Efficient implementation of parallel decision feedback decoders for broadband applications

H. Lou, M. Rupp, R.L. Urbanke, H. Viswanathan and R. Krishnamoorthy

Bell Labs, Lucent Technologies
600 Mountain Ave
Murray Hill, NJ 07974

ABSTRACT

Parallel Decision-Feedback Decoders (PDFD) use a joint equalization and channel decoding scheme that performs decision-feedback equalization based on each survivor path in the channel decoder. In this paper, we propose novel efficient computation methods to implement the PDFD. We show that, contrary to the belief that PDFD is too complex to implement, standard Viterbi decoders can be modified to implement the PDFD and achieve very high bit-rates.

I. INTRODUCTION

Decision Feedback Equalizers (DFE) are typically used to combat inter-symbol interference of the transmitted symbols. It is well known that the performance of a system using separate DFE and channel decoding is significantly degraded by error propagation in the DFE [1]. Several joint equalization and decoding schemes have been proposed in the literature to improve the performance of coded modulation schemes when a DFE is used. This paper is on using a joint DFE and **Trellis Coded Modulation (TCM)** decoding scheme that performs decision-feedback equalization based on each survivor in the TCM decoder. This technique, called **Parallel Decision-Feedback Decoding (PDFD)**, was first proposed in [1]-[3] to reduce the complexity of an optimum joint maximum-likelihood equalization and decoding scheme. Even though PDFD is suboptimal in performance compared to an optimum joint MLSE scheme, depending on the channel conditions, it can still achieve significant performance gains [4]. For broadband applications, PDFD is still considered very complex to implement. In this paper, we present novel efficient methods to implement the PDFD and show that standard channel decoders, using the Viterbi algorithm [5], can be modified to implement PDFD and achieve very high bit-rates.

The following section first describes the PDFD scheme. After that, efficient methods to implement the PDFD applicable to most TCM schemes of practical interest are presented in Section III.

II. THE PDFD ALGORITHM

A Viterbi decoder generally has a latency of L decoding stages, where L is the decoding depth [5]. Thus, the DFE cannot wait for the decoder to obtain the global most-likely decoded symbols and then feed these symbols into the feedback section of the DFE. Therefore, a conventional DFE makes a tentative decision before the Viterbi decoder and feeds back this value to the feedback path of the DFE. On the other hand, PDFD is a technique in which no tentative decisions before decoding are made for the feedback part of the DFE. Instead, the feedback section of the DFE is computed separately for each survivor path of each state in the trellis in order to determine the soft output of the equalizer for each state. This is done at each decoding stage of the Viterbi decoder. That is, PDFD maintains a DFE for each survivor path corresponding to each state in the given TCM trellis (and thus the name parallel decision feedback decoder). The feedback part of each DFE uses the last N_b decoded symbols in the survivor path of that state. N_b is the number of taps in the feedback part of the DFE and is a design parameter.

The PDFD scheme can be described mathematically as follows: Denote the survivor path at state s at decoding stage t by $\underline{x}_t(s) = (x_t(s,i), 1 \le i \le t)$ where $x_t(s,i)$ is the decoded symbol on the branch between decoding stage $i-1$ and decoding stage i. If the feed-forward and feedback taps of the DFE are $\mathbf{w} = (w_1, \cdots, w_{N_f})$ and $\mathbf{b} = (b_1, \cdots, b_{N_b})$, respectively, then the survivor path at state s would be extended based on the branch metric calculated using the soft input:

$$y_t(s) = \sum_{i=0}^{N_f-1} w_i r(t+i) - \sum_{j=1}^{N_b} b_j x_t(s, t-j), \qquad (1)$$

where $r(t)$ is the output of the feed-forward filter at time t, and N_b and N_f are the number of feedback and feed-forwards taps, respectively.

Thus, the branch metric calculation has to be repeated for each state. This is in contrast to the conventional Viterbi decoder that obtains the same output value from the DFE. Therefore, the PDFD not only has to update the feedback section of the DFE, given the current Viterbi decoder survivor path for each state, it also has to compute the branch metrics using a different input value $y_t(s)$ for each state.

