AN ANALOG TURBO DECODER FOR AN (8,4) PRODUCT CODE

Neiyer Correal and Joe Heck

Florida Communications Research Labs
Motorola Labs
Plantation, FL 33322
[N.Correal,Joe.Heck]@Motorola.com

Matthew Valenti

Lane Dept. of Comp. Sci. & Elect. Eng.
West Virginia University
Morgantown, WV 26506-6109
mvalenti@wvu.edu

ABSTRACT

This paper illustrates how analog circuitry can be used to decode turbo and turbo-like codes. For ease of exposition, the focus is on a simple two dimensional product code comprised of a 2 by 2 array of (3,2) single parity check codes. It is shown that the heart of the decoder is a soft exclusive-or operation which is easily implemented in analog as a modified Gilbert multiplier. Finally, a complete decoder design is presented which is capable of achieving throughput on the order of hundreds of Mbps.

1. INTRODUCTION

Iteratively decodable codes, such as turbo [1] and low density parity check (LDPC) [2] codes, are capable of performing within a decibel of the Shannon limit. These sophisticated codes are constructed by concatenating multiple simple codes. Rather than performing maximal-likelihood decoding of the entire code, each simple code is decoded individually. The result of each simple decoding is bit reliability information which is exchanged with the other decoders and used as side information during the next iteration. This iterative process of exchanging the results of multiple simple decodings is called *turbo decoding*.

Turbo decoders are typically implemented in digital hardware. However, digital hardware has its limitations. The clock rate sets an upper limit on achievable throughput and a large portion of the power consumed by the circuit is associated with clock functions [3]. Furthermore, turbo decoding algorithms have nonlinearities that are awkward and expensive to implement with digital logic but lend themselves naturally to analog circuitry. For these reasons and others, there has been a growing interest in analog decoding architectures [4–7]. It has been found that analog decoder implementations can operate at higher data rates, consume less power, and have a smaller footprint than their digital counterparts [6].

This paper presents an illustrative example of a turbo decoder implemented with analog circuitry. For ease of exposition, the example is a simple product code comprised of a 2 by 2 array of single parity check (SPC) codes. The remainder of this paper is organized as follows: Section 2 discusses SPC codes and a suitable a posteriori probability (APP) decoder, Section 3 presents the 2 by 2 product code built from the SPC and the corresponding turbo decoder, Section 4 discusses the implementation details of this decoder, and Section 5 gives some concluding remarks.

2. APP DECODING OF SPC CODES

A SPC encoder produces an output of length n which is formed by appending a single parity bit to the end of the $k = n - 1$ bit input. The parity bit is found by *xor*-ing all the input bits, and thus the code word will always have even parity. Such a code is capable of detecting a single error, but can also be used to correct errors when used as the constituent code of a product code. In this paper we focus on the simple (n,k) = (3,2) SPC. The two input bits are denoted u_j and u_k while the parity bit is denoted $u_{j,k} = u_j \oplus u_k$. The code word $\mathbf{u} = \{u_j, u_k, u_{j,k}\}$ is BPSK modulated according to $x = 1 - 2u$, i.e. $0 \rightarrow +1$ and $1 \rightarrow -1$. The modulated word $\mathbf{x} = \{x_j, x_k, x_{j,k}\}$ is transmitted over an arbitrary channel and the received word is denoted $\mathbf{y} = \{y_j, y_k, y_{j,k}\}$.

