

A VLSI ARCHITECTURE FOR INTERPOLATION IN SOFT-DECISION LIST DECODING OF REED-SOLOMON CODES

Warren J. Gross¹, Frank R. Kschischang¹, Ralf Koetter², and P. Glenn Gulak¹

¹ Department of Electrical and Computer Engineering, University of Toronto
Toronto, Ontario, M5S 3G4, Canada

² Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
Urbana, IL, 61801, U.S.A.

ABSTRACT

The Koetter-Vardy algorithm is an algebraic soft-decision decoder for Reed-Solomon codes which is based on the Guruswami-Sudan list decoder. There are three main steps: 1) multiplicity calculation, 2) interpolation and 3) root finding. The Koetter-Vardy algorithm is challenging to implement due to the high cost of interpolation. We propose a VLSI architecture for interpolation that uses a transformation of the received word to reduce the number of iterations of the interpolation algorithm. We also show how the memory requirements can be reduced and an important operation, the Hasse derivative, can be efficiently implemented in VLSI.

1. INTRODUCTION

Reed-Solomon codes are powerful error-correcting codes that can be found in a wide variety of digital communications systems, from digital media to wireless communications and deep-space probes. The ubiquitous nature of these codes continues to fuel research into decoding algorithms some forty years after their introduction.

Classical decoders for Reed-Solomon codes of length n and dimension k can correct $t = \lfloor d_{min}/2 \rfloor$ errors where $d_{min} = (n - k + 1)$ is the minimum distance of the code. Recently, a new class of *list decoding* algorithms for Reed-Solomon codes were introduced that can correct $t' > t$ errors [1] [2]. The list decoding problem is to find the set of codewords at a Hamming distance of t' from the received word. Since there might not be a unique codeword at a distance $t' > d_{min}/2$, the decoder returns a list of candidate codewords. The Guruswami-Sudan (GS) algorithm can correct up to $n - \sqrt{nk}$ errors [2].

To improve the error-correction capability of a decoder even further, the decoder can take advantage of the *soft* reliability information available from the channel. Soft-decision

decoders can provide an asymptotic coding gain of 2-3 dB on Gaussian channels [3] and 10 dB or more on Rayleigh fading channels.

The traditional Reed-Solomon decoding algorithms are algebraic; that is, they exploit the underlying algebraic structure of the code to generate a system of equations that is solved using finite field arithmetic. This operation does not appear to be compatible with the real-valued, *soft* information available from the channel. Koetter and Vardy have recently proposed an algebraic soft-decision decoding algorithm by extending the list decoder of Guruswami and Sudan. The Koetter-Vardy (KV) algorithm [4] [5] can achieve up to about 4 dB of coding gain at a frame-error-rate (FER) of 10^{-3} on a Gaussian noise channel, with a practical range of 1-1.5 dB and gains of 2-7 dB on a Rayleigh fading channel [6].

The Koetter-Vardy algorithm shows a lot of promise from the point of view of error correcting performance. Unfortunately, it seems to be quite computationally complex and not straightforward to implement in VLSI. In this paper we propose a VLSI architecture for the implementation of bivariate interpolation which is one of the major steps of the KV and GS algorithms.

2. THE LIST DECODING ALGORITHM

Consider the finite field with Q elements, $\text{GF}(Q)$. The message to be transmitted, f , consists of k elements of $\text{GF}(Q)$. The message symbols can be considered to be the coefficients of a degree $k-1$ univariate message polynomial $f(x)$. An (n, k) Reed Solomon code over $\text{GF}(Q)$ represents the k -symbol message f by an n -symbol codeword c , formed by evaluating the message polynomial $f(x)$ at n elements of $\text{GF}(Q)$. Usually $n = Q - 1$ and the set of evaluation elements is the set of nonzero elements of $\text{GF}(Q)$. This *evaluation map* encoding method is useful because it provides insight leading to interpolation-based decoding algorithms.

