Applications of Algebraic Soft-Decision Decoding of Reed–Solomon Codes

Warren J. Gross, *Member, IEEE*, Frank R. Kschischang, *Fellow, IEEE*, Ralf Koetter, *Member, IEEE*, and
P. Glenn Gulak, *Senior Member, IEEE*

Abstract—Efficient soft-decision decoding of Reed–Solomon (RS) codes is made possible by the Koetter–Vardy (KV) algorithm which consists of a front-end to the interpolation-based Guruswami–Sudan (GS) list decoding algorithm. This paper approaches the soft-decision KV algorithm from the point of view of a communications systems designer who wants to know what benefits the algorithm can give, and how the extra complexity introduced by soft decoding can be managed at the systems level. We show how to reduce the computational complexity and memory requirements of the soft-decision front-end. Applications to wireless communications over Rayleigh fading channels and magnetic recording channels are proposed. For a high-rate RS(255,239) code, 2–3 dB of soft-decision gain is possible over a Rayleigh fading channel using 16-quadrature amplitude modulation. For shorter codes and at lower rates, the gain can be as large as 9 dB. To lower the complexity of decoding on the systems level, the *redecoding* architecture is explored, which uses only the appropriate amount of complexity to decode each packet. An error-detection criterion based on the properties of the KV decoder is proposed for the redecoding architecture. Queueing analysis verifies the practicality of the redecoding architecture by showing that only a modestly sized RAM buffer is required.

Index Terms—List decoding, Reed–Solomon (RS) codes, soft-decision decoding (SDD).

I. INTRODUCTION

REED–SOLOMON (RS) codes are found in many digital communications and recording systems. They are powerful error-correcting codes whose symbols are chosen from a Galois field (GF), GF(q). Their nonbinary nature makes them particularly suitable to correct error bursts, say, produced by an inner decoder of a concatenated code. However, the traditional algebraic decoders are not able to use *soft* reliability information that is readily available from the output of maximum *a posteriori* (MAP) or turbo decoders. The recent introduction of an efficient *algebraic* soft-decision decoding (SDD) algorithm due to Koetter and Vardy [1], [2], based on the list decoding algorithm of Guruswami and Sudan [3], [4], has sparked a renewed interest in soft-decision decoders.

This paper approaches the soft-decision Koetter–Vardy (KV) algorithm from the point of view of a communications systems designer who wants to know what benefits the algorithm can give, and how the extra complexity introduced by soft decoding can be managed at the systems level. Details on efficient implementations of the list decoding algorithm itself are given in [5]–[11]. Section II is a brief review of the algebraic SDD algorithm. In Section III, we show how the complexity of converting the soft information into algebraic conditions needed by the decoder can be reduced, and also how the memory requirements for this information can be lowered to a reasonable level. Section IV gives simulation results for proposed applications in wireless communications and magnetic recoding. Section V examines a systems-level architecture designed to minimize the average complexity of decoding by only using soft information when it is needed. Conclusions are offered in Section VI.

II. ALGEBRAIC SOFT-DECISION DECODING

RS codes are nonbinary linear block codes over GF(q) of length n and dimension k with minimum distance $d_{\min} = n - k + 1$, denoted as RS(n, k). A k symbol message is represented by the coefficients of a degree $k - 1$ message polynomial $f(x)$, and a length $n = (q - 1)$ RS codeword is formed by evaluating $f(x)$ at the nonzero elements of GF(q)

$$c = (f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{q-2})) \tag{1}$$

for some fixed ordering of the n field elements $\{\alpha_i\}$. This evaluation map view of RS codes leads to the Guruswami–Sudan (GS) list decoding algorithm [3], [4], which can decode beyond the traditional error-correction bound of half the minimum distance. The KV algorithm [1], [2] is a soft-decision extension to the GS algorithm, which encodes soft information into algebraic conditions used to control an interpolation in the GS algorithm. The transmitted symbols are elements of the finite field GF(q) = $\{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{q-1}\}$. At the output of a memoryless channel, we can tabulate the *a posteriori* probabilities (APPs) for each of the q possible transmitted symbols, given the observation β_j received corresponding to symbol position j as

$$\pi_{i,j} = \Pr(\alpha_i \text{ sent} \mid \beta_j \text{received}). \tag{2}$$

The $(q \times n)$ values of $\pi_{i,j}$ can be written as the elements of a $(q \times n)$ matrix Π called the *reliability matrix*, which represents

the full soft information available from the channel. In practice, APPs are readily available from the output of MAP or turbo decoders.

The GS algorithm decodes by finding a minimal bivariate polynomial that passes through the received data. The KV algorithm computes weights (or multiplicities) proportional to the soft information. These weights, stored in a *multiplicity matrix*, are used to control a biased interpolation in the GS algorithm.

Definition 1: The (a, b)-*weighted degree* of a bivariate polynomial $P(x, y) \in GF(q)[x, y] \setminus \{0\}$ is $\deg^{(a,b)} P(x, y) = \max\{au + bv p_{u,v} \neq 0\}$, where $a, b \in \mathbb{N}$.

If $P(x, y) = 0$, then $\deg^{(a,b)}(0) = -\infty$. Given soft information in the form of a $(q \times n)$ reliability matrix, Π, the KV algorithm is as follows.

1) Compute a $(q \times n)$ integer *multiplicity matrix M*, where the (i, j)th entry is denoted as $m_{i,j}$, from Π.
2) Find the bivariate interpolation polynomial $P(x, y)$ of minimal $(1, k - 1)$-weighted degree that, for each i, j, passes through the point (x_j, y_i) with multiplicity at least $m_{i,j}$ for every nonzero $m_{i,j}$.
3) Given the interpolation polynomial $P(x, y)$, identify all the factors of $P(x, y)$ of the form $y - f(x)$ with $\deg f(x) < k$. Produce a list of the codewords that correspond to these factors.
4) Choose the most likely output codeword by using the symbol probabilities from the reliability matrix or by using the Hamming distance from the hard-decision word (which can be derived from the reliability matrix). The output list usually contains only a single codeword, and therefore, this step is trivial, in practice.

The number of linear constraints, or *cost* of interpolation with multiplicity matrix M is [2]

$$C_M = \frac{1}{2} \sum_{i=1}^{q} \sum_{j=1}^{n} m_{i,j}(m_{i,j} + 1). \qquad (3)$$

We briefly summarize some important facts from [2]. The *score* of a vector with respect to a multiplicity matrix M is the inner product

$$S_M(v) = \langle M, [v] \rangle \qquad (4)$$

where $v = (v_0, v_1, \ldots, v_{n-1})$, and $[v]$ is a $q \times n$ matrix formed from the vector v, by setting $[v]_{i,j} = 1$, if $v_j = \alpha_i$, and 0, otherwise. The score represents how closely the matrices $[v]$ and M correspond. Define

$$\Delta_{1,k-1}(C_M) = \min\{\delta \in \mathbb{Z} : N_{1,k-1}(\delta) > C_M\} \qquad (5)$$

where

$$N_{1,k}(\delta) = \left\lceil \frac{\delta + 1}{k} \right\rceil \left(\delta - \frac{k}{2} \left\lfloor \frac{\delta}{k} \right\rfloor + 1 \right) \qquad (6)$$

which is the number of monomials whose $(1, k)$th weighted degree is at most δ. Then, if weighted interpolation is applied to

a multiplicity matrix M to produce an interpolation polynomial $P(x, y)$, the factorization of $P(x, y)$ will contain a factor corresponding to a codeword c if

$$S_M(c) > \Delta_{1,k-1}(C_M). \qquad (7)$$

Another important result from [2] is an algorithm, Algorithm A, for generating a multiplicity matrix M from a reliability matrix Π. The algorithm is optimal in the sense that it maximizes the score, subject to a constraint on the sum of the entries of M, $\sum_{i=0}^{Q-1} \sum_{j=0}^{n-1} m_{i,j} = s$. This gives the KV algorithm a tunable complexity parameter s to tradeoff performance with decoding complexity.