III. EFFICIENT IMPLEMENTATION OF PDFD

From the previous section, we see that conventional separate DFE and channel decoding scheme has to update only one feedback path in the DFE while PDFD has to update up to 2^ν feedback paths in the DFE, corresponding to the 2^ν states (or survivors) in the Viterbi decoder. (ν is the constraint length of the channel code.) Furthermore, since a different input value $y_t(s)$ is computed for each survivor path coming into each state, the branch metrics for paths coming into

0-7803-5682-9/99/$10.00©1999 IEEE.

each state have to be computed. This is in contrast to standard Viterbi decoders where the branch metrics computed can be shared among all states. Thus, this section focuses on efficient computation methods to update the feedback paths of the DFE and to compute the branch metrics. Implementation of the rest of the Viterbi decoder, as well as the finite precision requirement, in a PDFD is similar to that of a standard Viterbi decoder. Since there is abundant literature written on standard Viterbi decoder design (an extensive reference list is given in [6]), other implementation issues regarding the Viterbi decoders are not addressed here.

A. *Efficient decision-feedback path updates*

As described in the Section II, the DFE output for each state, s, at decoding stage t, $y_t(s)$, has to be updated using Equation 1. Since each $x_t(s, t-j)$ is a constellation point in a PSK or QAM constellation, the multiplication of a tap coefficient, b_i, with a known constant, x_t can be implemented by shifters and adders, as is done in the current state-of-the-art systems [7]. We propose that rather than using the conventional constellation, shown in Figure 1, with the constellation points, $x_i \in \{1/\sqrt{2}(\pm 1, \pm j)\}$, a rotated constellation with points $x_i \in \{1, j, -1, -j\}$ is used instead. Rotating the constellation points does not change the behavior of the digital modulation scheme nor its transmission since the hardware dependent implementation will add arbitrary rotations anyway.

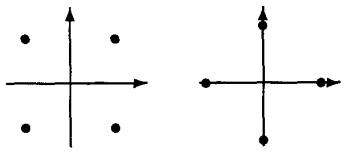

Fig. 1. *Conventional and proposed QPSK constellations.*

For the conventional QPSK constellations, $x_i \in \{\pm 1, \pm j\}$. No multiplication is required but two adders are required to compute $x_i b_i$. By using the rotated QPSK constellation with $x_i \in \{1, j, -1, -j\}$, the multiplication becomes a selection operation. No adders are required.

$$x_i = 1: \quad x_i b_i = \quad \text{Real}(b_i) + j\text{Imag}(b_i) \qquad (2)$$

$$x_i = j: \quad x_i b_i = \quad -\text{Imag}(b_i) + j\text{Real}(b_i) \qquad (3)$$

$$x_i = -1: \quad x_i b_i = \quad -\text{Real}(b_i) - j\text{Imag}(b_i) \qquad (4)$$

$$x_i = -j: \quad x_i b_i = \quad \text{Imag}(b_i) - j\text{Real}(b_i)\,. \qquad (5)$$

Figure 2 depicts the basic selector structure that implements $b_i x_i$, the multiplication of a complex-valued filter coefficient with a constellation point. We denote the two bits that define a QPSK constellation point, bit c_1 and bit c_0. A possible mapping scheme to map these two bits into a QPSK constellation point is: $(c_1, c_0) = (0,0)$ for $x_i = 1$, $(c_1, c_0) = (0,1)$ for $x_i = j$, $(c_1, c_0) = (1,0)$ for $x_i = -1$ and $(c_1, c_0) = (1,1)$ for $x_i = -j$. In the figure, the inverters are assumed to be active high. The switches are assumed to be connected to the upper position when the control signal is "1".

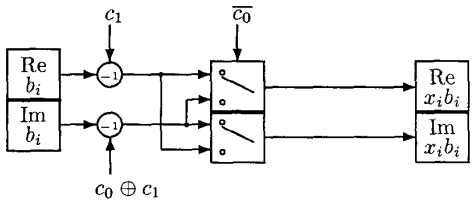

Fig. 2. Basic selector operation to implement a complex multiplication of a filter tap b_i with a point in a QPSK constellation.

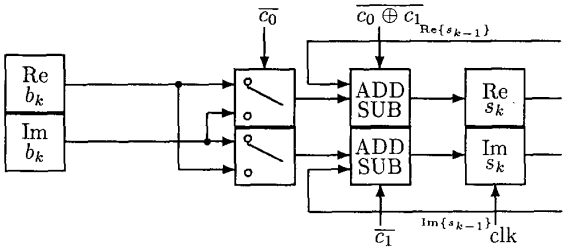

Fig. 3. Recursive structure to implement the the feedback filter tap updates.

