THE NEUROMORPHIC ENGINEER

VOLUME 2 NUMBER 1 MARCH 2005

Large-scale field-programmable arrays

Reconfigurable hardware has long been of interest to circuit designers and engineers. In the digital domain, programmable logic devices (PLDs) have made a large impact on the development of custom digital chips by enabling the designer to try custom designs on easily-reconfigurable hardware. Since their conception in the late 1960s and early 1970s, PLDs have evolved into today's high-density field-programmable gate arrays (FPGAs).[1] These are widely used in the lab for rapidly prototyping digital hardware, as well as in production goods to decrease time to market and to allow products to be easily upgraded after being deployed.

However, reconfigurable analog hardware has been progressing much more slowly. While early analog integrated circuits (ICs) were often tunable with adjustable biases, truly reconfigurable analog circuitry in the form of field-programmable analog arrays (FPAAs) did not emerge until the late 1980s,[2,3] and commercial offerings did not reach the market until 1996.[4,5] Operational transconductance amplifier (OTA)-based FPAA designs have also been demonstrated,[6,7] but, having been in the marketplace for nearly a decade, current FPAAs have struggled to establish a solid market base. They have been plagued by poor performance, small size, and a lack of generality/functionality.

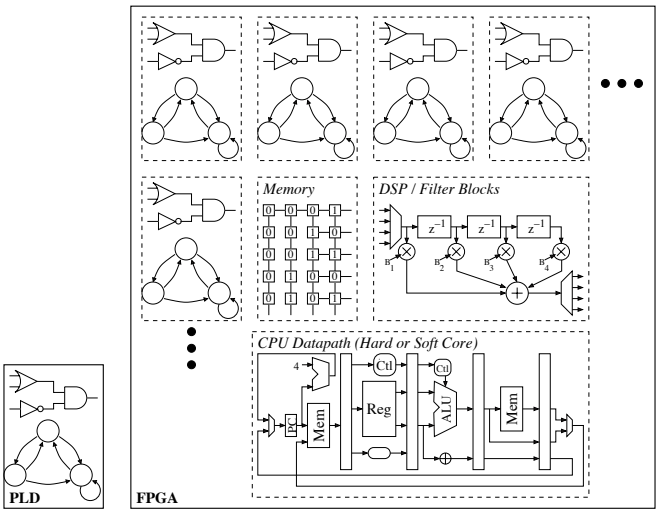
Similar to the digital evolution from PLDs to FPGAs, FPAA chips are moving from these early stages to a large number of reconfigurable analog blocks. We recently introduced a new class of FPAA technology that resembles the architecture, large number of computing elements, functionality, and computational power of FPGA devices, while preserving the many orders of magnitude of power savings typical of analog signal processing.[8,9] We typically refer to these devices as large-scale FPAAs (illustrated in Figure 1). These are accurately-programmable (through floating-gate approaches[10]), continuous-time, analog devices that are capable of implementing full-scale analog signal processing systems.

Fundamentally, FPAAs include two functions: routing (architecture) and computation. In our FPAA block, the switching device is an analog programmable element that can act as an ideal switch, variable resistor, current source, and/or configurable computational element in a single analog programmable memory element. Eliminating the switched banks of scaled devices typical in early FPAA designs creates a large savings in the area required for each

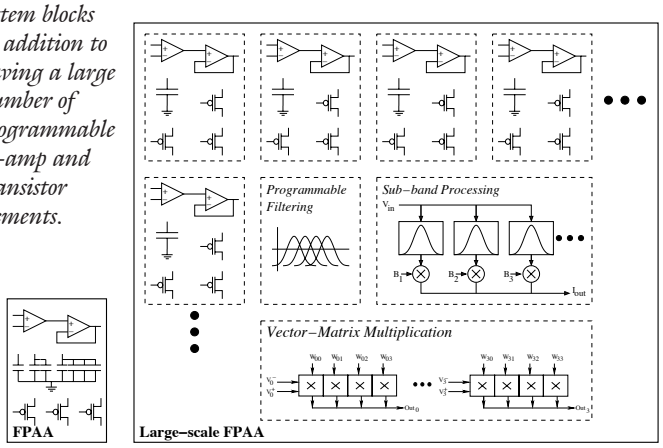
Hasler, *continued p. 10*

In this issue:

Figure 1. Digital PLDs (programmable logic devices) can be used to implement small, carefully-defined pieces of a complex system, while FPGAs (field-programmable analog arrays) can be used to implement entire systems including processor datapaths, complex digital signal processing functions, and more. Modern FPGAs can be 100-10,000 times larger and more complex than the PLDs of the1970s and1980s. Analagously, traditional FPAAs (field-programmable gate arrays) resemble early PLDs in that they are focused on small systems such as low-order filtering, amplification, and signal conditioning. However, the FPAAs based on floating-gate devices presented here are much larger devices, with the functionality needed to implement high-level system blocks in addition to having a large number of programmable op-amp and transistor elements.



(a)



(b)

THE NEUROMORPHIC ENGINEER

is published by the

INSTITUTE OF NEUROMORPHIC ENGINEERING

Editor
Sunny Bains
Imperial College London
sunny@sunnybains.com

Editorial Assistant
Stuart Barr
newsletters@sunnybains.com

Editorial Board
Avis Cohen
Ralph Etienne-Cummings
Rodney Douglas
Timothy Horiuchi
Giacomo Indiveri
Christof Koch
Shih-Chii Liu
Shihab Shamma
Chris Toumazou
André van Schaik

This material is based upon work supported by the National Science Foundation under Grant No. IBN-0129928. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The Institute of Neuromorphic Engineering
Institute for Systems Research
AV Williams Bldg.
University of Maryland
College Park, MD 20742

http://www.ine-web.org

Telluride Workshop 2005 and highlights of 2004

As is usually the case this time of the year, we are currently preparing for the Telluride Neuromorphic Engineering (NE) Workshop, which will take place between June 26th and July 16th 2005. The organizing committee has been busy inviting interesting guest speakers, spanning many of the disciplines of NE. We plan to expand the participation in the areas of neuroprosthesis and man-machine interfaces to complement our usual emphasis on biological and computational sciences. We are looking forward to another exciting workshop! Please surf to our website[1] for all the details. To give you an idea, here is a summary of the highlights from last year's workshop.

Robotics group
The team designed two new muscle-like actuators, one of which was a helical linear actuator that converts rotational motion from a conventional motor to linear motion. A prototype was successfully fabricated using a rapid prototyping machine: a patent on the design is now being submitted.

Distributed sensor networks
This group implemented network-distributed filtering modules to extract features from an address-event image sensor. A transmitter mote was connected to an address-event camera, sending camera data to the distributed filter motes. Each of these implements one filter template—here vertical, horizontal, and 45° orientation—and sends the filtered version of the image to the receiver mote for further processing.

Auditory work group
This year's research went particularly well, partly because the senior researchers met at Stanford in the spring of 2004, talked about a number of interesting projects and determined their suitability and resource needs. Specific progress on four different projects.
Cepstral perception: A pilot experiment spectacularly demonstrated the perceptual relevance of the cepstral signal representation (where you treat the Fourier transform of the decibel spectrum as if it were a signal).
Event spotting: A number of experiments using long temporal windows demonstrated this novel, biologically-plausible means of spotting auditory events. However, superior performance has so far eluded the team.
AER Ear: We demonstrated a fully-functional binaural hearing chip using the address-event representation (see article on page 12).

Vision workgroup
We combined two independently-proposed workgroups: *Networks for the motion pathway* and *Vision chips*. We realized that the 'compositionality' approach to recovering surface properties covered by the first workgroup would be an excellent way of guiding the development of neuromorphic vision systems. Participants derived techniques by which both the local and global computations required by compositionality could be implemented in neuromorphic hardware, and demonstrated their feasibility through a software simulation. This initial work laid the groundwork for a collaborative research proposal submitted to the INE by researchers from the University of Maryland, the University of Pennsylvania and the Hong Kong University of Science and Technology.

Computational neural systems
The two CNS themes—configuring hardware using programmable logic and configuring networks of silicon neurons using spike-timing dependent plasticity—truly blossomed this year, achieving an all-time high in the number of projects on these topics that were completed successfully. This success is unequivocal evidence of the increasing level of sophistication achieved by the neuromorphic community over the years in configurable neuromorphic systems

Locomotion workgroup
This year saw the beginning of a migration towards prosthetics and rehabilitation devices. One project involved measuring foot-fall timing during walking and then using this information to entrain a computer model of a spinal central pattern generator. This can be used in rehabilitation training for paraplegics and spinal neuroprosthetic devices.

This coming year looks like being even more successful. Please come and join us!

