Analog Decoding of Product Codes

Chris Winstead[1], Chris Myers,
Christian Schlegel, Reid Harrison
University of Utah[2]
e-mail: winstead@eng.utah.edu

Abstract — **A design approach is presented for soft-decision decoding of block product codes ("block turbo codes") using analog computation with MOS devices. Application of analog decoding to large code sizes is also considered with the introduction of serial analog interfaces and pipeline schedules.**

I. Introduction

Several authors have suggested use of analog circuits to implement probability propagation algorithms for soft-decision decoding [1][2], and some successful results have been presented [3][4][5]. Many soft decoding algorihtms have been shown to be special cases of probability propagation on graphs (e.g. the sum-product algorithm [6]).

The structure of an analog decoder reflects the structure of the code's graph, providing a straightforward path from code design to physical implementation. These design principles may be applied in the same way to product codes, allowing a natural transition from current analog decoder designs to more powerful block turbo codes.

II. Factor Graphs and Product Codes

A factor graph, as defined in [6] and [7], represents a function $g = f_1 \cdot f_2 \cdot f_3 \cdots f_N$ in terms of its factors f_1, f_2, ..., f_N. A factor graph contains function nodes, variable nodes, and edges. An edge exists between a function node f and a variable node x if and only if f depends on x. A-posteriori probabilities for variable nodes may be computed by propagating probability mass functions through the graph.

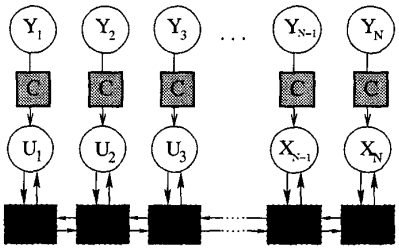

Figure 1: Factor graph for trellis decoder.

As an example, a factor graph for a typical trellis decoder is shown in Figure 1. The variable nodes are represented by circles. Channel observations, denoted in the graph by Y, are passed into the decoder via the channel model, which is represented by the box labeled 'C'. If the code is systematic, then some of the transmitted symbols will be information (denoted U) and some symbols will be parity (denoted X). Each solid black box is a function node, and represents a trellis section.

As indicated by the arrows in Figure 1, messages are passed throughout the network. Each directed edge transmits a probability mass function. A message passed on an edge to or from a binary variable x will thus be a vector containing two values, a conditional probability for $x = 0$ and a conditional probability for $x = 1$. The messages passed between trellis blocks will carry as many values as there are states in that trellis section, and so on. The algorithm which results from this message passing is equivalent to a BCJR decoder [8].

A block product code encodes the same information in two dimensions, as indicated in Figure 2. Systematic component block codes are used to separately encode the rows and the columns, thus adding separate blocks of row parity checks and column parity checks.

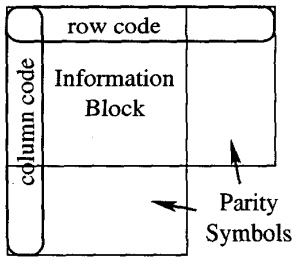

Figure 2: Product codeword structure.

Graphs for the row and column codes may be individually designed as trellis codes, or Tanner graphs, or some other systematic factor graph structure. Product codes have been considered in terms of iterative decoding in which information is exchanged between separate row and column component decoders [9]. A factor graph design would instead combine the component graphs into one large product-code graph. Application of the sum-product algorithm then provides concurrent decoding throughout a single graph.

The factor graph of a product code can be obtained by introducing the "equal gate," a function node which indicates equality among the variables to which it connects. Because the component codes share information symbols, they may be connected through equal gates at any node

[1]This work was supported by NSF grant CCR-9971168.
[2]EE Dept, 50 S. Central Campus Dr, Rm 3280 MEB, Salt Lake City, UT 84112-9206

which is common to the two decoders (parity nodes remain unaffected). This connection is illustrated in Figure 3. Channel observations are input to the decoder at variable node Y and passed through the equal gate to the component graphs. The overall decoder output arrives at the variable node U.

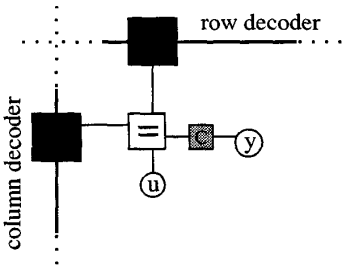

Figure 3: Connecting shared nodes through an equal gate.

