A QoS Network Architecture to Interconnect Large-Scale VLSI Neural Networks

Stefan Philipp, Johannes Schemmel and Karlheinz Meier

***Abstract*—This paper presents a network architecture to interconnect VLSI[1] neural network chips to build a distributed ANN[2] system. The architecture combines techniques from circuit switching and packet switching to provide two different service classes: isochronous connections and best-effort packet transfers. The isochronous connections are able to transport the axonal data of artificial neurons between VLSI ANN models that feature a speedup of multiples orders of magnitudes compared to biology. The connections use reserved bandwidth to provide loss-less transmissions as well as a low end-to-end delay with bounded jitter. Best-effort packet transfers use the remaining bandwidth for on-demand multi-purpose communication. The data forwarding is performed between synchronized instances of a dedicated switch architecture used at each network node. The switch is scalable in terms of port numbers and line speed. Its low complexity allows for an implementation within programmable logic or directly within a VLSI neural network chip. A reference implementation of the proposed network architecture is presented within an existing framework that hosts VLSI neural network chips operating at speedups of 10^4 to 10^5. The network architecture is further not limited to VLSI neural networks, but it can in principle be used in all network environments that require isochronous connections as well as packet processing.**

***Index Terms*—VLSI neural networks, quality of service, isochronous communication, crossbar switches, AER**

I. Introduction

VLSI implementations of ANNs are an interesting alternative to the modeling of neural networks in software. In VLSI neural networks, biological characteristics of neurons and synapses like membrane potentials and ion channels are represented by electrical counterparts like voltages, currents and conductances. An overview of VLSI implementations of neural networks can be found in [1]. Implementations by our research group have been reported in [2], [3].

The chips in [2], [3] use a mixed-signal design, that is, analog and digital circuit techniques are combined. Analog design allows for a compact continuous-time implementation of the neuron and synapse circuits, whereas digital design is well-suited for configuration and external communication. The chip in [3] integrates 384 leakage integrate-and-fire neurons [4] with 256 synaptic inputs on a single die (cf. Figure 1). The small area used by the neurons leads to small time constants and thus to an operational *speedup* of the membrane time constants of 10^4 to 10^5 compared to biology.

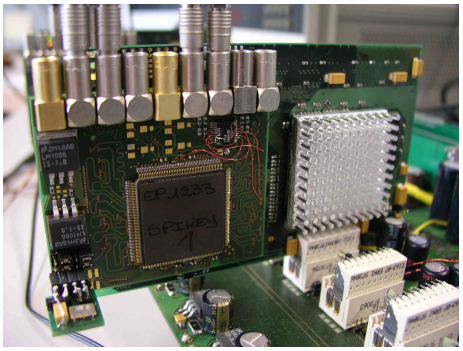


Fig. 1. A network module equipped with the VLSI ANN of [3] on the left.

Action potentials (or *spikes*) are created as digital events in continuous time. Synaptic connections between the neurons can be implemented directly on-chip. In addition to this, a digital external interface allows to interconnect multiple chips to a large-scale neural network by using a dedicated *communication network*. These inter-chip connections then represent long-range axonal connections of biological neurons. In contrast to VLSI neural networks that operate at biological speeds, chips like the latter have strong QoS[3] [5] requirements in terms of throughput, delay and delay variation (or *jitter*) on the connections provided by the communication network.

The expected data rates depend on the encoding of the spike events and on the estimated spike frequencies, which in turn depend on the neural network configuration. As a calculation example, a modeled neuron with a biological mean firing rate of 10 Hz at a speedup of 10^5 causes a physical spike event rate of 1 MHz. The spike event rate can be even higher in the case of bursting. Example bandwidth requirements for a single neuron and a whole chip are given in Table I.

The transmission of spike events via a connection network will result in end-to-end delays that are significantly larger than if using ANN-internal connections. Besides the pure throughput, the communication network therefore has to guarantee an upper bound to the delay of the inter-chip connections to model axonal connections with biologically relevant timings. As an example, a physical delay of only 100 ns at a speedup of 10^5 corresponds to a biological axonal delay of 10 ms. The timing requirements on the interconnection network become even more critical within a large distributed

Stefan Philipp, Johannes Schemmel and Karlheinz Meier are with the Kirchhoff-Institut für Physik, Ruprecht-Karls-Universität Heidelberg, INF 227, 69120 Heidelberg, Germany (email: sphilipp@kip.uni-heidelberg.de).

The work presented in this paper is supported in part by the European Union under the grants no. IST-2001-34712 (SenseMaker) and no. IST-2005-15879 (FACETS).

[1] Very Large Scale Integration

[2] Artificial Neural Network

[3] Quality-of-Service

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

TABLE I
EXAMPLE DATA RATES FOR THE CHIP OF [3] WITH MEAN RATES OF 1 MHZ AND PEAKS AT FOUR TIMES LARGER RATES. SPIKE EVENTS ARE ENCODED WITH 32 BITS. [7]

		ANN neuron	ANN chip
Spike event rate	average	1 MHz	384 MHz
	peak	4 MHz	1536 MHz
Inter-chip connection	average	4 MB/s	1.5 GB/s
	peak	16 MB/s	6.1 GB/s

system in which not all chips are directly connected, but spike events have to be transported via multiple intermediate hops. Since the timing of incoming spike events is used in plasticity models like STDP[4] [6], the transmission delay should further be as constant as possible (i.e. with low jitter). Although end-to-end jitter can be reduced using buffers at the destination, this increases the overall delay, complicates the VLSI design and reduces the space left for neuron and synapse circuits.

