Interconnecting VLSI Spiking Neural Networks Using Isochronous Connections

Stefan Philipp, Andreas Grübl, Karlheinz Meier, and Johannes Schemmel

Ruprecht-Karls-Universität Heidelberg, Kirchhoff-Institut für Physik
Im Neuenheimer Feld 227, 69120 Heidelberg, Germany
sphilipp@kip.uni-heidelberg.de

Abstract. This paper presents a network architecture to interconnect mixed-signal VLSI[1] integrate-and-fire neural networks in a way that the timing of the neural network data is preserved. The architecture uses isochronous connections to reserve network bandwidth and is optimized for the small data event packets that have to be exchanged in spiking hardware neural networks. End-to-end delay is reduced to the minimum by retaining 100 % throughput. As buffering is avoided wherever possible, the resulting jitter is independent of the number of neural network chips used. This allows to experiment with neural networks of thousands of artificial neurons with a speedup of up to 10^5 compared to biology. Simulation results are presented. The work focuses on the interconnection of hardware neural networks. In addition to this, the proposed architecture is suitable for any application where bandwidth requirements are known and constant low delay is needed.

1 Introduction

Great efforts have been made in the field of modeling neural networks. Neural networks have been simulated in software (see e.g. [1]) and implemented in VLSI technology to exploit its parallel nature. An example implementation using a perceptron-based neuron model is given in [2]. Further recent developments [3, 4] implement spiking neurons based on the integrate-and-fire neuron model in VLSI hardware.

The mixed-signal artificial neural network (ANN) chip presented in [4] features 384 neurons and 100K synapses including synaptic plasticity. It operates with a speedup factor of 10^4 to 10^5 compared to biology according to the model and contains analog as well as digital elements. Neurons and synapses use analog computation whereas the external interface remains purely digital. This allows to interconnect multiple chips with digital hardware to create a large network of spiking neurons.

The ANN operates within a hardware and software framework [5], which has been created for the parallel operation of multiple artificial hardware neural networks. As the neural network chip models biological behavior in continuous

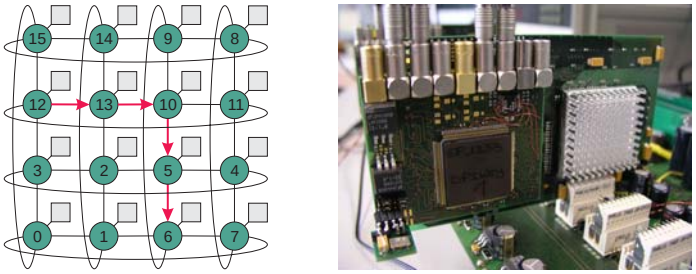
[1] Very Large Scale Integration

time, the timing is important: The time constants of the neuron operation determine the timing requirements for the inter-chip connections in terms of delay, jitter (delay variation) and throughput. This paper presents the implementation of a global communication network which accomplishes these needs. The strong timing requirements of spiking hardware neural networks are met by enabling isochronous connections between the network nodes.

2 Interconnecting VLSI Neural Networks

The work presented in this paper has been done using the neural network framework presented in [5]. It consists of backplanes that hold up to 16 *network modules* each. If the ANN of [4] is used, a single fully equipped backplane allows the parallel operation of 6144 artificial spiking neurons with 1.57 million synapses.

Each ANN is interfaced by a commercial FPGA[2] that provides both, the local ANN interface and the physical interconnection. The programmable logic allows to implement the global communication network for neural spike events within the FPGA. The resulting neural network consists of ANN-internal connections as well as global connections. The physical network topology is graphed in Figure 1(a), whereas Figure 1(b) shows a photograph of a single network module.



(a) (b)

Fig. 1. (a) The topology of a fully equipped backplane corresponds to a 2-dimensional toroidal structure with an ANN at each node (depicted as a square). Nodes not directly connected communicate via intermediate nodes. An example path is shown from node 12 to node 6. (b) A single network module with ANN (left) and FPGA (right). Connections to other modules are provided by the connector beneath the FPGA.