III. A REDUCED-COMPLEXITY SOFT-DECISION FRONT-END

In the previous section, we described how soft information, in the form of a reliability matrix Π, can be incorporated into an algebraic interpolation-based decoder by representing that information in a matrix of integer multiplicities, the multiplicity matrix M. Koetter and Vardy's solution to the problem of finding a matrix M subject to a constraint s on the sum of the entries in M is Algorithm A. In this section, we explore an alternative method of calculating a multiplicity matrix.

There are several implementation challenges in the soft-decision front-end. The first, independent of the actual algorithm used to compute M, is the size of the matrices involved. Both Π and M are $q \times n$ matrices (where $n = q - 1$ in this paper, but for extended RS codes, $n = q$). For example, if $q = 256$, then the reliability matrix contains $q \times n \approx 64$ k entries. If the entries are quantized to, say, 8 bits each, then this matrix requires $\approx$ 64 k bytes of storage. Then the multiplicity matrix also requires 64 k entries, with the number of bits per entry depending on the choice of M (for example, by setting the parameter s in Algorithm A). Considering the specifics of Algorithm A, we require a scratch matrix Π^*, which further increases the memory requirements. Providing this much storage on-chip in a very large-scale integration (VLSI) implementation can be challenging.

The other problem is the time complexity of Algorithm A. The algorithm requires s passes through a $q \times n$ matrix. We would like to avoid zero columns in the multiplicity matrix, so a necessary (but not sufficient) condition is $s \geq n$. Therefore, the number of memory accesses is at least $(q \times n \times s) = O(n^3)$. For example, if q is large, say 256, then 2^{24} memory accesses will be required. Storing, transferring, and processing this much information can quickly become a bottleneck, and this is just to decode a single codeword.

A. A Reduced-Complexity Method for Finding a Multiplicity Matrix

We will approach the problem by first tackling the time complexity of Algorithm A. The solution will then expose properties of the matrices from which further insight and optimizations can be derived. Intuitively, a multiplicity matrix M should be chosen proportional to the reliability matrix Π, and this is exactly what Algorithm A guarantees as $s \to \infty$. What if s is small and finite? It has been shown [2] that for every real number $\lambda \geq 0$,

there is a multiplicity matrix M_s generated by Algorithm A with parameter s such that

$$M_s = \lfloor \lambda \Pi \rfloor. \tag{8}$$

This suggests a way of generating multiplicity matrices by simply scaling the reliability matrix by a real number and truncating to get an integer matrix (which fits our intuition). However, this method is not exactly equivalent to Algorithm A. For reliability matrices that are quantized to a finite precision, the converse of (8) is not true; that is, there are some matrices that can be generated by Algorithm A that cannot be generated by any choice of λ in (8). An example of this is seen when there are two or more values in Π that are exactly equal. Note that for a Gaussian channel with infinite-precision quantization, two equal entries in Π will occur with zero probability and the two algorithms are equivalent. Algorithm 1 is a "one-pass" algorithm, that in practice, although not strictly equivalent to Algorithm A, always produces a valid multiplicity matrix.

Algorithm 1: Reduced complexity "one-pass" algorithm for calculating M from Π subject to complexity coefficient λ

---

for $j = 0$ to $n - 1$ **do**

    for $i = 0$ to $q - 1$ **do**

        $m_{i,j} \leftarrow \lfloor \lambda \pi_{i,j} \rfloor$

    end for

end for

This algorithm has a time complexity of $O(n^2)$, since it requires only a single pass through the reliability matrix. The performance/complexity tradeoff is now controlled by the complexity coefficient λ.

B. Reducing the Size of the Multiplicity Matrix

Now we will tackle the space complexity of the multiplicity matrix. Although the matrix has $q \times n = O(n^2)$ entries, in practice, we find that multiplicity matrices are quite sparse. To illustrate this, we ran an experiment that measures the density of nonzero elements in a multiplicity matrix for an RS(255,239) code. The $q \times n$ multiplicity matrix for this code will have 65 280 entries. We would expect the most dense matrices to result from an unreliable channel with a very low E_b/N_0. This is because as the channel noise decreases, we have more reliable information about a smaller number of likely symbols. When the noise is small enough, SDD reduces to hard-decision decoding (HDD) with only one nonzero entry per column of M. We also expect to see the density increase as the value of the complexity coefficient λ increases. Simulations show that even for $\lambda = 256$ and $E_b/N_0 = 0.0$ dB, the multiplicity matrix is only 5% dense. At more practical operating points, such as $\lambda = 16$, $E_b/N_0 = 6.0$ dB, the density is about 0.5%. Clearly, storing the full matrix is a waste of resources.

We now will quantify just how dense a multiplicity matrix can get. Since the entries of the reliability matrix are probabilities,

the maximum possible entry in Π is 1.0. Therefore, the maximum possible entry in M is

$$m_{\max} = \lfloor \lambda \rfloor. \tag{9}$$

.

Lemma 1: Consider a $(q \times n)$ reliability matrix Π and a $(q \times n)$ multiplicity matrix $M = \lfloor \lambda \Pi \rfloor$, where λ is a nonnegative real number. The sum of the entries in any given column of M is less than or equal to λ.

Proof: The sum of the entries in any given column j of M is $\sum_i m_{i,j} = \sum_i \lfloor \lambda \pi_{i,j} \rfloor \leq \lambda \sum_i \pi_{i,j}$. But since the $\pi_{i,j}$ are probabilities, $\sum_i \pi_{i,j} = 1$, and therefore, $\sum_i m_{i,j} \leq \lambda$. ∎

Corollary 2: The maximum number of nonzero entries in any given column of a $(q \times n)$ multiplicity matrix M is $\min(m_{\max}, q)$.

Proof: Since the entries $m_{i,j}$ are nonnegative integers, the maximum number of nonzero $m_{i,j}$ occurs when all the nonzero entries in a column are $m_{i,j} = 1$. This means that the maximum number of nonzero entries in a column is $m_{\max} = \lfloor \lambda \rfloor$, up to the maximum number of symbols in the alphabet, q. ∎

Corollary 3: The maximum number of nonzero entries in a $(q \times n)$ multiplicity matrix M is $N_M = n \times \min(m_{\max}, q)$.

Proof: The result follows from *Corollary 2* and the fact that there are n columns in M. ∎

The maximum possible density for a multiplicity matrix is therefore

$$D = \frac{\min(m_{\max}, q)}{q}. \tag{10}$$

We will see in the next section that the values of λ we will choose, and hence, $m_{\max}$ are low, usually between 4 and 16. In light of the low-density M matrices that result, a reduced data structure called a *compressed multiplicity matrix* $M^*_{m_{\max}}$ can be stored instead of M. The matrix consists of n columns, but only $m_{\max}$ rows. Column labels are implicit, but a label has to be stored for each row. Therefore, the number of bits of storage required for $M^*_{m_{\max}}$ are

$$N_{b,M} = (\lceil \log_2(q) \rceil + \lceil \log_2(m_{\max} + 1) \rceil) \times \min(m_{\max}, q) \times n \tag{11}$$

and the compression ratio is

$$\Upsilon = \left(\left(1 + \frac{\lceil \log_2(q) \rceil}{\lceil \log_2(m_{\max} + 1) \rceil} \right) D \right)^{-1}. \tag{12}$$

At a certain point, there is a tradeoff between the smaller number of entries and the extra label bits. For the range of $m_{\max}$ that we will be interested in, we can achieve compression ratios of at least six, as shown in Table I. It should be noted that simulations show, in practice, only a small number of nonzero entries in each column, usually two, suffice to achieve essentially all the soft-decision gain. Therefore, in practice, the compression ratio can be improved by an additional factor of two to eight.