Ralph Etienne-Cummings
Johns Hopkins University
Baltimore, MD

Reference
1. http://www.ini.unizh.ch/telluride

Iqr: a simulator for large scale real-world neuronal systems

The brain is organized at many different levels, from sub-cellular processes to the overall system. A key challenge in studying and developing technology based on the brain is rooted in the question of how the organ can most effectively be described and studied. Another compelling reason for tackling the brain's organization lies in the fact that its different levels are not independent but intricately coupled.[1] We have developed a multi-level neuronal simulation environment, Iqr, that tries to deal with these open issues. The software provides an efficient graphical environment for designing large-scale multi-level neuronal systems that can control real-world devices—robots in the broader sense—in real-time. Iqr was originally written for UNIX platforms using C and the Motif library.[2] The new version of the software was developed under the Linux operating system using C++ and the cross-platform widget set Qt (TrollTech A.S., Norway).

The software

Models in Iqr are organized at different levels: at the top is the *system*, which contains an arbitrary number of *processes* and *connections*. Processes consist of an arbitrary number of *groups*, and allow the model to be divided into logical units. A group is an aggregation of neurons of identical type. Connections are used to feed information from group to group, and consist of synapses of identical type, plus the definition of the arborization pattern of the dendrites and axons. Since connectivity plays a key role in neuronal computation, Iqr provides a number of tools and methods to define and manipulate connections.

A graphical user interface is used for model design, run-time control, and real-time display of the internal states of the model. Parameters can be changed on the fly, and the effects of these changes can be assessed immediately. Processes, groups, and connections can be copied to and pasted from the clipboard, and diagrams can be exported as images, and printed. With Iqr the user can visualize and save the states of the model's elements (time and group plots), and visualize the static and dynamic properties of connections and synapses (connection plots). To allow more detailed evaluation, the data sampler saves states of selected elements of the model.

To make it as useful as possible, Iqr comes with a wide range of predefined interfaces to hardware devices. These include modules to control Khepera and Koala robots (K-Team S.A., Lausanne, Switzerland), Lego MindStorms, and the blimp robot used in the AMOTH project[3] (full name, "A fleet of *a*rtificial chemosensing *moth*s for distributed environmental monitoring"). Modules can be run synchronized or in their own thread, meaning that their update speed is independent of that of the main simulation. Also, the extensibility architecture of Iqr's design allows users to write their own neuron and synapse types, and to define new interfaces to hardware.

Finally, Iqr uses an XML-based format, thus making it easy to exchange models between different research groups, and to transform them into a wide range of other description grammars. Tools to extract information from systems can be implemented easily.

Integration with aVLSI and other projects

The combination of using Iqr's GUI and connection specification methods provides an easy way to design connectivity, export it, and to use it as the basis for defining the mapping on a chip, or between chips. During the 2004 Telluride workshop an Iqr module to readout an AER (address-event representation) retina via the PCI (peripheral component interface)-AER was developed and tested.[4] The interface module sampled the AER output of the retina every 10ms, and represented the activity of the retina by an array of 32×32 neurons. This interface can easily be extended to connect a large variety of AER chips and to provide a bridge among a large number of them.

Iqr has already been used successfully in a number of projects. Among these was the building of the Distributed Adaptive Control (DAC) series of models[1] of learning and memory, which allowed a mobile robot to form associations about color patterns during the exploration of an arena. The memories stored allowed the robot to

***Bernardet,** continued p. 7*

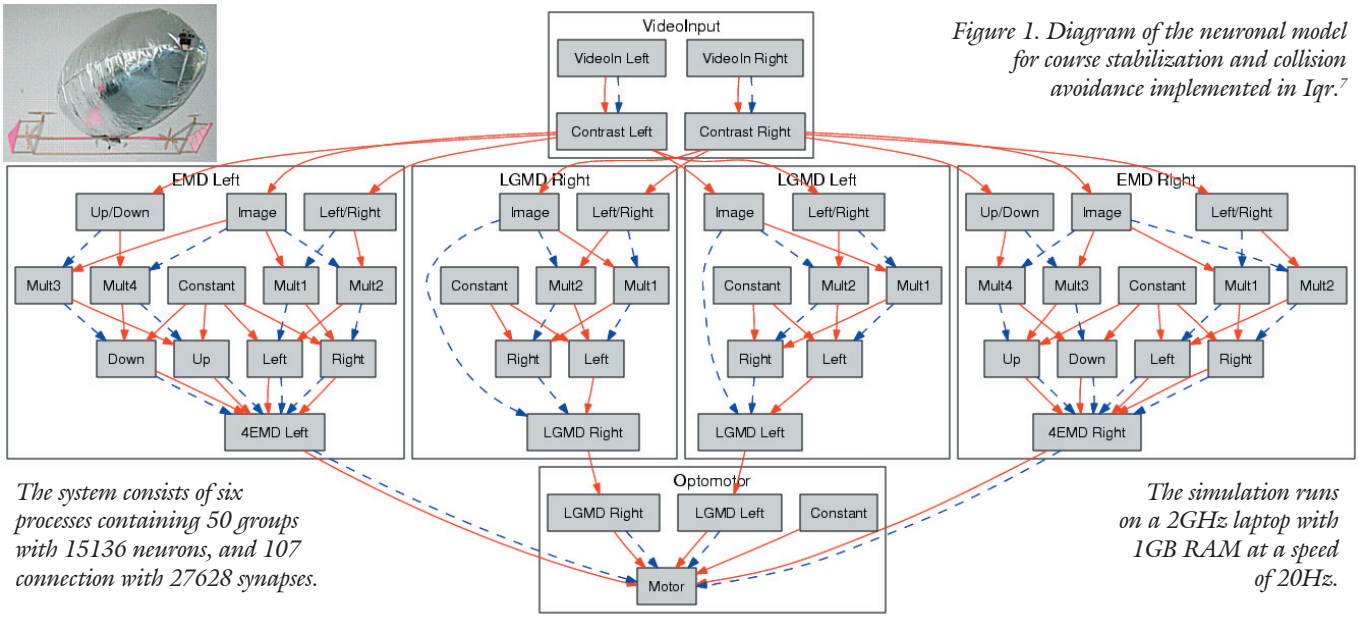


Figure 1. Diagram of the neuronal model for course stabilization and collision avoidance implemented in Iqr.[7]

The system consists of six processes containing 50 groups with 15136 neurons, and 107 connection with 27628 synapses.

The simulation runs on a 2GHz laptop with 1GB RAM at a speed of 20Hz.

VLSI reconfigurable networks of integrate-and-fire neurons with spike-timing dependent plasticity

In the past few years we have seen an increasing number of projects and demonstrations at the Telluride Neuromorphic Engineering Workshops that involve multi-chip spiking systems interfaced via the address-event representation (AER). Indeed, a renewed interest in spiking neural networks is leading to the design and fabrication of an increasing number of VLSI AER devices that implement various types of networks of integrate-and-fire (I&F) neurons.[1-5] Such devices can be used in conjunction with PCI-AER boards of the type described in the article by Dante *et al.* on page 5. In this case, they can be considered as a new generation of hardware neural-network emulators that enable researchers to carry out simulations of large networks of spiking neurons with complex dynamics in real-time, possibly solving computationally demanding tasks.

These systems use mixed-mode analog/digital circuits, exploiting the best of both worlds, and use asynchronous logic to implement event-based communication across multiple chips. They are mainly used to implement specific models of biological neural systems for basic research and scientific investigation: but with the additional aim of developing the technology and infrastructure for engineering applications along the way.

In this article we present one of these new breed of devices. We show how it can be used in conjunction with the PCI-AER board described on page 5 to emulate in real-time spiking neural networks with realistic synaptic dynamics, and to characterize spike-based learning algorithms in arbitrarily-configured networks of spiking neurons.

The aVLSI chip was implemented using a standard AMS 0.8μm CMOS process, and comprises a linear array of 32 low-power I&F neurons, a 2D array of 32×8 synaptic circuits, and input/output AER interfacing circuits (see Figure 1). The chip's silicon synapses can update their synaptic weights via a learning algorithm based on the recently-proposed spike-timing-dependent plasticity (STDP) mechanism. If a pre-synaptic spike arrives at the synaptic terminal before a post-synaptic spike is emitted, within a critical time window, the synaptic efficacy is increased. Conversely, if the post-synaptic spike is emitted just before the pre-synaptic spike arrives, synaptic efficacy is decreased.

