Copyright Notice

Let us now check that $S, T,$ and Y satisfy the conditions of Lemma 5.

Observation (4) implies $|X| \geq |S|d_0/2$. Together with (3) and since $N = \Delta n$, this implies that $|S|$ satisfies (2). Point 1) of Lemma 5 holds by the definition of S. Point 2) of Lemma 5 holds by (5).

Therefore, the conclusion of Lemma 5 holds and we have

$$|T| \leq \beta |S|$$

with $\beta = 1/(2-\alpha) < 1$.

The proof of convergence consists of repeating the argument. For $i \geq 1$ let $X^{(i)}$ be the set of edges in error after decoding step i. For $i \geq 0$, let $S^{(i)}$ be the set of vertices defined as

- $S^{(i)} = \{v \in A,\ E_v \cap X^{(i+1)} \neq \emptyset\}$ if i is even
- $S^{(i)} = \{v \in B,\ E_v \cap X^{(i+1)} \neq \emptyset\}$ if i is odd.

We have just proved that $|S^{(1)}| \leq \beta |S|$. Therefore, $S^{(1)}$ also satisfies (2) and we have $|S^{(2)}| \leq \beta |S^{(1)}|$ and more generally $|S^{(i)}| \leq \beta^i |S|$. When $S^{(i)} = \emptyset$ then $X^{(i+1)} = \emptyset$ and the decoding is complete. □

Remark: The above proof consisted of showing that the sets $S^{(i)}$ have strictly decreasing cardinalities. The weight of the error vector, however, does not necessarily decrease at each iteration.

Theorem 2 is a direct consequence of Theorem 6.

IV. A Proof of Lemma 4

The proof is very much in the spirit of [6].

Let $\boldsymbol{A} = (\boldsymbol{a_{ij}})$ be the $2n \times 2n$ adjacency matrix of the bipartite graph G, i.e., $a_{ij} = 1$ if the vertex indexed by i is adjacent to the vertex indexed by j and $a_{ij} = 0$ otherwise; a fixed ordering of the vertices is assumed but does not influence the computations to come. Let $\boldsymbol{X_{ST}}$ be the column vector of length $2n$ such that every coordinate indexed by a vertex of S or of T equals 1 and the other coordinates equal 0. It is straightforward to check that

$$^t\boldsymbol{X_{ST}} \boldsymbol{A} \boldsymbol{X_{ST}} = \sum_{\boldsymbol{v} \in \boldsymbol{S} \cup \boldsymbol{T}} \boldsymbol{d_{G_{S \cup T}}}(\boldsymbol{v}) \tag{6}$$

where $d_{G_{S\cup T}}(v)$ stands for the degree of v in the subgraph induced by $S \cup T$, i.e., the number of neighbors of v that belong to S or to T.

Now let $\boldsymbol{j}$ be the all-one vector and let $\boldsymbol{k}$ be the vector such that every coordinate indexed by a vertex of A equals 1 and every coordinate indexed by a vertex of B equals -1. $\boldsymbol{j}$ and $\boldsymbol{k}$ are eigenvectors of $\boldsymbol{A}$ associated to the eigenvalues Δ and $-\Delta$, respectively. Next define $\boldsymbol{Y_{ST}}$ as the vector such that

$$\boldsymbol{X_{ST}} = \frac{|\boldsymbol{S}| + |\boldsymbol{T}|}{\boldsymbol{2n}} \boldsymbol{j} + \frac{|\boldsymbol{S}| - |\boldsymbol{T}|}{\boldsymbol{2n}} \boldsymbol{k} + \boldsymbol{Y_{ST}}.$$

It is straightforward to check that $\boldsymbol{Y_{ST}}$ is orthogonal to $\boldsymbol{j}$ and $\boldsymbol{k}$. Because the eigenspaces of $\boldsymbol{A}$ are orthogonal we can therefore write

$$^t\boldsymbol{X_{ST}} \boldsymbol{A} \boldsymbol{X_{ST}} = \left(\frac{|S| + |T|}{2n}\right)^2 \Delta \boldsymbol{j} \cdot \boldsymbol{j} - \left(\frac{|\boldsymbol{S}| - |\boldsymbol{T}|}{\boldsymbol{2n}}\right)^2 \Delta \boldsymbol{k} \cdot \boldsymbol{k} + {}^t\boldsymbol{Y_{ST}} \boldsymbol{A} \boldsymbol{Y_{ST}}$$

which reduces to, since $\boldsymbol{j} \cdot \boldsymbol{j} = \boldsymbol{k} \cdot \boldsymbol{k} = \boldsymbol{2n}$,

$$^t\boldsymbol{X_{ST}} \boldsymbol{A} \boldsymbol{X_{ST}} = {}^t\boldsymbol{Y_{ST}} \boldsymbol{A} \boldsymbol{Y_{ST}} + \boldsymbol{2} \frac{|\boldsymbol{S}||\boldsymbol{T}|}{\boldsymbol{n}} \Delta.$$

Now, since $\boldsymbol{Y_{ST}}$ is orthogonal to $\boldsymbol{j}$ and since the eigenspace associated to the eigenvalue Δ is of dimension one (G is connected) we have $^t\boldsymbol{Y_{ST}} \boldsymbol{A} \boldsymbol{Y_{ST}} \leq \lambda \|\boldsymbol{Y_{ST}}\|^2$. Together with (6) we obtain therefore

$$(|S| + |T|)\overline{d}_{ST} \leq \lambda \|\boldsymbol{Y_{ST}}\|^2 + \boldsymbol{2} \frac{|\boldsymbol{S}||\boldsymbol{T}|}{\boldsymbol{n}} \Delta \tag{7}$$

where $\overline{d}_{ST}$ is the average degree in the induced subgraph $G_{S\cup T}$. There remains the computation of $\|\boldsymbol{Y_{ST}}\|^2$. Note that $\boldsymbol{Y_{ST}}$ has $|S|$ coordinates equal to $1 - |S|/n$, $|T|$ coordinates equal to $1 - |T|/n$, $n - |S|$ coordinates equal to $-|S|/n$, and $n - |T|$ coordinates equal to $-|T|/n$. After some rearranging we obtain

$$\|\boldsymbol{Y_{ST}}\|^2 = |\boldsymbol{S}| + |\boldsymbol{T}| - \frac{|\boldsymbol{S}|^2 + |\boldsymbol{T}|^2}{\boldsymbol{n}}.$$

Together with (7) this yields Lemma 4.