A widely used protocol for the transport of spike events is the *AER*[5] protocol [8], which makes use of the fact that electronic communication is several orders of magnitudes faster than the firing rates of biological neurons. AER uses a handshake mechanism and preserves the asynchronous timing of the spike events. However, asynchronous handshakes are not suitable for inter-chip communication at high speedups. The physical firing rates of the neurons and the communication delays between the chips would lead to unrealistic biological uncertainties of the spike event delivery between the neurons in a large and distributed setup [3].

This paper presents a digital and synchronous network architecture suitable for the interconnection of VLSI neural network chips at high speedups. The presented architecture operates as a separate communication layer with QoS control techniques. To be more precise, it provides *isochronous* end-to-end connections between the chips, that is, independent communication channels with guaranteed throughput and small, nearly constant delay with minimum jitter for the transport of axonal neuron data. Since complex distributed systems often have additional transport requirements for on-demand multi-purpose data (e.g. synaptic weights or similar configuration data), the network architecture provides an additional traffic class for best-effort packet-based communication without QoS guarantees by using the remaining bandwidth of the physical links. The integration of the two traffic classes is performed by the bypass-switch, a dedicated switch architecture that combines principles of circuit switching and packet switching in a single design.

The paper is organized as follows: Section II first describes the overall architecture of the network. The transport of priority and best-effort traffic is presented separately. This is done in a more general way since the target applications of the network are not limited to VLSI neural networks. Section III then describes the central bandwidth reservation algorithm in

[4]Spike-Time Dependent Plasticity
[5]Address Event Representation

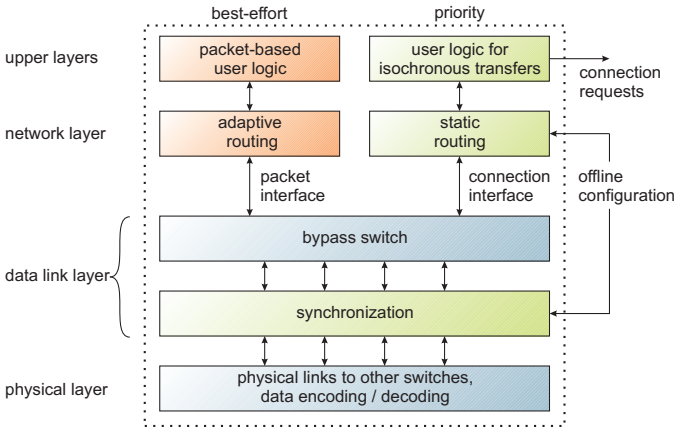


Fig. 2. Overview of the proposed network architecture.

detail and section IV discusses the theoretical performance and scalability of the network architecture. Section V presents a reference implementation of the proposed architecture within an existing framework that interconnects a larger number of the VLSI neural network chips of type [3].

II. NETWORK ARCHITECTURE

A. Overview

Figure 2 illustrates an overview of the proposed network architecture. It consists of multiple parts: (A) A framing strategy of the physical link data. (B) A synchronization and timing sublayer. (C) The bypass-switch multiplexing the traffic classes. (D) Interface specifications for upper-layer access to isochronous and packet-based data. (E) A bandwidth reservation mechanism, which is executed offline prior to the operation of the network. The main idea of the network is to compute a compact reservation pattern for all isochronous connections offline prior to the network operation. The calculated pattern is then written into routing tables for priority traffic within the switch to simplify the online forwarding process. Best-effort packets are handled purely online and use the remaining bandwidth.

The central part of the proposed architecture is the bypass-switch, which integrates the two traffic classes. It contains one or multiple global ports to the physical links (multiplexing both classes) as well as local ports to upper-layer user logic (separated for each class). To simplify the discussion, the following description of the switch only considers multiplexed ports without a limitation of generality.

Figure 3 shows a schematic design of the switch. It is based on an input-buffered crossbar switch, but uses a queuing bypass for priority data at each input. While best-effort data is stored into queues, priority data is forwarded immediately *without* any queuing. An N-Port bypass-switch has N input ports and N output ports. There is no internal speedup, that is, each internal data path operates at the same external line rate. The central crossbar has $2N \times N$ ports and multiplexes data of both traffic classes. By doubling the size of the crossbar, the switch removes conflicts at inputs between priority and

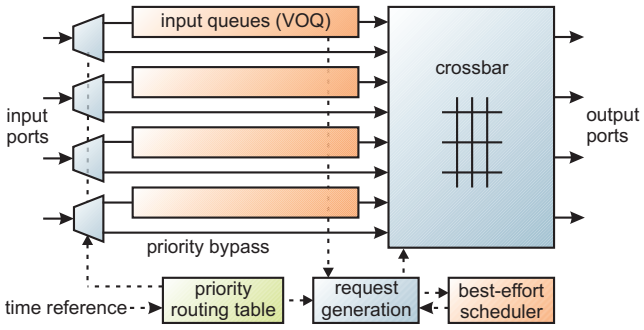


Fig. 3. High-level view of a 4-port bypass-switch. Incoming best-effort data is stored in input queues whereas priority data bypasses all queuing. Best-effort traffic uses unreserved or unused priority bandwidth.

best-effort data with moderate additional costs. The space requirement of the crossbar is still of O(N^2) complexity.