Suitable designs have been implemented for analog BCJR-style component decoders [3][4][5]. Sum-product implementation of equal gates is simple, as discussed in [10]. The equal gate propagates probabilities on a given edge by transmitting along that edge the product of messages received on each other edge. Message passing in one direction through a binary equal gate is illustrated in Figure 4. Following the CMOS current-mode design method of [1], one direction of message passing is implemented by the analog circuit of Figure 5. A separate propagation circuit must be built for each edge connected to the equal gate.

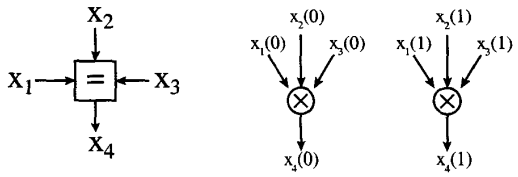

Figure 4: Message passing in one direction.

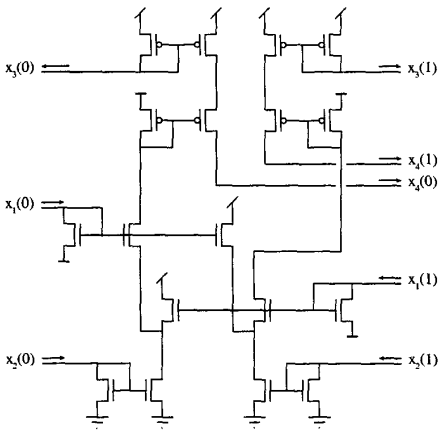

Figure 5: Analog implementation of equal gate in one direction.

The factor graph of Figure 3 illustrates the ease with which recent analog decoder designs may be used to build more substantial product decoders. The resulting analog product decoder replaces iterative decoding with continuous-time processing.

III. Interfacing for Larger Codes

Fully analog decoders have thus far only been implemented for small codes where inputs may be provided in parallel. A realistic design must accomodate serial channel information.

Serial-to-parallel conversion of analog data requires a sample-and-hold (S/H) circuit which can accurately hold an analog signal for the total time needed to decode. Leakage currents cause stored voltages to decay in ordinary S/H circuits. Storage duration may be improved by encoding signals differentially.

When used for probability propagation, analog circuits provide a natural duality for message encoding: currents may encode probabilities while differential voltages encode log-likelihood ratios in the same circuits. The circuit of Figure 6 uses two S/H cicruits to store one differential voltage. If the two storage cells are well-matched, then their stored voltages decay at roughly the same rate. The differential voltage remains unaffected. This circuit may be used as an effective buffer for log-likelihood ratios encoded as differential voltages.

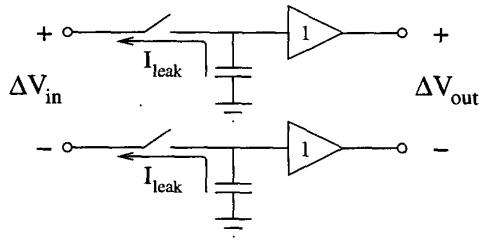

Figure 6: Differential S/H circuit.

Differential S/H circuits may be used to store a stream of analog signals which are then provided simultaneously, in parallel, to the decoder. A new block of inputs may then be collected while the decoder converges to a solution for the previous block (a "pipeline" schedule for analog data).

The signal stored in the differential S/H circuit must not fall below levels needed to bias the gates of MOS load transistors. Limitations must thus be placed on the amount of common-mode voltage decay which may be tolerated by a real circuit.

In the pipelined interface, soft channel information for a single bit is provided to an array of differential S/H circuits every T_s seconds. The information for each arriving bit is stored in the array until the entire block of length N is received. All of the stored analog values are then provided to an analog decoder in parallel.

We assume that a minimum-length transistor pass gate is used as the switch to the S/H circuit, and that the pass gate has resistance R_{on} when the switch is on. A leakage

current I_{leak} flows through the pass gate when the switch is off. To ensure that the capacitor C has adequate time to store its signal, we may place a restriction on the time-constant $\tau = R_{on}C = \frac{T_s}{20}$.

The total voltage loss ΔV in the time NT_s is given by

$$\Delta V = \frac{I_{leak}}{C} NT_s$$

The leakage current is proportional to transistor width: $I_{leak} = k_1 W$. The switch's on-resistance is inversely proportional to the width of the same transistor: $R_{on} = k_2 W^{-1}$.