References

[1] M. Sipser and D. A. Spielman, "Expander Codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710–1722, Nov. 1996.

[2] M. Tanner, "A recursive approach to Low-complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.

[3] A. Lubotsky, R. Philips, and P. Sarnak, "Ramanujan graphs," *Combinatorica*, vol. 8, no. 3, pp. 261–277, 1988.

[4] G. A. Margulis, "Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators," *Probl. Inform. Transm.*, vol. 24, no. 1, pp. 39–46, 1988.

[5] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: Elsevier, 1977.

[6] N. Alon and F. R. K. Chung, "Explicit construction of linear sized tolerant networks," *Discr. Math.*, vol. 72, pp. 15–19, 1988.

Probability Propagation and Decoding in Analog VLSI

Hans-Andrea Loeliger, *Member, IEEE*,
Felix Lustenberger, *Student Member, IEEE*, Markus Helfenstein, and
Felix Tarköy, *Member, IEEE*

***Abstract*—The sum-product algorithm (belief/probability propagation) can be naturally mapped into analog transistor circuits. These circuits enable the construction of analog-VLSI decoders for turbo codes, low-density parity-check codes, and similar codes.**

***Index Terms*—Analog circuits, belief propagation, factor graphs, iterative decoding, turbo codes.**

I. Introduction

It has recently been observed that a number of important algorithms in error-control coding, signal processing, and computer science can be interpreted as instances of a general "sum-product algorithm" which operates by message passing on a graph (the factor graph [1], [2]; see also Aji and McEliece [3]). These algorithms include the forward–backward [Bahl–Cocke–Jelinek–Raviv (BCJR)] algorithm

Manuscript received December 16, 1999; revised September 9, 2000. The work of H.-A. Loeliger was supported by the Swiss National Science Foundation under Grant 21-49619.96. The material in this paper was presented in part at the 1998 IEEE International Symposium on Information Theory, Cambridge, MA, August 16–21, 1998, and at several other conferences.

H.-A. Loeliger was with Endora Tech AG, Basel, Switzerland. He is now with the Signal and Information Processing Laboratory (ISI), ETH Zentrum, CH-8092 Zürich, Switzerland.

F. Lustenberger is with the Signal and Information Processing Laboratory (ISI), ETH Zentrum, CH-8092 Zürich, Switzerland.

M. Helfenstein was with ISI/ETH Zurich, Switzerland. He is now with Globespan Semiconductor Inc., Red Bank, NJ 07701 USA.

F. Tarköy is with Endora Tech AG, CH-4051 Basel, Switzerland.

Communicated by R. Koetter, Guest Editor.

Publisher Item Identifier S 0018-9448(01)00718-0.

[4], Kalman filtering, "belief propagation" in Bayesian networks [5], and, in particular, iterative decoding of turbo codes [6], low-density parity-check codes ("Gallager codes") [7], [8], and similar codes [9], [10]. In all these mentioned cases (but not in *all* cases), the sum-product algorithm works with signals ("messages") that are, or may be viewed as (exact or approximate descriptions of) probability distributions;[1] such instances of the sum-product algorithm will be referred to as *probability propagation*.[2]

In some important cases, which include iterative decoding, the sum-product algorithm operates with signals (messages) that represent probability distributions defined on *finite* sets and can be decomposed into elementary computations (defined in Section III) that we will alternatively refer to as *sum-product modules* or *probability gates*.

Digital implementations of sum-product modules require real-number arithmetic and are thus quite complex. The main point of this correspondence is that sum-product modules can be realized directly as simple analog transistor circuits. The entire family of sum-product modules is obtained as variations of a single simple circuit, which provides an amazing match between probability propagation and transistor physics.

With these circuits, any network of sum-product modules (probability gates) can be directly implemented in analog VLSI. This holds, in particular, for the iterative decoding of turbo codes and low-density parity-check codes. A main attraction of such analog decoders is that the iterations disappear: the decoder is just an asynchronous electrical network that stabilizes (usually) in a state that corresponds to the transmitted codeword.

We have found that such decoding networks are quite robust against the nonidealities of real VLSI and can outperform comparable digital implementations by two orders of magnitude in terms of speed (when designed for the same power consumption) or power consumption (when designed for the same speed). A more detailed account of these issues will be given elsewhere.

Prior related work includes the analog Viterbi decoders of [11]–[16] as well as the "diode decoder" [17], [18] (see also [19]). The work by Wiberg *et al.* [9], [20] on Tanner graphs and the sum-product algorithm was motivated in part by speculations on analog decoding networks (cf. [21]). Hagenauer *et al.* were the first to simulate continous-time versions of the sum-product algorithm and to explicitly suggest analog implementations, without, however, proposing transistor level circuits [22] (see also [23]). The circuits of this paper were first presented in [24] and some more details were given in [25] and [26]. A summary of this research was given in [27]. Recently, circuits similar to those of this paper were also proposed by Moerz *et al.* [28], [29].

This correspondence is structured as follows. An introductory example of a sum-product module and its circuit realization is given in Section II. Sum-product modules are defined in Section III. The transistor circuits are presented in Section IV, and some comments and conclusions are given in Section V.

