request pulse $reqSER$ is sent to the receiver. If it is not busy processing a past event, it accepts the event by activating $ackSER$. In case of not being able to receive a new event, the acknowledge activation is delayed and transmission is temporarily stopped. Signal $reqSER$ is also used by the receiver to initialize all the blocks for receiving a new serial event.

The transmission clock is recovered by a CDR block designed to extract it from the Manchester encoded data. This circuit is in charge of generating a half data rate clock, which is used to decode the serial stream. For proper clock extraction in a Manchester encoding scheme, the receiver tunes a delay [68]. This parameter is very sensitive to temperature, supply voltage and process variations. For this reason, a DLL analyzes the extracted clock generating an analog voltage that controls the tunable delay elements. At steady state, this loop keeps the receiver synchronized with the transmission clock.

In Sections IV and V we describe in detail the operation of the transmitter and receiver circuits. Handshaking signals are active-low, while the rest of digital signals are active-high.

IV. Transmitter Design

A. Serializer

Parallel data are latched and serialized by the reconfigurable capture/store/shift register shown in Fig. 3. When a new parallel input event is received, data is latched in this register through the *Capture path*, by setting signal $CAPTURE$ high. After this, the serialization is activated during 34 clock cycles using a high-speed clock (ClockGen). Two preamble bits are added for alignment purposes, letting 32 bits for the AER address. In a shift register based solution, the maximum speed of operation is given by the delay of the critical path. In this case, this path corresponds to the delay of a flip-flop, plus the shift register branch propagation time, plus the set-up time of the next flip-flop. Therefore, this solution requires 34 high-speed flip-flops, but it can work at higher frequencies than multiplexer based solutions. It also has a fixed delay per bit when increasing the number of bits.

Fig. 4(a) shows the details of the "Input AER handshake" block. It uses a very simple finite state machine (genQ FSM), whose functional description is shown in Fig. 4(b). The FSM starts in the "IDLE" state, where output signal $CAPTURE$ is low. When $reqIN$ becomes active (low), the FSM switches to state "genQ," where $CAPTURE$ is set high. One clock cycle afterwards, the FSM switches to state "WAIT," setting $CAPTURE$ back low and waiting for the request to be deactivated. This way, an input request $(reqIN)$ activates signal $CAPTURE$ during one clock cycle.

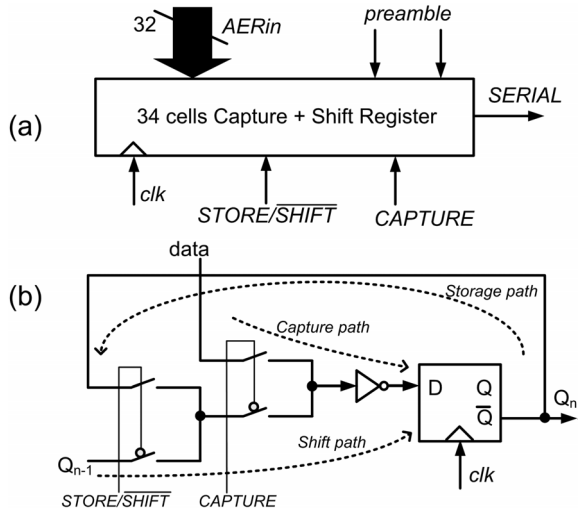


Fig. 3. "Ser" block. (a) Reconfigurable capture/store/shift register used to store and serialize parallel input data. (b) Details of one bit cell. If $CAPTURE$ is high, incoming parallel data bits are latched through the *Capture path*. If $CAPTURE$ is low, either data is stored for $STORE/\overline{SHIFT}$ high through the *Storage path*, or data is shifted for $STORE/\overline{SHIFT}$ low through the *Shift path*.

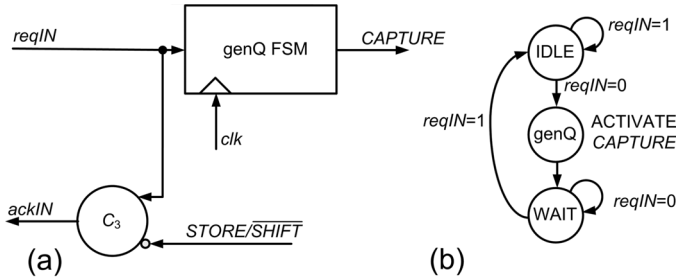
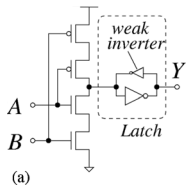


Fig. 4. (a) Details of 'Input AER handshake' block. (b) Functional description of the genQ FSM.

Fig. 6 shows the details of the "SerialTX handshake" block, which handles the serial handshaking as well as the store/shift operation of the parallel input event data. The serialization process begins with the $CAPTURE$ pulse generated by the FSM. C-element C_0 sends a $reqSER$ pulse to the receiver indicating that there is a new event to be transmitted. This signal is kept at low level until the acknowledge $ackSER$ coming from the receiver is received. At the same time, two cascaded C-elements C_1 and C_2 are used to generate an asynchronous signal a_1, which is activated when the $CAPTURE$ pulse is detected and the receiver chip has activated $ackSer$, indicating that the event can be transmitted. Signal a_1 is synchronized with the transmission clock using a flip-flop for the $STORE/\overline{SHIFT}$ signal generation. This signal is low during the serialization process. C-element C_3 [in Fig. 4(a)] is used to handshake the communication with the transmitting parallel block. C-elements (or Muller C-gates) [69] are commonly used in asynchronous logic circuits. Their output switches to "1" or "0" only when all inputs have switched to "1" or "0," respectively. Fig. 5(a) shows a possible circuit implementation of a 2-input C-element, and Fig. 5(b) its truth table description.

The 33-bit auxiliary shift register in Fig. 6 is used to calculate the $STORE/\overline{SHIFT}$ signal duration at low level. This register shifts a zero from the register input to the last position. Signal $endSER$ is at high level while the register is shifting.



A	B	Y
0	0	0
0	1	Y_{n-1}
1	0	Y_{n-1}
1	1	1

(b)

Fig. 5. C-element description. (a) Example implementation using latch based on weak inverter. (b) Truth table, where Y_{n-1} denotes "no change" condition.

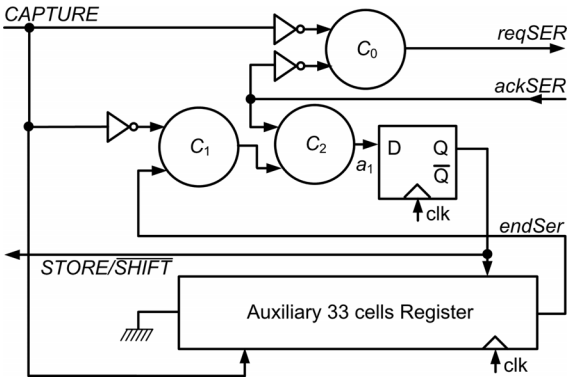


Fig. 6. Details of block "SerialTX handshake."

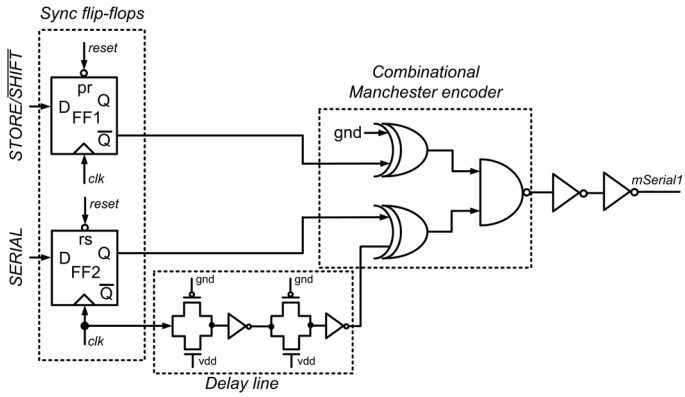


Fig. 7. Implementation of the high-speed Manchester encoder.

