A simplified computational kernel for trellis-based decoding

Kamuf, Matthias; Anderson, John B; Öwall, Viktor

Published in:
IEEE Communications Letters

DOI:
10.1109/LCOMM.2004.825743

2004

A Simplified Computational Kernel for Trellis-Based Decoding

Matthias Kamuf, *Student Member, IEEE*, John B. Anderson, *Fellow, IEEE*, and Viktor Öwall, *Member, IEEE*

***Abstract*—A simplified branch metric and add-compare-select (ACS) unit is presented for use in trellis-based decoding architectures. The simplification is based on a complementary property of best feedforward and some systematic feedback encoders. As a result, one adder is saved in every other ACS unit. Furthermore, only half the branch metrics have to be calculated. It is shown that this simplification becomes especially beneficial for rate 1/2 convolutional codes. Consequently, area and power consumption will be reduced in a hardware implementation.**

***Index Terms*—Add-compare-select (ACS), branch metric, logMAP, very large-scale integration (VLSI), Viterbi decoding.**

I. Introduction

TRELLIS-BASED decoding is a popular method to recover convolutionally encoded information corrupted during transmission over a noisy channel. For example, the Viterbi algorithm (VA) [1] and the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [2] are two schemes that work on an underlying trellis description of the encoded sequence.

Basic computations in either algorithm involve branch metric (BM) calculations and add-compare-select (ACS) operations. In case of the VA, an ACS operation successively discards branches that cannot be part of the survivor path. In case of the BCJR in the logarithmic domain (the logMAP algorithm [3]), this operation corresponds to an add-max* operation [4], which is basically an ACS operation with an added offset (ACSO) to correct for the Jacobian logarithm. Hence, the presented considerations for the ACS hold for the ACSO as well.

Almost all good rate $1/n$ convolutional codes, n an integer, have the property that the code symbol labels on the two branches into each trellis node are complementary. However, simplifications to the BM and ACS units that result have not been published yet, to our knowledge. Consequently, we use this property and present a simplified architecture with reduced complexity for these units, thus saving hardware.

After having introduced a convenient notation in the following section, we address the simplification in Section III and the resulting implications for hardware realization in Section IV.

Manuscript received July 4, 2003. The associate editor coordinating the review of this letter and approving it for publication was Dr. I. Fair. This work is supported by the Swedish Socware program and by the EU Pacwoman project.

M. Kamuf and V. Öwall are with the Department of Electroscience, Lund University, SE 221 00 Lund, Sweden (e-mail: mkf@es.lth.se; vikt@es.lth.se).

J. B. Anderson is with the Department of Information Technology at Lund University, Lund University, SE 221 00 Lund, Sweden (e-mail: anderson@it.lth.se).

Digital Object Identifier 10.1109/LCOMM.2004.825743

II. Notation

The ACS operation is best described by (1). Let $\Gamma(s, k+1)$ be the updated metric of state s at time $k+1$, based on the preceding state metrics at time k and the respective branch metrics $\lambda()$:

$$\Gamma(s,k+1) = \min\left\{\Gamma(s',k) + \lambda(s',s), \Gamma(s'',k) + \lambda(s'',s)\right\} \tag{1}$$

A channel symbol received from a soft-output demodulator is quantized with q bits and denoted y_i. Clearly, there are 2^q quantization levels and $y_i \in [0, 2^q - 1]$. This symbol is the output of a discrete memoryless channel with binary input x_j and transition probabilities $P(y_i|x_j)$. The expected code symbol $c_i(s', s)$ along the branch from state s' to state s is derived by the mapping $x_0 \mapsto 0$ and $x_1 \mapsto 2^q - 1$.