II. Introductory Example: The Soft Exclusive–OR Gate

Let X and Y be two independent binary random variables, and let $Z \triangleq X \oplus Y$, where "$\oplus$" denotes mod-2 addition (exclusive–OR). With the notation $p_X(b) \triangleq P(X = b)$ we have

$$\begin{bmatrix} p_Z(0) \\ p_Z(1) \end{bmatrix} = \begin{bmatrix} p_X(0)p_Y(0) + p_X(1)p_Y(1) \\ p_X(0)p_Y(1) + p_X(1)p_Y(0) \end{bmatrix}. \tag{1}$$

A computational unit that performs the computation (1) of p_Z from p_X and p_Y, for whatever representation of the probability distributions, will be referred to as a "soft exclusive–OR gate." (This is what Hagenauer [23] calls a "box-plus" element.)

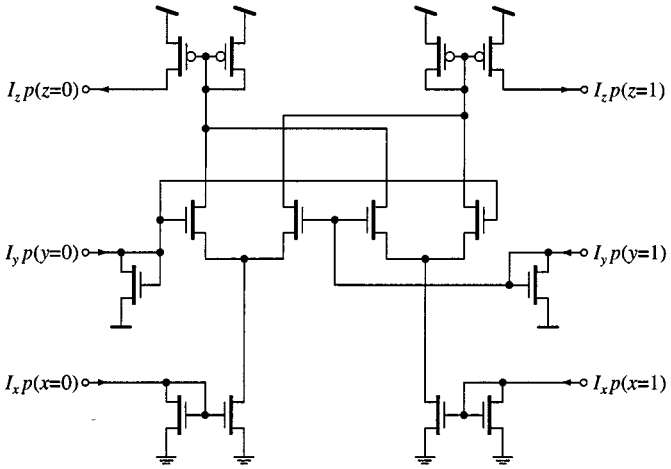

Fig. 1. Soft exclusive–OR circuit.

Clearly, the complexity of a digital implementation of such a soft XOR gate depends on the representation of the probability distributions. For example, any probability distribution p defined on $\{0, 1\}$ can be represented by the difference $p(0) - p(1)$. In this representation, the computation (1) becomes

$$p_Z(0) - p_Z(1) = (p_X(0) - p_X(1))(p_Y(0) - p_Y(1)). \tag{2}$$

Yet another representation of any such p is the "log-likelihood ratio" $L_p \triangleq \ln(p(0)/p(1))$. In this representation, (1) becomes

$$L_Z = 2 \tanh^{-1}(\tanh(L_X/2)\tanh(L_Y/2)). \tag{3}$$

Note that, in all these representations, the soft XOR gate requires at least one multiplication.

A circuit for the computation of (1) is shown in Fig. 1. The input probabilities $[p_X(0), p_X(1)]^T$ and $[p_Y(0), p_Y(1)]^T$ as well as the output probabilities $[p_Z(0), p_Z(1)]^T$ are represented by current vectors $[I_X p_X(0), I_X p_X(1)]^T$, $[I_Y p_Y(0), I_Y p_Y(1)]^T$, and $[I_Z p_Z(0), I_Z p_Z(1)]^T$, respectively, where the total input currents I_X and I_Y can be chosen freely in a large range.

The circuit of Fig. 1 is a version of the so-called *Gilbert multiplier*, which is a standard circuit for real-number multiplication that has been known for over 30 years [30]. The transistor pairs at the four corners of the circuit are current mirrors, which make such modules freely cascadable; the actual computation is done by the six transistors in the middle row. The full description of this circuit is given in Section IV.

It is noteworthy that all three formulations (1)–(3) are closely related to this circuit. While (1) corresponds arguably most directly to the circuit (see Section IV), we shall see that the log-likelihood ratios L_X, L_Y, and L_Z of (3) appear also in the circuit: L_X and L_Y appear as the voltage between the corresponding input terminals and L_Z appears as the voltage between the gates of the output transistors. Equations (2) and (3) correspond to two different classical ways of using the circuit as a real-number multiplier.

III. Sum-Product Modules

A sum-product module (probability gate) is a computational unit as shown in Fig. 2 that performs a computation of the following type. As input, it takes two probability mass functions p_X and p_Y defined on $\mathcal{X} \triangleq \{x_1, \ldots, x_m\}$ and $\mathcal{Y} \triangleq \{y_1, \ldots, y_n\}$, respectively;

[1]We use the terms *probability mass function* and *probability distribution* synonymously.

[2]This could also be called "belief propagation," although the underlying factor graph need not represent a Bayesian network.

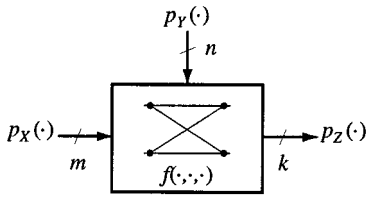

Fig. 2. General sum-product module.

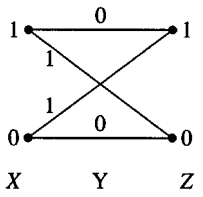

Fig. 3. Trellis diagram for $f(x, y, z) = 1$ iff $x \oplus y = z$ (Example 1).

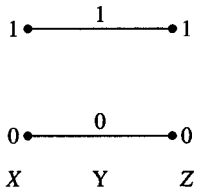

Fig. 4. Trellis diagram for $f(x, y, z) = 1$ iff $x = y = z$ (Example 2).

as output, it computes a probability mass function p_Z, defined on $\mathcal{Z} \triangleq \{z_1, \ldots, z_k\}$, according to

$$p_Z(z) = \gamma \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_X(x) p_Y(y) f(x, y, z) \qquad \forall\, z \in \mathcal{Z} \quad (4)$$

where f is a function $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \{0, 1\}$ and where γ is an appropriate scale factor that does not depend on z. Note that (4) defines a large family of modules that is parameterized by the function f.

If the reader is familiar with Tanner graphs or factor graphs [20], [2], he will notice that both the function-to-variable computation and the variable-to-function computation of the sum-product algorithm are of the form (4), provided that the local function f is $\{0, 1\}$-valued. In particular, both the forward recursion and the backward recursion of the BCJR algorithm [4] are of this form.

