Low Power Sparse Approximation on Reconfigurable Analog Hardware

Samuel Shapero *Student Member, IEEE**, Adam Charles *Student Member, IEEE*, Christopher Rozell *Senior Member, IEEE*, and Paul Hasler *Senior Member, IEEE*

Abstract— **Compressed sensing is an important application in signal and image processing which requires solving non-linear optimization problems. A Hopfield-Network-like analog system is proposed as a solution, using the Locally Competitive Algorithm (LCA) to solve an overcomplete ℓ_1 sparse approximation problem. A scalable system architecture using sub-threshold currents is described, including vector matrix multipliers (VMMs) and a nonlinear thresholder. A 4x6 nonlinear system is implemented on the RASP 2.9v chip, a Field Programmable Analog Array with directly programmable floating gate elements, allowing highly accurate VMMs. The circuit successfully reproduced the outputs of a digital optimization program, converging to within 4.8% RMS, and an objective value only 1.3% higher on average. The active circuit consumed 29 μA of current at 2.4 V, and converges on solutions in 240 μs. A smaller 2x3 system is also implemented. Extrapolating the scaling trends to a N=1000 node system, the Analog LCA compares favorably with State-of-the-Art digital solutions, using a small fraction of the power to arrive at solutions ten times faster. Finally we provide simulations of large scale systems to show the behavior of the system scaled to non-trivial problem sizes.**

I. Introduction

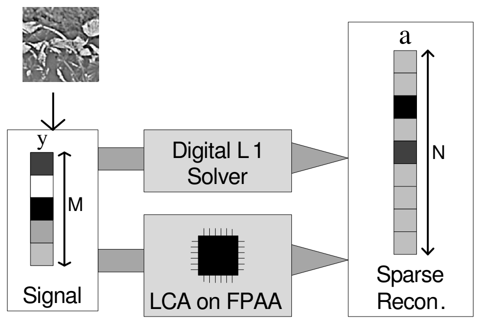
SPARSE approximation is an optimization program that seeks to represent a vector (i.e., signal) by using just a few elements of a prescribed dictionary (Fig. 1(a)). Modern signal processing has seen increasing movement toward using tools based on nonlinear optimizations rather than linear filtering because these approaches correspond to inference in statistically rich (i.e., non-Gaussian) signal models. In particular, sparse approximation is a fundamental component in state-of-the-art approaches for many application areas, including inverse problems (e.g., denoising, restoration, and data recovery from undersampled measurements [3]), computer vision and machine learning [4].

To highlight one specific example, consider the emerging literature on Compressed Sensing (CS) [5], [6]. In brief, the CS results give performance guarantees for inverse problems when the signals are highly undersampled ($M \ll N$ where M measurements are taken of a length N signal) and the signal is assumed to be sparse (having only $S \ll N$ nonzeros). The main CS results essentially show that for certain sensing matrices Φ (generally taken to be random), S-sparse signals can be recovered (up to the noise level) by solving an ℓ_1 regularized least-squares optimization problem as long as $M \sim O\left(S \log(N/S)\right)$. These results mean that in situations where measurements are costly, a signal can be undersampled during acquisition in exchange for using more computational resources to recover the signal at a later time.

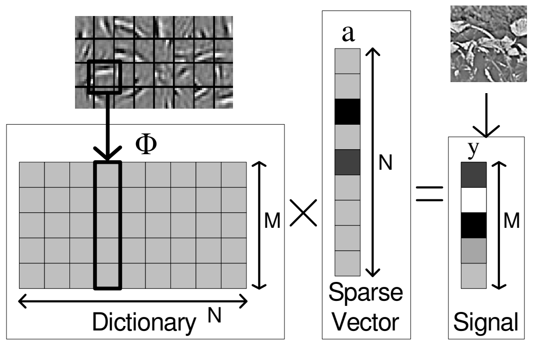
This insight is leading to the design of new coded aperture sensing systems that spend fewer resources to collect data at a specified resolution, relying instead on computational post-processing to reconstruct the signal. Unfortunately, the optimization problems used for signal recovery are computationally expensive, preventing practical deployment of digital

Manuscript received February 20, 2012. This work was supported in part by NSF grant CCF-0905346.

* Corresponding author. The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0250 USA (e-mail: {sshapero3,acharles6,crozell,phasler}@gatech.edu). Preliminary versions of portions of this work were presented in [1], [2]. The authors are grateful to Bruno Olshausen, Justin Romberg, and Pierre Garrigues for valuable discussions related to this work, and to Marijn Brummer, Emory University and Children's Healthcare of Atlanta for providing MRI data.



(a) Sparse Approximation Solutions



(b) Generative Model

Fig. 1. (a) The LCA implemented on the FPAA is capable of performing the same sparse encodings as a digital solver, but at a fraction of the power and speed. (b) Sparse encodings assume a linear generative model, where a signal is the sum of a sparse set of dictionary elements. For example, [7] showed that natural images can be constructed with a sparse set of wavelets.

solutions for portable, low-power applications. This paper demonstrates a system implementation for solving a widely used sparse approximation problem via sub-threshold current mode circuits on a Field Programmable Analog Array (FPAA).

A. Impact of an Analog Implementation

Despite the long history of optimization in the field of signal processing (see Mattingley & Boyd [8] for a detailed discussion), the recent advent of applications that utilize optimization directly to perform CS highlights a specific need for solvers that can operate in real time or under power constraints. For example, CS techniques have been proposed for both medical imaging [9] and channel estimation for wireless communications [10], that may respectively benefit from real-time or low-power systems for sparse approximation.

Given the importance of solving sparse approximation problems in state-of-the-art algorithms, recent research has focused on dramatically reducing their solution times. These optimization programs are particularly challenging due to the presence of the ℓ_1-norm in the objective, making the program non-smooth. Despite much recent progress in developing both general and special purpose convex optimization solvers, this non-smoothness provides particular challenges for obtaining real-time results for moderate-sized problems.

Recent work in computational neuroscience has demonstrated a continuous-time dynamical system where the steady-state response is the solution to a regularized least-squares optimization and the architecture of the system is designed to efficiently deal with sparsity-inducing non-smoothness conditions [11]. The Hopfield Neural-Network-like architecture of this system makes it amenable to analog circuit implementation, which promises several benefits.

In particular, even the most efficient current iterative digital algorithms require $O(N^2)$ floating point operations per iteration while the solution time in a parallel analog architecture is proportional to the RC time constant (which scales $O(N)$) [12]. Total energy consumption is also reduced by using analog vector matrix multipliers VMMs) that require only one transistor per multiplication, instead of the large multipliers required for digital processing. Using a programmable analog device like the FPAA allows the implementation and testing of circuits without the time-consuming process of industrial fabrication [13], and allows compensation for errors caused by the inherent mismatch in transistor sizes.

In total, a successful analog approach may provide solutions with lower power, greater speed, and better scaling properties than is possible in digital solutions. The implementation of such a system significantly impacts many practical applications that will simply be out of reach even with substantial future improvements in digital algorithms due to either time or power constraints. An analog system could be especially powerful when coupled with the CS techniques mentioned above, allowing signals to be acquired (with coded apertures) *and recovered* at very fast time scales, potentially eliminating the post-processing that has become the accepted bottleneck with CS systems.

B. Optimization Problem Formulation

Sparse approximation methods achieve efficient signal representations using only a small subset of dictionary elements by taking advantage of the known statistical structure of the signal [14]. These methods assume a linear generative model (Fig. 1(b)) for signal representation:

$$y = \Phi a + \nu \tag{1}$$

where a vector input $y \in \mathbb{R}^M$ is represented with an overcomplete dictionary $\Phi = [\phi_1, \dots, \phi_N]$ using coefficients $a \in \mathbb{R}^N$, with additive Gaussian white noise ν. We wish to find the Maximum A-Posteriori (MAP) estimate of the linear generative model, assuming a sparse prior on coefficients a:

$$\begin{array}{l} \arg\max_a P(a|y) = \arg\max_a P(y|a)P(a)\,, \\ P(a) \propto \prod_j e^{-C(a_j)} \end{array} \tag{2}$$

where $C(\cdot)$ is a sparsity-inducing cost function (e.g., the ℓ_0-Norm). Unfortunately, direct optimization of this problem - where $C(\cdot)$ counts non-zeros - is intractable. In Basis Pursuit De-Noising (BPDN) [15], one of the most common surrogates of ℓ_0-Norm optimization, $C(\cdot)$ is set to the ℓ_1-Norm, making (2) equivalent to the convex optimization:

$$\arg\min_a \left(\frac{1}{2} \|y - \Phi a\|_2^2 + \lambda \|a\|_1 \right). \tag{3}$$

The first term in the objective function represents the mean squared error of the approximation, the second term represents the sparsity of the solution via the ℓ_1-Norm, $\|a\|_1 = \sum_i |a_i|$, and λ is a tradeoff parameter balancing data fidelity with the solution sparsity.