The timing of the switch is induced by the underlying synchronization sublayer, which synchronizes all switches of the network. The bandwidth of the physical links is framed by dividing the time axis into equally-sized *time frames*. Each time frame further consists of a synchronization character plus f *time slots*, where f is a globally constant number (cf. Figure 4). Bandwidth is assigned on a time-slot basis such that the slot data either belongs to isochronous priority data or to a best-effort packet. The switch operates according to the slotted timing. Each time slot, the crossbar transfers the corresponding amount of data in parallel from its inputs to its outputs.

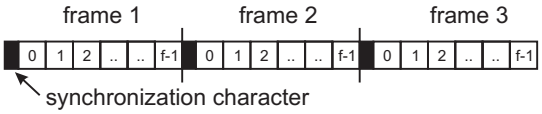


Fig. 4. The bandwidth of each link is divided into equally-sized frames with the globally constant number of f time slots.

The crossbar itself is controlled by a request logic computing the port assignments. Each time slot, the request logic first denotes reserved input and output ports according to the priority routing table. Unused ports are further scheduled for best-effort traffic in a second step according to the queue occupancies. Note that priority data (of isochronous connections) is forwarded without investigating the data content. Since this avoids the usage of slot headers, it allows very tiny slots sizes that transport only few bytes, which is a major advantage of the switching scheme. The hierarchical design of the switch allows to further discuss the forwarding process for both traffic classes separately.

B. Isochronous Priority Transfers

The synchronization sublayer provides a global synchronization of all switches with the precision of a single time slot. It is assumed, that synchronization can be established depending on the application environment of the network, e.g. in embedded environments or SoCs[6], synchronization can use a global reference clock. Successful synchronization can be verified via the arrival times of the synchronization character of each frame. Due to physical delay variations, the offset numbers of time slots entering a switch may have a constant shift between different ports at the same time. This can be resolved by a logical renumbering scheme at each switch. Furthermore, f can be selected to an arbitrary large number independent of the physical link delays to minimize the bandwidth loss due to the synchronization character [7]. To simplify the discussion, it is assumed that frames at all ports of all switches arrive aligned without a limitation of generality. The minimum bandwidth waste due to the synchronization character is ignored in the following.

The set of all isochronous connections to be established has to be known a priori or has to remain unchanged for a long time compared to the operation of the network. The reservation pattern is then calculated by a global *bandwidth mapping algorithm*, which is executed initially at the time the network is set up. For each connection, the algorithm assigns one or multiple time slots at all links of the route between the two end-points according to its bandwidth requirement. The slots are reserved exclusively based on a periodic scheme such that no link is overbooked. The corresponding port and slot reservation scheme is stored in the priority routing tables of the switches.

The deterministic reservation of slots eases the forwarding process: After priority data enters the switch, its output port is identified only by its time slot according to the synchronized time. No header processing of slot data is performed and the full bandwidth of reserved connections is available for payload data. Furthermore, the forwarding process is of only O(1) complexity, which allows the usage of small time slots and high line rates. Note that the throughput is guaranteed and the forwarding delay is reduced to the small and constant value of the traversal time of the crossbar and the physical layer of the switch, i.e. no jitter is introduced at intermediate switches.

The mapping algorithm further resolves the usual problem of *contention* that arises at the output ports of typical switch architectures that are based on slotted timing [9], [10], [11]: Since an output can accept data from a single input only during each time slot, the case that data from multiple inputs should be forwarded to the same output results in blocked input ports or data drops. In contrast to this, the solution presented in this work removes contentions with an appropriate reservation scheme and also avoids any buffering. The connections are mapped to time slots in a way that contention is avoided and that the data can be forwarded immediately without any buffering and with minimum delay (cf. Figure 5).

Due to the reservation of bandwidth, the network is not capable of handling excess data for connections, the amount of data injectable into a connection by the user logic is limited. Comparable to [10], a slot *admission policy* is needed to regulate the traffic at the user interface of the source network node of a connection: The switch denotes the upper layer when reserved slots became available for transmission. The resulting waiting process is the only point where end-to-end

[6]System on Chip

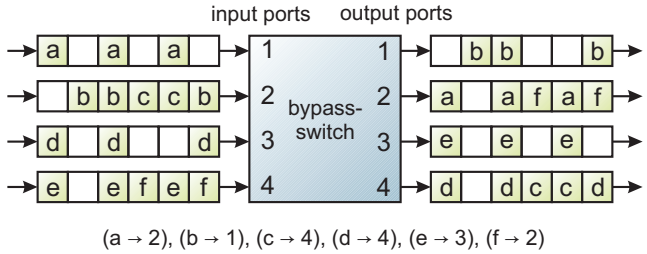


Fig. 5. Contention resolution of priority traffic. Example of a 4-port switch with 6 time slots per time frame ($N = 4$, $f = 6$). Reserved slots belong to connections a-f. Output contention is removed by the connection reservation pattern according to the local routing table (ports written below the switch). No buffers are needed for incoming priority data.