Specific $\{0, 1\}$-valued functions f are conveniently illustrated by trellis diagrams as in Figs. 3, 4, and 7. The left-hand nodes correspond to the elements of $\mathcal{X}$, the right-hand nodes correspond to the elements of $\mathcal{Z}$, and an edge between $x \in \mathcal{X}$ and $z \in \mathcal{Z}$ with label $y \in \mathcal{Y}$ exists if and only if $f(x, y, z) = 1$. Note that the trellis diagram uniquely defines f.

Example 1 (Soft XOR Gate): Let $\mathcal{X} = \mathcal{Y} = \mathcal{Z} = \{0, 1\}$ and let $f(x, y, z) = 1$ if $x \oplus y = z$ (where "$\oplus$" denotes mod-2 addition) and $f(x, y, z) = 0$ else. The corresponding trellis diagram is shown in Fig. 3. With this function f, the module of Fig. 2 becomes a soft exclusive-OR gate as in Section II. □

In generalization of Example 1, "soft" versions of all standard logic gates can be constructed by a suitable choice of f.

Example 2 (Componentwise Product): Let $\mathcal{X} = \mathcal{Y} = \mathcal{Z}$ and let $f(x, y, z) = 1$ if $x = y = z$ and $f(x, y, z) = 0$ else. The corresponding trellis diagram (for $\mathcal{X} = \mathcal{Y} = \mathcal{Z} = \{0, 1\}$) is shown in Fig. 4. The computation (4) then reduces to the componentwise product of p_X and p_Y. Such computations appear in nearly all applications because they arise whenever independent information (p_X and p_Y) about the same random variable is combined. □

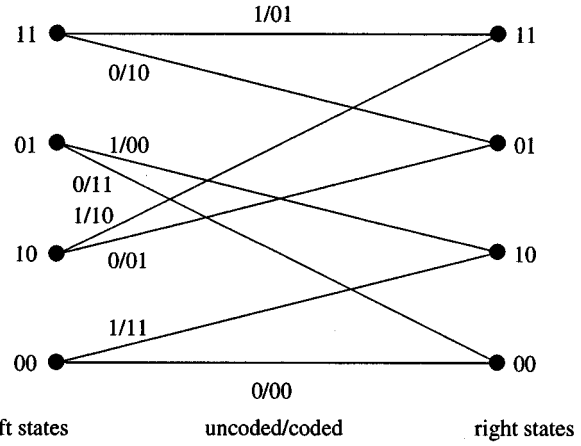

Fig. 5. One trellis section of a 4-state tail-biting convolutional code.

The specific modules from Example 1 and 2 actually suffice for the decoding of low-density parity-check codes. The general form (4), with general finite alphabets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, and with an arbitrary $\{0, 1\}$-valued function f, suffices for iterative decoding in general.

We conclude this section with an example of a complete decoding network for a tail-biting convolutional code. This decoder was actually implemented as an analog VLSI chip using the circuits of this correspondence [25], [26]. (A similar analog decoder for a two-state tail-biting code was implemented by Moerz *et al.* [28].) The code is defined by the four-state tail-biting trellis consisting of nine identical trellis sections, each trellis section as shown in Fig. 5. In the figure, each trellis branch is labeled both with the corresponding input bit (information bit) and with the two corresponding encoded bits (output bits). The code is a linear block code of length 18 and dimension 9.

A decoding network for this code is given in Fig. 6. Each signal line in Fig. 6 represents a whole probability mass function. This decoding network is a direct implementation of the BCJR forward–backward algorithm [4] (adapted to a tail-biting trellis), which is a special case of the general sum-product algorithm. Each module in Fig. 6 is a sum-product module of the form (4). The trellis diagrams of (the function f corresponding to) these modules are shown in Fig. 7. The type-B modules perform the "forward" computation on the tail-biting trellis, the type-C modules perform the "backward" computation on the trellis, the type-A modules precompute the branch metrics, and the type-D modules compute the final output.

Assuming, for the convenience of notation, a memoryless channel, the input to the decoder are the probability distributions

$$p_{\mathrm{in},\,i}(x_i) \triangleq \gamma p(y_i|x_i), \qquad i = 1, \ldots, 18$$

where $p(y_i|x_i)$ is the channel law, where $y_1, \ldots, y_{18}$ are the (fixed) received channel output symbols, and where γ is the scaling factor required to satisfy $p_{\mathrm{in},\,i}(0) + p_{\mathrm{in},\,i}(1) = 1$. We shall see in the next section that such scaling factors are implicit in the physical representation of probability distributions and need not be computed explicitly. Assuming a uniform *a priori* distribution over all codewords $(x_1, \ldots, x_{18})$, the output of the decoder consists of approximate *a posteriori* probability distributions $\tilde{p}_i(u_i) \approx p(u_i|y_1, \ldots, y_{18})$, for the information bits $u_i, i = 1, \ldots, 9$. The output probability distributions are not exact *a posteriori* probability distributions because the decoding network has cycles [2].

