All-Analog Decoder for a Binary (18,9,5) Tail-Biting Trellis Code

Felix Lustenberger, Markus Helfenstein, and George S. Moschytz
ETH Zentrum
CH-8092 Zürich
Tel. +41 1 632 76 01
[lustenbe,helfenst,moschytz]@isi.ee.ethz.ch

Hans-Andrea Loeliger and Felix Tarköy
Endora Tech AG
Hirschgässlein 40
CH-4052 Basel
Tel. +41 61 273 50 15
[haloeliger,ftarkoey]@endora.ch

Abstract

An all-analog high-speed decoding technique is described which is suitable for magnetic recording (MR) and other computationally demanding applications. A decoder for a binary (18,9,5) tail-biting trellis code, which is much simpler than the codes used for MR, has been chosen to demonstrate this technique. It achieves a decoding rate of 100 Mbit/s at a single 5V power supply. The power consumption is 50 mW. Higher speed can essentially be traded for higher power consumption. A comparison shows that a digital implementation is outperformed by more than two orders of magnitude in terms of speed and/or power consumption.

1. Introduction

The ever increasing speed and capacity of modern hard disk drives (HDD) ask for more complex coding and decoding techniques. Analog implementations of partial-response maximum-likelihood (PRML) Viterbi decoders have been proven to be a viable alternative to purely digital implementations [1]: speed, size and power constraints can be partially met by using an analog Viterbi decoder in the processor core of PRML read channels. In the following, we go a step further in implementing the entire decoder in analog VLSI technology. It has first been shown in [2, 3] that decoding networks consisting of simple analog building blocks, comparable to soft logic gates, can outperform digital implementations by a distinct margin in terms of speed and/or power consumption. The design technique has a wide application range far beyond magnetic recording, such as state-of-the-art error-correcting coding for wireless and wireline communications, complex signal-processing tasks, and artificial intelligence. It is strongly related to what Mead calls the “neuromorphic design style” [4]. The presented decoder was built to validate our decoding concept; a real-world decoder would be more complex, but can be built using the same technique. Related work in the field of all-analog decoding techniques has been published by Hagenauer [5].

2. Generic Sum-Product Module

The decoder networks considered in this article consist of modules as in Fig. 1. The input and output signals of each module are current vectors that represent discrete probability distributions. A probability distribution defined on some finite set $\mathcal{X}$ is simply a real-valued function $p$ such that $p(x) \geq 0$ for all $x \in \mathcal{X}$ and $\sum_{x \in \mathcal{X}} p(x) = 1$. For some given set $\mathcal{X} = \{x_1, \ldots, x_m\}$, a current vector $[I_1, \ldots, I_m]$ with non-negative components represents the probability distribution $p(x_i) = I_i / \sum_{k=1}^{m} I_k$. Conversely, a probability distribution $p$ can be represented by a current vector $[p(x_1), \ldots, p(x_m)]$ with an arbitrary non-negative sum current $I_s$.

The generic module (Fig. 1) takes two probability distributions $p_\mathcal{X}$ and $p_\mathcal{Y}$ and computes a probability distribution $p_Z$ according to

$$p_Z(z) = \gamma \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_\mathcal{X}(x) p_\mathcal{Y}(y) f(x, y, z) \tag{1}$$

$\forall z \in \mathcal{Z}$, where $f$ is a $[0, 1]$-valued function and $\gamma$ is an appropriate scale factor that does not depend on $z$. Note that (1) defines a large family of modules parameterized by the function $f$.

Figure 1. Generic sum-product module.

Figure 2. One trellis section of a binary (18,9,5) tailbiting trellis code.

Specific functions $f$ are conveniently illustrated by trellis diagrams as in Fig. 2. 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 $f$. 
A given function $f$ can be directly mapped onto a transistor circuit, and the topology of the trellis is evident in the circuit modules. The factor $\gamma$ in (1) can be adjusted by appropriate scaling circuit blocks. A more detailed description and derivation of these building blocks can be found in [3].