TABLE I
SIZE OF THE COMPRESSED MULTIPLICITY MATRIX IN BITS $N_{b,M}$, AND
THE COMPRESSION RATIO Υ, FOR TYPICAL VALUES OF THE MAXIMUM
MULTIPLICITY $m_{\max}$ FOR AN RS$(255, k)$ CODE

m_{max}	2	4	6	8	10	16
$N_{b,M}$	5100	11220	16380	24480	30600	53040
Υ	25.6	17.4	11.6	10.6	8.5	6.1

C. Interpolation Cost of a Multiplicity Matrix

We now turn to investigate the cost of interpolation for a multiplicity matrix M. The maximum entry in a column of M is $m_{\max}$, and the sum of all the entries in that column must be at most $m_{\max}$. How does the distribution of multiplicities in the column affect the cost of interpolation? For a given symbol position, the most reliable possible scenario gives a corresponding column in M with a single entry of $m_{\max}$. As the reliability of a position decreases, the maximum multiplicity in the column will begin to decrease and other nonzero entries might arise, as long as the sum of the column entries does not exceed $m_{\max}$. We show in the following theorem that the largest possible interpolation cost for a given column arises when there is a single entry of value $m_{\max}$.

Theorem 4: Consider a $(q \times n)$ reliability matrix Π and a $(q \times n)$ multiplicity matrix $M = \lfloor \lambda \Pi \rfloor$, where λ is a nonnegative real number. The cost of the interpolation step for SDD with multiplicity matrix M is at most the cost of interpolation for HDD with the GS algorithm with a fixed multiplicity $m = \lfloor \lambda \rfloor$.

Proof: We will use an exchange argument to show that grouping multiplicities in one entry always produces a higher cost than distributing them amongst several entries. Consider a column $\mathbb{A} = (m_0, m_1, \ldots, m_{q-1})^T$ of M with at least one nonzero entry, and pick two entries m_i and m_j in $\mathbb{A}$ such that $m_i \geq m_j$ and $m_j > 0$. Construct a length-q column vector $\mathbb{B}$ whose lth entry is $\mathbb{B}_l = m_l + 1$, if $l = i, m_l - 1$, if $l = j$, and m_l otherwise, for $0 \leq l \leq q-1$. The difference in the cost of columns $\mathbb{A}$ and $\mathbb{B}$ is $m_i(m_i+1) + m_j(m_j+1) - ((m_i+1)(m_i+2) + (m_j-1)m_j) = m_j - m_i - 1$. But $m_i \geq m_j$, therefore the difference is less than zero, which means that the cost is always increased. It follows that the maximum cost in a column occurs when all the multiplicities are in the same row. The maximum cost in a column occurs when there is a single entry of value $m_{\max}$, and the maximum cost due to an n-column multiplicity matrix is $(n/2)(m_{\max})(m_{\max}+1)$, which is exactly the cost of GS hard decoding with fixed multiplicity $m_{\max}$. ∎

D. Reducing the Size of the Reliability Matrix

Since the multiplicity matrix is usually quite sparse, it makes little sense to store a full reliability matrix, as most of its entries will never result in a corresponding nonzero multiplicity. *Corollary 2* places a strict limit on the maximum number of entries in a column of a reliability matrix that will ever be useful. Therefore, it is not necessary to provide a full reliability matrix, but just a compressed version with only the $\min(m_{\max}, q)$ most likely symbols per column. One way of doing this is to exploit
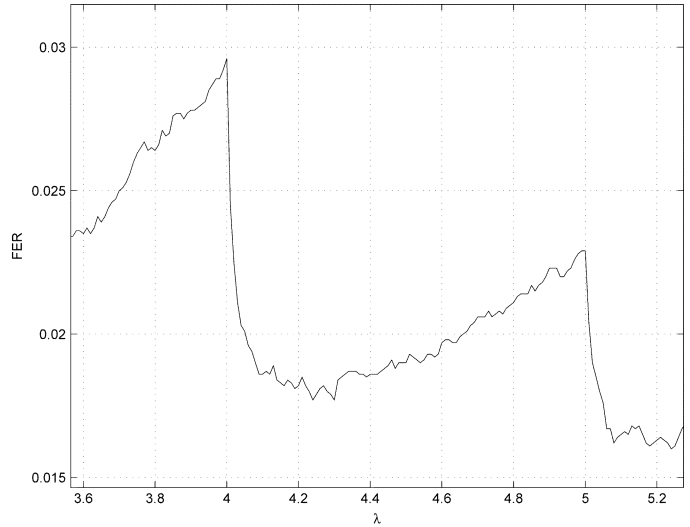

Fig. 1. FER for an RS(15,11) code plotted with respect to λ. The period of the local fluctuation of the FER with λ is 1.0.

the geometry of modulation schemes such as quadrature amplitude modulation (QAM), where each constellation point maps to a symbol from the finite field GF(q). The most likely symbols will be those with the smallest Euclidean distance to the received point. Storage in the reliability matrix has to be provided only for those constellation points in a local neighborhood around the received point. If a compressed reliability matrix is available, the one-pass algorithm has to only consider $\min(m_{\max}, q) \times n$ entries. If $m_{\max}$ is small, then the number of computations is $m_{\max} \times n = O(n)$.

E. Choosing the Complexity Coefficient

The frame-error rate (FER) of a KV decoder is not a monotonic decreasing function of the value of the complexity coefficient λ (or the complexity parameter s), leading to "good" and "bad" local choices of s. In general, a large increase in λ (or s) will result in a lower FER; however, there is no guarantee that small increases will lower the FER. In certain cases, the behavior of the FER is periodic. Fig. 1 shows the local variation of the FER with λ for an RS(15,11) code. The maxima occur at integer values of λ with a minimum somewhere between the two. To explain this phenomenon, consider an interval between two integer values $\lambda = [\phi \ldots \phi + 1)$, as illustrated in Fig. 1, where $\phi = 4$. The maximum possible multiplicity is $m_{\max} = \lfloor \phi \rfloor$. At $\lambda = \phi$, the only way a multiplicity $m_{\max}$ would occur in M is if there was an entry $\pi_{i,j} = 1.0$, which is extremely unlikely. As λ is increased, the probability of an entry of $m_{\max}$ increases, and then we begin to see the benefit of interpolation with higher multiplicities. The FER begins to go down until a point where the multiplicity matrix best models the reliability matrix. After a certain point, because of the coarseness of the multiplicity matrix, M does not accurately model Π anymore and the FER begins to rise. In this regime, entries in M other than the largest ones begin to rise, but the largest ones cannot become greater than $m_{\max}$. Finally, when $\lambda = \phi + 1$, the largest entries can turn to $m_{\max} + 1$ and the FER begins to decrease again. Fig. 2 illustrates this behaviour for an RS(255,239) code.
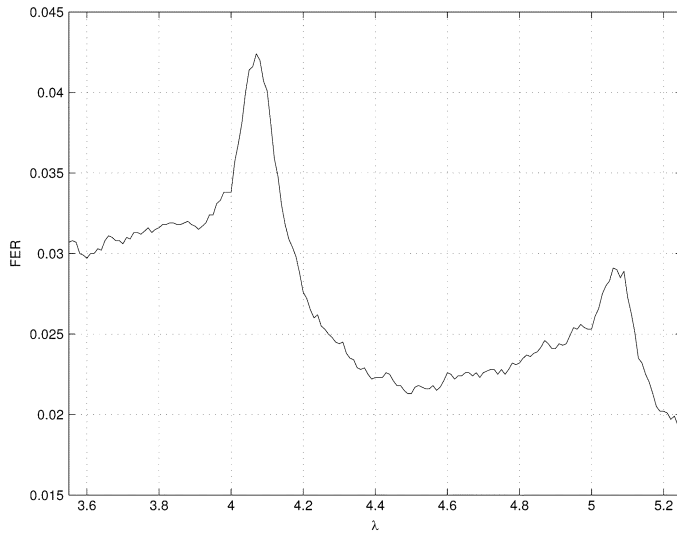
Fig. 2. FER for an RS(255,239) code plotted with respect to λ. The period of the local fluctuation of the FER with λ is 1.0.