II. Description of the LCA Architecture

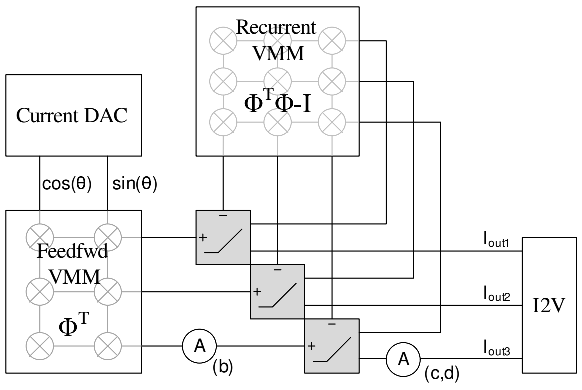
The LCA is described by a system of nonlinear ordinary differential equations (ODEs). These equations translate readily into a Hopfield-Network-like system architecture.

A. System of Differential Equations

The LCA is a continuous time algorithm which acts on a set of internal state variables, $u_m(t)$ for $m = 1, \dots, M$. These internal states are guaranteed to exponentially converge to the equilibrium state, which is the solution to the objective function (3) [11], [12]. Restricting $a(t) > 0$, the dynamics of the nodes are described by the following set of ODEs:

$$\begin{array}{c} \tau \dot{u}(t) + u(t) = b - (\Phi^t \Phi - I)\; a(t), \\ a(t) = T_\lambda(u(t)) = \max(0, u(t) - \lambda) \end{array} \quad . \tag{4}$$

In (4), τ is the time constant of the system, and $b \in \mathbb{R}^M = \Phi^t y$ is the vector of driving inputs. The feedback between the nodes is computed by $H = \Phi^t \Phi - I$. The sparsity constraint and the nonlinearity are introduced by the threshold operator $T_\lambda(\cdot)$, which decreases the absolute value of $u(t)$ by λ. Once the state variables $u(t)$ have reached equilibrium, the output vector $a(t)$ is the solution to the objective function.



(a) Block diagram of 2x3 LCA

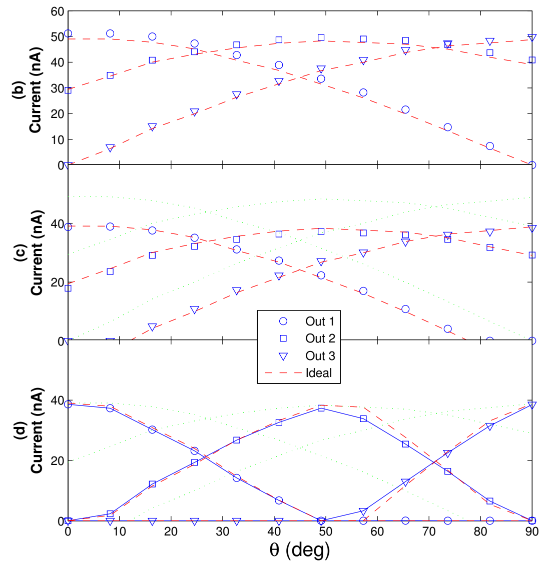


Fig. 2. (a) Diagram of the small LCA system, with ammeters indicating the currents represented in plots (b)-(d). Inputs are produced by an onchip current DAC, while outputs are converted to voltages which are then projected offchip. (b) The output of the feedforward VMMs, when the two inputs are swept along the unit curve, as $\cos(\theta) \cdot 50\,\text{nA}$ and $\sin(\theta) \cdot 50\,\text{nA}$. Compared against an ideal multipliers. (c) Output of the LCA without any recurrent inhibition, compared with the ideal. (d) Output of the full LCA system, compared to a digital solver. The difference here is caused by a slight error in the recurrent VMM, magnified when two nodes have strong lateral inhibition. Note that the competition sharpens the response curves.

B. System Architecture for Hardware

As in most neural networks, the internal state variables in (4) evolve in a parallel fashion. The architecture of the LCA implemented as an Analog Hardware System is presented in Fig. 2(a). The system is composed of current mode VMMs and current mirrors (including a double current mirror that implements the soft-threshold operation).

The first VMM is the feedforward multiplier. It accepts the input vector y from the Current DACS (after they are mirrored) and performs the operation $b = \Phi^t y$ to compute the driving inputs. The second block, the recurrent VMM, performs the operation $h(t) = Ha(t)$ and computes the recurrent feedback. The feedback is similar to a stable, convergent Hopfield Network [18]; nodes do no inhibit themselves ($H_{m,m} = 0$) and the inhibition between nodes are symmetric ($H_{m,n} = H_{n,m}$).

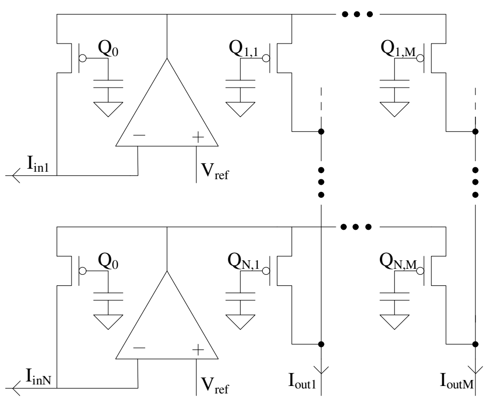


Fig. 3. Implementation of a VMM using floating-gate transistors [16]. Each input is log-compressed to produce a voltage, which is broadcast to the transistors in the row. Each transistor than generates a current, which is a scalar multiple of the original: $w_{(n,m)} = e^{-(Q_{(n,m)}-Q_0)/U_T}$, where Q is the charge programmed into a floating gate, and Q_0 is the charge programmed on all the input floating gates. The performance of this structure on the RASP 2.9v was characterized in [17].

Both of these VMMs are implemented as current mode devices (Fig. 3), which have a small area, low power, and an easily scalable design while operating in the sub-threshold region [16]. The VMMs perform the linear operation $I_{OUT} = WI_{IN}$. The charge on each FGE determines the weight of each scalar multiplication.

Scalar multiplication accuracy requires the input and output devices to have matching drain voltages. The input drain voltage is regulated with an OTA that provides a power source to both the input and output currents; the OTA must therefore scale in power with the number and strength of the outputs.

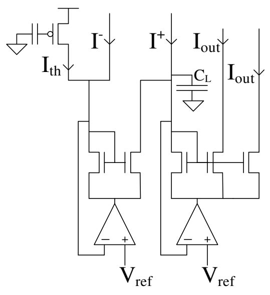
The last system component is a double nFET current mirror (Fig. 4), which finds the difference of the linear terms $b-(h+\lambda)$, and applies a capacitive load to induce a low pass filter with time constant. The active current mirrors, based on [19], each accept a current into an nFET. The circuit forces another nFET to have the same gate, and source voltages, thus assuring that it will produce the same current. Since the input nFET also acts a rectifying diode, the current mirror can only pass positive currents. Introducing the negative offset λ makes the device an effective soft-thresholder (Fig. 4(b)).

Accuracy of the current mirror requires the nFETs to be well matched, and to have identical drain voltages. Mismatch is minimized simply by enlarging the devices; this enlargement is not a major impediment to system density, since there are $O(N)$ mirrors, while the VMM area scales $O(N^2)$. As with the VMMs, OTAs are used to regulate the input drain voltage. Since the mirror outputs are the VMM inputs (which also have a regulated voltage), this allows matching of both drain voltages. The current mirror OTAs likewise allow matching of the drain voltages in the VMM.

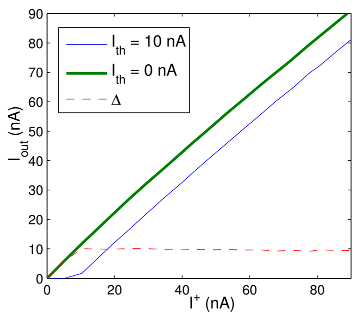
The transfer function of the double current mirror is then:

$$\begin{aligned}\tau\dot{u}(t) + u(t) &= b - h(t) \\ a(t) &= T_\lambda(u(t))\end{aligned} \quad . \tag{5}$$

From the VMMs, we get $b = \Phi^T y$ and $h = (\Phi^T\Phi - I)a(t)$. Combining these relationships yields our original ODE (4).