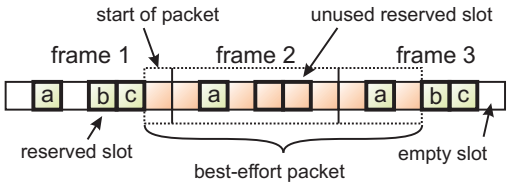


Fig. 6. Embedding a best-effort packet of 8 slots into successive time frames on the same link. Slots 2, 4 and 5 are reserved for priority data. The packet uses unreserved and unused priority slots and is extended over several time frames.

connection jitter is introduced. The mapping process takes care of connections having extreme jitter requirements and distributes time slots of such connections more equally among the frames. In the case of bursts, a traffic shaping algorithm like leaky bucket [12] can be used to smoothen the data at the source node.

Note that the network has to be reconfigured if the connection requests change. This depends on the application. Since a new reservation can be calculated in parallel during the network operation, a synchronous update of all routing tables can be used.

C. Best-Effort Packet Transfers

Best-effort traffic is transferred as in packet switched networks, i.e. its bandwidth requirements are assumed to be unknown and may change during the operation of the network. Requests for packet transfers may arise on-demand, a pre-calculated mapping is not required. Best-effort packets are embedded in unreserved or unused priority slots. Since time slots are assumed to be small, a best-effort packet may occupy multiple time slots and can be extended over several time frames (cf. Figure 6). The packets contain an additional header to allow global routing.

The switch architecture concerning best-effort traffic equals an input-buffered crossbar switch with VOQs[7]: Packets entering the switch are stored in separate FIFO[8] queues for each output at each input (cf. Figure 7). Incoming packets are collected out of the priority slots depending whether slots are used for priority traffic or not. The header of each packet is

[7]Virtual Output Queuing
[8]First-In First-out

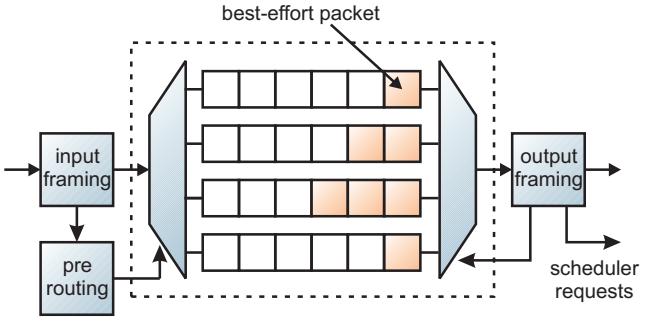


Fig. 7. Virtual output queues at a single input of a 4-port switch. Packets are pre-routed and stored in separate queues for each output. The framing logic extracts and inserts best-effort traffic into the slotted time frames. Only a single dual-port memory element is needed at each input (depicted with the dotted line) and no speedup is required.

looked up and a routing algorithm computes the local output. The packet is then stored in the appropriate VOQ. Note that only one queue is written and read at the same time. Therefore, all N^2 queues for an N-port switch can be implemented with only N dual-port memory elements and N^2 FIFO controllers.

In each time slot, the request generation logic of the switch collects the requests from all currently unmatched VOQs for which inputs and outputs are not used by priority traffic in the *following* time slot. The best-effort scheduler then calculates a bipartite matching of inputs and outputs within the duration of a time slot. At the time the scheduler completes, the framing logic at the queue outputs is informed of the scheduling result. Since packets are not merged within the time frames, the scheduling result for a dedicated output remains valid until a packet has been transmitted completely.

The scheduler used for best-effort traffic can in principle be any scheduler which is usable for virtual output queuing to calculate a bipartite matching, e.g. iSlip [13], [14], RPA or DPA [15] etc. The duration of a time slot imposes an upper bound on its complexity.

The interface to the upper-layer logic is slightly more complex than for isochronous data. Packets are transmitted slot by slot at the source node and re-assembled at the destination. Furthermore, the switch forwards the online routing requests to the upper-layer. This allows to implement different packet types with even different routing strategies at the same switch.

III. Mapping Algorithm

The bandwidth mapping algorithm computes the contention-free slot reservation pattern of the isochronous connections prior to the network operation. The challenge to perform a contention-free forwarding and to guarantee QoS for the connections has therefore been moved from online to an offline pre-calculation. The algorithm is a central part of the network architecture. It may be executed in software, since the isochronous connections are assumed to remain unchanged for a long time compared to the operation of the network.

To be more formally, the topology of the network is represented by a graph $G = G(V, E)$ with V comprising the upper layer processes and switches of the network and E the

set of its interconnects (upper-layer switch ports and physical links). The mapping algorithm processes the n requests for the isochronous connections $c_1, ...c_n \in C, c_i = (v_s, v_d, w_c)$ with $v_s, v_d \in V$ and $w_c \in \mathbb{R}, 0 < w_c < 1$ the bandwidth requested by the connection c. The algorithm tries to find a valid slot assignment at each link $e \in E$ for a given period m. The synchronized framing requires m to hold $f = m \cdot n, n \in \mathbb{N}$, but f can selected to be large due to the logical renumbering of frames at each switch input (cf. section II-B). A possible algorithm consists of three steps: (A) The assignment of a number of time slots to each connection according to its bandwidth requirement. (B) The routing of the connections between its end-nodes. (C) The calculation of the contention-free periodic slot assignment.