The silicon synapses also comprise bistability circuits for driving the synaptic weight to one of two possible analog values (either potentiated or depressed). These circuits drive the synaptic-weight voltage with a current that is superimposed on that generated by the STDP circuits and which can be either positive or negative. If, on short time scales, the synaptic weight is increased above a set threshold by the network activity via the STDP learning mechanism , the bistability circuits generate a constant weak positive current. In the absence of activity (and hence learning) this current will drive the weight toward its potentiated state. If the STDP circuits decrease the synaptic weight below the threshold, the bistability circuits will generate a negative current that, in the absence of spiking activity, will actively drive the weight toward the analog value, encoding its depressed state.

The STDP and bistability circuits allow us to implement learning and long-term storage of the synaptic states. The use of the AER communication protocol allows us to access individual synapses of the network for providing input signals, to read from each neuron generating output signals, and most importantly to (re)-configure the neural network topology.

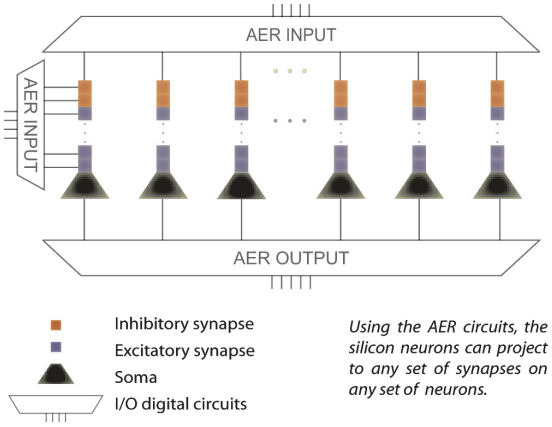


Figure 1. Block diagram of the aVLSI chip architecture. Small trapezoids in the array represent integrate-and-fire neurons, while squares represent inhibitory and excitatory synapses. Output spikes can be redirected to input synapses via the address-event-representation (AER) interface.

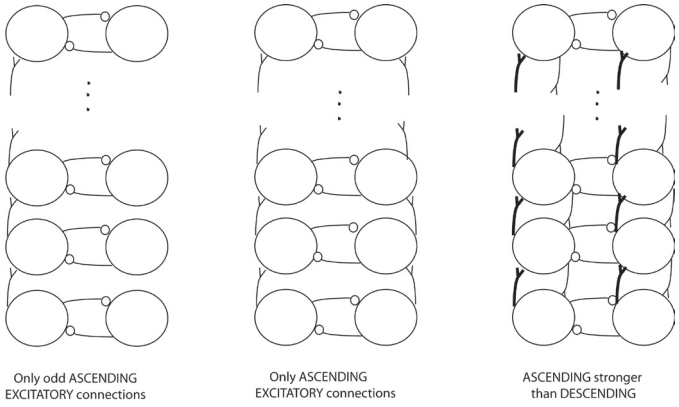


Figure 2. Network of central pattern generators (CPGs) coupled via excitatory connections.

At the 2004 Neuromorphic Engineering Workshop at Telluride, CO, Francesco Tenore and Murat Sekerli used the aVLSI chip and PCI-AER board to implement central-pattern-generator (CPG)-type spiking neurons with different kinds of biologically-plausible neuron couplings. Specifically, they coupled 16 pairs of mutually-inhibiting neurons with excitatory connections (see Figure 2). They were able to change the strength of the synaptic weight, which was controlled by an external voltage reference, and vary the type of connectivity patterns in the network (via the PCI-AER board). This way, they were both able to observe different types of network behaviors and reproduce activity patterns that have been seen, in some form, in biology. This simple experiment, set-up and carried out in the last three days of the workshop, demonstrated the system's flexibility in configuring arbitrary types of networks of spiking neurons.

Another experiment started at the 2004 workshop involved the characterization of the learning properties of the STDP synapses as a function of the mean firing rates of pre- and post-synaptic neurons. This experiment (initially carried out by Massimiliano Giulioni and Patrick Degenaar) was recently

***Indiveri,** continued p. 7*

Hardware and software for interfacing to address-event based neuromorphic systems

Address event representation (AER)[1] has long been considered a convenient transmission protocol for neuromorphic devices. This is because messages are transmitted by an asynchronous sequence of all-or-none, 'digital' spikes, carrying information in the (analog) pattern of inter-spike time intervals. One evolution of the originally proposed point-to-point AER protocol is a many-to-many, fully-arbitrated protocol that supports the design of fairly general-purpose prototyping systems (as in the Silicon Cortex project[2]).

One missing and long-needed feature of AER-based systems is the ability to acquire data from complex neuromorphic systems, to stimulate them using suitable data, and to support the design and the operation of multi-chip AER-based systems. Such needs have been only partially met so far through commercial or home-made hardware and software built for a particular purpose and idiosyncratically dependent on the specific system. We have implemented a general-purpose solution to this problem in the form of a PCI (peripheral component interconnect) board,[3] developed at the Italian Institute of Health, supported by software developed at the Insitute for Neuroinformatics. The article on page 4 of this issue illustrates the use of the this board with VLSI networks of spiking neurons

The PCI-AER board can handle up to four sender and four receiver chips. Figure 1 illustrates the three main functions that the board can perform independently in a typical environment: monitoring, sequencing, and mapping.

The monitoring function involves reading and time-stamping events from the AER bus, and making them available to the host PC for further processing. The *monitor* taps any AER transaction and, as the transaction is detected, stores a time label and the address in the FIFO (first in, first out). This decouples the management of AER events from the PCI read operation. Extra bits are used to identify time labels and address words so that the software can recover if words are missing due to FIFO overruns. Figure 2, left, illustrates the monitor data path.

Another function that is performed is sequencing: generating spike traffic on the AER bus on the basis of pre-computed information (e.g., to emulate hardware components). The ability to inject pre-computed spike sequences into the AER bus also allows software simulations and VLSI hardware to be seamlessly interchanged. The *sequencer* is decoupled from the PCI bus using a FIFO. The sequencer checks the FIFO state and, if it is not empty, reads its content. Depending on the two most significant bits, it can either generate a transaction on the AER bus (a spike), or it can generate a relative or an absolute time delay. The transactions generated by the sequencer can be transmitted via any one of the four output channels. Figure 2, middle, illustrates the sequencer data path.

Finally, the *mapper* maps incoming AER-*in* to outgoing AER-*out* addresses, and can operate in three modes: *pass-through, one-to-one,* and *one-to-many*. In the pass-through mode, the AER-in address is simply replicated at AER out, whereas in one-to-one mode the AER-in address is used as a pointer to a look-up table. The retrieved content will be the target address at AER out. In the case of the one-to-many mode, the AER-in address generates multiple events to be dispatched to different targets. This is achieved by using the AER-in address as a pointer into the same look-up table as in one-to-one mode, but in this case the retrieved content is used as a further pointer to a list of AER-out addresses.

The mapper itself is composed of three main parts. First, the *mapper-in* forwards AER input, as events, to the mapper FIFO. It can also complete the AER transaction in the absence of a receiver chip. Second, the FIFO is needed to decouple the management of the AER-in and AER-out fluxes. Lastly, the *mapper-out* manages the look-up table scanning and the corresponding AER-out transactions. Figure 2, right, illustrates the mapper data path.

The main interface can adapt to different AER standards, and is connected via a cable to a small board that is directly connected to the AER bus. To enable the functionality of the board to be accessed robustly and conveniently from user programs, we have provided a device driver for Linux and, layered on top of this, a C library.

The open-source, GNU-General-Public-Licensed (GPL)[4] driver provides full integration of the PCI-AER board into Linux systems following the Unix *everything-is-a-file* model. This allows AE data streams to be read and written using standard Unix read and write calls and/or supports the use of standard shell redirection and command-line tools. The driver supports separate logical devices for each of the major functional blocks of the board—mapper, monitor, and sequencer—and can support multiple boards. It also ensures that the AE data streams being read or written remain coherent. For instance, it prevents multiple programs from attempting to read the monitored AE data at the same time, thus splitting the data stream unpredictably. It also forces word-multiple-sized reads

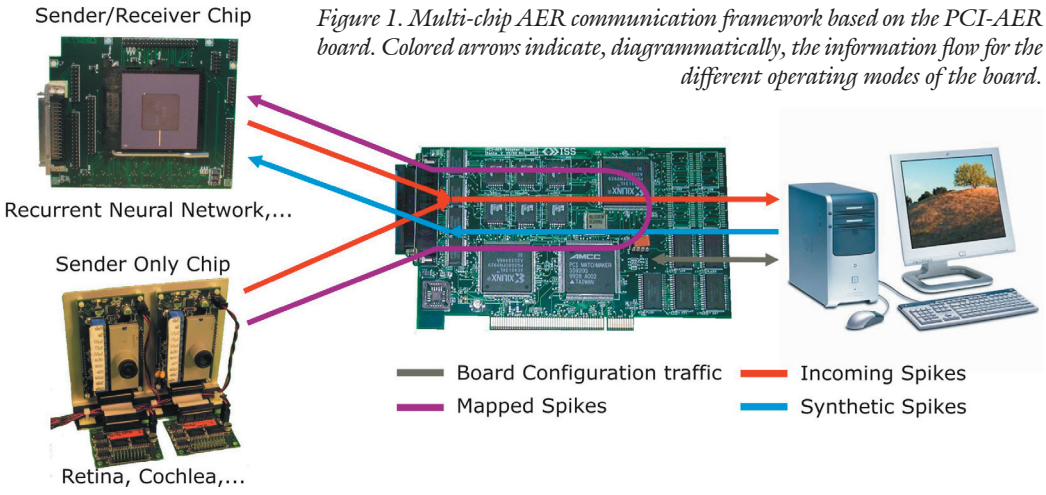


Figure 1. Multi-chip AER communication framework based on the PCI-AER board. Colored arrows indicate, diagrammatically, the information flow for the different operating modes of the board.