Since there are N_b filter tap coefficients (Equation 1), addition operations are required. Assume that the partial sum is computed up to position $k-1$ where

$$s_{k-1} = \sum_{i=1}^{k-1} x_i b_i\,, \qquad (6)$$

the next step is to compute s_k where

$$s_k = s_{k-1} + x_k b_k\,, \quad k = 2...N_b.$$

Thus, apart from some logic and selectors, only two add/sub units are required if the sum is to be computed recursively. The add/sub units are assumed to perform the addition operation when the control signal is high. Figure 3 is an example to implement the recursive operation described in Equation 6. Parallel structures can also be introduced depending on the throughput required. As we can see, using our method, $2(N_b-1)$ additions are required to update one feedback path. Thus, if $N_b = 3$ and a fully parallelized structure is used, the throughput that can be achieved by implementing the the complete feedback path update is limited by the amount of time required to process two stages of adders.

To apply the same idea to a larger constellation, we use a 16-QAM constellation as an example. Similar to the QPSK case, the set of 16 constellation points, $x_i \in \{\pm 1 \pm j, \pm 1 \pm 3j, \pm 3 \pm j, \pm 3 \pm 3j\}$ is rotated by 45^o as shown in Figure 5. The figure also shows how the constellation can be separated into four subsections. Each subsection is a translated QPSK constellation. Thus, in order to implement the multiplication of a filter coefficient with a symbol from a 16-QAM constellation, the corresponding filter coefficient is to be selected as described in (2)-(5) followed by a shift operation if necessary. After that, this value is added to another selected value as displayed in Figure 6. In the figure, the block labeled "SEL" is given in Figure 2 and the one labeled "REC" is given in Figure 4. "L-SH"

denotes the left shift operation as mentioned above. As an example, the constellation point $(1+3j)$ in the conventional 16-QAM constellation is now (due to rotation and stretching) mapped into $(2+j)$. (Note that the definition of one unit in actual implementation is arbitrary.) In the first step, the operation "mult by 2" is performed using a selection operation followed by a shift operation. After that, the filter coefficient is multiplied by j, which is another selection operation. Finally, the two values obtained from the two steps are added together. This complex multiplication thus requires two real add operations. We note further that due to symmetry in the 16-QAM constellation, the multiplications of one filter tap coefficient with different constellation points may be shared [8].

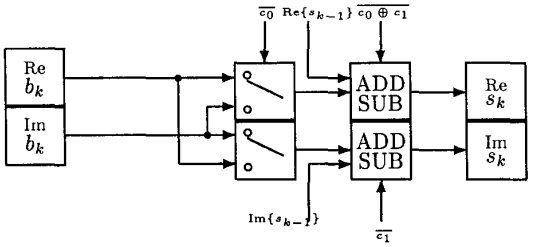

Fig. 4. Structure to compute $s_k = s_{k-1} + x_k b_k$.

B. Efficient branch metric computation

As discussed earlier, the main complexity increase in the Viterbi decoder due to PDFD is in the branch metric computation. Since each state in the Viterbi decoder has a different feedback path for the PDFD, the input value for each state is different. Thus, a branch metric for each transition in the TCM trellis has to be computed. Therefore, the branch metric computation in a PDFD can become the bottleneck in the decoder, especially for high bit-rate applications.

We find that during the search for the survivor path, commonly known as the add-compare-select process in standard Viterbi decoders [6], linear distances can be used to represent the branch metrics. Even though squared distances may have to be added to the path metric of the survivor path for each state, 2^{ν} squared distances have to be computed rather than $2^{(k+\nu)}$. Furthermore, the computation of the squared distances can be done in parallel to the add-compare-select units. Thus, they are no longer in the critical path and can no longer be the bottleneck.

In this section, we denote

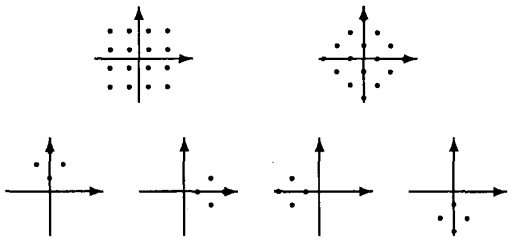

Fig. 5. Splitting a 16-QAM in four subsets.

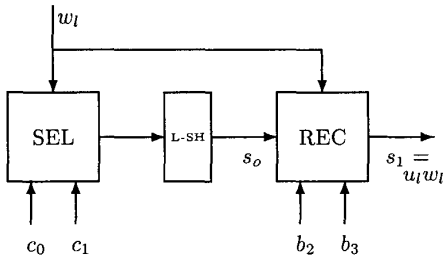

Fig. 6. Two-step chain for multiplication in a 16 QAM set.