First we need to define some concepts related to bivariate polynomials. Consider the bivariate polynomial with co-

This work was supported by NSERC, the Government of Ontario and the National Science Foundation under grant CCR-0073490

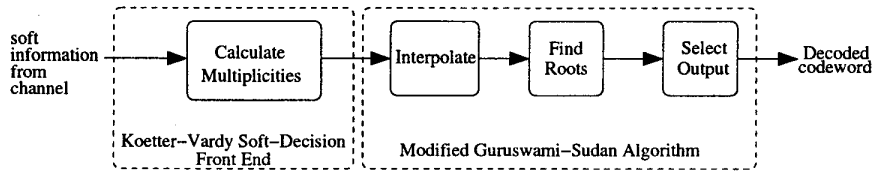


Fig. 1. The Koetter-Vardy algorithm.

efficients chosen from a finite field

$$P(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} p_{i,j} x^i y^j \in \text{GF}(Q)[x, y]. \quad (1)$$

Let w_x and w_y be nonnegative real numbers. The (w_x, w_y) -weighted degree of $P(x, y)$, $\deg^{(w_x, w_y)}(P)$, is defined as the maximum over all the numbers $i w_x + j w_y$ such that $p_{i,j} \neq 0$. The (α, β) 'th Hasse derivative of $P(x, y)$ is defined for integers $\alpha, \beta \geq 0$ as [6] [7]:

$$P^{[\alpha, \beta]}(x, y) = \sum_{a \geq \alpha, b \geq \beta} \binom{a}{\alpha} \binom{b}{\beta} p_{a,b} x^{a-\alpha} y^{b-\beta}. \quad (2)$$

$P(x, y)$ is said to pass through a point (x_i, y_i) with multiplicity m_i if $P^{[\alpha_i, \beta_i]}(x_i, y_i) = 0$, $\alpha_i + \beta_i < m_i$.

Consider the received word $y = c + e$ where e is an error vector. An element of $\text{GF}(Q)$, x_i , can be uniquely associated with each y_i to form the list of points in $\text{GF}(Q) \times \text{GF}(Q)$, $L_n = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. If there is no noise ($e = 0$) then $y_i = f(x_i)$ and a bivariate polynomial $P(x, y) = y - f(x)$ passes through all the points in L_n with a multiplicity of one. This suggests that an interpolation-based approach can be used to decode Reed-Solomon codes. In the presence of noise ($e \neq 0$), the interpolation polynomial will pass through some points that are not part of the codeword. The GS algorithm ensures that under certain conditions, the codeword polynomial “lives inside” the interpolation polynomial [1] [2]. The GS algorithm is an interpolation-based list decoder with two main steps:

1. Interpolation Step: Given the set of points L_n and a positive integer m , compute $P(x, y) \in \text{GF}(Q)[x, y] \setminus \{0\}$ of minimal $(1, k-1)$ -weighted degree that passes through all the points in L_n with multiplicity m .
2. Factorization Step: Given the interpolation polynomial $P(x, y)$, identify all the factors of $P(x, y)$ of the form $y - f(x)$ with $\deg f(x) < k$. The output of the algorithm is a list of the codewords that correspond to these factors.

A complete factorization of $P(x, y)$ is not necessary since we are just looking for linear y -roots of degree $< k$. An appropriate root-finding algorithm is given in [8].

Guruswami and Sudan hint at a possible soft-decision extension to their algorithm by allowing each point on the interpolated curve to have its own multiplicity. Koetter and Vardy proposed a method to translate soft-information into multiplicities proportional to the reliability of the points [4] [5]. We note that all possible $(Q \times n) = O(n^2)$ points are considered. In practice the actual number of points is $O(n)$. A low-complexity algorithm for implementing the KV front-end is proposed in [6]. The rest of the decoding is identical to the GS algorithm except that $P(x, y)$ is found through a biased interpolation with unequal multiplicities. Figure 1 is a block diagram of the KV algorithm.