The ANN operates with a speedup factor of 10^4 to 10^5 compared to biology according to the implemented neuron model. Information transferred between the neurons is encoded within the spike event timing. This results in certain requirements on event throughput, delay and jitter the network has to met. Event data loss has to be avoided. The expected data rates depend on the encoding of a spike event and the estimated spike frequency, which in turn depends on the

[2] Field Programmable Gate Array

neural network configuration. Table 1 shows an overview of expected data rates for an estimated mean spike rate of 1 MHz for a single neuron.

Table 1. Expected mean and peak data rates at the ANN-FPGA interface (3 events encoded using 64 bit) and on global connections (32 bit per event) at the time factor of 10^5 and a biological mean firing rate of 10 Hz. Peak data rates are assumed to be 4 times higher than mean rates. The interface is limited to 1.6 GB/s in single direction. Only parts of the chip neurons can use global interconnects at this time factor.

	Spike Event rate	Interface ANN-FPGA (single direction)		Global Connections (single direction)	
	average	average	peak	average	peak
ANN Neuron	1 MHz	2.7 MB/s	10.7 MB/s	4 MB/s	16 MB/s
ANN Chip	384 MHz	1 GB/s	(4 GB/s)	1.5 GB/s	6.1 GB/s

The delay of events being transported using global connections will be significantly larger as if using ANN-internal connections. The number of intermediate nodes (network modules) that have to be passed causes different classes of end-to-end delays and thus allows different classes of inter-neuron connections to be simulated. According to the time factor of the network, the delay of single-hop connections has to be as small as possible (in the range of few 100 ns) to be able to simulate short-range neural connections.

Incoming events have to be resynchronized at the destination to the local ANN timing. The capacity of local resynchronization buffers is limited by the FPGA. Therefore, it is important for the neural network operation that the *delay variation* (jitter) of neuron connections introduced by the network is as small as possible, too. Network connections with nearly constant bandwidth and delay between two network nodes are called *isochronous* connections.

3 Implementation of Isochronous Connections

The implementation of isochronous connections has been done in multiple steps: Resource reservation (c.f. Section 3.1) and global synchronization (Section 3.2) are performed prior to the neural network experiment. At runtime, a switching algorithm implemented within the FPGA forwards spike events through the network according to the reserved resources (Section 3.3). Section 3.4 presents the mapping algorithm which calculates the reservation pattern.

3.1 Resource Reservation

It is assumed that the synaptic inter-neuron connections are known and remain fixed throughout the experiment. The knowledge about bandwidth requirements between any two network nodes simplifies the global communication network

implementation: For any two network nodes A and B, all synaptic connections starting at A and ending at B are aggregated to a *virtual connection* c_{AB} with an estimated mean data rate r_{AB}. Aggregating lots of neurons to a connection reduces the additional bandwidth to be reserved for bursting and thus increases the link utilization.

Bandwidth is reserved by framing, which is done by dividing the time axis into time slots of equal length that last the duration of the transfer time of a 32 bit spike event. Multiple time slots are aggregated to a network frame, which includes a globally constant number of slots. A virtual connection may consist of one or multiple time slots depending on its bandwidth requirement and the available link capacity (c.f. Figure 2).

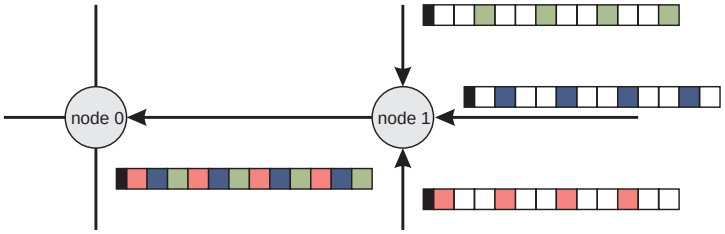


Fig. 2. Example frames with sync character and 12 time slots. Node 1 forwards incoming event data according to its routing table. Output conflicts are removed by the slot assignment pattern.