In the additive white Gaussian noise channel the optimal distance measure is the squared Euclidean distance

$$\sum_{i=0}^{n-1} \left|y_i - c_i(s',s)\right|^2. \tag{2}$$

However, given the preceding symbol constraints, it is shown in [5] that this measure simplifies to

$$\lambda_i(s',s) = \begin{cases} y_i, & \text{for } c_i(s',s) = 0 \\ (2^q - 1) - y_i, & \text{for } c_i(s',s) = 2^q - 1 \end{cases} \tag{3}$$

and the complete branch metric is then written as

$$\lambda(s',s) = \sum_{i=0}^{n-1} \lambda_i(s',s). \tag{4}$$

III. Using the Complementary Property

This discussion is restricted to rate 1/2 codes, that is $n = 2$, although the considerations can be generalized to $1/n$ codes. Rate 1/2 codes play by far the most important role in today's communication systems since they are a good compromise between achievable coding gain, bandwidth efficiency, and implementation complexity. In practice, high-rate codes are usually obtained by puncturing a basic rate 1/2 code. However, we begin with a general notation that shows that the most beneficial simplification results for $n = 2$.

We consider both feedforward encoders and some systematic feedback encoders. The best feedforward encoders of memory m are defined by two shift register tap sets which are delayfree [6]. Some best systematic feedback encoders can be found in [7].

These encoders have one thing in common: The code symbols of merging branches are always complementary. In Fig. 1,

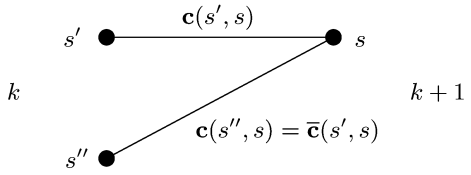

Fig. 1. Complementary property of merging branches.

the complementary operation on $\mathbf{c}$ is defined as the complementation of its elements, that is $\overline{\mathbf{c}} = (\overline{c}_0 \ldots \overline{c}_{n-1})$, where $c_i + \overline{c}_i = 2^q - 1$.

From the considerations in Section II it is clear that the branch metrics share this property since they linearly depend on the code symbols. Hence, one branch metric can be expressed by means of the other and we write

$$\begin{aligned}\lambda(s'', s) &= n(2^q - 1) - \lambda(s', s) \\ &= (n-1)\lambda(s', s) + n\left[(2^q - 1) - \lambda(s', s)\right]. \quad (5)\end{aligned}$$

We define the modified branch metric

$$\lambda^*(s', s) \equiv n\left[(2^q - 1) - \lambda(s', s)\right] \quad (6)$$

which is a signed number, and (5) becomes

$$\lambda(s'', s) = (n-1)\lambda(s', s) + \lambda^*(s', s). \quad (7)$$

Substituting (7) into (1) gives

$$\begin{aligned}\Gamma(s, k+1) = \min\{&\Gamma(s', k) + \lambda(s', s), \Gamma(s'', k) \\ &+ (n-1)\lambda(s', s) + \lambda^*(s', s)\}. \quad (8)\end{aligned}$$

Finally, the factor $\lambda(s', s)$ in the first argument of (8) can be taken out of the comparison and we get

$$\begin{aligned}\Gamma(s, k+1) = \lambda(s', s) + \min\{&\Gamma(s', k), \Gamma(s'', k) \\ &+ (n-2)\lambda(s', s) + \lambda^*(s', s)\}, \quad (9)\end{aligned}$$

or, equivalently

$$\Gamma(s, k+1) = \lambda(s', s) + \tilde{\Gamma}(s, k+1) \quad (10)$$

where $\tilde{\Gamma}(s, k+1)$ is the new outcome of the min operation.

There are several things to be observed in (9) and (10). First, considering that the branch metrics are precalculated, there is one addition less needed to carry out the comparison since the first argument in the comparison remains unchanged. Second, for $n = 2$ the factor $\lambda(s', s)$ disappears in the second argument and the comparison solely depends on one (modified) branch metric. Third, in order to retain the numerical relation between interconnected state metrics with different $\lambda()$ we have to add this factor after having determined $\tilde{\Gamma}(s, k+1)$. However, one can subtract this factor from all state metrics and it will be shown that in that case half the ACS units do not need this correction, that is $\Gamma(s, k+1) = \tilde{\Gamma}(s, k+1)$. Note that if the butterflies in a trellis were disjoint, this correction could be neglected in all ACS units.

