THE NEUROMORPHIC ENGINEER

VOLUME 1 NUMBER 2 AUTUMN 2004

In this issue:

Neuromorphic approaches to rehabilitation

Research over the last forty years has shown that the spinal cord, even when completely disconnected from the brain, retains the ability to produce rhythmic patterns of output that can control relatively simple actions such as walking. This capacity, first demonstrated in the cat, could not be demonstrated in humans. However, more recent evidence suggests that the human spinal cord can control leg muscles with walking-like patterns even when completely disconnected from the brain.[1] This discovery promises exciting advances in rehabilitative technologies for individuals with spinal cord injury.

Current research in therapy after spinal cord injury is studying the benefit of weight-bearing stepping on regaining motor function. In a typical setup, the injured patient is partially supported by a harness and held over a moving treadmill: therapists on either side of the patient assist with motion and foot placement. Although the patients can produce the basic pattern of stepping, they often lack the ability to pull up their foot prior to placing it at the beginning of a stride, a deficit referred to as *foot drop*. Assistance in this portion of their gait is provided by therapists, but it may be desirable to have an automated sensing-actuation system to provide this assistance.

It is generally thought that neural oscillatory circuits in the spinal cord, called central pattern generators (CPG), underlie the production of rhythmic motor behavior such as locomotion.[2] To assist recovery, it may be desirable to build a hybrid circuit containing the injured CPG and artificial sensing, computation, and actuation elements to attempt to correct for CPG deficits after injury. During the 2004 workshop in Telluride, we considered the sensing and computation components of the artificial CPG (see Figure 1).

It has been shown that partial weight-bearing is extremely important for eliciting motor patterns in spinal-cord injured individuals, suggesting pressure input from the foot is a key component of normal locomotion. Taking our lead from this result, we used force-sensitive resistors (FSR, Interlink Electronics) placed on the bottom of each foot to measure contact pressure (Figure 1a). Output from the FSR was read into a computer for on-line processing (Figure 1b).

We used a single oscillating integrate-and-fire neuron model, with the stride frequency as its single input to the neuron, to simulate the CPG (Figure 1c). The goal was to use the CPG output, which was also at the stride frequency, to estimate the timing of ankle actuation so that foot drop could be avoided. It was, therefore, important to have the CPG oscillation at the appropriate phase of the step cycle. To accomplish this, the neuron had its voltage set to half of threshold value at the time of each foot strike. Since the neuron was tuned to have a voltage threshold of 1, this reset ensured that the neuron would fire a spike halfway through the step cycle. As with biological CPGs, the occurrence of such a spike could signal the actuation of a particular joint. Thus, since this circuit generates an artificial CPG that produces spikes at accurate points during the step cycle, its output could be used to actuate the ankle through external pneumatic actuators, and thus reduce the problem of foot drop (Figure 1d).

The artificial CPG neuron that we have studied represents the simplest possible implementation of a hybrid biological/neuromorphic CPG. It is likely that more realistic CPG model circuitry will allow the integration of more complex cues from the patient's gait pattern, and may be able to correct gait abnormalities more fully. It will be interesting to explore the potential of artificial CPG circuits, acting in parallel with their injured biological counterparts, for influencing plasticity and recovery in the injured nervous system.

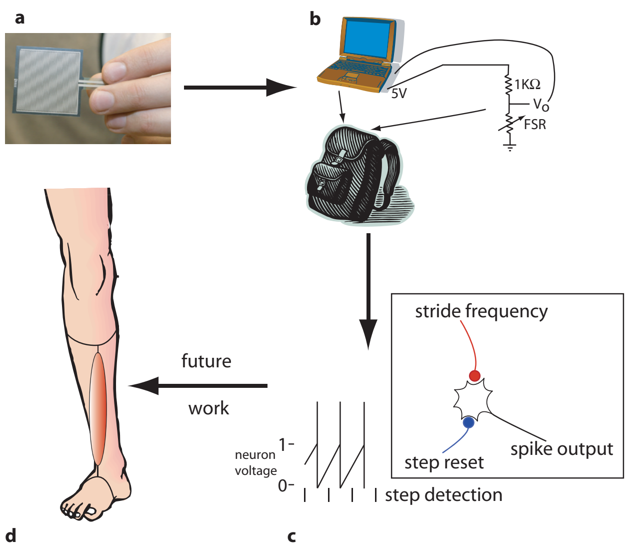

Figure 1 (a) The force-sensitive resistor (FSR) used in this application. (b) Voltage-divider circuit with FSR, showing that power and data acquisition were provided by a laptop computer, and components were placed in a backpack allowing the user to move freely. (c) The central pattern generator (CPG) neuron received stride frequency as direct input, step time as a reset, and put out spikes in steady state with the desired frequency and phase relative to the step cycle (step times shown as vertical lines). (d) Future work would involve coupling the CPG neuron to an actuator capable of lifting the foot.

***Kutch,** continued p. 2*

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

Available on the web at:
http://www.ine-web.org

The Institute of
Neuromorphic Engineering
Institute for Systems Research
AV Williams Bldg
University of Maryland
College Park, MD 20742
http://www.ine-web.org

Word-serial address-event transceiver layout compiler

Neuromorphic systems use multiple neuron arrays to implement large systems that are both modular and scalable. These arrays, which may be on separate chips, communicate with each other using the address-event representation (AER), where each neuron in an array is assigned a unique binary address that is encoded and transmitted to other arrays when it spikes.[1] To facilitate the design of these neuromorphic systems, our lab has designed software for automatic placement and routing of AER transceiver circuitry and neuron arrays, and for padframe generation, pad routing, and design verification.

Our layout compiler—ChipGen, implemented using Tanner Inc.'s L-Comp tool—is continually modified to incorporate the latest developments in AER communication. Several people, including Kwabena Boahen, Kareem Zaghloul, and Brian Taba, have contributed to its development over the years. In its present incarnation, ChipGen utilizes word-serial address events, where the transmitter encodes all of a row's events in a single burst: the row's address followed by a column address for each event.[2] Similarly, the receiver decodes the burst into a row-wide data-word that is written to the selected row.[3]

In addition to compiling AER transmitter and receiver circuitry, ChipGen provides the option to include a scanner for the continuous sensing of currents through a clock-driven multiplexor. A more sophisticated scanner that outputs signals to a standard VGA monitor[4] may also be selected, in lieu of the transmitter.

To use ChipGen, the designer must first create a metapixel, which ChipGen tiles to create a neuron array that will use AER to communicate with others within the larger neuromorphic system. The metapixel contains a user's custom-designed neural model as well as standard circuitry for AER communication. This neural model can range from a single spiking neuron to a complex arrangement of dendrites, somas, and both excitatory and inhibitory synapses. A multiplicity of neurons is supported by assigning more than one row or column per metapixel.

The final step in the chip-design process is layout verification. To this end, we have developed a netlist generator, NetGen, that creates a SPICE netlist of the chip. This can then be compared to a netlist extracted from the layout in a procedure known as LVS (layout versus schematic). Starting with the original version written by Kai Hynna a few years ago, NetGen also continues to evolve in step with ChipGen.

We compiled ChipGen into a dynamically-linked library (DLL): it is loaded as a user programmable interface (UPI) macro in Tanner's L-Edit Pro version 11.[5] Our cell library is currently laid out in MOSIS SCN_DEEP (deep submicron) rules. Taiwan Semiconductor Manufacturing Company's (TSMC) 0.25μm CMOS process is the most advanced technology we have used to fabricate chips so far. NetGen is implemented as a stand-alone windows executable program.

Using ChipGen and NetGen, the chip designer can take their neural model and create a verified chip layout within minutes. Both of these programs, in addition to our AER cell library, are freely available upon request. More information can be obtained from our website.[6] In the near future, we plan to host a hands-on workshop that will go through the process of chip layout and design verification.

Joseph Lin
Neuroengineering Lab
University of Pennsylvania
E-mail: linjh@seas.upenn.edu

References
1. M. Mahowald, **An Analog VLSI Stereoscopic Vision System**, Kluwer Academic, Boston, MA, 1994.
2. K. Boahen, *A Burst-Mode Word-Serial Address-Event Channel-I: Transmitter Design*, **IEEE Trans. on Circuits and Systems I 51** (7), pp. 1269-1280, July 2004.
3. K. Boahen, *A Burst-Mode Word-Serial Address-Event Channel-II: Receiver Design*, **IEEE Trans. on Circuits and Systems I 51** (7), pp. 1281-1291, July 2004.
4. C. A. Mead and T. Delbrück, *Scanners for visualizing analog vlsi circuitry*, **Analog Integ. Circuits Signal Process. 1**, pp. 93-106, 1991.
5. *http://www.tanner.com/EDA/products/ledit/*
6. *http://www.neuroengineering.upenn.edu/boahen/meth/fs_tools.htm*

Kutch, Continued from p. 1

Jason J. Kutch
Department of Mathematics
University of Michigan, USA
E-mail: jkutch@umich.edu
http://www.math.lsa.umich.edu/~jkutch

References
1. M. Maegele, S. Muller, A. Wernig, V.R. Edgerton and S.J. Harkema, *Recruitment of spinal motor pools during voluntary movements versus stepping after human spinal cord injury,* **J. Neurotrauma 19,** pp. 1217-29, October 2002.
2. V. Dietz, *Spinal cord pattern generators for locomotion,* **Clinical Neurophysiology 114,** pp. 1379-1389, August 2003.