(a) Schematic



(b) Results

Fig. 4. Analog Thresholder (a) Implementation of the Soft-Threshold with a single sided output. The floating gate generates the threshold current I_{th}, while current mirrors invert and rectify their inputs. The full operation performed is thus $I_{out} = \max\left(0, I^{+} - (I^{-} + I_{\lambda})\right)$. Additional output branches of the current mirrors allow the current to be read. A large loading capacitor C_L adds a large, dominating time constant. The OTAs eliminate drain voltage mismatch by pinning the input voltages to V_{ref}. $V_{ref} = 1.2$ V, $V_{dd} = 2.4$ V. (b) Response of the Soft-Thresholder. With $I_{th} = 0$, the thresholding function is a rectifier, but when I_{th} is increased to 10 nA, it effectively creates a soft threshold at I_{th}.

III. Implementing the LCA circuit on Reconfigurable Analog Hardware

The experimental results presented in this paper were obtained on the Reconfigurable Analog Signal Processor (RASP) 2.9v (Fig. 5), a 350 nm double-poly CMOS chip designed in our lab [17]. The chip includes several Computational Analog Blocks (CABs), a large matrix of programmable floating gate elements (FGEs) that can be used for routing, and 26 chip spanning volatile switch lines that allow rapid scanning of every internal node in the chip. Most of the CABs contain a variety of analog elements, including the operational transconductance amplifiers (OTAs) and nFETs used in the LCA. There are also 18 CABs dedicated for current-mode Digital to Analog Conversion (DACs), which allow system inputs to be quickly reprogrammed.

The RASP 2.9v includes several design innovations that make it particularly well suited for implementing and testing the LCA. The majority of the FGEs are directly programmed devices, meaning that the programmed device is directly in the final circuit. While this adds a selection register to the signal path, it eliminates mismatch issues seen in earlier FPAAs [20]. The direct devices allow the programming of current sources (needed for the threshold current and inputs) to 7 bits of accuracy (less than 1% error).

An automated calibration routine from [20] uses the Enz-Krummenacher-Vittoz (EKV) [21] model to find the exact relationship between the floating gate programming targets and the multiplier weight. On the RASP 2.9v, it allows the programming of current-mode VMMs to 6 bits of accuracy.

We controlled and communicated with the RASP 2.9v using a USB connection to a AT91sam7s Microcontroller. The microcontroller also communicates with onboard ADCs and DACs, which allowed us to set and read analog voltages on the FPAA. We interfaced with the Microcontroller with a suite of Mathworks MATLAB© commands. See [22] for greater detail on the Hardware and Software infrastructure. Scripts written

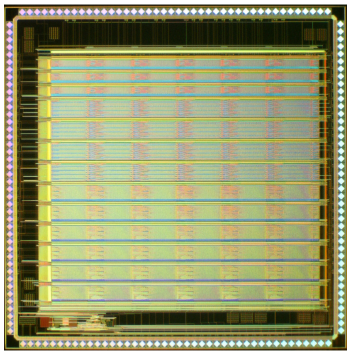


Fig. 5. A die photo of the RASP 2.9v, an FPAA chip on which the hardware implementation of the LCA was programmed. The RASP 2.9v is 5 mm x 5 mm and fabricated in 350 nm process

in MATLAB© also allow the programming and testing to be automated. Our lab has developed a whole chain of tools for the RASP chips, allowing the user to convert an entire library of functions into circuits [23], and then to place and route these circuits on the RASP 2.9v [24].

We implemented multiple LCA systems on a FPAA, the RASP 2.9v [17]. The smaller of these was a single-ended 2x3 system (two inputs, three outputs), built for illustrative purposes; since the input vector had to lie on the unit circle, the input in practice had only one degree of freedom, making the results easier to display. Its dictionary was:

$$\Phi = \begin{bmatrix} 1 & .6 & 0 \\ 0 & .8 & 1 \end{bmatrix}.$$

We also implemented a larger single-ended 4x6 system in order to demonstrate the scalability of the system architecture:

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & .47 & .59 \\ 0 & 1 & 0 & 0 & .59 & .47 \\ 0 & 0 & 1 & 0 & .65 & .1 \\ 0 & 0 & 0 & 1 & .1 & .65 \end{bmatrix}.$$

The six dictionary elements were chosen to fully span the input domain and to observe the Restricted Isometry Property (RIP), where the eigenvalues of the matrix are restricted to a certain range. While a matrix of random Gaussian variables is typically used to satisfy the RIP [5], the dimensions were small enough here that a set matrix could do so more easily.

In addition to the necessary VMMs and current mirrors, onchip 8-bit current DACs were programmed to allow control of the input currents. These inputs were normalized to a ratio of 60 nA:1. The threshold current I_{λ} was programmed to $6\,nA$, for a tradeoff parameter of $\lambda = 0.1$.

Each soft threshold node was implemented with multiple output transistors. One of these was used to drive the rest of the circuit. A second device was used as a system output. The output currents were scanned out by a volatile switch lines, and then sent to either to an onchip current-to-voltage (I2V) converter (Fig. 8(b)) for rapid measurement, or a picoammeter (used for debugging the circuit and calibrating the I2V).

IV. Results of the Fully Implemented System

Using the dynamical switches, we were able to quickly test individual components and the system as a whole. We used

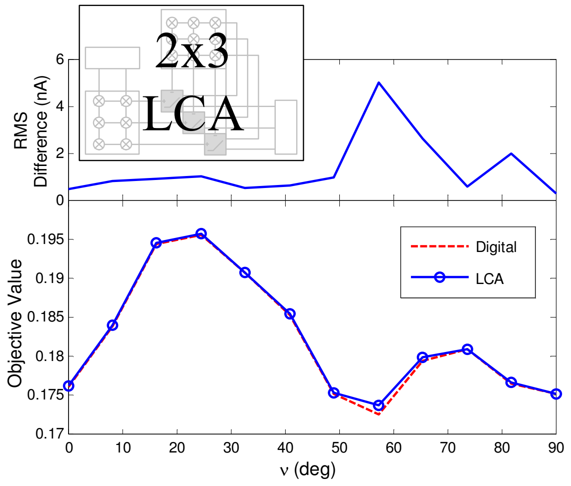


Fig. 6. Metrics of the 2x3 hardware LCA over the input domain $I_{in1} = cos(\theta) \cdot 50\,\text{nA}$, $I_{in2} = sin(\theta) \cdot 50\,\text{nA}$, where $\theta = [0, \pi/2]$. (Top) RMS difference in the outputs between the hardware LCA and the digital solver l1-ls. Scaling for the analog to digital comparison is 50 nA:1. (Bottom) Comparison of the optimized objective function (3) from the analog and digital solvers. Even at the point where the analog and digital solutions diverge by almost 10% of the signal output, the analog objective value is less than 1% higher than the digital.

the onchip current DACs to inject currents with a constant ℓ^2-Norm into the circuit. For the 2x3 network, we swept the input on the unit circle, while we randomly generated 100 inputs for the 4x6 network. For both systems, we separately measured the input currents, the outputs of the feedforward VMMs (with and without thresholding), the outputs of the recurrent VMMs, and the system outputs. Fig. 2 illustrates the progression of these results for the smaller network.

A. *Accuracy of Results*

In order to test the accuracy of the analog LCA, we also ran the inputs through l1-ls [25], a digital sparse approximation algorithm. For both the 2x3 and 4x6 systems, the solution produced by the hardware network was very similar to that produced by the digital solver. For the smaller network (Fig. 6), the root-mean-square (RMS) difference of the analog and digital solutions was at maximum 5.1 nA, and averaged less than 1 nA, or less than 2% of the magnitude of the input. The larger network showed higher divergence, with a max RMS difference of 9.2 nA, or 15.3%, and an average RMS 2.9 nA, or 4.8%.

Despite some deviation from the digital solution, the large network converged on a moderately optimized sparse code. The final value of the objective function averaged only 1.3% higher for the 4x6 network than for the l1-ls solution, and in the worst case was only 3.2% higher. Most of the increase in the objective function in the analog solution came from the MSE term, which averaged 4.6% higher than in the digital case. The average ℓ_1-Norm was virtually identical for both analog and digital solutions.

The support vector of the analog system (the list of active nodes) was identical to that of the digital solution in 63 of 100 trials, and never differed by more than one node. Matching the support set is an important achievement, since the optimal sparse approximation solution can be fully recovered if the correct support set is identified.

B. *Analysis of Sources of Error*

To diagnose the sources of the discrepancies between the analog and digital solutions, each point in the circuit was compared against a digital ideal. We individually tested each scalar multiplication in the VMMs by serially inputing the vectors of the identity matrix into each VMM. This was easily accomplished for the feedforward VMM, since its inputs are directly controlled by the current. We measured $B = \Phi^T Y$, setting $Y = I$.

In order to verify the scalar weights in the recurrent VMM, we had to force the output to be a vector of the identity matrix. We measured $Z = HA$, setting $Y = \Phi$ which gives $A = I(1 - \lambda)$.