The first step is basically a quantization process. For each connection request $c \in C$ with bandwidth requirement w_c, it calculates the number of data slots $m_c \in \mathbb{N}$ to be reserved as the smallest integer number that holds:

$$\frac{w_c}{w_f} \leq \frac{m_c}{m} \leq 1, \tag{1}$$

where $w_f \in \mathbb{R}, 0 < w_f < 1$ is the total bandwidth usable for data per link. In other words, the requested bandwidth is rounded up to the next possible value according to the time slot granularity imposed by the global framing scheme. The algorithm does not calculate the exact positions of the reserved slots, but only the number of slots to be reserved within each reservation period exclusively for that connection on the whole route from the connections source node to its destination.

The second step routes all connections one by one by respecting the available bandwidth per link. The routing of a single connection can be done using the *shortest-path* algorithm from Dijkstra [12] with a slight modification. The algorithm uses weights that are assigned to the links $e \in E$ to represent its already-assigned network load. After a particular connection c has been routed, the weights of the affected links are increased according to its number of time slots m_c. During the processing of the algorithm, the link weights increase and paths for connections to be routed depend on the routing of previous connections such that succeeding connections are routed around heavily loaded edges and use free edges first. Doing so, the algorithm ensures that each reservation period is booked with not more than m time slots. In the case that the Dijkstra algorithm fails, the set of connection requests is rejected. New requests with less bandwidth requirements or a finer slot granularity with larger values m can be made.

The third step of the algorithm computes the contention-free assignment. It is solved by transforming the assignment problem to the *vertex-color* problem [16] from graph theory. Without limitation of generality, it is assumed that $m_c = 1$ for each connection (connections with multiple time slots can be modeled by an appropriate number of parallel connections using the same route). Due to the deterministic forwarding process, the reserved slot positions at intermediate links of a connection's route are determined just by the position of the reserved slot at the first link. The assignment problem can therefore be reduced to the calculation of the position q_c of the time slot at the first link for each connection c (the first link will be an upper-layer input port of the switch). This problem is solved with an undirected *collision graph* $G' = G'(V', E')$. The graph consists of n vertices v_i, one for each connection c_i. The graph edges $e' \in E'$ denote the traffic constraints of the local forwarding processes: An edge $e' = (v'_1, v'_2)$ is created between v'_1 and v'_2 in the case that the routes of c_1 and c_2 share a link $e \in E$ (cf. Figure 8 and Figure 9).

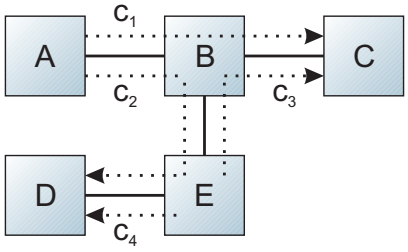


Fig. 8. An example network of five nodes A to E with four connection requests c_1 to c_4 depicted with dotted lines between the nodes.

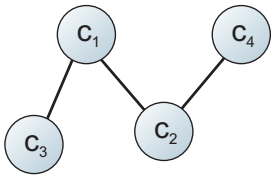


Fig. 9. The connection collision graph of Figure 8.

The slot assignment problem now equals the problem of assigning different numbers q_c for vertices v' that are connected by an edge e', i.e. to find a vertex coloring of G'. Vertex coloring is a highly investigated problem and a couple of algorithms exist [16]. The difficulty to find a valid coloring clearly depends on the connection requests, the topology of the network and the maximum number of available "colors" i.e. the m time slots per reservation period. The problem is more likely being solved for large numbers m with the drawback of increased connection jitter.

IV. Theoretical Performance

A. QoS Guarantees for Priority Traffic

Since priority traffic is transported within reserved slots without contentions, no data is dropped and 100 % throughput is guaranteed during the transportation process. Connection data arriving at the switch is forwarded without buffering and with O(1) online complexity. Its delay is caused mainly by the physical layer and by the traversal of the crossbar. Therefore, the transport delay is small, constant and the jitter is independent of the number of network hops. Variation of the overall connection delay is introduced only at the source node during the waiting process for reserved slots. The amount of introduced jitter depends on the distribution of reserved slots within a reservation period. A period of m slots ensures that the jitter is bounded by m time slots in the worst case. If

constant overall delay (zero jitter) is required, a jitter buffer at the end node can be used. Its size is bounded according to the jitter bounds. The latter setup requires additional timing information to be sent along with priority data.

B. Performance of Best-Effort Traffic

The transport of best-effort packets comprises the fragmentation to slots, the re-assembly, the routing and the scheduling. Since the routing can be implemented via table look-ups within the upper layer, only the scheduling process is of significant complexity. The scheduling has to be performed within a single time slot. Different schedulers are conceivable: maximum-size matching (MSM) is known to achieve near optimum results, however is not useful due to its high complexity of $O(N^{2.5})$. iSLIP [13] is very common in literature and easy to implement in hardware. RPA and DPA [15] are also suitable to be implemented in hardware and perform well. Note that at high reservation rates, only few slots are available for best-effort traffic. In this case, the remaining scheduling tasks becomes simple and a complex scheduler is not required [7].