Visual target tracking based on fly figure-detection cells

Insects are supremely successful biological autonomous systems that, despite their diminutive size, have survived in the real world and adapted to changing physical environments for more than 400 million years. Insects, which were highly successful long before the existence of mankind, are today the most species-rich and diverse of all animal taxa. Incorporating a vast array of sensors—far beyond the fondest dreams of modern roboticists—flying insects smoothly fuse the output of multimodal sensor arrays to achieve complex flight and landing trajectories. At the same time, they adapt robustly to sensor failure, and all of this is accomplished with a brain of around a million neurons.

With the goal of endowing an autonomous robot with a robust visual-tracking system, we have taken inspiration from the visual system of the fly. Male flies are extremely proficient at tracking females in flight. The exact mechanisms for this are as yet unknown, but figure detection (FD) cells in the brain of a fly have been identified that are sensitive to the motion of small objects,[1] and these cells have been suggested to underlie such tracking behavior. A computational model of these cells was proposed to describe the underlying neuronal architecture,[2] based on relative motion of the target and background features.

We put this computational model of fly vision, for the first time, in a simulated closed-loop tracking situation, and discovered that the model—which was devised to agree with biological data taken from flies—required some modifications to accomplish the tracking task.[3] Some biologically-plausible modifications, made to properly account for the expansive optical flow encountered during forward translation, ensured that the simulated fly never turned away from a target and strengthened the response to small moving targets. The result was excellent tracking performance even in cluttered visual scenes (see Figure 1). In fact, it turns out that the model actually *requires* a strong visual motion signal from background features in order to strongly respond to a contrarily moving target.

After our success in simulation, we proceeded to design a hardware implementation of the model for target tracking (see Figure 2). Despite the complexity of the model, the required computations can be accomplished completely in continuous-time analog circuitry by the cooperative activity of two vision chips representing the two compound eyes of the fly. No microcontroller is required to process the output of this two-chip vision system in order to steer the robot towards moving targets. Each pixel of these VLSI vision chips (see Figure 3) includes a low-level visual motion detector and several stages of analog circuitry to compare local motion signals with the wide-field motion response in order to respond only to small moving targets.

Using an LCD (liquid-crystal display) visual stimulus, we have been able to show that the response of the chip to moving targets of varying size is quite comparable to that of the fly FD cells upon which the computational model was based.[4] We are currently experimenting with the use of this hardware system for robotic target tracking: with promising initial success.

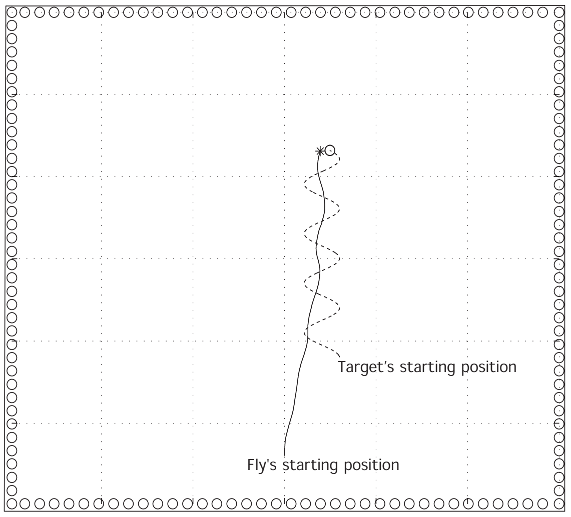

Figure 1: The path of the simulated fly as it tracks a moving object during a simulation experiment. Circles around the periphery are fixed background objects. The solid line shows the path of the fly, and the dashed line the path of the moving target. As the fly moves, the raw visual motion it detects is completely dominated by the background objects, and yet it tracks the target by its relative motion.

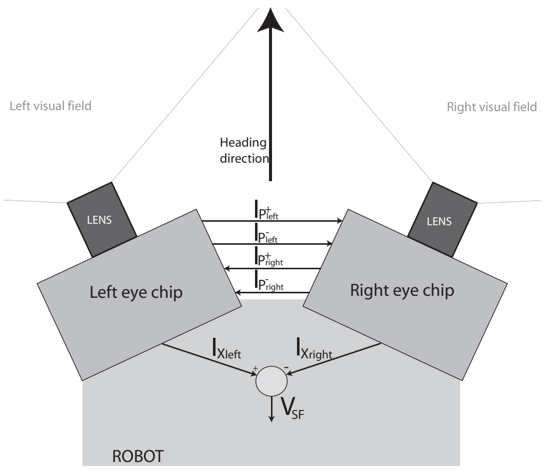

Figure 2: High-level hardware architecture of the tracking model implementation. Two vision chips with divergent fields of view interact via analog currents to produce a tracking output used to guide the robot.

Charles M. Higgins and Vivek Pant
Electrical and Computer Engineering
ARL Division of Neurobiology
University of Arizona
E-mail: {higgins,viv_pant}
@neuromorph.ece.arizona.edu
http://neuromorph.ece.arizona.edu

References
1. M. Egelhaaf, *On the neuronal basis of figure-ground discrimination by relative motion in the visual system of the fly. II. Figure detection cells, a new class of visual interneurons,* **Biological Cybernetics 52**, pp. 195-209, 1985.
2. W. Reichardt, M. Egelhaaf, and A. K. Guo, *Processing of figure and background motion in the visual system of the fly,* **Biological Cybernetics 61**, pp. 327-345, 1989.
3. C. M. Higgins and V. Pant, *An elaborated model of fly small target tracking,* **Biological Cybernetics,** *in press*.
4. C. M. Higgins and V. Pant, *A biomimetic VLSI sensor for visual tracking of small moving targets,* **IEEE Trans. on Circuits and Systems I,** *in press.*

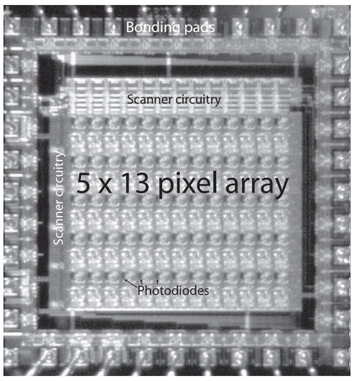

Figure 3: Photo-micrograph of prototype VLSI tracking sensor.

Prototyping neural networks for legged locomotion using custom aVLSI chips

From the very inception of the field of neuromorphic engineering, a significant part of the community focused its efforts on the development and the implementation of arrays of silicon neurons for the rapid prototyping of neural networks. One of the first successful silicon neuron implementations was proposed by Mahowald and Douglas in the early nineties.[1] This circuit generated spikes that had striking similarities with the ones produced by real cortical neurons, as it implemented a 'conductance-based model' that emulated the various ion currents responsible for producing nerve impulses. This approach, still being pursued by various groups today,[3-5] captures many details of how a real neuron works. However, it also requires the ability to successfully tune a large number of parameters, and typically takes up a large area of silicon real estate (due to a large number of transistors, large capacitors, or both).

Alternatively, a less sophisticated but smaller type of circuit design, is based on the integrate-and-fire (I&F) neuron model. This type of circuit, thanks to its smaller silicon-area requirements, has been frequently used both to implement large arrays of neurons on single chips, and in conjunction with the address-event-representation (AER) communication infrastructure, for transmitting spikes (events) among different chips. (This has been explained in the previous issue of the Neuromorphic Engineer[6] and elsewhere[7-8].) One of the first implementations of an I&F neuron model was the 'axon-hillock' circuit, originally proposed by Mead et al..[2]

Every year, the Neuromorphic Engineering workshop in Telluride, Colorado, allows scientists from many different disciplines and parts of the world to come together to discuss—and try to engineer—ideas into robotics and/or neural prostheses and other purposes. This year, four different chips were used for different tasks. One of these[8] was an application-specific chip, requiring AER for communicating with the outside world. The other three chips[9,10,12] were more general-purpose integrate-and-fire neurons, where the first two used AER for communication with the outside world, and the second had an output for each neuron.