We compared the achieved multiplications with the target weights for both matrices, and found that the RMS error of multiplication was 1.9% of the target. This corresponds to a precision of 5.7 bits, which matches previous recordings on the same chip [17]. This error can be reduced by using a programming algorithm with finer precision, at the cost of longer programming times.

We then analyzed the network in order to calculate the expected effect of the multiplication errors on the final output. In steady state $\dot{u}(t) = 0$, and for active nodes $a > 0$ (active coefficients are in the active set Γ), the LCA reduces to:

$$\begin{aligned} u &= \Phi_\Gamma^T y - Ha \\ a &= u - \lambda \end{aligned} \quad , \tag{6}$$

where $H = \Phi_\Gamma^T \Phi_\Gamma - I$, and all vector and matrix terms with subscript Γ are restricted to the subset or subspace corresponding to the active nodes.

We can solve this system for equations for a, yielding:

$$a = (\Phi_\Gamma^T \Phi_\Gamma)^{-1} (\Phi_\Gamma^T y - \lambda) \quad . \tag{7}$$

Introducing feedforward gain error ξ_Φ and recurrent gain error ξ_H terms into the LCA modifies the solution to $\widetilde{a} = (\Phi_\Gamma^T \Phi_\Gamma + \xi_H)^{-1} \left((\Phi_\Gamma^T + \xi_\Phi) y - \lambda \right)$. For small error, the term $(\Phi_\Gamma^T \Phi_\Gamma + \xi_H)^{-1}$ can be approximated as the more tractable $(\Phi^T \Phi)^{-1} - (\Phi^T \Phi)^{-1} \xi_H (\Phi^T \Phi)^{-1}$. Using the identity from (7), we can approximate the output of the LCA with error to be:

$$\widetilde{a} \approx a - (\Phi_\Gamma^T \Phi_\Gamma)^{-1} \xi_H a + (\Phi_\Gamma^T \Phi_\Gamma)^{-1} \xi_\Phi y \quad . \tag{8}$$

Both major error terms will be multiplied by $(\Phi_\Gamma^T \Phi_\Gamma)^{-1}$. Given eigenvalue decomposition $\Phi_\Gamma^T \Phi_\Gamma = V \Lambda V^T$, then if all the eigenvalues are nonzero, $(\Phi_\Gamma^T \Phi_\Gamma)^{-1} = V \Lambda^{-1} V^T$.

The eigenvalues of the inverse matrix bound the amplification of any multiplication errors; the inverse of the smallest eigenvalue of $(\Phi_\Gamma^T \Phi_\Gamma)^{-1}$ is the upper bound of error amplification. This amplification is clearly seen in the small 2x3 network. In the narrow region where the two dictionary elements $[0, 1]$ and $[.6, .8]$ are both active,

$$\Phi_\Gamma^T \Phi_\Gamma = \begin{bmatrix} 1 & .8 \\ .8 & 1 \end{bmatrix} \quad ,$$

which has eigenvalues of 1.8 and 0.2. In this region, therefore, certain errors are multiplied by 5, which causes a noticeable deviation from the digital solution (seen in Fig. 2).

The larger network has a much larger set of possible subspaces of active dictionary elements, which gives a much larger range of amplification. When, for example, the first, second, third and fifth nodes are active, the upper bound for amplification is 200. In 100 random trials, however, this state was never observed; the typical upper bound on amplification was closer to 6. In future implementations we could use these large deviations from digital solutions to estimate the sources of the error from (8). An iterative programming and testing algorithm would be able to gradually eliminate the most prevalent sources of error.

Dictionaries that limit the range of their eigenvalues are considered to observe the Restricted Isometry Property (RIP). Enforcing the RIP proved difficult for a 4x6 matrix (especially with the other constraints placed upon it), but is easier for larger matrices. Candés et al. discuss a number of ways in which these matrices may be generated [5]. In CS applications, randomly populating the dictionary creates a matrix Φ that observes the RIP with high probability. For instance random Gaussian matrices will satisfy the RIP condition when $M > KS\log(N/S)$ and randomly sampled discrete Fourier matrices will satisfy the RIP when $M > KS\log^4(N)$ for some constant K. In these cases, we note that M is significantly larger than S, which when the elements of a matrix are random can readily ensure that the eigenvalues of $\Phi_\Gamma^T\Phi_\Gamma$ will be sufficiently large for any subset of columns Γ. This indicates that the errors amplified in the small scale implementations will not occur for large CS matrices.For example, in the simulations in Section V-B (when M/S = 0.05), we never observed an eigenvalue less than 0.6, corresponding to error amplification of 1.66. We would therefore expect an analog implementation at that scale to have errors 3 times smaller than those in 4x6 system.

The RIP will not necessarily be observed in other applications, but the average eigenvalues should not change with the size of the system. We can therefore predict that the average error should not increase with matrix size.

C. Power and Scaling

The power used by the RASP 2.9v implementation of the LCA is dominated by two terms. First is overhead: 703 µA used by the FPAA even when nothing is programmed, and an additional 20 µA for the high speed current-to-voltage converter.

The rest of the current flow can be accounted for with the OTAs, since every source to sink path in the LCA passes through at least one OTA. The OTAs are differential pairs with a double current mirror, so they naturally use twice their bias current, separate from any sourcing or sinking any current. Since every signal in the LCA chip sinks into an OTA, we can simply sum all the active currents in the chip to find the total additional power use.

Each VMM input requires an OTA. The current mirrors for the inputs also require an OTA, and they sink twice the input

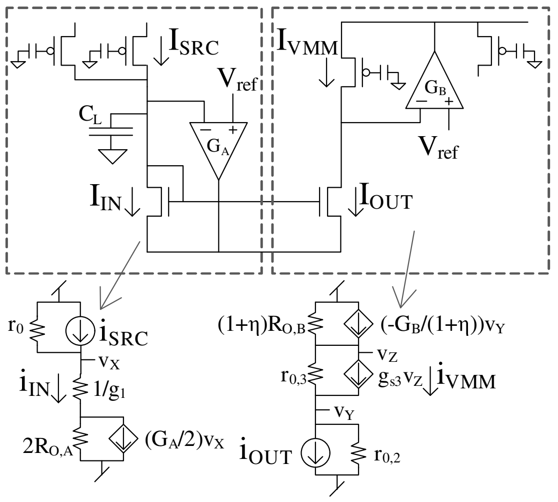


Fig. 7. Small signal model of the current mirror and VMM used to determine OTA biasing.

current. The soft thresholder requires two OTAs, and sinks twice the lateral inhibition Ha, twice the threshold current λ, and twice the output a. The total current used by the system is therefore:

$$I_{TOT} = (M+N)(2I_V) + (M+2N)(2I_M) \qquad (9)$$
$$+2||y||_1 + 2||Ha||_1 + 2N\lambda + 2||a||_1 \quad ,$$

where I_V is the bias current of the VMM OTAs, and I_M is the bias current of the mirror OTAs.

In both of the networks we built, I_M was set to 500 nA, sufficient to sink three 60 nA currents (the third being used only when the node is directly measured) while maintaining a high OTA transconductance. I_V was set to 500 nA in the 2x3 network, and to 800 nA in the 4x6 network.

Excluding overhead, the active circuits of the 2x3 LCA had a total current of 11.8 µA, with small variations from the signals being passed. This is actually less than the 13 µA that would be expected from (9)). The total power use of the 4x6 system was only 31.1 µA, again somewhat less than the 32 µ predicted by (9). These discrepancies are most likely due small innacuracies of the bias current programming.

The OTAs must have a bias current large enough to sink or source all the appropriate currents while maintaining a high transconductance, as indicated in the following analysis.

1) Determining OTA Biasing via Circuit Analysis: The bias currents of the OTAs (and thus the power budget of the chip) are set so as to maintain signal independent gain across the nFET current mirrors. A small signal model of the current mirror and its interface with the VMMs are illustrated in Fig. 7. There are two sources of non unity gain: a conductive divider at the input, which prevents all of the source current i_{SRC} from entering the mirror as i_{IN}; and a conductive divider at the output of the mirror, which prevents the output current i_{OUT} from fully entering the VMM as i_{VMM}.