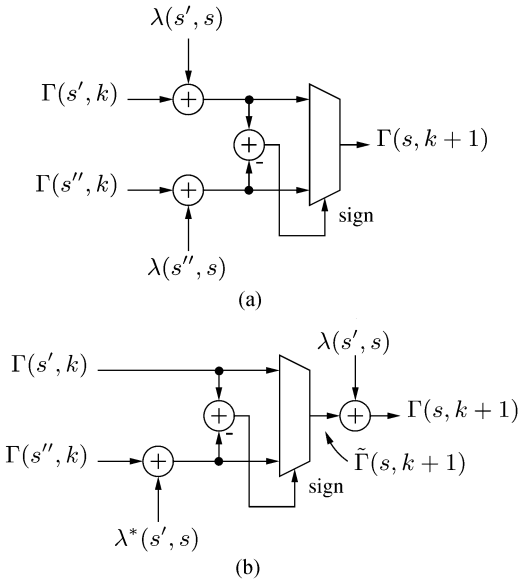

Fig. 2. (a) Conventional and (b) transformed ACS unit for a rate 1/2 code. Both units have the same complexity but the latter needs one adder less to determine the outcome of the comparison.

IV. Modified BM and ACS Units for Rate 1/2

We start by noting that the branch metric $\lambda(s', s)$ can take four different values, namely $\lambda(x_0 x_1)$ for every possible combination of symbols $x_j \in \{0, 1\}$.

Fig. 2 shows both the conventional and the transformed ACS unit. Both units have the same complexity but the latter needs one adder less to determine $\tilde{\Gamma}(s, k+1)$. The hardware savings now become apparent by looking at an example, an ACS unit setup for decoding a (7,5)-code in Fig. 3. In this picture, the factor $\lambda(s', s)$ of Fig. 2(b) to be added in an ACS unit is either $\lambda(00)$ or $\lambda(10)$. However, we can subtract, for example, $\lambda(00)$ from all state metrics. This factor belongs to the two ACS units on the left and, therefore, the state metric corrections in these units become unnecessary while

$$\Delta\lambda = \lambda(10) - \lambda(00) \quad (11)$$

has to be added to the other units. Hence, for rate 1/2 codes that have the complementary property half the ACS units save one adder compared to a conventional setup. If speed is an issue, $\Delta\lambda$ could be stored in the BM unit and added in the next computation cycle instead, thus maintaining the original critical path of the conventional ACS unit. However, the BM unit becomes slightly more complex in this case.

The calculation of the modified branch metric $\lambda^*(s', s)$ based on (6) for $n = 2$ is shown in Fig. 4. Normally, the expression in square brackets in (6) would be the bit-complement of $\lambda(s', s)$. However, since $\lambda(s', s) \in [0, 2(2^q - 1)]$ one has to exclude the most significant bit (MSB), which indicates the sign of the modified branch metric, from the negation. Since $n = 2$ the multiplication in (6) reduces to a left shift by one bit. Note that if n is not a power of two, this multiplication cannot be reduced to bit-shift operations.

As stated in [5], if there are 2^n distinct code sequences, a conventional BM unit requires $2^n(n-1)$ additions and n negations to calculate 2^n branch metrics. Hence, for a rate 1/2 code

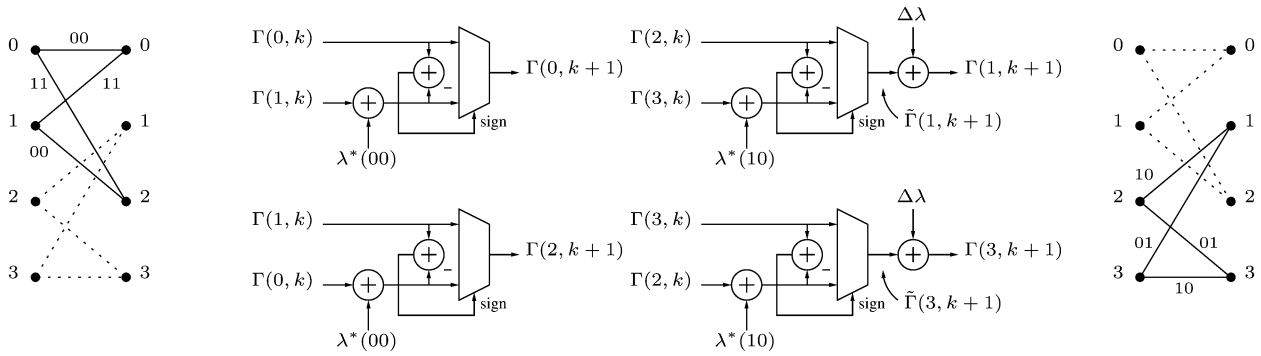

Fig. 3. Proposed ACS unit setup for decoding a (7, 5)-code.