Here we focus on the last type of device (see Figure 1), the main purpose of which is to carry out several projects for implementing networks of I&F neurons with different (reconfigurable) types of architectures, operating in real time and able to interface with control signals for appropriate motor actuation. The chip, an evolution of a previous version,[11] is made up of ten integrate-and-fire neurons specifically designed to be used in a central pattern generator (CPG) configuration for locomotion of robotic bipeds. Each neuron has 19 synapses, of which eight allow external circuitry or sensory information to modify the output spike trains, ten allow each neuron to feed back to all the neurons on the chip, and one synapse is used to provide a tonic drive for the neurons. Each neuron/synapse pair can

Tenore, continued p. 8

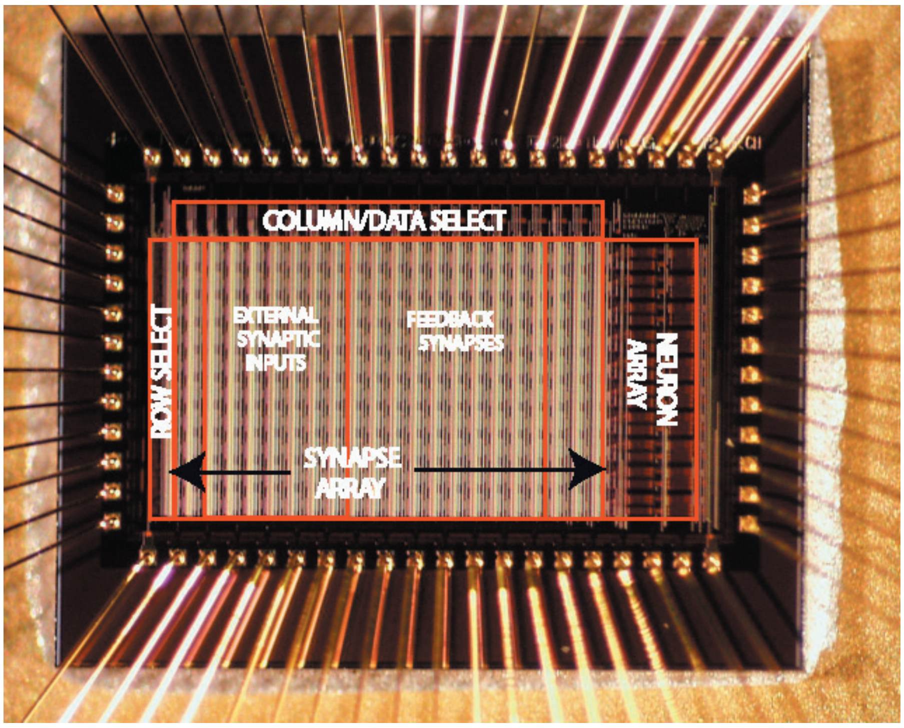

Figure 1. Chip micrograph: the modification of a neuron/synapse/weights triplet occurs through the use of shift-registers.

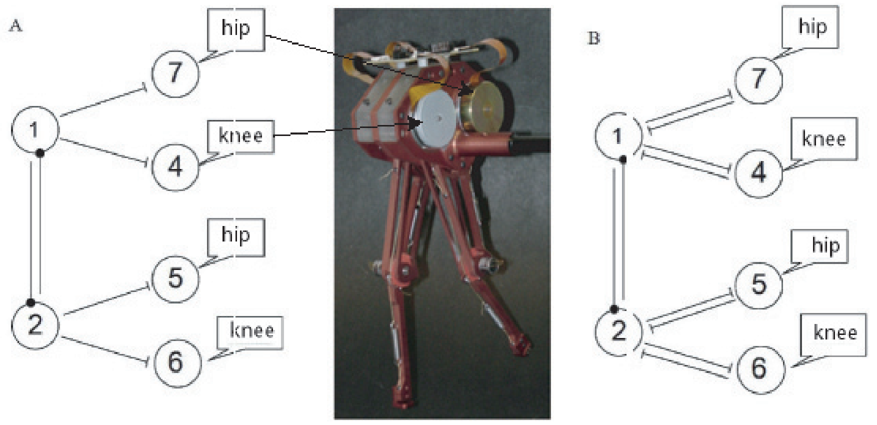

Figure 2: Two of the many possible central-pattern-generator networks implemented on the chip to allow the robot to walk in an open-loop system. The Snappy robot is also shown.

A distributed network for visual processing

The availability of commercial sensor networks has opened up interesting applications for neuromorphic engineers.[1,2] The address-event data-stream representation, originally conceived to allow the exchange of information between processing nodes organized in a network, can now be used to convey information from sensor network processing nodes to multiple receiving nodes. This information can be analyzed and extracted while hopping between network nodes[3] in a way that is very similar to biological neural networks.

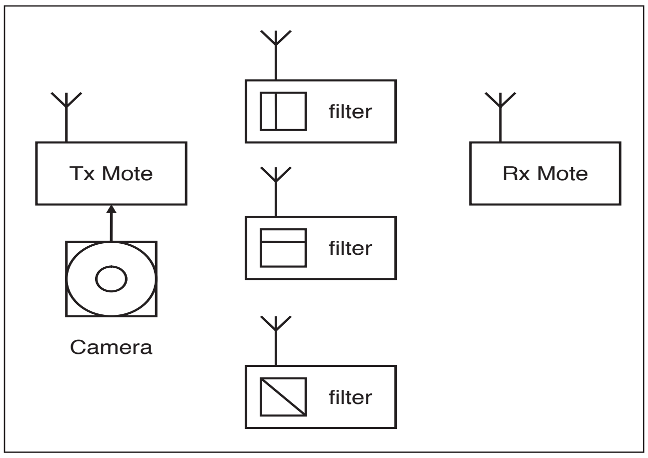

Figure 1. Distributed filter network: a transmitter mote (Tx) is connected to an address-event camera, sending data to the distributed filter motes. Each of these implements one filter template, here at vertical, horizontal and 45° orientation. The filter motes send the filtered version of the image to the receiver mote (Rx) for further processing.

We intend to demonstrate here the capabilities of this kind of sensor network by implementing network-distributed filtering modules to extract features from an address-event image sensor. The distributed filter network implementation is illustrated in Figure 1. A transmitter mote (Tx) is connected to an address-event camera and sends image data to the distributed filter motes, each of which implements one filter template (here with vertical, horizontal, and 45° orientation). The filter motes send the filtered version of the image to the receiver mote (Rx) for further processing.

The transmitter wireless image sensor was developed at Johns Hopkins University using a custom address-event image sensor we designed[4] (the *ALOHA* image sensor) and commercially-available sensor network nodes (see Figure 2).[5] The *ALOHA* sensor is a 32×32 pixel array that converts light intensity into a frequency of events. The address of the pixel is read by the sensor network node and transmitted using a low-bandwidth radio link to a host computer. During the development phase, we demonstrated a wireless imaging capability with limited power consumption: the system operates with two AA batteries and can run for over five days. The image sensor reported good image quality even at very low bit rates and frame latency as low as 1s (see Figure 3).

A filtered version of the original image can be obtained by transmitting a series of pulses representing the pixel intensity through an orientation-hypothesis projection filter. We implemented a proof-of-concept Matlab program to demonstrate the filtering capabilities of the distributed filter network. The MATLAB script *visproc* generates a fictitious image with some geometrical features. Noise is added to the system to evaluate the robustness of the algorithm. The added noise was about 5% of the maximum pixel value. The image is converted in address-event by the MATLAB script: each pixel's intensity is encoded as a sequence of spikes, the frequency of which is proportional to the pixel value in the image.

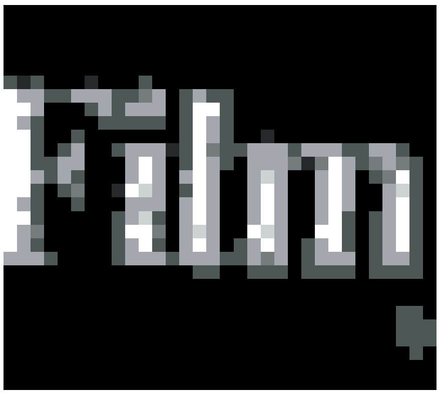

Figure 3. Figure example collected by the wireless image sensor and displayed on a receiving host computer.

