CMOS Analog MAP Decoder for (8,4) Hamming Code

Chris Winstead, *Student Member, IEEE*, Jie Dai, *Member, IEEE*, Shuhuan Yu, *Student Member, IEEE*, Chris Myers, *Member, IEEE*, Reid R. Harrison, *Member, IEEE*, and Christian Schlegel, *Senior Member, IEEE*

***Abstract*—Design and test results for a fully integrated translinear tail-biting MAP error-control decoder are presented. Decoder designs have been reported for various applications which make use of analog computation, mostly for Viterbi-style decoders. MAP decoders are more complex, and are necessary components of powerful iterative decoding systems such as Turbo codes. Analog circuits may require less area and power than digital implementations in high-speed iterative applications. Our (8, 4) Hamming decoder, implemented in an AMI 0.5-μm process, is the first functioning CMOS analog MAP decoder. While designed to operate in subthreshold, the decoder also functions above threshold with a small performance penalty. The chip has been tested at bit rates up to 2 Mb/s, and simulations indicate a top speed of about 10 Mb/s in strong inversion. The decoder circuit size is 0.82 mm^2, and typical power consumption is 1 mW at 1 Mb/s.**

***Index Terms*—Analog decoding, error-control codes, iterative decoders, maximum *a posteriori* (MAP) decoding, translinear circuits.**

I. Introduction

THE maximum *a posteriori* (MAP) decoder is in general the optimal decoder solution for an error-control coded system. A compact MAP decoding algorithm, called the BCJR algorithm, is available for systems which use trellis coding [1]. In a trellis coded system, the maximum likelihood solution provided by a Viterbi decoder is often equivalent to the MAP solution [2]. Viterbi implementations are much simpler than BCJR implementations, and have been traditionally preferred.

Dramatic gains in error-control performance are possible with Turbo codes and similar systems which use iterative decoding techniques. In these systems, multiple MAP-style decoders are allowed to share information during the decoding process [3]. The need for MAP decoders in Turbo implementations has prompted considerable research in efficient MAP decoder designs.

Numerous researchers have reported gains in complexity and power consumption through analog implementations of Viterbi decoders (e.g., [4], [5]). It was recently suggested that similar gains might be obtained in analog implementations of MAP decoders [6], [7]. Several proof-of-concept designs have been reported using BiCMOS circuits, and CMOS approaches have been proposed.

Analog MAP and Turbo-style decoders are typically designed for complete parallel decoding of received data blocks. Larger block lengths provide better error control in coded systems (Turbo-coded systems typically use block lengths greater than 1000 bits). Analog decoders for large block lengths are therefore expected to accommodate speeds of several gigabits per second.

Several small analog MAP decoder designs have been demonstrated using BiCMOS circuits [8], [9]. A more ambitious, strictly CMOS design followed, but failed to produce a working chip [10]. Our design implements a code with a very small block length, similar to previous BiCMOS designs. It is intended to serve as a proof-of-concept implementation for CMOS analog MAP decoders.

Ours is the first functioning CMOS analog implementation of the MAP algorithm for which detailed physical measurements have been made. It is also the first analog MAP decoder to be tested with a serial sample-and-hold (S/H) interface, which would be required in a complete receiver implementation. Our design also incorporates a final stage of on-chip comparators, which would also be necessary in a complete receiver system. Previous measurements for analog MAP decoders have not included the effects of interface circuits.

Because of its small block length, the Hamming decoder must be biased in strong inversion (bias current above 14 nA) in order to operate at testable speeds (above 1 kb/s). The circuits are formally designed to operate in weak inversion. It has been conjectured that CMOS analog MAP decoders based on the circuits of [7] may continue to perform well in strong inversion. Some simulation results have added weight to this conjecture [11], [10], but no previous performance measurements have been made for a physical decoder in strong inversion.

Section II provides a brief review of the (8,4) Hamming code and the operations required for MAP decoding. Design details of the decoder and interface circuit are presented in Section III. Section IV presents bit-error-rate (BER) measurements for our design in strong inversion and compares them with simulation results.

II. BCJR Decoding Algorithm

In a trellis coded system, information is passed through an encoder which adds redundancy before the data is transmitted

Manuscript received January 6, 2003; revised September 11, 2003. This work was supported by the National Science Foundation under Grant CCR9971168.

C. Winstead and C. Schlegel are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4 Canada (e-mail: winstead@ee.ualberta.ca).

J. Dai was with the Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT 84112-9206 USA. He is now with XGI Technology, Inc., Santa Clara, CA 95054 USA.

S. Yu, C. Myers, and R. R. Harrison are with the Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT 84112-9206 USA.

Digital Object Identifier 10.1109/JSSC.2003.820845

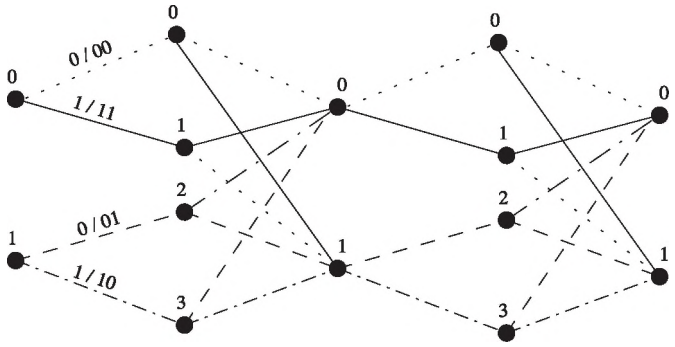


Fig. 1. Tail-biting trellis for (8, 4) Hamming code, unwrapped.

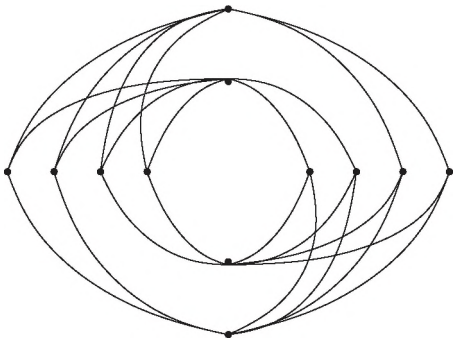


Fig. 2. Actual tail-biting Hamming trellis shape.

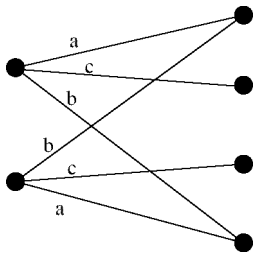


Fig. 3. Example trellis section.

or stored. The receiving device must then infer the encoder's original inputs based on observations of the noise-corrupted outputs. This inference, called *decoding*, can be performed using a hidden Markov model of the encoder.

The Markov state-transition graph for the encoder is called the code's *trellis*. A trellis graph consists of *states* (dots) arranged in columns, and branches which connect the states between the columns. Each column represents the set of possible system states at a particular time. Time increases in discrete steps from left to right in the trellis, and the branches indicate the kinds of transitions that can occur in a single time interval. The decoding problem is to find the most probable sequence of transitions in the graph, given observations of the channel's output. The sequence of trellis transitions has a one-to-one correspondence with the encoder's original input. The trellis for the (8,4) Hamming code is shown in Fig. 1. The trellis of Fig. 1 has minimum complexity for this code, as reported in [12].

