INTEGRATION, the VLSI journal 45 (2012) 61–75

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

A 32 GBit/s communication SoC for a waferscale neuromorphic system

Stefan Scholze, Holger Eisenreich, Sebastian Höppner, Georg Ellguth, Stephan Henker, Mario Ander, Stefan Hänzsche, Johannes Partzsch, Christian Mayr *, René Schüffny

Endowed Chair of Highly-Parallel VLSI-Systems and Neural Microelectronics, Institute of Circuits and Systems, Faculty of Electrical Engineering and Information Technology, University of Technology Dresden, 01062 Dresden, Germany

ARTICLE INFO

Article history:
Received 7 December 2010
Received in revised form
24 May 2011
Accepted 24 May 2011
Available online 1 June 2011

Keywords:
Gigaevent packet-based AER
Configuration over AER
Low-voltage-differential-signaling
Serial data transmission
Clock-to-data alignment

ABSTRACT

State-of-the-art large-scale neuromorphic systems require a sophisticated, high-bandwidth communication infrastructure for the exchange of spike events between units of the neural network. These communication infrastructures are usually built around custom-designed FPGA systems. However, the overall bandwidth requirements and the integration density of very large neuromorphic systems necessitate a significantly more targeted approach, i.e. the development of dedicated integrated circuits. We present a VLSI realization of a neuromorphic communication system-on-chip (SoC) with a cumulative throughput of 32 GBit/s in 0.18 μm CMOS, employing state-of-the-art circuit blocks. Several of these circuits exhibit improved performance compared to current literature, e.g. a priority queue with a speed of 31 Mkeys/s at 1.3 mW, or a 1 GHz PLL at 5 mW. The SoC contains additional neuromorphic functionality, such as configurable event delays and event ordering. The complete configuration of the neuromorphic system is also handled by the spike communication channels, in contrast to the separate channels required in the majority of current systems. At 865 Mevent/s, the SoC delivers at least a factor of eight more bandwidth than other current neuromorphic communication infrastructures.

1. Introduction

Neuromorphic ICs implement mathematical abstractions of brain functions in order to realize novel cognition-derived computational functions [1,2]. The last years have seen a steady increase in the size of neuromorphic systems in order to handle more advanced cognitive tasks. These large-scale hardware systems for spiking neural networks require high-speed communication for transmitting so-called spike events between the different units of the neural network. Neuromorphic hardware systems commonly employ the address event representation (AER) protocol for pulse transmission [3,4]. Based on this, several routing/interface boards have been developed in recent years [4,5], usually employing a reconfigurable device such as an FPGA for greater flexibility. Those designs were predominantly optimized for asynchronous operation and low latency, whereas demands on integration density and bandwidth were relatively relaxed. Also, at least part of the configuration of the neuromorphic chips was usually performed by separate, proprietary interfaces.

However, requirements on integration and speed change profoundly when moving to a large-scale hardware system, such as the waferscale neuromorphic hardware developed in the FACETS and BrainScaleS projects [6], depicted in Fig. 1. This system employs waferscale integration technology to gain a high connection density. Furthermore, it is designed for operating at a speed-up factor of 10^4 compared to biological real time, which increases simulation speed and integration density of the analog neuron and synapse circuits at the same time [7].

We have developed a communication infrastructure [8] for this waferscale neuromorphic system centered around an application-specific digital communication IC, called digital network chip (DNC). In contrast to conventional parallel asynchronous AER interfaces, the DNC employs a significantly more versatile synchronous high-speed serial packet communication of time-stamped spike events. The resulting packet network is advantageous for spike transmission, because it efficiently copes with the large number of sources and targets of the waferscale system, exploiting the limited bandwidth of the single transmission channels and offering flexible configuration of the connectivity between the units. Transmission of conventional digital data packets also enables the system to embed the complete configuration information for the neuromorphic wafer in the regular data stream. The DNC employs various building blocks from conventional high-speed SoC design, adapting and advancing

* Corresponding author. Tel.: +49 351 463 34943.
E-mail address: Christian.Mayr@tu-dresden.de (C. Mayr).

0167-9260/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2011.05.003

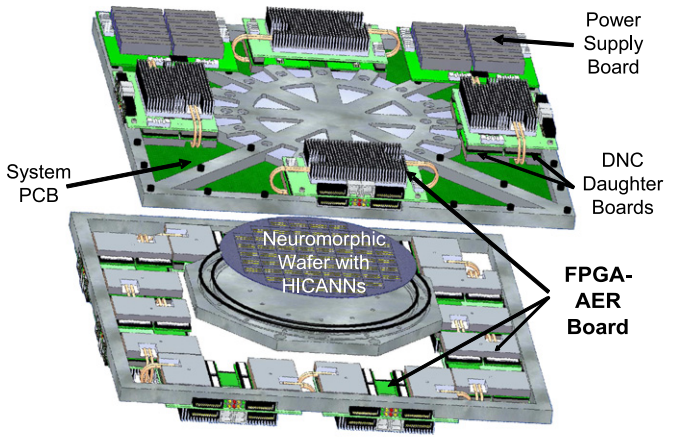


Fig. 1. Overview of one wafer module of the FACETS/BrainScaleS waferscale system [6].

them for this specialized task. Section 2 gives an overview of the waferscale system and derives the specifications for the DNC based on the overall system constraints. In Section 3, the design of the DNC is detailed, with Section 3.2 focussed on the functional aspects and Section 3.3 giving the circuit design. Measurements of single building blocks are distributed in the text, while the overall performance figures are given in Section 4.

2. System overview

2.1. The waferscale neuromorphic system

The FACETS/BrainScaleS waferscale neuromorphic system consists of several of the wafer modules as depicted in Fig. 1. It has been designed for integrating a maximum of neurons and synapses at sufficient flexibility for connectivity and model parametrization. As a consequence of the high integration density, all neuron and synapse circuits are accelerated by a factor of 10^4 compared to biological time. On the one hand, this prevents the system from being used in a real-time operating and learning setup, i.e. it cannot interact directly with the environment; on the other hand, stand-alone simulations of neural networks can be accelerated immensely.

The pulse frequencies resulting from the speed-up factor call for dedicated, high-speed pulse communication. This is achieved by a two-layer approach: On the wafer, individual synapse-and-neuron blocks, called high input count analog neural networks (HICANNs) [6], are connected by post-processed metal interlinks on top of the wafer, forming a high-density pulse routing grid [9,6]. This pulse routing grid covers the pulse communication inside a single wafer module. This intra-wafer communication is complemented by a packet-based network, connecting the wafer to the surrounding system (i.e. the stimulating host PCs and other wafer modules). The main building blocks of this packet-based network are the DNCs, which are situated on a PCB physically linking the wafer to the outside. As the packet-based network is the only communication link to/from the wafer, it has to provide pulse stimulation and monitoring as well as control and configuration of all the circuits on the wafer via the same links.

This packet-based off-wafer communication is hierarchically organized as shown in Fig. 2. Eight of the HICANNs are placed on one reticle on the wafer. Communication streams from eight HICANNs are bundled in one DNC. In turn, four DNCs communicate with one custom-designed FPGA–AER board [8]. This tree-like

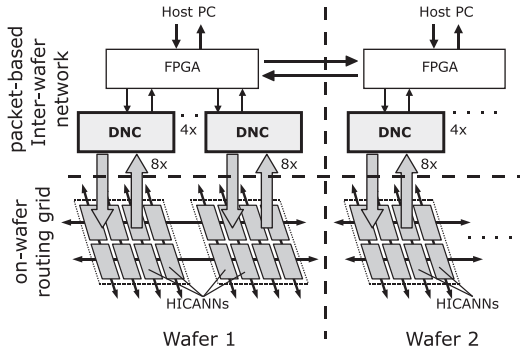


Fig. 2. Logical structure of the off-wafer packet-based network and the on-wafer routing grid.

structure enables one FPGA to control 32 HICANNs on the wafer. Overall, a single wafer module requires 12 FPGA–AER boards [6]. Since the packet-based network is tasked with the inter-wafer (i.e. FPGA–FPGA) communication, the hierarchy is always fully traversed for the pulse packets, i.e. a modular connection scheme operating in several stages of the hierarchy such as in [10] is not necessary.

2.2. Constraints

The architecture of the off-wafer routing network was chosen for two reasons: bandwidth distribution and geometrical restrictions. Thereby, the spike traffic constitutes the critical case, because spike event transmission cannot be slowed down, since the analog neuron and synapse circuits on the wafer cannot be suspended. To extract an estimate of the required throughput, we assume a reasonable mean spike rate of 10 Hz [11] per neuron. The only spike sources that send from the wafer are the neuron circuits on the HICANNs. With 512 such neurons [6] and a speed-up factor of 10^4, a total of 51.2 Mevent/s need to be transmitted from a single HICANN. For a whole wafer with 352 HICANNs [6], this adds up to 1.8×10^5 spike sources with 18 Gevent/s total rate. For a symmetric design, the same bandwidth is assumed for the connections to the wafer. With 24 Bit pulse packets, a throughput of 1.2 GBit/s is thus needed per DNC–HICANN connection, not taking into account packet overhead.

Besides the spike rate of a neural connection, its transmission delay is a crucial property that can significantly influence processing and adaptation behavior [12]. This delay is commonly assumed to be constant and in the order of several milliseconds [13]. With the introduced speed-up factor, this translates to some 100 ns in the system time domain, so that delay and jitter of the routing network have to be taken into account. Therefore, pulse event packets contain a 15 Bit timestamp in order to buffer them at the DNC until a target delay is reached [14]. This is in contrast to real-time hardware systems, where transmission delays are neglected [4] or handled by a discretized system update [15]. At the same time, this offers the possibility of configuring individual delays based on the time stamp, a feature which is especially useful in dynamical neuromorphic computation [16,17]. For details of this functionality, see Section 3.2.1.

3. The digital network chip

3.1. Toplevel overview

As described in Section 2.1, the DNC provides communication between the FPGA and up to eight HICANNs. Due to the high integration density, the number of physical wires between the HICANNs on the wafer and the outside is limited. Therefore,

low-voltage-differential-signaling (LVDS) transmission of one clock line and one data line is employed, forming a minimal communication interface between a HICANN and a DNC. For this interface, a raw data rate of 1.2 GBit/s is required, as detailed in Section 2.2. Incorporating some overhead for packet handling, a raw data rate of 2 GBit/s is realized at a clock frequency of 1 GHz for each HICANN–DNC connection, using double data rate (DDR) transmission. Furthermore, a fall-back mode is implemented with 1 GBit/s bandwidth at 500 MHz clock frequency. The serial communication is source-synchronous, i.e. the physical connection consists of two differential lines (clock and data) for each direction of a DNC-HICANN connection. All HICANNs together thus produce data with a maximum rate of 16 GBit/s per direction across the DNC. A matching 16 GBit/s data rate is provided by the DNC–FPGA connection, combining 16 LVDS data lines with one LVDS clock line running at a frequency of 500 MHz.