After 33 clock cycles, the zero reaches the output and the serialization process is finished, setting $endSer$ high. This implementation was preferred over a counter-based option (where the output signal is generated using the counter values) because the counter limits the serialization speed as the number of bits increases. In a shift register, there is no speed per bit penalty when increasing the number of bits and the number of bits per event is not limited by the physical implementation.

Note that the shift registers in Figs. 3 and 6 can be made of programmable length, so that the user could freely configure the number of bits per event. However, in the prototype presented in this paper both registers are of fixed length of 34 and 33 bits, respectively.

B. High-Speed Manchester Encoder

Fig. 7 presents the circuit that encodes the serializer output stream into a Manchester format. As the link operates at very high speed, an asynchronous solution is desirable in order to avoid the double rate clock generation. Such an implementation implies an XOR operation between the serial data flow and the transmission clock. A high-speed implementation of this combinational circuit requires a careful delay compensation.

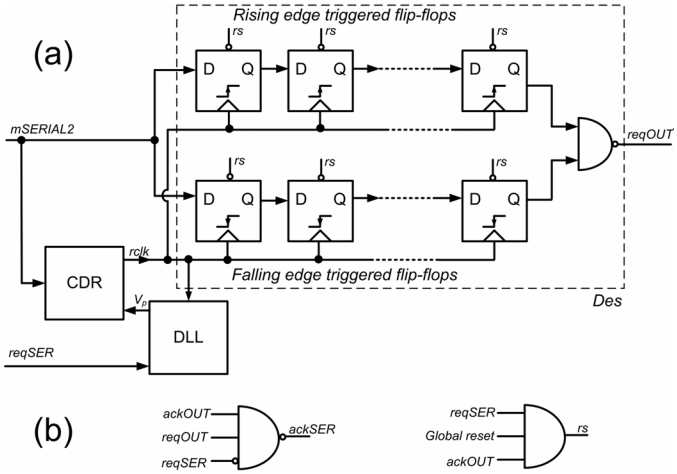


Fig. 8. (a) Details of deserialization block *Des* in Fig. 2. (b) Details of block *Serial-RX handshake* in Fig. 2.

Two flip-flops are used to synchronize data $SERIAL$ and signal $STORE/\overline{SHIFT}$. This latter one is used to frame the AER event in the output bursts, because it is active during all the serialization process. A replica of the flip-flop direct path is included to compensate the delay between the output and the clock. A dummy XOR gate is used to equalize the delay between the $SERIAL$ and the $STORE/\overline{SHIFT}$ paths. Finally, a NAND gate stops the encoding when the transmitter is not enabled.

V. Receiver Design

The deserializing scheme is shown in Fig. 8, where two shift registers are used: one triggered by the recovered clock $rclk$ rising edges and the other by the falling edges. This way, a half rate clock can be used to decode the input data, reducing power consumption and system complexity. The incoming bits *mSERIAL2* are shifted through the whole register until the two header bits reach a NAND gate. Then, signal $reqOUT$ goes low, starting a new handshaking cycle at the parallel AER output port. This signal is also used to latch the data in a capture register, waiting for this port to read the data. When $ackOUT$ is received, the registers are reset to zero and the parallel output request $reqOUT$ is deactivated. During this process, the receiver does not send any $ackSER$ pulse to the transmitter, suspending any new data transmission that could overwrite the current event.

A. Clock Extraction Circuit

Manchester data include additional edges that are used by the receiver to extract the synchronization information. The CDR must generate a recovered clock through these extra edges, ignoring bit data dependent edges. The circuit in Fig. 9 is able to extract this information and to generate a half rate clock for data recovery [68]. The circuit consists of a double edge triggered flip-flop (DETFF) configured as a frequency divider. The DETFF's clock input is directly connected to the serial input signal *mSERIAL2* provided by the "physical LVDS receiver" (see Fig. 2). As information edges of the Manchester signal

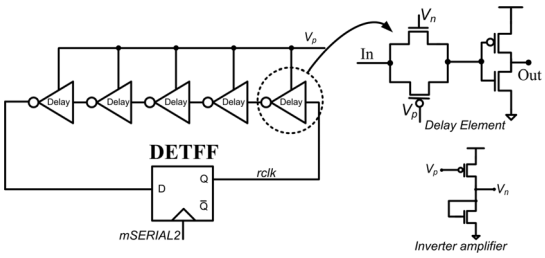


Fig. 9. Details of block CDR in Figs. 2 and 8.

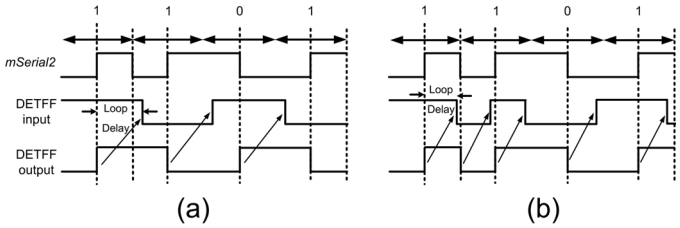


Fig. 10. CDR circuit chronogram in which (a) delays are properly tuned or (b) are out of lock.

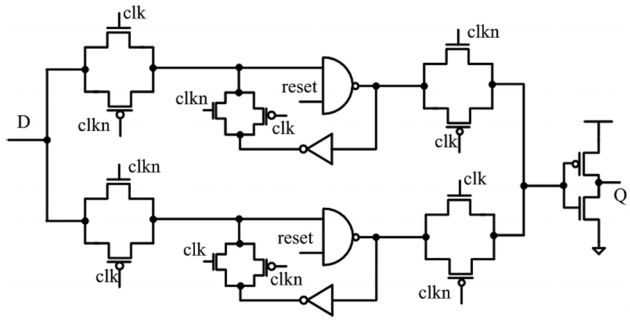


Fig. 11. Double edge triggered flip flop for clock extraction.

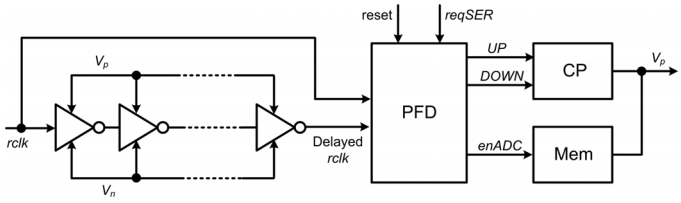


Fig. 12. Details of the delay locked loop (DLL) block in Figs. 2 and 8.

mSERIAL2 want to be filtered, a five inverter delay line is included between the DETFF input and output.

The delay value is critical for the CDR proper operation. Controlling the delay of an inverter in a CMOS technology is difficult due to process, temperature and supply voltage variations. For this reason, these delay elements have analog control voltages that tune their delay value. This delay can be controlled by adjusting the gate voltages of a CMOS switch at the inverter input. This implementation allows a very wide tuning range so that the CDR can extract the transmission clock in a large range of frequencies.

If this delay is grater than $T_b/2$ and lower than T_b (T_b is the bit time), data bit dependent edges will be filtered and the DETFF will not switch at them. However, if the delay is not properly tuned, the DETFF would switch at any *mSERIAL2* edge. Fig. 10 shows the CDR circuit chronogram for two different conditions of the analog control voltage. In Fig. 10(a) all delays are properly tuned and the CDR is extracting the recovered clock properly. In case (b) there is no lock in the delay tuned loop and extra edges of Manchester code are causing recovered clock triggering. If there are no incoming serial data, there are no clock edges, saving dynamic power during pauses. The value given for the jitter tolerance 0.4 UI in Table I allows the input clock to change its frequency in the CDR locking range during each bit. If the frequency changes slowly, the DLL will track it in the loop and larger frequency changes can de adapted. This is because the Manchester code provides extra data edges for phase comparison at every bit time. Therefore, 0.4 UI is a very pessimistic estimation for the tolerated input data jitter.

A DETFF [70], [71] clocked by the data stream generates the recovered clock edges. Hence, there will be a clock edge after a data edge if delays are properly tuned. As recovered clock edges are generated directly through data edges, the receiver is very robust against jitter. Even when the transmitter has a very poor jitter performance, that will not impact on the bit error rate because of this feature. That opens the door for very low cost clock generation solutions that do not require very power hungry clock generation circuits.