Fig. 3. Simulation of a (255,239) RS code with binary phase-shift keying (BPSK) modulation over an AWGN channel.

IV. SIMULATION RESULTS

In this section, we explore the amount of soft-decision gain that the KV algorithm provides on different channels. In Section IV-A, the soft-decision gain achievable on additive white Gaussian noise (AWGN) channels is studied. Section IV-B gives results for an application over a magnetic recording channel. Section IV-C considers an application to wireless communications over Rayleigh fading channels. The multiplicities are calculated by Algorithm 1. It has been shown in [12] that there are multiplicity assignment schemes that give gains over the algorithm in this paper of about 0.20 dB for long RS codes (length 255), and 0.75 dB for RS codes of length 15 on AWGN channels.

A. AWGN Channels

In order to speed up the simulations, all of the simulations are performed according to (7). Random data is generated, encoded, and transmitted over a channel. Then a reliability matrix is calculated, and the one-pass algorithm is applied to get a multiplicity matrix. The score is calculated from the multiplicity matrix and knowledge of the transmitted codeword, and then the threshold of (7) is applied. If the threshold condition is satisfied, then successful decoding is guaranteed. Since the decoding could still be successful otherwise, these simulation results might be slightly pessimistic. Simulations show that the estimated performance matches the actual decoder performance very closely. Hybrid simulations are possible where the full decoder is only employed on failure of the threshold condition.

Fig. 3 shows the performance of the KV algorithm for a very common high-rate (255,239) RS code. We see that for very high complexities, a maximum soft-decision gain of 0.47 dB can be achieved. For reasonable complexities, say with $m_{\max} = 4$, a gain of 0.27 dB is achieved at an FER of 2×10^{-4}.

B. Magnetic Recording Channels

To investigate the performance of the KV algorithm on magnetic recording, we examine a concatenation of a PR4 partial-re-
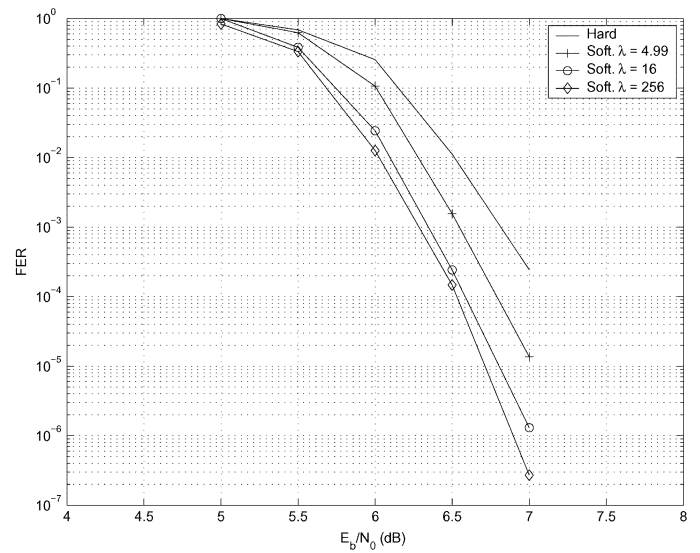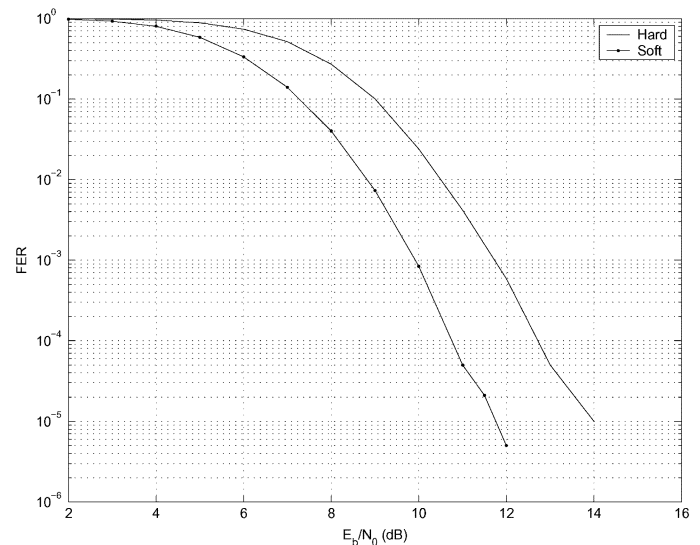

Fig. 4. Simulation of an RS(15,7) code over a PR4 partial-response channel. The decoder is a concatenation of a soft-output log-MAP algorithm and a KV RS decoder with $\lambda = 16.99$.

sponse channel and a (15,7) RS code. The log-MAP algorithm was used as a soft-detection algorithm for the PR4 channel to provide soft information to the KV algorithm. The complexity coefficient was set to $\lambda = 16.99$. From the simulation results in Fig. 4, the soft-decision gain is 2 dB at an FER of 10^{-5}. Recent results for magnetic recording channels by other authors can be found in [13]–[15].

C. Rayleigh Fading Channels

We also investigated the performance of the KV algorithm over a Rayleigh fading channel with 16-QAM. Fig. 5 shows the performance of a (15,11) RS code. Four-bit symbols from GF(16) are mapped directly to 16-QAM constellation points. The multiplicative fading factors are independent, simulating the effect of an ideal interleaver. The reliability matrix is cal-
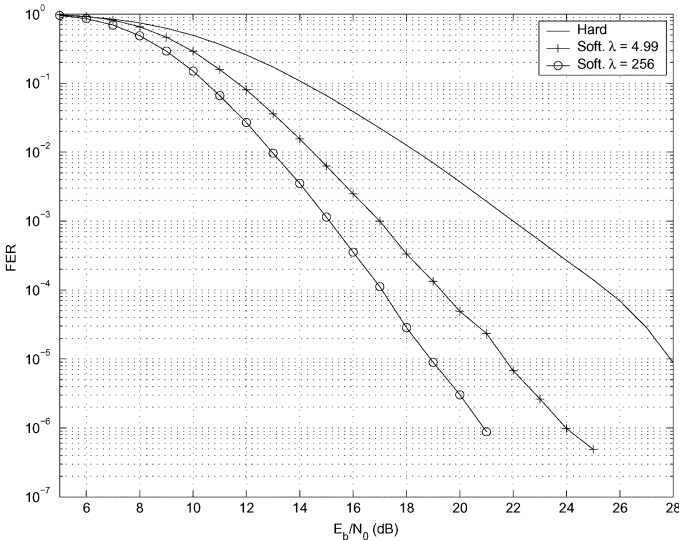
Fig. 5. Simulation of a (15,11) RS code with 16-QAM over a Rayleigh fading channel.

culated directly from the received soft information, assuming perfect channel state information (CSI). We see that much larger gains are realized on a fading channel. At an FER of 10^{-5}, the soft-decision gain for a (15,11) code is 5 dB for $\lambda = 4.99$ and 9 dB for $\lambda = 256$.

A simulation setup for a (255,191) RS code is shown in Fig. 6. Each eight-bit symbol of GF(256) is split into two four-bit symbols, and two 16-QAM channel uses are needed to transmit the symbol.