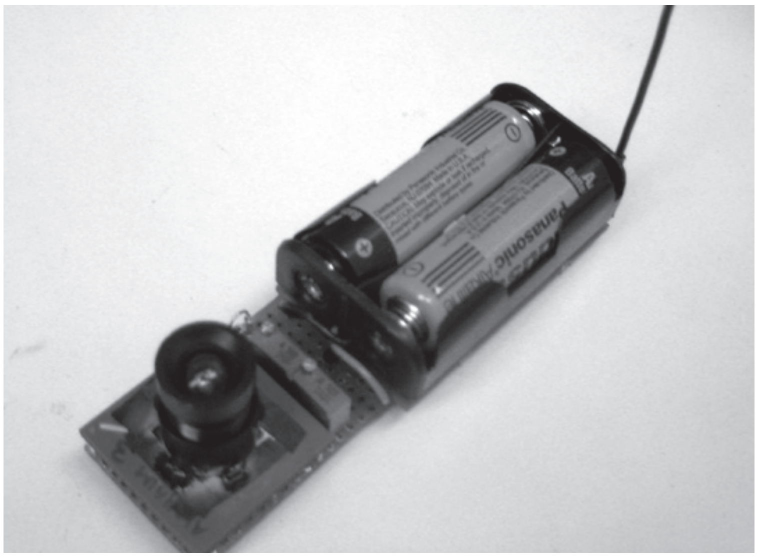

Figure 2. The transmission (Tx) mote connected to the address-event ALOHA camera.

Figure 4 shows the simulation result. The top left image is the original address-event image generated by the MATLAB script. The top-right, middle-left, middle-right and bottom pictures are the result of the vertical-, horizontal-, 135°- and 45°-oriented filters respectively.

The filters are implemented using projective fields: as an event is produced by the address-event image sensor, it is communicated to the filter motes. Once received by each specific mote, it is duplicated multiple times in the shape of the desired template. For example, for a vertically-oriented filter, an address in the (x, y) coordinates will be mapped to three addresses: $((x, y), (x, y-1), (x, y+1))$. The values of the pixel at these addresses will thus be incremented by 1. This technique is called projective field because it makes a prediction about the direction of the image contour. Once a fixed number of events

Culurciello et al., *cont. p. 8*

An electromyography-controlled tail

Ever thought that a prehensile tail would be useful? The potential applications span from enhanced physical agility to social expression to a simple 'helping hand'. A longtime dream of mine has been to construct a robotic tail controlled by signals from electromyography (EMG) using skin-surface electrodes. EMG-based control of prosthetic limbs has a long history and is being developed for future tele-operated robotic actuators for use in space and other hazardous environments. Because EMG measures muscle activation, the signal represents the force that the muscle is exerting on the body and the environment.

At this summer's Telluride Neuromorphic Engineering Workshop, a team of enthusiastic participants decided to attempt a proof-of-concept project, demonstrating a simple two-channel (four-state) tail control. Our project consisted of three components: EMG signal detection, signal processing/command recognition, and the motorized tail. Our team consisted of: Pamela Abshire (University Maryland), Chris Assad (Jet Propulsion Laboratory), Rodrigo Alvarez (University of Pennsylvania), Lena Ting (Georgia Institute of Technology), Nima Mesgarani (University of Maryland), and Massimiliano Giulioni (Italian Institute of Health).

EMG signals measured on the skin surface over an active muscle can be as large as a few millivolts in amplitude, with frequencies mainly between 20-400Hz. They are typically measured with two skin electrodes placed along the length of the muscle and a high-input-impedance differential amplifier. A low-impedance ground electrode is placed on the body surface away from the muscle to control the common-mode voltage for the amplifier. Most clinical (commercial) systems use standardized wet electrodes (Ag/AgCl), each with an adhesive patch to hold it and the conductive gel securely against the skin surface. While we successfully designed and tested our own amplifiers, we ultimately used two commercial amplifiers that had better noise characteristics and could be connected directly onto the electrode patches. Figure 1 shows example signal data from a forearm muscle being rapidly twitched. A gain of about 1000 was used here. In our final version, two sets of electrodes were placed vertically over the lower back muscles (the erector spinae muscle group) about 1.5in (4cm) from the spine.

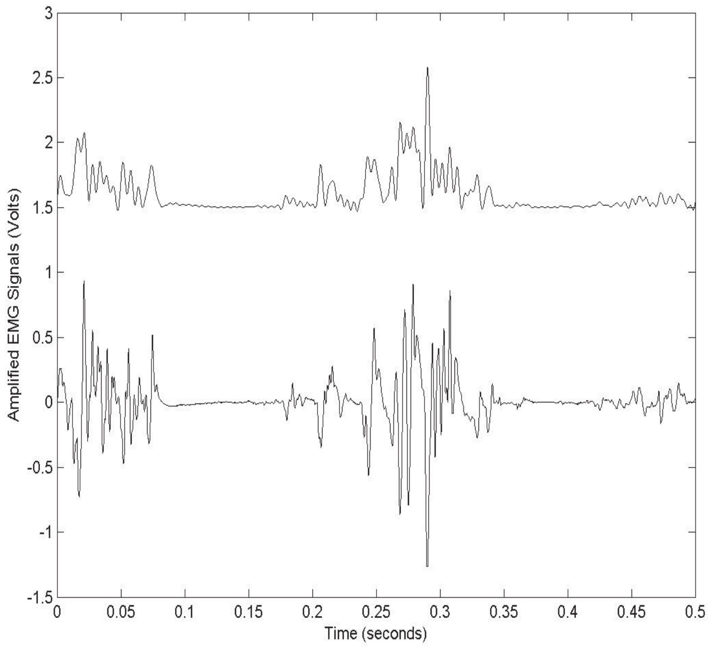

Figure 1. The lower trace shows the amplified raw electromyography (EMG) signal from electrodes placed on the forearm. The upper trace shows the EMG signal after envelope detection and low-pass filtering.

Following analog amplification, we sampled the waveform using a multi-channel Measurement Computing USB (universal serial bus) analog-to-digital converter (ADC) at 500 samples/sec in 100ms blocks. This USB device included digital outputs as well. A laptop computer running MATLAB was used to control data acquisition and provide software control. The waveforms were subsequently rectified, low-pass filtered and compared against a level threshold. We now had two digital channels that indicated when the left, right, or both back muscles were contracting.

To test the feasibility of controlling a robotic device using the EMG signals, we built a prototype robotic tail. Its core consisted of a light, flexible, steel cable that was easily bent yet resisted compression. This cable was fitted with eight circular flanges through which three parallel strings (Spectra Cable 0.030in) were threaded through holes on the edges. Applying tension to one string effectively bends the assembly as the effective length of this string reduces while the length of the steel cable remains constant forming an arc. The three tensional strings were separated by 120° along the circumference of the guide to control the tail's two degrees of freedom. Tension was applied to each string using the shaft of a Solarbotics motor as a winch mechanism. The motors and the steel cable were both mounted to a base plate that served as the base of the tail. The default gear ratio was reduced by removing a gear stage to facilitate

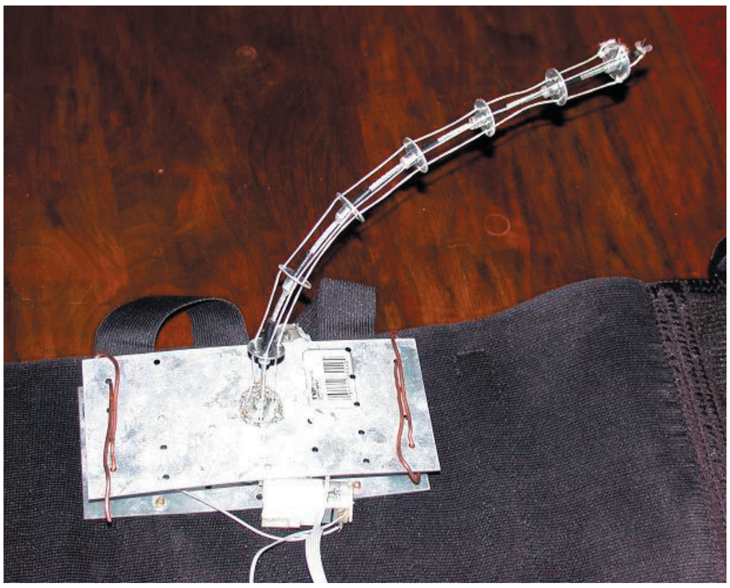

Figure 2. The tail mechanism included three motors in the base, each pulling a string that bent the tail towards its side.

Horiuchi, continued on p. 7

Horiuchi, Continued from p. 6

passive back-driving of the motor.

The three motors were driven by one of three possible EMG commands: a left muscle contraction drove the left motor to pull, a right muscle contraction drove the right motor to pull, and co-contraction of the back muscles drove the third (downward) motor to pull. The three motors were connected in a star configuration with a common node in the center. When one motor was driven in the pulling direction, the current flowed through the motor towards the center of the star and then outward through the two other motors in the releasing direction. The pulling motor bent the tail and the other motors weakly released their strings. This system, while clever, suffered from both weak motor strength and frictional imbalances producing either too much or too little tension.