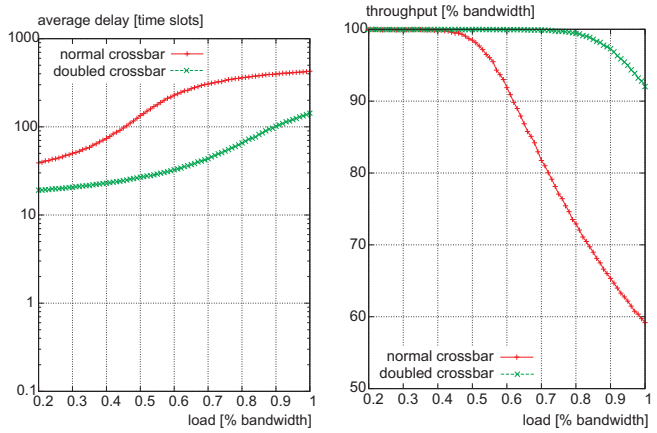
C. Scalability and Complexity

Concerning isochronous connections, the online scheduling algorithm is of low complexity and a large number of ports can be used. In the case that only isochronous traffic is required, the complexity of the switch reduces to a single crossbar with a static periodic routing table as in [17]. Scalability in the line speed is limited by the best-effort scheduler. If the line speed increases, the duration of a single time slot can be increased accordingly. This is important for network environments where transmission techniques may be updated at times.

The scalability in the number of network nodes depends on the difficulty to find a valid mapping for the isochronous connections with a sufficiently small value of m. This further depends on the particular network topology and the set of connection requests of the application. The complexity of the computation can be reduced by combining requests for multiple connections between the same two network nodes into a single isochronous connection. This may further balance the network load during operation and allows for a more tight bandwidth assignment. Section V-A presents mapping results for a different number of VLSI neural network chips within an existing framework.

Concerning best-effort traffic, the switch behaves as a common input-buffered switch with VOQs. Scalability results and QoS guarantees depend on the scheduling algorithm used.

The space complexity of the $2N \times N$ crossbar that multiplexes the two traffic classes is of $O(N^2)$. Although this doubles its size, it also improves the throughput for best-effort traffic. To see this, the switch has been compared in simulation against an typical architecture that uses a single $N \times N$ crossbar and thus suffers from contention between the traffic classes at each input. The simulation uses a 5-port switch with a packet size of 11 slots and 40 % of reserved slots. Incoming best-effort traffic has been simulated with independent statistical arrivals at each port with a certain arrival property for each time slot. Figure 10 shows the results. It can be seen that by doubling the inputs of the crossbar, contention at the input ports between both traffic classes is eliminated and the average delay as well as the throughput of best-effort packets is significantly improved.



(a) average delay (b) throughput

Fig. 10. Comparison of scheduling results for switch designs with and without separate crossbar inputs for the traffic classes. It can be seen that the bypass-switch architecture outperforms the standard design.

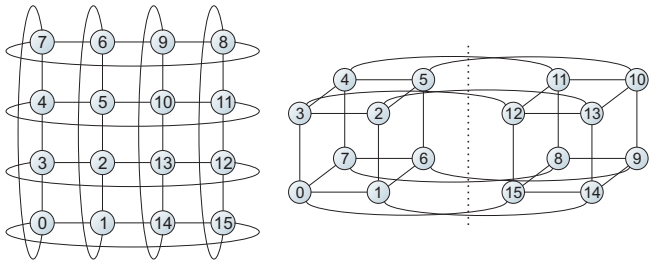
V. Reference Implementation

This section presents the reference implementation of the described network architecture within the ANN hardware and software framework of [18], [19]. The framework consists of backplanes that hold up to 16 *network modules* each. A backplane fully equipped with 16 ANNs of type [3] provides 6144 artificial spiking neurons with 1.57 million synapses.

Figure 1 shows a photograph of a network module. It can be equipped with a single VLSI neural network chip and comprises up to 1 GByte of local memory and further support components like DACs[9] and ADCs[10]. The local ANN chip and all other components on the module are interfaced by a central Xilinx Virtex-II Pro FPGA[11] [20], which also provides the physical interconnects to other modules. The programmable logic of the FPGA contains the interface code of the ANN chip as well as the online part of the reference implementation of the proposed network architecture.

The physical layer of the network is realized by eight FPGA-internal multi-gigabit transceivers [21], which operate at 3.125 Gbit/s external line rate. The overall end-to-end delay of the transceivers has been reduced to 128 ns by applying the modifications from [22]. Four of the transceivers are hardwired by the backplane such that each network module is directly physically connected to four others. The resulting topology equals a two-dimensional toroid and also a four-dimensional binary cube (cf. Figure 11). The four remaining connectors can be used to add additional connections with cables between the modules or to interconnect multiple backplanes.

[9] Digital-to-Analog Converter
[10] Analog-to-Digital Converter
[11] Field Programmable Gate Array



(a) two-dimensional toroid (b) four-dimensional binary cube

Fig. 11. The hardwired topology of a backplane illustrated as toroid and cube. Each node hosts a single ANN chip as well as an FPGA, which accesses the physical interconnects. Nodes not directly connected communicate via intermediate nodes.

A. Isochronous Priority Transfers

Isochronous connections are used for the modeling of inter-neuron axonal connections between the ANN chips. For that reason, the ANN chip contains a digital control part that is able to transmit spike events of neurons to its external interface and to receive incoming events. A spike event is transmitted within 32 bits, which is the time slot size of the network. The slot duration is 12.8 ns. The ANN chip requires a dedicated controller logic within the FPGA to be interfaced to the communication network [23]. Prior to an experiment, the user has to define the required set of the inter-chip isochronous connections and their bandwidth demands depending on the physical neuron placement and the synaptic connections of the neural network to be modeled. The transport of multiple inter-neuron connections within a single isochronous connection reduces the amount of bandwidth to be reserved for it to cope with bursting of single neurons.