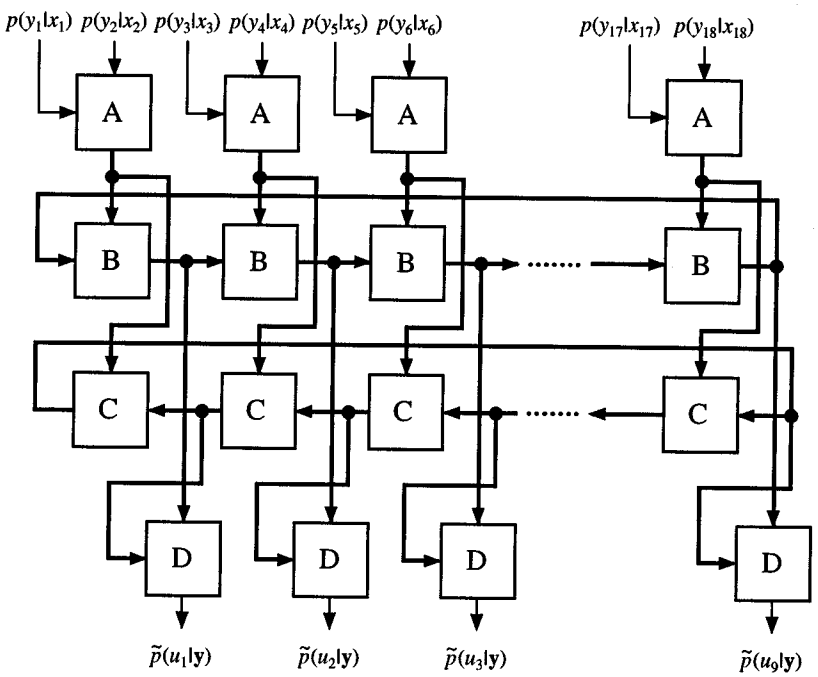

Fig. 6. Decoding network for a tail-biting convolutional code.

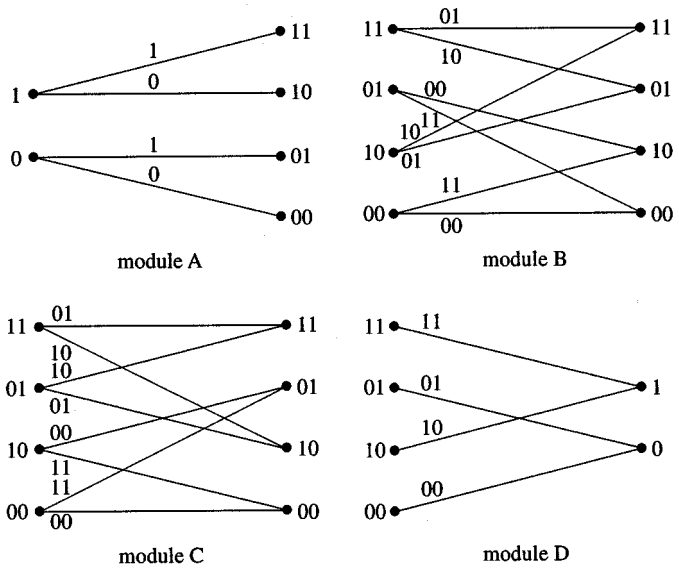

Fig. 7. Trellis diagrams for (the function f of) the sum-product modules in Fig. 6.

IV. Transistors

We now turn to the realization of probability propagation networks as analog transistor circuits. In such analog networks, probabilities will usually be represented as currents (and voltages will represent log-likelihood ratios). A probability mass function $p_{\mathcal{X}}$ defined on some finite set $\mathcal{X} \triangleq \{x_1, \ldots, x_m\}$ is represented by a current vector $(I_x p_{\mathcal{X}}(x_1), \ldots, I_x p_{\mathcal{X}}(x_m))$ with an arbitrary positive sum current I_x. Conversely, any current vector $(I_1, \ldots, I_n)$ with nonnegative components, not all of them zero, may be thought of as representing a probability mass function $p_{\mathcal{Y}}$ defined on some set $\mathcal{Y} \triangleq \{y_1, \ldots, y_n\}$ with values $p_{\mathcal{Y}}(y_i) \triangleq I_i/(I_1 + \cdots + I_n)$.

The fundamental circuit that underlies the realization of the modules of Section III is shown in Fig. 8. Its inputs are the currents $I_{x,i}$, $i = 1, 2, \ldots, m$ and the currents $I_{y,j}$, $j = 1, 2, \ldots, n$; its outputs are the currents $I_{i,j}$. All transistors are modeled as ideal voltage-controlled current sources, for which the current I_{drain} into the drain terminal depends exponentially on the voltage $V_{\mathrm{gate}} - V_{\mathrm{source}}$ between the gate and source terminals according to

$$I_{\mathrm{drain}} = I_0 e^{(\alpha V_{\mathrm{gate}} - \beta V_{\mathrm{source}})/U_T}. \tag{5}$$

The parameters I_0, α, and β depend on the fabrication process and on the temperature; U_T is the so-called thermal voltage, which depends only on the temperature. For $\alpha \neq \beta$, the transistor is effectively a four-terminal device rather than a three-terminal device; the extra (hidden) terminal is the "bulk" or "substrate," which serves as

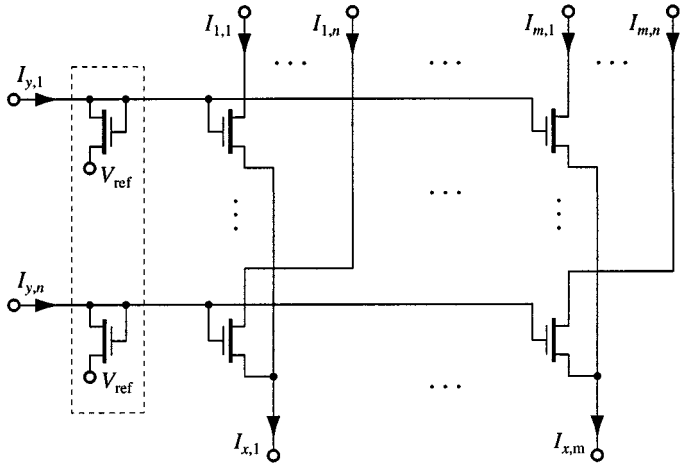

Fig. 8. Fundamental circuit.