The final system (see Figure 3) was 'portable', consisting of a laptop, USB-based ADC, batteries, and a tangle of wires. The tail mechanism was mounted on a commercial lower-back support product. The tail would wag naturally while walking, due to the alternating muscle activations with each step. Leaning forward or deliberate stomach-muscle contractions produced back muscle co-contractions that pulled the tail down. Though it worked well, the lack of proprioceptive or visual feedback made tail control somewhat confusing.

This project initiated new ideas about how to use many more electrodes and signal-separation techniques to resolve the signals of muscle subsets that are currently blurred together on a single set of electrodes. The signal-to-noise ratio of the back-muscle measurements were quite high, which allowed a close look at the complex muscle activations that underlie balance and posture in humans.

Quicktime movies of the tail in action,[1] and another interesting EMG-driven tail project,[2] are available online.

Timothy Horiuchi
Department of Electrical Engineering
Institute for Systems Research
Neurosciences and Cognitive Sciences Program
University of Maryland, USA
E-mail: timmer@isr.umd.edu
http://www.isr.umd.edu/~timmer

References
1. Quicktime movie of tail: http://www.isr.umd.edu/Labs/CSSL/horiuchilab/horiuchilab.html
2. Another EMG-driven tail: http://www.wolftronix.com/biotail/biotail.html

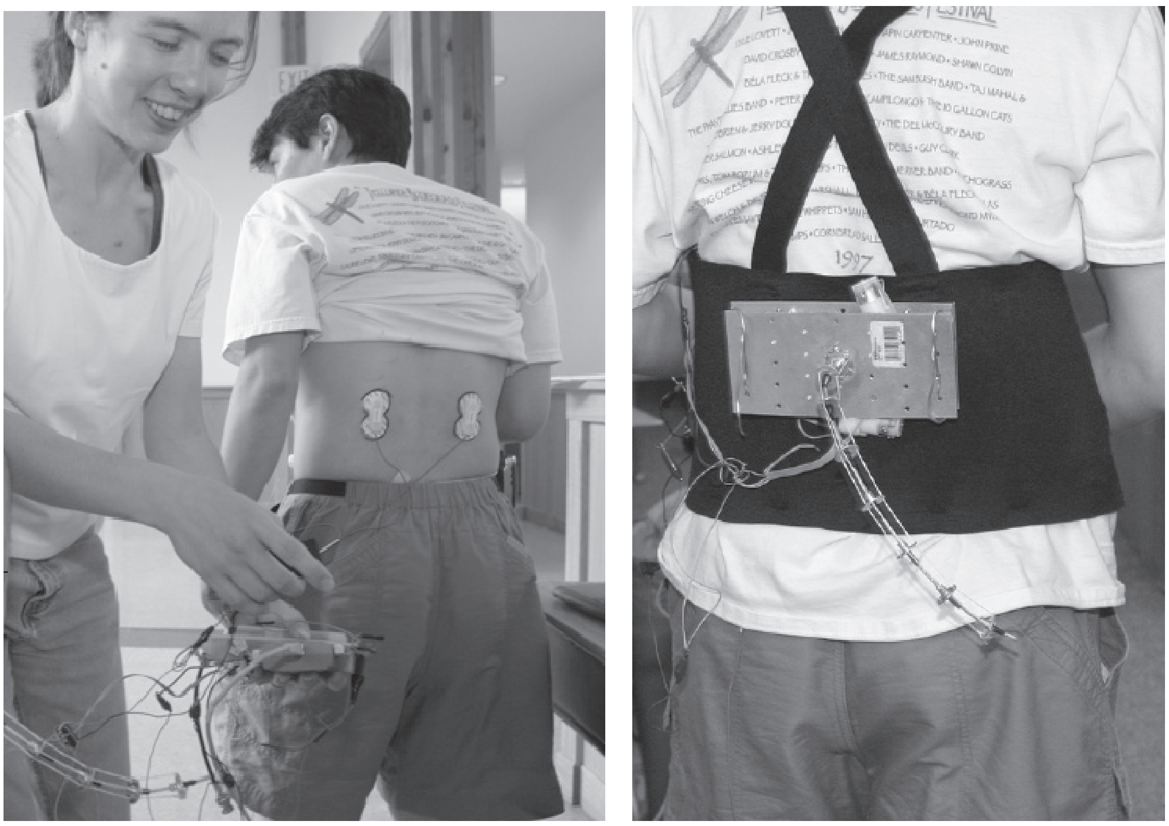

Figure 3. Left: Differential electrode signals from the lower back muscles were locally amplified and digitized by an external ADC. A laptop performed the signal processing and sent the digital commands to the tail motors. Right: The tail mechanism was then mounted on the lower back with wires running to a laptop for control.

Tenore, Continued from p. 4

be set with an 8bit excitatory (or inhibitory) weight that modulates the currents charging (or discharging) the membrane capacitance. The chip can output spikes or pacemaker-type signals depending on the value set on the on-chip pulse width tuner. A refractory period modulator, finally, prevents the membrane capacitance from charging up for an amount of time dictated by the modulator. CPG-type signals can thus be created using the neurons as pacemaker cells or in a spiking configuration.

The key aspect of these signals is that they must be frequency-locked and out-of phase by any arbitrary degree.[12] In a typical human walking motion, for example, the hips are 180° out-of-phase between each other and the knees are 90° out-of-phase with the ipsilateral hip. Different gaits, in turn, require the limbs to have different phase relationships. The chip was used by the locomotion workgroup to create various such waveforms (see Figure 2), and tuned with appropriate weights to create a human-like walking gait for the Snappy robot, developed at Iguana Robotics, Inc.. Each motor on the robot's hips and knees requires direction and speed information to fix the robot's overall gait and velocity. To generate the direction signals for the motors, pacemaker neurons were set up to form the signals as described above, so all the outputs were frequency-locked and out-of-phase by appropriate amounts to achieve a quasi-human walking gait. The signals thus generated allowed the robot to be controlled open loop. Continuing efforts by various group members after the end of the workshop should soon be able to show that all four limbs can be properly controlled and with different gaits.

Francesco Tenore
Department of Electrical & Computer Engineering
Johns Hopkins University, Baltimore, MD
E-mail: ftenore@jhu.edu

Locomotion Workgoup
Project leaders: A.Cohen, R. Etienne-Cummings, and M.A. Lewis
Project participants: R. Alvarez, C. Assad, K. Feller, J. Kutch, A. Horchler, K. Nakada, R. Reeve, J. Scrivens, M. Sekerli, F. Tenore, J. Tian, and L. Ting

References
1. M. Mahowald and R. Douglas, *A silicon neuron*, **Nature 354,** pp. 515-518, 26 November 1991.
2. C. Mead, **Analog VLSI and Neural Systems,** Addison Wesley, 1989.
3. C. Rasche, *An aVLSI basis for dendritic adaptation,* **IEEE Trans. on Circuits and Systems II 48** (6), pp. 600-605, June 2001.
4. M. F. Simoni, G. S. Cymbalyuk, M. E. Sorensen, R. L. Calabrese and S. P. DeWeerth, *A multiconductance silicon neuron with biologically matched dynamics,* **IEEE Trans. on Biomedical Engineering 51** (2), pp. 342-354, February 2004.
5. E. Farquhar and P. Hasler, *A bio-physically inspired silicon neuron,* **Proc. of the Int'l Symp. on Circuits and Systems 1,** pp. 309-312, 23-26 May 2004.
6. R. J. Vogelstein, U. Mallik and G. Cauwenberghs, *Beyond address-event communication: dynamically-reconfigurable spiking neural systems,* **The Neuromorphic Engineer 1** (1), pp. 1, 9, Spring 2004.
7. D. H. Goldberg, G. Cauwenberghs and A. Andreou, *Analog VLSI spiking neural network with address domain probabilistic synapses,* **Proc. of the Int'l Symp. on Circuits and Systems 2,** pp. 241-244, 6-9 May 2001.
8. P. Merolla and K. Boahen, *A Recurrent Model of Orientation Maps with Simple and Complex Cells,* **Advances in Neural Information Processing Systems 16,** pp. 995-1002, 2004.
9. 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
10. E. Chicca, G. Indiveri and R. Douglas, *An event-based VLSI network of integrate-and-fire neurons,* **Proc. of the Int'l Symp. on Circuits and Systems 1,** 23-26 May 2004.
11. M. A. Lewis, R. Etienne-Cummings, A. H. Cohen, M. Hartmann and Z. R. Xu, *An in silico central pattern generator: silicon oscillator, coupling, entrainment, and physical computation,* **Biological Cybernetics 88,** pp. 137-151, 2003.
12. F. Tenore, R.Etienne-Cummings and M.A. Lewis, *Entrainment of silicon Central Pattern Generators for legged locomotory control*, **Advances in Neural Information Processing Systems 16,** pp. 1043-1050, 2004.