Rather than operating on $\mathbf{y}$ directly, the decoder requires that the observations first be normalized in an appropriate fashion. Define the *a priori* log-likelihood ratio (LLR) of the transmitted BPSK symbol to be:

$$L(x) = \ln \left(\frac{P[x = +1]}{P[x = -1]} \right), \quad (1)$$

where $P[.]$ denotes probability. Likewise, the conditional LLR of the received symbol is:

$$L(y|x) = \ln \left(\frac{f(y|x = +1)}{f(y|x = -1)} \right), \quad (2)$$

where $f(y|x)$ denotes the conditional pdf of y given that symbol x was transmitted. Then, the input to the decoder is $\mathbf{r} = \{L(y_j|x_j), L(y_k|x_k), L(y_{j,k}|x_{j,k}\} = \{r_j, r_k, r_{j,k}\}$. The implication of (2) is that the decoder operates on *soft-decisions*, which generally provides more coding gain than *hard-decision decoding.* The exact nature of (2) depends on the type of channel. For an additive white Gaussian noise (AWGN) channel, $r = 4y\mathcal{E}_s/N_o$ where $\mathcal{E}_s/N_o$ is the symbol signal-to-noise ratio (SNR), while for a flat fading channel, $r = 4ay\mathcal{E}_s/N_o$, where a is the fading amplitude.

The decoder input accepts the scaled channel observations $\mathbf{r}$, while its output produces the *a posteriori probability* (APP) LLRs of the two data-bearing BPSK symbols, $L(\mathbf{x}|\mathbf{y}) = \{L(x_j|\mathbf{y}), L(x_k|\mathbf{y})\}$. Because the goal of the decoder is to produce APP estimates (rather than hard bit decisions), it is called an *APP decoder.* Since there is a one-to-one mapping from $\mathbf{u}$ to $\mathbf{x}$, $\mathbf{u}$ can be estimated by making hard decisions on $L(\mathbf{x}|\mathbf{y})$, i.e. $\hat{u} = 1$ when $L(x|\mathbf{y}) < 0$ and $\hat{u} = 0$ when $L(x|\mathbf{y}) > 0$.

When the decoder is used in an iterative architecture, it also needs to take into account information computed by the other constituent decoders during the previous iteration. Thus, the decoder input must also accept the *a priori* LLRs $\{L(x_j), L(x_k)\}$ which are computed by the other decoders. Likewise, the decoder must produce information which can be used as the a priori LLRs by the other decoders during subsequent iterations. These values are called the *extrinsic information* [1] and are denoted $\{L_e(x_j), L_e(x_k)\}$. The extrinsic information of a data-bearing BPSK symbol represents an estimate of the probability of that symbol given all channel observations *other* than that of the data symbol itself. In other words, the extrinsic information is obtained by exploiting the structure of the code, rather than by using a direct observation of the symbol.

The equations used to describe the input-output relationship of the decoder are based on the BCJR algorithm of [8] and are derived in [7]. The basic building block of the analog decoder is a soft-*xor* operation, also called the "box-plus" operator [5]:

$$w \boxplus z \quad \equiv \quad 2\tanh^{-1}\left[\tanh\left(\frac{w}{2}\right)\tanh\left(\frac{z}{2}\right)\right] \qquad (3)$$

Using this operator, the extrinsic information can then be concisely expressed as:

$$L_e(x_j) \quad = \quad r_{j,k} \boxplus [r_k + L(x_k)] \qquad (4)$$
$$L_e(x_k) \quad = \quad r_{j,k} \boxplus [r_j + L(x_j)] \qquad (5)$$

and the APP output of the decoder is:

$$L(x_j|\mathbf{y}) \quad = \quad r_j + L(x_j) + L_e(x_j) \qquad (6)$$
$$L(x_k|\mathbf{y}) \quad = \quad r_k + L(x_k) + L_e(x_k) \qquad (7)$$

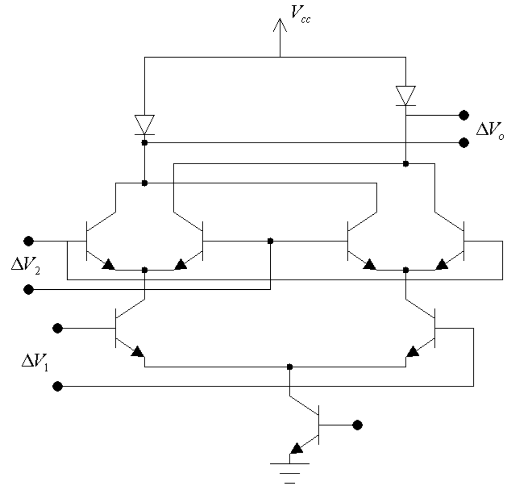

Figure 1: Circuit realization of the $\boxplus$ operator [6].