The first conductive divider is split between the output conductance of the VMM, $g_0 = 1/r_0$, and the effective input conductance of the mirror g_{IN}. Since the OTA creates an amplifier on the other side of the input resistance, we can use the Miller Effect to calculate the effective conductance:

$$g_{IN} = \frac{1 + G_A R_{0,A}}{1/g_1 + 2R_{0,A}} \quad ,$$

where g_1 is the transconductance of the input nFET, and g_A and r_A are the transconductance and the output resistance of the OTA. The total conductive divider for the input is therefore:

$$\frac{i_{IN}}{i_{SRC}} \approx \frac{G_A R_{0,A}}{(g_0/g_1 + 2g_0 R_{0,A}) + (G_A R_{0,A})} \quad . \tag{10}$$

From [20], the output conductance of the floating gates used in the VMM is dominated by transconductance generated by the capacitive coupling from the drain to the floating gate. In subthreshold operation, we can therefore model $g_0 = \beta \frac{I_{SRC}}{U_T}$ where $\beta \approx .04$ is the coupling coefficient, I_{SRC} is the drain current of the floating gate, and $U_T \approx 26\,\mathrm{mV}$ is a constant. Since $g_1 = \frac{I_{SRC}}{U_T}$, we can substitute $g_0/g_1 = \beta$. We can rewrite (10) as a function of currents:

$$\frac{i_{IN}}{i_{SRC}} = \frac{R_{0,A} I_A/U_T}{(\beta + 2\beta R_{0,A} I_{SRC}/U_T) + (R_{0,A} I_A/U_T)}$$

$$\approx \frac{I_A}{2\beta I_{SRC} + I_A} \quad , \tag{11}$$

where I_A is the bias current of the amplifier. To maintain unity gain, with an error less than 1% we must maintain $I_A > 8I_{SRC}$. Note that the required bias current is independent of the number of nodes.

A similar analysis can be done at the VMM input:

$$\frac{i_{VMM}}{i_{OUT}} \approx \frac{G_B}{(1+\eta)1/r_{0,2} + G_B}$$

$$\approx \frac{I_B}{(1+\eta)\sigma I_{OUT} + I_B} \quad , \tag{12}$$

where I_B is the bias current of the VMM OTA, η is the sum of all the weights being generated by the OTA, and $\sigma = \kappa U_T/V_A \approx .0125$, is a constant for these devices. In order to maintain error less than 1%, we must maintain $I_B > 1.25(1+\eta)I_{OUT}$.

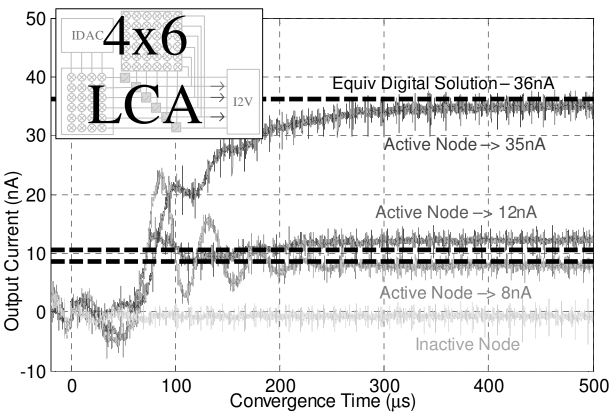
The bias current of the VMM OTAs scale with η, the total weight being generated by that row of the VMM. As the number of nodes N grows, the VMM will have to source that many currents. The average weight of the multipliers will tend to decrease as the dictionary elements spread out through the M dimensional input space. η should therefore approximately scale as $N/\sqrt{M}$. For very large N, the $M+N$ VMM OTAs will dominate the total power use of the system:

$$I_{TOT} \propto (M+N)(2\frac{N}{\sqrt{M}}). \tag{13}$$

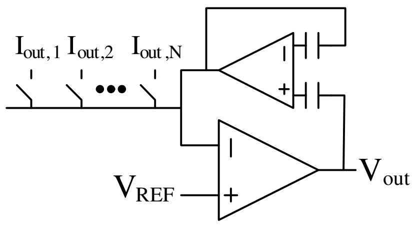
For a fixed M/N, this yields total power scaling of $O(N\sqrt{N})$.

D. Temporal Evolution of the System

Fig. 8 shows the evolution of the 4x6 LCA for a typical input. The temporal evolution of the analog LCA was measured by sending the current-mode outputs through a fast current-to-voltage converter, which was then sent to a high speed oscilloscope. Each relevant node was measured in this way, and their time courses following the setting of the current DACs are superimposed in . The outputs settled to within 1 nA RMS of their final values in 240 µs.



(a) Current output over time



(b) Current to Voltage Converter

Fig. 8. (a) Evolution of the output nodes on the 4x6 LCA for a typical input case, converging to within 1 nA RMS of the final value in under 240 µs. The output of the I2V is sent to an oscilloscope, which begins measuring at time 0 when the current DACs are fully loaded. Nodes have slow dynamics at 0 nA, accounting for long ramp up time. Dashed lines represent equivalent digital solution for each of the three active nodes. (b) The current to voltage converter (I2V), as implemented on the LCA the RASP 2.9v. Currents from the nodes are input serially. A wide range amplifier (a component of the RASP 2.9v CAB) is used in feedback, providing an effective transimpedance. The currents can be independently measured by a picoammeter, allowing characterization of the I2V and accurate estimation of the output currents.

The convergence curves varied considerably from predicted LCA dynamics. Theoretical analysis [12] and simulations (V-B) of the LCA's temporal evolution show exponential convergence for active nodes, in less than 10τ. The theoretical upper bound on convergence time was proportional to τ/γ, where τ is the RC time constant, and γ is the smallest eigenvalue of the active subspace of the matrix Φ (the same term that determines error amplification in Sec. IV-B).

We do not observe this purely exponential convergence in Fig. 8. Instead we see a delayed start and decaying oscillations that eventually converge on a solution. The slow ramp time results from the dynamics of the current mirror circuit used in the thresholder, which is not a simple RC filter when the current is low.

The input resistance R can be derived from the small signal model (Fig. 7) as:

$$R = r_0 || \frac{1/g_1 + 2R_{0,A}}{1 + G_A R_{0,A}} \approx \sigma \frac{U_T}{\kappa I_{IN}} + 2\frac{U_T}{\kappa I_A} \tag{14}$$

For small I_{IN} , the first term dominates, and the system

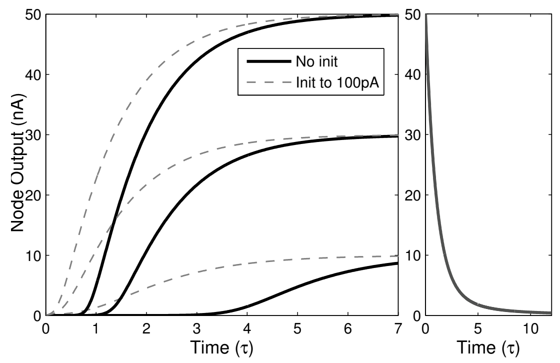


Fig. 9. Dynamics of the thresholder circuit. (Left) If initialized to a near zero current, the nodes have a slow ramp time. Time is measured in terms of the time constant τ, which characterizes the dynamics for currents above 3 nA. Initializing the nodes at 100 pA would require adding switching circuitry to the thresholder, but would reduce the long ramp up. (Right) Nodes should not be initialized too far above zero, as signals take a long time to decay.

dynamics approach:

$$\frac{C_L U_T \sigma}{\kappa} \frac{\cdot I_{IN}}{I_{IN}} + I_{IN} = I_{SRC} \quad . \tag{15}$$

These dynamics can be observed in Fig. 9. By initializing system inputs to zero, we guarantee that all nodes will also start there. This initialization prevents the slow decay that would be required if a signal changed from 50 nA to 0 nA. The dynamics still impose a long startup latency while the input node voltage is charged. This latency could be mitigated by initializing the nodes to a higher value (100 pA in Fig. 9).

For $I_{IN} > I_A\sigma/2 \approx 3$ nA, the second term in (14) dominates, and the system acts as a low pass filter with RC time constant $\tau = 2C_L U_T/(\kappa I_A)$. In order to make this the dominant pole in the LCA system, the load capacitance C_L was made extremely large—over 50 pF—by shorting it to a chip pad. The capacitance could be reduced to 2-3 pF, the capacitance of a vertical routing wire, at the cost of deviating somewhat from the LCA dynamics. This could speed up convergence times by a factor of 10 or more.

In addition to the approximately 240 µs required for convergence, each 8-bit input DACs takes 5.8 µs to load, and reading an output node requires 520 ns, adding about 26µs for interfacing. These costs are imposed by the microcontroller, and are not inherent to the RASP 2.9v.

As the system scales, we would expect the convergence time to scale with the time constant $\tau = 2C_L U_T/(\kappa I_A)$. Of these constants, only the load capacitance C_L will increase with scale at roughly $O(N)$. But since C_L is already much larger than necessary, a custom built large N implementation would actually be expected to converge faster.