It has been shown in section II-B that the reliability of the isochronous connections depends on a successful synchronization of all switches. The implemented synchronization uses a global clock reference and is executed once during network startup. Adjustable delay elements at the output ports of the switches compensate different physical transmission delays with a precision of 6.4 ns. The stability of the synchronization has been verified with a dedicated test-setup that uses measurement circuits at the end-points of the isochronous connections.Test spike events have been injected into the connections and the accurate timing of its arrival at the end-points has been verified. Once established, the synchronization has proven to be stable during all tests. A final end-to-end delay of only 154 ns has been achieved for connections between neighbored modules.

Although the throughput of the isochronous connections is guaranteed by design, the end-to-end jitter depends on the minimum possible value m of the periodic assignment and thus on the performance of the mapping process. A reference implementation of the algorithm of section III has been developed in software. The performance of the implemented algorithm has been analyzed against a set of pseudo-random neural networks of a regular binary cubic topology with $k = 2^d$ nodes for different dimensions d (cf. Figure 11(b)). Each network node consists of an ANN that uses all 384 neurons and is connected to d others. The 2×256 synaptic inputs of each chip are equally distributed to neuron outputs of all chips, such that the inter-chip network has to implement $k(k-1)$ aggregated connections. Each connection is assigned a single slot, the jitter therefore equals the number m of time slots per reservation period times the slot duration of 12.8 ns.

TABLE II
MAPPING RESULTS FOR REGULAR NETWORKS OF $k = 2^d$ NODES IN A d-DIMENSIONAL BINARY CUBIC TOPOLOGY (SEE TEXT).

nodes	links	max hops	connections	jitter [slots]	occupancy
4	8	2	12	2	100 %
8	24	3	56	4	100 %
16	64	4	240	8	100 %
32	160	5	992	20	80.1 %
32	160	2	480	7	71.4 %

The results of the bandwidth mapping are shown in table II. The first four lines show results for networks that contain inter-chip connections between all nodes as described above. The last line corresponds to a network whose connections are limited to a distance of two network hops. The slot occupancy denotes the percentage of time slots the algorithm succeeded to use. It can be seen that the mapping algorithm is able to occupy 80 % of all available slots even with nearly 1000 isochronous connections (with multiple inter-neuron connections each). Furthermore, by avoiding far hops with large delays, the number of connections decreases and the jitter per connection improves significantly. Note that 100 % slot occupancy of the hardwired links between the 16 chips of a fully equipped backplane allows for an overal rate of up to $5 \cdot 10^9$ spike events/s to be transmitted between the chips.

B. Best-Effort Packet Transfers

Best-effort packets are used by a global shared memory subsystem implemented within the programmable logic. Its task is to transfer multi-purpose data such as configuration data, control data or synaptic weights between the memory chips on the modules. Each packet carries 32 bytes of memory data and occupies 11 slots (3 slots for header information). The implemented routing algorithm performs a simple dimensional routing and is of low complexity.

The VOQs at each input port are implemented with FPGA-internal dual-port memories (only a single memory block is needed at each port). Two different best-effort schedulers have been implemented in hardware, namely iSLIP [13], and DPA [15]. The schedulers perform a matching decision every 12.8 ns. Table III shows the amount of logic needed for different switch configurations.

VI. CONCLUSION

This paper presents a network architecture feasible to interconnect VLSI neural network chips to large-scale neural networks using digital communication techniques. The data of axonal connections between neurons on different chips

TABLE III
SLICE USAGE FOR THE BYPASS-SWITCH IMPLEMENTATION AT DIFFERENT CONFIGURATIONS. PORT NUMBERS ARE FOR GLOBAL PORTS, LOCAL ISOCHRONOUS PORTS AND LOCAL BEST-EFFORT PORTS. THE PERCENTAGE VALUES ARE GIVEN FOR A XILINX VIRTEX-II PRO XC2VP7 FPGA [20]

Ports GL PR BE	Scheduler	Memory Blocks	4-input look-up tables
4 + 1 + 1	iSLIP-1	5	1621 (16 %)
4 + 1 + 1	DPA	5	1751 (18 %)
6 + 1 + 1	iSLIP-1	7	3338 (34 %)
6 + 1 + 1	DPA	7	3411 (35 %)

is transported with isochronous connections, which provide QoS in terms of guaranteed throughput, low delay and low jitter. The low delay and jitter values allow for an operation of VLSI ANN models at speedups of multiple orders of magnitudes. Multi-purpose data such as configuration data, control data or synaptic weighs is transported on-demand with best-effort packets using the same physical resources and without provision of QoS.

The network architecture is based on a slotted timing scheme. It combines connection-based transfers and packet-based transfers by means of global synchronization, a global bandwidth reservation algorithm and the dedicated bypass-switch architecture. In contrast to other slotted timing schemes, its deterministic arrival pattern releases from investigating the slot content, which leads to high speeds and a small slot size for fine-grained bandwidth reservations. Due to its low online complexity, the network architecture is scalable in terms of the number of network hops, port numbers and line speed.