3. Binary (18,9,5) Tail-Biting Trellis Code

To demonstrate the new design technique, a binary (18, 9, 5) block code has been chosen. In the terminology of coding theory, this means that each codeword consists of 18 bits, the code contains $2^9 - 512$ codewords, and any two codewords differ in at least 5 positions. The code is defined by the trellis diagram in Fig. 2. The full tail-biting trellis is built by concatenating 9 such sections to form a ring structure. The codewords correspond to those paths through the trellis that start and end in the same trellis state. For any such path, both the information bits and the corresponding coded bits can be read off the edge labels along the path. From here on, it is assumed that codewords are transmitted over a memoryless channel with transition probabilities $p(y|x), x \in \{0, 1\}$.

A block diagram of the decoding network for this code is shown in Fig. 3 with the computation modules as defined in Fig. 4. Each signal line in Fig. 3 represents a whole probability mass function (i.e. a current vector in the circuit realization).

The input to the decoder consists of the probabilities $p(y_0 | x_0)$ and $p(y_{11} | x_1), i = 1, \ldots, 18$, where $y = [y_1, \ldots, y_{18}]$ is the channel output data. The outputs of the decoder are approximate a-posteriori probabilities $\tilde{p}(b_j | y)$ for all information bits $b_j$ with $j = 1, \ldots, 9$. A final decoding decision may be obtained by comparing $\tilde{p}(b_j = 1 | y)$ to $\tilde{p}(b_j = 0 | y)$.

This decoder is a direct implementation of the “forward-backward” algorithm [6], which is a special case of the general sum-product algorithm [7]. In Fig. 3, the type-$B$ modules perform the “forward” computation and the type-$C$ modules perform the “backward” computation on the tail-biting trellis. The type-$A$ modules precompute the branch metrics, and the type-$D$ modules compute the final probabilities for each information bit.

Note that this decoding network contains two loops, which correspond to the forward and backward computation. In general, networks with loops may not always converge to a stable state. However, it can be shown that networks as in Fig. 3 with no interacting loops are guaranteed to be stable unless some of the input probabilities are zero.

4. Experimental results

The decoder network of the previous section was designed as an analog network with about 940 BJTs and 650 pMOS transistors in the AMS 0.8 $\mu$m double-metal BiCMOS technology. Fig. 5 shows the circuit implementation of the module of type $B$. Other modules are built similarly. For each edge in the trellis representation (Fig. 4), a transistor can be identified in the middle row of the circuit diagram. Additionally, dummy transistors are introduced such that each state has the same number and types of outgoing branches. The shaded path (Fig. 5) corresponds to the multiplication of $py_{11}, px_{00}$. 

Figure 3. Block diagram of the considered decoder

Figure 4. Trellis representation of the calculations carried out by the different modules

Figure 5. Circuit implementation of type-$B$ module
A typical simulation response is shown in Fig. 6. It demonstrates the correction of two bit errors for a binary symmetric channel (BSC) with crossover probability 0.05. The transient curves in Fig. 6 show the computed approximate \( p(b|y) \) for all nine information bits. In this example, it takes about 30 ns until the sign of all output probabilities have reached their final value. The bias current of each module was 50 \( \mu \)A. A probability of 1 corresponds to a current value of 50 \( \mu \)A. A single 5 V supply was used, and the total (static and dynamic) power consumption was 50 mW.

For testing the implemented circuit chip, an HP 83000 digital circuit tester was used to command the large number of high-speed D/A converters which generate the input waveforms. Since these signals have to be applied in parallel, the converters act as external analog memory. The input voltages are converted on chip into the currents as needed by the decoder core. In addition, the testboard contains I-V converters and high-speed ECL comparators for measurement purposes.