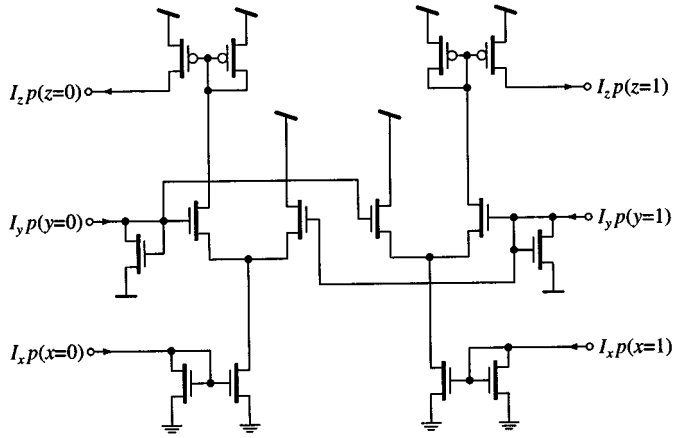

Fig. 9. Circuit of Example 2.

the reference for all potentials. Equation (5) is an excellent approximation both for CMOS transistors in the so-called subthreshold mode and for bipolar transistors. Our terminology and notation correspond to the former. For bipolar transistors, "drain" is replaced by "collector," "gate" by "base," "source" by "emitter," and we have $\alpha = \beta \approx 1$. It is shown in the appendix that the behavior of the circuit is given by

$$I_{i,j} = I_z(I_{x,i}/I_x)(I_{y,j}/I_y) \tag{6}$$

with $I_x \triangleq \sum_i I_{x,i}$, $I_y \triangleq \sum_j I_{y,j}$, and $I_z \triangleq \sum_i \sum_j I_{i,j} = I_x$. The circuit thus computes the pairwise product of the two probability mass functions $\tilde{p}_X(i) \triangleq I_{x,i}/I_x$ and $\tilde{p}_Y(j) \triangleq I_{y,j}/I_y$.

The application of the circuit of Fig. 8 to the computation of (4) is straightforward. The input terminals of the circuit are fed with the currents $I_{x,i} \triangleq I_x p_X(x_i)$, $i = 1, \ldots, m$, and $I_{y,j} \triangleq I_y p_Y(y_j)$, $j = 1, \ldots, n$, respectively, where the sum currents I_x and I_y can be chosen freely within the range of validity of (5). The output currents then equal $I_{i,j} = I_z p_X(x_i) p_Y(y_j)$.

The computation of (4) is completed by summing the currents $I_{i,j}$ for each $z \in \mathcal{Z}$ for which $f(x_i, y_j, z) = 1$. This is easily accomplished by connecting wires together and relying on Kirchhoff's current law. If a term $p_X(x_i)p_Y(y_j)$ is used more than once, the corresponding current $I_{i,j}$ must first be replicated by current mirrors.

Example 1 (Continued): For f as in Example 1, the circuit of Fig. 1 results. Note that the structure of the trellis diagram (Fig. 3) is evident in the topology of the circuit. □

Example 2 (Continued): For f as in Example 2, a complete circuit (for the binary case) is given in Fig. 9. In contrast to Example 1, the currents $I_{i,j}$ for $i \neq j$ are not used in the summation. □

The circuits for other sum-product modules (e.g., those of Fig. 6 and Fig. 7) are easily constructed according to the same pattern.

In the circuit of Fig. 1, the total output current I_z equals the input current I_x. This property is not shared by most other circuit modules as is exemplified by Fig. 9. Therefore, in general, it will often be necessary to scale the current vectors to some desired level. Such a scaling may be achieved by a degenerate version of the fundamental circuit of Fig. 8 with $m = 1$ and $I_{x,1}$ fixed to some constant current. This circuit is known as Gilbert's vector scaling circuit [31]. By adding current mirrors at the outputs, this circuit can be expanded into a building block of its own. Alternatively, it can also be integrated into the other building blocks.

As we have seen, in the circuits of this correspondence, currents represent probabilities. It is easy to see that voltages represent logarithms of probabilities (or of probability ratios). For example, the transistors

within the dashed box in Fig. 8 convert the input current vector

$$(I_{y,1}, \ldots, I_{y,n}) = (I_y p_{\mathcal{Y}}(y_1), \ldots, I_y p_{\mathcal{Y}}(y_n))$$

into a voltage vector $(V_{y,1}, \ldots, V_{y,n})$ with voltages

$$V_{y,j} = \frac{1}{\alpha}(U_T \log(p_{\mathcal{Y}}(y_j)) + U_T \log(I_y/I_0) + \beta V_{\text{ref}}). \quad (7)$$

By omitting these transistors within the dashed box, the circuit can be converted into one with voltage inputs instead of current inputs. The voltage input vector $(V_{y,1}, \ldots, V_{y,n})$ is then given by

$$V_{y,j} = \frac{U_T}{\alpha} \log(p_{\mathcal{Y}}(y_j)) + V_{\text{offs}} \quad (8)$$

where V_{offs} can be chosen freely, corresponding to the free choice of the sum current I_y in (7), as long as all involved transistors operate within the validity of the exponential model (5). Similarly, the current mirrors in Figs. 1 and 9 may be viewed as logarithmic current-to-voltage converters followed by exponential voltage-to-current converters. (The current mirrors can also be realized by standard quadratic-law CMOS transistors, in which case they may be viewed as square-root current-to-voltage converters followed by quadratic voltage-to-current converters.) Indeed, Moerz *et al.* [28] prefer to describe their circuit modules—which use the circuit of Fig. 8 without the dashed box—in terms of voltages. It should be noted, however, that the scale factor U_T/α in (8) depends both on process parameters and on the temperature.

In our simulations and design, we have so far been focussing on bipolar CMOS (BiCMOS) technology and used bipolar transistors for the fundamental circuit. We have mostly been using minimal-size transistors. The precision of individual circuit modules is then limited mainly by transistor mismatch. For example, the accuracy of a current mirror is on the order of 5–10%, which translates into a corresponding error in the represented probabilities. With realistic voltage swings and transistor parameters, this accuracy corresponds to about 5–7 bit resolution of log-likelihood values, i.e., logarithms of probabilities. The dynamic range of the circuits is three to six decades in terms of currents/probabilities.