***Dante et al.,** continued on p. 6*

Dante, continued from p. 5

and writes to prevent corruption of the data streams due to misalignment problems. IOCTL (input/output control) calls are provided to set and get all possible sensible configuration states, and user programs are prevented from putting the board into an inconsistent state. The driver also performs memory management for the mapper SRAM (static random-access memory) so that the user does not need to perform necessary but onerous and error-prone indexing arithmetic. This helps the user to avoid accidentally corrupting the mapping table. At the time of writing, the driver is available for Linux kernel versions 2.4 and 2.6.

The library consists mainly of thin, fast wrappers for the driver functions (open, close, read, write, flush) and IOCTL calls. However, functions are also provided to convert from the PCI-AER hardware-specific format to a generic inter-spike interval/address-event format for reading, and vice versa for writing. The conversion function for reading also implements a state machine to attempt recovery when data are received out of order in the event of monitor FIFO overruns or other hardware errors.

Both driver and library are fully documented.

Some users have written small applications in C or C++ for spike-train generation and data logging directly using the API (application programming interface) provided by the library. Dylan Muir of the Institute of Neuro Informatics (INI) has developed a Matlab toolbox for the offline generation and manipulation of spike trains to be sent to, or read from, the PCI-AER board via library and driver. Matthias Oster (see bottom of page 9) has developed a client-server architecture on top of the library to enable the use of the board on-line from within Matlab, including real-time data display. Future developments should include a refinement of this client-server architecture to enable multiple data-sinks (including simple graphical display tools and other software packages) to read the monitored AE stream concurrently in a coordinated way. Other possible future developments include a command-line-driven configuration tool, Java support, and a stimulation tool for the on-line generation of AE patterns to drive the sequencer.

The design of the PCI-AER interface is now being developed to provide better integration of the AER-related functions with other needs, such as setting and control. We are aiming for better performance. In the meantime, the current board and associated software are being used by several groups (including two European-Union-funded projects), and are made available to interested research groups through INI thanks to a formal agreement between it and the ISS.

Analog VLSI: Circuits and Principles

Shih-Chii Liu, Jörg Kramer, Giacomo Indiveri, Tobias Delbrück, and Rodney Douglas

Publisher: MIT Press

Presents concepts required for the design of analog VLSI circuits. Topics include device physics, linear and nonlinear circuit forms, translinear circuits, photodetectors, floating-gate devices, noise analysis, and process technology.
http://www.ini.unizh.ch/~giacomo/book.html

Vittorio Dante, Paolo Del Giudice, and Adrian M. Whatley*

Complex Systems Unit
Department of Technologies and Health (ISS)
Rome, Italy
E-mail: {dante, paolo.delgiudice}@iss.infn.it
http://neural.iss.infn.it

*Institute of Neuroinformatics (INI)
Uni/ETH Zurich, Switzerland
E-mail: amw@ini.phys.ethz.ch
http://www.ini.unizh.ch/~amw

References

1. J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, *Winner-take-all networks of O(n) complexity,* **Advances in Neural Information Processing Systems,** pp. 703-711, Morgan Kaufmann, San Mateo, CA, 1989.
2. S. R. Deiss, R. J. Douglas and A. M. Whatley, *A pulse code communication infrastructure for neuromorphic systems,* **Pulsed Neural Networks**, pp. 157-178, MIT Press, Cambridge, MA, 1999
3. V. Dante and P. Del Giudice,*The PCI-AER interface board,* **2001 Telluride Workshop on Neuromorphic Engineering Report,** pp. 99-103, 2001: http://www.ini.unizh.ch/telluride/previous/report01.pdf
4. http://www.gnu.org/copyleft/gpl.html

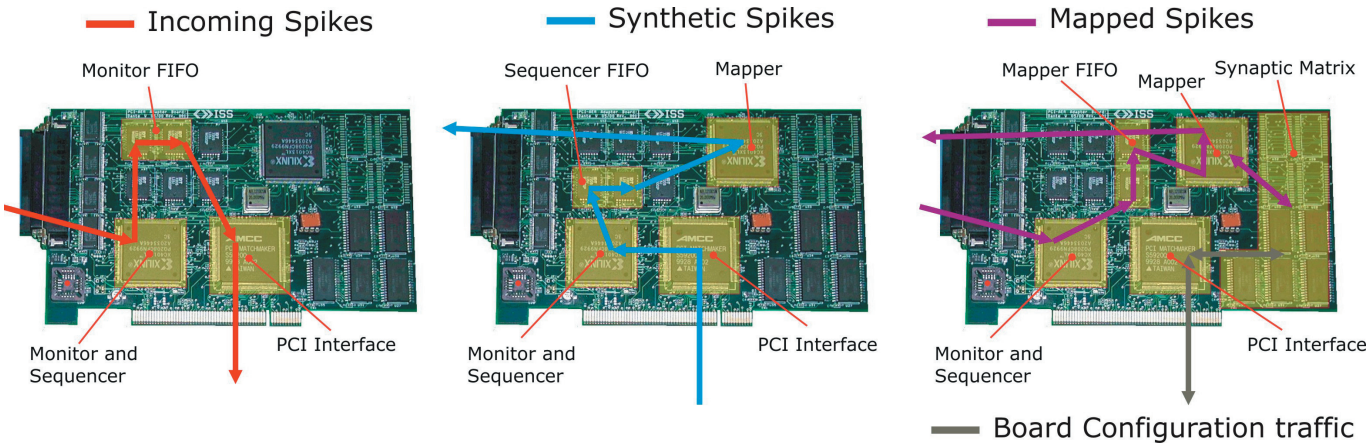


*Figure 2. The three panels show, from left to right, information flow for the **monitor**, **sequencer,** and **mapper** functionalities.*

***Indiveri,** continued from p. 4*

extended: we computed the probabilities of inducing long-term potentiation (LTP) and long-term depression (LTD) in all of the synapses of all of the chip's neurons (see Figure 3).

In this experiment we stimulated each synapse with Poisson-distributed spike trains via the PCI-AER board, and generated post-synaptic firing rates by injecting constant currents into the neurons. To measure the probability of LTP of a synapse we first reset its weight to its low (depressed) state and then applied pre- and post-synaptic stimulation. We subsequently determined the state of the synapse by measuring the response of the post-synaptic neuron to a regular pre-synaptic spike train. We repeated this procedure 50 times and applied it to all synapses of all neurons in the array.

To measure the probability of LTD of a synapse we applied a similar experimental protocol, but first initialized the synapse to its high state. The particular shape and the position of the region where LTP/LTD occurs can be modified by varying the parameters of the bi-stability and the STDP circuits. For low values of mean post-synaptic firing rates, LTP does not occur. For high and increasing values, on the other hand, the probability of LTP varies with a bell-shape dependence on the mean pre-synaptic firing rates. The probability of LTD follows a complementary tendency and is in accordance with theoretical predictions.[5]

The STDP learning circuits allow us to implement a learning mechanism useful for real-time unsupervised learning tasks. They also provide a mechanism to set arbitrary (bistable) synaptic weights in a supervised way, by producing appropriate pre- and post-synaptic firing rates, without requiring dedicated pins or wires for each synapse in the array. The AER communication infrastructure and devices of the type presented are actively under development by the neuromorphic engineering community. The simple examples described here indicate that this technology can be used reliably in massively parallel VLSI networks of I&F neurons for real-time simulation of complex spike-based learning algorithms.