For 16-QAM, each constellation point can be mapped to an element of GF(16), $\alpha_j, j = 0, \ldots, 15$. Define this mapping by the functions C_x and C_y, which map an integer index in the range of 0–15 to the real-valued x and y coordinates of the corresponding constellation point, respectively. The received points are $(r_{x,1}, r_{y,1})$ and $(r_{x,2}, r_{y,2})$ corresponding to the two 16-QAM channel uses. Assuming perfect CSI, we know the fading factor γ and assume that the channel is varying slowly enough such that it is the same for both $(r_{x,1}, r_{y,1})$ and $(r_{x,2}, r_{y,2})$. Then

$$P(\alpha_j | \{(r_{x,1}, r_{y,1}), (r_{x,2}, r_{y,2})\}) = \frac{f_{1,j} f_{2,j}}{\sum_j f_{1,j} f_{2,j}} \quad (13)$$

where

$$f_{1,j} = \frac{1}{2\pi\sigma^2} e^{\left(\frac{-1}{2\sigma^2}((r_{x_1} - \gamma c_x(j_{3,\ldots,0}))^2 + (r_{x_2} - \gamma c_y(j_{3,\ldots,0}))^2)\right)} \quad (14)$$

$$f_{2,j} = \frac{1}{2\pi\sigma^2} e^{\left(\frac{-1}{2\sigma^2}((r_{x_1} - \gamma c_x(j_{7,\ldots,4}))^2 + (r_{x_2} - \gamma c_y(j_{7,\ldots,4}))^2)\right)} \quad (15)$$

where $j_{7,\ldots,4}$ and $j_{3,\ldots,0}$ are the integers whose binary representations are the most significant four bits and the least significant four bits of the binary representation of j, respectively.

The simulation results are plotted in Fig. 7. The soft-decision gain is 2.4 dB for $\lambda = 4.99$ and 3.1 dB for $\lambda = 256$ at FER $= 10^{-4}$. We note that the coding gain on a Rayleigh channel

is not constant, but increases as the SNR increases, since the two curves diverge. The results given here for a high FER will improve as the SNR increases.

V. SYSTEMS-LEVEL ARCHITECTURE

Section III showed ways to lower the implementation complexity of the KV soft-decision front-end. In this section, the complexity of SDD is considered from a systems level. The basic idea is that SDD is only needed to correct a small fraction of the received packets, and therefore, should only be used when needed. The implications are that for software implementations, the average cost of decoding is reduced. For hardware architectures, the speed requirements on the soft decoder are relaxed. This idea was presented for binary linear block codes in [16]. A similar idea has recently been presented in [17].

A. Redecoding Architecture

Given a fixed-complexity soft-decision decoder, the idea is to reduce overall system complexity by using the expensive soft decoder as little as possible. Most of the time, a hard-decision decoder is capable of correcting a received word. For example, a hard-decision decoder for the RS(255,239) code transmitted over an AWGN channel at $E_b/N_0 = 6.8$ dB, has an FER $\approx 10^{-3}$. This means that 99.9% of all frames are decoded correctly. A soft-decision decoder can achieve an FER of 3×10^{-6}, almost three orders of magnitude lower, but is really only needed once every thousand frames.

To implement this idea, we propose the *redecoding* architecture which only uses as much complexity as is necessary to correctly decode a codeword. Central to this scheme is an error-detection test which can be applied to the output of a KV decoder. A cyclic redundancy check (CRC) code can be used as recently proposed in similar work by Xia in [17]. To avoid the loss of rate in using a CRC, we propose an alternative error-detection technique in Section V-C.

The *L-stage redecoding architecture* does an initial first pass using a low-complexity decoder which could be a hard-decision decoder, or in general, a KV decoder with a small value of the complexity coefficient λ. Then a test is applied to the output to determine if particular acceptance criteria are met. If so, then the output is judged to be "reliable" and we are done. If deemed "unreliable," another trial with a soft-decision decoder of higher complexity is performed. Decoding can continue for an arbitrary number of stages L. If the test is able to pick out most of the errors at every stage, then only a very small number of frames will need to be passed all the way down to the later, more complex stages. The *average* complexity of decoding is, therefore, much lower than applying a high-complexity decoder to every frame, and the error rate can approach the error rate achievable at the higher complexity.

Define $P_{u,i}$ as the probability that the test at stage i declares a frame unreliable. By definition, $P_{u,0} = 1$. The average complexity of decoding N frames is, therefore

$$\bar{C} = NC_1 + NP_{u,1}C_2 + NP_{u,1}P_{u,2}C_3 + \cdots$$
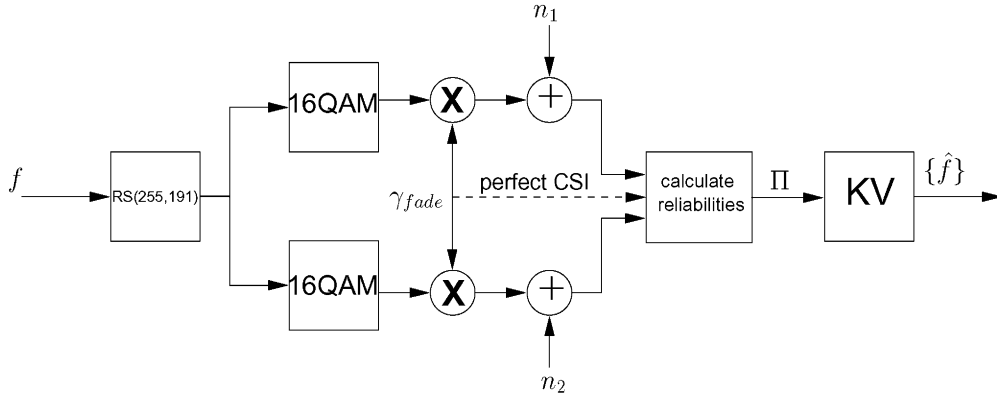$$+ NP_{u,1}P_{u,2} \ldots P_{u,L-1}C_L \quad (16)$$

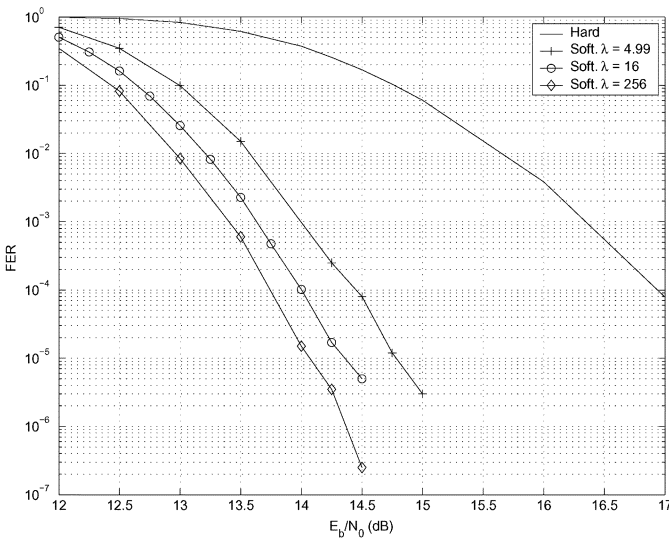Fig. 6. Simulation setup for decoding the RS(255,191) code over a Rayleigh fading channel with 16-QAM.

Fig. 7. Simulation of a (255,191) RS code with 16-QAM over a Rayleigh fading channel.

where C_i is the average complexity of decoding at stage i. The average cost per frame is, therefore

$$\bar{C}_f = \sum_{i=1}^{L} \left(\prod_{j=0}^{i-1} P_{u,j} \right) C_i. \tag{17}$$

As the coefficients in front of the complexities $\{C_i\}$ are products of probabilities, the higher complexities contribute only a small amount to the overall complexity, as long as the error rates are not too high.