The clock is recovered thanks to a DETFF that needs to operate at very high speeds. The circuit in Fig. 11 is a high-speed implementation of a DETFF used in the burst-mode CDR. The flip-flop is composed by two latches, corresponding to the parallel branches in Fig. 11. Each path is high and low level sensitive, respectively. The output is combined in a shared node. Both branches have a reset signal that puts the flip-flops in a known state before starting the deserialization.

B. Delay Tuning Circuit

Fig. 12 shows the DLL circuit used to tune the delay in the CDR. This circuit compares the recovered clock signal $rclk$ with a one clock cycle delayed version, independently of the clock frequency, as long as it is within the delay elements tuning range. This delayed version is generated by 16 cascaded delay elements, identical to the five used in the CDR. A phase and frequency detector (PFD) compares the extracted clock and the delayed clock phases and generates two correction signals for a charge pump (CP). Signals UP and $DOWN$ make the analog control voltage V_p evolve to correct the phase lag between the delayed and recovered clocks. The link can tune the CDR for any frequency that belongs to the delay element tuning range.

Fig. 13 shows the schematics of the PFD used to compare the phase of the recovered and delayed clocks. Flip-flops FF1 and FF2 are used to detect the phase lag between both signals by triggering their outputs with every new edge. Rising edges cause that $DOWN$ and UP signals get activated and the combinational logic resets the flip-flops when both are at high level. The duration difference between both signals codes the existing phase shift. While the counter is counting, output signal $enADC$ is low. After counting 16 clock cycles $enADC$ is set high, activating the memorization circuit ("Mem" in Fig. 12) for control voltage V_p. When a new event is received ($reqSER$ goes low) the counter starts counting again, $enADC$ goes low again, the memorization circuit is deactivated, and the charge pump controls again voltage V_p.

If the DLL compares a 360° delayed signal with the reference clock, there is an initial clock edge that should not be compared. For this reason, the counter shown in Fig. 13 resets the PFD until

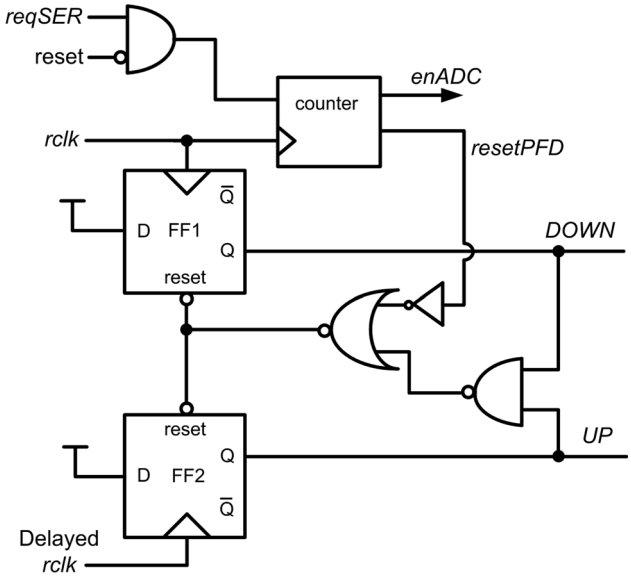


Fig. 13. Details of the PFD circuit used in Fig. 12.

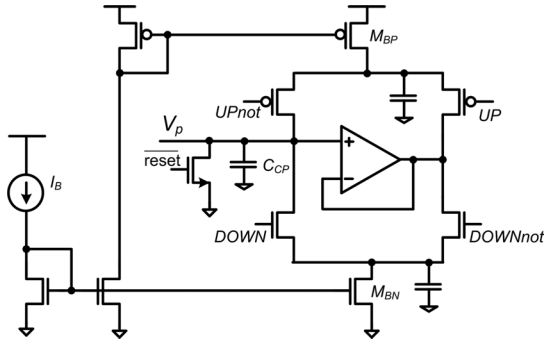


Fig. 14. High-speed charge pump design.

the first recovered clock is produced. Then, the PFD is enabled to compare the phase for a number of cycles given by the number of bits used to implement the counter. In this design, a 4-bit counter was chosen and 8 phase comparisons are performed for every AER event.

Fig. 14 shows the schematics of the charge pump that pulls up and down the analog control voltage V_p, depending on the information provided by the PFD. The CP integrates a bias current I_b on the on-chip capacitor $C_{\rm CP}$ (of value 1.5 pF) during a time slot given by the difference in the duration of UP and $DOWN$ pulses. A current steering topology was chosen [72] to reduce spurious current injected in the output capacitor when switches controlled by $DOWN$ and $UPnot$ are turned on and off. These currents can cause a phase offset in the DLL loop that can lead to a significant error in the delay tuning precision. This is particularly important in high-speed charge pumps, where very low phase shifts want to be tracked.

Bias current I_B is always flowing through current sources M_{BP} and M_{BN}, keeping these transistors saturated. This reduces the current peaks generated when current is switched from one branch to another. Capacitors were included at the current sources output nodes to help these bias currents keep their terminal voltages constant. An analog buffer was included to clamp both CP branches and maintain the same conditions in the two current paths. This way, both current paths are completely symmetric and current peaks are reduced to a minimum.

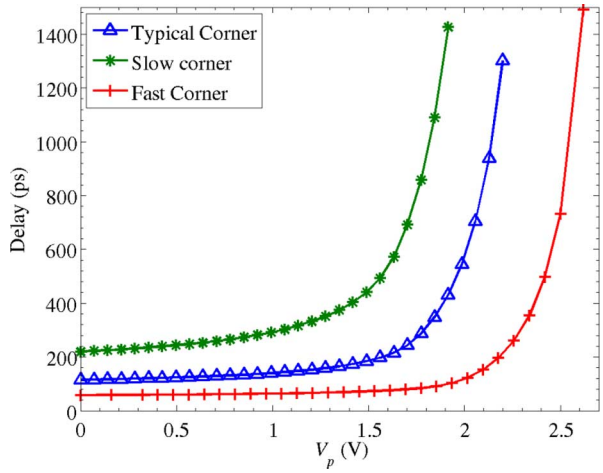


Fig. 15. Corner simulations of CDR "Delay-Element" delay as function of control voltage V_p.

The matching between NMOS and PMOS currents can also cause phase offset in the DLL loop. In this design, the current sources transistor lengths were carefully designed to achieve a good trade-off between speed and matching.

C. Control Voltage Memorization Circuit

Arbitrarily long event pauses can make the CP capacitor $C_{\rm CP}$ discharge through leakage currents in the switches and elements connected to it. A memorization circuit is required to retain this voltage during event pauses. A digital storage element is mandatory in order to guarantee an arbitrarily long memorization time. The synchronization is controlled by analog voltage V_p which must be properly interfaced for digital conversion and storage using adequate precision.

The circuit in Fig. 16 is used to store the delay tuning voltage V_p. Intensive simulations showed that a 5-bit analog to digital converter (ADC) architecture provides enough precision to keep the link synchronized between consecutive events. This can be understood with the help of Fig. 15, which shows the corner simulated delay in pico seconds of one "Delay Element" (see Fig. 9) as function of control voltage V_p. The tuning loop has to adjust the delay to 1/8 of bit time T_b [68] (i.e., 250 ps for 500 Mbps). The maximum V_p range is $[0\text{ V}, 2.7\text{ V}]$. Quantizing to 4-bit results in half an LSB of 84.4 mV, or 42.2 mV for 5-bit, resulting in phase errors of 55 ps and 20 ps, respectively, for the worst case scenario, e.g., around the 250 ps delay. On the other hand, the CDR delay has to satisfy [68] $5T_{\rm delay} > T_b/2$ and $5T_{\rm delay} < T_b$. Consequently, for the nominal case $T_b = 2$ ns this is 200 ps $< T_{\rm delay} <$ 400 ps. In the ideal situation the loop would lock to $T_{\rm delay} = T_b/8 = 250$ ps, so that a quantization error of up to 50 ps could ideally be tolerated (i.e., 5 bits). Nonetheless, we performed extensive corner and mismatch simulations of the complete circuit including all circuit nonidealities to make sure no extra bit was necessary to guarantee safer margins.