Culurciello, Continued from p. 5

has been collected, the pixel values of the filtered image are thresholded to eliminate noise. The result is the desired orientation filtered image, and the Matlab scripts demonstrates that the algorithm functions correctly even in the presence of noise.

In future, we plan to include implementation of the filter kernels in a multitude of motes to perform the distributed filter.

This work was conducted during the recent workshop on Neuromorphic Engineering at Telluride,CO.

Eugenio Culurciello,* Andreas Andreou, and Pablo Sergio Mandolesi†**
*Dept. of Electrical Engineering
Yale University
E-mail: eugenio.culurciello@yale.edu
**Dept. of Electrical and Computer Engineering, Johns Hopkins University
†U. Nacional del Sur, Argentina
E-mail: pmandolesi@uns.edu.ar

References
1. Fermuller, C. Aloimonos, Y. Baker, P. Pless, J. R. Neumann and B. Stuart, *Multi-camera networks: eyes from eyes,* **Proc. IEEE Workshop on Omnidirectional Vision 12,** pp. 11-18, June 2000.
2. P. Doubek, I. Geys, T. Svoboda and L. V. Gool, *Cinematographic Rules Applied to a Camera Network,* **Omnivis,** 2004.
3. I. Akyildiz, W. Su, Y. Sankarasubramancam and E. Cayirci, *A survey on sensor networks,* **IEEE Communications Magazine 42** (5), pp. 102-114, 2002.
4. E. Culurciello and A. G. Andreou, *ALOHA CMOS imager,* **Proc. Int'l Symp. on Circuits and Systems 2004 (ISCAS),** May 2004.
5. Crossbow, **Mica2 mote datasheet,** 6020th ed., Crossbow Technology Inc., San Jose, USA.
http://www.xbow.com

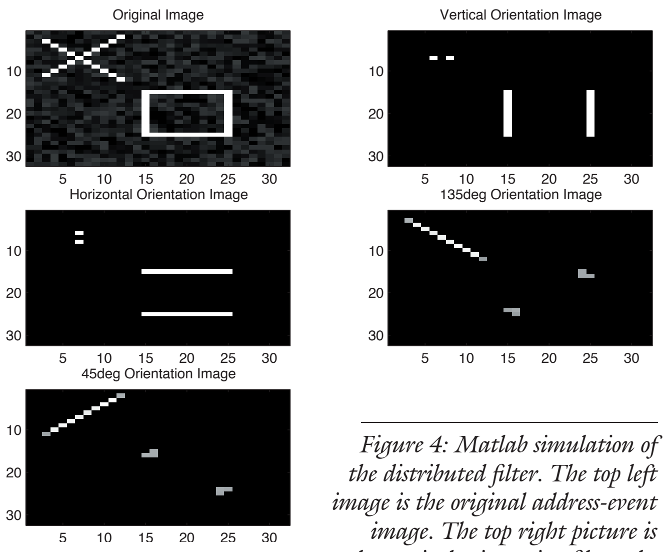

Figure 4: Matlab simulation of the distributed filter. The top left image is the original address-event image. The top right picture is the vertical orientation filter, the middle left is the horizontal filter, the middle right the 135 degrees filter and the last image is the 45 degrees filter.

LABORATORY NOTES

Saliency on a chip: a digital approach with an FPGA

Selective-visual-attention algorithms have been successfully implemented in analog VLSI circuits.[1] However, in addition to the usual issues of analog VLSI—such as the need to fine-tune a large number of biases—these implementations lack the spatial resolution and pre-processing capabilities to be truly useful for image-processing applications. Here we take an alternative approach and implement a neuro-mimetic algorithm for selective visual attention in digital hardware.

The overarching aim of this project is to demonstrate the feasibility of using programmable logic for aiding the development and acceleration of image and video processing applications in vision. Saliency was picked as a design driver for this purpose so that the design flow could be understood and added to the neuromorphic engineer's bag of tricks. The data-intensive and computationally challenging nature of the human visual attention system makes it an interesting algorithm for this study.

Itti, Koch, and Niebur[2] have suggested an influential model for saliency-based bottom-up visual attention and applied it successfully to a variety of visual tasks. The model attempts to represent visual attention in a computationally-efficient manner. The existing software implementation of this model, on a personal computer, runs at 30 frames per second (fps) at quarter-VGA (320×240) resolution.[3] Field-programmable gate arrays (FPGAs), on the other hand, offer an elegant solution to implementing the saliency computation in hardware, taking full advantage of the data parallelism available in the image processing operations.[4] The reprogrammable nature of the FPGAs provides for a quick, cheap platform for prototyping and debugging, and greatly simplifies development.

We wanted to be able to process a video stream at 30fps and at VGA resolution of 640×480 pixels (R, G, and B colors at 8bits/color/pixel), outputting the coordinates of the most salient pixel. Our current design exceeds that specification by processing composite video at 720×525 pixels and 30fps. The hardware was composed using modular elements that implement color-space conversion, Gaussian filtering, interpolation, decimation, and basic image arithmetic transformations, in addition to support for mapping image streams to and from an external memory. The computation of saliency is performed in a series of steps where intermediate 'maps' of the image are created by series of transformations.

The incoming video stream (see Figure 1) is first de-interlaced by a write-read operation through the external synchronous dynamic random-access memory (SDRAM). The de-interlaced image is then separated into its constituent components, which are post-processed to compute red-green and blue-yellow color opponencies, luminance, and orientation-filtered streams. Gaussian pyramids are generated for the incoming streams where each pyramid level is computed by successively low-pass filtering and subsampling the previous level. We have developed an architecture that computes the entire pyramid with a single filter kernel by time multiplexing the different scales/levels of the pyramid and buffering them into a single dual-ported memory using a specialized, mutually exclusive write addressing scheme.

The filtered data is then buffered in the off-chip SDRAM due to the on-chip memory capacity constraints of the FPGA. Feature maps are created next by calculating the difference of chosen pairs of center (finer) and surround (coarser) spatial scales. The latter is first scaled up by stretching it to the finer scale followed by point-wise subtraction of the two streams. Merging the different center-surround scales by decimating all the streams to a specific scale[4] and performing point-wise addition gives us 'conspicuity maps'. These are then normalized with the global maximum of the previous image (as an approximation to the global maximum of the current image, for efficiency). The final saliency map is produced by merging the intensity, color, and orientation streams, and is processed to compute the most salient point.

The Berkeley development board[5]—with its video support, decent-sized Xilinx FPGA and generous amount of SDRAM, see Figure 2—was used as the platform for developing the saliency algorithm. A key contribution of this research was the design of a decentralized memory controller for mapping streams to an external memory and a modular method for plugging in new streams. An area- and memory-efficient mechanism for computing Gaussian pyramids was also demonstrated.

Much work concerning the implementation of the entire design stack needs to be done before more conclusive lessons can be learned, but we hope this project will eventually provide

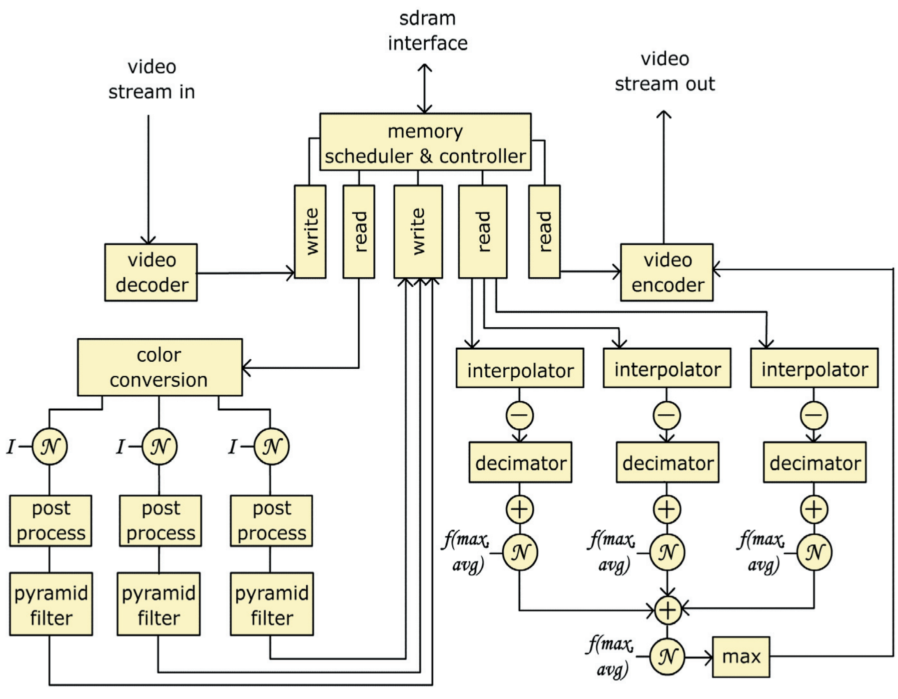

Figure 1: Block diagram of the FPGA saliency implementation.