The system overview of the DNC is depicted in Fig. 3. Each connection consists of the corresponding LVDS transceiver, the serializer and deserializer circuits, responsible for the parallel to serial conversion, and the link packet control that implements a packet protocol for the different types of data with error detection and correction mechanisms. The various transmission packets of the DNC, which differ in size depending on the target device and the packet data, are shown in Fig. 4. The FPGA connection uses a

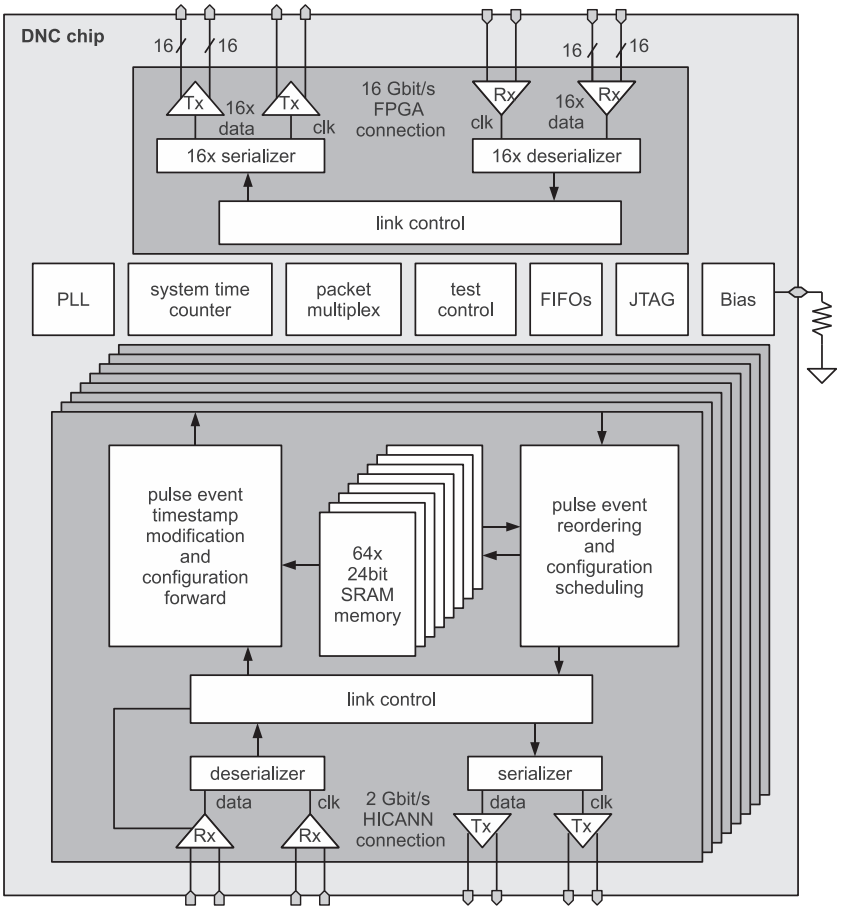


Fig. 3. Block diagram of the DNC with its eight HICANN interfaces and the FPGA connection.

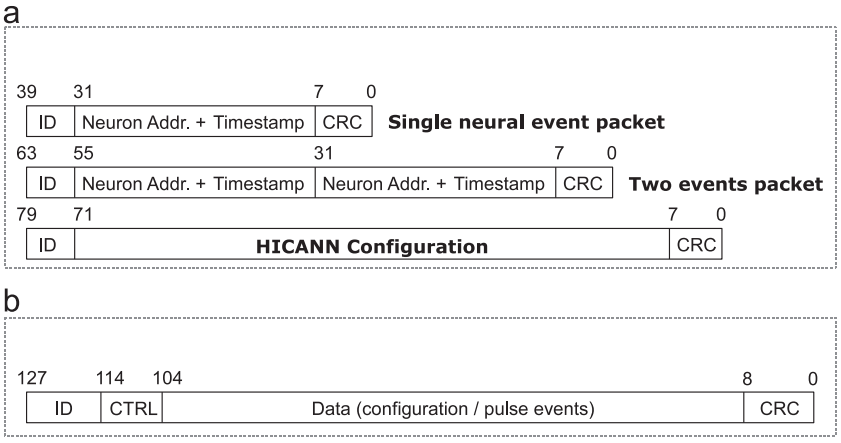


Fig. 4. Various communication packets of (a) the HICANN connection and (b) the FPGA connection.

single packet structure for different contents, while the HICANN connection implements several specialized packet formats. The content of the packets, i.e. the pulse events and the configuration data, is processed in each of the eight HICANN connection modules. Pulse events towards the HICANN need to be reordered based on their timestamps to compensate for the variable-delay packet routing (see Section 3.2.1). Events from the HICANN are modified in their timestamps, realizing configurable event delays.

Configuration packets may contain settings for the DNCs or for the different HICANN components, as well as current readout values of neural building blocks, like synapse weights or neural pulse release statistics. These packets have lower priority than pulse events. All packets include a CRC checksum to identify corrupted data. A phase-locked-loop (PLL) supplies the digital and analog environment with specialized clock signals and an on-chip-bias circuit with external reference resistor provides reference currents for the custom circuit blocks. Low-level test access is provided by a JTAG interface. The main components and functionalities are explained in detail in the following sections, as well as their contribution to the neural packet-based event routing.

3.2. Functional design

General design. The DNC needs to supply several mechanisms for handling the different packet types and for implementing the timestamp-based event routing. Thereby, low transmission latency needs to be ensured, especially for the pulse events, which need to be delivered within a biologically realistic time window (cf. Section 2.2). The latency crucially depends on the architecture of the packet flow. As there are nine possible sources for neural events and configuration packets, one strongly active channel could block the other channels. This effect can be circumvented by keeping all channels independent in their packet handling as much as possible.

On the DNC–FPGA link, up to four pulse events or one configuration packet can be transmitted in parallel in each packet per direction, requiring two clock cycles of the 250 MHz system clock of the communication system. Two HICANN channels share one pulse event slot of an FPGA packet, so only scheduling between two channels is required for pulses. Packets arriving from the FPGA are forwarded directly to the targeted HICANN channel and stored there in local FIFO buffers for further processing. From this point, all HICANN channels are identical and operate independently from each other. Each HICANN channel generates packets for the connected DNC–HICANN link with either one configuration segment or up to two neural events.

The packet handling in the system is done with source-based address routing and timestamp-based prioritization. The packet is routed through the network by using the source identifier. At each network node a look up table (LUT) is used to identify the links that the packet needs to be forwarded to. By this mechanism, packets can be duplicated at each node, allowing for a tree-like routing of the event to the different connection targets. This minimizes the total network traffic.

The stages of the routing are detailed in Fig. 5. Pulse prioritization in the network employs the target time of the pulse event, which is contained in each pulse packet. Pulses are sorted according to this value and events with the lowest timestamp are sent first. Initially, the target time is calculated in the DNC for each pulse coming from a HICANN, adding a source-specific delay to the sender time generated in the HICANN. When packets are duplicated at a network node, the new packets get a modified timestamp by adding a delta time. With the 250 MHz system clock of the communication system, the resolution of the timestamps equals 4 ns, corresponding to 40 μs in biological time.

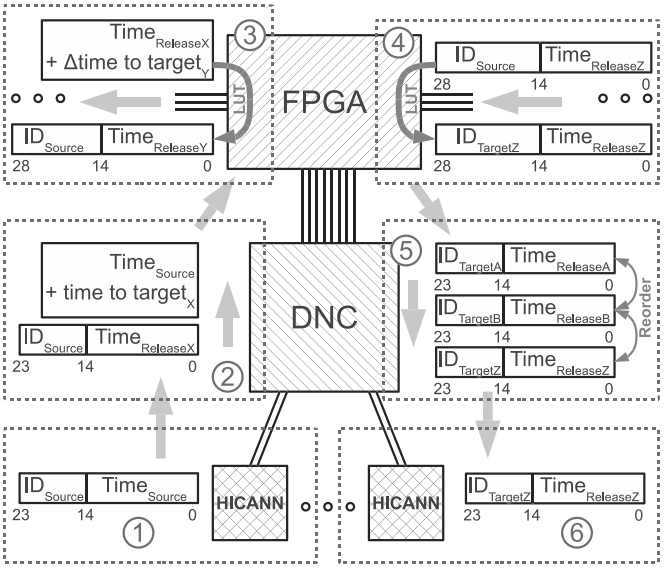


Fig. 5. Flow of pulse events in the waferscale packet network.

Thus, the entire 15 Bit timestamp corresponds to 1.31 s in biological time. Downstream to the HICANN, this represents the main delay that can be sorted in the priority queue before events become non-unique. This should be sufficient, as any larger delays can be handled by the FPGA board. Upstream to the host PC, these 15 Bit timestamps can easily be extended using the FPGA clock/counter to cover arbitrarily long timespans.

Upstream transmission. Neural events are initially generated in one of the HICANNs, consisting of a 9 Bit identifier and a snapshot of the current 15 Bit system time counter representing the sender time (label 1 in Fig. 5). Incoming packets from the HICANN are processed in the DNC by adding a delay value to the sender time, which can be configured individually for each source address (label 2). After correction of the timestamp the event is forwarded to the FPGA. If two event packets arrive at the same time, the second one is delayed until the first packet is modified and then processed itself.

In the FPGA, the source address is extended by HICANN and DNC identifiers to a 14 Bit label. This value is then used to designate the next target communication channels, again employing a LUT (label 3). If the packet is duplicated, the LUT entry also contains delay delta values for each copy, which are added to the original target timestamp. By this mechanism, different connection targets of a source address can be configured for individual transmission delays. Because the timestamp deltas are always positive, connection delays in the duplicates are always increased. Thus, it has to be ensured by the system configuration that a pulse packet always contains the target timestamp corresponding to the target with the smallest delay.

Downstream transmission. When a pulse packet is sent from the FPGA to a DNC, the identifier of the packet is modified, containing the target HICANN and the target neuron address of the event (label 4 in Fig. 5). Thus, no further address conversion is needed in the DNC.

In the DNC, all incoming events are ordered and buffered by their target timestamp (label 5). This reordering is done using a custom developed priority queue memory (see Section 3.2.1), always providing the event with the minimum target time. This event's timestamp is constantly compared with the current system time and released to the HICANN if the difference of both is smaller than a preconfigured limit value. This combination of buffering and scheduling reduces the demand on buffering capabilities on the wafer, maximizing silicon area for the neuromorphic circuits. The

limit for releasing accounts for the link delay down to the HICANN, which is dependent on the link traffic. It can be set in increments of 32 clock cycles. Up to eight priority queue memories are filled in parallel per HICANN channel, reducing sorting latency and utilizing the full bandwidth of the DNC–HICANN channel. In the HICANN the events are released at the system time represented by the target timestamp, employing parallel buffers (label 6).

The configuration packets in contrast do not require complex routing information. They are forwarded from the host control to the FPGA, the DNC and the HICANN directly, with lower priority than the timing-critical pulse packets. The host control provides the configuration data to the FPGA together with the target address for the addressed device. Each communication channel includes a buffering FIFO memory for the configuration data to hold it back until it can be submitted. These mechanisms are the same for all components in both directions towards and from the HICANN.

3.2.1. Priority queue

The priority queue block depicted in Fig. 6 handles the ordered release of the pulse events down to the HICANN based on their timestamp. It is based on a weakly sorted binary heap [18], which centers on finding the next closest event in time. The binary heap structure has been chosen as it offers very low hardware effort and a significantly faster insertion of new entries compared to a sorted array [19], since no search operation is carried out for an entire list at each insertion step [20]. Instead, the weak ordering of the heap allows continuous streaming where incoming packets are inserted while the closest pending event is released at the same time. The binary heap offers logarithmic-time insertion and removal of entries at a constant number of comparators [21]. The algorithm and its implementation are described in detail in [14]. The applied SRAM memory blocks offer 64 entries of 24 Bit each and have a dual port interface. The latter significantly reduces the clock cycles needed per

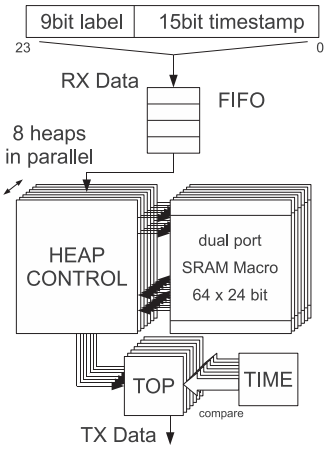


Fig. 6. Block diagram of the pulse event reordering unit for one HICANN channel, using custom priority queuing.

algorithm loop, with the core loop of the retrieve operation requiring only two clock cycles [14]. Each HICANN channel has eight priority queues operating in parallel, i.e. a single SRAM block and its surrounding digital circuitry correspond to one 64×24 Bit priority queue and its control (see also Fig. 6).