V. Discussion and Conclusion

We have described a new type of analog computing network that exhibits a natural match between probability theory and transistor physics. The elementary modules of which these networks are composed include probabilistic versions of all standard logic gates as well as more general nonbinary sum-product modules. The obvious application of such networks is to the decoding of error-correcting codes. However, any factor graph all of whose local functions (except those of degree one) are $\{0, 1\}$-valued can be mapped into such an analog network.

The main advantages of such networks over digital implementations are higher processing speed or lower power consumption or both. According to our (still limited) experience, this advantage can amount to two orders of magnitude.

While traditional analog design is plagued with precision problems (sensitivity to component variations, susceptibility to noise and power supply disturbances, temperature dependency, etc.), we found that such analog decoding networks are quite robust against the nonidealities of real transistors. While most of our actual experience is with bipolar transistors, we believe that subthreshold CMOS implementations can be made similarly robust. This robustness may be explained by noting that the input signals to the decoder are noisy anyway and thus need not be represented and processed with high precision. (It is well known that for digital iterative decoders the number of bits used to represent messages can be quite low.) The quantitative analysis of these effects is a challenging theoretical problem.

An interesting option for enhancing the robustness of the network against the stated imperfections is the use of redundant factor graphs, i.e., of redundant equations (parity checks) in the decoding network. Preliminary experience suggests that extremely robust networks can be obtained in this way. This opens the problem of systematical construction of such redundant code descriptions, which appears to be a new topic in coding theory.

In most of the potential application areas (in particular, iterative decoding), the computation network contains loops. It is in such applications that the full advantage over digital implementations appears because the computationally demanding iterations required in the digital case are subsumed by the natural settling behavior of the analog network.

A network with loops may not always converge to a stable state. However, this holds also for the corresponding iterative digital algorithm. Indeed, we have so far observed little difference between the convergence behavior of an analog decoding network and its digital (i.e., discrete-time) counterpart. For decoding applications, experience with (digital) iterative algorithms indicates that nonconvergence is not a serious problem for good decoder architectures.

Analog networks in the spirit of this correspondence appear generally suitable for the realization of a mapping from some high-dimensional analog input space into a finite output space ("bits") that is insensitive to small disturbances at the input. This opens the prospect that all-analog receivers—from the demodulator output to the error-correcting decoder—can be built with such networks. In particular, "soft" feedback loops between the various elements (equalizer, channel tracking, multiuser separation, decoder, etc.) would fit naturally into that approach. A first step in that direction was made in [23].

Appendix
Derivation of (6)

Let $V_{x,i}$ and $V_{y,j}$ denote the potentials at the input terminals for $I_{x,i}$ and $I_{y,j}$, respectively. On the one hand, we have

$$\frac{I_{i,j}}{I_{x,i}} = I_{i,j} \Bigg/ \sum_{\ell=1}^{n} I_{i,\ell} \quad (9)$$

$$= I_0 \exp \frac{\alpha V_{y,j} - \beta V_{x,i}}{U_T} \Bigg/ \sum_{\ell=1}^{n} I_0 \exp \frac{\alpha V_{y,\ell} - \beta V_{x,i}}{U_T} \quad (10)$$

$$= \exp \frac{\alpha V_{y,j}}{U_T} \Bigg/ \sum_{\ell=1}^{n} \exp \frac{\alpha V_{y,\ell}}{U_T}. \quad (11)$$

On the other hand, we have

$$\frac{I_{y,j}}{I_y} = I_{y,j} \Bigg/ \sum_{\ell=1}^{n} I_{y,\ell} \quad (12)$$

$$= I_0 \exp \frac{\alpha V_{y,j} - \beta V_{\text{ref}}}{U_T} \Bigg/ \sum_{\ell=1}^{n} I_0 \exp \frac{\alpha V_{y,\ell} - \beta V_{\text{ref}}}{U_T} \quad (13)$$

$$= \exp \frac{\alpha V_{y,j}}{U_T} \Bigg/ \sum_{\ell=1}^{n} \exp \frac{\alpha V_{y,\ell}}{U_T}. \quad (14)$$

Combining (11) and (14) and noting that $I_z = I_x$ yields (6).

Acknowledgment

The authors wish to thank Prof. J. L. Massey and Prof. G. S. Moschytz for encouragement and support.

REFERENCES

[1] B. J. Frey, F. R. Kschischang, H.-A. Loeliger, and N. Wiberg, "Factor graphs and algorithms," in *Proc. 35th Allerton Conf. Communications, Control, and Computing*. Monticello, IL, Sept. 29–Oct. 1, 1997, pp. 666–680.

[2] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[3] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, vol. 46, pp. 325–343, Mar. 2000.

[4] 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. IT-20, pp. 284–287, Mar. 1974.

[5] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 1988.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannonlimit error-correcting coding and decoding: Turbo codes," in *Proc. Int. Conf. Communications (ICC'93)*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[7] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[8] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[9] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecommun.*, vol. 6, pp. 513–525, Sept./Oct. 1995.

[10] G. D. Forney Jr., "On iterative decoding and the two-way algorithm," in *Proc. Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 1997.

[11] A. S. Acampora and R. P. Gilmore, "Analog Viterbi decoding for high speed digital satellite channels," *IEEE Trans. Commun.*, vol. COM-26, pp. 1463–1470, Oct. 1978.

[12] T. W. Mathews and R. R. Spencer, "An integrated analog CMOS Viterbi detector for digital magnetic recording," *IEEE J. Solid-State Circuits*, vol. 28, pp. 1294–1302, Dec. 1993.

[13] M. H. Shakiba, D. A. Johns, and K. W. Martin, "A 200 MHz 3.3 V BiCMOS class-IV partial-response analog Viterbi decoder," in *Proc. CICC '95*, Santa Clara, CA, May 1995, pp. 567–570.