In a block code, the data is encoded block-by-block: k bits of uncoded information become n bits of coded information, and the corresponding trellis terminates after L transitions. For the (8,4) Hamming trellis of Fig. 1, $k = 4$, $n = 8$, and $L = 4$. Each branch in the Hamming trellis of Fig. 1 is labeled with the appropriate input/output pair. The line-styles used on branches in Fig. 1 indicate the output: solid branches indicate an output of 11, dotted branches indicate 00, dashed, 01, and dash-dot, 10. Of the two branches leaving any state in Fig. 1, the upper branch always corresponds to an input of 0 and the lower lower branch to an input of 1. One bit of input and two bits of output occur with each transition. Valid trellis paths are restricted to begin and end in the same state, i.e., the trellis is connected in a loop and all valid paths must be continuous along that loop. Fig. 2 illustrates the circular structure of the Hamming trellis.

The BCJR decoding algorithm, named for the authors who introduced it in [1], performs MAP decoding by propagating information along branches of the trellis. A simplified version based on Markov transition matrices is examined in [13], and is summarized here. Each section of the trellis consists of left states (denoted σ_i), right states (denoted σ'_j), and labeled branches which connect them. The Markov transition matrix Γ is constructed by letting $\gamma_{ij} = Pr\left\{\sigma'_j \mid \sigma_i\right\}$ if a branch exists, and $\gamma_{ij} = 0$ otherwise. Thus, γ_{ij} is the probability of a transition occurring between σ_i and σ'_j. An example trellis section is shown in Fig. 3, in which the branches are labeled with output symbols only. The transition matrix for this section is

$$\Gamma = \begin{bmatrix} Pr(a) & Pr(c) & 0 & Pr(b) \\ Pr(b) & 0 & Pr(c) & Pr(a) \end{bmatrix}.$$

In the BCJR algorithm, the transition probabilities are obtained from channel observations: $\gamma_{ij} = P\left\{l\left(\sigma_i, \sigma'_j\right) \mid y\right\}$, where $l\left(\sigma_i, \sigma'_j\right)$ denotes the output on the branch between σ_i and σ'_j, and y is a measurement from the channel. Using this procedure, and after a complete block of channel observations is made, a transition matrix can be constructed for each section of the trellis, indexed $\Gamma_1, \Gamma_2, \ldots, \Gamma_L$.

The states at each time index in the trellis are initialized with a pair of uniform distributions, and the algorithm is carried out by propagating transition probabilities forward and backward around the trellis. A *state-probability distribution* is a row vector which represents the conditional probabilities for a particular column of states in the graph. The α distribution represents forward-propagating information, and the β distribution represents backward-propagating information. The propagation rule is

$$\alpha_{l+1} = \alpha_l \cdot \Gamma_l \tag{1}$$

$$\beta_l = \beta_{l+1} \cdot \Gamma_l^T. \tag{2}$$

The $l+1$ and $l-1$ operations are modulo-L. The propagation is continued until information has had time to make two or three complete passes around the trellis in both directions. The final decoding decision is made bitwise:

$$P_0 = \sum_{(i,j) \in U_0} \alpha_l^{(i)} \cdot \beta_{l+1}^{(j)} \cdot \gamma_{ij} \tag{3}$$

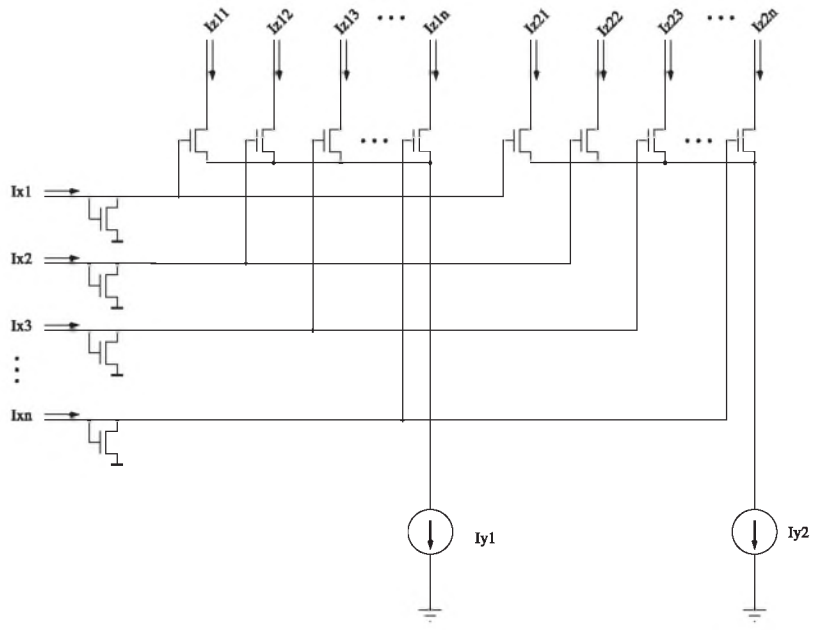


Fig. 4. Gilbert-style vector multiplier circuit.

where U_0 is the set of all branches in section l for which the input bit was 0, $\alpha_l^{(i)}$ is the ith member of α_l, $\beta_{l+1}^{(j)}$ is the jth member of β_{l+1}, and γ_{ij} is the $(i,\ j)$th member of Γ_l. Similarly

$$P_1 = \sum_{(i,j)\in U_1} \alpha_l^{(i)} \cdot \beta_{l+1}^{(j)} \cdot \gamma_{ij}. \tag{4}$$

The final decision rule for bit l is

$$\widehat{u}_l = \operatorname{sgn}(P_1 - P_0). \tag{5}$$

For the Hamming trellis of Fig. 1, an additional simplification may be made. In each section l of the Hamming trellis, the odd-numbered states correspond to $u_{l-1} = 1$ and the even-numbered states correspond to $u_{l-1} = 0$. Each α_{l+1} can be written as a partial sum of (3) or (4), as can be verified by expanding the terms of the matrix multiplication (1). Thus, the decision rule may be revised as follows:

$$P_1 = \sum_{i\ odd} \alpha_{l+1}^{(i)} \cdot \beta_{l+1}^{(i)} \tag{6}$$

$$P_0 = \sum_{i\ even} \alpha_{l+1}^{(i)} \cdot \beta_{l+1}^{(i)} \tag{7}$$

$$\widehat{u}_l = \operatorname{sgn}(P_1 - P_0). \tag{8}$$

This simplification has been used in previous decoder designs, as in [10].

III. Circuit Description

A. MOS Transistors in Weak Inversion

When the drain current I_d of an MOS transistor is sufficiently low, the channel is "weakly inverted" and current flow is primarily a diffusion process. When the drain current is sufficiently high, the channel is strongly inverted, where drift is the dominant mechanism for current flow. The transition between weak and strong inversion occurs in the vicinity of the *specific current* $I_s = (1/\kappa)\, 2\mu C_{\rm ox} U_t^2\, (W/L)$, where U_t is the thermal voltage, $C_{\rm ox}$ and μ have their usual meanings, and κ is a unitless constant which depends on the fabrication process [14]. A typical value for κ is 0.7. We will say that a MOS device with drain current I_d is in *weak inversion* if $I_d < I_s/10$. If $I_d > 10 \cdot I_s$, the device is said to be in *strong inversion*. If $I_s/10 < I_d < 10 \cdot I_s$, the device is said to be in *moderate inversion*. Both drift and diffusion mechanisms play a significant role in current flow in moderate inversion. For the process in which our design is fabricated, $I_s \approx 140$ nA.