A comparison of a single priority queue with three sorting implementations in current literature is given in Table 1. Since there is a lot of variation especially with respect to number of keys and key size, the numbers in brackets give the approximate comparison values when adjusting the implementations found in literature to the priority queue specifications. When assuming that a single FPGA slice is at least equivalent to five gates, our implementation is significantly less complex than [20]. When adjusting the throughput of [20] for the higher keysize and architectural differences of our implementation, an equivalent throughput of approx. 50 Mkeys/s can be estimated. The serial architecture of [19] carries out a full sort on the input keys, rather than the weak sort of our priority queue. However, a comparison is still justified as it is used for the same purpose, i.e. a streaming prioritized key release. With key size and number of keys identical, this serial architecture requires many more gates and is significantly slower. The large-scale parallel implementation of [22] has to be adjusted for the smaller number of keys but higher key size of our design to enable a comparison. After adjustment, [22] would have ca. 50% more throughput rate and gate count 64 times less than stated in Table 1. However, the gate count is still a factor of 42 higher than our design. The power consumption of [22] in the adjusted version should be similar to our implementation. The parallel design of [23] achieves a significantly higher throughput at a slightly lower clock frequency, but also at the cost of increased complexity.

In general, parallel designs enable higher throughput at the cost of a fragmented memory and separate, area-consuming control stages at each heap level. In contrast, the design target for the presented implementation was not maximum throughput, but minimal area/power at the throughput required by the application. As can be seen, this is best served by a parsimonious serial implementation. Overall, the proposed priority queue combines a competitive throughput with one of the lowest complexities and the lowest power dissipation. It is both less complex and faster than previous serial sort implementations [19]. It is somewhat worse in terms of throughput compared to serial/parallel [20] or tree-parallel [22] algorithms, but exhibits a factor 20–40 less gate count.

3.2.2. Initialization

The communication between all components is established with a hierarchical initialization order. It follows the system structure from the FPGA via the DNC down to the HICANN. It is master–slave based, this means the FPGA controls the initialization as a master and the DNC reacts as a slave. After successful start up of the 16 GBit/s connection, the DNC starts the initialization of the HICANN interfaces, taking the master's role of these

Table 1
Comparison of a single priority queue with current state-of-the-art. The numbers in brackets give the approximate comparison values adjusted to the priority queue specifications [14].

Reference	Technology	Gate count	Power	f_{clock}	Throughput	#Cycles	Keysize (Bit)	#Keys
[20]	FPGA	2784 Slices	1.8 W	133 MHz	133(50) Mkeys/s	128	8	128
[19]	–	26,964	–	–	–	190	16	64
[22]	0.13 μm	1,604,261(25,066)	71 mW	143 MHz	35.8(54) Mkeys/s	–	12	4096
[23]	0.18 μm	78 k	–	200 MHz	100 Mkeys/s	2	18	64
This design	0.18 μm	593	1.3 mW	250 MHz	31 Mkeys/s	2–14	15	64

connections. As both connections are based on the same system concept, the initialization process is identical for both links.

The initialization algorithm for each data line is an adaptation of the application note of Xilinx for Virtex-V FPGAs [24] to stay compatible with the communication interface of the connected FPGA. It runs entirely in the semi-custom part of the chip with the system clock of 250 MHz for the HICANN connections and 125 MHz for the FPGA connection. The components control the different required settings for the configurable delay of the data (Section 3.3.2) and the word alignment mechanism of the deserializer (Section 3.3.3). As the system clock is much slower than the transmission clock, the algorithm needs to operate with the parallelized data from the deserializer. By delaying the serial data stream, the algorithm searches for two subsequent data edges. Afterwards, the data delay is set such that the clock captures directly in the middle between these two transitions. We use shifting of the data instead of the clock signal due to power saving issues, because the clock is an alternating signal that permanently causes the delay cells to switch levels. The different steps of the algorithm are depicted in Fig. 7.

Fig. 7(a) shows the transmitted signals coming from the LVDS pads. Due to DDR transmission, the data Bits have a length of 500 ps at a clock frequency of 1 GHz. They are captured into a serial shift register chain alternately on both edges of the clock. To capture valid data, the clock-to-data phase needs to be 90° and this delay needs to be established independently of temperature, process corner, power supply distribution and transmission frequency. For finding an optimal clock-to-data phase, we use the fact that at the phase values 0° and 180° a data transition occurs, so that the 8 Bit parallel data vector changes its value.

The algorithm starts by continuously increasing the delay of the data (Fig. 7(b)), while monitoring the output vector. When the parallel vector changes, the start of the first transition is found (Fig. 7(c)). To find the end of the transition region, the algorithm adds more delay until the received vector stays constant over 128 system clock cycles (Fig. 7(d)). The delay setting at this point is stored as first delay value. Afterwards, the algorithm increases the delay until the data vector changes again, denoting the beginning of the next transition (Fig. 7(e)). The corresponding delay value is averaged with the value for the first transition, resulting in the desired clock-to-data phase of 90° (Fig. 7(e)).

The most important requirement to get the best final position is a monotonously increasing controllable delay. The smaller the single steps of the delay are, the better is the final relation between clock and data near 90°. The longest required combined delay length is the period of the clock. When the algorithm starts and the clock lies right behind a transition, it is required to delay the data nearly half of a period to reach the first recordable transition. At this point another half of the period is required to capture the second data transition safely. So depending on the transmission frequency, different maximum delay lengths are required, implemented with different numbers of small equal-step delay elements. In our implementation, there are 32 delay elements available for a transmission with 2 GBit/s mode at 1 GHz and 64 delay elements for 1 GBit/s mode at 500 MHz.

Having found an optimum clock-to-data alignment, the Byte words of sender and receiver need to be aligned at the receiving side. The correspondingly required Bitwise shifting of the data is provided in the deserializer (see Section 3.3.3). The corresponding alignment algorithm shifts the selected tap until the word alignment is established. The selected initialization pattern for this detection is `0x2C` which allows only one valid alignment configuration. Fig. 8 shows the structure of the shifted data stream. The deserializer realizes 15 alignment positions, realized via eight flip flop registers on the positive edge and seven registers on negative edge. This offers eight different possibilities to capture the initialization pattern, enough for a Byte-wise alignment.

The initialization of the HICANN channel is finished at this point and the channel can start normal operation. In contrast, the FPGA channel consists of 16 of these LVDS connections, so here all single channels need to be initialized independently and afterwards combined to one large communication channel with a parallel data width of 128 Bit.

The figured algorithm differs significantly from solutions in literature [25–27] by avoiding the use of a special DLL/PLL with multiple clock phases to capture the different data eye sections or clock-data recovery, which reclaims the clock frequency and clock phase out of the data stream. Furthermore, it is completely synthesizable. It runs once automatically in the initialization phase, thus optimizing power consumption. Our realization does not realign the clock-data-phase while communication is established itself, because it is not possible to measure the complete data eye again, without influencing the current transfers. However, a completely new realignment of the edges can optionally be started if a certain level of communication errors is detected by the CRC error check.

The proposed algorithm offers a large flexibility for different environments. Primarily, it compensates different LVDS signal runtimes, as the system PCB (see Fig. 1) has limitations in trace length compensations due to sparse available routing space. The PCB needs to supply the whole wafer with different power domains and signals, but the FPGA–AER boards are located only at the corners of the wafer. This results in varying distances between the communication partners and also in unequal lengths of different LVDS traces. Furthermore, the DNC is used for characterization of single HICANN prototype chips in other measurement setups. Thus, we are also able to compensate trace

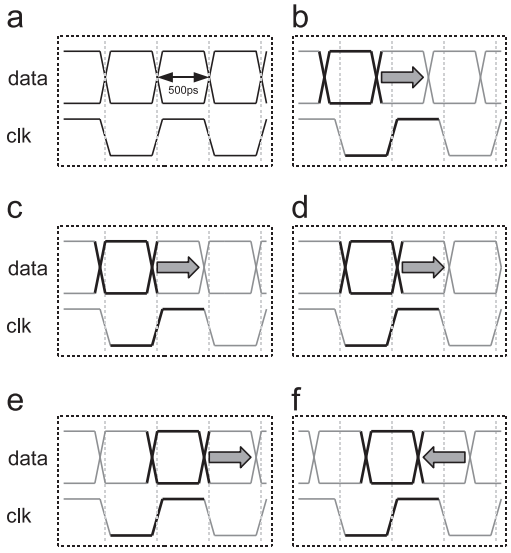


Fig. 7. Different phases of the data eye-search algorithm. Only the data is delayed.

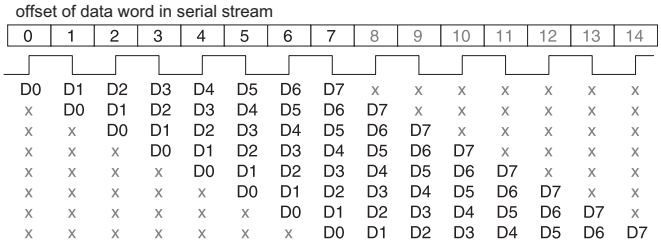


Fig. 8. Word alignment in serialized data stream.

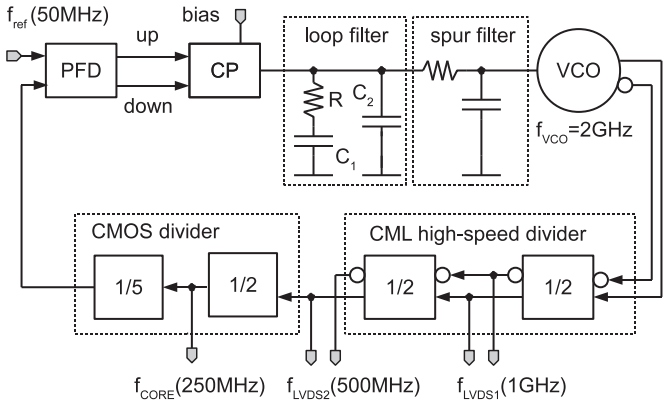


Fig. 9. PLL block level schematic.

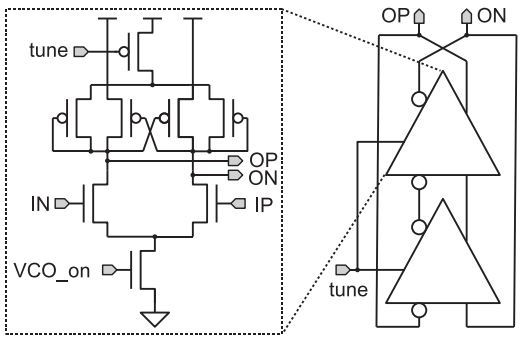


Fig. 10. VCO schematic.