The FER achievable by redecoding can be slightly higher than without, because of the possibility that the test at each stage will not detect an unreliable frame. Therefore, it is not desirable to have too many stages of redecoding. The achievable FER is FER $= \text{FER}_{\lambda_L} + P_{\lambda_1,...,\lambda_{L-1}}$, where FER_{λ_L} is the FER of decoding just with a decoder with λ_L, and $P_{\lambda_1,...,\lambda_{L-1}}$ is the error rate due to missed unreliable frames in the redecoding stages. Note that we only count decoding errors and not decoder failures in $P_{\lambda_1,...,\lambda_{L-1}}$ as a failure is always detectable. Then, in order for a small redecoding loss, we require that $P_{\lambda_1,...,\lambda_{L-1}} \ll \text{FER}_{\lambda_L}$ or $P_{\lambda_1,...,\lambda_{L-1}} = \delta \text{FER}_{\lambda_L}$, where δ can be chosen arbitrarily small.

For example, one or two orders of magnitude ($\delta = 10^{-1}$ or 10^{-2}) is a good rule of thumb. To determine $P_{\lambda_1,...,\lambda_{L-1}}$, define $P_{\text{miss},i}$ as the fraction of errors at stage i that were missed by the test. Then

$$P_{\lambda_1,...,\lambda_{L-1}} = \sum_{i=1}^{L-1} P_{\text{miss},i} \left(\prod_{j=1}^{i-1} P_{u,j} \right). \tag{18}$$

B. Differential Interpolation

The interpolation polynomial does not have to be computed from scratch at each stage, as the intermediate interpolation results from a stage can be used as the starting point for the next stage. The full factorization algorithm still has to be run at each stage. To use the intermediate results, the full set of scratch polynomials has to be supplied to the next stage, along with a copy of the previous multiplicity matrix M_{i-1}. At each point, if the multiplicity satisfied in the previous stage is m and the new multiplicity is $m + \Delta$, the new interpolation has to run for an additional $(N/2)(\Delta^2 + 2m\Delta + \Delta)$ iterations, where N is the number of interpolation points. The number of scratch polynomials needed is the total number that will be required in the last stage. Therefore, this *differential interpolation* architecture is only practical if the difference between the two multiplicity matrices is not large, or in hardware implementations where the scratch polynomials are updated in parallel.

C. An Acceptance Test for Detecting Errors

In order to implement the redecoding architecture, an appropriate acceptance test is needed. Error detection by CRC is proposed in [17]. A drawback of using a CRC is the loss of rate. In this section, we propose an acceptance test based only on the properties of the decoder which does not require any external code.

We propose a heuristic method for detecting decoder errors that are not flagged as decoder failures. After transmitting the codeword c through a channel, the soft-decision front-end is applied to the corresponding reliability matrix to generate a multiplicity matrix M with cost C_M. Then, from (7), the KV decoding is guaranteed to be successful if the score of c with respect to M is greater than a threshold $S_M(c) > \Delta_{1,k-1}(C_M)$.

We do not have knowledge of the codeword c at the decoder, but we do have the decoded codeword $\bar{c}$, which is an estimate of the actual codeword. Substituting in $\bar{c}$ for c, we have the heuristic acceptance criterion $\langle M, [\bar{c}] \rangle > \beta$, where β is an adjustable threshold. The overall acceptance test including the decoder failure condition is given in Algorithm 2.

Algorithm 2: Acceptance test for a decoder output $\bar{c}$ given a multiplicity matrix M

---

if the decoder declares a decoder failure **then**

    declare $\bar{c}$ **UNRELIABLE**

else if $\langle M, [\bar{c}] \rangle > \beta$ **then**

    declare $\bar{c}$ **RELIABLE**

else

    declare $\bar{c}$ **UNRELIABLE**

end if

The complexity of the acceptance test is the complexity of calculating the inner product $O(n)$, since there are only n nonzero entries in $[\bar{c}]$. Note that, in general, β can be calculated as a function of $\Delta_{1,k-1}(C_M)$.

Example 1: To determine the complexity and error performance of a redecoding architecture for an RS(255,239) code, the characteristics of the acceptance test need to be studied. A receiver operating characteristic (ROC) [18] plots the *miss rate* P_{miss_i} versus the *false alarm rate*. The miss rate is the fraction of frames that contain errors but were not detected by the test. The false alarm rate is the fraction of correctly decoded codewords that were mistaken as unreliable by the test. The miss rate should be small to keep down the residual error rate at each stage, and the false alarm rate should be small to keep the amount of work transferred to the next stage low. As the threshold β is increased, we would expect the miss rate to decrease, accompanied by a rise in the false alarm rate. Fig. 8 is an ROC where the miss rate versus the false alarm rate is plotted for an RS(255,239) code decoded with a KV decoder with maximum multiplicity $m_{\mathrm{max}} = 4$. From the curve, the point corresponding to $\beta = 984$ has a miss rate of 1.87×10^{-8} and a false alarm rate of less than 0.3%.

We would like to implement a KV decoder, so that the performance is close to that of a decoder with $m_{\mathrm{max}} = 16$. The FER of this decoder is 8×10^{-5}. The complexity of a KV decoder with $m_{\mathrm{max}} = 16$ is C_{16}. We implemented this decoder in software using the fast decoding algorithm described in [5]–[9]. The measured average decoding time on a 2-GHz Pentium 4 was 469 ms. For $m_{\mathrm{max}} = 4$, the average decoding time was 2.73 ms. We note that this software is not highly optimized and that a faster implementation is possible.

To achieve a low average complexity, consider a two-stage redecoding architecture with an initial decoding with $m_{\mathrm{max}} = 4$, followed by a test, and then final decoding with $m_{\mathrm{max}} = 16$. The cost per frame of redecoding is

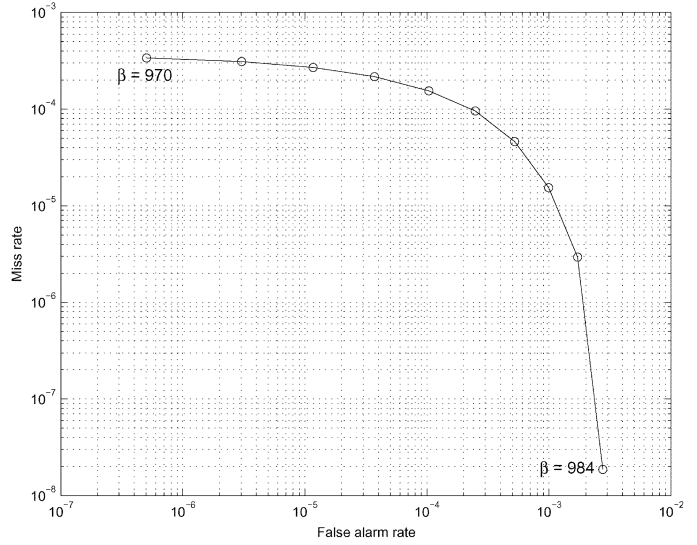$$\bar{c}_{f,\mathrm{redecode}} = C_4 + P_u C_{16}$$

Fig. 8. ROC for Algorithm 2 for an RS(255,239) code transmitted over an AWGN channel at $E_b/N_0 = 6.6$ dB and decoded with a KV decoder with $\lambda = 4.99$.

where P_u is the fraction of frames passed from the first decoder to the second (which depends on the false alarm rate), and was measured to be 3.15×10^{-3}. Therefore, the average cost per frame of redecoding is 4.21 ms, or about 1.54 times the cost of decoding with the first decoder alone. However, the cost of decoding without redecoding is 126 times as expensive. The FER is

$$\mathrm{FER} = \mathrm{FER}_{16} + P_{\mathrm{miss}}$$

where FER_{16} is the FER achievable by decoding with the second decoder alone, and P_{miss} is the miss rate of the first decoder, which was determined from the ROC to be 1.87×10^{-8}. Therefore, the overall FER is 8.002×10^{-5}, which represents only a very small redecoding loss. Therefore, relative to a decoder with $m_{\mathrm{max}} = 4$, we have achieved almost a 0.2-dB gain, approaching the performance of a decoder with $m_{\mathrm{max}} = 16$ very closely. However, a decoder with $m_{\mathrm{max}} = 16$ and not using redecoding is 126 times as expensive.