When operating in weak inversion, the transistor's current responds exponentially to the gate–source voltage v_{gs}. If the drain–source voltage v_{gd} is greater than about 250 mV, then the device is said to be in *saturation*. For a saturated transistor in weak inversion, the drain current approximately obeys $I_d = I_s \cdot \exp(\kappa \cdot v_{gs}/U_t)$, or $I_d \propto \exp(v'_{gs})$, where v'_{gs} is the normalized gate–source voltage with units of $(\text{Volts}) \cdot \kappa/U_t$. Weak inversion is for the most part equivalent to *subthreshold* operation, in which the transistor's gate–source voltage is below the threshold voltage of the device. MOS devices in strong inversion obey the usual square law.

B. Multiplier Circuit

The central feature of a BCJR decoder is matrix multiplication of probability distributions (1), (2). At each stage of propagation, the α or β vectors may be multiplied by a normalizing constant without affecting the decoding result. Therefore, as proposed in [7], a normalizing translinear Gilbert multiplier may serve as the basis for the multiplication operation.

A CMOS Gilbert-style vector multiplication circuit is shown in Fig. 4. The circuit is based on the *translinear principle* [14]. The translinear principle states, roughly, that in a closed loop of gate–source device terminals, $\sum_i v_{gs}^{(i)} = \sum_j v_{gs}^{(j)} \longrightarrow \prod_i I_D^{(i)} = \prod_j I_D^{(j)}$. The translinear principle is derived from Kirchoff's law and the fact that

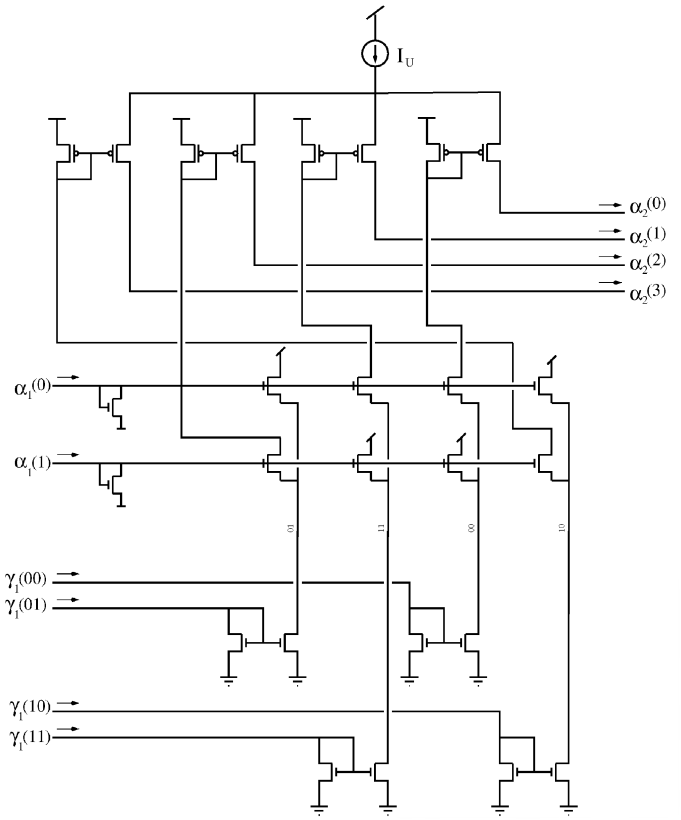


Fig. 5. Forward-propagating multiplier for the first trellis section.

$I_D \propto \exp\left(v'_{gs}\right)$ when all transistors are assumed to operate in weak inversion. If the left input currents ($\vec{\alpha}$) and the bottom input currents ($\vec{\gamma}$) are represented by column vectors, then the operation performed by the multiplier is approximately

$$Z = \frac{\vec{\alpha} \cdot \vec{\gamma}^{T}}{\sum_i \alpha_i} \tag{9}$$

where Z is a matrix of output currents. A detailed derivation for (9) is provided in [7].

This circuit thus produces every pair of products of components of $\vec{\alpha}$ and $\vec{\gamma}$. The Gilbert multiplier is therefore suitable for BCJR implementation if probabilities are represented by analog currents. A global unit-probability current I_U is used to normalize all distributions so that the denominator in (9) is equivalent to a probability of one and can be removed from the expression. To complete the matrix multiplication, unwanted products are discarded by tying the drains of corresponding transistors to V_{DD}, and addition is accomplished by shorting wires. Fig. 5 shows a circuit for the forward propagation of the first (leftmost) trellis section from Fig. 1. Other sections have similar implementations.

All operations required for forward and backward propagation [(1) and (2)] and output summarization [(6) and (7)] can be represented as matrix operations, and can be directly implemented by Gilbert vector multipliers. The final decision of (8) is performed by a current comparator.

A top-level diagram of the decoder is shown in Fig. 6. The numbered boxes indicate individual trellis sections. The inputs

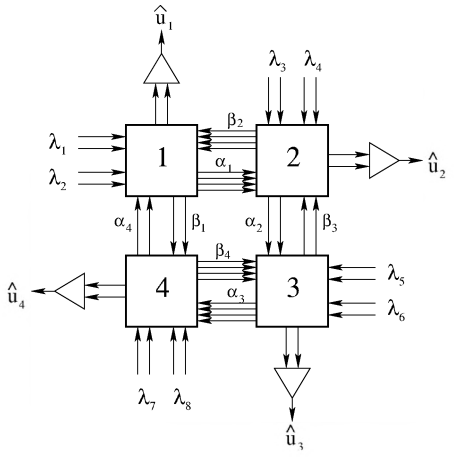


Fig. 6. Block diagram of the decoder.

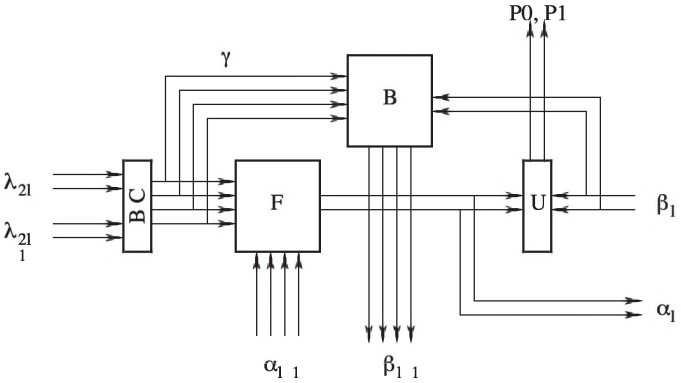


Fig. 7. Block diagram of one trellis section.

to the circuit, denoted by λ_i, correspond bit value probabilities derived from individual channel observations. The trellis branch probabilities γ_l are the joint probabilities of a pair of adjacent bit values.

