Analog Integrated Circuits and Signal Processing, 18, 289–299 (1999)

A Nonlinear Noise-Shaping Delta-Sigma Modulator with On-Chip Reinforcement Learning*

GERT CAUWENBERGHS

gert@bach.ece.jhu.edu

Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218-2686

Abstract. The stability and quality of noise shaping is a concern in the design of higher-order delta-sigma modulators for oversampled analog-to-digital conversion. We reformulate noise-shaping modulation alternatively as a nonlinear control problem, where the objective is to find the binary modulation sequence that minimizes signal swing in a cascade of integrators operating on the difference between the input signal and the modulation sequence. Reinforcement learning is used to adaptively optimize a nonlinear neural classifier, which outputs modulation bits from the values of the input signal and integration state variables. Analogous to the pole balancing control problem, a punishment signal triggers learning whenever any of the integrators saturate. Experimental results obtained from a VLSI modulator with integrated classifier, trained to produce stable noise shaping modulation of orders one and two, are presented. The classifier contains an array of 64 locally tuned, binary address-encoded neurons and is trained on-chip with a variant on reinforcement learning.

Keywords: delta-sigma modulation, reinforcement learning, neural networks, analog VLSI

1. Introduction

Noise-shaping modulation architectures such as delta-sigma modulators are attractive for high resolution oversampled A/D (analog-to-digital) data conversion [1], trading bandwidth of a fast, low-resolution converter for improved resolution by oversampling and modulating the signal to push quantization noise out of the signal band. Higher-order delta-sigma modulators offer high resolution at significantly increased signal bandwidths (lower oversampling ratios); however, they are prone to instabilities for orders beyond two. Most higher-order noise-shaping architectures considered so far are direct linear extensions on the standard first-order delta-sigma modulator, shown in Figure 1 (a).

*This work was supported by NSF under Career Award MIP-9702346 and by ARPA/ONR under MURI grant N00014-95-1-0409. Chip fabrication was provided through MOSIS.

We approach noise-shaping modulation alternatively as a nonlinear control problem, where the binary modulation sequence controls a "plant" which consists of a cascade of integrators that operate on the quantization error, *i.e.*, the difference between analog input and binary modulation sequences, shown in Figure 1 (b). The optimal modulation sequence is defined as the one which minimizes the energy (signal swing) of the integration variables. As we will show, this is equivalent to minimizing the low-frequency in-band quantization noise of the modulator. The control problem in our formulation has strong similarities with the pole balancing problem, solved efficiently using reinforcement learning [3].

It may seem counter-intuitive that adding more analog circuit complexity to the design of a delta-sigma modulator could improve rather than degrade performance, due to increased sensitivity to noise and process parameters. In the case considered here, the addi-

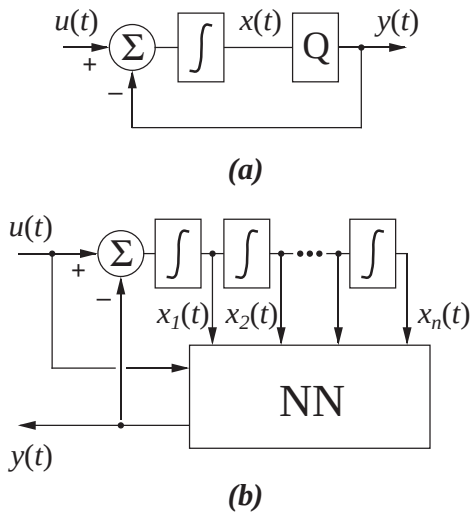

Fig. 1. Noise-shaping modulators for oversampled A/D conversion. *(a)* First-order delta-sigma modulator. *(b)* Generalized higher-order structure with nonlinear classifier. All operations are sampled, discrete-time.

tional circuitry does not affect the analog signal path, but affects the quantized output only. By replacing the quantizer by a more sophisticated nonlinear classifier that adapts to the statistics of the input signal, noise-shaping properties are improved and less quantization noise leaks into the signal frequency band.

Before proceeding, it is important to point out, up front, that the purpose of this paper is *not* to introduce alternative delta-sigma modulator topologies that have better noise-shaping properties *per se*. Rather, we investigate the use of *adaptive* techniques which, combined with more powerful classifier structures and intrincally more stable modulator topologies than used here for demonstration purposes, are expected to yield better noise-shaping performance, better in terms of increased stability at higher orders and more uniform spectral properties. The techniques applied here are general, and are demonstrated on a simple example: for a binary classifier, which is limited for practical use. Although the results are verified through experimental results from a VLSI implementation, the purpose of this paper is thus not to propose new delta-sigma modulator topologies that are ready for practical use, but to define new directions in research on adaptive systems for higher-order noise-shaping, in a longer-term effort towards higher-performance oversampled data converters.

We first define noise-shaping modulation in the nonlinear control framework, and then formulate reinforcement learning as applied to the optimization problem. Next, we present results on training a simple binary address-encoded classifier to produce noise-shaping modulation of integration orders 1 through 3, giving rise to some interesting modulation structures. Finally, we present experimental results on an analog VLSI modulator comprising a cascade of integrators and an array of 64 locally tuned, address-encoded neurons with on-chip learning on a 2 μm CMOS chip.