At runtime, all nodes forward incoming traffic to local inputs, if the node is the destination, or to local outputs, if the event is transit data. Care has to be taken if data arrives at different inputs and has to be forwarded to the same output. In packet switched networks, this problem is solved by buffering incoming packets in local queues and a scheduler decides which queue is to be served for output [6]. Packets have to provide a header to store its destination and the queuing process may result in unpredictable delays. The scheduler commonly is of significant complexity to ensure fairness and throughput.

The solution presented in this paper uses a reordering scheme comparable to [7, 8] but without local buffering. Consider the packet frame of Figure 2. The reservation of time slots is done in a way that only at most one event arriving at a network node have to be forwarded to the same local output. Note that the destination of an incoming event solely depends on the time slot, the event has been placed in. Neither local queues nor routing headers are needed and the online switching process is reduced to O(1) complexity. Furthermore, the throughput is guaranteed and the delay introduced by the switch is constant.

3.2 Synchronization

The framing strategy relies on global synchronization of all nodes. Buffers have been avoided to allow incoming events to be immediately forwarded to outputs,

which requires correct frame alignment up to the precision of the external data sampling rate. Every network node implements an internal clock counter which periodically counts the slots of the current time frame. For the initial setup of the system, the clocks of all network nodes are shifted internally until being synchronous. Every node sends a synchronization symbol at the beginning of each frame to control the synchronization process. The switch at each node monitors the occurrence of the symbol to check the synchronization with all of its neighbors.

Care has to be taken as the FPGA design is clocked by a local oscillator, which has only limited accuracy. For oscillator speeds of 100 MHz, even at accuracies of 10^{-8}, the network synchronization will be lost after about a second. Due to this, the system uses a global clock source, i.e. a single globally unique oscillator defines the discrete time reference for all network nodes. This ensures that the synchronization holds infinitely long if no signaling error occurs.

3.3 FPGA Implementation

The network architecture has been implemented within the Xilinx Virtex-II Pro FPGA [9] of each network module. Figure 3 shows a design overview.

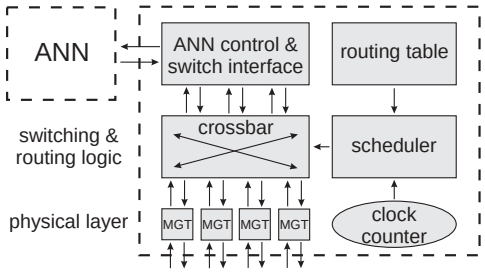


Fig. 3. Schematic of the FPGA logic at every network module. The crossbar has an equal number of event input and output ports, which are served in parallel for each time slot according to the local routing table.

The physical layer of the network performs the external communication. It is implemented using four of the FPGA internal multi-gigabit serial transceivers (MGT) [10], which operate at an external line rate of 3.125 Gbit/s allowing a usable data rate of 312.5 MB/s. Additional adjustments [11] have been made to reduce the delay of the MGTs to 128 ns.

The switching and routing logic consists of a crossbar and a scheduler. Queuing has been avoided as described in Section 3.1. The crossbar has n bidirectional ports for packet input and output which are connected to the local ANN interface and the MGT links. The routing table contains the time slot assignment of the virtual connections. Every time slot, the scheduler makes a forwarding decision based on the table information. Events are then forwarded in parallel

from each input via the central crossbar to its dedicated output. Note that no header processing of incoming event data is necessary.

The interface to the ANN control logic is realized with multiple ports operating at the same data rate as the external ports. It is kept as simple as possible to allow for a convenient exchange of the current neural network hardware with upcoming applications. The timing at the interface is determined by the network frame timing. The scheduler informs the ANN control logic at every time slot, when incoming event data is available and output data can be accepted. Events have to be dropped if the amount of event data generated by the ANN exceeds the bandwidth reserved.