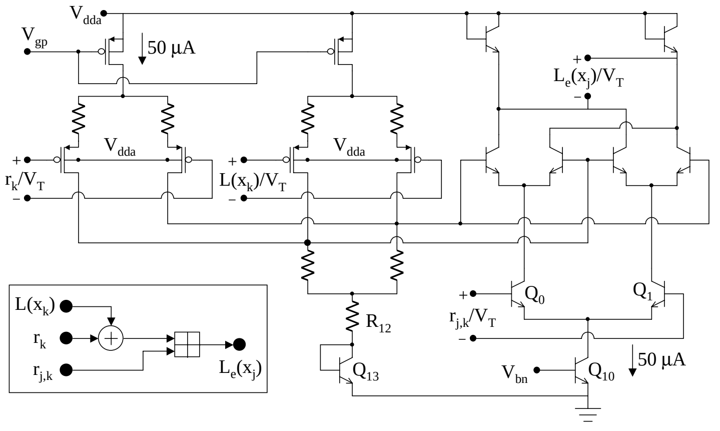

Figure 2: Circuit for computing extrinsic information.

While the implementation of the soft-*xor* operator is awkward in digital hardware, it is simple in analog. In particular, the soft-*xor* can be implemented using the modified Gilbert multiplier shown in Fig. 1 [6]. The differential voltages are related by:

$$\frac{\Delta V_o}{V_T} \quad = \quad 2\tanh^{-1}\left[\tanh\left(\frac{\Delta V_1}{2V_T}\right)\tanh\left(\frac{\Delta V_2}{2V_T}\right)\right] (8)$$

Thus (3) can be implemented by simply setting the first input to $w = \Delta V_1/V_T$, the second input to $z = \Delta V_2/V_T$, and the output to $w \boxplus z = \Delta V_0/V_T$.

Equation (4) can then be implemented by first adding r_k with $L(x_k)$ and then soft-*xor*ing this sum with $r_{j,k}$. A circuit capable of this operation is shown in Fig. 2. A more detailed description of this circuit is given in Section 4.

The only other major operation is the addition of three values to compute the a posteriori LLR. The circuit implementation of (6) is shown in Fig. 3.

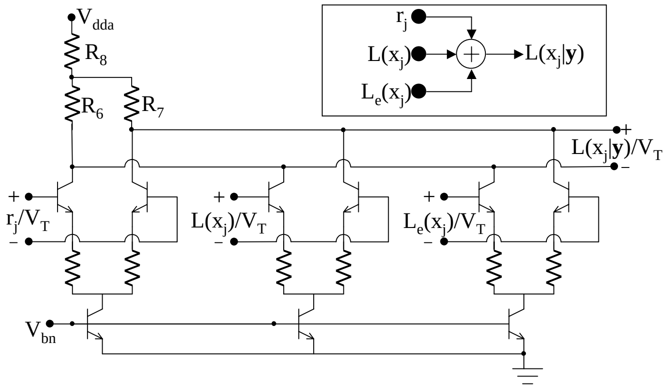

Figure 3: Circuit for computing a posteriori LLR.

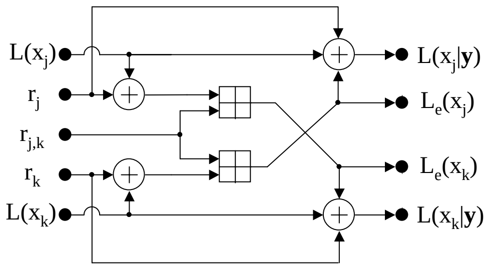

Figure 4: SISO decoder for (3,2) SPC code.