- $y_{n,m} \equiv$ the m^{th}-dimensional received noisy symbol at decoding stage n. When $m = 1$, $y_{n,m} = y_n$.
- $C_{i,j} \equiv$ the output symbol associated with the transition from state i to state j. This symbol is a codeword for convolutional codes and a coset number for trellis codes.
- $p_{i,j,m} \equiv$ the nearest codeword in coset $C_{i,j}$ of trellis codes to the m^{th}-dimensional received symbol, $y_{n,m}$.
- $B_{i,j,n,m} \equiv m^{th}$-dimensional branch metric for the transition from state i to state j at decoding stage n. When $m = 1$, $B_{i,j,n,m} = B_{i,j,n}$.
- $M_{j,n} \equiv$ the path metric for state j at decoding stage n
- $\{i\} \equiv$ the set of states that have transitions to state j

At each decoding stage, the VA computes the path that has the minimum path metric coming into each state [5]. For example, to find the survivor path coming into state j at decoding stage n, the VA computes

$$M_{j,n} = \min_{\{i\}}[M_{i,n-1} + B_{i,j,n}]. \tag{7}$$

To determine the survivor path, comparison of the path metrics of the path from state i to state j and that of state k to state j has to be made. That is,

$$\begin{aligned}
\Delta M &= M_{i,n-1} + B_{i,j,n} - (M_{k,n-1} + B_{k,j,n}) \\
&= (M_{i,n-1} - M_{k,n-1}) + \textstyle\sum_{m=1}^{N} B_{i,j,n,m} - \sum_{m=1}^{N} B_{k,j,n,m} \\
&= (M_{i,n-1} - M_{k,n-1}) + \\
&\quad \textstyle\sum_{m=1}^{N} (y_{n,m} - p_{i,j,m})^2 - (y_{n,m} - p_{k,j,m})^2 \\
&= (M_{i,n-1} - M_{k,n-1}) + \\
&\quad \textstyle\sum_{m=1}^{N} -2y_{n,m}(p_{i,j,m} - p_{k,j,m}) + p_{i,j,m}^2 - p_{k,j,m}^2 \\
&= M_{i,n-1} - M_{k,n-1} + \\
&\quad \textstyle\sum_{m=1}^{N} (p_{k,j,m} - p_{i,j,m}) \left[(y_{n,m} - p_{i,j,m}) + (y_{n,m} - p_{k,j,m}) \right]
\end{aligned} \tag{8}$$

The comparison rule is

$$\Delta M \begin{cases} \geq 0 & \text{for } (M_{i,n-1} + B_{i,j,n}) \geq (M_{k,n-1} + B_{k,j,n}) \\ < 0 & \text{for } (M_{i,n-1} + B_{i,j,n}) < (M_{k,n-1} + B_{k,j,n}). \end{cases} \tag{9}$$

We apply the properties derived in Equation 8 to convolutional codes, trellis codes [9] of depth 2 (C-level partitioning) and 3 (D-level partitioning) that are of practical interest. The results are summarized below. Detailed derivation can be found in [10].

Case I: QPSK constellations

For a QPSK constellation, we can represent [6]

Branch metric per dimension for a QPSK symbol:

$$B_{i,j,n,1} = -y_{n,1}p_{i,j,1} = \begin{cases} -y_{n,1} & \text{for } p_{i,j,1} = 1 \\ y_{n,1} & \text{for } p_{i,j,1} = -1 \end{cases} \quad (10)$$

To search for the survivor path, compute
Comparison of path metrics in the QPSK case:

$$\begin{aligned} \Delta M &= M_{i,n-1} + \textstyle\sum_{m=1}^{2} B_{i,j,n,m} - \\ &\quad (M_{k,n-1} + \textstyle\sum_{m=1}^{2} B_{k,j,n,m}). \end{aligned} \quad (11)$$