Giacomo Indiveri
Institute of Neuroinformatics
Uni/ETH Zurich
Zurich, Switzerland
E-mail: giacomo@ini.phys.ethz.ch
http://www.ini.unizh.ch/~giacomo

References

1. R. J. Vogelstein, U. Mallik, and G. Cauwenberghs, *Silicon spike-based synaptic array and address-event transceiver,* **Proc. of IEEE Int'l Symp. on Circuits and Systems,** pp. 385-388, 2004.
2. F. Tenore, R. Etienne-Cummings, and M. Lewis, *A programmable array of silicon neurons for the control of legged locomotion,* **Proc. IEEE Int'l Symp. on Circuits and Systems,** pp. 349-352, 2004.
3. E. Chicca, D. Badoni, V. Dante, M. D. Andreagiovanni, G. Salina, S. Fusi, and P. Del Giudice, *A VLSI recurrent network of integrate and fire neurons connected by plastic synapses with long term memory,* **IEEE Trans. Neural Net. 14 (5),** pp. 1297-1307, September 2003.
4. S. -C. Liu and R. Douglas, *Temporal coding in a silicon network of integrate-and-fire neurons,* **IEEE Trans. on Neural Networks 15 (5),** pp. 1305-1314, September 2004.
5. G. Indiveri, E. Chicca, and R. Douglas, *A VLSI array of low-power spiking neurons and bistable synapses with spike timing dependent plasticity,* **submitted to IEEE Trans. on Neural Networks,** 2004.

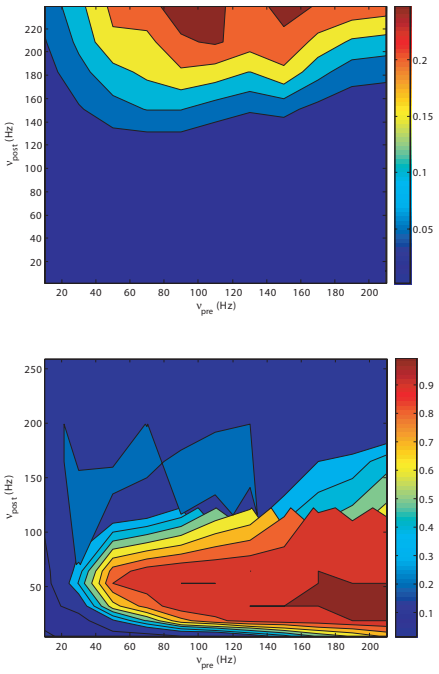


Figure 3. (top) Mean probability of LTP over all STDP synapses in the chip, as a function of pre- and post-synaptic firing rates. (bottom) Mean probability of LTD.

***Bernardet,** continued from p. 3*

reliably navigate to target locations. The software has also been used for models of classical conditioning, including a model of learning in the auditory cortex.[5,6]

The project *Ada: the intelligent space* for the Swiss National Exposition (Expo02) used Iqr as the basis for the large scale integration of multiple sensors and effectors.[7] Ada was developed to behave as an artificial creature: interacting with visitors in an intuitive manner, and reflecting our present understanding of neurobiological systems. Finally, a model that integrated the fly's elementary motion detector (EMD) for course stabilization and the locust lobula giant movement detector (LGMD) for collision avoidance (see Figure 1), was used to control the flight of a blimp.[8]

Ulysses Bernardet and
Paul F. M. J. Verschure
Institute of Neuroinformatics
Uni/ETH Zurich, Switzerland.
E-mail: {ulysses, pfmjv}@ini.phys.ethz.ch
http://www.ini.unizh.ch/~ulysses/iqr/

References

1. P. F. M. J. Verschure, T. Voegtlin, and R. J. Douglas, *Environmentally mediated synergy between perception and behaviour in mobile robots,* **Nature 425,** pp. 620-624, 2003.
2. P. F. M. J. Verschure, *Xmorph: a software tool for the synthesis and analysis of neural systems,* **Technical report, Institute of Neuroinformatics,** June 1997.
3. http://www.amoth.org
4. **Report on the 2004 Workshop On Neuromorphic Engineering,** 2004.
5. M. A. Sánchez-Montañés, P. F. M. J. Verschure, and P. König, *Local and global gating of synaptic plasticity,* **Neural Comput 12,** pp. 519-529, 2000.
6. C. Hofstoetter, M. Mintz, and P. F. M. J. Verschure, *The Cerebellum in Action: A Simulation and Robotics Stud,* **European J. Neuroscience 16,** p. 1361, 2002.
7. K. Eng, D. Klein, A. Bäbler, U. Bernardet, M. Blanchard, M. Costa, T. Delbrück, R. J. Douglas, K. Hepp, J. Manzo lli, M. Mintz, F. Roth, U. Rutishauser, K. Wassermann, A. M. Whatley, A. Wittmann, R. Wyss, and P. F. M. J. Verschure, *Design for a brain revisited: the neuromorphic design and functionality of the interactive space 'Ada',* **Rev. Neurosci 14,** pp. 145-80, 2003.
8. S. Bermúdez i Badia and P. F. M. J. Verschure, *A collision avoidance model based on the lobula giant movement detector neuron of the locust,* **Proc. Int'l Joint Conf. on Neural Networks,** pp. 1532-37, Budapest, 2004.

LABORATORY NOTES

Helping neuromorphic sensors leave the designer's desk

A variety of impressive neuromorphic sensors have been developed over the years, but hardly any have so far made it into applications outside carefully-controlled laboratory setups. Even when on the designer's desk, most neuromorphic sensors only operate well under tightly-controlled bias settings that require much experience and patience in twiddling potentiometers. To change this, we recently developed the sensor-control-interface board[1] (SCI). This provides a small and inexpensive microprocessor-controlled 'mobile laboratory environment' optimized for the low-noise operation of aVLSI circuits. Tuning and operation of chips in this environment is simpler and faster than previous procedures, because the sensor setup and data recording is managed by PC. Additionally, the SCI board can operate as a mobile stand-alone board, providing a variety of control signals for real-world systems such as small robots.

Hardware

The compact SCI board is easy to setup and operate: it only requires external power and—depending on the application—a connection to a PC at 921,600 baud over RS232, USB, or wireless Bluetooth. The board has a variety of connectors for external components, such as (neuromorphic) sensors, motors, servos, and additional microcontroller-boards. Many components plug directly into the SCI board: others require a small customized adapter board.

Before describing the input/output (I/O) -functionality in more detail, we should point out an important difference between the SCI board and most other interfaces. The SCI board consists of two separate, electrically isolated areas: the digital section (DS) and the analog section (AS). The few required signals between DS and AS are routed through optocouplers, so that DS and AS stay completely separate and do not even share a common ground level (see Figure 1). Devices on the DS—e.g. microprocessors or external motors—typically generate substantial electric noise on the power supply rails. Filtering this high-frequency noise to suit aVLSI requirements is almost impossible (see Figure 1b, left). However, the SCI board provides a virtually noise-free environment for analog devices (see Figure 1b, right). For lowest-noise operation, two separate power sources for DS and AS are required: typically two unconnected batteries or two separate desk power supplies.

Back to functionality, the AS has four connectors (P0-P3), each with sufficient flexibility to control a typical neuromorphic aVLSI sensor at either 3.3V or 5V. Each port has 16 analog output pins with 16-bit resolution, eight analog input pins with 12-bit resolution, eight binary output pins, eight binary input pins, and an analog power supply for external hardware. In a typical setup, the analog output pins generate bias voltages with a resolution of 0.1mV, the analog inputs read sensor data, and the binary I/O pins control clocks and scanners. The AS thus provides a low-noise environment with a total of 64 analog output voltages, 32 analog input channels, 32 binary outputs, and 32 binary inputs.

The DS offers a variety of different ports: four amplified outputs for bi-directional pulse-width-modulation-controlled motors, five control signals for standard hobby servos, an I²C (inter-IC bus)-compatible two-wire interface, and a digital expansion port (PD). The PD provides direct I/O connections to the main microprocessor, which operate at much higher speed than those on the AS. However, due to electric noise, the direct connections are not suitable for aVLSI sensors. The DS is typically used to connect to small motors on mobile robots, servos, or rotating drums to provide stimuli of varying velocities during vision-chip tuning. The major advantage of connecting devices to the DS is that the electric noise they introduce will not influence the aVLSI sensors attached to the AS.

Software

Two different options exist for working with the SCI board: sending individual actions using a command-line interface, or adding software to the on-board microprocessor. The simplest option uses the former to send atomic commands—such as *read voltage on port 3, pin 5*—from a terminal program. Alternatively, an open-source C library generates the atomic commands for custom software. Additional Matlab functions or XML-based GUIs (see bottom of p. 9) allow applications for tuning chips and running simple closed-loop control algorithms. All of these interactions, however, require an external PC and a communication link to the SCI board. They are useful for tuning sensors and visualizing data, but are rather inconvenient for robot control.