2. Nonlinear Noise-Shaping Modulation

We state noise-shaping modulation for oversampled A/D conversion as an optimal nonlinear control problem. The order-n modulator comprises a cascade of n integrators $x_i(t)$ operating on the difference between the analog input $u(t)$ and the binary (± 1) modulated output $y(t)$, Figure 1 (b):

$$\begin{aligned} x_1(t+1) &= x_1(t) + a\,(u(t) - y(t)) \\ x_i(t+1) &= x_i(t) + a\,x_{i-1}(t)\,, \quad i = 2, \cdots n \end{aligned} \tag{1}$$

with modulation gain $a = 0.5$. Notice that this choice of noise-shaping filtering through successive integration is not critical, and the cascade of filters in (1) can be replaced by a bank of filters, of which the last one has n poles located near $z = 1$. For generality, denote the z-transforms of the n filters as $H_i(z)$, with

$$X_i(z) = H_i(z)\,(U(z) - Y(z)), \quad i = 1, \cdots n \tag{2}$$

or, equivalently

$$Y(z) = U(z) - {H_i}^{-1}(z)\,X_i(z), \quad i = 1, \cdots n\,. \tag{3}$$

The second term in (3) represents the error in quantizing $U(z)$ as $Y(z)$. The effect of noise shaping is apparent from the high-pass nature of the noise transfer function (NTF) ${H_i}^{-1}(z)$, shaping the noise spectrum $X_i(z)$ to push low-frequency components outside of the signal band. Although this is the same NTF as appears in linear analysis of the delta-sigma modulator [1], where it shapes quantization noise, we are not making any assumptions on explicit use of a *quantizer*. Rather, noise is considered as being contributed directly by the amplitude of the integration state variable $x_i(t)$.

The control objective is to choose the binary sequence $y(t)$ such as to minimize the energy of the state variables $|x_i(t)|^2$ over time. This corresponds to minimizing the power spectrum of the noise $|X_i(z = e^{j\omega T})|^2$ near zero frequencies, $z \approx 1$. This in turn yields a modulation output $Y(z)$ which most faithfully

represents the input signal $U(z)$ at lower frequencies, *i.e.*, in the signal band.

Preference among components i in the minimization of $|x_i(t)|^2$ is given to the last stage in the cascade, $i = n$, which yields noise shaping of the highest order n. The reason for still including the $n-1$ intermediate integration variables is because a cascade of integrators easily saturates for even small perturbations of the input. Saturation of the i-th integrator causes instability of noise shaping at order i, leading to cancellation of the zero in the NTF in (3) at zero frequency ($z = 1$). By still minimizing the preceding integrators in the chain, $x_j(t)$ with $j < i$, stability of noise shaping is enforced at least at a lower order, and at worst at order one ($j = 1$). In other words, stability of noise-shaping can be guaranteed at all times, albeit not necessarily at the maximum order.

Because of the natural tendency of the integrators to saturate, minimizing $|x_i(t)|^2$ is almost equivalent to simply constraining the excursion of the integration variables within saturation limits, $|x_i(t)| < x_{\rm sat}$. Then the control objective becomes equivalent to that in multi-segment pole balancing: control the movement of a cart $y(t)$ driving a multi-segment pole under gravity such as to maintain the segments upward along the vertical axis, trying to confine segment excursions $x_i(t)$ within limits of stability.

3. Reinforcement Learning

Reinforcement learning techniques are applicable to general learning tasks defined by a discrete, delayed, external reward or punishment signal $r(t)$ which serves as the only indication of performance available for training [2, 3, 4, 5, 6]. The classical example where reinforcement learning is applied is the pole-balancing problem, which it efficiently solves [3]. Similarly, in the context of noise-shaping modulation, we use a failure signal, indicating saturation in one or more of the integrators, to reinforce stability of noise-shaping.

A nonlinear neural classifier, depicted in the box labeled "NN" in Figure 1 (b), produces the modulation sequence from the state of the input and integrators, $y(t) = f(u(t), x_i(t))$. Similar to the "boxes-system" used in [3], the classifier chosen here has locally tuned, hard-thresholding neurons, effectively implementing a look-up table from a binary address-encoded representation of the state space spanned by $u(t)$ and $x_i(t)$. In particular, $y(t) = y_{\chi(t)}$ where $\chi(t)$ is the index of the address selected from address bits generated from the sign of the components $u(t)$ and $x_i(t)$:

$$\begin{aligned} \mathrm{U}^+(t) &= \mathrm{sgn}(u(t)) \\ \mathrm{X}_i^+(t) &= \mathrm{sgn}(x_i(t)) \ , \quad i = 1, \cdots n \ . \end{aligned} \tag{4}$$

This simple choice of input representation is by no means designed to be optimal, but serves as a proof of principle, transparent for interpretation and suited to implementation in VLSI. Still, this representation is powerful enough to produce first-order and second-order noise shaping as illustrated below. The limitations of this simple choice of classifier structure on performance will be illustrated with experimental results, and alternatives will be proposed, in Section 5.

As in [3], a "reward" signal $r(t)$ is generated, equal to zero as long as the modulator is stable, and -1 for punishment when failure occurs (saturation of one or more integrators). To reinforce past actions $y(t)$ of the network leading to future reward $r(t)$, an "adaptive critic element" $q(t)$ is trained along with the network to predict the future reward from the present state of the network. Similar to the classifier $y(t)$, the adaptive critic $q(t)$ is implemented as a look-up table $q(t) = q_{\chi(t)}$ using the same address encoding $\chi(t)$ of the state space.

We reformulate reinforcement learning defined in [3] as follows. For a related exposition on reinforcement learning in the framework of perturbative stochastic learning, the reader is referred to [7]. Let y_k and q_k be the adaptive parameters of the classifier and adaptive critic, and let e_k denote the "eligibility" traces of the corresponding neurons. The eligibility is used to keep track of those neurons which were active in the recent history, which become "eligible" for parameter updates under reinforcement. The parameter updates are then given by:

$$\begin{aligned} y_k(t+1) &= y_k(t) + \alpha\, \hat{r}(t)\, e_k(t)\, y_k(t) \\ q_k(t+1) &= q_k(t) + \beta\, \hat{r}(t)\, e_k(t) \end{aligned} \tag{5}$$

where the internal (incremental) reinforcement $\hat{r}(t)$ is constructed as

$$\hat{r}(t) = r(t) + \gamma q(t) - q(t-1) \tag{6}$$

and the eligibilities $e_k(t)$ are updated as

$$\begin{aligned} e_k(t+1) &= \lambda\, e_k(t) + (1-\lambda) \quad && k = \chi(t) \\ &= \lambda\, e_k(t) && \text{otherwise} \end{aligned} \tag{7}$$

with $\chi(t)$ the index of the address selected according to (4).