A block diagram of the computation required for a trellis section is shown in Fig. 7, in which Gilbert multiplier circuits are indicated by boxes. The box labeled *BC* is a *bit-combining circuit* which multiplies pairs of symbol probabilities λ to compute branch probabilities γ. The box labeled *F* does the forward propagation, *B* does the backward propagation, and *U* does the final "upward" computation of (6) and (7). Fig. 7 specifically represents the implementation of sections two and four of Fig. 6. Sections one and three are implemented using the same basic structure.

C. Sample-and-Hold Input Buffer

The BCJR algorithm occurs in parallel once a complete block of channel observations has been made. Channel information arrives serially, and must be stored in an analog serial-to-parallel buffer, from which it is pipelined to the decoder as an entire block of samples. A simple differential S/H circuit, shown in Fig. 8, suffices to store the incoming analog samples. To facilitate testing, a parallel chip input mode can be selected in our design which bypasses the S/H circuits.

The circuit uses CMOS transmission gates as switches. The N- and P-type transistors in the transmission gates have size $W/L = 1.8\ \mu\mathrm{m}/0.6\ \mu\mathrm{m}$. The two S/H stages are isolated by

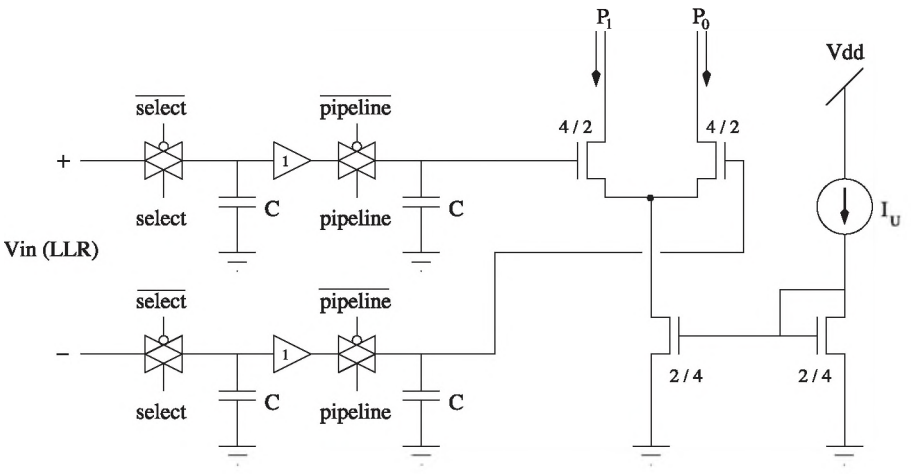


Fig. 8. Circuit differential S/H buffer.

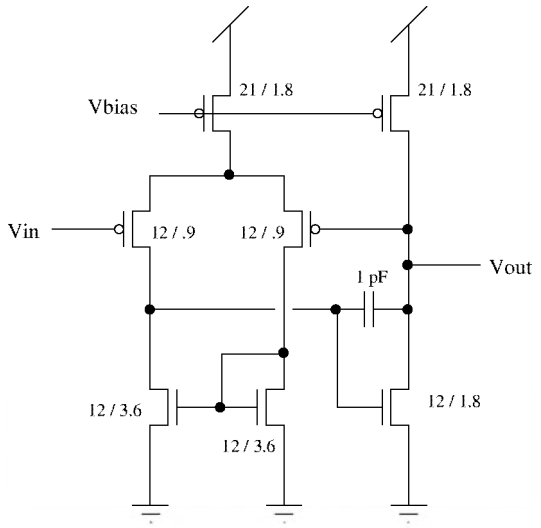


Fig. 9. Unity-gain buffer circuit. Transistor sizes are indicated in microns.

a unity-gain buffer, shown in Fig. 9. The buffer has a -3-dB frequency of 100 MHz. Each S/H subcircuit uses a 200-fF capacitor. The timing of the pipeline and other signals is discussed in Section III.

The decoder accepts differential voltage inputs which represent log-likelihood ratios (LLRs). The LLR format is commonly provided by analog receiver front-end circuits. A log-likelihood ratio X for a binary random variable x is defined as $X = \ln\left(Pr(x=1)/Pr(x=0)\right)$. LLRs, represented by differential voltages, are converted into probability currents by a differential pair biased in weak inversion, as illustrated in Fig. 8. If the differential input $V_X = s \cdot X$ for a suitable scaling constant s with units V/LLR, and all transistors are in saturation, then the current outputs are approximately

$$P_1 = I_U \frac{e^X}{1+e^X} \tag{10}$$

$$P_0 = I_U \frac{e^{-X}}{1+e^{-X}}. \tag{11}$$

When the differential pair is biased in strong inversion, the fit to (10) and (11) is less exact. An approximate fit is obtained by adjusting s.

In weak inversion, the scaling factor is $s = U_t/\kappa \approx 0.04$ V/LLR. In moderate and strong inversion, the best fit must be found by simulation for each I_U. The best-fit scaling factor in strong inversion is approximately linearly proportional to I_U. For testing at 1 Mb/s, $s = 0.07$ V/LLR was used to obtain the results in Section IV. This value was found by minimizing the mean squared error between SPICE simulations of the differential pair and the ideal behavior (10) and (11).

Differential storage eliminates distortion caused by leakage currents and reduces the effect of charge injection. When a CMOS transmission gate is turned off, a reverse-biased diode is effectively created via the source/substrate junctions of each device. This causes a nearly constant leakage current which drains charge from each capacitor in Fig. 8. This current is very weakly dependent on the stored voltage. With time, nearly the same amount of charge loss is experienced by each capacitor, which has no effect on the stored differential value.

Charge injection is a significant source of distortion in S/H circuits. When a CMOS transmission gate is switched off, the channel charge is expelled and a portion of it, ΔQ, is deposited on the capacitor C. ΔQ is signal dependent, so that, if the voltage stored on the capacitor is V_C, then $\Delta Q = f(V_C)$. While f is in general a nonlinear function, it is sometimes appropriate to approximate it by a linear function $\Delta Q \approx k \cdot V_C$. This approximation is appropriate for our design because the variation in scaled LLR inputs is typically small (less than 100 mV).