D. Hardware Architectures for High-Speed Implementations

For high-speed implementations, the redecoding architecture can be simplified to two stages comprising a hardware hard-decision decoder and a soft-decision coprocessor, as shown in Fig. 9. The acceptance test reduces to simply the Hamming distance between the received word and the decoded codeword. The average decoding time of a packet is

$$t = t_{\mathrm{hard}} + t_{\mathrm{test}} + P_u t_{\mathrm{soft}} \tag{19}$$

where t_{hard} is the average decoding time for the hard decoder, t_{test} is the average time to perform the test, t_{soft} is the average decoding time for the soft decoder, and P_u is the rate that the acceptance test declares packets unreliable, and sends them to the coprocessor. The decoder area required is the sum of the areas of the hard and soft decoders. However, since the soft decoder
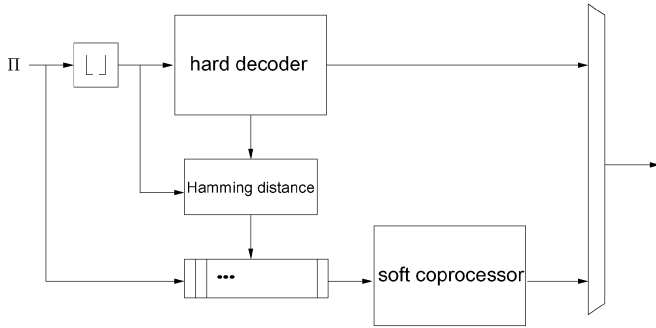
Fig. 9. Hardware implementation of SDD based on the redecoding architecture.

is only needed infrequently, the speed requirements are relaxed, and a lower area may be required than if a soft decoder was used at full speed. In [6]–[9], an implementation of the factorization algorithm is explored that uses the Berlekamp–Massey algorithm. Therefore, a Berlekamp–Massey algorithm can be implemented in hardware and shared between the hard and soft decoders to save area.

For high-speed, low-latency applications, such as in wireless communications, a hardware coprocessor can be used; however, for applications where latency is not as critical, such as magnetic recording, the frames requiring soft decoding can be handled by a software coprocessor.

E. Buffer Length

The redecoding architectures require a buffer at the input to the soft-decision coprocessor to hold any packets that arrive while it is busy processing another packet. In order for this architecture to be practical, the buffer size required has to be reasonably small. A straightforward queueing analysis can be used to determine the buffer size required.

The system consists of a GEO/D/1/K queue which has space to hold $(K-1)$ packets and a server (the soft coprocessor) which processes packets from the queue. Since the server has space for one packet, the total system capacity is K packets. Arrivals to the queue occur when the acceptance test determines that a decoding trial with the hard-decision decoder is unreliable. The discrete-time arrival process is a Bernoulli process with packets arriving at the input to the queueing system at *slot* boundaries (slots) at a rate of λ_a packets/s. The arrivals are considered to arrive immediately before the end of a slot, and service begins at the beginning of the next slot. This model is called *late arrivals with delayed access* [19]. In general, the service time depends on the cost of the multiplicity matrix corresponding to a packet, and can be specified by a probability distribution. However, we prefer a worst-case analysis, since *Theorem 4* gives an upper bound on the complexity of a multiplicity matrix for a given value of λ. The service period, which can only begin on a slot boundary, is given as T slots, where T is an integer.

If a packet arrives at the system and finds that it is full, then it is rejected and a decoding failure is declared by the soft coprocessor. Therefore, the buffer only needs to be large enough so that the failure rate of the coprocessor caused by rejected packets does not dominate the natural FER of the coprocessor. The buffer can be sized so that the rejection rate or *blocking*
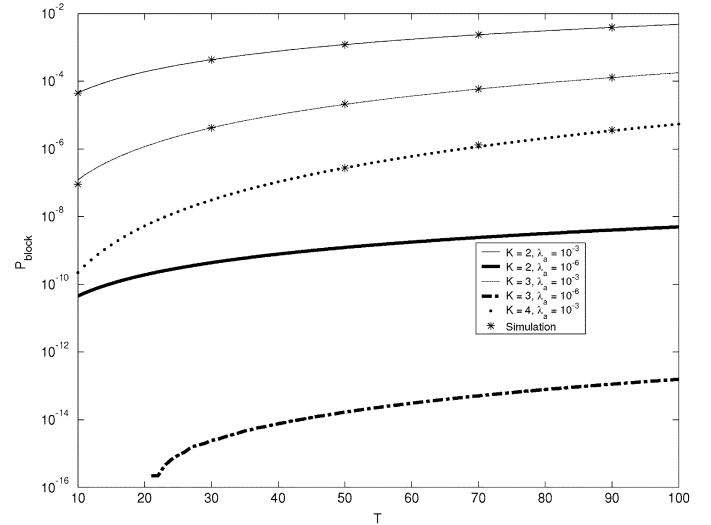

Fig. 10. Blocking probability for the GEO/D/1/K queue for different values of the arrival rate, service time, and queue length.

probability is several orders of magnitude lower than the FER, say, 10^3 or 10^4 times.

The blocking probability for this queue is given by Takagi in [19] as

$$P_{\text{block}} = 1 - (p_0 + \lambda_a T)^{-1} \qquad (20)$$

where p_0 is the probability that the queue is empty immediately after the completion of a service time. The complete probability distribution $\{p_m\}, 0 \le m \le K-1$ can be found by solving the system of equations

$$p_m = p_0 a_m + \sum_{j=1}^{m+1} p_j a_{m-j+1}, \quad 0 \le m \le K-2 \qquad (21)$$

$$\sum_{m=0}^{K-1} p_m = 1 \qquad (22)$$

where a_m is the probability of m packets arriving in a period of T slots, defined by

$$a_m = \binom{T}{m} \lambda_a^m (1-\lambda_a)^{T-m}. \qquad (23)$$

Fig. 10 gives results of the analysis for buffers lengths of one, two, and three $(K=2,3,4)$. Also plotted are the results of simulations of the queueing system which correspond very closely with the analytical model. A hard-decision decoder can be easily implemented at 100 Mb/s (for example, see [20]). As shown in [6] and [21], a software coprocessor can be implemented to decode at 1 Mb/s and a hardware coprocessor can be implemented at 10 Mb/s, corresponding to $T=100$ and $T=10$, respectively. If the goal is to have a blocking probability of three orders of magnitude lower than the FER, we see that in the most pessimistic case, where the arrival rate is 10^{-3} and the coprocessor is 100 times slower than the hard decoder, a queue of at most three packets is required. At lower FERs and if the coprocessor is only 10 times slower than the hard decoder, then a buffer as

small as one packet will suffice. If the Hamming distance is used to pick a word from the output list, then the multiplicity matrices can be stored in the queue, otherwise, reliability matrices need to be stored.

Example 2: Consider an RS(255,k) code. At one extreme, three compressed reliability matrices with 16 entries per column quantized to 8 bits would require 24 k bytes of storage. At the other extreme, storing a single multiplicity matrix with four entries per column would only require only 1.4 k bytes. It is, therefore, possible to implement the memory as on-chip SRAM.

VI. CONCLUSION

In this paper, we showed that the implementation complexity of algebraic SDD can be brought down to practical levels through system-level considerations. The strategy proposed is to limit the amount of soft information supplied to a decoder, and also to limit the use of the soft decoder to only when it is necessary. We also showed that significant soft-decision gains can be achieved using high-rate codes, even at moderate complexities. Algebraic SDD seems promising when applied to wireless communications over Rayleigh fading channels. A complete solution combines the systems-level optimizations with efficient decoder implementations. We anticipate efficient practical implementations of KV decoders using the coprocessor approach.