V. Simulated CS Recovery

The system described in Section III provides evidence that the LCA can be implemented in an analog hardware system with reasonable accuracy. While these results are encouraging, the current size of this implementation makes it impossible to evaluate its performance on CS recovery applications directly. In this section, we demonstrate the possible performance of the LCA on large-scale CS recovery problems by simulating the ideal dynamical system (described in equations (4)), illustrating that the potential benefits justify continued efforts to scale up the current implementation. Specifically, in the first set of simulations (Sections V-A and V-B), we use synthetic stylized data to thoroughly explore the solution quality and solution times with (simulated) analog and digital approaches for $N = 1000$. In the second set of simulations (Section V-C), we use very high dimensional MRI data to show performance on a large-scale problem of practical importance.

A. LCA solution quality

To begin, we investigate the quality of simulated LCA solutions on CS recovery problems with synthetic data to verify that they are comparable to standard digital algorithms. While the LCA system is proven to converge asymptotically to the unique BPDN solution, the approximate solution achieved by any algorithm in finite time can have different characteristics depending on the particular solution path. In the general problem setup, the unknown signal $a_0 \in \mathbb{R}^N$ is S-sparse and is observed through $M < N$ Gaussian random projections, $y = \Phi a_0 + \nu$, where ν is additive Gaussian noise. We compare the simulated performance of the LCA at recovering a_0 BPDN against the interior-point method l1-ls [25] and the gradient projection method GPSR [26]. This enquiry will address two main questions. First, are solutions produced by the simulated LCA as accurate as the digital comparison cases? Second, what solution times are possible in the simulated LCA?

We draw the nonzero coefficients of a_0 using a Gaussian distribution with variance 1 and we draw the locations from a uniform distribution. The choice of regularization parameter λ depends on the variance of the additive noise ν which is not necessarily known a priori. We have empirically observed that $\lambda = .01\|\Phi^T y\|_\infty$ gives good performance in this task when the noise variance is 10^{-4}. Additionally, we observe that as with many other algorithms, implementing a continuation method by gradually decreasing λ (similar to that used in FPC [27]) also improves convergence time in the LCA. Specifically, we initialize $\lambda = \|\Phi^T y\|_\infty$ and allow a multiplicative decay of 0.9 at each iteration of the simulation until λ reaches the desired value given above. Although the implementation of the current hardware only supports a constant threshold value over time, inclusion of a decaying threshold is possible by having temporally changing threshold currents I_{th} at the threshold units. To ensure that the comparison among the algorithms is fair, we use the same stopping criterion for convergence based on the duality gap upper bound proposed in [25].

To explore solution quality we display the results of solving the CS recovery optimizations using plots inspired by the phase plots described by Donoho & Tanner [28]. We parameterize the plots using the indeterminacy of the system indexed by $\delta = M/N$, and the sparsity of the system with respect to the number of measurements indexed by $\rho = S/M$. We vary δ and ρ in the range $[.1, .9]$ using a 50 by 50 grid. For a given value (δ, ρ) on the grid, we sample 10 different signals using the corresponding (M, N, S) and recover the signal using BPDN. We compare the results of the simulations by displaying in

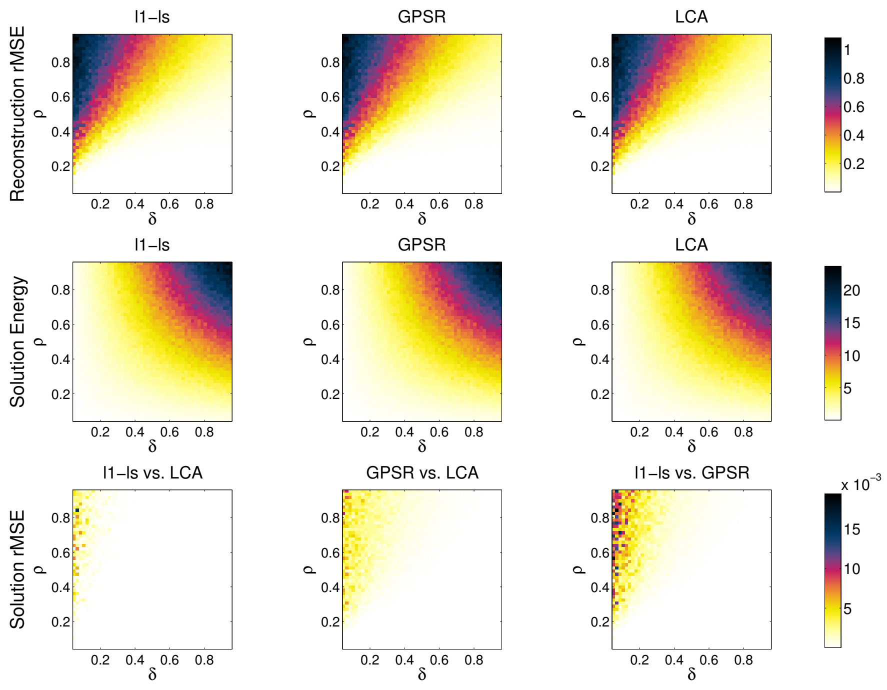


Fig. 10. The solution quality of the simulated LCA on a compressed sensing recovery task is comparable to the standard digital solvers GPSR and l1-ls. The top row plots the relative MSE of the estimated signal for synthetic data, with indeterminacy of the system indexed by $\delta = M/N$, and the sparsity of the system with respect to the number of measurements indexed by $\rho = S/M$. The middle row plots the value of the BPDN objective function at the solutions. The bottom row plots the relative MSE in the solutions between the solvers, indicating the the differences in the LCA solutions are within the normal range of differences between the digital algorithms themselves. Note that all solvers demonstrate more variability in regions where the problems are more difficult and signal recovery cannot be performed well.

the top row of Fig. 10 a phase plot for each algorithm, where the color code depicts the average relative MSE of the CS recovery for each algorithm (calculated by $\|\widehat{a} - a_0\|_2^2 \,/\, \|a_0\|_2^2$). In a similar vein, the middle row of Fig. 10 shows the energy function (i.e., the BPDN objective function) evaluated at the solution, $0.5\,\|y - \mathbf{\Phi}\widehat{a}\|_2^2 + \lambda\,\|\widehat{a}\|_1$.

The near identical plots for the two metrics above demonstrate that the LCA is indeed finding solutions of essentially the same quality as the comparison digital algorithms, both in terms of signal recovery of the compressively sensed signal, and in terms of the optimization objective function. When the LCA and digital solutions are compared directly, we find that the average difference in the solutions differs only by a relative mean-squared distance (calculated by $\|\widehat{a}_{LCA} - \widehat{a}_{DIG}\|_2^2 \,/\, \|\widehat{a}_{DIG}\|_2^2$) of $1.97 \cdot 10^{-4}$ when compared to l1-ls and $6.64 \cdot 10^{-4}$ when compared to GPSR. For comparison, the rMSE of the difference between the l1-ls solutions and the GPSR solutions is $9.71 \cdot 10^{-4}$, meaning that the LCA solutions have variability comparable to what the pair of comparison digital algorithms has between their solutions. We note that the solution differences are significantly larger between all of the algorithms in the regimes where CS recovery is difficult and poor solutions are found by all solvers, as demonstrated by the bottom row of plots in Fig. 10.

B. LCA convergence time

To observe the potential solution times for the LCA in large-scale CS problems, we compare the convergence of the LCA and GPSR on three specific signals in easy, medium and hard CS recovery problems with the same synthetic data as above (corresponding to different values of δ, ρ). Figure 11 shows the convergence of the relative MSE as a function of time for GPSR and the simulated LCA for three example signals. GPSR times are reported using measured CPU[1] time,

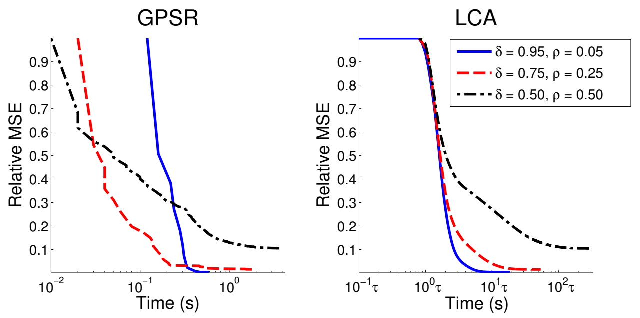


Fig. 11. Temporal convergence of the simulated LCA compared to GPSR. The plot shows the relative MSE of the signal recovery as a function of time for sample trials (N=1000) from the results in Fig. 12 using GPSR (left) the simulated LCA (right). The convergence behavior is approximately the same, with harder problems taking both algorithms longer and decreasing the fidelity of the recovery. For the easy and medium difficulty problems where BPDN recovers the signal with good fidelity, GPSR takes 0.1-1 seconds to converge and the simulated LCA takes $10^1\tau$-$10^3\tau$ seconds to converge. For reasonable values of τ, the LCA solution times can still be as low as 10µs, supporting datarates of up to 100 kHz

[1]Time is measured on a Dell Precision Desktop with dual Intel Xeon E5420 Processors and 14GB of DDR3 RAM.