Let $V_{\text{in}} = V_{\text{in}}^+ - V_{\text{in}}^-$ refer to the scaled-LLR differential input to the S/H circuit, and let $V_{\text{out}} = V_{\text{out}}^+ - V_{\text{out}}^-$ refer to the output. The circuit's output after charge injection is approximately

$$V_{\text{out}} = V_{\text{in}}\left(1 + \frac{k}{C}\right). \tag{12}$$

As shown in (12), the effect of charge injection is approximately equivalent to multiplication of V_{in} by a constant slightly greater than one. Simulations of the (8,4) Hamming decoder indicate that performance is not affected when inputs are scaled by factors as large as 2 and as small as 0.5. This is not a surprising result; the textbook MAP decoding procedure for additive white Gaussian noise channels, based on minimum-Euclidean

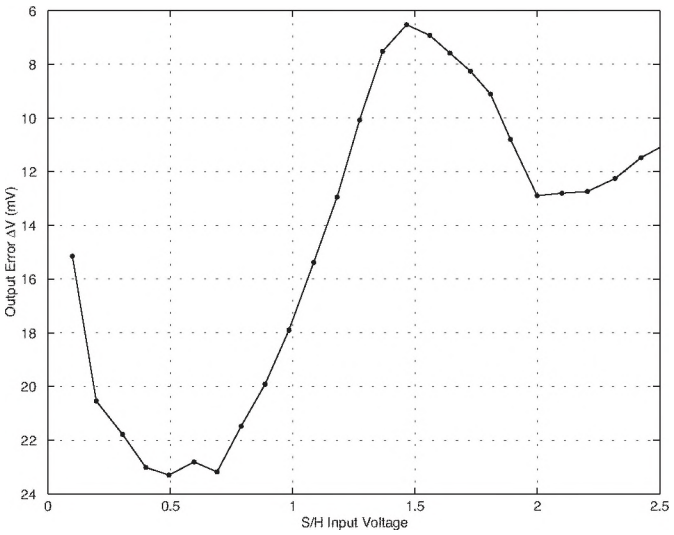


Fig. 10. Error caused by charge injection in the S/H.

distance, is completely invariant under scaling by a constant positive factor.

The linear approximation used to obtain (12) is further validated by physical measurements shown in Fig. 10. The curve in Fig. 10 is the measured shape of the charge-injection function $f(V_C)$. The details of these measurements are explained in Section IV-A. Fig. 10 shows that $f(V_C)$ is roughly linear about an offset of 1.2 V. Therefore, 1.2 V is chosen as the common-mode voltage of input samples used to test the (8,4) Hamming decoder.

D. Output Comparator and Digital Control Circuits

The final bit decisions are made by a latched current comparator, shown in Fig. 11. Monte Carlo SPICE simulations were performed on the comparator circuit, using an estimate of mismatch characteristics for our process. The details of mismatch measurements are discussed in Section IV-A. Based on this analysis, the input offset of the current-comparator has an estimated standard deviation of between 15% and 20% of the operating current I_U. The effect of comparator offset on the decoder's performance is largest at low signal energies, diminishing as the signal-to-noise ratio is increased.

The comparator outputs are passed to a shift register, allowing serial output of the decoded bits. A parallel output mode is also selectable, and the decoder's analog outputs are mirrored to separate pins for testing. A clock-generator circuit coordinates the comparator latch signal and the input select signals for the S/H buffer. There are eight separate select signals. Each select signal is enabled, then disabled, sequentially until a block of samples is received. A global *reset* signal (provided from off chip) identifies the start of a block.

A *pipeline* signal coincides with the eighth *select* signal, and causes all stored samples to be resampled simultaneously by a second S/H buffer. The second buffer holds the samples, presenting them in parallel to the decoder until decoding is complete. Five clock cycles are allocated for decoding (including the time during which *pipeline* is high). During decoding, the comparator latch signal, *vlatch*, is low. The *vlatch* signal is high during the fifth through the seventh clock cycles. This pipelining scheme results in a two-codeword delay before outputs can be sampled.

IV. Experimental Results

The decoder chip, shown in Fig. 12, was fabricated in an AMI 0.5-μm process. A second chip containing test structures was also fabricated. Basic design features of the decoder chip are summarized in Table I. Transistor sizes are reported for the core decoder circuit, in which each transistor has a W/L ratio of 2 for transistors used in Gilbert multipliers and 0.5 for transistors used in current mirrors. The reported decoder power consumption refers to the power consumed in the core decoder, excluding the interfaces. The chip's behavior was verified at speeds from 1 kb/s to 2 Mb/s. Typical power consumption is between 10 and 100 μW, corresponding to speeds between 1 and 100 kb/s.

A. Test Chip Results

The test chip contains an array of 41 P-type and 41 N-type transistors used to measure the current-mode mismatch variance for the AMI process. This data was used to build a rough model of transistor mismatch used for the Monte Carlo simulations on the comparator circuit, as mentioned in Section III-D. However, the number of samples used in this measurement is small, and the results should not be considered exact.

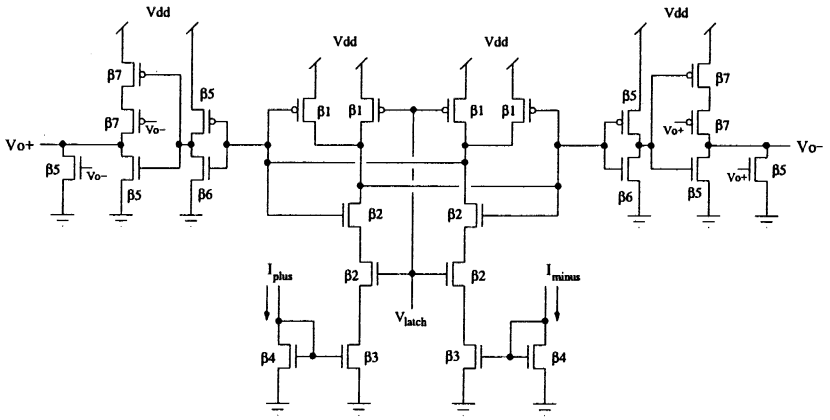
The test chip also contains an S/H circuit as in Fig. 8, which is used to measure charge injection and leakage currents. The S/H circuit used in the test chip only contains one storage capacitor and a unity-gain buffer. The remaining circuitry of Fig. 8 is not of interest in these tests.

Five identical test chips were measured and their data combined. Transmission gate leakage currents are found to be in the range of 5.6×10^{-17} A to -1.1×10^{-17} A. Measured charge-injection offsets in this design are shown in Fig. 10. The data of Fig. 10 was collected by exposing the circuit to a fixed input voltage, then switching off the transmission gate and measuring the output. This process was repeated over a range of voltages, for each chip. Fig. 10 represents the average charge injection, over all chips.

B. Decoder Results

Using a pair of arbitrary waveform generators to produce input samples and a synchronized clock signal, the chip can be tested in full-speed serial mode. A Matlab script is used to generate random information bits and encode them. The script then adds Gaussian noise samples to simulate the additive white Gaussian noise (AWGN) channel. The resulting samples are appropriately scaled so that they represent LLR values, thereby simulating the output of an idealized matched-filter receiver. The LLR samples are sent to the waveform generator via GPIB, where they are provided to the chip as a serial input stream. The chip's digital outputs are sampled by an oscilloscope and returned to the Matlab script, which counts the errors.

The decoder's maximum throughput (the number of decoded bits per second) depends on the bias current I_U. SPICE simulations give an indication of the allowable operating speed, based on the crossover time and the 90% rise time of the analog output



Transistor sizes (W/L):
$\beta 1 = 1.8\mu/.9\mu$
$\beta 2 = 3\mu/.9\mu$
$\beta 3 = 5.1\mu/2.1\mu$
$\beta 4 = 3.9\mu/2.1\mu$
$\beta 5 = 3\mu/.6\mu$
$\beta 6 = 1.8\mu/.6\mu$
$\beta 7 = 6\mu/.6\mu$

Fig. 11. Circuit for latched current comparator.