Table 1. Trained values y_k for $n = 1, 2$.

U^+	$X_1{}^+$	$X_2{}^+$	y (n=1)	y (n=2)
0	0	0	0	0
		1		0
0	1	0	1	1
		1		1
1	0	0	0	1
		1		1
1	1	0	1	1
		1		1

The results of applying reinforcement learning to a classifier to stabilize a cascade of one and two integrators are listed in Table 1 and illustrated in Figure 2. The learning constants used in the simulations were $\alpha = 0.5$, $\beta = 0.01$, $\gamma = 0.9$ and $\lambda = 0.8$. During training, the input $u(t)$ was presented $4,092$ random samples uniformly distributed in the $[0.5 \cdots 0.5]$ interval.

Because of the simple functional form of the classifier (4), we can easily interpret the learned structure from the values of the parameters at convergence, listed in Table 1. For $n = 1$, information on the polarity of the input (U^+) is discarded, and the classifier reduces to a one-bit quantizer $y \equiv X_1^+$ as in the standard first-order delta-sigma modulator. For $n = 2$, the learning apparently yielded a novel structure that uses the available information on the polarities U^+, X_1^+ and X_2^+ in the following way:

$$\begin{aligned} y(t) &= 1 \quad && X_1^+ = X_2^+ = 1 \\ &= -1 \quad && X_1^+ = X_2^+ = -1 \\ &= U^+ \quad && \text{otherwise} . \end{aligned} \tag{8}$$

With no more information on $u(t)$ and $x_i(t)$ available to the classifier, this policy is indeed most sensible in order to minimize risk of future saturation of the integrators.

The output modulation spectra produced by the trained noise-shaping modulators of orders $n = 1, 2$ for a sinusoidal input $u(t)$ with amplitude 0.5 and period 128 are shown in Figure 2. The different orders of noise shaping are apparent from the noise spectrum at lower frequencies.

We have tried to generate third-order noise-shaping with the same state space encoding (4) for $n = 3$, but failed to get stable and consistent results. We attribute this to the coarse encoding, and note that satisfactory results have been obtained for $n = 3$ using a neural classifier with continuous distributed representation, trained through a perturbative version of reinforcement learning [7].

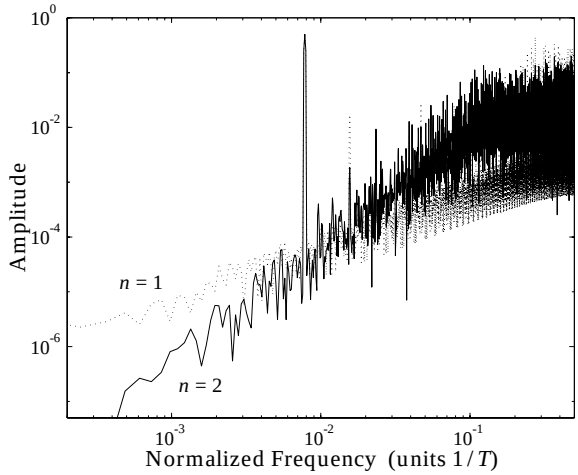

Fig. 2. Simulated output modulation spectra of the trained noise-shaping modulators of orders $n = 1, 2$ for a sinusoidal input with amplitude 0.5 and period 128.

4. Analog VLSI Implementation

4.1. System Level Architecture

We have integrated a cascade of 6 integrators, an 11-bit address state encoder, and an address-encoded classifier with 64 reinforcement learning neurons on a 2.2 mm $\times$ 2.2 mm chip in 2 μm CMOS technology. A floorplan and a micrograph of the VLSI chip are shown in Figures 3 and 4. Although of limited use as presently implemented for demonstration purposes, this chip is to our knowledge the first analog VLSI implementation of delayed reinforcement learning embedded in a real-time application. The first analog VLSI implementation of reinforcement learning was reported in [8]. Reinforcement learning in a digital VLSI framework is described in [9], and "instinct-rule" learning in an analog neural classifier for robot navigation is presented in [10].

The bank of cascaded integrators $x_i(t)$ is implemented using fully differential switched-capacitor circuits. The state encoder χ contains a bank of comparators and logic to encode the state of the input signal and integration variables $u(t)$ and $x_i(t)$, assigning one sign bit and one amplitude bit to each variable, except no amplitude bits for $x_5(t)$ and $x_6(t)$. Each sign bit encodes polarity as in (4). Each amplitude bit encodes a threshold on the absolute signal level, a feature which we did not use in the experiments reported here. Selected bits from the state encoding χ are used to ad-

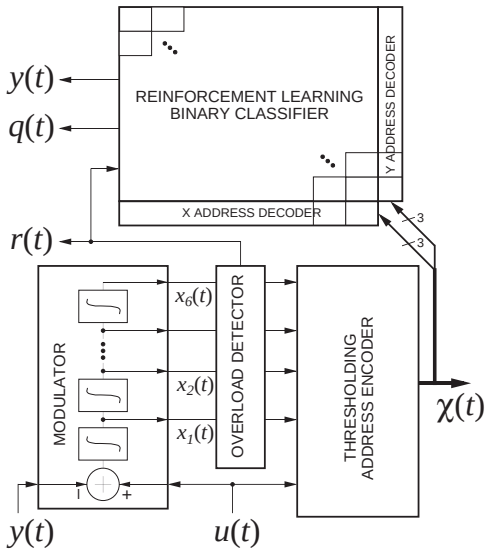

Fig. 3. Floorplan of the noise-shaping modulator with nonlinear classifier and integrated reinforcement learning.