The advanced software option allows adding customized C functions to the existing open source program on the microcontroller. On-board software can directly call functions to set and read values, bypassing the command line interface to provide significantly-increased operational speed and fully-autonomous devices.

Current applications

A number of ongoing projects apply neuromorphic sensors mounted on SCI boards. We have developed a small autonomous holonomic-drive robot that carries the SCI board with neuromorphic sensors, e.g. for sound-source localization or object tracking (see Figure 2). This is a simple system used mainly in short-term student projects or to demonstrate the benefits of neuromorphic

***Conradt,** continued on p. 9*

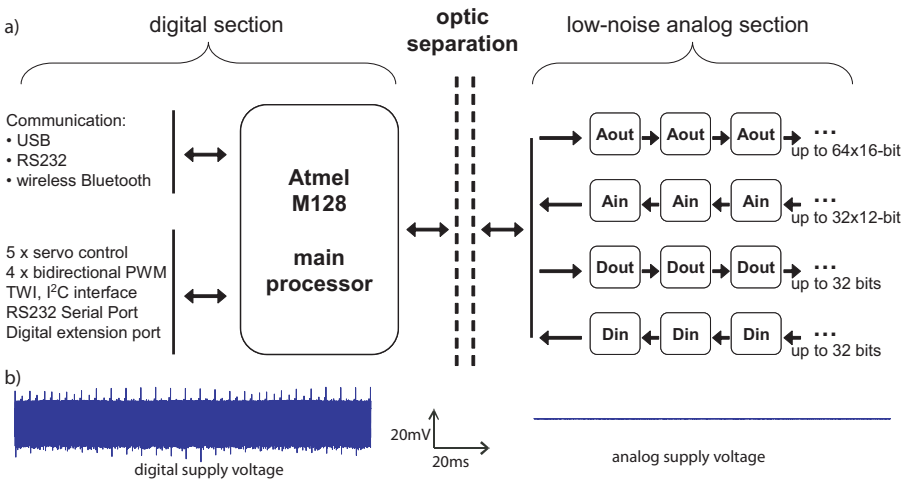


Figure 1. a) Simplified board schematics clearly showing the electrical separation between digital and analog sections. b) The graphs show low-pass-filtered power supplies while running electric motors (both traces plotted on same scale).

Conradt, Continued from p. 8

sensors over digital solutions. In another Institute-of-Neuromorphic-Engineering-funded project, a simplified SCI board stabilizes a flying blimp using global motion information from neuromorphic optic flow sensors. This project ultimately aims to combine input from a variety of sensors connected to the board. Other research groups are exploring the possibility of automatic chip tuning: simple algorithms may suffice to control external stimuli, sweep bias values, record responses from chips, and visualize data. In simple cases, a gradient-ascent procedure might also tune chip biases to their optimal settings.

Conclusion

The SCI board provides a flexible interface for exploring neuromorphic sensors in controlled environments on real-world devices. Our experience shows that students, without prior knowledge of chip design or bias tuning, can set up and operate small systems quickly and apply neuromorphic sensors easily. Until now, most laboratories have re-invented interfaces for their particular designs; we aim to make this unnecessary by creating a simple but flexible interface that works for a large variety of setups. Furthermore, the modular hardware design of the SCI can be reduced to optimize for cost- or weight-constraints: it thus provides an environment for aVLSI sensors to move from the desk into real life.

We are hoping to see the board used as a standard tool in the neuromorphic engineering community, and are therefore offering assembled and tested boards, schematics, open-source software, and documentation. For further information, please consult our web page.[1]

This project received funding from the Institute of Neuroinformatics and the Institute of Neuromorphic Engineering.

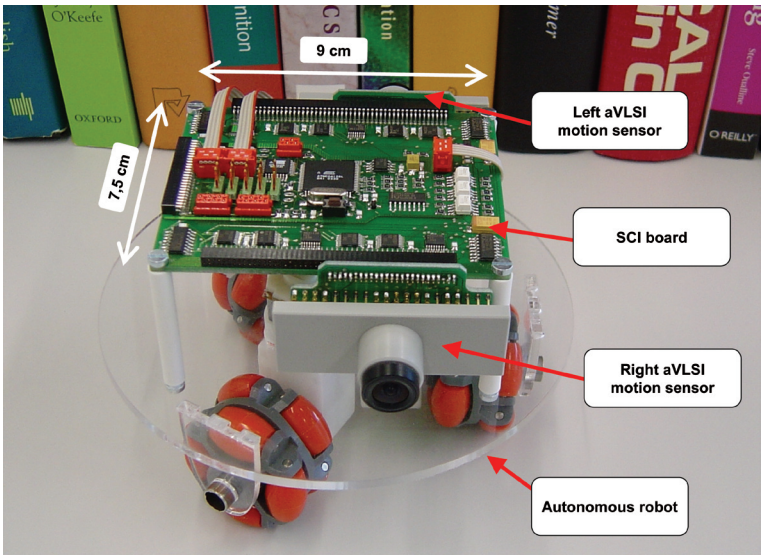


Figure 2: The SCI board with two neuromorphic motion sensors (optic flow) controlling a simple autonomous robot.

Jörg Conradt
Institute of Neuroinformatics
Uni/ETH Zurich, Switzerland
E-mail: conradt@ini.phys.ethz.ch
http://www.ini.ethz.ch/~conradt

Reference

1. http://www.ini.ethz.ch/~conradt/projects/SensorControlInterface

Tuning aVLSI chips with a mouse click

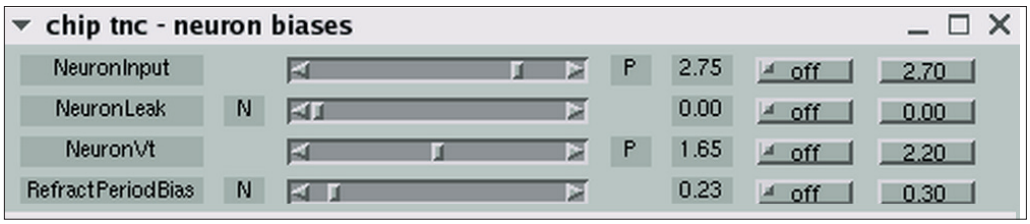
Assuming we use a computerized I/O board for an aVLSI chip like the SCI, what are the advantages compared to twisting a potentiometer for tweaking the bias voltages of an aVLSI chip? To facilitate the tuning and characterization of the chips, we have written software that encapsulates all the low-level functionality of a computerized I/O board in the MATLAB environment. This includes a graphical user interface like the one shown here, which allows setting bias voltages with a single mouse click, by just moving a slider, or by pressing a button. Complete tuning sets can be saved and restored for later experiments.

The user interface is automatically created based on information stored in a database which defines the pinout of chips and boards in an easy-to-use and versatile text language (XML) that makes it simple to include new chip/board definitions. The database allows access to the biases directly by name, making scripts easier: for example, sweeping a bias obtains measurement curves for characterization. In the more advanced case, the bias parameters are automatically optimized to achieve the best performance: that is, tuning aVLSI chips *without* even a single mouse click.

The interface to the underlying hardware is kept simple so that the user interface can easily be adapted to a variety of I/O boards. Currently it is used by researchers in three different projects inside and outside the Institute of Neuroinformatics and on two types of I/O boards.

For the database and the user interface software, see:

http://www.ini.ethz.ch/~mao/ChipDatabase/



Matthias Oster
Institute of Neuroinformatics
Uni/ETH Zurich, Switzerland
E-mail: mao@ini.phys.ethz.ch

***Hasler,** continued from p. 1*

computational analog block (CAB). In current CMOS technologies, one can envision thousands of CABs on a single IC, similar to the number of computational blocks for FPGAs. In minutes, a single FPAA device can be configured to implement several different circuit topologies that can be tested and compared. In addition, modern FPAAs can contain analog-to-digital converters (ADC) and digital-to-analog converters (DAC) that ease the interfacing of analog systems with digital logic implemented on FPGAs and/or microcontrollers.

Figure 2 shows one early instantiation of this element: a 64-block FPAA based around the CAB shown in Figure 1b, the functionality of which is enhanced by a mixture of medium- and coarse-grained computational blocks similar to many modern FPGA designs. These blocks were carefully selected to provide a sufficiently-flexible, generic architecture while optimizing certain frequently-used signal-processing blocks. The high-level CABs used in this design consist of a capacitively-coupled current conveyor (C_4) used as a bandpass filter module, a peak detector, and the 4×4 vector-matrix multiplier block. In general, the C_4 module provides a straightforward method of sub-banding an incoming signal. This allows Fourier analysis analogous to performing a fast Fourier transform (FFT) in the digital domain. The vector-matrix multiplier block allows the user to perform a matrix transformation on the incoming signals.