Measurements of the transient behavior of the output probabilities, shown in Fig. 7, match well with the simulation results. The measured approximate output probabilities of bit 1 and bit 2 are drawn in this plot; the error correction of bit 1 can be seen clearly. Furthermore, the output of the external comparator for the hard-decision on bit 1 is shown, making the decoding delay of approx. 31 ns evident. We observed heavy ringing of the output currents, caused by coupling between the pins of the package, which could not be calibrated out completely. However, it was found that decoding speed and errors were not affected by this ringing.

In Fig. 8, the result of a different measurement is shown. The sourceword \( b = [1, 0, 1, 1, 0, 1, 1, 0, 1] \) is encoded and applied to a BSC with crossover probability 0.05. Consecutively, zero, one, two, and three bit errors were applied, where the three-error configuration corresponds to another codeword with two bit errors. The error correction capability for two applied errors can clearly be observed in Fig. 8. In this measurement setup, a probability of 1 corresponds to 200 \( \mu \)A and one oscilloscope division corresponds to 25 \( \mu \)A. The differences of the output amplitudes of bit 1 and bit 3 stem from device mismatch effects within the decoder core. The building blocks were designed with virtually minimal transistor sizes as in digital design. Extensive Monte-Carlo simulations were made in which none of the simulated configurations failed and the standard deviation of the output currents was within 3 to 4% of the nominal value. Therefore it is assured that the decoder is very robust against mismatch errors up to a crossover probability of 25% in the BSC.

A chip micrograph of the prototype implementation is shown in Fig. 9. The entire chip area is 2.8 \( \times \) 2.6 mm\(^2\) including pads. The area of the decoder itself is 1.7 \( \times \) 0.7 mm\(^2\). The remaining area is taken by VI-converter needed for measurement purposes.

With a decoding time of 90 ns per decoded 9-bit sourceword, a data rate of 100 Mbit/s can be achieved, which includes ample margin and reset time. The main chip characteristics are summarized in Table 1.
5. Comparison with Digital Implementation

In the following, an estimate of the power consumption of an equivalent digital decoder is deduced. We found by high-level discrete-time simulations that the decoding algorithm converges to its final value after 15 iterations per section, which corresponds to a total of 4400 multiplications and 1100 additions (all in floating-point format). In order to achieve the desired bit rates, fast but low-power 5-bit multiply and add processors are assumed at each node of the trellis. The maximum operating frequency of these node processors is 167 MHz (15 iterations within 90 ns). Defining a gate as the basic digital circuit with up to $n$ inputs and 1 output, 30 gates are needed for implementing the full adder, and 110 for the one-step multiplier.

Furthermore, assuming that all gates have the same delay time $t_D$, an average node capacitance of 0.1 pF, an activity per gate of 1/4 (i.e. every node charges and discharges within 8 calculation steps) and an energy loss due to overlap currents of 20%, the dissipated power per multiplication can be estimated as 2.5 mW. Similarly, 0.65 mW is required per 5-bit addition. Therefore, operating from a single 3 V power supply and neglecting additional scaling operations and buffering, the overall power consumption for the decoder under consideration can be estimated to be above 11.5 W. Accordingly, the power efficiency of our analog decoder is superior by a factor of $2 \times 10^3$ compared to its digital counterpart. Similar numbers could be found for the efficiency in the use of die area. A digital implementation of such or similar iterative decoders is possible with today’s CMOS processes if a $t_D = 0.45$ ns is assured for $n = 6$. However, such digital implementations are limited to small codes, since the burden on power consumption and die area becomes physically unacceptable.

6. Conclusions

A decoder for a binary (18,9,5) tail-biting trellis code based on a new design technique has been presented. A data rate of 100 MBit/s was measured at 50 mW power consumption. A comparison to an equivalent digital implementation exhibits more than two orders of magnitude less power and/or more speed. A next chip implementing a decoder for a more complex code and including digital interfaces and analogue memories is in preparation.

7. Acknowledgement

The authors are grateful to Felix Frey and Patrick Schweizer for designing and setting up the HP83000 measurement environment.