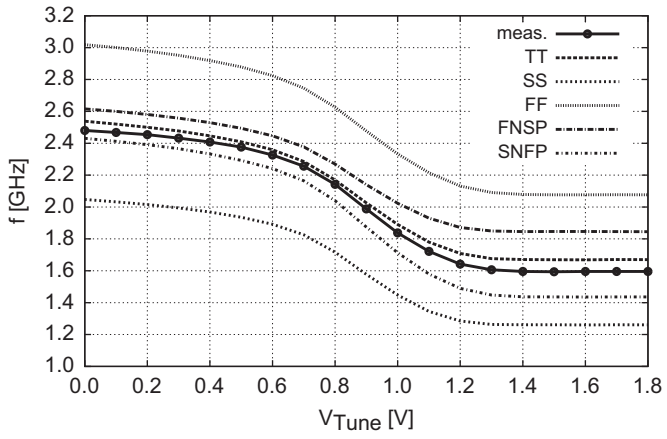


Fig. 11. VCO tuning characteristics at room temperature and 1.8 V supply voltage, measurement and post-layout corner simulation results (TT: both NMOS and PMOS transistors typical; SS: PMOS slow, NMOS slow; FF: both fast; FNSP: NMOS fast, PMOS slow; SNFP: NMOS slow, PMOS fast).

properties of commercial FPGA development boards which interface the various prototype measurement setups.

3.3. Custom circuit components

3.3.1. PLL clock generator

The PLL provides a 1 GHz clock for the serial high-speed LVDS I/O, a 500 MHz clock for the FPGA interface and a 250 MHz core logic clock from a 50 MHz system reference clock. A standard integer-*N* charge pump PLL architecture [28] is used here. Fig. 9 shows the block level schematic of the PLL. The output signal of a voltage controlled oscillator (VCO) is divided using a frequency divider and compared with a reference signal using a phase-frequency-detector (PFD). If there is a phase or frequency deviation detected, the charge pump (CP) adds or removes charge from the loop filter capacitors and therefore adjusts the VCO oscillation frequency. If the PLL is locked, the output frequency is *N* times the reference frequency. Here it is $f_{VCO}=2$ GHz, $f_{ref}=50$ MHz and $N=20$. The outputs of the first divide-by-2 stages are directly used as clock signals for the high-speed LVDS I/Os which operate at 1 GHz or 500 MHz. This ensures a good clock duty cycle close to 50% which is mandatory for the targeted source-synchronous data transmission with DDR. The CMOS frequency divider provides the 250 MHz core clock.

The VCO is a two-stage differential ring oscillator with positive feedback within the cells as presented in [33]. Fig. 10 shows its schematic. Each delay stage with a cross-coupled PMOS pair provides 90° phase shift. The signal inversion by crossing the differential wires in the feedback path provides additional 180° phase shift, leading to 360° for fulfillment of the oscillation criterion. It is sized for operation at 2 GHz oscillation frequency for all relevant process, voltage and temperature corners. Fig. 11 shows the simulated and measured VCO tuning characteristics. The frequency divider is built up using fast current-mode-logic (CML) cells for the first two divide-by-2 stages and static CMOS logic for the slower divide-by-2 and divide-by-5 stages in Fig. 9. The PFD is a standard implementation [28] using static CMOS gates. It determines if the rising edge of the reference clock signal or the divider output is leading and generates a corresponding up/down control signal for the charge pump. The charge pump is designed for good matching between pump-up and pump-down currents and low parasitic charge injection into the loop filter to reduce ripple on the VCO control voltage and therefore jitter in the PLL output signals. Its nominal current that is steered to or from the loop filter node is derived from the reference current bias circuitry. Thus, the charge pump current is adjustable by the current bank for compensation of process variations which might affect the closed loop PLL dynamics. The low-pass filter consists of a second order passive loop filter that is completely integrated on chip, i.e. no external components like resistors or capacitors are used which reduces pin count and the number of discrete devices on the PCB. Therefore the capacitor values are limited by the available chip size of the PLL. A total capacitance of 50 pF is implemented on chip using MOS capacitances. The capacitance

Table 2
Performance comparison of PLL clock generators.

Reference	Technology	Area (μm²)	f_{out} (GHz)	*P* (mW)	Peak-peak jitter (ps)	RMS jitter (ps)
[29]	0.18 μm	240,000 (2 PLLs)	0.125–1.250	90	–	7.2 (at 1.25 GHz)
[30]	0.15 μm	67,600	0.25–2.0	7.6 (at 2 GHz)	–	4.3 (at 1.2 GHz)
[31]	0.18 μm	160,000	0.1–0.5	24.3	32 (at 0.27 GHz)	4 (at 0.27 GHz)
[32]	0.18 μm	150,000	0.5–2.5	25	–	3.29
This design	0.18 μm	23,900	1.0, 0.5, 0.25	5.0	55.0 (at 0.5 GHz)	7.7 (at 0.5 GHz)

ratio C_1/C_2 of Fig. 9 is $\approx$ 10. The resistor $R \approx 2\,k\Omega$ is sized to achieve a phase margin of at least 60° of the closed PLL control loop. An analytical, linear small signal model (see [34]) is used for this design purpose. A first order passive spur filter is added to reduce VCO control voltage ripple caused by the switching of the charge pump which additionally reduces jitter of the PLL output signal. Its bandwidth is much higher than the PLL closed loop bandwidth and therefore it is not degrading the PLL dynamics in terms of phase margin and closed loop bandwidth. Furthermore, a time domain noise and jitter model has been developed [35] which is used to predict timing jitter from simulated phase noise characteristics of the PLL components to estimate and improve the overall jitter performance of the frequency synthesizer early in the design phase.

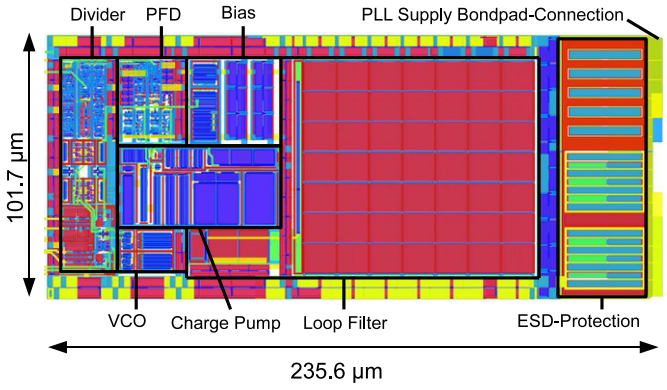


Fig. 12. Layout of PLL clock generator.

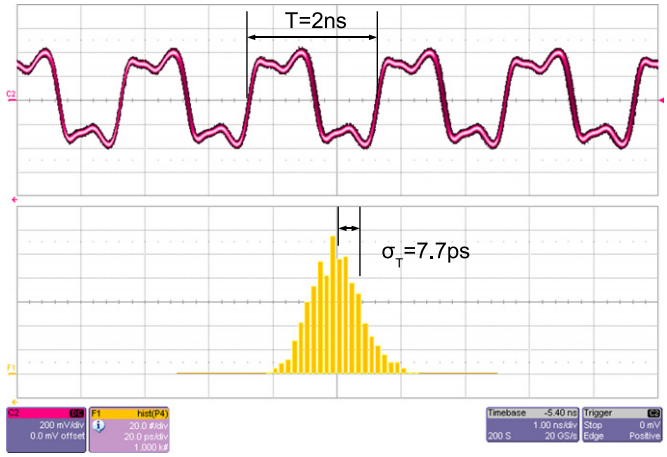


Fig. 13. Measured 500 MHz PLL clock signal at LVDS TX pad.

Fig. 13 shows the measured 500 MHz output clock signal at the LVDS TX pad output. Table 2 summarizes measured performances of the PLL clock generator and compares it to previously published ring-oscillator based PLLs in similar CMOS technology nodes and output frequency ranges. Our design exhibits low power consumption and small chip area at some cost of jitter performance. This is achieved by reducing the power consumption of the VCO as main contributor to total power at cost of jitter due to VCO noise. Incorporating minimal loop filter capacitances leads to significant area reduction as these capacitances are the main contributors to chip area. This leads to slightly increased jitter due to reference clock noise and feedthrough. Thus, this design optimizes power and area for the given jitter design target, i.e. sufficient jitter performance for the source-synchronous serial data transmission scheme. The proposed clock generator requires only a small chip area and fits in the I/O padframe of the DNC. Fig. 12 shows the layout of the PLL.

3.3.2. *LVDS transceiver*

LVDS is a standardized high-speed link [36]. In the implementation discussed here, two wires are used for data transmission which are differentially current driven based on the approach presented in [37]. The logic level of transmitted data is defined by the direction of the current flow through the wires. The current direction is read out via the voltage drop over the transmission line termination resistor.

LVDS transmitter. The block level diagram of the LVDS transmitter is shown in Fig. 14. The transmitter first shifts the voltage level of the input signal from 1.8 V core-domain to 3.3 V I/O-domain which is necessary to realize the I/O voltage levels of the LVDS standard [36]. Following the level shifter a buffer amplifies the signal strength to be suitable to drive the current bridge. Thereafter a switchable current bridge (Fig. 15) is employed to determine the current flow direction. It feeds a current up to 7 mA through both ends of the transmission line. Either M1 and M4 or M2 and M3 are conducting

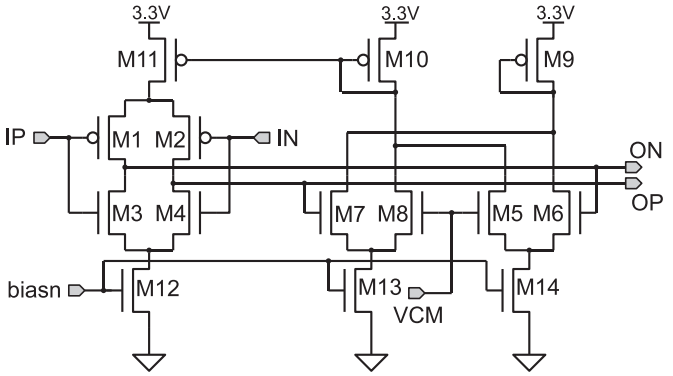


Fig. 15. Transmitter current bridge.

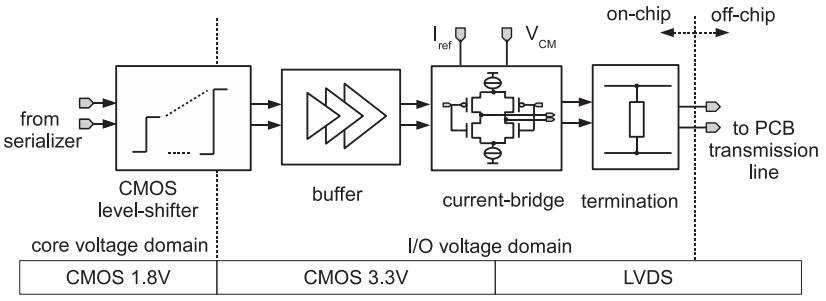


Fig. 14. LVDS transmitter overview with different voltage/signal domains indicated.

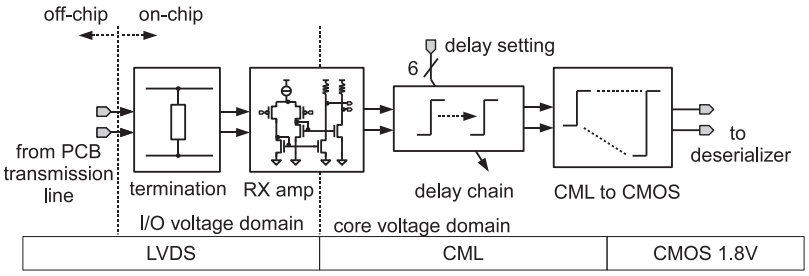


Fig. 16. LVDS receiver overview with different voltage/signal domains indicated.