A reference implementation of the proposed network architecture in programmable logic and software within an existing ANN framework has been presented. The framework comprises VLSI ANN models with a speedup of 10^4 to 10^5 on backplanes with 16 units each. The proposed reservation algorithm succeeds to compute compact reservation patterns of 71 % to 100 % slot occupancy for different network sizes and is able to reserve bandwidth even for networks with nearly 1000 isochronous connections. The stability of the framing and synchronization has been verified. An implemented switch with 4 global ports and a single local connection occupies only 1621 of the 4-input look-up tables of the FPGA used. This shows the feasibility of the architecture for embedded systems and SoCs. More generally, the network architecture can be used in all environments with the need for isochronous connections and packet transmissions.

In particular, the network architecture is suitable to be the top-level network for the next stage of the framework, which uses wafer-scale integration of neural networks [24], [25]. Isochronous connections are required to interconnect 20 cm wafers with about 50 million synapses each. Due to its low online complexity and due to its scalability in terms of line speed and the number of network hops, the proposed network architecture can be scaled to this application without major changes.

REFERENCES

[1] L. Reyneri, "Implementation issues of neuro-fuzzy hardware: Going toward HW/SW codesign," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 176–194, January 2003.

[2] J. Schemmel, S. Hohmann, K. Meier, and F. Schürmann, "A mixed-mode analog neural network using current-steering synapses," *Analog Integrated Circuits and Signal Processing*, vol. 38, no. 2-3, pp. 233–244, 2004.

[3] J. Schemmel, A. Grübl, K. Meier, and E. Mueller, "Implementing Synaptic Plasticity in a VLSI Spiking Neural Network Model," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, IEEE Press (2006).

[4] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[5] S. N. Bhatti and J. Crowcroft, "QoS-Sensitive Flows: Issues in IP Packet Handling," *IEEE Internet Computing*, vol. 4, no. 4, pp. 48–57, 2000.

[6] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[7] S. Philipp, "Design and Implementation of a Multi-Class Network Architecture for Hardware Neural Networks," Ph.D. dissertation, Universität Heidelberg, 2008.

[8] A. Mortara and E. Vittoz, " A communication architecture tailored for analog VLSI artificial neural networks: intrinsic performance and limitations," in *IEEE Trans. on Neural Networks*, vol. 5, 1994, pp. 459–466.

[9] A. Hung, G. Kesidis, and N. McKeown, "ATM input-buffered switches with guaranteed-rate property," in *Proc. of IEEE ISCC'98, Athens*, 1998, pp. 331–335.

[10] S. Li and N. Ansari, "Input-queued switching with QoS guarantees." in *Proceedings of IEEE INFOCOM'99*, New York, 1999, pp. 1152–1159.

[11] T. Felicijan, "Quality-of-Service (QoS) for Asynchronous On-Chip Networks," Ph.D. dissertation, University of Manchester, Dept. of Computer Science, 2004.

[12] A. S. Tanenbaum, *Computer Networks*. Pearson Education Int., 2004.

[13] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," IEEE/ACM Trans. Networking, vol. 7, no. 2, pp. 188-201, Apr. 1999., 1999.

[14] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, "Tiny Tera: A packet switch core — using new scheduling algorithms to build a 1-terabits packet switch with a central hub no larger than a can of soda," *IEEE Micro*, vol. 17, no. 1, pp. 26–33, /1997.

[15] J. Hurt, A. May, X. Zhu, and B. Lin, "Design and implementation of high-speed symmetric crossbar schedulars," In IEEE Int. Conf. on Communications, June 1999, 1999.

[16] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[17] S. Philipp, A. Grübl, K. Meier, and J. Schemmel, "Interconnecting VLSI Spiking Neural Networks Using Isochronous Connections," in *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN'2007)*, vol. 4507, San Sebastián, Spain, Sept. 2007, pp. 471–478.

[18] A. Grübl, "Eine FPGA-basierte platform für neuronale netze," Diploma thesis (german), University of Heidelberg, HD-KIP-03-2, 2003.

[19] J. Fieres, A. Grübl, S. Philipp, K. Meier, J. Schemmel, and F. Schürmann, "A platform for parallel operation of VLSI neural networks," in *Proc. of the 2004 Brain Inspired Cognitive Systems Conference (BICS2004)*, 2004.

[20] *Virtex-II Pro Platform FPGA Handbook*, Xilinx, Inc., www.xilinx.com, 2002.

[21] *RocketIO Ttransceiver User Guide*, Xilinx, Inc., www.xilinx.com, 2003.

[22] *Xilinx Application Note 670, Minimizing Receiver Elastic Buffer Delay in the Virtex-II Pro RocketIO Transceiver*, Xilinx, Inc., www.xilinx.com, 2003.

[23] A. Grübl, "VLSI Implementation of a Spiking Neural Network," Ph.D. dissertation, Universität Heidelberg, 2007.

[24] J. Schemmel, J. Fieres, and K. Meier, "Wafer-Scale Integration of Analog Neural Networks," in *in Proc. of the International Joint Conference on Neural Networks IJCNN'08*, Apr. 2008.

[25] J. Fieres, J. Schemmel, and K. Meier, "Realizing Biological Spiking Network Models in a Configurable Wafer-Scale Hardware System," in *in Proc. of the International Joint Conference on Neural Networks IJCNN'08*, Apr. 2008.