Gaussian elimination could be used to solve the interpolation step with complexity $O(n^3)$ however if we exploit the special nature of this problem the complexity can be reduced. The interpolation step finds a Gröbner basis for the ideal of bivariate polynomials which vanish at a set of points with prescribed multiplicities. Fast algorithms for interpolation are described in [8] [9] [10] [11] [12] and [13]. We use the algorithm from [13] for the GS algorithm which is easily modified to handle unequal multiplicities (Algorithm 1). Let the multiplicity of the point (x_i, y_j) , $m_{i,j}$, be a nonnegative integer. Then the number of linear constraints on the interpolation problem is $C = \sum_{i=1}^Q \sum_{j=1}^n m_{i,j} (m_{i,j} + 1) / 2$. Algorithm 1 runs for C iterations. At the end of each iteration, all $b = \lfloor ((k-1) + \sqrt{(k-1)^2 + 8C(k-1)}) / (2(k-1)) \rfloor$ polynomials in the set G satisfy all of the linear constraints considered up until that point. If we want to read out an answer, we choose the polynomial with the smallest $(1, k-1)$ -weighted degree.

The root-finding algorithm from [8] outputs a list of up to $b-1$ candidate messages. The main computational task in the root-finding algorithm is finding the roots of a univariate polynomial.

3. A VLSI ARCHITECTURE FOR INTERPOLATION

In this section we describe an architecture for the implementation of interpolation for the KV and GS algorithms. The interpolation step is the most computationally demanding part of the algorithm and its complexity must be reduced if the algorithm is to be implemented in VLSI. There are three

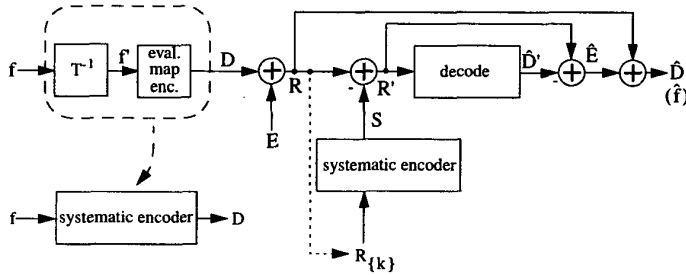


Fig. 2. The decoding problem can be transformed into an easier one by reencoding.

Algorithm 1 Interpolation algorithm.

```

 $G \leftarrow \{g_0 = 1, g_1 = y, g_2 = y^2, \dots, g_{b-1} = y^{b-1}\}$ 
for each point  $(x_i, y_j)$  with multiplicity  $m_{i,j} > 0$  do
  for  $\alpha \leftarrow 0$  to  $m_{i,j} - 1$  do
    for  $\beta \leftarrow 0$  to  $m_{i,j} - 1 - \alpha$  do
       $f \leftarrow \min_{\deg(1,k-1)} \{g \in G \text{ such that } g^{[\alpha,\beta]}(x_i, y_j) \neq 0\}$ 
      for  $g \in G$  such that  $g \neq f$  do
         $g \leftarrow g \cdot f^{[\alpha,\beta]}(x_i, y_j) - f \cdot g^{[\alpha,\beta]}(x_i, y_j)$ 
      end for
       $f \leftarrow (x - x_i)f$ 
    end for
  end for
end for

```

problems that must be addressed. First, is the large number of linear constraints that must be satisfied. The *cost* of interpolation, C , is the number of constraints and hence the number of iterations of Algorithm 1. We have determined experimentally that the cost of hard-decision decoding with multiplicity m at each point is an upper bound to the cost of soft-decision decoding with a maximum multiplicity of m (for high SNR, soft-decision decoding reduces to hard-decision decoding). The cost is $C \leq (n/2)(m)(m+1)$, which has a squared dependence on m and a linear dependence on n . This “multiplicity explosion”, combined with a large value of n (e.g. $n = 255$) results in very high costs. For example, for a $(255, k)$ Reed-Solomon code and $m = 4$, the cost is $C = 2550$ iterations. Each received symbol imposes 10 linear constraints on the interpolation problem.