A flash architecture is used because of its simplicity and an asynchronous design is implemented to reduce its inherently high power consumption. The "Bank of 31 comparators" is used to determine the resistor ladder level V_{rj} that better approximates the analog control voltage V_p. The asynchronous digital controller takes this decision and controls the bank of switches,

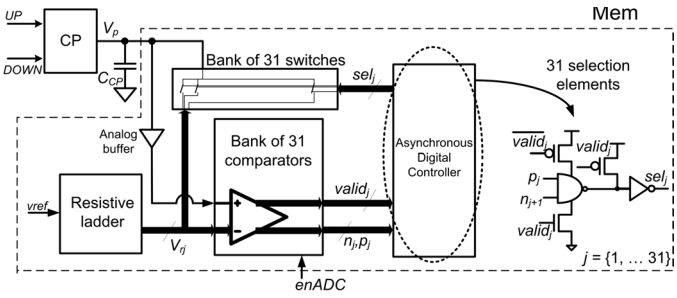


Fig. 16. Analog control voltage digital storage system.

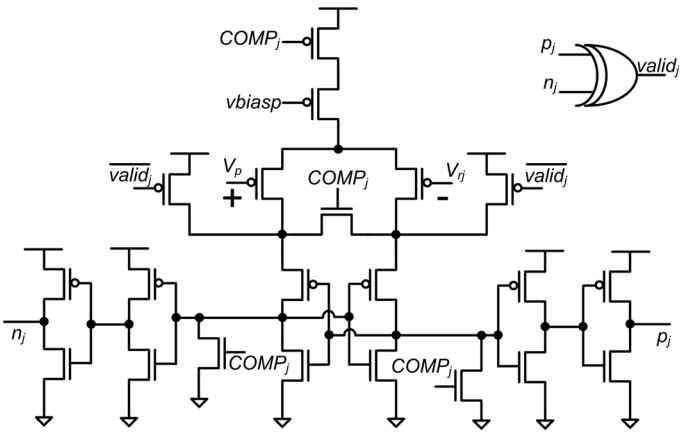


Fig. 17. Circuit schematics of comparator for asynchronous flash architecture.

connecting the chosen level with the output node. The asynchronous digital controller is composed of 31 selection elements that generate the enable signals sel_j for the bank of switches. They perform a logical operation between the outputs of consecutive comparators to find out the DAC output that makes a comparator to switch from a positive to a negative comparison. Each comparator provides an additional output signal $valid_j$ to notify that the comparison has settled. The operation of the asynchronous controller is enabled through signals $valid_j$ of the comparators.

Fig. 17 shows the schematics of the comparator used in the ADC [73]. It was designed as a two-stage comparator with a preamplifier, represented by a PMOS differential pair and a latch built with a two inverter loop, that stores the comparison result. Signal $COMP$ is used to reset the circuit before any comparison, precharging all the comparator nodes to known initial values. Moreover, the bias current is switched off to save power when no comparison is being carried out. When the comparison process finishes, an XOR gate sets signal $valid_j$ high to notify the digital controller that the comparison value is ready to be read.

One of the main drawbacks of flash ADC architectures is their high power consumption, since all the comparators operate in parallel when the device is converting. In our asynchronous implementation, the output bits are calculated sequentially and the comparators are switched off and on at every stage [74]. Fig. 18 shows the treelike connectivity of the comparators to implement this asynchronous conversion process. The tree has $log_2(n+1)$ levels, where $n = 31$ is the number of comparators. In our case there are 5 levels. Depending on the result at one level and the corresponding $valid_j$ signals, only one comparator is activated at each level. Although this is a slow process, the conversion rate

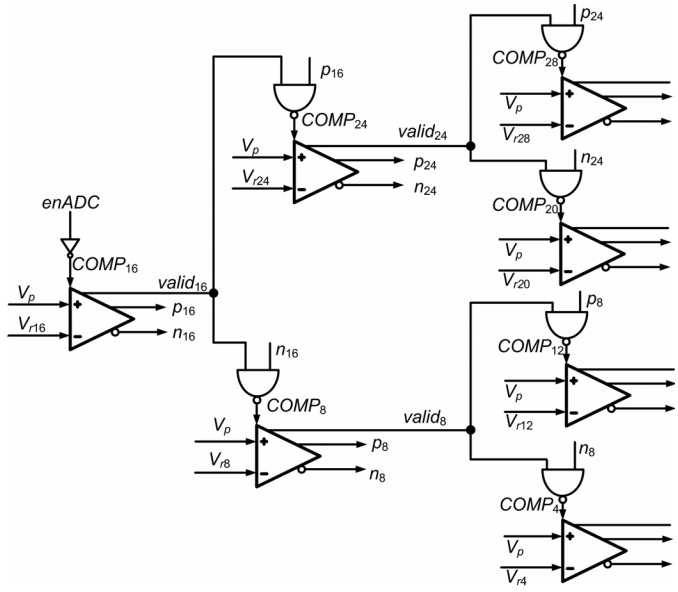


Fig. 18. Asynchronous low power flash architecture in bank of 31 comparators in Fig. 16.

is not a major concern in this application because the CP voltage variation in pauses is caused by leakage currents with a time constant in the order of milliseconds. Hence, besides the asynchronous implementation, currents of 10 μA are used to bias the comparators in order to further reduce the power consumption.

VI. Experimental Results

A proof of concept test prototype using the presented serial AER link has been implemented in a 0.35 μm CMOS technology. Nominal data rate was chosen to be 500 Mbps using Manchester encoding and supply voltage was 3.3 V. The transmitter requires an area of 350×375 μm^2 and the receiver occupies 900×380 μm^2, not including LVDS pads. Fig. 19 shows a microphotograph of the fabricated chip, highlighting the main components.

The test channel used in the experiments consists of a pair of PCB traces forming a 100 Ω differential micro strip line of 3 cm length. The clock was generated through a simple VCO based on a ring oscillator that can be externally tuned by an analog voltage. No PLL or DLL based solution to generate a jitter clean master clock reference was integrated. This is the worst situation [77] for clock jitter performance. However, it is useful for showing the circuit robustness to high jitter and demonstrate that a very simple clock solution is enough for this architecture.

Fig. 20 shows the test set-up. Two 16-bit USB-AER boards [76] receive events from a PC connected to them through USB ports. These events are sent through a parallel connector to the test board using the AER protocol. Each USB-AER board provides a 16-bit AER bus, but a 32-bit version is needed for the serial transmitter. For this reason, streams coming from two different boards must be synchronized to form up a 32-bit AER bus. A CPLD implements a C-element for $req1$ and $req2$ to generate $reqIN$. This way, both 16-bit parallel input streams are merged into a single 32-bit one which is serially transmitted. At the receiver side, the output flow must be split into two AER streams, each of which can be captured by a single 16-bit parallel USB-AER board. The CPLD performs this task in the same

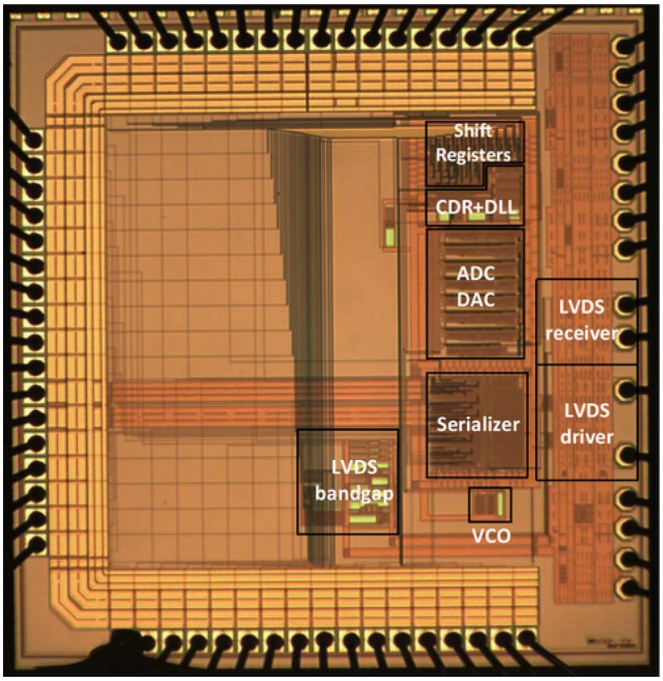


Fig. 19. Microphotograph of fabricated test prototype.

