A Hardware/Software Framework for Real-time Spiking Systems

Matthias Oster, Adrian M. Whatley, Shih-Chii Liu, and Rodney J. Douglas

Institute of Neuroinformatics, Uni/ETH Zurich
Winterthurerstr. 190, 8057 Zurich, Switzerland
mao@ini.phys.ethz.ch

Abstract. One focus of recent research in the field of biologically plausible neural networks is the investigation of higher-level functions such as learning, development and modulatory functions in spiking neural networks. It is desirable to explore these functions in physical neural network systems operating in real-time. We present a framework which supports such research by combining hardware spiking neurons implemented in analog VLSI (aVLSI) together with software agents. These agents are embedded in the spiking communication of the network and can change the parameters and connectivity of the network. This new approach incorporating feedback from active software agents to aVLSI hardware allows the exploration of a large variety of dynamic real-time spiking network models by adding the flexibility of software to the real-time performance of hardware.

1 Introduction

Much recent research in biologically plausible, spiking neural networks focuses on the dynamic properties of network models such as learning algorithms based on synaptic plasticity and global reward signals, development of connectivity, and modulatory functions such as gain control. It is desirable to explore such properties in a physical system in real-time: first, because such a system forces the model to include the real-time timing properties that are important for biological systems; and second, only a system that interacts with its physical environment can demonstrate that the model being studied works correctly under real-world conditions. In order to achieve this real-time behaviour, aspects of network models are often implemented in hardware [1].

We present a framework that allows the exploration of the dynamic properties of network models in real-time neural networks by combining hardware spiking neurons and software agents. Local analog and continuous-time computation is performed in the hardware, while *higher-level functionality* is implemented in software. By higher-level functionality we understand whatever algorithms are not implemented in the currently available hardware, e.g. learning algorithms based on synaptic plasticity and global reward signals, development of connectivity, and modulatory functions such as gain control. This new approach allows

a wide variety of algorithms to be tested quickly and will enable real-time systems with large computational power to be assembled.

Several projects have focused on combining hardware based spiking neurons with dynamically reconfigurable connectivity: the Silicon Cortex (SCX) project [2] proposed connecting multiple chips using spiking communication and already incorporated the possibility of building integrated hardware and software models of the kind we propose here, although no such models were actually implemented at that time due to the presence of a critical bug in the host communication channel of the SCX. Similar systems are used by various groups. The IFAT system [3] aims for a similar goal to that of this work, however, we separate the hardware and software parts to achieve greater flexibility, higher performance and easier implementation of the algorithms (e.g. in MATLAB).

2 System Architecture

Hardware systems implementing the analog and continuous-time computation performed by neurons and synapses in transistor circuits have long been a subject of research and many examples can be found in the literature, e.g. [4]. The circuits approximate models of biological neurons which are then integrated in large arrays on a chip using Very Large Scale Integration (VLSI).

The connectivity between neurons is implemented by the transmission of spikes over a multiplexed bus using the address-event representation (AER) protocol [5]. Each spike is represented by the address of the source neuron or the receiving synapse and is transmitted asynchronously. A mapper translates the addresses of the sending neurons to lists of receiving synapse addresses using a look-up table, thus allowing for arbitrary intra- and inter-chip connectivity between neurons. Various networks and input sensors can be combined to form a real-time multi-chip system (see Fig. 1). A monitor translates spikes from hardware to software, while a sequencer provides the reverse translation. The mapper, monitor and sequencer are integrated on a PCI-AER board [6] which plugs into a PCI slot in a desktop computer.

Software agents embedded in this system perform the higher-level functions as defined in section 1. In this framework, an agent is an independent software process that implements a particular higher-level algorithm. At present, there are agents for analysis, on-line display, learning, modulation functions and stimulation. Multiple agents can run concurrently. Each agent communicates with the hardware neural network by receiving spike trains or activity from the network, and can change the synaptic connectivity and adjust the parameters of the neurons. Agents can also stimulate the network with artificial spike trains, providing input from parts of the system which are not implemented in hardware. Event-based agents, i.e. agents that perform computation based on single events, are implemented in C++, while agents that operate on the statistics of the activity of the network (as discussed in section 2.1) and do not require a low latency, can also be implemented in MATLAB.