The second problem is the large memory requirement. Algorithm 1 needs to store b bivariate polynomials. Since a homogeneous linear system must have more unknowns than equations, the length (number of terms) of the polynomials must be at least C . Therefore the memory requirements of interpolation are $\approx bC$.

The third problem is the implementation of the Hasse derivative (equation (2)) which is a computationally demanding operation.

3.1. Reducing the Number of Iterations

We can use the trick of reencoding the received word to reduce the interpolation complexity in the spirit of the Berlekamp-Welch (BW) algorithm [14] [15]. We will first apply this technique to the hard-decision GS algorithm. Consider the communications system shown in Figure 2. A message f is systematically encoded to a codeword D . Systematic encoders are easily implemented with a linear feedback shift register. After transmission through a noisy channel, the received hard-decision word is $R = D + E$ where E is an error vector. Take the last k symbols of R , $R_{\{k\}}$, and systematically reencode them to create a new codeword from the same code, S . Then $R' = R - S = (D + E) - S = (D - S) + E$ is a codeword (by the linearity of the code) that is corrupted by the same error pattern as R . R' has a very interesting property; the last k symbols of R' are zero, meaning that only $n - k$ symbols of the received word are actually dependent on the transmitted codeword. The reencoded word corresponds to a set of k points whose y -components are zero: $z = \{(x_i, 0), (x_{i+1}, 0), \dots, (x_{i+k-1}, 0)\}$. Define $v(x)$ as the univariate polynomial that passes through the points in z with a multiplicity of one, $v(x) = \prod_{j=i}^{i+k-1} (x - x_j)$. A univariate polynomial that passes through these points with fixed multiplicity m can be written as $v(x)^m$. The advantage is that $v(x)$ and its powers can be precomputed offline once and stored. Algorithm 1 is run at decode time starting from the initial polynomial set

$$G = \{v(x)^m, v(x)^{m-1}y, \dots, v(x)^{m-(b-1)}y^{b-1}\}, \quad (3)$$

and run for $C' = ((n - k)/2)(m)(m + 1) = (1 - \frac{k}{n})C$ iterations. The reduced cost, C' , gets smaller as the code rate k/n increases.

The estimated error pattern \hat{E} can be extracted from the decoded codeword \hat{D}' by subtracting R' and then added to R to obtain the estimate of the transmitted codeword, \hat{D} . Note that in the usual choice of field, $\text{GF}(2^q)$, subtraction and addition are equivalent. Here, another advantage of using a systematic encoder in the transmitter becomes evident. The generator matrices for an arbitrary encoding \mathbb{G}

and the evaluation map encoding \mathbb{G}_e of the same code span the same linear space. We have $\mathbb{G}_e = T\mathbb{G}$, where T is a $k \times k$ transformation matrix. Since the GS decoder assumes an evaluation map encoder, the message that is recovered is \hat{f}' and $\hat{f} = \hat{f}'T$. The systematic encoding eliminates this calculation since \hat{f} can be read off directly from \hat{D} .

The reencoding technique can also be applied to soft-decision KV decoding. The polynomial $v(x)^m$ can be calculated for k points that have the maximum multiplicity m . This is a reasonable assumption, as we will see below. We do not know in advance the x -values of these k most reliable points and the powers of $v(x)$ have to be calculated at decode time. However, there are still significant savings from reencoding as k points are handled by a univariate interpolation, eliminating the ‘‘multiplicity explosion’’ associated with the bivariate interpolation. A bivariate interpolation is only needed for $O(n - k)$ points.