Note that the ANN switch interface is the only point where delay variations may be introduced. Neural network events created within the ANN have to wait for a time slot belonging to its virtual connection. After being accepted by the switch, event data will arrive at its destination with a *fixed* delay.

3.4 Mapping Process

The mapper is a software algorithm created in C++ and calculates the framing scheme. It uses experimental setups (neural network configurations) as input and generates table entries for the switching logic at every network node.

First, the mapper calculates the mean bandwidth requirement of each virtual connection. In the next step, the shortest path algorithm from Dijkstra [6] is carried out to find an interconnected set of physical links from the source node to its destination. Next, the time slot reservation pattern is calculated for all virtual connections. The algorithm calculates a collision-free assignment in which each local output is assigned to at most one input within each time slot. This is done by translating the problem to the common *vertex-color*[12] problem from graph theory which is then solved. After the algorithm converged, all time slots for each connection are found and the local routing tables at every node are initialized.

4 Performance and Results

The network architecture has been implemented in the programmable hardware of the FPGA network module. As described in section 3, the throughput of the virtual connections is guaranteed by design. To measure the delay performance, the system has been set up with a delay measurement logic at the ANN interface which counts reference clock cycles of 6.4 ns. The results can be seen in Table 2. Each hop adds a delay of 147.2 ns. Note that 128 ns (87%) of the delay is introduced by the physical layer (MGT) of the programmable logic device. The switching and routing logic introduces only 13 % due to its simplicity.

Delay variation (jitter) is introduced only at the source node at the ANN interface to the switch if spike events wait for its associated time slot. As an example, with a frame size of 12 time slots and equally distributed slots, a connection using 33 % bandwidth will have to wait for at most 32 ns. Once

Table 2. Delay results at the switch interface for global ANN interconnects.

	Cycles	Delay (ns)	Biological Time (ms)	
			at 10^4	at 10^5
Direct Interconnect	24	153.6	1.5	15
One Intermediate Node	47	300.8	3.0	30
Two Intermediate Nodes	70	448.0	4.5	45
Three Intermediate Nodes	93	595.2	6.0	60

sent to the network, spike events arrive with *constant* delay at the destination, hence the jitter is independent of the number of intermediate network nodes. The numbers show that the isochronous connections of the network are able to guarantee the throughput and delay requirements necessary for VLSI neural networks operating at time factors of up to 10^5 compared to biology.

A software simulation has been created in C++ to verify the operation in a cycle-accurate precision. Figure 4 shows results for the simulation of 250 neurons. Only about 1.2 % of the events have to be dropped in the case of simulated bursting whereas no drops at all occur during the non-bursting operation.