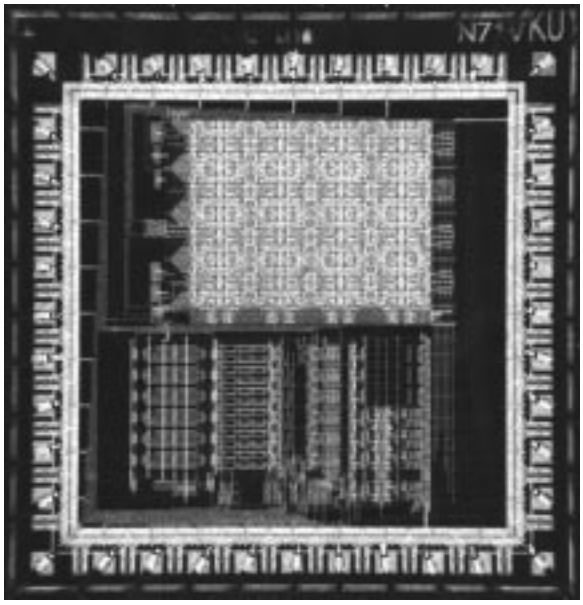

Fig. 4. Micrograph of the 2.2 × 2.25 sq. mm chip in 2 μm CMOS technology.

dress the input space of the classifier, shown on top of the floorplan in Figure 3.

The classifier implements a network of 64 neurons with locally tuned input response, as defined above. For convenience of implementation, the neuron cells are arranged in an 8×8 2-D array as in a RAM, activated through 3 vertical and 3 horizontal address bit lines. The (negative) reinforcement learning signal is obtained from an overload detector which signals saturation of one or more integrator stages in the modulator.

The remainder of this section focuses on the design of the reinforcement learning classifier. The other circuits are fairly standard and their details are beyond the scope of this paper.

4.2. Hysteretic Reinforcement Learning

For a more compact and robust implementation of the reinforcement learning classifier cell, the reinforcement learning algorithms (5)-(7) updating cell parameters y_k and q_k are modified as follows:

$$\begin{aligned} y_k(t+1) &= \mathrm{hyst}\left(y_k(t) - \alpha\,\mathrm{sgn}(y_k(t))\,\mathrm{UPD}_k(t)\right) \\ q_k(t+1) &= q_k(t) - \beta\,\mathrm{UPD}_k(t) \end{aligned} \tag{9}$$

where the auxiliary variable $\mathrm{UPD}_k(t)$ is constructed as

$$\begin{aligned} \mathrm{UPD}_k(t) &= -\hat{r}(t) \quad e_k(t) > 0 \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{10}$$

which is active "one" only when the cell is so unlucky to be eligible during a punishment. The expression $\mathrm{hyst}(\cdot)$ in (9) denotes (optional) hysteresis in the dynamics of y_k: $\mathrm{hyst}(y) \equiv y$ when y retains the same polarity as the previous value of y, and $\mathrm{hyst}(y) \equiv \mathrm{sgn}(y)$ when the polarity of y has flipped from its previous value. In other words, it takes several "punishments" before the cell changes its polarity of y_k, but once it does, the value of y_k moves all the way to the other side. The reason for including this hysteresis in the updates of y_k is to avoid oscillations around zero that may arise under repeated punishment which is inevitable in the initial stages of learning, and to reduce the effect of leakage in volatile storage during training. Finally, the eligibility is constructed as

$$\begin{aligned} e_k(t+1) &= 1 \quad k = \chi(t) \\ &= e_k(t) - \delta \quad \text{otherwise}\,. \end{aligned} \tag{11}$$

The effect of the change in (11) with respect to (7) is that a cell remains equally "eligible" over a fixed time interval after it is selected, as opposed to an exponential decay of eligibility over time through low-pass filtering (7). This implies that all cells active over a given time window prior to failure are equally punished, an assumption which does not drastically affect learning performance but simplifies implementation.

The reinforcement $\hat{r}(t)$ in (10) is presented to all cells as a binary active-low ($\hat{r}(t) = -1$) pulse-code modu-

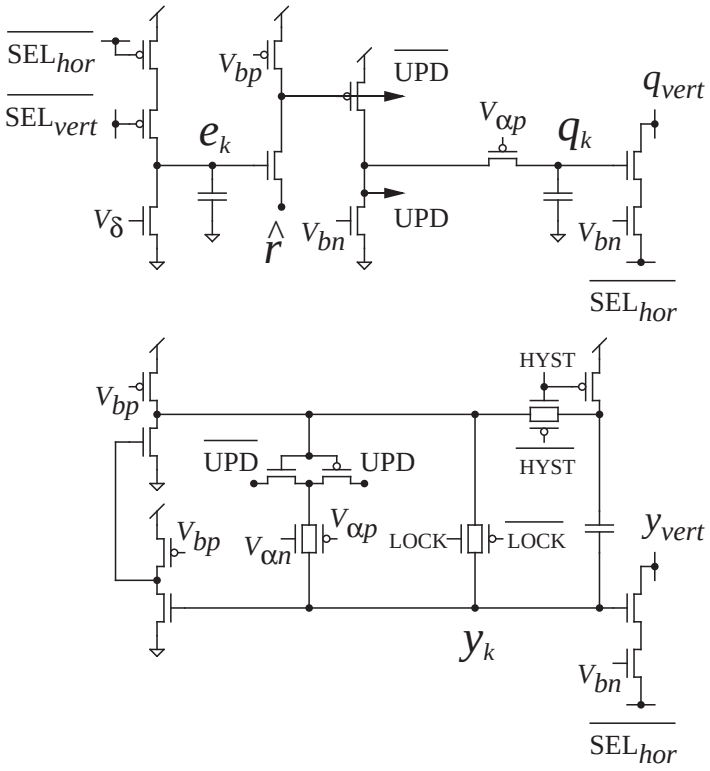

Fig. 5. Circuit diagram of neuron cell with embedded reinforcement learning.

lated signal, and is constructed based on (6). The binary punishment signal $r(t)$ is obtained from the overload detector, and is active-low when any integrator $i \leq n$ saturates.