A soft-input/soft-output (SISO) module for decoding the (3,2) SPC is formed by combining a pair of the circuit shown in Fig. 2 with a pair of the circuit shown in Fig. 3. A block diagram representation of the decoder is given in Fig. 4.

3. TURBO PRODUCT CODES

The (3,2) SPC code discussed in the previous section is not particularly useful when used just by itself. However, it can be used as the building block of a more powerful multidimensional *product code*. A two-dimensional (8,4) product code which uses (3,2) SPC codes is shown in Fig. 5. Four data bits, $\{u_1, u_2, u_3, u_4\}$ are placed into a 2 by 2 array. Each row and column of the array is encoded by the (3,2) SPC, resulting in a total of 4 parity bits. The entire 8-bit code word is transmitted over the channel and the resulting set of observations is used as the input to the decoder.

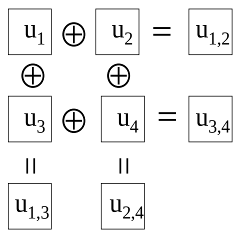

Figure 5: An (8,4) product code.

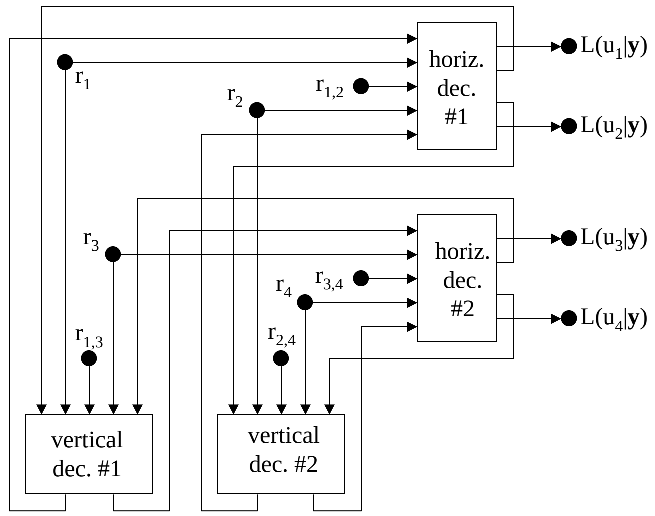

Figure 6: Turbo decoder for the (8,4) code.

A hard-decision decoder could correct a single error in the received 8-bit code word. An erroneous data bit will cause the parity check to fail along the corresponding row *and* column. Alternatively, the error could be amongst the parity bits, in which case the parity check will only fail along either a single row or a single column. However, such a decoder fails to exploit the soft-information available if the code is transmitted over an AWGN or flat-fading channel. For such channels, a soft-input decoder will offer superior performance.

The soft-input decoder for the product code is based upon the turbo decoding principle, and for this reason this type of code is called a *turbo product* or *block turbo* code [9]. Each row and column of the product code is decoded using the structure shown in Fig. 4. For the first iteration of decoding, the a priori inputs $L(x)$ are simply set to zero. The extrinsic information generated from each horizontal decoding is used as a priori information by the vertical decodings during the next iteration (and vice-versa). The entire process runs for a specified amount of time, or until some convergence criteria is met, at which point hard decisions can be made on the a posteriori LLRs computed by the horizontal decoding (or the LLRs from the vertical decoding could be used instead).

A block diagram description of the decoder is shown by Fig. 6. Note that since only the horizontal decoders are used to obtain the a posteriori LLRs, the vertical decoders do not need to compute them. Since the vertical decoders don't use the $L(x_j|\mathbf{y})$ and $L(x_k|\mathbf{y})$ outputs, the two 3-input analog adders on the right side of Fig. 4 are not needed and can be deleted. Thus the overall circuit requires just 8 stages for computing extrinisic information and 4 stages for computing the a posteriori LLRs.

While the concept of iteration is at the heart of digital implementations, it is not relevant for analog implementations. Since the analog implementation is not clocked, the rate at which information is processed and fed back is limited only by the resistance and capacitance within the circuit. Convergence is merely the steady state operating point of the circuit, and thus settling time is a more relevant way to describe the throughput.