[14] A. Demosthenous and J. Taylor, "Current-mode approaches to implementing hybrid analogue/digital Viterbi decoders," in *Proc. IEEE Int. Conf. Electronics, Circuits, and Systems*, 1996.

[15] X. Wang and S. B. Wicker, "An artificial neural net Viterbi decoder," *IEEE Trans. Commun.*, vol. 44, pp. 165–171, Feb. 1996.

[16] M. H. Shakiba, D. A. Johns, and K. W. Martin, "BiCMOS circuits for analog Viterbi decoders," *IEEE Trans. Circuits Syst. II*, vol. 45, pp. 1527–1537, Dec. 1998.

[17] R. C. Davis, "Diode-configured Viterbi algorithm error correcting decoder for convolutional codes," U.S. Patent 4 545 054, Oct. 1, 1985.

[18] R. C. Davis and H.-A. Loeliger, "A nonalgorithmic maximum likelihood decoder for trellis codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1450–1453, July 1993.

[19] L. Bu and T.-D. Chiueh, "Solving the shortest path problem using an analog network," *IEEE Trans. Circuits Syst. I*, vol. 46, pp. 1360–1363, Nov. 1999.

[20] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation 440, Linköping Studies in Science and Technology, Univ. Linköping, Linköping, Sweden, 1996.

[21] ——, "Approaches to neural-network decoding of error-correcting codes," Thesis 425, Linköping Studies in Science and Technology, Univ. Linköping, Linköping, Sweden, 1994.

[22] J. Hagenauer, "Decoding of binary codes with analog networks," in *Proc. 1998 Information Theory Workshop*, San Diego, CA, Feb. 8–11, 1998, pp. 13–14.

[23] J. Hagenauer, E. Offer, C. Méasson, and M. Mörz, "Decoding and equalization with analog nonlinear networks," *Europ. Trans. Telecommun.*, vol. 10, pp. 659–680, Nov.-Dec. 1999.

[24] H.-A. Loeliger, M. Helfenstein, F. Lustenberger, and F. Tarköy, "Probability propagation and decoding in analog VLSI," in *Proc. 1998 IEEE Int. Symp. Information Theory*, Cambridge, MA, Aug. 16–21, 1998, p. 146.

[25] F. Lustenberger, M. Helfenstein, H.-A. Loeliger, and F. Tarköy, "An analog VLSI decoding technique for digital codes," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, Orlando, FL, June 1999, pp. 424–427.

[26] F. Lustenberger, M. Helfenstein, H.-A. Loeliger, and F. Tarköy, "All-analog decoder for a binary (18, 9, 5) tail-biting trellis code," in *Proc. ESSIRC 1999*, Duisburg, Germany, Sept. 1999, pp. 362–365.

[27] H. A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarköy, "Decoding in analog VLSI," *IEEE Commun. Mag.*, pp. 99–101, Apr. 1999.

[28] M. Moerz, T. Gabara, R. Yan, and J. Hagenauer, "An analog 0.25 μ BiCMOS tailbiting MAP decoder," in *Proc. ISSCC 2000*, San Francisco, CA, Feb. 2000, pp. 356–357.

[29] M. Moerz, J. Hagenauer, and E. Offer, "On the analog implementation of the APP (BCJR) algorithm," in *Proc. 2000 IEEE Int. Symp. Information Theory*, Sorrento, Italy, June 25–30, 2000, p. 425.

[30] B. Gilbert, "A precise four-quadrant multiplier with subnanosecond response," *IEEE J. Solid-State Circuits*, vol. 3, pp. 365–373, 1968.

[31] ——, "A monolitic 16-channel analog array normalizer," *IEEE J. Solid-State Circuits*, vol. 19, pp. 956–963, 1984.

Unified Design of Iterative Receivers Using Factor Graphs

Andrew P. Worthen, *Student Member, IEEE,* and Wayne E. Stark, *Fellow, IEEE*

***Abstract*—Iterative algorithms are an attractive approach to approximating optimal, but high-complexity, joint channel estimation and decoding receivers for communication systems. We present a unified approach based on factor graphs for deriving iterative message-passing receiver algorithms for channel estimation and decoding. For many common channels, it is easy to find simple graphical models that lead directly to implementable algorithms. Canonical distributions provide a new, general framework for handling continuous variables. Example receiver designs for Rayleigh fading channels with block or Markov memory, and multipath fading channels with fixed unknown coefficients illustrate the effectiveness of our approach.**

***Index Terms*—Channel estimation, fading channels, iterative decoding, low-density parity-check (LDPC) codes.**

I. INTRODUCTION

It is well known that for many communication systems joint demodulation and decoding is required for optimum performance. Typically, this processing is too complex for practical implementation and some sort of serial processing is employed. Iterative algorithms which approximate optimal joint decoding for a variety of concatenated codes [1]–[3] are known to have excellent performance. This has led to an interest in iterative algorithms for approximating joint channel estimation, demodulation, and decoding, which we call *iterative receivers.* The usual paradigm for these designs is the interconnection of soft-input/soft-output (SISO) modules [4].

Graphical models for codes [5]–[7] lead to iterative algorithms for decoding, including the turbo decoding algorithm [8]. Factor graphs

Manuscript received December 16, 1999; revised July 10, 2000. This work was supported by the Department of Defense Research and Engineering (DDR&E) Multidisciplinary University Research Initiative (MURI) on "Low-Energy Electronics Design for Mobile Platforms" and managed by the Army Research Office (ARO) under Grant DAAH04-96-1-0377. The work of A. P. Worthen was supported by a National Science Foundation Graduate Research Fellowship.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: worthena@eecs.umich.edu; stark@eecs.umich.edu).

Communicated by B. J. Frey, Guest Editor.

Publisher Item Identifier S 0018-9448(01)00723-4.