4.3. Reinforcement Learning Cell

Figure 5 shows the circuit schematic of the reinforcement learning neuron cell, measuring $100 \times 130\ \mu\mathrm{m}^2$ in 2 μm CMOS technology. The eligibility e_k is pulled high when the cell is selected, *i.e.*, both $\overline{\mathrm{SEL}_{hor}}$ and $\overline{\mathrm{SEL}_{vert}}$ are low. When deselected, the eligibility decays thereafter at a constant rate δ determined by the current set by bias voltage V_δ. This implements the modified eligibility equation (11). When $\hat{r}$ is low, and at the same time e_k exceeds a given threshold which depends on V_{bp}, the signal $\overline{\mathrm{UPD}}$ goes low, and its complement UPD high. This in turn implements equation (10), activating the cell to update its parameters.

When UPD is active in the cell, both q_k and y_k are updated. The learning updates are provided by charge-pump circuitry free of switch injection noise, described in [7]. The values of the learning update constants α and β are set through global voltages $V_{\alpha n}$ and $V_{\alpha p}$, biased in the subthreshold MOS region. q_k is always increased by an update as given by (5), and decreases spontaneously only through the leakage current from a reverse diode to ground, not shown in Figure 5. Conversely, the update in y_k is bipolar, always in the direction opposite to y_k. When the control signal HYST is active, the update is hysteretic, by means of positive capacitive feedback through two cascaded pseudo-nMOS inverters in the lower left of Figure 5. This implements the modified hysteretic update equation (9). When LOCK is active, the binary value of y_k is latched, allowing static storage of the y_k parameters after learning.

5. Experimental Results

5.1. Array of Reinforcement Learning Cells

Measurements characterizing the time response of the learning cells to external stimuli, in terms of the constants δ and α in the model (11) and (9), are repre-

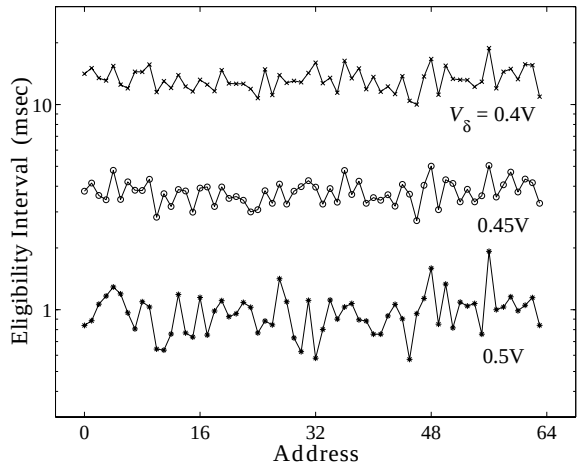

Fig. 6. Time span of the eligibility interval across the array of reinforcement learning cells, measured for different values of δ as set by V_δ. A one-time address selection marks the start of the interval.

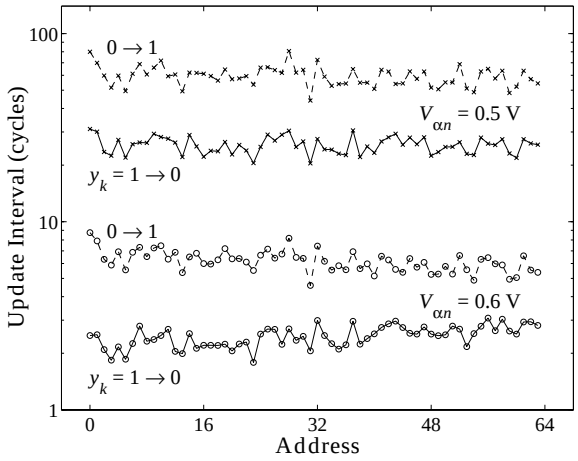

Fig. 7. Number of update cycles between hysteretic alternation in y_k, across the array of reinforcement learning cells, measured for different values of α as set by $V_{\alpha n}$ (and $V_{\alpha p}$). A single 20 μsec negative reinforcement pulse is applied once every update cycle.

sented in Figures 6 and 7, respectively. The eligibility time interval, during which $e_k(t) > 0$, triggered by selection of address k, is shown in Figure 6 for all 64 cells as a function of the bias setting V_δ.

Characteristic measurements of the learning rate α (or, equivalently, the update latency time) are given in Figure 7, which shows the time intervals between consecutive toggles of the binary y_k parameter, during continuously eligible negative reinforcement ($\mathrm{UPD}_k(t) \equiv 1$), and with hysteresis enabled. Note the large asymmetry in the update latency time in the '0' and '1' state of y_k, due to asymmetries in the circuit and its operation. This is easily alleviated in practice by setting $V_{\alpha p}$ at a larger current bias than $V_{\alpha n}$, although this is not important for learning to be successful.

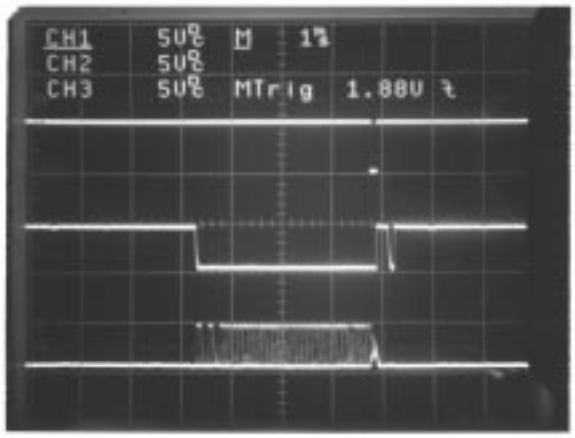

Fig. 8. Oscillogram of $\hat{r}(t)$ (top), $q_k(t)$ (center) and $y_k(t)$ (bottom) waveforms under periodic impulsive reinforcement. See text for explanation.