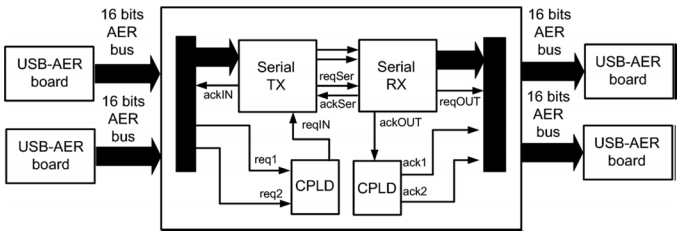


Fig. 20. Test setup to generate a 32-bit AER pattern.

way than in the transmitter side, managing $ack1$ and $ack2$. Each output USB-AER board is able to transmit captured events to the PC through its USB connection.

Fig. 21 shows the measured AER protocol signals a) at the transmitter parallel input, b) at the serial interchip link, and c) at the receiver parallel output, as well as the serial data stream. The CPLD generates $reqIN$ when a new 32-bit event is ready to be transmitted. When this event data is latched by the receiver, $ackIN$ is activated during all the serial transmission process. The delay between the $reqIN$ and $ackIN$ activations in Fig. 21(a) is 10 ns for a 550 MHz transmission clock. The $ackIN$ pulse duration is 62 ns, corresponding to 34 clock cycles. The transmitter cannot send a new event until this acknowledge signal is deactivated.

The transmitter uses the serial AER signals to perform a flow control mechanism. The receiver can send a new event if the transmitter activates $ackSER$ after $reqSER$. The delay between the request $reqSER$ activation and the acknowledge $ackSER$ activation is 2 ns. In this case, the protocol is completely asynchronous and only depends on the logic and pads delays. Handshaking is also implemented at the receiver output. This way, the serial AER transceiver can communicate with any parallel AER chip without any protocol conversion. The 21 ns delay between $reqOUT$ and $ackOUT$ is due to the USB-AER boards.

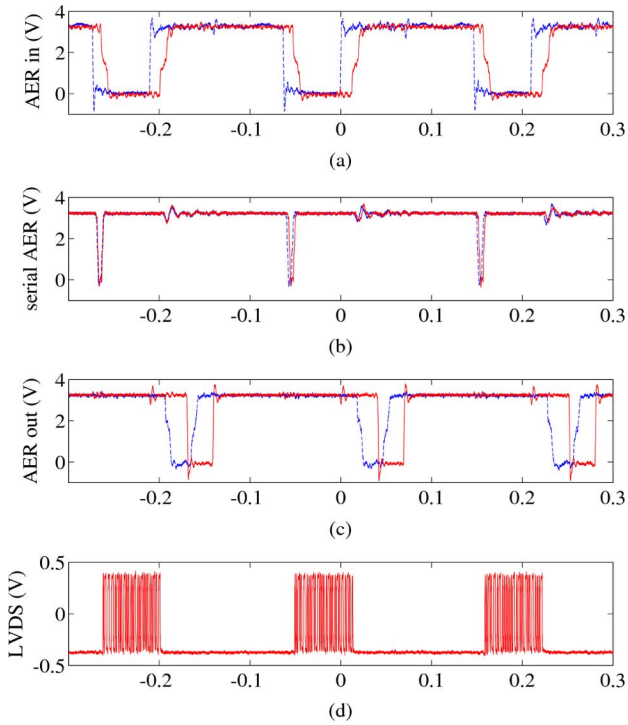


Fig. 21. AER protocol management at (a) $reqIN$ (dotted line) and $ackIN$ (continuous line) signals; (b) $reqSER$ (dotted line) and $ackSER$ (continuous line) signals; (c) $reqOUT$ (dotted line) and $ackOUT$ (continuous line) signals; (d) differential mode of the LVDS signal. Time scale is μs.

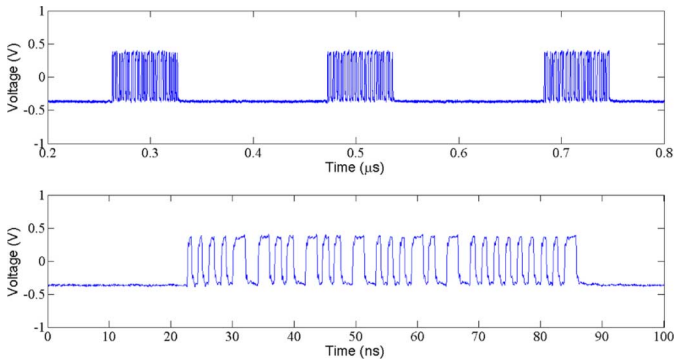


Fig. 22. Measured LVDS signals at receiver input.

Fig. 22 shows the high-speed serial AER signal measured at the receiver input. This figure illustrates the asynchronous event driven nature of the AER data sent through the link. Transmission frequency in this case is 500 Mbps and event rate is 4.8 Meps. Maximum event rate through the interface is limited by the transmission frequency and delays associated with the AER protocol signals. To measure this link event rate, $reqOUT$ and $ackOUT$ were shorted. Maximum event rate for the nominal 500 Mbps bit rate was measured to be 12.5 Meps, corresponding to an input-to-output request latency of 80 ns. If a 670 MHz transmission frequency is set, a 15 Meps and 66 ns latency can be achieved.

Fig. 23 shows the eye diagram measured for a 500 Mbps bit rate using an Agilent DSO81304B Infinium 12 GHz bandwidth oscilloscope. A 40 ps of rms jitter was measured and event error rate (EER) resulted to be below 3.3×10^{-10} (3.1×10^{9} events where analyzed without any error found). This demonstrates the

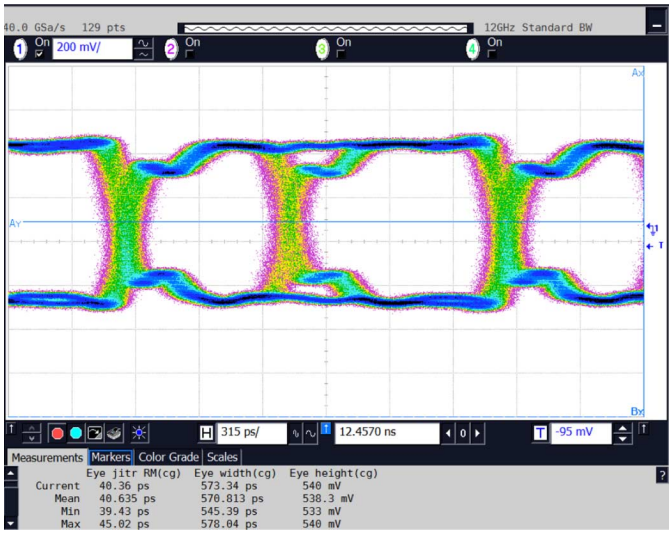


Fig. 23. Measured eye diagram at receiver input.

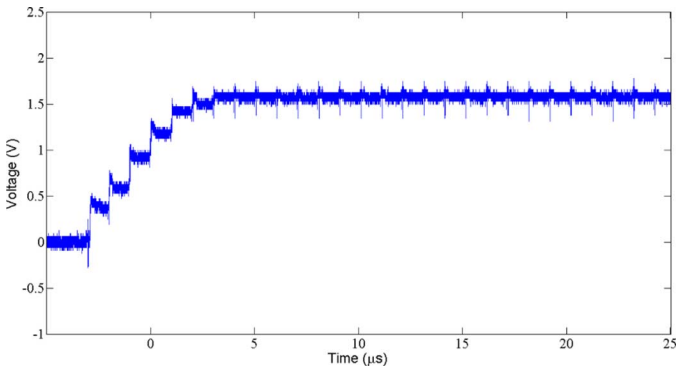


Fig. 24. Delay control voltage V_p convergence at power-up.

robustness of the approach, capable of keeping very low EER without very stringent jitter requirements.