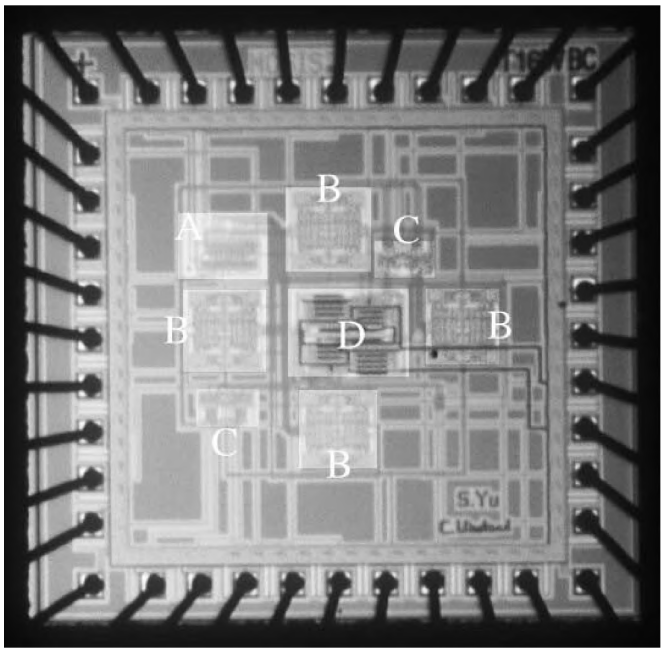


Fig. 12. Photograph of the decoder chip.

TABLE I
SUMMARY OF HAMMING DECODER CHARACTERISTICS

Die Size	1.5mm×1.5mm
Circuit Area	.81mm^2
Decoder Area	.083mm^2
Transistor Size	2μm × 4μm
Tested Speed	up to 2Mbps
Core Decoder Power	1mW at 1Mbps 16μW at 20kbps
Digital Power	44.2mW
Pad Power	135μW

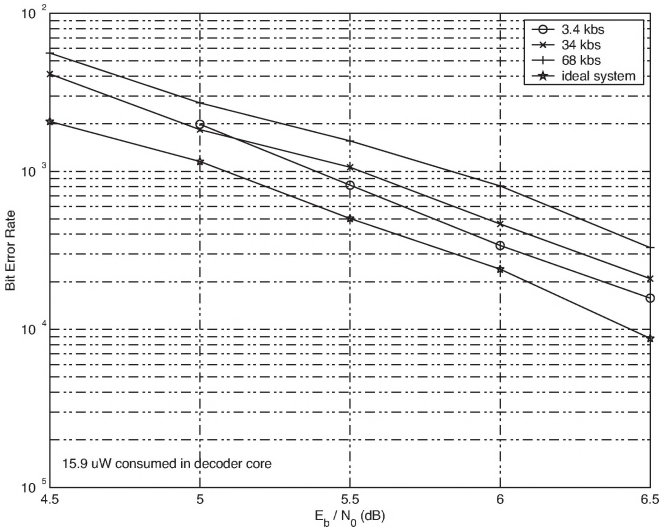


Fig. 13. Measured performance at different speeds in moderate inversion.

decisions. The crossover time is the instant at which the *sign* of a differential output has attained its final value. The rise time is the time it takes for the output to attain 90% of its final *magnitude*. Because the crossover and rise times may vary from sample to sample, we may only use them to roughly estimate the maximum speed. The chip should operate somewhere between the limits predicted by the rise and crossover times.

With our test setup, the variation of performance with speed is most conveniently observed when the decoder's speed is limited to between 1 and 10 kHz. At $I_U = 58$ nA, the 90% rise time for a transition in the output decisions is about 150 μs, corresponding to a speed of 27 kb/s. The output crossover time (the time after which the actual decision changes) is about 90 μs, for a speed of 44 kb/s, which should give some indication of the maximum possible speed. Performance results for this I_U at different speeds are shown in Fig. 13. The power consumed in the core decoder at this speed is 16 μW.

All reported BER measurements have a 95% confidence interval of better than $\pm 30\%$. Results are not available for the fourth output bit. A simple parallel/serial output mode-select circuit suffers from a floating node which should have been connected to ground. This mistake results in a stuck output on the fourth bit. The analog outputs of this bit are still measurable, but off-chip current comparators introduce additional problems such as glitches, phase shifts, and limited speed, which corrupt test results. The reported results therefore represent the three observable digital outputs.

Due to limitations in the oscilloscope, the serial-mode chip test can only measure performance at speeds above 1 kb/s. The

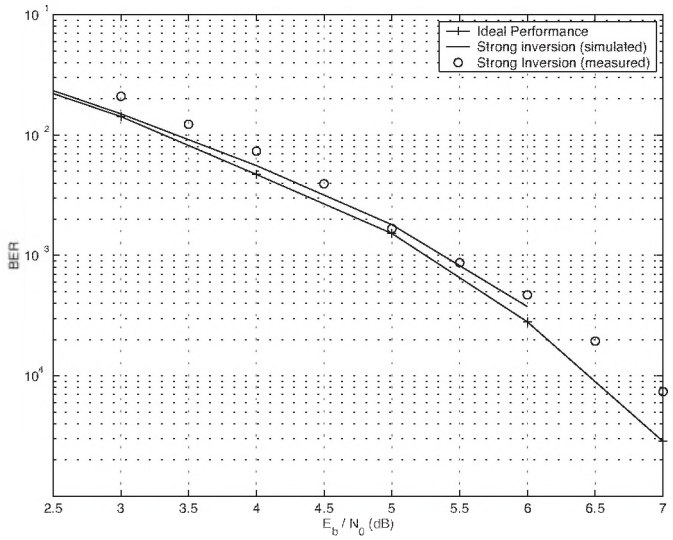


Fig. 14. Measured performance in strong inversion.

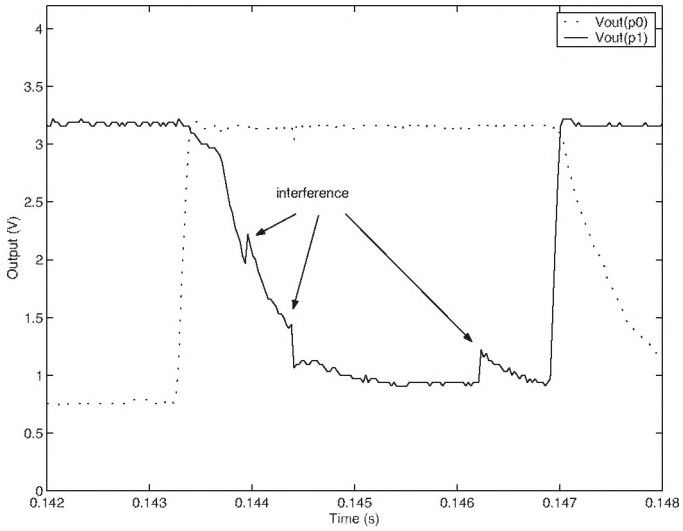


Fig. 15. Analog chip output showing interference from a digital signal.

code's small block length requires moderate-inversion biasing ($I_U > 14$ nA) to achieve testable speeds. While designed to operate in weak inversion, the chip also functions with strong inversion bias currents, and has been tested up to $I_U = 4$ μA, which is well into strong inversion.