Kapre, *continued on p. 11*

LABORATORY NOTES

A bias-current-generators toolkit

Analog or mixed-signal CMOS chips usually require a number of fixed reference currents for biasing amplifiers, determining time constants and pulse widths, powering loads for static logic, and so forth. The required currents can span a wide range: if the dynamics of the circuits span timescales from nanoseconds to seconds, for instance. Often, in experimental chips, these references are left out because designers assume that these ‘standard’ circuits could be easily added in a later revision. As a result, chips are designed that must be individually tuned for correct operation with a multitude of external pots to set gate voltages. The required voltages, unfortunately, depend on chip-to-chip variation in threshold voltage, and secondary users must be tutored on the tuning of the parameters. Moreover, the very-small bias currents that are often needed in these chips are problematic to generate externally.

We have developed a toolkit that allows the designer to automatically generate the schematics and layout of a bias generator circuit, using the Tanner EDA tools.[1] The bias generator derives a wide-ranging set of fixed bias currents from a single master, and the desired bias currents are specified in the schematic, using parameter cells. A compiler parses the netlist from the schematic, computes the range of biases, the number of required splitter cells, and the master bias current. It then builds the layout of the complete generator using a set of predefined cells combined with generated routing. The cells are constructed to be used with MOSIS-scalable λ-based design rules, with two metal single-poly processes.

The circuit shown in Figure 1 generates the bias currents, all of which are produced by dividing down a single master current I_m using a current-splitter network. These smaller currents can then be copied or scaled with current mirrors, or passed through a differential pair to provide a variable bias in a particular current range.

The master current I_m is generated by the familiar bootstrapped current reference attributed to Widler and first reported in CMOS by Vittoz et al..[2] Transistors M_{n1} and M_{n2} have a gain ratio $(W_{n1}/L_{n1})/(W_{n2}/L_{n2})=M$. Since the currents in the two branches are forced to be the same by the mirror M_{p1}-M_{p2}, the M_n current density ratios set up a difference in their gate-source voltage, which is expressed across the load R. Resistance R and ratio M determine the current. The master current I_m—which flows in the loop—is computed by equating the currents in the two branches. In subthreshold, this yields the remarkably simple yet accurate formula:

$$I_m = \ln(M)\frac{U_T}{R}, \quad U_T = \frac{kT}{q}$$

With ideal transistors, I_m does not depend on the supply or threshold voltages, but is monotonic in temperature (approximately PTAT, proportional to absolute temperature, in subthreshold). In reality, it is slightly affected by the supply voltage through drain conductance, and also by mismatch of the threshold voltage and β between the transistors in the current mirrors.

Transistor W/L	
M_{n2}	24/6
M_{n1}	M*24/6
M_{p1}, M_{p2}, M_{p3}	76/65
hM_{c1}, M_{c2}	24/6
C_{k1}, C_{k2}	132/20
M	40
M_R, M_{2R}	24/12
M_{pd}, M_{k1}, M_{k2}	6/6
Capacitance	
C_n	~10pF
C_{k1}, C_{k2}	~1pF

The ratio M is not critical as long as it is substantially larger than 1. We have used values from 20 to 120. A very large ratio can destabilize the circuit through the parasitic capacitance C_R on V_R. A common error in this circuit is to have too much capacitance to ground at V_R; this excessive capacitance causes large-signal limit-cycle oscillations. The circuit can be stabilized by making the compensation capacitor C_n several times C_R. In practice, we usually bring out V_n to ensure that the master bias can be stabilized.

A startup circuit is necessary to avoid the stable zero-current operating point. Transistors M_{k1}, M_{k2}, M_{pd}, and MOS capacitors C_{k1} and C_{k2} enable the startup and power control functionality. V_{pd} allows for soft power control and is tied to ground for normal operation. Raising V_{pd} to Vdd turns off the master bias and the derived biases, and returning V_{pd} to ground turns the bias generator back on.

The master current is copied to a Bult and Geelen[3] current splitter and divided successively by it to form a geometrically-spaced series of smaller currents. At each branch, a fixed fraction of the current is split off, while the rest continues to later stages. The last stage is sized to terminate the line as though it were infinitely long. The current splitter principle accurately splits currents over all operating ranges from weak to strong inversion, independent of everything but the effective device geometry. Figure 1 shows an R-2R splitter—built from unit transistors—that splits by octaves. The

van Schaik, cont on p. 11

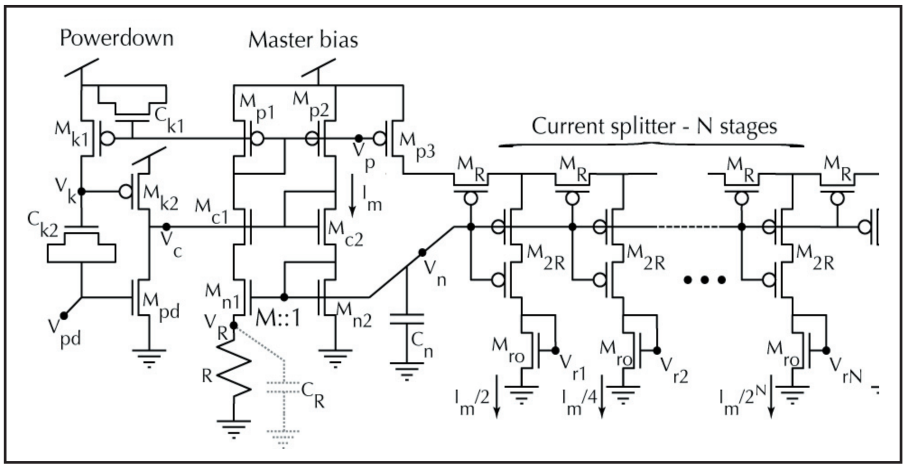

Figure 1: Bias generator circuits. Transistor sizes are in units of λ (scalable parameter). All sizes are 24/6 unless listed above. C_{k1} and C_{k2} are MOS capacitors. M_{2R} are unit transistors.

van Schaik et al., Continued from p. 10

splitter has N stages; the current at the k^{th} stage is $I_m/2^k$.

We have used these bias generators in several generations of CMOS process technology (1.6μm, 0.8μm, and 0.35μm) with no striking differences in performance. Here we show a result from a bias generator with a 20-stage octave splitter built in a 0.35μm process using the design kit. Figure 2 shows the measure output currents of the octave splitter biased with a single generated master bias current of 10 μA. It is amazingly ideal over 20 octaves (six decades) spanning strong to weak inversion. A current of 10pA is reliably generated from a master current of 10μA. For more details of the bias generator, the design kit, and measurements, readers are referred to References 1 and 4.

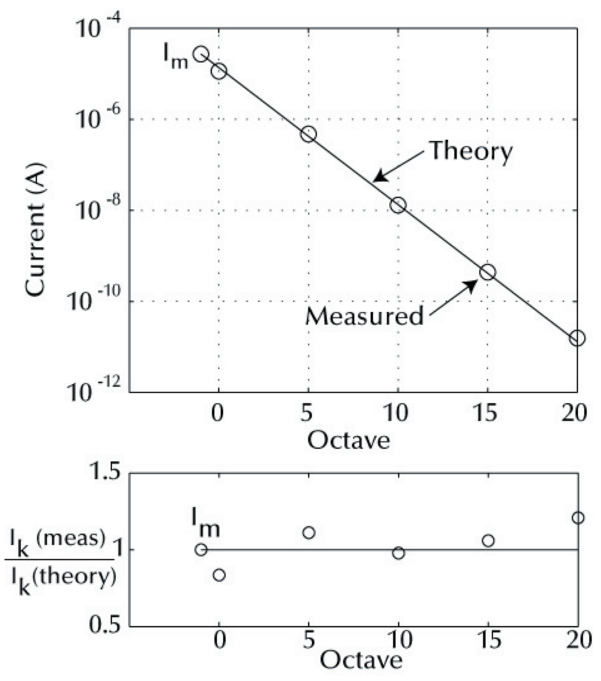

Figure 2: Behavior of the octave splitter.

André van Schaik and Tobi Delbrück*
Univ. of Sydney, Australia
E-mail: andre@ee.usyd.edu.au
*Inst. of Neuroinformatics, ETH/Univ. Zürich Switzerland
E-mail: tobi@ini.phys.ethz.ch

References
1. *www.ini.unizh.ch/~tobi/biasgen*
2. E. Vittoz and J. Fellrath, *CMOS analog integrated circuits based on weak inversion operation,* **IEEE J. of Solid-State Circuits SC-12,** pp. 224-231, 1977.
3. G. Bult and G. Geelen, *An inherently linear and compact MOST-only current division technique,* **IEEE J. of Solid-State Circuits 27,** pp. 1730-1735, 1992.
4. T. Delbrück and A. van Schaik, *Bias Current Generators with wide dynamic range,* **IEEE ISCAS,** 2004.