Fig. 24 shows the control voltage V_p evolution when the link is powered up the first time. Voltage V_p is available to be measured through an on-chip buffer to not affect the CP integration capacitor. After reset, the control voltage is set to ground by a switch and evolves to its steady state value when events are received. The analog control voltage convergence process lasts 6 μs and requires 6 events in this particular case. In general, this delay depends on the input event rate, the input patterns, and the bias settings. When the DLL stabilizes at the steady state, this control voltage remains constant. The ADC block keeps it memorized, tolerating arbitrarily long event pauses. This was measured in the laboratory by stopping the interface and measuring the control voltage with an oscilloscope. It was noticed that it does not change during hours of pause.

The link can tune the delays to receive serial streams with a wide range of transmitting frequencies. Maximum frequency is limited by parasitic effects, not only by the delays tuning range. As the oscillator frequency can be programmed by controlling a test board potentiometer, the transmission data rate was swept in the range of 175 to 670 Mbps, in which the link operates correctly.

Transmitter and receiver power consumption depends on event rate. Fig. 25 shows the current consumption of the serializer and the deserializer for different event rates. As

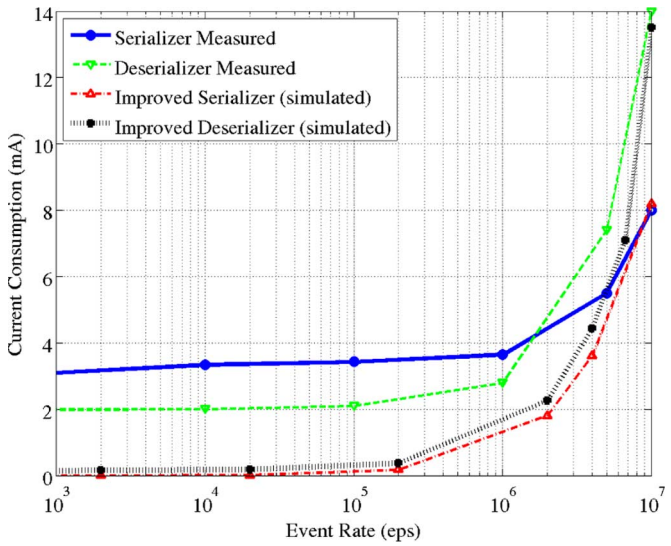


Fig. 25. Serializer and deserializer current consumption as a function of event rate.

can be seen, for high event rates, current consumption grows proportionally to data event rate. For lower event rates, current consumption stabilizes to constant values of 3.1 mA and 2.0 mA, respectively. Besides the serializer and deserializer circuits, other components consume a constant event-rate independent current. For example, the LVDS physical transmitter pad pair (driver) consumes 8.2 mA, while the LVDS physical receiver pad pair consumes between 8.4 mA (for low event rate) and 9.8 mA (for high event rate). The VCO used for the clock generator consumes 12.55 mA.

In this first proof-of-concept test prototype we were mainly concerned with a scheme for quickly turning on and off the ser/des pair. However, for a practical realization to be used in a system like in Fig. 1(a), all power hungry components need to be turned off and on quickly as well. For example, standby power consumption of the serializer can be drastically reduced by simply adding a clock-gating mechanism during pauses. Fig. 25 shows the simulated power consumption of an improved serializer with this simple addition. The fabricated deserializer also has an important stand-by power consumption because of the following three components: a) the small "inverter amplifier" in Fig. 9 which requires a high current to be fast; b) the "analog buffer" in Fig. 16 to buffer voltage V_p and isolate it from switching noise coming from the comparators; and c) the charge pump. Cases b) and c) are corrected easily by switching between ON and OFF bias currents. For case a) the inverter current cannot be switched because voltage V_n needs to be kept stable during pauses. However, this circuit can be resized for lower power while satisfying speed requirements. By introducing these modifications, we resimulated the deserializer and obtained the current consumption shown in Fig. 25, which shows a remaining standby current consumption of 150 μA.

Also, the flash ADC is an over engineered solution (although it simplified the overall design), resulting in an excessive area consumption (see Fig. 19). In a final implementation, since the ADC does not require high speed and it only has one comparator ON at a time, the flash ADC would be substituted by a successive approximation version with just one comparators.

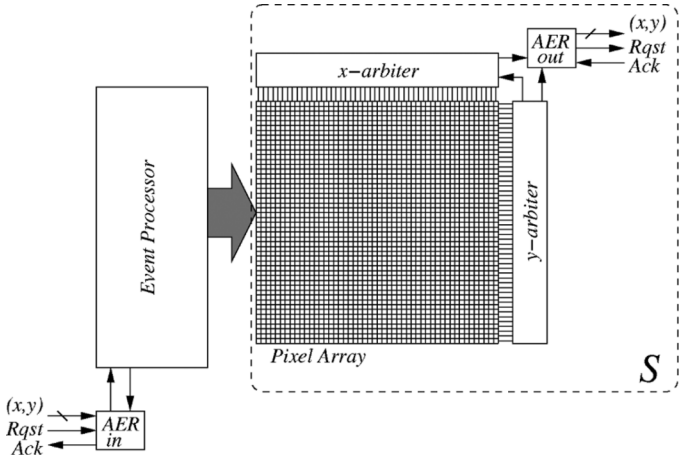


Fig. 26. Generic conceptual floorplan of AER module.

VII. Conclusions and Future Work

We have presented a serializer/deserializer design meant for serial links in AER systems, which are asynchronous and event-driven. The presented design does not require to keep the link active during absence of data. The receiver includes a means for memorizing the tuned state during data pauses. This way, when a new event is transmitted, the communication is reestablished without information lost. As a result, the power consumption of the serializer/deserializer circuit is proportional to data event rate. The proof of concept test prototype has been fabricated in 0.35 μm CMOS technology and is capable of achieving a maximum event transmission rate of 15 Meps with 32-bit events. The system is jitter tolerant and does not require a very low jitter clock. The proposed architecture uses simple components that do not require critical matching, jitter, nor supply voltages. Also, the achieved bit rate is as close to the technology cutoff frequency f_t as the highest performance CDR circuits in Table I. As a consequence of all this, this architecture should, in principle, be scalable to the most recent technology nodes.

Future work is oriented towards developing LVDS pads that can be turned off during data pauses and turned back quickly on [78], so that power consumption of the pads can also be made proportional to data event rate. Similarly, future work will also focus on developing a high-speed clock that can be also turned off during data pauses and quickly on.

Appendix

In this Appendix we briefly illustrate the operation of a typical AER sensor or processor chip. Fig. 26 shows a generic conceptual floor plan of an AER module for artificial vision type of applications. If the module is a sensor (artificial AER retina [10], [12]–[26]) it only includes the blocks comprised by box labeled "S" generating output AER events only. If it is an AER processor, it does both, receive input AER events and generate output events. In case of vision sensors, pixels include a photo sensor and in-pixel preprocessing circuits to compute, for example, spatial contrast, temporal derivative, motion, etc. Each pixel generates autonomously and asynchronously an output pulse (spike or event) when the result of its local computation exceeds a threshold. This pixel event is signaled to peripheral arbiters, which arbitrate among events produced by all pixels. The AER-out block labels each event (usually using the pixel (x, y) coordinate and an optional sign bit, depending on the in-pixel computation performed) and sends off-chip this multibit label using an asynchronous handshaking protocol with lines $Rqst$ and Ack.