Some performance loss occurs in strong inversion, as seen in the $I_U = 1.74$ μA measurement reported in Fig. 14. The test was conducted at a speed of 424 kb/s. The performance loss at this bias current agrees closely with simulations. The solid curve in Fig. 14 was obtained using a hybrid analog model implemented in VHDL. The model includes square-law transistor behavior and a one-pole system to model the dynamics of Gilbert multiplier circuits [11]. The close fit between simulated and measured points is taken to be a validation of this simulation model. Fig. 14 reports simulation results alongside performance of an "ideal" Hamming decoder. The distance between the ideal and measured curves is accounted for by moderate-inversion biasing. The performance loss of roughly 0.3 dB at BER $= 7 \times 10^{-5}$ matches the prediction made by high-level simulations.

The measured points of Fig. 13 represent the performance averaged over the three observable bit positions. A measurable amount of interference from on-chip digital circuitry occurs on the first bit position, due to the layout proximity between the analog outputs for those positions and the comparator latch signal. Fig. 15 shows the measured analog decoder output of a pair of output pins with interference. The two waveforms represent the probability values for an output bit.

The analog output wire labeled *p1* in Fig. 15 was found, upon examination of the layout, to be routed parallel to the *vlatch* signal wire, at minimum spacing, for a distance of 187 μm. The discontinuities in the interference pattern correspond precisely to the rising and falling edges of the *vlatch* signal. Interference from *vlatch* is visible on one of the other analog outputs, but it is comparatively faint. This amount of interference seems to result in a very small performance loss on the affected bit position, but the precise amount of loss is too small to be resolved by the current test method.

C. Discussion

It is of interest to compare our analog implementation with alternative digital designs. Much of the current interest in MAP decoders is motivated by their applicability to Turbo codes with large block lengths, so it is also important to explore the possibility of implementing analog BCJR circuits on a much larger scale. While the small block length of the Hamming decoder requires high bias currents (greater than 1 μA) to attain speeds above 1 Mb/s, a large-scale analog decoder is expected to attain high speeds with lower bias currents. Speed is therefore not expected to be a limitation of very large analog decoder designs.

A variety of synthesized digital decoders for a 3.3-V 0.5-μm process are presented in [15]. This fabrication process is very close to the AMI process used in our (8,4) Hamming decoder design. The study includes a hard-decision decoder for a (7, 4) Hamming code with an area of 0.055 mm^2. The (7, 4) Hamming code is comparable to the (8, 4) Hamming code in complexity, and a soft-decision decoder (e.g., a MAP decoder) such as ours is algorithmically much more complex than a hard-decision decoder. As noted in Table I, the area required by the (8, 4) analog Hamming decoder is 0.083 mm^2, only slightly larger than the hard-decision decoder of [15].

Data from [15] has been adapted to show how size varies with performance in Fig. 16 and how power varies with performance in Fig. 17. Measured data for our analog (8,4) Hamming decoder are included in these figures, along with the projected size and simulated power consumption of an analog (16,11)^2 Turbo product decoder with block length 231. The projection for the product decoder's size is derived from manually drawn layouts for a 0.18-μm process, adjusted for comparison with designs from the 0.5-μm process.

Figs. 16 and 17 indicate that large CMOS analog decoders may offer significant gains over conventional designs in both power and area. The analog decoders use more than an order of magnitude less power. It should also be noted that the analog decoder sizes include interface circuits. The data listed for digital decoders do not include the contribution of analog-to-digital converter (ADC) circuits, which can add significantly to

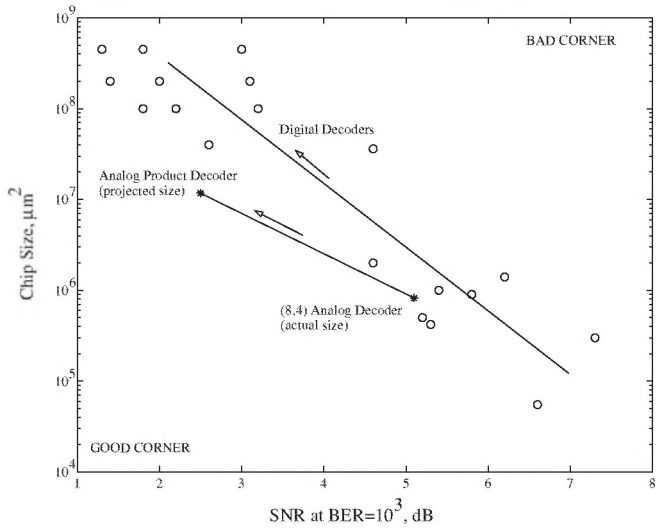


Fig. 16. Comparison of layout size versus performance for analog versus digital decoders.

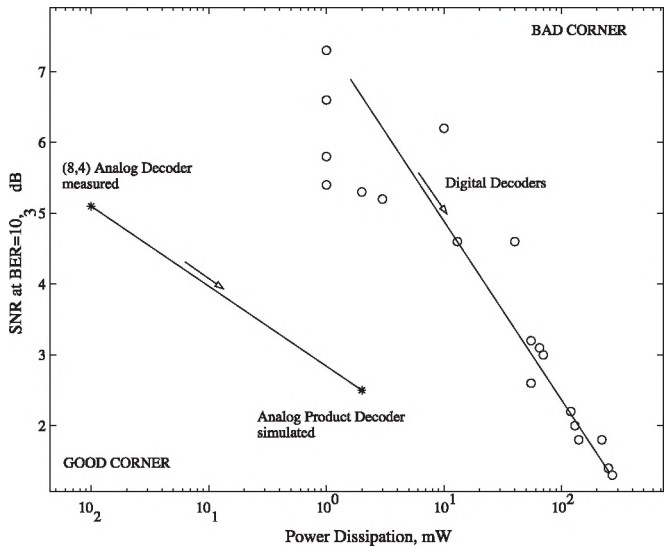


Fig. 17. Comparison of power versus performance for analog versus digital decoders.

their area and power consumption. An analog decoder can be thought of as a joint ADC/decoder circuit. CMOS analog Turbo-style decoders may therefore prove to be several times smaller than digital options and use many times less power, making them quite competitive for use in low-cost applications with low power budgets, such as portable wireless devices.

V. Conclusion

We designed an (8, 4) analog Hamming decoder with a goal of verifying the feasibility of CMOS translinear circuits for implementing MAP-style soft error-control decoders. The expected robustness of these circuits is confirmed: the Hamming decoder performs as expected under a wide range of bias currents, and in the presence of interference from digital circuits on the chip. Test results from the analog Hamming decoder also confirm the usefulness of a low-complexity analog VHDL model for analog computation. This model provides a valuable tool for efficient design and verification of analog MAP-style decoders. Our (8,4) Hamming code achieves an order of magnitude better power consumption than a comparable digital implementation, and consumes less silicon area. Similar gains are predicted for larger analog Turbo-style decoder implementations.