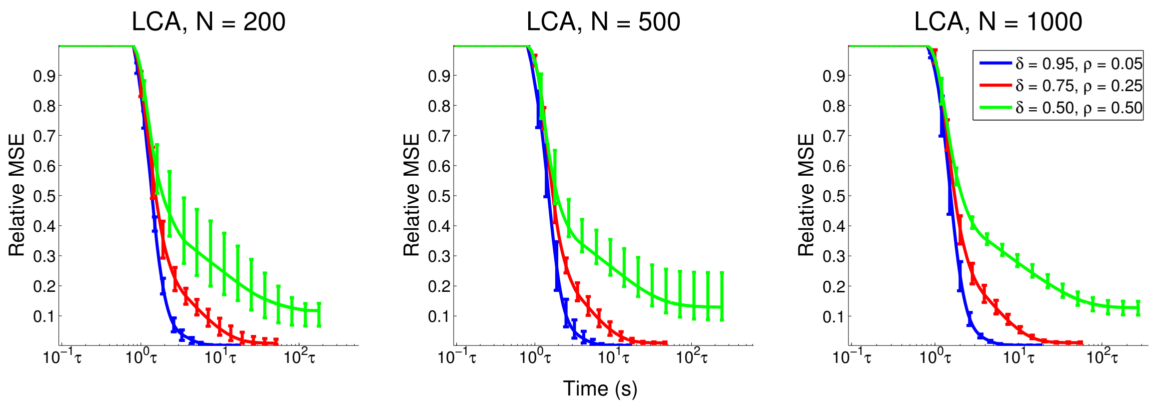


Fig. 12. Convergence behavior for the simulated LCA for a number of different problem sizes (N,δ,ρ). Each plot demonstrates the change in convergence based on easy, medium and hard CS recovery problems (i.e., 3 combinations of (δ, ρ)) for N = 200 (left), N = 500 (middle) and N = 1000 (right). While there is no appreciable increase in convergence time with increased problem size (larger N), similar to standard behavior with other optimization algorithms the LCA convergence time does increase with problem difficulty (smaller δ and larger ρ).

and LCA times are reported using the number of simulated system time constants τ. The simulation parameters used are identical to the previous simulations. While the solution paths have generally similar characteristics, the time scales are dramatically different. Focusing on the easy and medium CS problems that produce good recovery using ℓ_1 minimization, GPSR is converging in times on the order of 0.3 seconds, whereas the LCA is converging in times on the order of ten time constants (10τ). These simulated times are consistant with the reported times for the hardware implementation described in section IV. We also note that while the results in Fig. 11 are for individual signals for direct comparison with GPSR, the analysis of average case convergence for the LCA shown in Fig. 12 and discussed below also support the same basic conclusions about the LCA convergence time.

Though the time constant of an analog circuit depends on many factors (including the bias current and resulting power consumption of the circuit), $\tau = 1$ µs is a reasonable projected value for a dedicated implementation based on the discussion in section IV-D and previous reports [16]. Under this assumption, the simulated LCA is converging for CS recovery problems in approximately 10µs of simulated time. Even state of the art digital solutions using high performance computing (either multi-core processing [29] or graphical processing units [30]) currently only achieve speeds in the tens of milliseconds for comparably sized problems. This type of solution speed from the LCA is several orders of magnitude faster than GPSR and could support solvers running in real time at rates of 100 kHz.

Finally, we also investigate the effect of problem size N and problem difficulty (δ, ρ) on the convergence speed of the LCA. For the same parameters corresponding to easy, medium and difficult CS recovery problems as used above, we sample 10 signals at three different problems sizes (N = 200, N = 500 and N = 1000) to perform CS recovery. Figure 12 displays the relative distance of the signal estimate $a^{(t)}$ from the true solution a as a function of simulated time, $\|a^{(t)} - a\|_2/\|a\|_2$. The plots are again shown as a function of the simulated time in terms of the number of system time constants τ. As expected, convergence is faster and more reliable (i.e., less variance) for easier recovery problems (i.e., lower sparsity or more measurements). Interestingly, we note that increasing the signal size N does *not* appear to increase the number of time constants required for the LCA solution. In a digital algorithm such as GPSR, while the number of iterations may not increase substantially, the solution time scales with N^2 because the cost of each iteration (e.g., a matrix multiplication) increases significantly. In an analog system like the LCA, increasing the size of a matrix multiply requires increasing the circuit size and complexity, which may increase the time constant as discussed in section IV-D.

C. MRI Reconstruction

The previous subsection demonstrated that for stylized problems with synthetic data the LCA can achieve BPDN solutions and signal recoveries comparable to standard digital solvers. Furthermore the LCA appears to converge to solutions at speeds that would represent an improvement of several orders of magnitude over digital algorithms. In this section we demonstrate the potential value of this system on a medical imaging application that could be significantly impacted by having real-time CS recovery techniques. Specifically, in this section we simulate the LCA recovery of undersampled MR images to evaluate the solution quality and speed. Compressive MRI is of particular interest because it allows shorter scan times, which improves both patient throughput and lowers risk (e.g., shorter scan times mean that pediatric MRIs may be taken more often without general anesthesia [9]). Furthermore, compressive MR imaging combined with real-time image reconstruction would potentially allow new medical procedures to be performed using real-time 3-D imaging without using ionizing radiation.

We simulate CS data acquisition on 21 frames of a dynamic cardiac MRI sequence[2] by subsampling the Fourier transform of each image (i.e., taking random columns of k-space). Each image is 256x192 pixels, and we recover the images by solving

[2]The MRI data used was acquired using a GE 1.5T TwinSpeed scanner (R12M4) using an 8 element cardiac coil.

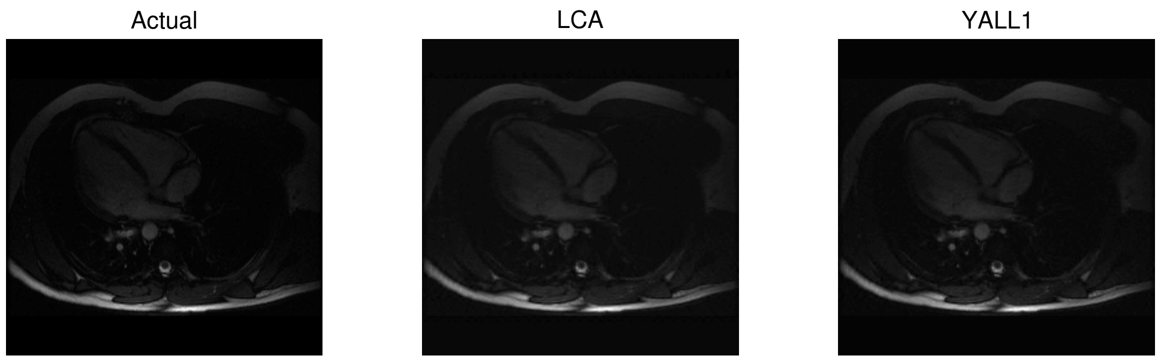


Fig. 13. Reconstruction of 256x192 pixel MR images from simulated CS acquisition. The simulated LCA and the comparison digital algorithm (YALL1) find solutions of approximately the same quality in terms of relative MSE and image quality. YALL1 finds the solution in approximately 10s, while the LCA finds the solution in approximately 20 time constants ($20\mu s$ for $\tau = 1$ µs).

BPDN to find sparse coefficients in a wavelet transform. Specifically, we solve the BPDN optimization program where the sensing matrix $\Phi = FW^H$ is an inverse wavelet transform followed by a subsampled Fourier matrix, and recover the image by taking the wavelet transform of the solution to the BPDN problem. The choice of wavelet transforms in this case is very important, as transforms which are coherent with the Fourier subsampling scheme can result in poor results. We follow the work of [9] and use a 4 level 2-dimensional Daubechies wavelet transform as the sparsifying basis. The resulting optimization is more difficult than the synthetic data in the previous two sections because the signals are larger and the images are sparse in a wavelet basis instead of the canonical basis.

We compare results of recovery using the simulated LCA and another standard digital solver YALL1 [27]. Figure 13 shows an example MRI image and its reconstruction using both the LCA and YALL1. The average relative MSE (using λ = 0.001) over all 21 recovered images was 0.0109 for YALL1 and 0.0106 for the simulated LCA. The relative differences between the LCA and YALL1 solutions was 0.0042, indicating that the solution quality is essentially the same for both approaches. YALL1 took approximately 10 second of computation time to reach this solution (on the same computer platform used in the previous simulations), while the LCA took approximately 20τ simulated seconds. For $\tau = 1$ µs, this translates to datarates of approximately 50 kHz. Recovery for such large-scale problems may require more nodes than a single chip can provide. In these cases stringing together a series of smaller chips or developing a block-wise method of recovery would still allow the benefits of using analog hardware for the CS recovery.