3.2. Reducing the Memory Requirements

For a fixed multiplicity, m , the interpolation polynomial can be expressed as

$$P(x, y) = \sum_{j=0}^{b-1} w_j(x)v(x)^{m-j}y^j, \quad (4)$$

which means that common factors of $v(x)$ are being carried around needlessly, wasting memory. It would be nice to factor out the powers of $v(x)$ and only have to calculate the $w_j(x)$ in real-time. Consider the original set of polynomials $G_1 = \{1, y, \dots, y^{b-1}\}$. After applying the reencoding technique, the starting polynomial set for decoding is $G' = \{v(x)^m, v(x)^{m-1}y, \dots, v(x)^{m-(b-1)}y^{b-1}\}$. To eliminate powers of $v(x)$, perform the change of variables $\tilde{y} = y/v(x)$. Note that $\deg^{(1,k-1)}(\tilde{y}) = \deg^{(1,k-1)}(y) - \deg^{(1,k-1)}(v(x)) = (k-1) - k = -1$. To perform the interpolation, rescale the y -coordinates of the first $n - k$ points: $\tilde{R}_i = \frac{R_i}{v(x_i)}$, $i = 1, \dots, n - k$. Then, starting from $\tilde{G} = \{1, \tilde{y}, \dots, \tilde{y}^{b-1}\}$, apply Algorithm 1 to the $n - k$ rescaled points where the min function is with respect to the $(1, -1)$ -weighted degree of the polynomials in x and \tilde{y} . This reduces the length of the stored polynomials. At the end of interpolation, $P(x, y)$ is reconstructed from $\tilde{P}(x, \tilde{y})$ according to (4). This memory reduction technique also decreases the interpolation time since fewer terms need to be updated at each iteration. This technique can be used to reduce the interpolation complexity of the GS algorithm for $m = 1$ to the same order of complexity as the BW algorithm, which is intuitive since they solve the same problem.

From equation (4) we see that this technique only works if $b \leq m + 1$. This restricts the rate of the code to be high enough that the GS algorithm does not correct any more errors than a classical decoder. However, this technique is

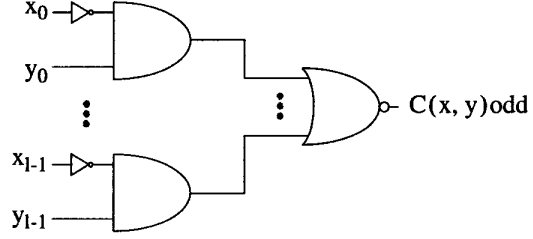


Fig. 3. Circuit for implementing the parity of the binomial coefficient $C(x, y)$.

applicable to the KV algorithm which achieves significant coding gain for all rates [6]. To apply this technique we have to ensure that there are at least k points with the maximum multiplicity. This is a good assumption for high SNR. For the cases where this is not true we have found that the performance loss from forcing the k most reliable symbols to have the same multiplicity is small. To do the reencoding, we need to systematically encode k symbols drawn from arbitrary positions in the received word. This encoder can be implemented as an erasures-only classical Reed-Solomon decoder, which is much simpler to implement than the standard Reed-Solomon decoder because the error-locator polynomial is already known.

3.3. VLSI Architecture for the Hasse Derivative

Calculating the Hasse derivative (HD) of a bivariate polynomial is a computationally demanding task that has the flavor of polynomial evaluation. We make the observation that in fields of characteristic two, the product

$$H = \binom{a}{\alpha} \binom{b}{\beta} p_{a,b} x_i^{a-\alpha} y_j^{b-\beta} \quad (5)$$

is nonzero only if $a \geq \alpha$, $b \geq \beta$ and both $C(a, \alpha) = \binom{a}{\alpha}$ and $C(b, \beta) = \binom{b}{\beta}$ are odd. The parity of the binomial coefficients can be calculated easily by Lucas’ theorem [16]. If we write the integers x and y in their binary representations, then $C(x, y)$ is odd and $x \geq y$ if x AND y is equal to y . A simple circuit for calculating the parity of the binomial coefficients is shown in Figure 3.