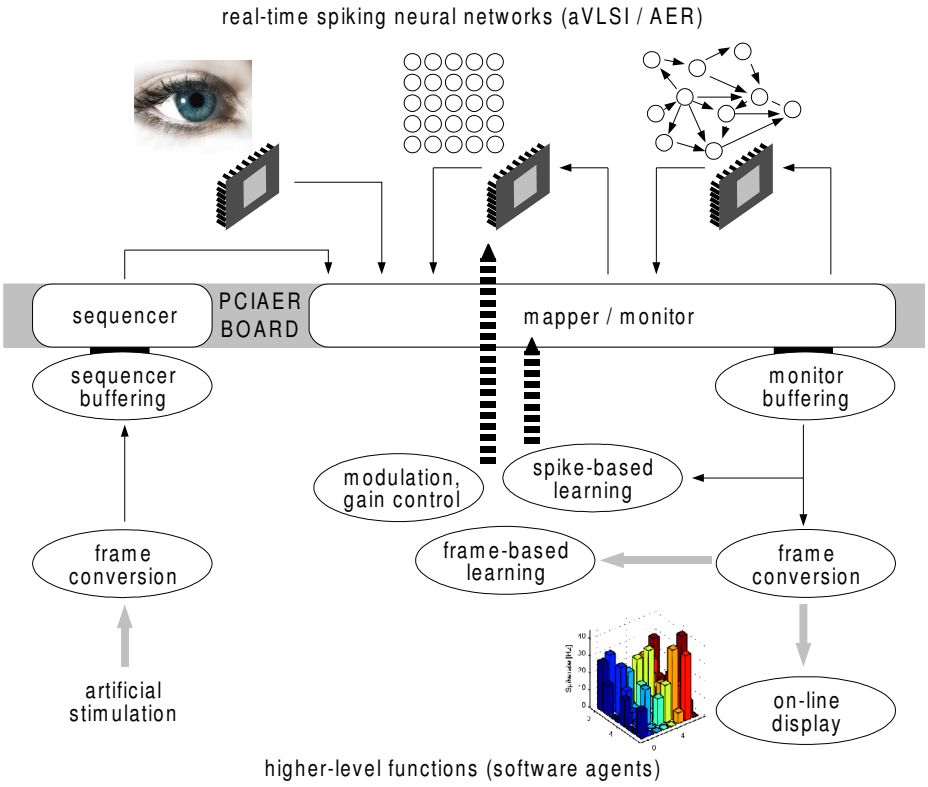

Fig. 1. Overview of the system architecture. Real-time spiking neural networks are implemented in VLSI and integrated on a chip (top). As examples, a retina, a feed-forward network and a recurrent network are shown. The neurons communicate using the address-event representation (AER) protocol (black arrows). A PCI-AER board monitors, sequences and remaps the spikes to implement the connectivity. Higher-level functions such as on-line analysis, learning algorithms, modulatory functions and artificial stimulation are implemented in C++ software agents (bottom). The agents transmit and receive spikes to and from the hardware using AER network packets and can change the connectivity and parameters of the network by modifying the mapping table and bias voltages (dashed arrows). Analysis agents transform the spike trains into a frame-based format which represents the activity of a neuron population in the chosen coding scheme (gray arrows). This allows agents implemented in slow environments such as MATLAB to be integrated into the framework. As an example, a 3D bar chart displaying the instantaneous firing rate is shown.

2.1 Software AER and Frame-Based Representation

In the software, a spike is represented by a data structure containing an address and a timestamp recorded when the spike is captured. The timestamp is required to preserve timing information when the spikes are buffered. The monitor agent sends blocks of spikes including their timestamps as network packets to receiving agents. We chose UDP for this software spiking communication because it is fast and allows several agents to receive the spike trains at the same time using multicasting.

For many applications, we are not interested in the spike train itself, but rather in the statistics of the activity of the neurons in the network. Depending on the chosen coding scheme, this can be an instantaneous spike rate, a spike count, time-to-first spike, or any other suitable measure. Analysis agents transform the spike train into one of these activity-based representations.

An agent further along the processing chain can request this activity. To do so, it first specifies the addresses of the neurons it is interested in. The analysis agent then transmits the activities of these neurons as a vector. We call this a frame-based representation. In contrast to conventional frame-based representations, the timing is asynchronous since the frame can be requested at any time and the analysis agent will calculate the contents of the frame at that time. Frames are transmitted between the agents using a TCP network connection.

The frame-based representation makes it possible to include agents in the framework that have such a long response time that they could not keep up with the real-time spike train. This allows agents to be implemented in slow environments such as MATLAB. As an example, an on-line display agent can request frames and display them with a fixed refresh rate independently of the amount of spikes received.

2.2 Learning Agents

The framework allows a variety of learning and modulation algorithms to be explored by implementing them as agents. An example of an event-driven agent is an agent that implements spike-time-dependent plasticity (STDP) [7]. The agent is configured with the addresses of the post-synaptic neurons and their pre-synaptic afferents. All incoming spikes are buffered and the agent checks whether a post-synaptic neuron spiked. If so, the buffer is scanned for spikes from pre-synaptic neurons that fall within the time window around the post-synaptic spike and long-term depression or potentiation is calculated. The synaptic efficacy is then changed on the fly in the mapper's look-up table using burst length variation [8]. Exploring STDP with this software-based approach has advantages over a hardware implementation in that the implementation time is shorter and testing is easier, since no new hardware has to be added on chip and all of the algorithm's variables are accessible.