ACKNOWLEDGMENT

REFERENCES

[1] R. Kotter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2000, p. 61.

[2] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[3] M. Sudan, "Decoding of Reed-Solomon codes beyond the error correction bound," *J. Complexity*, vol. 13, no. 1, pp. 180–193, 1997.

[4] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1757–1767, Sep. 1999.

[5] W. J. Gross, F. R. Kschischang, R. Koetter, and P. Gulak, "A VLSI architecture for interpolation in soft-decision list decoding of Reed-Solomon codes," in *Proc. IEEE Workshop Signal Process. Syst.*, San Diego, CA, Oct. 2002, pp. 39–44.

[6] ——, "Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders," *J. VLSI Signal Process.*, vol. 39, pp. 93–111, Jan.–Feb. 2005.

[7] R. Koetter, J. Ma, A. Vardy, and A. Ahmed, "Efficient interpolation and factorization in algebraic soft-decision decoding of Reed-Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2003, p. 365.

[8] R. Koetter and A. Vardy, "A complexity reducing transformation in algebraic list decoding of Reed-Solomon codes," in *Proc. Inf. Theory Workshop*, Paris, France, Mar. 31–Apr. 4 2003, pp. 10–13.

[9] A. Ahmed, R. Koetter, and N. R. Shanbhag, "VLSI architectures foi soft-decision decoding of Reed-Solomon codes," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2004, vol. 5, pp. 2584–2590.

[10] A. Ahmed, N. R. Shanbhag, and R. Koetter, "Systolic interpolation architectures for soft-decoding Reed-Solomon codes," in *Proc. IEEE Workshop Signal Process. Syst.*, Aug. 2003, pp. 81–86.

[11] X. Zhang and K. K. Parhi, "Fast factorization architecture in soft-decision Reed-Solomon decoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 4, pp. 413–426, Apr. 2005.

[12] F. Parvaresh and A. Vardy, "Multiplicity assignments for algebraic soft-decoding of Reed-Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2003, p. 205.

[13] H. Xia and J. R. Cruz, "Application of soft-decision Reed-Solomon decoding to magnetic recording channels," *IEEE Trans. Magn.*, vol. 40, no. 5, pp. 3419–3430, Sep. 2004.

[14] M. K. Cheng, J. Campello, and P. H. Siegel, "Soft-decision Reed-Solomon decoding on partial response channels," in *Proc. IEEE Globecom*, 2002, vol. 2, pp. 1026–1030.

[15] M. K. Cheng and P. H. Siegel, "Iterative soft-decision Reed-Solomon decoding on partial response channels," in *Proc. IEEE Globecom*, Dec. 2003, vol. 3, pp. 1588–1592.

[16] P. F. Swaszek and W. Jones, "How often is hard-decision decoding enough?," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1187–1193, May 1998.

[17] H. Xia, H. Song, and J. R. Cruz, "Retry mode soft Reed-Solomon decoding," *IEEE Trans. Magn.*, vol. 38, no. 5, pp. 2325–2327, Sep. 2002.

[18] H. L. Trees, *Detection, Estimation, and Modulation Theory, Part I*. New York: Wiley, 1968.

[19] H. Takagi, *Queueing Analysis*. Amsterdam, The Netherlands: Elsevier, 1993, vol. 3.

[20] CS3210/12 Reed-Solomon Decoders, Amphion, data sheet.

[21] W. J. Gross, "Implementation of algebraic soft-decision Reed-Solomon decoders," Ph.D. dissertation, Univ. Toronto, Toronto, ON, Canada, 2003.
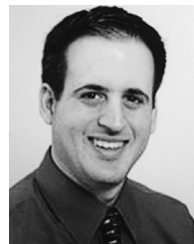
Warren J. Gross (S'92–M'04) received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1996, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1999 and 2003, respectively.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada. During the summers of 2004 and 2005, he was a Visiting Professor at the Université de Bretagne-Sud, Lorient, France. His research interests are in the design and applications of signal processing microsystems and custom computer architectures.

Dr. Gross served on the Program Committee of the 2006 IEEE Symposium on Field-Programmable Custom Computing Machines, and is a member of the Design and Implementation of Signal Processing Systems Technical Committee of the IEEE Signal Processing Society.

Frank R. Kschischang (S'83–M'91–SM'00–F'06) received the B.A.Sc. degree (with honors) from the University of British Columbia, Vancouver, BC, Canada, in 1985 and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1988 and 1991, respectively, all in electrical engineering.

He is a Professor of Electrical and Computer Engineering and Canada Research Chair in Communication Algorithms at the University of Toronto, where he has been a faculty member since 1991. During 1997–1998, he spent a sabbatical year as a Visiting Scientist at the Massachusetts Institute of Technology (MIT), Cambridge. His research interests are focused on the area of coding techniques, primarily on soft-decision decoding algorithms, trellis structure of codes, codes defined on graphs, and iterative decoders. He has taught graduate courses in coding theory, information theory, and data transmission.

Dr. Kschischang was the recipient of the Ontario Premier's Research Excellence Award. From October 1997 to October 2000, he served as an Associate Editor for Coding Theory for the IEEE TRANSACTIONS ON INFORMATION THEORY. He also served as Technical Program Co-chair for the 2004 IEEE International Symposium on Information Theory.

Ralf Koetter (S'92–M'95) received the Diploma degree in electrical engineering from the Technical University Darmstadt, Darmstadt, Germany, in 1990, and the Ph.D. degree from the Department of Electrical Engineering, Linköping University, Linköping, Sweden.

From 1996 to 1998, he was a Visiting Scientist at the IBM Almaden Research Laboratory, San Jose, CA. He was a Visiting Assistant Professor at the University of Illinois at Urbana-Champaign, and Visiting Scientist at CNRS, Sophia Antipolis, France. He joined the faculty of the University of Illinois at Urbana-Champaign in 1999, and is currently an Associate Professor at the Coordinated Science Laboratory at the University. His research interests include coding and information theory and their application to communication systems.

Dr. Koetter served as Associate Editor for Coding Theory and Techniques for the IEEE TRANSACTIONS ON COMMUNICATIONS from 1999 to 2001. In 2000, he started a term as Associate Editor for Coding Theory of the IEEE TRANSACTIONS ON INFORMATION THEORY. He received an IBM Invention Achievement Award in 1997, an NSF CAREER Award in 2000, and an IBM Partnership Award in 2001. He is a member of the Board of Governors of the IEEE Information Theory Society.

P. Glenn Gulak (S'82–M'83–SM'96) received the Ph.D. degree from the University of Manitoba, Winnipeg, MB, Canada.

Currently, he is a Professor in the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. He held the L. Lau Chair in Electrical and Computer Engineering for the period 1999–2004, and currently holds the Canada Research Chair in Signal Processing Systems. From 1985 to 1988, he was a Research Associate in the Information Systems Laboratory and the Computer Systems Laboratory at Stanford University, Stanford, CA. From March 2001 to March 2003, he was the Chief Technical Officer and Senior VP LSI Engineering of a fabless semiconductor startup headquartered in Irvine, CA, that focused on wireline and wireless communication ICs. His present research interests are in the areas of algorithms, circuits, and system-on-chip architectures for digital communications. He has authored or co-authored more than 100 publications in refereed journal and refereed conference proceedings.

Dr. Gulak is a registered Professional Engineer in the Province of Ontario. In addition, he has received numerous teaching awards for undergraduate courses taught in both the Department of Computer Science and the Department of Electrical and Computer Engineering at the University of Toronto. He served on the ISSCC Signal Processing Technical Subcommittee from 1990 to 1999, was ISSCC Technical Vice-Chair in 2000, and served as the Technical Program Chair for ISSCC 2001. He received the IEEE Millenium Medal in 2001.