ΔV may therefore be expressed as

$$\Delta V = \frac{k_1 W}{C} 20N\tau = 20Nk_1k_2$$

where k_1 and k_2 are constants of the process, and N is the number of bits we wish to store. This result only holds for achievable time-constants, which are limited by process design rules.

A conservative CMOS design should permit sampling in the range of 1 to 10 MHz, and have the capacity to store soft information for tens of thousands of bits with negligible common-mode voltage loss.

IV. Implementation

If trellis-style graphs are used to represent the row and column codes, a tilable analog module design emerges. Figure 7 shows a module which consists of two analog trellis decoder sections (the solid black boxes), an equal gate, a differential S/H, and an output comparator. The comparator converts the product decoder's output to a final digital value.

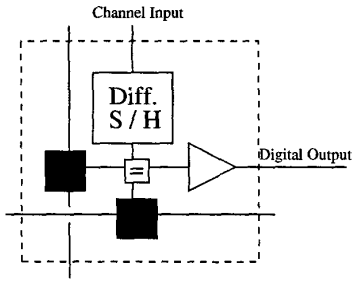

Figure 7: Tilable analog module for a product decoder.

The module of Figure 7 may be replicated and tiled to build the complete product decoder. The trellis decoder sections in each instance must be modified to reflect their code position. The result is a simple modular design, as depicted in Figure 8. The double-boxes represent systematic modules, and the single boxes represent parity sections from the original trellis decoders.

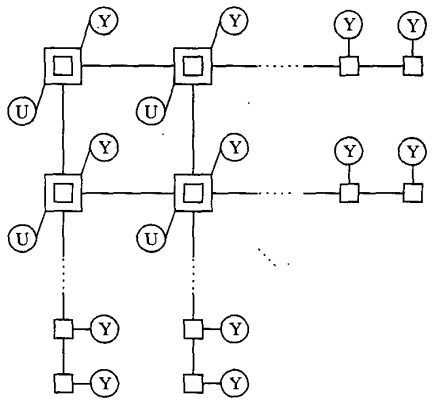

Figure 8: Modular product decoder design.

V. Conclusions

Analog decoding of product codes is conceptually a small step beyond previously implemented analog block decoders. Physically producing such a decoder requires mass storage of analog information within the decoder circuit. The differential sample-and-hold circuit is a promising solution to this problem. It has shown that, allowing basic design restrictions, the amount of analog information which can be stored in the circuit is a constant of the fabrication process. Current CMOS processes appear to be generous in this regard.

References

[1] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarköy, "Probability propagation and decoding in analog VLSI," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 837-843, February 2001.

[2] J. Hagenauer, M. Mörz, E. Offer, "A circuit-based interpretation of analog MAP decoding with binary trellises," *Proc. 3rd ITG Conference Source and Channel Coding, München*, pp. 175-180, January 2000.

[3] L. Lustenberger, M. Helfenstein, H.-A. Loeliger, F. Tarköy, and G. S. Moschytz, "All-Analog decoder for a binary (18,9,5) tailbiting trellis code," *Proc. European Solid-State Circuits Conference*, pp. 362-365, Duisburg, September 1999.

[4] M. Mörz, T. Gabara, R. Yan, and J. Hagenauer, "An analog .25μm BiCMOS tailbiting MAP decoder," *IEEE Proc. International Solid-State Circuits Conference*, pp. 356-357, San Francisco, February 2000.

[5] C. Winstead, J. Dai, W. J. Kim, S. Little, Y.-B. Kim, C. Myers, C. Schlegel, "Analog MAP decoder for (8, 4) Hamming code in subthreshold CMOS," *Advanced Research in VLSI Conference*, pp. 132-147, Salt Lake City, March 2001.

[6] F. Kschischang, B. Frey, H.-A. Loeliger, "Factor graphs and the Sum-Product Algorithm," *IEEE Trans. on Information Theory*, vol. 47, no.2, pp. 498-519, February 2001.

[7] G. D. Forney, "Codes on graphs: normal realizations," *IEEE Trans. on Information Theory*, vol. 47, no. 2, February 2001.

[8] 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, March 1974.

[9] J. Hagenauer, E. Offer, L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. on Information Theory*, vol. 42, no. 2, pp. 429-445, March, 1996.

[10] F. Lustenberger, *On the design of analog VLSI iterative decoders*, Ph.D. Thesis, Swiss Federal Institute of Technology, Zürich, 2000.