Update latency for the q_k parameters is qualitatively identical to that for the y_k parameters, except that by construction the latency of the 1 to 0 transition is much more pronounced, set only by diode leakage on the capacitor node q_k in Figure 5.

The relatively large variance of the measured δ and α parameters across cells can be attributed to transistor current mismatch. This level of variance is typical of minimum size MOS devices operated in subthreshold, and its effect does not present a significant problem.

Figure 8 further illustrates the implemented reinforcement learning mechanisms. Shown are the responses of $y(t)$ and $q(t)$ under periodic activation of an impulse reinforced punishment ($\hat{r}(t) = -1$) while synchronously cycling the input address $\chi(t)$ through the 64 neuron cells in sequence, at a rate of 100 μsec per cell. As is evident from Figure 8, the credit assignment mechanism of reinforcement learning identifies those cells k which are active during a time interval immediately preceding the punishment, and updates only these parameters y_k and q_k under the external reinforcement. When activated, y_k steadily changes polarity, and q_k remains low, as expected.

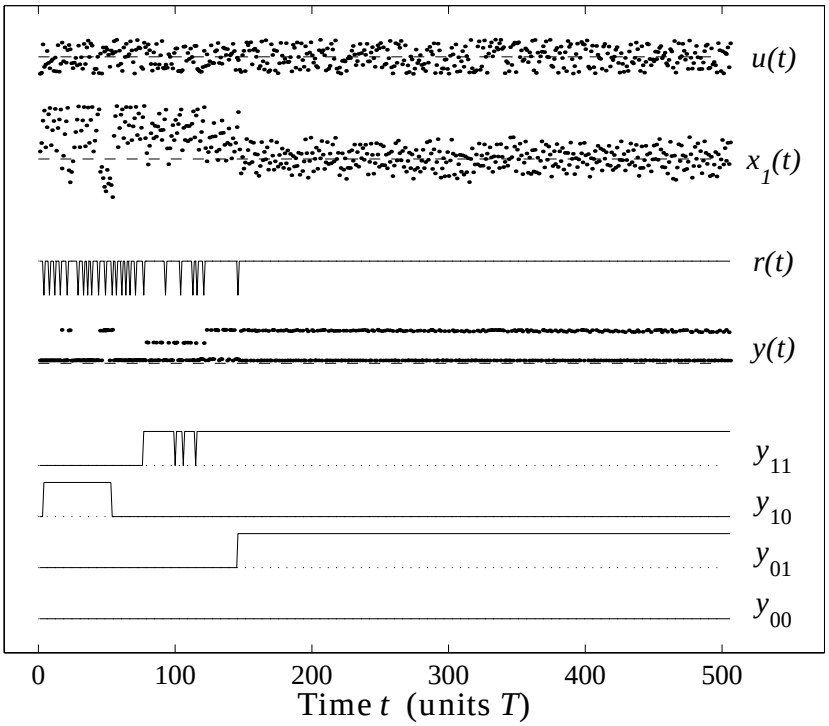

Fig. 9. First-order modulator experiments: recorded dynamics of state variables and parameters during on-chip learning.

Clearly, in an embedded application, the sequence of selected states $\chi(t)$ is not fixed but adapts together with the policy y_k, and the parameters y_k and q_k settle rather than oscillate as the learning converges (*i.e.*, as a more desirable policy is found and punishment occurs less frequently). This is demonstrated below for the purpose of noise-shaping modulation.

5.2. First-Order Noise Shaping Modulation

Test results on training the classifier on-chip to produce noise shaping modulation of order $n = 1$ using the first integrator are shown in Figures 9 and 10. With hysteresis enabled in (9), this fairly simple learning task can be solved without the adaptive critic ($q(t) \equiv 0$), and accordingly we set $\hat{r}(t) \equiv r(t)$. As in the simulations above, the input sequence $u(t)$ during training is uniformly random with half full-scale maximum amplitude (1 V pp), and the integrator variables $x_i(t)$ as well as the eligibilities $e_k(t)$ are reset to zero after every occurrence of failure, $r(t) = -1$. The dynamics of the state variables and parameters recorded during one learning session are given in Figure 9, showing convergence after roughly 150 input presentations. The time step in the experiments was $T = 2.5$ msec, limited by the bandwidth of the instrumentation equipment in the recording. The learned pattern of y_k conforms to that for $n = 1$ in Table 1. Learning succeeded at various values of the learning constants δ and α, affecting mainly the rate of convergence.

Figure 10 shows a record of the time interval between failure in consecutive trials, for 5 different learning sessions, each from random initial conditions of the parameters y_k. As in [3], the displayed time intervals in Figure 10 are numerically averaged over consecutive trials. In most of the cases, convergence is reached in less than 25 trials, *i.e.*, with fewer than 25 parameter update cycles. At convergence, failures still persist, although at a scale of several thousand time steps T. The persistence of failures at convergence is due to the volatile capacitive storage of y_k which causes the correct values to decay and drift in absence of reinforcement, triggering failure each time one of the components y_k reverses polarity.