The (8,4) Hamming code demonstrates parallel analog decoding on a small scale. Powerful error-control codes, such as Turbo codes and low-density parity check codes, require decoding on very large graphs containing thousands of bits. Future work in analog decoders therefore targets very large parallel analog networks. An analog $(16,11)^2$ Turbo Product Decoder is being designed to study such large decoding networks. Other subjects of interest include the effect of device mismatch on the performance of large networks, the efficient synthesis of large analog designs, and the design of suitable interfaces between analog decoders and other receiver components.

References

[1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[2] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Communications*, vol. 2, Geneva, May 1993, pp. 1064–1070.

[4] S. Hong and W. Stark, "Decoding performance and complexity analysis for analog and digital channel decoders," in *Proc. IEEE Vehicular Technology Conf.*, May 2001, pp. 1277–1281.

[5] D. A. Johns and B. Zand, "High-speed CMOS analog Viterbi detector for 4-PAM partial-response signaling," *IEEE J. Solid-State Circuits*, vol. 37, pp. 895–903, July 2002.

[6] J. Hagenauer and M. Winklhofer, "The analog decoder," in *Proc. IEEE Int. Symp. Information Theory*, Aug. 1998, p. 145.

[7] H. A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog VLSI," *IEEE Trans. Inform. Theory*, vol. 47, pp. 837–843, Feb. 2001.

[8] M. Moerz, T. Gabara, R. Yan, and J. Hagenauer, "An analog 0.25 μm BiCMOS tailbiting MAP decoder," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2000, pp. 356–357.

[9] F. Lustenberger, M. Helfenstein, G. S. Moschytz, H. A. Loeliger, and F. Tarkoy, "All analog decoder for (18,9,5) tail-biting trellis code," in *Proc. Eur. Solid-State Circuits Conf. (ESSCIRC)*, Sept. 1999, pp. 362–365.

[10] F. Lustenberger, "On the design of analog VLSI iterative decoders," Ph.D. dissertation, Swiss Federal Inst. Technol., Lausanne, 2000.

[11] J. Dai, "Design methodology for analog VLSI implementations of error control decoders," Ph.D. dissertation , Univ. Utah, Salt Lake City, 2001.

[12] A. R. Calderbank, G. D. Forney, and A. Vardy, "Minimal tail-biting trellises: The Golay code and more," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1435–1455, July 1999.

[13] J. B. Anderson and S. M. Hladik, "Tailbiting MAP decoders," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 297–302, Feb. 1998.

[14] T. Serrano-Gotarredona, B. Linares-Barranco, and A. G. Andreou, "A general translinear principle fo subthreshold MOS transistors," *IEEE Trans. Circuits Syst. I*, vol. 46, pp. 607–616, May 1999.

[15] A. Worthen, S. Hong, R. Gupta, and W. Stark, "Performance optimization of VLSI transceivers for low-energy communications systems," in *Proc. Military Communications Conf.*, Nov. 1999, pp. 1434–1438.

Chris Winstead (S'97) received the B.S.E.E. degree from the University of Utah. Salt Lake City. in 2000. He is currently working towards the Ph.D. degree at the University of Alberta. Edmonton. AB. Canada. studying VLSI and error-control coding.

His research interests include the theory of iterative error-control decoders. VLSI implementation of decoding algorithms. and information theory.

Mr. Winstead is a member of Tau Beta Pi.

Jie Dai (S'02–M'03) was born in China in December. 1973. He received the B.S. degree in electrical engineering from Wuhan University. China. in 1994. the M.S. degree in electrical engineering from Shanghai Jiao Tong University. China. in 1997. and the Ph.D. degree in electrical engineering from the University of Utah. Salt Lake City. in 2002.

He is currently with XGI Technology. Inc.. Santa Clara. CA. His research interests include low-power circuit design and error-control coding techniques.

Shuhuan Yu (S'01) received the B.S. degree in optical instrumentation and the M.S. degree in test and measurement from Zhejiang University. China. in 1993 and 1998. respectively. She is currently working towards the Ph.D. degree in electrical engineering at the University of Utah. Salt Lake City.

Chris Myers (S'91–M'96) received the B.S. degrees in electrical engineering and Chinese history from the California Institute of Technology. Pasadena. CA. in 1991 and the M.S.E.E. and Ph.D. degrees from Stanford University. Stanford. CA. in 1993 and 1995. respectively.

He is currently an Associate Professor in the Department of Electrical and Computer Engineering. University of Utah. Salt Lake City. He is the author of over 50 technical papers and the textbook *Asynchronous Circuit Design* (New York: Wiley. 2001). He is also a co-inventor on four patents. His current research interests are algorithms for the computer-aided analysis and design of real-time concurrent systems. analog error-control decoders. formal verification. asynchronous circuit design. and modeling of biological networks.

Dr. Myers received a National Science Foundation (NSF) Fellowship in 1991. an NSF CAREER Award in 1996. and a Best Paper Award at the Async'99 conference.

Reid R. Harrison (S'98–M'00) received the B.S. degree in electrical engineering from the University of Florida. Gainesville. in 1994. and the Ph.D. degree from the California Institute of Technology. Pasadena. CA. in 2000.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering. University of Utah. Salt Lake City. where he holds an adjunct appointment in the Bioengineering Department. After working at the Jet Propulsion Laboratory and at Los Alamos National Laboratory for a brief time. he joined the Computation and Neural Systems program at the California Institute of Technology. Pasadena. CA. where he received the Ph.D. degree. His research interests include low-power analog and mixed-signal CMOS circuit design. biomedical electronics for neural interfaces. and hardware for biologically inspired vision systems.

Dr. Harrison organized the 2001 IEEE SSCTC Workshop on Low-Power Circuits. Arlington. VA. and received the National Science Foundation Career Award in 2002.

Christian Schlegel (S'86–M'88–SM'97) received the Dipl. El. Ing. ETH degree from the Swiss Federal Institute of Technology. Zürich. in 1984. and the M.S. and Ph.D. degrees in electrical engineering from the University of Notre Dame. Notre Dame. IN. in 1986 and 1989. respectively.

He held a research appointment with Asea Brown Boveri. Ltd.. Baden. Switzerland. from 1988 to 1992. and academic positions at the University of South Australia. Adelaide. the University of Texas at San Antonio. and the University of Utah. Salt Lake City. In 2001. he was named iCORE Professor for High-Capacity Digital Communications at the University of Alberta. Edmonton. Canada. He is the author of the research monographs *Trellis Coding* (New York: IEEE Press. 1997). and *Trellis and Turbo Coding* (New York: Wiley/IEEE. 2003). He is currently working on a new book entitled *Coordinated Multiple User Communications*.

Dr. Schlegel received a Natinal Science Foundation Career Award in 1997 and a Canada Research Chair in 2001. He is currently Associate Editor for coding theory and techniques for the IEEE TRANSACTIONS ON COMMUNICATIONS. He served as the Technical Co-chair of the IEEE Information Theory Workshop 2001. Cairns. Australia. and serves as Technical Program Chair of the International Symposium on Information Theory (ISIT'05) 2005 and as General Chair of the IEEE Communications Theory workshop 2005.