4. IMPLEMENTATION DETAILS

A complete decoder was implemented using Cadence design software. The soft-*xor* portion of Fig. 2 is implemented as in [6] using bipolar npn transistors. Biased with 50 μA in the tail current source (Q_{10}), this block is capable of operation up to several hundred MHz. The two input adder utilizes PMOS devices and resistors to achieve a linear addition of the r_j and $L(x_j)$ input values over an input range of a few hundred millivolts. This particular implementation of the adder operates with a maximum input frequency of about 200-250 MHz. Q_{13} and R_{12} achieve the appropriate level translation of the input signals before application to the soft-*xor* block. The $r_{j,k}$ inputs to the soft-*xor* are applied to the bottom BJT pair Q_0-Q_1. All inputs and outputs are maintained fully differential for enhanced signal integrity. The three differential inputs must be scaled by $V_T = kT/q = 0.026$ at 27 C. Ideally these input amplitudes will be temperature compensated to be proportional to Kelvin temperature.

The output summer of the three-input adders shown in Fig. 3 consists of three npn differential pairs which effect a differential voltage to current transformation, allowing current mode summation. The differential output currents are transformed back to a differential voltage by R_6 and R_7. R_8 achieves a downward voltage level shift at the outputs. The current through R_8 is virtually constant, so there is essentially a DC voltage drop across it. The output of the summer is applied to a high speed differential input, digital output comparator.

5. CONCLUSIONS

Iterative decoding algorithms are a natural fit to analog circuitry. The two main reasons that analog circuitry is superior to digital are (1) The soft-*xor* operation which is awkward in digital is easily implemented in analog; and (2) No clock is required in an analog decoder. Because of the absence of a clock, data is continuously exchanged among constituent decoders and thus there is no concept of a finite number of iterations.

However, there are several remaining technical challenges that limit the applicability of analog decoding. One problem is how to get the received information into the decoder. Data is normally transmitted serially, yet the decoder operates on the entire 8-bit code word in parallel. One solution would be to convert the received data to digital solely for the purpose of buffering it, and then convert back to analog when it is time to decode. An alternative solution would be to use multicarrier techniques to transmit the data in parallel over separate channels. Another challenge is how to properly analyze the decoder, which is a nontrivial problem composed of nonlinear feedback differential equations. This makes the task of implementing and testing the circuit in software particularly important.

6. REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimasjshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes(1),” in *Proc., IEEE Int. Conf. on Commun.*, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[2] R. G. Gallager, *Low Density Parity-Check Codes.* MIT Press, 1963.

[3] R. Chen, N. Vijaykrishnan, and M. Irwin, “Clock power issues in system-on-a-chip designs,” in *VLSI ’99., IEEE*, pp. 48–53, 1999.

[4] H. Loeliger, F. Tarkoy, F. Lustenberger, and M. Helfenstein, “Decoding in analog VLSI,” *IEEE Communications Magazine*, pp. 99–101, April 1999.

[5] J. Hagenauer, M. Moerz, and E. Offer, “Analog turbo-networks in VLSI: The next step in turbo decoding and equalization,” in *in Proc. Int. Symp. Turbo Codes & Related Topics*, (Brest, France), pp. 209–218, Sept. 2000.

[6] M. Moerz, T. Gabarra, R. Yan, and J. Hagenauer, “An analog 0.25μm BICMOS tailbaiting MAP decoder,” in *IEEE int. Solid-State Circuits Conference*, pp. 357–356, 2000.

[7] N. Correal, “Principles of analog-domain turbo decoding,” in *Proc. Virginia Tech Symp. on Wireless Personal Commun.*, (Blacksburg, VA), pp. 155–161, June 2002.

[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. 20, pp. 284–287, Mar. 1974.

[9] R. M. Pyndiah, “Near-optimum decoding of product codes: Block turbo codes,” *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.