2.3 Performance

Table 2 shows the performance of the framework in the current state. We show both maximal values for standalone agents and values for a typical setup using an agent implementing STDP learning and a display agent. The main limitation is transferring data (spikes and synaptic weight updates) over the PCI bus. The driver for the PCI-AER board does not yet support interrupts, and the current board does not support bus mastering. Even with these limitations, the measured throughput is sufficient for many experiments because it refers to the continuous spike rate, whereas biologically plausible networks typically have short high-frequency bursts of spikes, and the average spike rate remains well below the maximum throughput.

Maximum values (standalone agent)			
	rate $[s^{-1}]$	avg.(max.) latency	CPU load [%]
AER communication (up to 4 chips)	1.2MSpikes	1.2μs	-
Monitoring	310kSpikes	10 (80) ms	97
Synaptic efficacy updates	129kUpdates	-	98
Typical setup (multiple agents)			
Monitor agent	53kSpikes	15 (90) ms	8
STDP spikes of postsynaptic neurons	24kSpikes		35
synaptic efficacy updates	16kUpdates	44 (220) ms	
Spike-rate to frame conversion	53kSpikes	25 (120) ms	3
On-line display (MATLAB/X)	2.3	-	24

Fig. 2. Performance measurements. All rates given are maximal rates at which no or very few spikes are lost ($<$ 1 packet in 1s). 'latency' denotes the mean (maximum) latency from a spike being recorded by the PCI-AER board until it is received and processed by an agent. All measurements were done on a standard PC (2.4GHz Pentium IV, Linux kernel 2.4.26).

3 Conclusion and Outlook

With its modular architecture, our framework supports multiple agents using event or activity based computation. Software spike trains are broadcast to multiple receivers and statistics relating to different spike coding schemes can be requested in a frame-based representation. Thanks to the use of standard network protocols, the system is scalable and can be distributed across several computers.

New hardware interfaces that implement the individual functionalities of the PCI-AER board in single hardware modules are being developed as part of a current project [9]. These hardware modules can be inserted where needed into

the data flow of the system. They will also support much higher spike rates than the current PCI-AER board, of up to 32MSpikes/s.

The framework can be used to quickly explore higher-level functionality in a real-time system. Through the use of software agents, it provides a rapid prototyping tool to test learning and modulation algorithms in a real-time system. With input sensors such as a silicon retina, it can be used to build more complex spike-based neural network systems than presented here that are capable of reacting to their real-world environment.

Acknowledgments

We would like to acknowledge Vittorio Dante and Paolo Del Giudice (Istituto Superiore di Sanità, Rome, Italy) for the original design of the PCI-AER board, and Gerd Dietrich and other members of the Institute of Neuroinformatics involved in the development of the PCI-AER board, of its drivers, and software library components. We thank Wolfgang Einhäuser for fruitful discussions relating to the implementation of STDP. This work was supported in part by the IST grant IST-2001-34124 (CAVIAR).

References

1. Douglas, R., Mahowald, M., Mead, C.: Neuromorphic analog VLSI. Annual Review of Neuroscience **18** (1995) 255–281
2. Deiss, S.R., Douglas, R.J., Whatley, A.M.: A pulse-coded communications infrastructure for neuromorphic systems. In Maass, W., Bishop, C.M., eds.: Pulsed Neural Networks. MIT Press, Cambridge, Massachusetts (1999) 157–178
3. Vogelstein, R., Mallik, U., Cauwenberghs, G.: Beyond address-event communication: dynamically-reconfigurable spiking neural systems. In: The Neuromorphic Engineer. Volume 1. Institute of Neuromorphic Engineering (INE) (2004)
4. Douglas, R., Mahowald, M.: Silicon neurons. In Arbib, M., ed.: The Handbook of Brain Theory and Neural Networks. MIT Press, Boston (1995) 282–289
5. Lazzaro, J., Wawrzynek, J., Mahowald, M., Sivilotti, M., Gillespie, D.: Silicon auditory processors as computer peripherals. IEEE Trans. Neural Networks **4** (1993) 523–528
6. Dante, V., Del Giudice, P.: The PCI-AER interface board. In Cohen, A., Douglas, R., Horiuchi, T., Indiveri, G., Koch, C., Sejnowski, T., Shamma, S., eds.: 2001 Telluride Workshop on Neuromorphic Engineering Report. (2001) 99–103
7. Abbott, L.F., Nelson, S.B.: Synaptic plasticity: taming the beast. Nature Neuroscience **3** (2000) 1178–1183
8. Oster, M., Liu, S.C.: A winner-take-all spiking network with spiking inputs. In: 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS). (2004)
9. IST-2001-34124: Caviar - convolution aer vision architecture for real-time. http://www.imse.cnm.es/~bernabe/CAVIAR (2002)