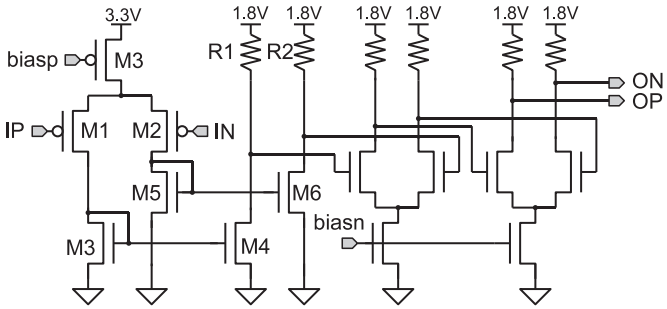


Fig. 17. LVDS receiver amplifier.

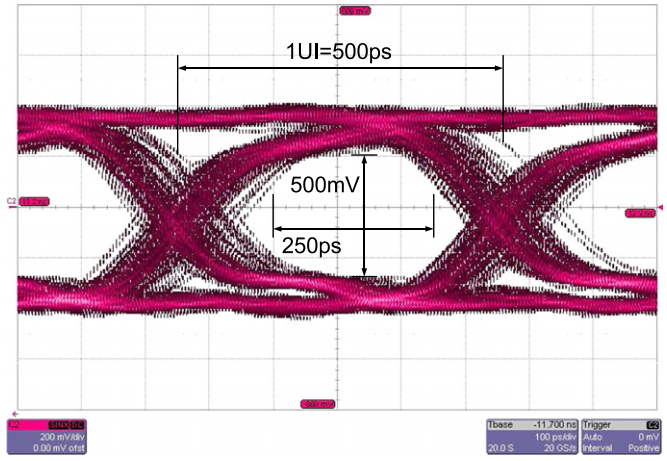


Fig. 18. Eye pattern diagram of the LVDS transmitter at 2 GBit/s data rate, measured after 10 cm differential transmission line on FR4 PCB.

Table 3
LVDS TX and RX design comparison.

Reference	Technology (µm)	Data rate (GBit/s)	Power TX/RX (mW)	Area (µm²)
[38]	0.18	1.8	19/9	87,000
[39]	0.18	2.5	4.8–9.3/–	67,000
[40]	0.18	1.6	35/6	420,000
[37]	0.18	1.2	23/–	22,000
This design	0.18	2.0	24/4 (+28 delay line)	42,000 (TX and RX)

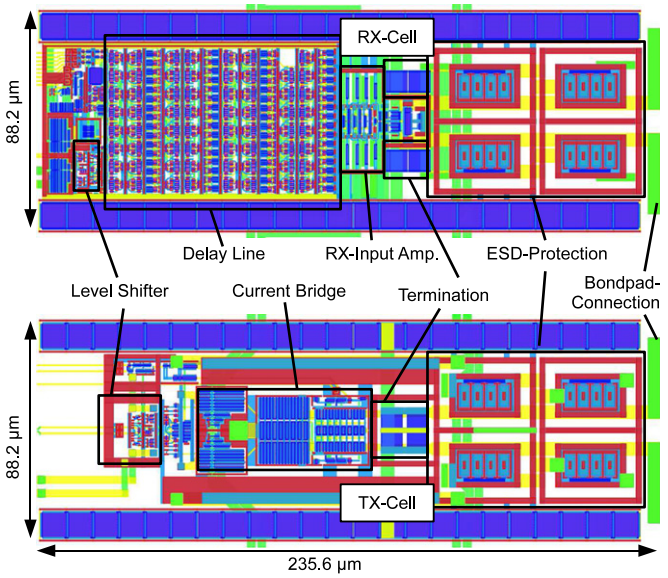


Fig. 19. Layouts of TX and RX cells.

at a time. The common mode output voltage is compared to a reference by the differential pairs M5&M6 and M7&M8, which controls the PMOS current source M11.

An integrated termination resistor is employed to match the transmission line impedance of 100 Ω, avoiding signal reflections at the end of the transmission line.

LVDS receiver. The block level diagram of the LVDS receiver is shown in Fig. 16. It consists of the transmission line termination resistor, the input amplifier, a variable-delay line and a comparator. The voltage drop over the transmission line termination resistor, caused by the current flow through the transmission line, is employed to feed the receiver amplifier.

The receiver amplifier is shown in Fig. 17. To allow wide input common mode range, a folded topology is used here. Due to an I/O-voltage domain of 3.3 V and an input common mode voltage range of 2.5 V [36] no rail-to-rail input is needed because low threshold voltage input transistor devices (M1, M2) are used. This keeps the input stage simple. The differential PMOS pair (M1, M2) converts the input voltage difference to a current difference which is mirrored (M3 to M6) to a resistor pair (R1, R2) to be converted to a voltage difference again. Hereby also the supply voltage domain crossing from 3.3 V I/O supply to 1.8 V core supply is realized. Two differential amplifier stages amplify the voltage signal and limit its swing to $I_{bias} \cdot R$. This allows to generate an output signal with well defined swing for a wide range of input signal amplitudes, which can be used directly as input for the delay line.

Each LVDS TX or RX cell including its termination resistors and ESD protection requires a chip area of 88.2 µm × 235.6 µm, which fits into the pad ring of the DNC. The layout of the cells is shown in Fig. 19. They consume a typical power of 24 mW (TX) and 32 mW (RX) (thereof 28 mW integrated delay line). Data rates up to 2 GBit/s per channel can be transmitted via the transmission line over the distance necessary to connect the waferscale system. Fig. 18 shows an eye pattern diagram measured after 10 cm FR4 PCB transmission line. Table 3 compares previously published LVDS designs with the proposed one. The power consumption of the presented design is comparable to current literature, while

the achieved data rate is among the highest. The presented TX and RX cells require the smallest chip area when taking into account that for [37], only the size of the TX cell is shown.

Delay line. The received LVDS signals have to be delayed by t_d to allow alignment of clock and data signals using the eye-search algorithm explained in Section 3.2.2. Therefore the delay range must be $t_{d,max}-t_{d,min}>1$ ns at 2 GBit/s data rate and $t_{d,max}-t_{d,min}>2$ ns at 1 GBit/s data rate respectively. The absolute delay value t_d is not critical because the alignment relies on the relative delay between the clock and data signals $t_{d,clk}-t_{d,data}$. The delay step Δt_d must be significantly smaller than 500 ps to allow a fine resolution of the eye-search algorithm at 2 GBit/s data rate. Finally, the delay line must not degrade the signal quality by addition of timing jitter due to inter symbol interference (ISI). This would reduce the data eye opening and might cause malfunction in the deserializer.

There exist different approaches for controlled delay lines. First, the delays of single logic stages can be modified by either adjusting their drive strength [41] or their load capacitance [42]. This approach is often used in applications where a fine delay resolution is required for accurate frequency or phase generation in digitally controlled PLLs or DLLs. The drawback is the limited tuning range. In a second approach cells with fixed delays are used which are connected in a chain whose length can be configured digitally [43]. Although the delay step cannot be smaller than the delay of one element, the tuning range can be extended arbitrarily by simply adding more delay cells to the signal chain. This approach is well suited for this application where a wide tuning range is required for the eye-search algorithm. Fig. 20 shows the block level schematic of the delay line. The signal enters the line at its input and based on the binary multiplexer settings the output after n cells is selected. The delay step corresponds to the delay of one element.

Fig. 21 shows a first order *RC* model of a delay element, which is valid for static CMOS cells [44] as well as for current-mode-logic (CML) implementations [45], where R models the cell driving resistance and C the load capacitance. If the delay cell is settled (e.g. after a long time of static input) and a pulse with width T is applied at the input, the corresponding output pulse exhibits a delay t_d and a reduced pulse width of $k\cdot T$ $(0\le k\le 1)$.

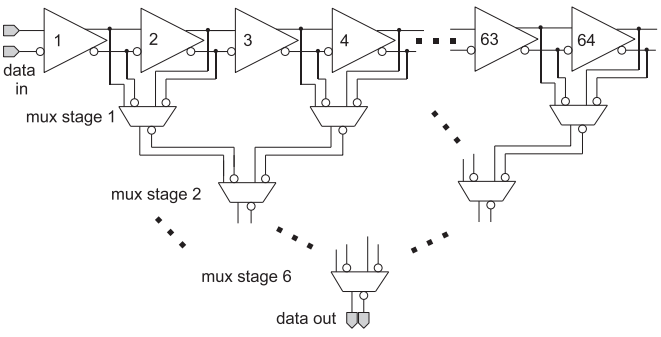


Fig. 20. Delay line schematic with 64 delay cells and 6 binary multiplexer stages.

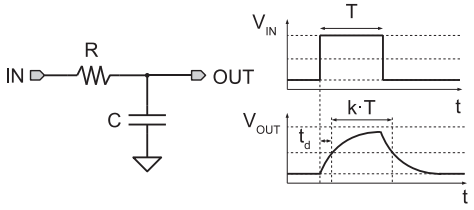


Fig. 21. Delay cell first order *RC* model and waveform.

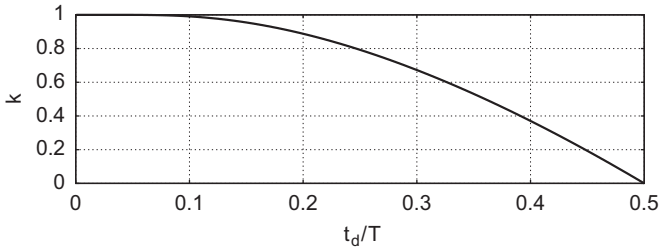


Fig. 22. Pulse width reduction in first order delay cell *RC* model, analytical solution.

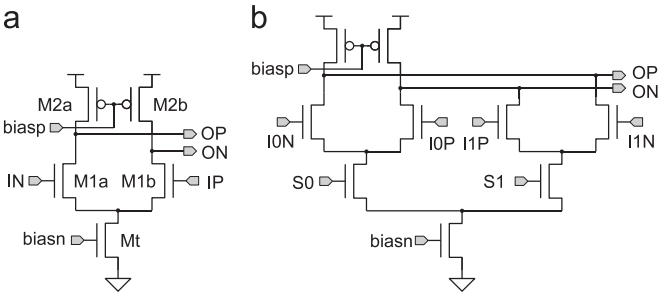


Fig. 23. Delay line component schematics. (a) Delay cell. (b) Multiplexer.

There exists a relationship between t_d and the pulse width reduction factor k which can be derived by solving the first order differential equation of the *RC* network in Fig. 21 analytically.

$$k\left(\frac{t_d}{T}\right)=1-2\cdot\frac{t_d}{T}\cdot\left[\ln 2-\ln\left(2-2\cdot e^{-T\ln 2/2t_d}\right)\right] \quad (1)$$

Fig. 22 shows this relation. At $t_d/T=0.1$ we obtain $k(0.1)=0.9936$. Based on this result and the required eye-search resolution, the delay of one element is targeted to be smaller than $0.1\cdot 500$ ps, where this and the constraints mentioned above must be fulfilled for all relevant process, voltage and temperature (PVT) corners. Thus, for the 2 GBit/s operation 32 delay cells are used for a delay range larger than 1 ns as shown in Fig. 20, resulting in a worst-case pulse width reduction of $k_{worst,2GBit/s}>k(0.1)^{32}=0.81$. For 1 GBit/s data rate 64 cells are used and we obtain $k_{worst,1GBit/s}>k(0.05)^{64}=0.99$.