The coefficients of the polynomials $G = \{g_i(x, y)\}$ in the interpolation algorithm are stored in an order sorted by weighted degree to facilitate finding the minimum $g_i(x, y)$. Horner’s rule [17] for the fast evaluation of polynomials can be used to calculate the HD if the terms of the polynomial are read out in order of decreasing weight. If the polynomial is sparse, exponentiation can be implemented efficiently using the right-to-left binary exponentiation algorithm (RLBE) [17]. The exponent N can be written in

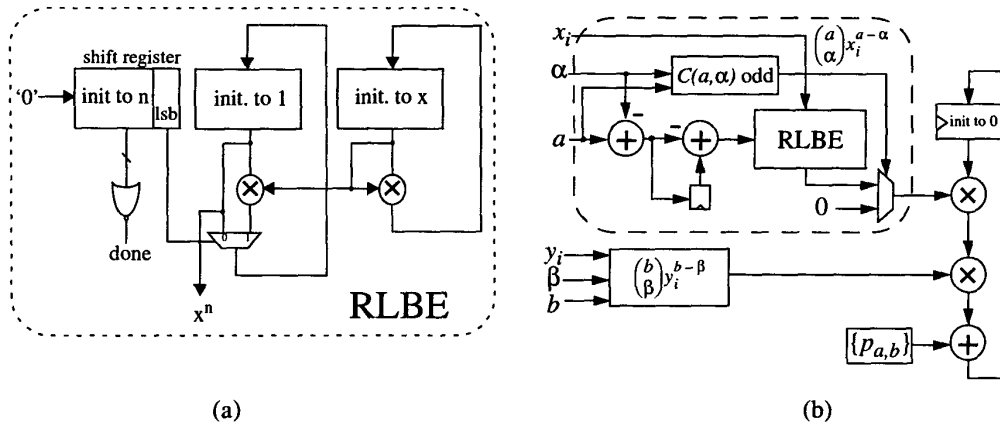


Fig. 4. (a) Circuit for right-to-left binary exponentiation. (b) Circuit for calculating the Hasse derivative using Horner's rule.

its binary representation $N = (N_{L-1}, N_{L-2}, \dots, N_1, N_0)$ where $N_{L-1} = 1$. In Figure 4(a) we present an implementation of the RLBE algorithm that executes in N_L clock cycles. Using Horner's rule, the input to the RLBE circuit is the exponent *difference*, which reduces the number of clock cycles. A circuit for implementing the HD is shown in Figure 4(b). If $C(a, \alpha)$ or $C(b, \beta)$ for a partial product is zero and the exponentiation can be skipped.

A pipelined architecture for the interpolation algorithm is shown in Figure 5. There are two possible updates for a bivariate polynomial g :

1. $g \leftarrow \lambda g + \delta f$, where $\lambda, \delta \in GF(2^q)$ and f is a bivariate polynomial, or
2. $g \leftarrow x_i g + xg$, where $x_i \in GF(2^q)$.

This architecture calculates the polynomial update for iteration i concurrently with the HD calculation for iteration $i + 1$.

4. CONCLUSIONS

The Koetter-Vardy (KV) soft-decision decoding algorithm is an extension of the Guruswami-Sudan (GS) list-decoding algorithm for Reed-Solomon codes. The KV algorithm seems fairly complex for VLSI implementation because of the large number of iterations in the interpolation step. We have showed that the interpolation problem can be transformed into one where, for a high rate code, a large number of points (k) can be handled by univariate interpolation while only a small number ($O(n - k)$) of points have to be sent to the expensive bivariate interpolation. A memory reduction technique was shown to reduce the complexity of GS interpolation to that of BW interpolation. We

proposed an application of this technique to the KV algorithm. A VLSI architecture was presented for interpolation that shows how to efficiently implement the Hasse derivative operation.