FPAAs introduce new opportunities to improve analog circuit design and signal processing education. By providing students with an easily reconfigurable, general-purpose analog device, laboratory projects can grow in scope and functionality. FPAAs are a natural fit in analog circuit courses, either in basic or more-advanced circuits courses, allowing students to use FPAAs to implement and test multiple circuit designs within a single laboratory period.

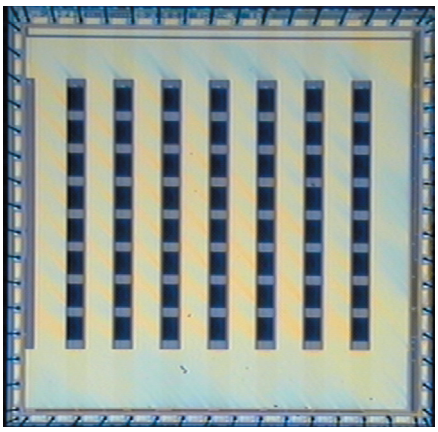
When these FPAA chips become available, we envision them impacting the teaching of neuromorphic IC circuits, the design efforts of early neuromorphic IC research, and the design of larger neuromorphic testbeds. The number of exploratory IC fabrication runs should be significantly reduced, particularly when investigating effects not well modelled by simulation tools. Initial teaching in neuromorphic IC design might no longer require training students to be proficient at IC layout or building custom test setups to show system functionality, but rather focus on the concepts being investigated. Eventually, custom ICs can be derived from the corresponding experimental results. FPAAs could expand to a set of ICs that include specialized blocks including a reconfigurable set of biological channels and synapses,[11] front-end cochlea models, on-chip sensors (e.g. pixel arrays for vision applications), and specialized digital interfaces (i.e. AER).

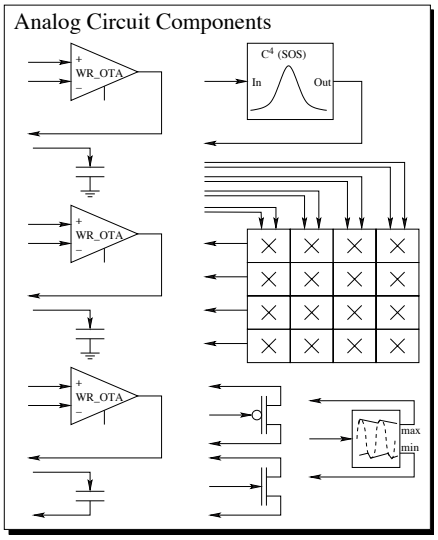
Paul Hasler, Tyson S. Hall*, and Christopher M. Twigg
Georgia Institute of Technology
Atlanta, GA
E-mail: phasler@ece.gatech.edu
*Southern Adventist University
Collegedale, TN

References

1. P. Chow, S. O. Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, *The design of an SRAM-based field-programmable gate array-part I: architecture,* **IEEE Trans. on Very Large Scale Integration (VLSI) Systems 7** (2), pp. 191-197, June 1999.
2. M. A. Sivilotti, *Wiring Considerations in Analog VLSI Systems, with Application to Field-Programmable Networks (VLSI),* **Ph.D. thesis,** California Institute of Technology, Pasadena, CA, 1991.
3. P. G. Gulak, *Field-programmable analog arrays: past, present and future perspectives,* **IEEE Region 10 Int'l Conf. on Microelectronics and VLSI,** pp. 123-126, November 1995.
4. David Marsh, *Programmable analogue ICs challenge spice-and-breadboard designs,* **EDN Europe,** pp. 30-36. Reed Business Information, http://www.ednmag.com, October 2001.
5. Fast Analog Solutions Ltd., http://www.zetex.com, **Totally reconfigurable analog circuit-TRAC R,** March 1999.
6. B. Ray, P. P. Chaudhuri, and P. K. Nandi, *Design of OTA based field programmable analog array,* **Proc. 13th Int'l Conf. on VLSI Design,** pp. 494-498, January 2000.
7. B. Pankiewicz, M. Wojcikowski, S. Szczepanski, and Y. Sun, *A field programmable analog array for CMOS continuous-time OTA-C filter applications,* **IEEE J. Solid-State Circuits 37** (2), pp. 125-136, Febuary 2002.
8. T. Hall, D. Anderson, and P. Hasler, *Field-Programmable Analog Arrays: A floating-gate Approach,* **12th Int'l Conf. on Field Programmable Logic and Applications,** Montpellier, France, September 2002.
9. T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, *Application performance of elements in a floating-gate FPAA,* **Proc. 2004 IEEE Int'l Symp. on Circuits and Systems,** pp. 589-592, May 2004.
10. M. Kucic, A. Low, P. Hasler, and J. Neff, *A programmable continuous-time floating-gate Fourier processor,* **IEEE Trans. on Circuits and Systems II 48** (1), pp. 90-99, January 2001.
11. E. Farquhar, D. Abramson, and P. Hasler, *A Reconfigurable Bidirectional Active 2-Dimensional Dendrite Model,* **Int'l Symp. on Circuits and Systems,** Vancouver, 2004.

Figure 2. Large-scale FPAA implementations. Examples of working compiled systems include sub-banding amplitude detectors, high-order ladder filters, and diffusor networks that make use of the routing fabric. Top: Die photograph of our first large-scale FPAA with 64 CABs (computational analog blocks), and over 150,000 analog programmable floating-gate devices in a 6mm×6mm area (0.35μm process). Bottom: The CAB design for this FPAA, based upon experimental results from smaller CAB arrays.[9]





Next deadline...

... is Friday 7 October 2005. Full details available at:

http://www.sunnybains.com/neuroeng.html

BOOK REVIEW

CMOS Imagers: From Phototransduction to Image Processing

Orly Yadid-Pecht and Ralph Etienne-Cummings (Eds.)
Publisher: Kluwer Academic Publishers
Hardcover: May 2004, 850pp, $125.00 **ISBN:** 1-4020-7961-3

This is the first book written on CMOS imagers and their application to focal-plane image processing, and will be of interest to those already working on analog circuit design and/or building image processing systems for various applications. In particular, the book presents the technology crucial for building neuromorphic vision systems, and should therefore be of great interest to this community.

I myself am interested in building human-like intelligent systems based on state-of-the-art silicon VLSI technology. To this end, CMOS imagers form an integral part of the system including ROI (region-of-interest) detection, motion detection, feature extraction from images, and so forth. Several of my students have worked on projects related to this topic, but—until now—there were no good textbooks that provided an overview for newcomers to the technology. In this respect I found the book very useful: it contains sufficient information for beginners to start their projects, as well as a lot of ideas and hints to guide students to develop their own new ideas in focal-plane processing. I believe the book is very valuable not only for Ph.D. students, but also for researchers and engineers in industries interested in silicon technology. Not in the traditional Moore's-Law-driven digital technology, of course, but the more fascinating analog neuromorphic or bio-inspired VLSI systems.

Though CCD has been the mainstream technology for image acquisition, CMOS imagers are now invading the market. Their advantages include low-voltage, low-power operation, and compatibility with standard CMOS fabrication processes. Of particular importance is the nondestructive readout of pixel data in CMOS imagers, enabling various kinds of image processing operations to be performed in the focal plane.

The first four chapters provide readers with sufficient practical information to start designing their own CMOS imagers. The book starts with an introduction to silicon device physics in relation to the photon-to-electronic-charge conversion in Chapter 1, which is followed by the analysis of the modulation transfer function (MTF) of CMOS active pixel sensors (APSs) in Chapter 2. In Chapter 3, a semi-analytical model is developed for the estimation of APS photo response. This is based on experimental data incorporating the effect of substrate diffusion of photo-generated carriers as well as geometrical shape and size of the photodiode active area. Review of APS design is presented in Chapter 4 from the very basics to more advanced system-on-chip examples. Smart vision-system-on-a-chip and tracking sensors are presented as examples of CMOS imagers with integrated intelligence.

Next, more examples of advanced focal-plane-processing applications are presented. Chapter 5 discusses three systems for imaging and visual-information processing at the focal plane, using three different representations of the photon flux density: current-mode, voltage-mode, and mixed-mode. The chapter also outlines how spatiotemporal image processing can be implemented. Chapter 6 looks at a stochastic adaptive algorithm for on-line correction of spatial non-uniformity in random-access addressable imaging systems. An adaptive architecture is implemented in analog VLSI, and is integrated with the photo sensors on the focal plane. The idea presented is particularly attractive for compact implementation using floating-gate MOS circuits.

The book is written by well-recognized leading researchers in the area. I believe its publication is timely, that it will serve as a good reference for this area of technology, and that it will prove invaluable for both experienced and novice designers.