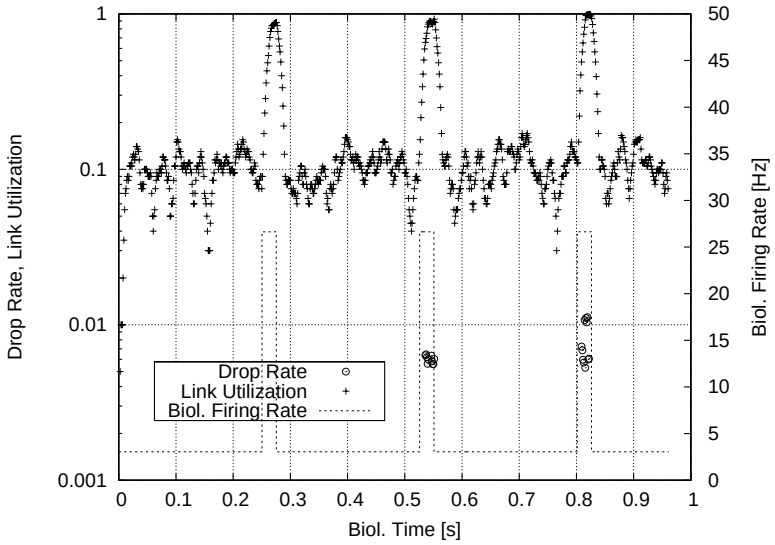


Fig. 4. Software simulation of link utilization and drop rate at the switch input interface. Spikes of 250 local neurons are transferred with the maximum bandwidth available. Bursting behavior is simulated by modulating the spike event rate of all neurons between 3 Hz for 250 ms and 28 Hz for 25 ms with a speedup of 10^4. The system can handle burst rates close to the maximum link capacity of 30 Hz per neuron at this setup. Less than about 1.2 % of the events are dropped during a burst.

5 Conclusion

This paper presented a solution to satisfy the transmission requirements which arise at the interconnection of VLSI spiking neural networks. The concept has successfully been set up in real hardware. It has been shown that a network infrastructure using isochronous connections can satisfy the strong delay and throughput needs to create a large neural network consisting of thousands of neurons and millions of synapses. With constant delay and an online complexity of O(1), the network architecture is scalable in terms of the number of network nodes and external line speed. A single backplane currently provides 6144 neurons and 1.6 million synapses. However, the network architecture can also be used to interconnect multiple backplanes by using remaining FPGA transmitters. Furthermore, the network can be used in the next stage of the current project to interconnect spiking neurons using wafer-scale integration [13].[3]

References

1. The Neural Simulation Technology (NEST) Initiative: Homepage. http://www.nest-initiative.org (2007)
2. Schemmel, J., Hohmann, S., Meier, K., Schürmann, F.: A mixed-mode analog neural network using current-steering synapses. Analog Integrated Circuits and Signal Processing **38**(2-3) (2004) 233–244
3. Indiveri, G., Chicca, E., Douglas, R.: A VLSI array of low-power spiking neurons and bistable synapses with spike–timing dependent plasticity. IEEE Transactions on Neural Networks **17**(1) (Jan 2006) 211–221
4. Schemmel, J., Grübl, A., Meier, K., Mueller, E.: Implementing synaptic plasticity in a VLSI spiking neural network model. In: Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN'06), IEEE Press (2006)
5. Fieres, J., Grübl, A., Philipp, S., Meier, K., Schemmel, J., Schürmann, F.: A platform for parallel operation of VLSI neural networks. In: Proc. of the 2004 Brain Inspired Cognitive Systems Conference (BICS2004). (2004)
6. Tanenbaum, A.S.: Computer Networks. Pearson Education Int. (2004)
7. Hung, A., Kesidis, G., McKeown, N.: ATM input-buffered switches with guaranteed-rate property. In: Proc. of IEEE ISCC'98, Athens. (1998) 331–335
8. Li, S., Ansari, N.: Input-queued switching with QoS guarantees. In: Proceedings of IEEE INFOCOM'99, New York (1999) 1152–1159
9. Xilinx, Inc. www.xilinx.com: Virtex-II Pro Platform FPGA Handbook. (2002)
10. Xilinx, Inc. www.xilinx.com: RocketIO Ttransceiver User Guide. (2003)
11. Xilinx, Inc. www.xilinx.com: Xilinx Application Note 670, Minimizing Receiver Elastic Buffer Delay in the Virtex-II Pro RocketIO Transceiver. (2003)
12. Brélaz, D.: New methods to color the vertices of a graph. Commun. ACM **22**(4) (1979) 251–256
13. Ehrlich, M., Mayr, C., Eisenreich, H., Henker, S., Srowig, A., Grübl, A., Schemmel, J., Schüffny, R.: Wafer-scale VLSI implementations of pulse coupled neural networks. In: Proc. of IEEE SSD07, Hammamet, Tunisia (March 2007)

[3] This work is supported in part by the European Union under the grants no. IST-2001-34712 (SenseMaker) and no. IST-2005-15879 (FACETS).