Since the input symbol y_n for each state is different, the factor y_n^2 has to be included in the path metric of the survivor path for each state. Thus, if the path from state i to state j is the optimum path coming into state j at decoding stage n, the updated path metric should be, **Path metric update for Case I:**

$$M_{j,n} = M_{i,n-1} + B_{i,j,n} + y_n^2. \quad (12)$$

Case II: Trellis codes with infinite constellations
In this case, we assume that the signal constellation is infinite in all dimensions. To determine the survivor path, compute
Comparison for Case II:

$$\begin{aligned} \Delta M &= (M_{i,n-1} - M_{k,n-1}) + \\ &\quad \textstyle\sum_{m=1}^{N} |y_{n,m} - p_{i,j,n,m}| - \sum_{m=1}^{N} |y_{n,m} - p_{k,j,n,m}|. \end{aligned} \quad (13)$$

That is, absolute distances can be used to represent the branch metrics without loss of performance. Thus, **Branch metric representation for Case II during comparison:**

$$B_{i,j,n} = \sum_{m=1}^{N} |y_{n,m} - p_{i,j,n,m}|. \quad (14)$$

To update the path metrics, if the path from state i to state j is the optimum path coming into state j at decoding stage n, **Path metric update for codes using C-level partitioning (infinite constellations):**

$$M_{j,n} = (M_{i,n-1} + B_{i,j,n}) + y_n^2. \quad (15)$$

Path metric update for codes using D-level partitioning (infinite constellations):

$$M_{j,n} = M_{i,n-1} + (y_n - p_{i,j,n})^2. \quad (16)$$

Even though a squared branch metric has to be computed in this case, it needs to be computed only once per state and it can be done in parallel to the add-compare-select process to search for the survivor path for each state. It is also removed from the critical path to search for the survivor path for each state.

Case III: Trellis codes with finite constellations
Figure 7 is a one-dimensional example of a typical trellis code with a finite number of constellation points. In Equation 8, the order of subtraction of the term $(p_{k,j,m} - p_{i,j,m})$ determines the sign. Given a code, such as the example shown in Figure 7, the constellation points are defined. Thus, the sign of subtraction can be derived [10]. In summary, referring to Figure 7, if C0 is associated with the transition from state i to state j and C1 is associated with the transition from state k to state j, and if $s_i = \text{sign}(y_n - p_{i,j,n})$, then to

Fig. 7. One-dimensional finite constellation used in Case III.

search for the survivor path
if $((s_i \oplus s_k) = 1$ and $s_i = 1)$ compute
$\Delta M = (M_{i,n-1} - M_{k,n-1}) - [(y_n - p_{i,j}) + (y_n - p_{k,j}$
else
$\Delta M = (M_{i,n-1} - M_{k,n-1}) + [(y_n - p_{i,j}) + (y_n - p_{k,j}$
Similar to other cases, to update the path metric, if the path from state i to state j is the optimum path coming into state j at decoding stage n, the updated path metric should be,
Path metric update for codes using C- and D-level partitioning (finite constellations):

$$M_{j,n} = M_{i,n-1} + (y_n - p_{i,j,n})^2. \quad (17)$$

References

[1] M. Eyuboglu and S. Qureshi, "Reduced-state sequence estimation for coded modulation on intersymbol interference channels," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 989–995, August 1989.

[2] K. Wesolowski, "Efficient digital receiver structure for trellis-coded signals transmitted through channels with intersymbol interference," *Electron. Letters*, pp. 1265–1267, November 1987.

[3] A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Trans. on Communications*, vol. 37, pp. 428–436, May 1989.

[4] H. Lou, H. Viswanathan, and R. Krishnamoorthy, "Performance and VLSI architectures of Parallel Decision-Feedback Decoders for high bit-rate applications," *Lucent Technologies Bell Labs Technical Memorandum*, 1998.

[5] G. D. Forney, Jr., "The Viterbi algorithm," *IEEE Proceedings*, vol. 61, pp. 268–278, March 1973.

[6] H. Lou, "Implementing the Viterbi algorithm. Fundamentals and real-time issues for processor designers," *IEEE Signal Processing Magazine*, vol. 12, pp. 42–52, September 1995.

[7] S. Aryavisitakul and G. Durant, "A broadband wireless packet technique based on coding, diversity, and equalization," *IEEE Communications Magazine*, pp. 110–115, July 1998.

[8] M. Rupp and H. Lou, "On an efficient multiplier-free implementation for channel estimation and equalization," *Lucent Technologies Bell Labs Technical Memorandum*, 1999.

[9] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets: Parts I and II," *IEEE Communications Magazine*, vol. 25, February 1987.

[10] H. Lou and R. Urbanke, "Linear Euclidean distance as branch metrics for high bit-rate Viterbi decoder implementation," *Lucent Technologies Bell Labs Technical Memorandum*, 1999.