The delay cell timing constraints cannot be realized using static CMOS logic cells in the given 0.18 μm CMOS technology, because the sensitivity of a CMOS delay with respect to PVT corners is too high. Therefore CML cells are used here. Fig. 23 shows the schematics of the delay element and the multiplexer. In the delay cell shown in Fig. 23(a) an NMOS transistor Mt acts as current source with I_{bias}. This current is steered to one of the PMOS loads M2a/b by the differential NMOS pair M1a/b based on its input voltage difference. The PMOS load devices are operated in linear region with a drain source resistance of R_{M2} and generate the output voltage difference. The output high voltage is V_{DD} and the low voltage is

$$V_{DD}-I_{bias}\cdot R_{M2}=V_{DD}-V_{swing} \quad (2)$$

The multiplexer shown in Fig. 23(b) consists of two basic CML stages which operate on a common differential output node and can be activated by the select signals S0 and S1 respectively. A CML to full swing CMOS level converter is used at the output of the multiplexer to interface with the deserializer logic.

Fig. 24 shows the circuit that is used to generate the bias voltages for the CML cells. The voltage V_{biasn} is generated using a diode connected NMOS device Md with a reference current I_{ref}. Thus this device forms simple current mirrors with the CML devices Mt.

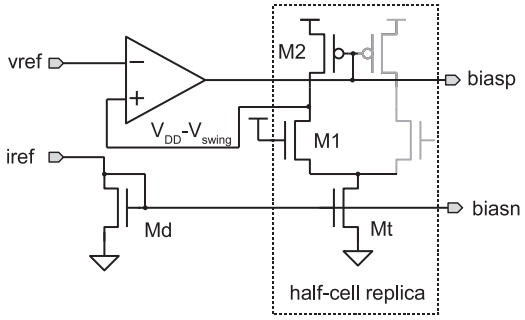


Fig. 24. Active bias circuit for current-mode-logic delay and multiplexer cells.

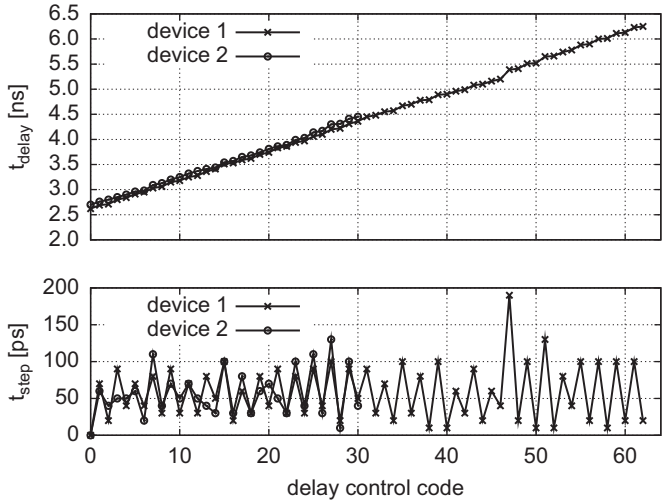


Fig. 25. Measured delay characteristics of two circuit realizations, device 1 in 1 GBit/s mode (64 cells) and device 2 in 2 GBit/s mode (32 cells).

The voltage V_{biasp} is generated using an active replica biasing circuit similar to the technique presented in [46,47]. A half cell replica of the CML gate is biased in its logic on state and the output low voltage swing $V_{DD}-V_{swing}$ is sensed by an operational transconductance amplifier (OTA) and compared to a voltage reference V_{ref} which is derived from the supply voltage by a resistive voltage divider. In this closed loop V_{biasp} is adjusted by the OTA such that

$$V_{swing} = V_{DD} - V_{ref} \tag{3}$$

The effective resistance R_{M2} reads

$$R_{M2} = \frac{V_{swing}}{I_{bias}} = \text{const} \tag{4}$$

and therefore is defined only by the reference values and does not depend on process and temperature variations. Thus the output time constant is

$$\tau = R_{M2} \cdot C_{load} \tag{5}$$

which determines the cell delay (see Fig. 21) which is a constant delay over process and temperature variations in a first order approach. This is mandatory to keep the clock-to-data alignment with respect to temperature variations during circuit operation without the need for re-calibration.

Fig. 25 shows the measured delay line characteristics. The delay time is monotonic with respect to its control signal. The average step size is ≈ 60 ps and there are at least eight steps within a 500 ps delay time window. Therefore the requirements for 2 GBit/s data eye-search operation are fulfilled. The total power consumption of the delay line is 28 mW.

3.3.3. Serializer/deserializer

Serializer/deserializer circuits are used to generate serial GBit data and clock stream from a parallel data stream and vice versa. The blocks also translate between the system clock and the transmission clock domain.

Both circuits contain interface blocks which operate in the system clock domain. They are designed using Verilog HDL, and synthesized/placed using standard place and route tools. In contrast, the components in the transmission clock domain operate up to 2 GHz nominal and are designed using specialized high-speed cells together with manual placement and routing. The main components are flip-flops consisting of dynamic latch cells, which are used in shift and control registers.

Serializer. The main components of the serializer are the digital interface block, two banks of shift registers and a control engine as shown in Fig. 26.

The interface block buffers incoming parallel data words (8 Bit), and enables the control engine. After synchronizing the enable signal, the control engine starts the transmission by copying the buffered data from the interface block to the serialization registers. The two shift register banks use four transmission clock cycles to transmit the eight data Bit, generating from those Bits the DDR data signal and the associated clock signal. As the shift register banks operate on contrary clock phases, also the associated control signals are generated for each phase separately. In contrast to [48], this ensures setup and hold time of a full clock cycle.

In order to allow arbitrary ratios of digital and transmission clock, the serializer control engine utilizes a dedicated 2 Bit counter. The counter is realized as Gray code counter with signals *Y*0, *Y*1, and also generates a LR (LoadRegister) signal which controls the data initialization of the serialization registers using

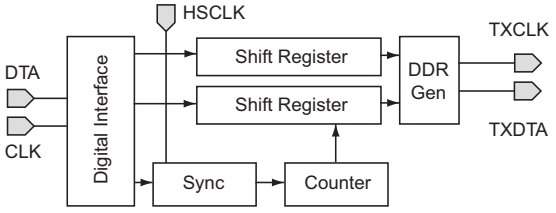


Fig. 26. Block schematic of the serializer.

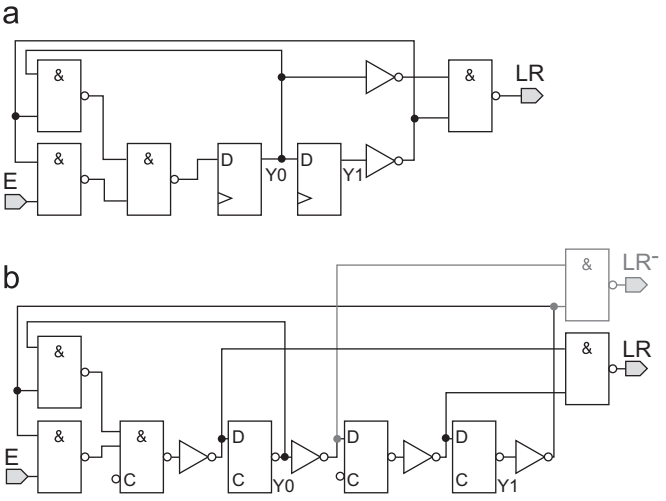


Fig. 27. Block schematic of (a) the naive and (b) the modified latch-based counter implementation.

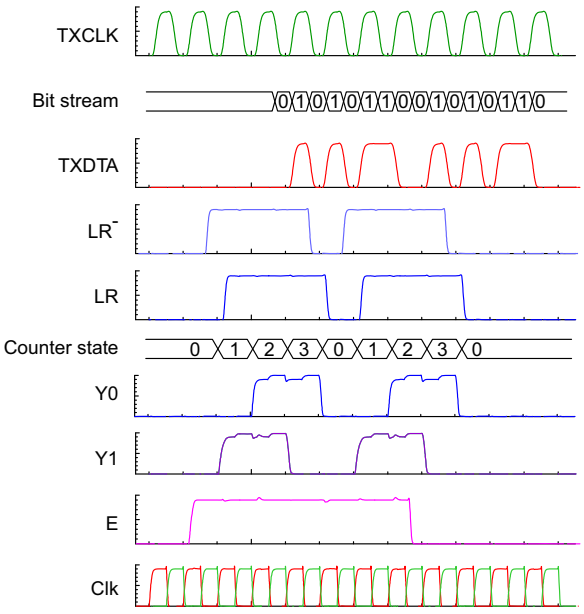


Fig. 28. Timing diagram of the serializer function.

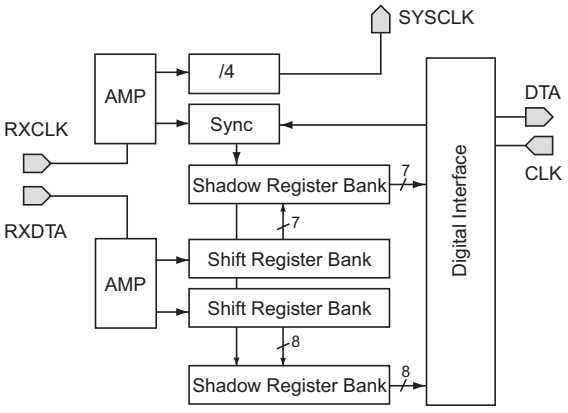


Fig. 29. Block schematic of deserializer.

the following Boolean functions:

$$Y0 = \overline{Y1} \cdot Y0 + \overline{Y1} \cdot E = \overline{(\overline{Y1} \cdot Y0)} \cdot \overline{(\overline{Y1} \cdot E)} \tag{6}$$

$$Y1 = Y0 \tag{7}$$

$$LR = Y0 + Y1 = \overline{\overline{Y0} \cdot \overline{Y1}} \tag{8}$$

To maintain functionality at all process, temperature and supply corners, further logic modification is necessary. The use of logic NAND functions is in general preferable over NOR, as the latter requires comparably large PMOS devices to guarantee the required output resistance, which on the other hand is impractical regarding the capacitive load on the input side. In the target application, the nominal delay of a single Boolean element is almost half of the available clock cycle. Thus, it is not possible to realize the equation $Y0$ in the naive way (Fig. 27(a)). The applied solution is to merge one logical NAND function with the dynamic latch (Fig. 27(b)).

Additionally, all internal signals requiring fanout are buffered as shown in Fig. 27(b), since the dynamic latch cells are sensitive to load.

Parallel to the LR output signal of the counter, a LR signal is generated separately for the contrary clock phase by using the inner signals of the counter flip-flops. The timing diagram with the representative signals is shown in Fig. 28. Here also the functionality of the counter is depicted, which allows an arbitrary start and ensures the complete execution of the transmission.

To drive the eight serialization registers, additional four fanout latches (two per phase) are implemented to reduce the signal load. This also relaxes the timing constraint at the serialization flip-flops by adding an additional clock cycle to their hold time.