In case of an AER (vision) processor [30], [31], [33]–[39], the module also receives input events through a similar asynchronous handshaking protocol. In this case, pixels do not sense external stimuli (such as light), but simply hold a pixel state. Input events are processed by an external processor, which then updates the state of one pixel in the array, a group of pixels, or all pixels. As a result of this pixel state update, pixels may generate new events, which are then arbitrated off-chip, as in the case of an AER sensor.

Acknowledgment

The authors are grateful to Andrea Boni for providing the LVDS physical pads.

References

[1] M. Sivilotti, "Wiring considerations in analog VLSI systems with application to field-programmable networks," Ph.D. dissertation, Comput. Neural Syst., California Inst. Technol., Pasadena, CA, 1991.

[2] M. A. Mahowald, "VLSI analogs of neuronal visual processing: A synthesis of form and function," Ph.D. dissertation, Comput. Neural Syst., California Inst. Technol., Pasadena, CA, 1992.

[3] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Silvilotti, and D. Gillespie, "Silicon auditory processors as computer peripherals," *IEEE Trans. Neural Netw.*, vol. 4, pp. 523–528, May 1993.

[4] G. Cauwenberghs, N. Kumar, W. Himmelbauer, and A. G. Andreou, "An analog VLSI chip with asynchronous interface for auditory feature extraction," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, pp. 600–606, May 1998.

[5] K. Boahen, "Retinomorphic chips that see quadruple images," in *Proc. Int. Conf. Microelectron. Neural, Fuzzy, Bio-Inspired Syst. (Microneuro99)*, Granada, Spain, 1999, pp. 12–20.

[6] K. Boahen, "A retinomorphic chip with parallel pathways: Encoding INCREASING, ON, DECREASING, and OFF visual signals," in *Int. J. Analog Integr. Circuits Signal Process.*, Feb. 2002, vol. 30, pp. 121–135.

[7] A. J. Martin and M. Nyström, "Asynchronous techniques for system-on-chip design," *Proc. IEEE*, vol. 94, no. 6, pp. 1089–1120, Jun. 2006.

[8] J. Sparsø and S. B. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*. Norwell, MA: Kluwer, 2001.

[9] A. Mortara, E. A. Vittoz, and P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE J. Solid-State Circuits*, vol. 30, no. 6, pp. 660–669, Jun. 1995.

[10] K. Boahen, "Retinomorphic vision systems," presented at the Microneuro'96: Fifth Int. Conf. Neural Netw. Fuzzy Syst., Lausanne, Switzerland, Feb. 1996.

[11] K. Boahen, "Point-to-Point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 416–434, May 2000.

[12] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.

[13] E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor," *IEEE J. Solid-State Circuits*, vol. 38, pp. 281–294, 2003.

[14] P. F. Ruedi *et al.*, "A 128 × 128, pixel 120-dB dynamic-range vision-sensor chip for image contrast and orientation extraction," *IEEE J. Solid-State Circuits*, vol. 38, pp. 2325–2333, 2003.

[15] S. Chen and A. Bermak, "Arbitrated time-to-first spike CMOS image sensor with on-chip histogram equalization," *IEEE Trans. Very Large Integr. Syst. (VLSI) Syst.*, vol. 15, no. 3, pp. 346–357, Mar. 2007.

[16] M. Azadmehr, J. Abrahamsen, and P. Häfliger, "A foveated AER imager chip," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 2005)*, Kobe, Japan, pp. 2751–2754.

[17] N. Massari *et al.*, "A 100 uW 64 × 128-Pixel contrast-based asynchronous binary vision sensor for wireless sensor networks," in *IEEE ISSCC Dig. Tech. Papers*, 2008, pp. 588–638.

[18] P. F. Ruedi *et al.*, "An SoC combining a 132 dB QVGA pixel array and a 32 b DSP/MCU processor for vision applications," in *IEEE ISSCC Dig. Tech. Papers*, 2009, pp. 46–47.

[19] J. Costas-Santos *et al.*, "A contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 7, pp. 1444–1458, Jul. 2007.

[20] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A five-decade dynamic-range ambient-light-independent calibrated signed-spatial-contrast AER retina with 0.1 ms latency and optional time-to-first-spike mode," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 10, pp. 2632–2643, Oct. 2010.

[21] U. Mallik *et al.*, "Temporal change threshold detection imager," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 362–363.

[22] C. Posch *et al.*, "A dual-line optical transient sensor with on-chip precision time-stamp generation," in *IEEE ISSCC Dig. Tech. Papers*, 2007, pp. 500–618.

[23] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 30 mW asynchronous vision sensor that responds to relative intensity change," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[24] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6 μs asynchronous frame-free event-driven dynamic-vision-sensor," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1443–1455, Jun. 2011.

[25] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip: Part 1," *IEEE Trans. Biomed. Eng.*, vol. 51, pp. 657–666, 2004.

[26] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip: Part 2," *IEEE Trans. Biomed Eng.*, vol. 51, pp. 667–675, 2004.

[27] R. Sarpeshkar *et al.*, "An analog bionic ear processor with zero-crossing detection," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 78–79.

[28] B. Wen and K. Boahen, "A 360-channel speech preprocessor that emulates the cochlear amplifier," in *IEEE ISSCC Dig. Tech. Papers*, 2006, pp. 556–557.

[29] V. Chan, S.-C. Liu, and A. van Schaik, "AER EAR: A matched silicon cochlea pair with address event representation interface," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, pp. 48–59, Jan. 2007.

[30] E. Chicca, A. M. Whatley, P. Lichtsteiner, V. Dante, T. Delbruck, P. D. Giudice, R. J. Douglas, and G. Indiveri, "A multichip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, pp. 981–993, May 2007.

[31] M. Oster, Y. Wang, R. Douglas, and S.-C. Liu, "Quantification of a spike-based winner-take-all VLSI network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 10, pp. 3160–3169, Nov. 2008.

[32] T. Teixeira, A. G. Andreou, and E. Culurciello, "Event-based imaging with active illumination in sensor networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 2005)*, pp. 644–647.

[33] P. Vernier, A. Mortara, X. Arreguit, and E. A. Vittoz, "An integrated cortical layer for orientation enhancement," *IEEE J. Solid-State Circuits*, vol. 32, no. 2, pp. 177–186, Feb. 1997.

[34] T. Y. W Choi, P. Merolla, J. Arthur, K. Boahen, and B. E. Shi, "Neuromorphic implementation of orientation hypercolumns," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 6, pp. 1049–1060, Jun. 2005.

[35] R. Serrano-Gotarredona *et al.*, "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 12, pp. 2548–2566, Dec. 2006.

[36] R. Serrano-Gotarredona *et al.*, "On real-time AER 2D convolutions hardware for neuromorphic spike based cortical processing," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1196–1219, Jul. 2008.

[37] L. Camuñas-Mesa, A. Acosta-Jiménez, C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 32 × 32 convolution processor chip for address event vision sensors with 155 ns event latency and 20 Meps throughput," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 4, pp. 777–790, Apr. 2011.

[38] R. Serrano-Gotarredona *et al.*, "CAVIAR: A 45 k neuron, 5 M synapse, 12 G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1417–1438, Sep. 2009.

[39] R. Vogelstein, U. Mallik, J. Vogelstein, and G. Cauwenberghs, "Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 253–265, Jan. 2007.

[40] M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. Furber, "Spinnaker: Mapping neural networks onto a massively-parallel chip multiprocessor," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN-WCCI)*, Jun. 2008, pp. 2849–2856.

[41] J. Fieres, J. Schemmel, and K. Meier, "Realizing biological spiking network models in a configurable wafer-scale hardware system," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN-WCCI)*, Jun. 2008, pp. 969–976.

[42] J. A. Pérez-Carrasco, B. Acha, C. Serrano, L. Camuñas-Mesa, T. Serrano-Gotarredona, and B. Linares-Barranco, "Fast vision through frame-less event-based sensing and convolutional processing. Application to texture recognition," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 609–620, Apr. 2010.

[43] P. Merolla, J. Arthur, B. E. Shi, and K. Boahen, "Expandable networks for neuromorphic chips," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 2, pp. 301–311, Feb. 2007.