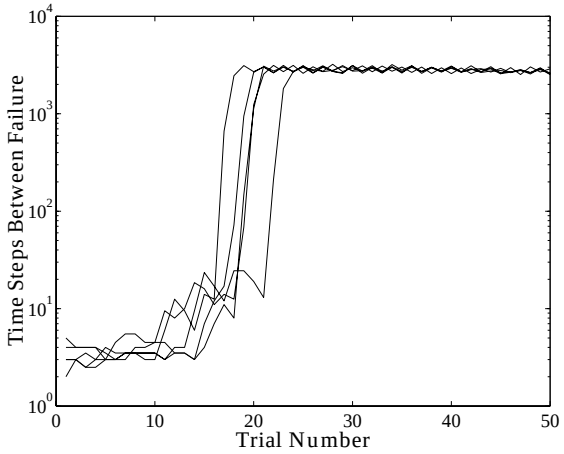

Fig. 10. First-order modulator experiments: failure intervals recorded for 5 learning sessions from random initial conditions.

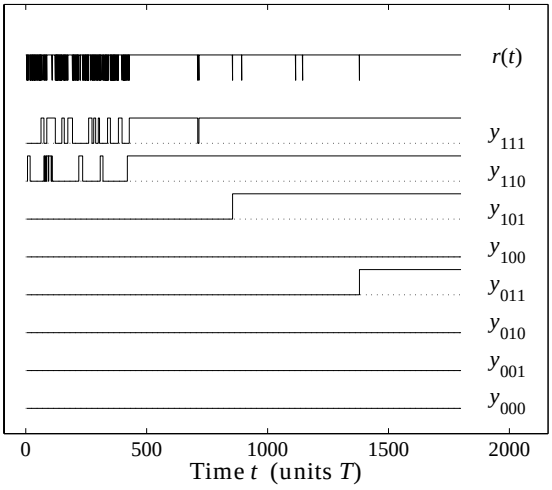

Fig. 11. Second-order modulator experiments: recorded dynamics of the classifier parameters $y_k(t)$ and reinforcement signal $r(t)$ during on-chip learning.

5.3. *Second-Order Noise Shaping Modulation*

Results on training the classifier to stabilize two stages of the integrator bank to produce noise-shaping modulation of order two are represented in Figure 11. Convergence is reached in about 1,400 cycles, which is a factor three slower than in the case $n = 1$. Learning experiments over longer time intervals (10^7 cycles) show that the mean time between failures is about 500 cycles. When the parameters y_k are frozen after convergence (by activating the LOCK switch in Figure 5), the time between failures peaks at 10^5 cycles, and varies with the particular input sequence $u(t)$.

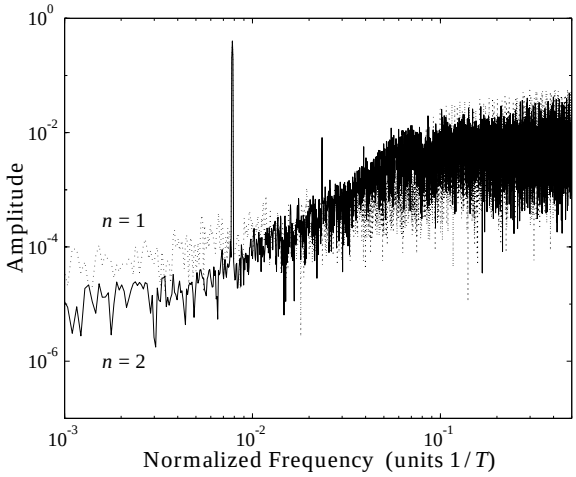

Fig. 12. Experimental output modulation spectra of the trained noise-shaping modulators of orders $n = 1, 2$ for a sinusoidal input with amplitude $0.8\ V$ and period 128.

5.4. *Harmonic Response of the Trained Modulators*

To analyze the effect of the nonlinear classifier and the reinforcement learning on the noise shaping, we tested the trained modulators of orders $n = 1, 2$ under a sinusoidal input. Figure 12 shows the modulation output spectra, obtained by fourier transforming the binary sequences recorded from the chip under a sinusoidal input of amplitude $0.8\ V$ and period 128 samples. Full scale is $1\ V$, and the sampling frequency $1/T$ is 400 Hz, limited by the instrumentation to acquire the data points in the experimental setup. The obtained spectra conform to the simulated spectra in Figure 2, except for the higher noise floor visible at lower frequencies.

The harmonic distortion in the second order modulator, observed both in the simulatated and experimental spectra, is typical for other second-order delta-sigma modulators with large input amplitude (80 % full-scale here). The distortion is due to nonlinear correlations between the integrator output x_2 and the input u over time. This effect is illustrated in Figure 13, showing a fragment of the recorded internal state variables of the second-order modulator. As a consequence of the simple binary structure of the classifier, with no more

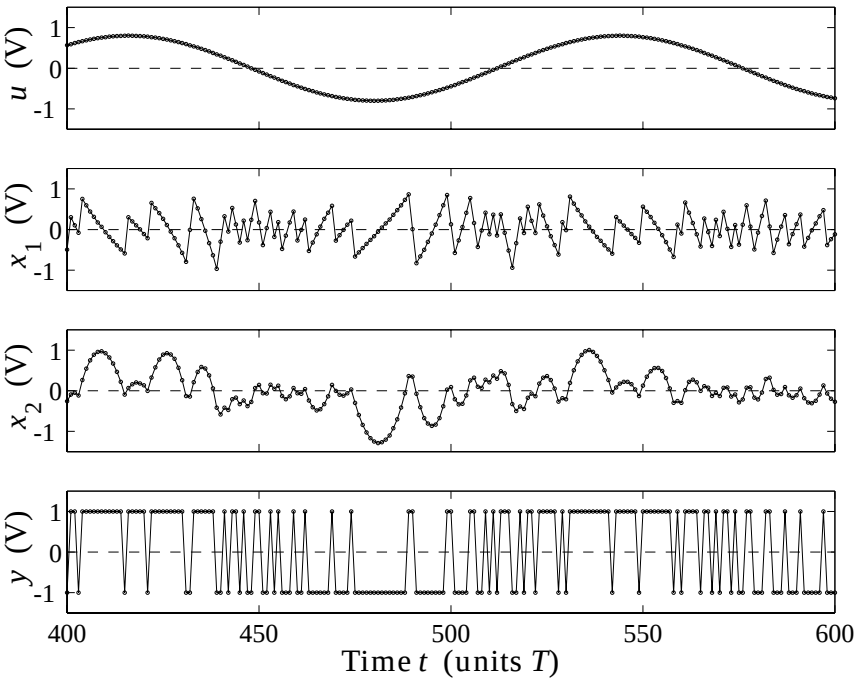

Fig. 13. Fragment of waveforms recorded from the second-order noise-shaping modulator after training, for a sinusoidal input with amplitude $0.8\ V$ and period 128.