5. REFERENCES

- [1] M. Sudan, "Decoding of Reed-Solomon codes beyond the error correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180–193, 1997.
- [2] Venkatesan Guruswami and Madhu Sudan, "Improved decoding of Reed-Solomon and Algebraic-Geometry codes," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1757–1767, September 1999.
- [3] A. Brinton Cooper III, "Soft decision decoding of Reed-Solomon codes," in *Reed-Solomon Codes and Their Applications*, Stephen B. Wicker and Vijay K. Bhargava, Eds., chapter 6, pp. 108–124. IEEE Press, New York, New York, 1994.
- [4] R. Kotter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," in *Proceedings of the 2000 IEEE International Symposium on Information Theory*, 2000, p. 61.
- [5] Ralf Koetter and Alexander Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," Submitted to *IEEE Transactions on Information Theory*, 2001.
- [6] Warren J. Gross, Frank R. Kschischang, Ralf Koetter, and P. Glenn Gulak, "Simulation results for algebraic soft-decision decoding of Reed-Solomon codes," in *Proceedings of the 21'st Biennial*

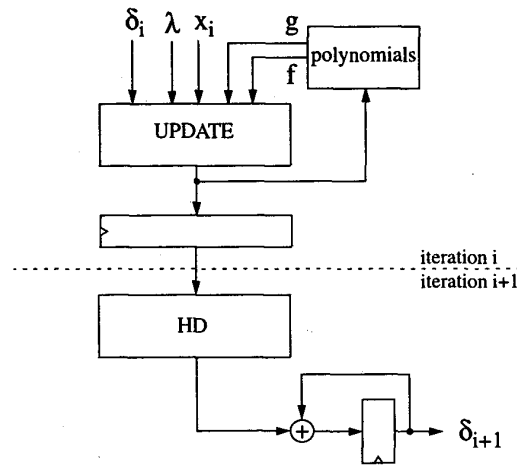


Fig. 5. Pipelined architecture for concurrently updating the polynomials of iteration i and calculating the Hasse derivative for iteration $i + 1$.

- Symposium on Communications*, Kingston, Ontario, Canada, June 2-5 2002, Queen's University, pp. 356–360, URL: http://www.eecg.toronto.edu/~wjgross/wjg/gkkg_queens02.pdf.
- [7] H. Hasse, "Theorie der höheren Differentiale in einem algebraischen Funktionenkörper mit vollkommenem Konstantenkörper bei beliebiger Charakteristik," in *J. Reine. Ang. Math.*, 1936, vol. 175, pp. 50–54.
- [8] Ron M. Roth and Gitit Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Transactions on Information Theory*, vol. 46, no. 1, pp. 246–257, January 2000.
- [9] H. M. Möller and B. Buchberger, "The construction of multivariate polynomials with preassigned zeros," in *EUROCAM '82, European Computer Algebra Conference*, Jaques Calmet, Ed., Marseille, France, April 1982, vol. 144 of *Lecture Notes In Computer Science*, pp. 24–31.
- [10] J. Abbott, A. Bigatti, M. Kreuzer, and L. Robbiano, "Computing ideals of points," *Journal of Symbolic Computation*, vol. 30, no. 4, pp. 341–356, 2000.
- [11] G. L. Feng and K. K. Tzeng, "A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes," *IEEE Transactions on Information Theory*, vol. 37, no. 5, pp. 1274–1287, September 1991.
- [12] Ralf Kötter, *On Algebraic Decoding of Algebraic-Geometric and Cyclic Codes*, Ph.D. thesis, Linköping University, 1996.
- [13] Rasmus Refslund Nielsen, "Decoding AG-codes beyond half the minimum distance," M.S. thesis, Technical University of Denmark, August 31 1998.
- [14] Llyod R. Welch and Elwyn R. Berlekamp, "Error correction for algebraic block codes," US Patent 4,633,470, 1986.
- [15] Elwyn Berlekamp, "Bounded distance +1 soft-decision Reed-Solomon decoding," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 704–720, May 1996.
- [16] Valdemar C. da Rocha Jr., "Digital sequences and the Hasse derivative," in *Communications Coding and Signal Processing*, Barham Honary, Michael Darnell, and Paddy Farrell, Eds., vol. 3, *Communication Theory and Applications*, pp. 256–268. John Wiley and Sons Inc., 1997.
- [17] Donald E. Knuth, *The Art of Computer Programming*, vol. 2, *Seminumerical Algorithms*, Addison-Wesley, 1969.