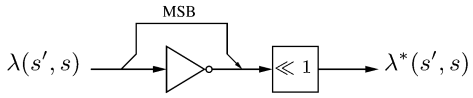

Fig. 4. Generation of $\lambda^*()$, $\ll 1$ denotes a left shift by one bit.

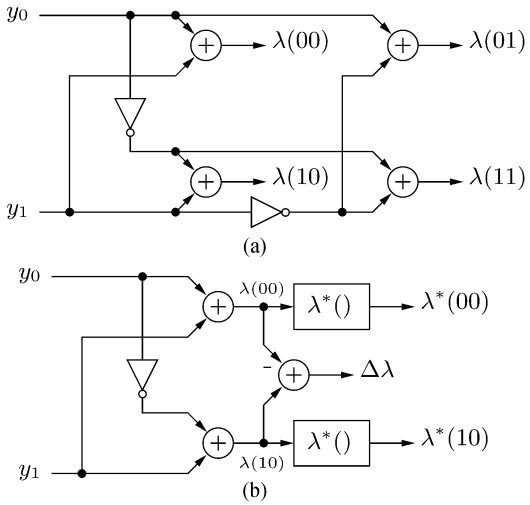

Fig. 5. (a) Conventional and (b) proposed BM unit for a rate 1/2 code.

we need four adders and two negations to calculate four branch metrics, see Fig. 5(a). The proposed BM unit shown in Fig. 5(b) requires only three additions, one negation of a channel symbol, and two negations of intermediate branch metrics to calculate two branch metrics. Notice that a bit-shift operation comes at negligible cost in a hardware implementation. Furthermore, the difference between the two branch metrics, $\Delta\lambda$, needed to normalize half the state metrics becomes in this case simply $(2^q - 1) - 2y_0$. This operation can be further simplified on the bit level into a bit-shift followed by a negation (MSB excluded) of y_0 and is hence not considered an adder in Table I.

This table shows the number of additions for a BM/ACS unit setup for code rate 1/2 and memory m. The proposed scheme

TABLE I
NUMBER OF ADDITIONS FOR BM/ACS UNIT SETUP OF A RATE 1/2 CODE

	ACS	BM
conventional	$3 \cdot 2^m$	4
proposed	$5 \cdot 2^{m-1}$	2

halves the additions in the BM unit and reduces the number of additions for the ACS units by 17%. By software simulation of the hardware circuits, we have verified that decoder error performance stays the same.

V. CONCLUSION

We have shown that the implementation of BM and ACS units in trellis-based decoding architectures can be simplified for a certain class of convolutional codes. For a rate 1/2 code, half the ACS units save one adder compared to a conventional implementation. Furthermore, only two branch metrics have to be calculated instead of four. These potential hardware savings will also lead to savings in power consumption.

REFERENCES

[1] G. D. Forney Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.
[2] 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.
[3] P. Robertson, E. Villebrun, and P. Höher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. on Communications (ICC)*, Seattle, WA, June 1995, pp. 1009–1013.
[4] E. Boutillon, W. J. Gross, and P. G. Gulak, "VLSI architectures for the MAP algorithm," *IEEE Trans. Commun.*, vol. 51, pp. 175–185, Feb. 2003.
[5] H.-L. Lou, "Implementing the Viterbi algorithm," *IEEE Signal Processing Mag.*, vol. 12, no. 5, pp. 42–52, Sept. 1995.
[6] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. Piscataway, NJ: IEEE Press, 1999.
[7] J. B. Anderson, "Best short rate 1/2 tailbiting codes for the bit-error rate criterion," *IEEE Trans. Commun.*, vol. 48, pp. 597–610, Apr. 2000.