[44] S. Joshi, S. Deiss, M. Arnold, J. Park, T. Yu, and G. Cauwenberghs, "Scalable event routing in hierarchical neural array architecture with global synaptic connectivity," in *Proc. 12th Int. Workshop Cellular Nanoscale Netw. and Their Appl. (CNNA)*, Berkeley, CA, Feb. 2010.

[45] K. Boahen, "A burst-mode word-serial address-event link-I,II,III," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1269–1300, Jul. 2004.

[46] J. V. Arthur and K. Boahen, "Synchrony in silicon: The gamma rhythm," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1815–1825, 2007.

[47] B. Wen and K. Boahen, "A silicon cochlea with active coupling," *IEEE Trans. Biomed. Circuits Syst.*, vol. 3, no. 6, pp. 444–455, Dec. 2009.

[48] D. B. Fasnacht, A. M. Whatley, and G. Indiveri, "A serial communication infrastructure for multi-chip address event systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 648–651.

[49] H. K. O. Berge and P. Häfliger, "High-speed serial AER on FPGA," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 857–860.

[50] L. Miró-Amarante, A. Jiménez-Fernández, A. Linares-Barranco, F. Gómez-Rodríguez, R. Paz, G. Jiménez, A. Civit, and R. Serrano-Gotarredona, "LVDS serial AER link performance," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 1537–1540.

[51] B. Razavi, *Design of Integrated Circuits for Optical Communications*. New York: McGraw-Hill, 2003.

[52] B. Razavi, "Challenges in the design high-speed clock and data recovery circuits," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 94–101, Aug. 2002.

[53] M.-T. Hsieh and G. Sobelman, "Architectures for multi-gigabit wire-linked clock and data recovery," *IEEE Circuits Syst. Mag.*, vol. 8, no. 4, pp. 45–57, 2008.

[54] J. Kim and D.-K. Jeong, "Multi-gigabit-rate clock and data recovery based on blind oversampling," *IEEE Commun. Mag.*, vol. 41, no. 12, pp. 68–74, Dec. 2003.

[55] S. I. Ahmed and T. A. Kwasniewski, "Overview of oversampling clock and data recovery circuits," in *Proc. Can. Conf. Electr. Comput. Eng.*, May 2005, pp. 1876–1881.

[56] Q. Du, J. Zhuang, and T. Kwasniewski, "A low-power, fast acquisition, data recovery circuit with digital threshold decision for SFI-5 application," *IEEE Trans. Very Large Integr. Syst. (VLSI) Syst.*, vol. 17, no. 12, pp. 1742–1748, Dec. 2009.

[57] M. van Ierssel, A. Sheikholeslami, H. Tamura, and W. W. Walker, "A 3.2 Gb/s CDR using semi-blind oversampling to achieve high jitter tolerance," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2224–2234, Oct. 2007.

[58] J. Terada, K. Nishimura, S. Kimura, H. Katsurai, N. Yoshimoto, and Y. Ohtomo, "A 10.3 Gb/s burst-mode CDR using a $\Delta\Sigma$ DAC," *IEEE J. Solid-State Circuits*, vol. 43, no. 12, pp. 2921–2928, Dec. 2008.

[59] M. Hossain and A. C. Carusone, "5–10 Gb/s 70 mW burst mode AC coupled receiver in 90-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 45, no. 3, pp. 524–537, Mar. 2010.

[60] K. Yamashita, M. Nakata, N. Kamogawa, O. Yumoto, and H. Kodera, "Compact-same-size 52- and 156 Mb/s SDH optical transceiver modules," *IEEE/OSA J. Lightw. Technol.*, vol. 12, no. 9, pp. 1607–1615, Sep. 1994.

[61] I. Radovanovic, W. van Etten, and H. Freriks, "Ethernet-based passive optical local-area networks for fiber-to-the-desk application," *IEEE/OSA J. Lightw. Technology*, vol. 21, no. 11, pp. 2534–2545, Nov. 2003.

[62] A. Li, J. Faucher, and D. V. Plant, "Burst-mode clock and data recovery in optical multiaccess networks using broad-band PLLs," *IEEE Photon. Technol. Lett.*, vol. 18, no. 1, pp. 73–75, Jan. 2006.

[63] S. Hartmann, S. Schiefer, S. Scholze, J. Partzsch, C. Mayr, S. Henker, and R. Schüffny, "Highly integrated packet-based AER communication infrastructure with •Gevents/s throughput," in *Proc. IEEE Int. Conf. Electr., Circuits, Syst. (ICECS)*, 2010, pp. 950–953.

[64] "Electrical characteristics of low voltage differential signalling (LVDS) interface circuits," Telecommunications Industry Association, ANSI/ TEIA/EIA-644-1995, 1995.

[65] *LVDS Owner's Manual*, 4th ed. Santa Clara, CA: Natl. Semicond., 2008 [Online]. Available: http://www.national.com/assets/en/app-notes/National_LVDS_Owners_Manual_4th_Edition_2008.pdf

[66] A. Boni, A. Pierazzi, and D. Vecchi, "LVDS I/O interface for Gp/s-per-pin operation in 0.35 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 36, pp. 706–711, Apr. 2001.

[67] C. Zamarreño-Ramos, R. Serrano-Gotarredona, T. Serrano-Gotarredona, and B. Linares-Barranco, "LVDS interface for AER links with burst mode operation capability," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2008, pp. 644–647.

[68] P. Popescu, A. Solheim, and M. Wight, "Experimental monolithic high speed transceiver for Manchester encoded data," in *Proc. Bipolar/CMOS Circuits Technol. Meet.*, Oct. 1995, pp. 110–113.

[69] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Proc. Int. Symp. Theory Switch.*, 1959, pp. 204–243.

[70] S. M. Mishra, S. S. Rofail, and K. S. Yeo, "Design of high performance double edge-triggered flip-flops," *IEE Proc. Circuits, Devices, Syst.,* , vol. 147, no. 5, pp. 283–290, Oct. 2000.

[71] T. L. W. Chung and M. Sachdev, "A comparative analysis of low-power low-voltage dual-edge-triggered flip-flops," *IEEE Trans. Very Large Integr. Syst. (VLSI) Syst.*, vol. 10, no. 6, pp. 913–918, Dec. 2002.

[72] S. Cheng, H. Tong, J. Silva-Martinez, and A. I. Karsilayan, "Design and analysis of an ultrahigh-speed glitch-free fully differential charge pump with minimum output current variation and accurate matching," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 9, pp. 843–847, Sep. 2006.

[73] J. Craninckx and G. Van der Plas, "A 65 fJ/conversion-step 0-to-50 MS/s 0-to-0.7 mW 9 b charge-sharing SAR ADC in 90 nm digital CMOS," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2007, pp. 246–600.

[74] G. Van der Plas and B. Verbruggen, "A 150 ms/s 133 W 7 bit ADC in 90 nm digital CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 12, pp. 2631–2640, Dec. 2008.

[75] Y.-Z. Lin, S.-J. Chang, Y.-T. Liu, C.-C. Liu, and G.-Y. Huang, "A 5 b 800 MS/s 2 mW asynchronous binary-search ADC in 65 nm CMOS," *IEEE ISSCC Dig. Tech. Papers*, pp. 80–81, Feb. 2009.

[76] F. Gómez-Rodríguez, R. Paz-Vicente, A. Linares-Barranco, M. Rivas, L. Miró, S. Vicente, G. Jiménez, and A. Civit, "AER tools for communications and debugging," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 3253–3256.

[77] A. A. Abidi, "Phase noise and jitter in CMOS ring oscillators," *IEEE J. Solid-State Circuits*, vol. 41, no. 8, pp. 1803–1816, Aug. 2006.

[78] C. Zamarreño-Ramos, T. Serrano-Gotarredona, B. Linares-Barranco, R. Kulkarni, and J. Silva-Martínez, "Voltage mode driver for low power transmission of high speed serial AER links," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Rio de Janeiro, Brazil, May 2011, pp. 2433–2436.