information available than just the polarity of the input and integrator state variables, the classifier makes non-optimal decisions in certain situations, especially for input and integrator values close to zero. It is clear that in order to reduce nonlinear distortion, a better classifier is needed which accounts for finer-grain analog amplitude information on the state variables, such as in a neural network with continuous neurons and differentiable sigmoids [7]. Further improvements can be expected from incorporating decision feedback, in which previous bit decisions are used in the classifier, such as commonly used in binary inter-symbol interference equalizers for hard drive storage retrieval [11].

These and other improvements necessarily incur a cost in implementation complexity. Besides algorithmic advances, their efficient implementation at this early stage of development are open problems for continued research in this area. The premise, in the long run, is adaptive higher-order noise-shaping modulation for high-speed oversampled data conversion, continually adapted to the statistics of the input signal.

6. Conclusions

We reformulated noise-shaping modulation for oversampled A/D conversion as a nonlinear control problem with solutions that are stable for higher orders n using an optimization criterion directly on the values of the integration variables. The linear thresholding classifier commonly used in delta-sigma modulation is replaced by a more general nonlinear classifier.

We applied reinforcement learning to train a nonlinear classifier which uses information only on the polarities of the input and integration variables of the modulator, achieving stable noise shaping of orders $n = 1$ and 2.

Finally, we implemented the adaptive modulator including neural classifier and reinforcement learning on a single analog VLSI chip, and presented experimental results on training the system as first-order and second-order noise-shaping modulators.

The remaining challenge is to increase the effective order of noise shaping that can be obtained with an experimental system beyond $n = 2$, requiring a classifier of more refined structure than we considered here. We note that simulations support satisfactory results for

order $n = 3$ using a distributed continuous neural classifier, trained with a perturbative stochastic version of reinforcement learning [7]. Further improvements, for orders $n = 4$ and beyond, require both more advanced training or classification techniques such as variants on heuristic dynamic programming using gradient estimation and prediction in the state space [6] or decision feeback in the classifer [11], as well as careful design of the modulator transfer functions $H_i(z)$ in (2) as a compromise between quality and stability of noise shaping. Besides their efficient implementation in VLSI, the design of such architectures remains an open issue.

Similar principles as the ones presented here for noise-shaping modulation can be applied to problems in communications and pattern recognition that call for adaptive control, with less exacting stability requirements than a bank of integrators, for which a simple VLSI approach as demonstrated here may be more than appropriate. Examples can be found in adaptive nonlinear predictive speech coding, and decision-feedback disk drive read inter-symbol interference equalization [11].

References

1. J.C. Candy and G.C. Temes, "Oversampled Methods for A/D and D/A Conversion," in *Oversampled Delta-Sigma Data Converters*, IEEE Press, pp 1-29, 1992.
2. S. Grossberg, "A Neural Model of Attention, Reinforcement, and Discrimination Learning," *International Review of Neurobiology*, vol. **18**, pp 263-327, 1975.
3. A.G. Barto, R.S. Sutton, and C.W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. **13** (5), pp 834-846, 1983.
4. R.S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, vol. **3**, pp 9-44, 1988.
5. C. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. **8**, pp 279-292, 1992.
6. P.J. Werbos, "A Menu of Designs for Reinforcement Learning Over Time," in *Neural Networks for Control*,, W.T. Miller, R.S. Sutton and P.J. Werbos, Eds., Cambridge, MA: MIT Press, 1990, pp 67-95.
7. G. Cauwenberghs, "Analog VLSI Stochastic Perturbative Learning Architectures," *Analog Integrated Circuits and Signal Processing*, vol. **13** (1/2), pp 195-209, 1997.
8. C. Schneider and H. Card, "Analog CMOS Synaptic Learning Circuits Adapted from Invertebrate Biology," *IEEE T. Circ. Syst.*, vol. **38** (12), pp 1430-1438, Dec. 1991.
9. T.G. Clarkson, C.K. Ng and Y. Guan, "The pRAM: An Adaptive VLSI Chip," *IEEE Trans. on Neural Networks*, vol. **4** (3), pp 408-412, 1993.
10. G. Jackson and A.F. Murray, "Competence Acquisition in an Autonomous Mobile Robot using Hardware Neural Techniques," in *Adv. Neural Information Processing Systems*, Cambridge, MA: MIT Press, vol. **8**, pp. 1031-1037, 1996.
11. B.C. Rothenberg, J.E.C. Brown, P.J. Hurst and S.H. Lewis, "A Mixed-Signal RAM Decision-Feedback Equalizer for Disk Drives," *IEEE J. Solid-State Circuits*, vol. **32** (5), pp 713-721, 1997.

Gert Cauwenberghs received the engineer's degree in applied physics from the University of Brussels, Belgium, in 1988, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology in 1989 and 1994, respectively. In 1994, he joined Johns Hopkins University as an assistant professor in electrical and computer engineering. His research covers analog and digital VLSI circuits, systems and algorithms for parallel signal processing and adaptive neural computation. He received the National Science Foundation Career Award in 1997.