VI. Scaling and Concluding Remarks

The LCA analog circuit has been presented as the solution to the class of sparse approximations defined in (3). A pair of example circuits were implemented on the RASP 2.9v, and successfully converged on results that were similar to a digital solver. This analog solution is particularly targeted for low powered applications, such as channel sensing [10] for portable devices. While we have demonstrated the successful operation of the system at small sizes (N=6), we must scale to much large sizes in order to create a viable application. The simulations provided in the last section demonstrates the potential value of the LCA for CS recovery, validating continued efforts to scale up our implementations.

TABLE I
Performance Comparison

System	LCA	LCA	LCA (Hyp.)	CPU [29]
Size	2×3	4×6	666×1k	1k
Power (Active)	28.3µW	74.6µW	149mW	≈3.8W
(Total)	1.76mW	1.81mW	151mW	≈100W
Time (Cvg.)		240µs	< 240µs	46ms
Time (Total)		266µs	4.62ms	46ms
Error (RMS)	2%	5%	≈ 5%	-
Extra Cost	0.2%	1%	≈ 1%	-

The RASP 2.9v will allow moderate scaling of the LCA. The chip contains 18 8-bit DACs and enough stand-alone nFETs for 36 current mirrors. Since the thresholder nodes require two current mirrors, the number of inputs M and outputs N is limited by $M + 2N \leq 36$. This suggests a practical maximum size of about 8x14. Scaling to this size would not significantly impact total power output (which would still be dominated by overhead costs), and would only meaningfully impact the interface time to load and retrieve data (since convergence time would be relatively fixed).

Scaling to much larger sizes $(N \approx 1000)$ would require a more application specific chip than the RASP 2.9v. This hypothetical chip would require on the order of one million FGEs, which should be implementable given our current technology. The RASP 2.9a, a 5 mm x 5 mm 350 nm process chip used in [20], already contains 133,000 FGEs. Switching to a 130 nm process would allow over one million FGEs on a chip the same size as the RASP 2.9v.

At this scale, we would still not expect the convergence time to change markably (since the capacitive load would not exceed that of a chip pad) as seen in the simulations; the total time would be dominated by interfacing costs, which would scale to 4.4 ms. Improvements could be achieved by implementing some parallelization. Power consumption would be dominated by the $O(N\sqrt{N})$ scaling of the VMM OTAs, to about 149 mW. Accuracy would remain relatively constant, since the average error and average eigenvalue do not scale with problem size. These results are summarized in Table I.

These hypothetical results compare extremely well with state-of-the-art digital BPDN implementations [29], [30]. Borghi et al. report solving BPDN for $N = 1024$ in 46 ms using an Intel i7 CPU (Table 2 in [29]). Estimating that this calculation required 1.2GMACs over 46ms, and that the i7 CPU calculates 7GMAC/s/W, we can estimate the active power requirements for the calculation at 3.8 W, more than 25 times as much power as the LCA.

The LCA could be increased to 4000 nodes by using a full 2 cm x 2 cm reticle, with over 16 million devices. Further scaling would require either multiple chips, or a denser process. Although the circuit shown here had only single sided inputs and outputs, four-quadrant behavior can be easily implemented. Extra nodes must be added to represent negative outputs, while negative multiplication can be induced simply by connecting the driving VMM outputs to the negative input of the thresholding device (or for the recurrent VMM, connecting them to the positive input terminal).

The hardware LCA should be easy to integrate into CS systems, since it already contains mechanisms for rapid data interface. Because the multipliers used here are reprogrammable, we can use the system to recover arbitrary linear compressions of sparse signals (using a number of recovery methods [31]). The multiplier weights can also be made to adapt to structure in the input and learn more efficient dictionaries (as in [7], [32]), allowing this system to be used even when the sparsity basis is unknown. Ultimately, we envision this technology enabling CS recovery with ultra low power (as in [10]) or real time processing (as in [9]).

References

[1] C. Rozell and P. Garrigues, "Analog sparse approximation for compressed sensing recovery," in *Proceedings of the 2010 ASILOMAR Conference on Signals, Systems and Computers*, 2010, pp. 822–826.

[2] S. Shapero, C. Rozell, A. Balavoine, and P. Hasler, "A scalable implementation of sparse approximation on a field programmable analog array," *IEEE Biomedical Circuits and Systems Conference*, 2011.

[3] M. Elad, M. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," in *IEEE Proceedings - Special Issue on Applications of Sparse Representation & Compressive Sensing*, vol. 98, no. 6, April 2010, pp. 972–982.

[4] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–44, 2010.

[5] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 489 – 509, feb. 2006.

[6] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21 –30, March 2008.

[7] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning in a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[8] J. Mattingley and S. Boyd, "Real-time convex optimization in signal processing," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 50–61, May 2010.

[9] S. S. Vasanawala, M. T. Alley, B. A. Hargreaves, R. A. Barth, J. M. Pauly, and M. Lustig, "Improved pediatric MR imaging with compressive sensing," *Radiology*, vol. 256, pp. 607–616, Aug 2010.

[10] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressed channel sensing: A new approach to estimating sparse multipath channels," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1058 –1076, june 2010.

[11] C. Rozell, D. Johnson, R. Baraniuk, and B. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Computation*, vol. 20, no. 10, pp. 2526–2563, Oct. 2008.

[12] A. Balavoine, J. Romberg, and C. Rozell, "Convergence and rate analysis of neural networks for sparse approximation," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1377–89, September 2012.

[13] T. Hall, C. Twigg, P. Hasler, and D. Anderson, "Developing large-scale field-programmable analog arrays," *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, p. 142, Apr. 2004.

[14] B. Olshausen, M. Lewicki *et al.*, "Sparse codes and spikes," *Probabilistic Models of the Brain: Perception and Neural Function*, pp. 257–272, 2001.

[15] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[16] C. Schlottmann and P. Hasler, "A highly dense, lowpower, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE Journal of Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 1–9, 2011.

[17] C. Schlottman, S. Shapero, S. Nease, and P. Hasler, "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE J. Solid-State Circuits*, no. 99, September 2012.

[18] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[19] T. Serrano-Gotarredona, B. Linares-Barranco, and A. Andreou, "Very wide range tunable cmos/bipolar current mirrors with voltage clamped input," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 46, no. 11, pp. 1398 –1407, nov 1999.

[20] S. Shapero and P. Hasler, "Precise programming and mismatch compensation for low power analog computation on an FPAA," *IEEE Trans. Circuits and Systems I, in press*, 2012.

[21] C. C. Enz, F. Krummenacher, and E. A. Vittoz, "An analytical mos transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications," *Analog Integrated Circuits and Signal Processing*, vol. 8, pp. 83 – 114, 1995.

[22] S. Koziol, C. Schlottmann, A. Basu, S. Brink, C. Petre, B. Degnan, S. Ramakrishnan, P. Hasler, and A. Balavoine, "Hardware and software infrastructure for a family of floating-gate based FPAAs," in *IEEE Int. Symp. Circuits and Systems*, May 2010, pp. 2794 – 2797.

[23] C. Schlottmann, C. Petre, and P. Hasler, "A high-level simulink-based tool for FPAA configuration," *IEEE Trans. on Very Large Scale Integr. (VLSI) Syst.*, p. 1, 2011.

[24] F. Baskaya, D. Anderson, P. Hasler, and S. K. Lim, "A generic reconfigurable array specification and programming environment (GRASPER)," in *Circuit Theory and Design, European Conference on*, 2009.

[25] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large scale l1-regularized least squares," *IEEE Journal on Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, Dec 2007.

[26] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, 2007.

[27] E. T. Hale, W. Yin, and Y. Zhang, "Fixed-point continuation for l1-minimization: Methodology and convergence," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1107–1130, 2008.

[28] D. Donoho and J. Tanner, "Sparse nonnegative solution of underdetermined linear equations by linear programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 27, p. 9446, 2005.

[29] A. Borghi, J. Darbon, S. Peyronnet, T. F. Chan, and S. Osher, "A simple compressive sensing algorithm for parallel many-core architectures," *Journal of Signal Processing Systems*, 2012, in Press.

[30] M. Andrecut, "Fast GPU implementation of sparse signal recovery from random projections," *Engineering Letters*, vol. 17, no. 3, pp. 151–158, 2009.

[31] A. Charles, P. Garrigues, and C. Rozell, "A common network architecture efficiently implements a variety of sparsity-based inference problems," *Neural Computation*, 2012, in press.

[32] M. Rehm and F. Sommer, "A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields," *Journal of Computational Neuroscience*, vol. 22, pp. 135–146, 2007.