Deserializer. The function of the deserializer is to receive the serialized data and transmission clock signals from the LVDS pads and to extract data words of 8 Bit from the serial Bit stream. To be able to select the correct word boundary, 15 Bit of the stream are sampled. This allows the data word selection in the slower system clock domain, significantly reducing custom logic in the transmission clock domain. A block schematic is shown in Fig. 29.

The deserializer receives differential data and transmission clock signals from the LVDS pads. The signals are buffered by input amplifiers and the transmission clock is further buffered

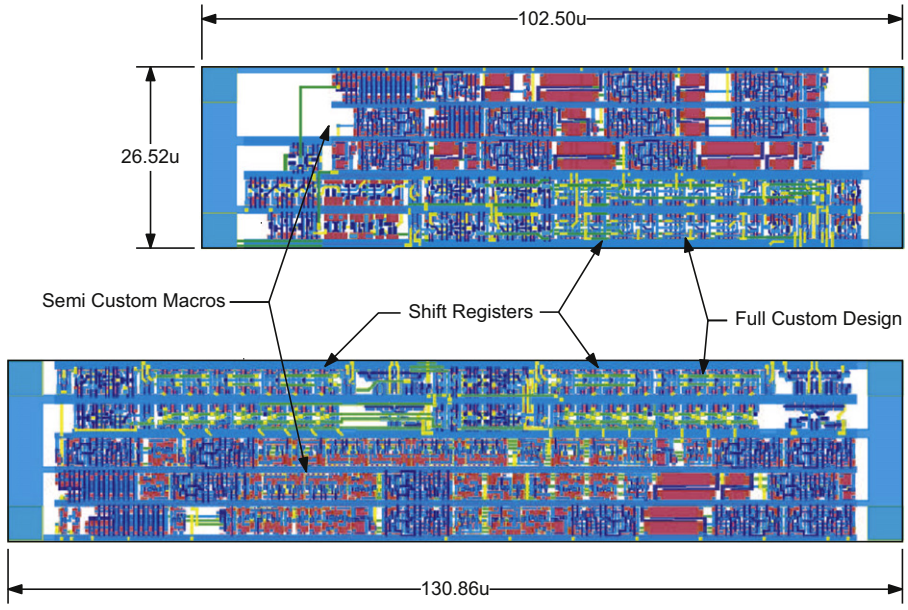


Fig. 30. Layouts of serializer/deserializer.

with six clock drivers. Four of these clocks are used to operate the 15 data shift registers, one is divided by 4 to be used as system clock and the last is used to synchronize the system clock back to the transmission clock domain.

The data signal is captured by two shift register banks consisting of 15 dynamic flip-flops, of which eight operate on the rising and seven on the falling edge of the transmission clock, similar to [49]. At the rising edge of the system clock (at every 4 transmission clock cycles), the data in the shift registers is transferred in parallel to 15 shadow registers, from which the digital control then selects the transferred data word. The correct word boundary is determined from the initialization pattern at the startup phase of the communication link (Fig. 8). In Fig. 30 the layout view is presented, showing the mixture of full-custom and semi-custom design elements.

Table 4
Characteristics of the presented neuromorphic communication chip (DNC).

Technology	UMC 0.18 μm 1P6 M
Chip area	5*10 mm^2
Clock frequency (system)	250 MHz
Clock frequency (LVDS)	1 GHz/500 MHz
Number of birectional LVDS channels plus clock channels	33
Overall number of gates	4,158,162
Power consumption	6.6 W

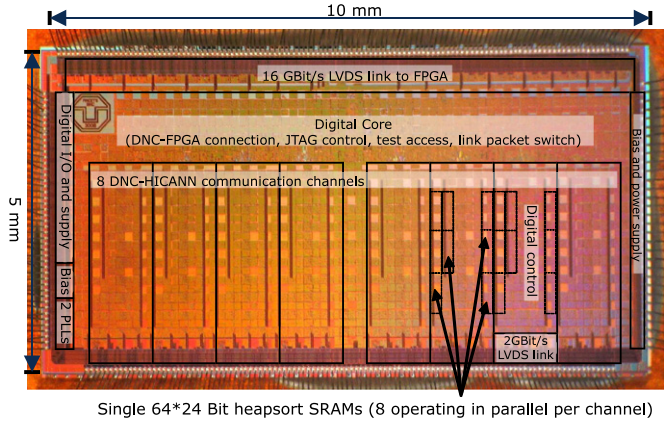


Fig. 31. Chip photograph of the digital network chip, realized in 5 mm × 10 mm chip area.

4. System results

Table 4 sums up the characteristics of the DNC. Fig. 31 shows a chip photograph with the main building blocks as detailed in the previous sections.

Table 5 gives a comparison with other AER system solutions. All of them are based on FPGAs for implementing their communication protocols and neuromorphic functionality. Their interfaces are either parallel asynchronous AER [5] or a variety of adapted commercial standards. In terms of speed, the systems of [3,4] come closest to our implementation, but are still (even for the planned speed-up of [3]) a factor of 4 slower. The presented implementation is currently the only one offering transmission of the complete configuration data over the same interface and at the full bandwidth as the pulse packets. It is also the only system where transmission integrity over all interfaces is ensured. This is especially important with respect to, e.g. pulse-based learning, which could be negatively influenced by erroneous pulse packets.

5. Discussion

A digital network system-on-chip (DNC) as part of a neuromorphic waferscale system [6] has been presented. It implements a communication infrastructure for packet-based address event representation (AER) transmission of neuron events (i.e. pulses). The DNC employs state-of-the-art full-custom circuitry for clock generation, high-speed data serialization and deserialization and off-chip data transmission using a low-voltage-differential-signalling (LVDS) scheme. Combined with digitally implemented algorithms for clock-to-data eye timing alignment, up to 2 GBit/s data rates per differential transmission line have been achieved, leading to a cumulative throughput of 32 GBit/s.

The DNC includes a sorting algorithm which can be employed to achieve configurable event delays, as required for various types of dynamical neuromorphic processing [16,17]. The measured performance of this priority queue algorithm achieves a competitive compromise between the different solutions in literature, combining low latency and high throughput required for the pulse routing application with the power and size reduction needed for a high integration density of the complete system (see Table 1). The clock generator supplying the DNC exhibits very low power consumption and requires significantly less chip area compared to similar solutions from literature (see Table 2). The jitter performance is somewhat worse than previously reported, but entirely sufficient for the transmission requirements. The power consumption of the presented LVDS transmitter and

Table 5
AER (i.e. spike transmission) performance figures of the presented DNC and comparable implementations. 'Partially' in the 6th column means that error detection as part of standardized interfaces such as USB has been implemented, but no error detection exists for the customized (e.g. serial) communication links. The entries for [5] are derived from the USB-AER board.

Ref.	Interfaces: event rate, pulse event size and type			Sum of all interfaces	Event error detection	Config. over AER	Topol. remap.	Additional functionality
	Host/PC	Inter-board	Neuro. chip					
DNC	625 kevent/s[a]	500 Mevent/s, 24 Bit, LVDS serial link	364 Mevent/s, 24 Bit, LVDS serial link	865 Mevent/s	Yes	Yes	No	Event sorting, configurable delays
[5]	6 Mevent/s, 16 Bit, USB2.0	25 Mevent/s, 16 Bit, parallel	25 Mevent/s, 16 Bit, parallel	56 Mevent/s	Partially	?	Yes	Video to/from event transform
[3]	Not implemented	41.7 Mevent/s, 20 Bit, MultiGBit transceiver	41.7 Mevent/s, 16 Bit, parallel	83 Mevent/s	No	Planned	No	Speed scaling possible to ca. 200 Mevent/s
[4]	5 Mevent/s, 64 Bit, USB2.0	78 Mevent/s, 32 Bit, serial ATA	30 Mevent/s, 16 Bit, parallel	113 Mevent/s	partially	Partially	Yes	Asynchronous flow control

[a] The entry for host communication for the DNC denotes the performance achievable via a backup JTAG configuration interface. In standard operation, the host bandwidth is equal to the inter-board bandwidth, i.e. the DNC is connected to a PC via an FPGA implementing the inter-board interface [8].

receiver cells is comparable to current literature, while the achieved data rate is among the highest (see Table 3). Also, the chip area required compares favorably to other similar LVDS implementations.

With respect to the overall waferscale neuromorphic system, the DNC-based communication infrastructure provides pulse stimulation and monitoring as well as full control and configuration of all the circuits on the wafer via the same links. This is in contrast to most classical AER implementations that are restricted to pulse transmission [50] or only offer partial configuration [4]. This feature is crucial since the DNC constitutes the sole interface between the neuromorphic wafer and the outside. Compared to other recent solutions, the full bandwidth of high-speed serial communications covers the entire chain from host PC via FPGA, DNCs down to the neuromorphic chips (HICANNs), avoiding a parallel AER bottleneck [4,3]. As Table 5 shows, the DNC SoC achieves a speed increase by an order of magnitude compared to the state-of-the-art.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under Grant agreement no. 269921 (BrainScaleS).