Tadashi Shibata

Tadashi Shibata is Professor of the Department of Frontier Informatics at The University of Tokyo. He is currently developing intelligent VLSI systems based on psychological brain models using state-of-the-art silicon technology.

Liu & van Schaik, *continued from p. 12*

ing cochleae will enable research into models of computational systems in the cortex that process sound signals and areas of multi-sensory fusion.

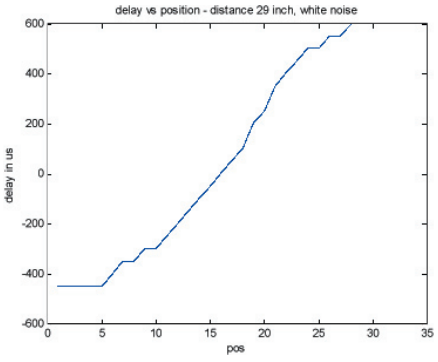
The work has been supported by the Institute of Neuromorphic Engineering.

Shih-Chii Liu and André van Schaik*
Institute of Neuroinformatics
Uni/ETH Zurich
Zurich, Switzerland
E-mail: shih@ini.phys.ethz.ch
*School of Electrical and Information Engineering
University of Sydney
Sydney, Australia
E-mail: andre@ee.usyd.edu.au

References

1. A. van Schaik and S. Shamma, *A neuromorphic sound localizer for a smart MEMS system,* **Analog Integrated Circuits & Signal Processing 39,** pp. 267-273, 2004.
2. A. van Schaik, E. Fragnière, and E. Vittoz, *Improved Silicon Cochlea using Compatible Lateral Bipolar Transistors,* **Advances in Neural Information Processing Systems 8,** MIT Press, Cambridge MA, pp. 671-677, 1996.
3. A. van Schaik, *An analog VLSI model of periodicity extraction in the human auditory system,* **Analog Integrated Circuits & Signal Processing 26,** pp. 157-77, 2001.
4. A. van Schaik and R. Meddis, *Analog very large-scale integrated (VLSI) implementation of a model of amplitude-modulation sensitivity in the auditory brainstem,* **J. of the Acoustical Society of America 105,** pp. 811-821, 1999.
5. K. A. Boahen, *Communicating neuronal ensembles between neuromorphic chips,* **Neuromorphic Systems Engineering,** Kluwer Academic Publishers, Boston MA, pp. 229-259, 1998.

Figure 3. Estimated delay as a function of position for a white-noise sound source. The source was 21in (53cm) away from the center of two microphones that were placed 7 in (18cm) apart. The orientation of the axis between the two microphones, relative to the line from the source to the their center, was varied between 0-180° in 28 steps.



An AER Ear

Silicon retinas with spiking outputs are currently the primary sensory input to spike-based multi-chip address-event representation (AER) systems. The availability of further sensory inputs has been long desired. Various implementations of silicon cochlea have been described over the years but, to date, none have had spike outputs that could easily interface with the existing AER systems. We now have a spike-based silicon auditory chip that can interface to current multi-chip AER systems. Our chip has two matched 32-section silicon cochlea plus simplified inner-hair-cell circuits that also generate auditory nerve spikes. These spikes are encoded using the now well-known and well-used AER circuits.

The silicon cochlea

The cochlea we used is identical to the one we presented in Reference 1, and a modern version of that presented in Reference 2. The latter had already proven its use in other neuromorphic sound processing systems.[3,4]

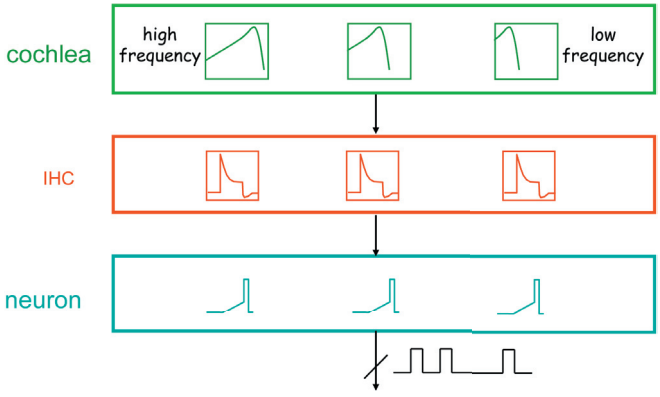
The basic building block for the filters in this cochlear model is a second-order low-pass filter section composed of transconductance amplifiers operating in weak inversion. Our device is implemented by cascading 32 of these second-order low-pass sections with exponentially decreasing cut-off frequencies. The exponential decrease is obtained by creating the bias currents of the second-order section with CMOS (complementary metal oxide semiconductor)-compatible lateral bipolar transistors.[2] The frequency response can be made to vary from 50Hz to 50kHz.

AER output

In biological cochlea, the inner hair cells (IHCs) are located along the basilar membrane where they detect deflections and convert these into spikes. We model the response of an IHC by transforming the band-pass-filtered output of the cochlear section into a single-ended current. We then pass a rectified version of this current to a first-order log-domain low-pass filter. The cut-off frequency of this latter filter should be set around 1kHz to model the reduction in phase-locking observed in biological IHCs.

The output of the low-pass filter is used to drive an integrate-and-fire neuron. This interfaces with arbitration AER circuits on the side to allow transmission of its spikes off chip.[1] When a neuron pulls the request line low, the arbiter will check if the data bus is free. If it is, it will put the neuron's address on the bus and acknowledge the neuron, which will cause it to be reset.

Figure 1. Block description of one of the cochleae on the chip. The output of each cochlea stage goes to an inner-hair-cell (IHC) circuit that supplies a rectified version of the cochlea output to a neural circuit. The output of the neuron goes to AER arbitration circuits.



Measurement results

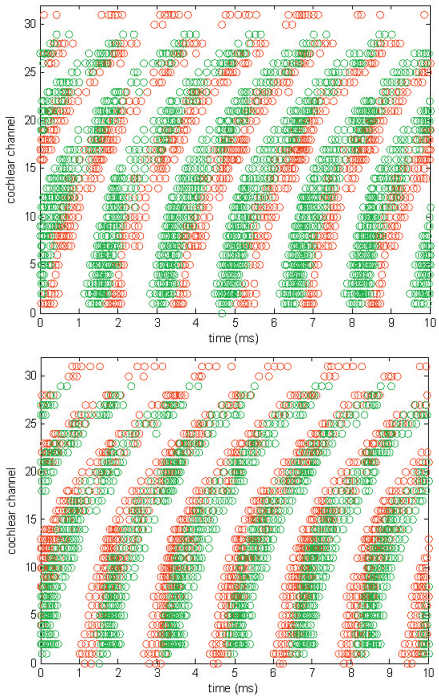
The spikes collected from the cochlea are illustrated in Figure 2 for an experiment in which two microphones are placed at different distances from a sound source for input. In this figure, called a cochleogram, the spikes are plotted for both the left (green) and the right (red) cochlea as a function of channel number versus time.

The sound source is closer to the left cochlea in Figure 2 (top), and to the right side in Figure 2 (bottom). It is clear that the spike bursts show the 600Hz periodicity of the test tone. Because of the limited velocity of sound, the spikes in the 'ear' closest to the sound source lead the spikes in the 'ear' furthest away within each channel. Furthermore, we see that as the sound travels though the cochlea to the higher channel numbers (with lower cut-off frequency) the delay increases exponentially, resulting in the typically curved appearance of the spike crests in the cochleogram.

Spatial localization

With the availability of these sensors with event-based outputs, we can start investigating different computational functions that are traditionally solved using continuous representations of the outputs of the spiking cells such as firing rates. Different types of spike encoding can be examined in this context: for example, the rate and temporal codes that are used in particular neural systems. One well-studied problem in audition is the location of a sound source from binaural auditory cues. One solution to this problem is to calculate the binaural cross-correlation by correlating the spike trains of each channel in the left cochlea with that from the corresponding channel in the right cochlea. An average across the 32 cochlear channels can then be obtained, of which the maximum will represent the estimated inter-aural time delay (ITD). Determining the maximum in the averaged cross-correlation for all positions yields Figure 3, which illustrates that it is possible to get a reasonable estimate of delay.

Figure 2. Spike output of the AER ear with a 600Hz pure tone at (top) 45° to the left and (bottom) 30° to the right.



As the microphones were rotated by hand through 180°, we do not have sufficiently accurate position measurements to work out the theoretical ITD, but we can see that the curves have the correct shape. A more rigorous investigation of sound localization with the AER ear is currently in progress.

The availability of the silicon spik-

Liu & van Schaik , cont. on p.11