Calendar

IEEE International Conference on Electronics, Circuits, and Systems—ICECS
13-15 December 2004
Tel-Aviv, Israel
http://www.ee.bgu.ac.il/~icecs04/

Neural Information Processing Systems—NIPS
13-18 December 2004
Vancouver, Canada
http://www.nips.cc/

IEEE International Symposium on Circuits and Systems—ISCAS
23-26 May 2005
Kobe, Japan
http://www.iscas05.org/

International Joint Conference on Neural Networks—IJCNN
31 July – 4 August 2005
Montreal, Canada
http://faculty.uwb.edu/ijcnn05/

Kapre, Continued from p. 9

a reusable image-processing core for other saliency-based applications that might benefit from accelerated saliency computation. The long-term aim is to put a design flow in place that reuses off-the-shelf components, with saliency being the first of a series of basic neuromorphic image-processing operators that can be reused. However, far more needs to be done to tackle the issues of design complexity and development time before widespread use of programmable logic becomes a reality in this potentially-rich application area.

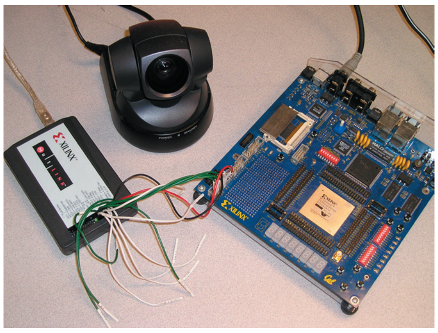

Figure 2: Photo of the Berkeley development board.

Nachiket Kapre, Dirk Walther, Christof Koch, and André DeHon*
California Institute of Technology
E-mail: {nachiket, walther, koch}@caltech.edu
*E-mail: andre@acm.org

References
1. G. Indiveri, *A Neuromorphic VLSI device for implementing 2D selective attention systems,* **IEEE Trans. on Neural Networks,** November, 2001.
2. L. Itti, C. Koch and E. Niebur, *A model of saliency-based visual attention for rapid scene analysis,* **IEEE Trans. on Pattern Analysis and Machine Intelligence,** 1998.
3. L. Itti, *The iLab Neuromorphic Vision C++ Toolkit: Free tools for the next generation of vision algorithms,* **The Neuromorphic Engineer,** 2004.
4. André DeHon. *The Density Advantage of Configurable Computing,* **IEEE Computer,** April 2000.
5. CALINX Development Board, http://calinx.eecs.berkeley.edu.

*If have a toolkit or other goodies to share, contact the Editor at **sunny@sunnybains.com***

LABORATORY NOTES

Tekkotsu: a Sony AIBO application development framework

When the Sony AIBO entertainment robot made its commercial debut in 1999, an initial run of 3,000 were snapped up by Japanese consumers in under 20 minutes. Another 2,000 earmarked for the American market sold out within four days. Since then, Sony has launched several new generations of the robot, improving their capabilities while lowering the price. Over 100,000 have been sold to date.

The AIBO is an attractive platform for robotics research because it combines substantial on-board processing power, a rich sensor array, remote monitoring via wireless Ethernet, and programmability using standard C++. The latest model, the ERS-7, includes a 576MHz RISC processor, 64MB of main memory, a color CCD camera, three infrared range sensors, a three-axis accelerometer/gravity sensor, stereo microphones, audio output, and an array of buttons and light-emitting diodes on the robot's body. The body has 18 degrees of freedom: three in each leg, four in the neck and head, and two in the tail. List price is $1799; academic discounts are available.

The AIBO was not immediately adopted by computer scientists and robotics researchers because, initially, it was not user-programmable. Only a few universities were granted access to AIBO development tools, under nondisclosure agreements, to allow them to compete in the Robosoccer legged robot league that Sony sponsors. But the situation changed in 2002, when Sony made the system development kit (SDK) available for free to the general public. Many researchers remain unaware of this development.

The AIBO SDK offers a documented interface to OPEN-R, Sony's proprietary software architecture that runs on top of Aperios, a real-time operating system. Additional software for vision and locomotion is available from various robosoccer teams who have released their code under the GPL (General Public License GNU).[1] But more tools are needed to simplify the task of programming this complex device.

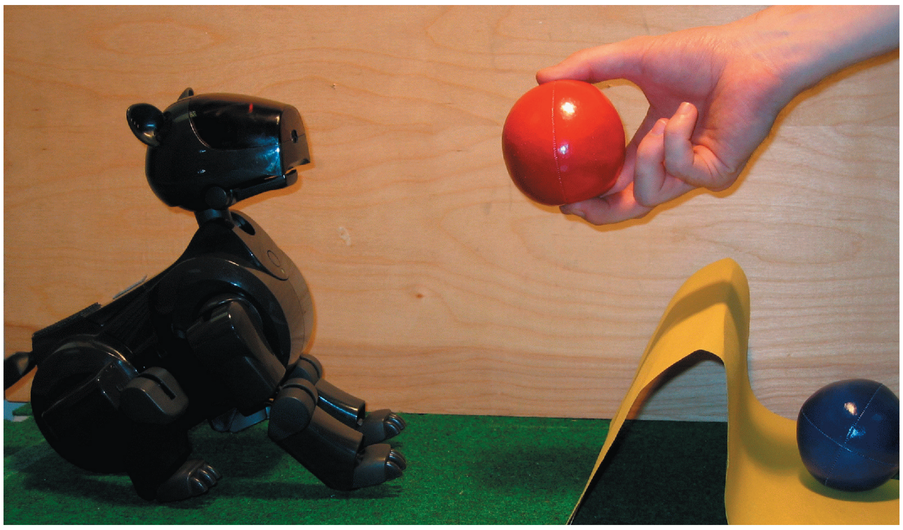
Figure 1: An AIBO being trained on the XOR task, using a colored ball as the stimulus.

Tekkotsu is an object-oriented application development framework for the AIBO that provides an additional layer of abstraction above OPEN-R. (*Tekkotsu* literally means *iron bones* in Japanese, and refers to a metal framework such as the skeleton of a building.) Because it makes extensive use of C++ templates and inheritance, application developers can create their own customized facilities without having to change the Tekkotsu source code. They simply define subclasses that inherit from the Tekkotsu base classes, overriding any member functions that require customization.

A Tekkotsu application is organized as a collection of *Behaviors* and *MotionCommands*. Their member functions run in two cooperating processes, *Main* and *Motion*. *Main* handles perception and decision making, while *Motion* is concerned with realtime control of effectors. A third process, *SoundPlay*, handles audio output, and operates under the same realtime constraints as *Motion*.

Tekkotsu's event router architecture provides a simple, easy-to-learn interface for communication among *Behaviors* and *MotionCommands*. Sensor events, timer expirations, and the completion of *MotionCommands* all generate event notifications to which any *Behavior* can subscribe. The 'vision pipeline', running in *Main*, offers multiple event streams of images at various resolutions and stages of processing, while a lazy evaluation discipline ensures that images are only generated for streams where a *Behavior* is listening. Other important Tekkotsu features include substantial kinematics support, and a collection of remote monitoring and teleoperation tools written in Java that can be run on any personal computer with a wireless connection.

Using Tekkotsu, we have implemented a neuromorphic learning algorithm on the AIBO: a combination of configural and temporal difference (TD) learning[2] that allows the robot to learn appropriately-timed responses in an exclusive-OR task (see Figure 1.)

We are currently developing an even-higher-level set of primitives intended to support a new course in Cognitive Robotics that will be offered at Carnegie Mellon in 2006. These primitives draw inspiration from ideas in cognitive science, such as 'visual routines' (Ullman), 'dual coding representations' (Paivio), and 'affordances' for object manipulation (Gibson). Our intent is to demonstrate a new style of robot programming at a higher level of abstraction than is common in robotics today.

Tekkotsu is open source, licensed to the public under the LGPL (Lesser General Public License GNU),[1] and available for free at the Tekkotsu.org web site.[3] The initial development of Tekkotsu was funded by a grant from the Sony Corporation.

David S. Touretzky and
Ethan J. Tira-Thompson*
Computer Science Department
Carnegie Mellon University, USA
E-mail: dst@cs.cmu.edu
*E-mail: ejt+@andrew.cmu.edu

References
1. http://www.gnu.org/licenses/licenses.html
2. D. S. Touretzky, N. D. Daw, and E. J. Tira-Thompson, *Combining configural and TD learning on a robo,* **Proc. of the Second Int'l Conf. on Development and Learning,** pp. 47-52, 2002.
3. http://www.Tekkotsu.org