References

[1] T. Koickal, A. Hamilton, S. Tan, J. Covington, J. Gardner, T. Pearce, Analog VLSI circuit implementation of an adaptive neuromorphic olfaction chip, IEEE Transactions on Circuits and Systems I: Regular Papers 54 (2007) 60–73.
[2] S. Mitra, S. Fusi, G. Indiveri, Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI, IEEE Transactions on Biomedical Circuits and Systems (2009) 32–42.
[3] H. Berge, P. Häfliger, High-speed serial AER on FPGA, in: ISCAS 2007, 2007, pp. 857–860.
[4] D. Fasnacht, A. Whatley, G. Indiveri, A serial communication infrastructure for multi-chip address event systems, in: ISCAS 2008, 2008, pp. 648–651.
[5] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares Barranco, R.e.a. Paz-Vicente, 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 Transactions on Neural Networks 20 (2009) 1417–1434.
[6] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, S. Millner, A wafer-scale neuromorphic hardware system for large-scale neural modeling, in: ISCAS 2010, 2010, pp. 1947–1950.
[7] J. Schemmel, K. Meier, E. Mueller, A new VLSI model of neural microcircuits including spike time dependent plasticity, in: IEEE International Joint Conference on Neural Networks, IJCNN 2004, vol. 3, 2004, pp. 1711–1716.
[8] S. Hartmann, S. Schiefer, S. Scholze, J. Partzsch, C. Mayr, S. Henker, R. Schüffny, Highly integrated packet-based AER communication infrastructure with 3Gevent/s throughput, in: IEEE International Conference on Electronics, Circuits, and Systems ICECS10, 2010, pp. 952–955.
[9] J. Schemmel, J. Fieres, K. Meier, Wafer-scale integration of analog neural networks, in: IJCNN 2008, 2008, pp. 431–438.
[10] S. Joshi, S. Deiss, M. Arnold, J. Park, T. Yu, G. Cauwenberghs, Scalable event routing in hierarchical neural array architecture with global synaptic connectivity, in: 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), 2010, pp. 1–6.
[11] M. Lundqvist, M. Rehn, M. Djurfeldt, A. Lansner, Attractor dynamics in a modular network model of neocortex, Network: Computation in Neural Systems 17 (2006) 253–276.
[12] E. Izhikevich, J. Gally, G. Edelman, Spike-timing dynamics of neuronal groups, Cerebral Cortex 14 (2004) 933–944.
[13] S. Hill, G. Tononi, Modeling sleep and wakefulness in the thalamocortical system, Journal of Neurophysiology 93 (2005) 1671–1698.
[14] S. Scholze, S. Henker, J. Partzsch, C. Mayr, R. Schüffny, Optimized queue based communication in VLSI using a weakly ordered binary heap, in: MIXDES 2010, 2010, pp. 316–320.
[15] X. Jin, S. Furber, J. Woods, Efficient modelling of spiking neural networks on a scalable chip multiprocessor, in: IJCNN 2008, 2008, pp. 2812–2819.
[16] M.A. Dahlem, G. Hiller, A. Panchuk, E. Schöll, Dynamics of delay-coupled excitable neural systems, International Journal of Bifurcation and Chaos 19 (2009) 745–753.
[17] U. Meyer, J. Shao, S. Chakrabarty, S. Brandt, H. Luksch, R. Wessel, Distributed delays stabilize neural feedback systems, Biological Cybernetics 99 (2008) 79–87.
[18] D. Knuth, The Art of Computer Programming, Sorting and Searching, vol. 3, Addison-Wesley, MA, 1973.
[19] E. Mumolo, G. Capello, M. Nolich, VHDL design of a scalable VLSI sorting device based on pipelined computation, Journal of Computing and Information Technology 12 (2004) 1–14.
[20] K. Ratnayake, A. Amer, An FPGA architecture of stable-sorting on a large data volume: application to video signals, in: Proceedings of the 41st Annual Conference on Information Sciences and Systems, 2007, pp. 431–436.
[21] C. Thompson, The VLSI complexity of sorting, IEEE Transactions on Computers C-32 (1983) 1171–1184.
[22] K. McLaughlin, S. Sezer, H. Blume, X. Yang, F. Kupzog, T. Noll, A scalable packet sorting circuit for high-speed WFQ packet scheduling, IEEE Transactions on VLSI Systems 16 (2008) 781–791.
[23] A. Ioannou, M. Katevenis, Pipelined heap (priority queue) management for advanced scheduling in high speed networks, IEEE/ACM Transactions on Networking 15 (2007) 450–461.
[24] G. Burton, 16-Channel, DDR LVDS Interface with Per-Channel Alignment, Xilinx, Inc., 1.0 ed., 2006 〈http://www.xilinx.com/support/documentation/application_notes/xapp855.pdf〉.
[25] Y. Choi, D.-K. Jeong, W. Kim, J.-B. Lee, C.-H. Kim, An 1.4 Gbps/Ch LVDS receiver with jitter-boundary-based digital de-skew algorithm, in: IEEE Asian Solid-State Circuits Conference, ASSCC 2006, 2006, pp. 383–386.
[26] M.-D. Ker, C.-H. Wu, Design on LVDS receiver with new delay-selecting technique for UXGA flat panel display applications, in: Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS 2006, 2006, p. 4.
[27] C.-C. Wang, C.-L. Lee, C.-Y. Hsiao, J.-F. Huang, Clock-and-data recovery design for LVDS transceiver used in LCD panels, IEEE Transactions on Circuits and Systems II: Express Briefs 53 (2006) 1318–1322.
[28] K. Shu, E. Sanchez-Sinencio, CMOS PLL Synthesizers: Analysis and Design, Springer, 2005.
[29] A. Maxim, B. Scott, E. Schneider, M. Hagge, S. Chacko, D. Stiurca, A low-jitter 125–1250 MHz process-independent and ripple-poleless 0.18 μm CMOS PLL based on a sample-reset loop filter, IEEE Journal of Solid-State Circuits 36 (2001) 1673–1683.
[30] M. Toyama, S. Dosho, N. Yanagisawa, A design of a compact 2 GHz-PLL with a new adaptive active loop filter circuit, in: 2003 Symposium on VLSI Circuits, Digest of Technical Papers, 2003, pp. 185–188.
[31] S. Williams, H. Thompson, M. Hufford, E. Naviasky, An improved CMOS ring oscillator PLL with less than 4 ps rms accumulated jitter, in: Proceedings of the IEEE Custom Integrated Circuits Conference CICC 2004, pp. 151–154.
[32] M. Brownlee, P. Hanumolu, K. Mayaram, U.-K. Moon, A 0.5 to 2.5 GHz PLL with fully differential supply-regulated tuning, in: IEEE International Solid-State Circuits Conference, ISSCC 2006, Digest of Technical Papers, 2006, pp. 2412–2421.
[33] W. Yan, H. Luong, A 900 MHz CMOS low-phase-noise voltage controlled ring oscillator, IEEE Transactions on Circuits and Systems 48 (2001).
[34] C.S. Vaucher, Architectures for RF Frequency Synthesizers, Kluwer Academic Publishers, 2002.
[35] S. Höppner, S. Henker, R. Schüffny, A behavioral PLL model with timing jitter due to white and flicker noise sources, in: Proceedings of GMM/ITG-Fachtagung Analog, 2008, pp. 125–130.
[36] Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits, ansi/tia/eia-644 ed., 2001.
[37] S. Jamasb, R. Jalilizeinali, P. Chau, A 622 MHz stand-alone LVDS driver pad in 0.18 μm CMOS, in: Proceedings of the 44th IEEE Midwest Symposium on Circuits and Systems MWSCAS 2001, vol. 2, 2001, pp. 610–613.
[38] R. Kleczek, The design of low power 11.6 mW high speed 1.8 Gb/s stand-alone LVDS driver in 0.18 μm CMOS, in: Proceedings of the 17th International Conference on Mixed Design of Integrated Circuits and Systems MIXDES 2010, pp. 337–342.
[39] A. Tajalli, Y. Leblebici, A slew controlled LVDS output driver circuit in 0.18 μm CMOS technology, IEEE Journal of Solid-State Circuits 44 (2009) 538–548.
[40] F. Zhang, Z. Yang, W. Feng, H. Cui, L. Huang, W. Hu, A high speed CMOS transmitter and rail-to-rail receiver, in: 4th IEEE International Symposium on Electronic Design, Test and Applications, DELTA 2008, 2008, pp. 67–70.
[41] J.A. Tierno, A.V. Rylyakov, D.J. Friedman, A wide power supply range, wide tuning range, all static CMOS all digital PLL in 65 nm SOI, IEEE Journal of Solid-State Circuits 43 (2008) 42–51.
[42] G. Torralba, V. Angelov, V. Gonzalez, V. Lindenstruth, E. Sanchis, A VLSI for deskewing and fault tolerance in LVDS links, IEEE Transactions on Nuclear Science 53 (2006) 801–809.
[43] H. Eisenreich, C. Mayr, S. Henker, M. Wickert, R. Schüffny, A novel ADPLL design using successive approximation frequency control, Elsevier Microelectronics Journal 40 (2009) 1613–1622.
[44] R.J. Baker, CMOS Circuit Design Layout and Simulation, second ed., IEEE, Wiley Interscience, 2005.
[45] R. Nonis, E. Palumbo, P. Palestri, L. Selmi, A design methodology for MOS current-mode logic frequency dividers, IEEE Transactions on Circuits and Systems I: Regular Papers 54 (2007) 245–254.
[46] J. Maneatis, Low-jitter process-independent DLL and PLL based on self-biased techniques, IEEE Journal of Solid-State Circuits 31 (11) (1996).
[47] S. Höppner, R. Schüffny, M. Nemes, A low-power, robust multi-modulus frequency divider for automotive radio applications, in: MIXDES—16th International Conference on Mixed Design of Integrated Circuits & Systems, MIXDES'09, 2009, pp. 205–209.

[48] G. Cervelli, A. Marchioro, P. Moreira, A 0.13 μm CMOS serializer for data and trigger optical links in particle physics experiments, IEEE Transactions on Nuclear Science 51 (2004) 1–12.
[49] C. Zamarreno-Ramos, R. Serrano-Gotarredona, T. Serrano-Gotarredona, B. Linares-Barranco, LVDS interface for AER links with burst mode operation capability, in: IEEE International Symposium on Circuits and Systems, ISCAS 2008, 2008, pp. 644–647.
[50] M. Giulioni, Networks of Spiking Neurons and Plastic Synapses: Implementation and Control, Ph.D. Thesis, Universita degli studi di Roma Tor Vergata, 2008.

Stefan Scholze received the Dipl.-Ing. (M.Sc.) in Information Systems Engineering from Technische Universität Dresden, Germany in 2007. Since 2007, he has been a research assistant at the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits, Technische Universität Dresden, Germany. His research interests include design and implementation of low-latency communication channels and systems.

Holger Eisenreich received the Dipl.-Ing. (M.Sc.) in Electrical Engineering from Technische Universität Dresden, Germany in 2003. Since 2003, he has been a research assistant with the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include IC design methodology, logical and physical implementation of complex SoCs in sub-100nm technologies and energy efficient Multi-Processor SoCs with advanced power management features.

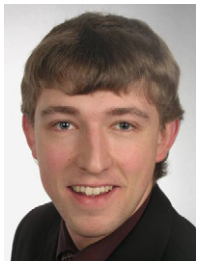
Sebastian Höppner received the Dipl.-Ing. (M.Sc.) in Electrical Engineering from Technische Universität Dresden, Germany in 2008. He is currently working as research assistant with the Chair for Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include circuit design for clocking, data transmission and power management in low power systems-on-chip as well as design methodologies for analog CMOS circuits.

Georg Ellguth received the Dipl.-Ing. (M.Sc.) in Electrical Engineering from Technische Universität Dresden, Germany in 2004. Since 2004, he has been a research assistant with the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include low-power implementation techniques in multi-processor system-on-chip.

Stephan Henker received the Dipl.-Ing. (M.Sc.) in Electrical Engineering in 1999 and his Ph.D. in 2005, both from Technische Universität Dresden, Germany. Since 1999, he has been a research assistant with the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include the design of algorithms as well as low power/high speed analog and mixed-signal VLSI.

Mario Ander received the Dipl.-Ing. (M.Sc.) in Electrical Engineering from Technische Universität Dresden, Germany in 2005. Since 2005, he has been a research assistant with the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include high-speed data transmission, charge-pumps and switched-capacitor-applications.

Stefan Hänzsche received the Dipl.-Ing. (M.Sc.) in Electrical Engineering from Technische Universität Dresden, Germany in 2006. Since 2006, he has been a research assistant with the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include circuit design of analog to digital converter and mixed signal simulation.



Johannes Partzsch obtained his M.Sc. in Electrical Engineering from Technische Universität Dresden, Germany in 2007. He is currently working on his Ph.D. in "optimized architectures for large-scale neuromorphic circuits" at the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits, Technische Universität Dresden, Germany. His research interests include bio-inspired circuits, topological analysis of neural networks and modeling of synaptic plasticity..

Christian Mayr received the Dipl.-Ing. (M.Sc.) in Electrical Engineering in 2003 and his Ph.D. in 'implementations of image processing in pulse-coupled neural networks' in 2008, both from Technische Universität Dresden, Germany. Since 2004, he has been a research assistant with the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include bio-inspired circuits, information processing in spiking neural nets and general mixed-signal VLSI-design. He is author or co-author of over 40 publications and has acted as reviewer for IEEE TCAS&TNN journals, Elsevier journals, MWSCAS and NIPS conferences.

Rene Schüffny received the Dr.-Ing (Ph.D.) and the Dr.-Ing. habil. (D.Sc.) degrees from Technische Universität Dresden, Germany, in 1976 and 1983, respectively. Since April 1997, he has held the Chair of Highly Parallel VLSI-Systems and Neuromorphic Circuits at Technische Universität Dresden. His research interests include CMOS image sensors and vision chips, design and modelling of analog and digital parallel VLSI architectures, and neural networks. He is author or co-author of numerous publications in the above field and has acted as reviewer for several international journals.