Efficient Interpolation Architecture for Soft-Decision List Decoding of Reed-Solomon Codes

Sungman Lee
The MTH
426-5, Gasan-dong, Kumcheon-gu
Seoul, Korea
e-mail: orozi318@naver.com

Taegeun Park
The Catholic University of Korea
San43-1, Yokok2-dong, Bucheon-shi,
Kyungki-do, Korea
e-mail: parktg@catholic.ac.kr

***Abstract*— Recently, algebraic soft-decision decoding algorithm for RS codes that can correct the errors beyond the error correcting bound has been proposed. The main task in the algorithm is the weighted interpolation of a bivariate polynomial that requires intensive computations. In this paper, we propose an efficient architecture with low hardware complexity for interpolation in soft-decision list decoding of Reed-Solomon codes. The proposed architecture processes the candidate polynomial in such a way that the terms of *X* degrees are processed in serial and the terms of *Y* degrees are processed in parallel. The processing order of candidate polynomials adaptively changes to increase the efficiency of memory access for coefficients. The proposed interpolation architecture for the (255, 239) RS list decoder is designed and synthesized using the DonbuAnam 0.18um standard cell library. The maximum operating clock frequency is 200MHz and the synthesized gate count is about 25.1K gates in two-input equivalent NAND gates.**

Keywords—VLSI architecture; Polynomial interpolation; Reed-Solomon codes; Soft-decision list decoding.

I. INTRODUCTION

Among the various kinds of error correcting codes in digital communication systems, Reed-Solomon (RS) codes are widely used block codes due to their excellent burst error-correcting capabilities. It is well known that an (n,k) RS codes have k message symbols and n coded symbols, where each symbol belongs to $GF(2^m)$. An (n,k) RS codes can correct ν symbols and ρ erasures with $2\nu+\rho\leq n-k$. Classical RS decoding scheme can be thought of as the bounded minimum distance (BMD) algorithm that decodes the received codewords through the channel by hard-decision. Efficient algebraic hard-decision decoding algorithms, such as Berlekamp-Massey algorithm and Euclid algorithm [1] have been widely used to decode the Reed-Solomon codes.

Recently, Guruswami-Sudan (GS) [2] proposed a polynomial-time list decoding algorithm that can correct the errors beyond the error correcting bound. The proposed list decoding algorithm has a decoding radius $t' > \lfloor d_{min}/2 \rfloor$ and corrects up to $n-\sqrt{nk}$ errors for all code rates [2]. With reliable soft-decision data, such as probabilistic channel information, RS decoding can achieve better performance in correcting errors. Koetter and Vardy (KV) [3] generalized the list decoding algorithm that can decode, as long as a certain weighted condition is satisfied. The soft-decision list decoding algorithm consists of two major processes: interpolation process with KV front end and factorization process. The interpolation process is quite computationally intensive with large latency, so it may suffer low performance. The re-encoding scheme can be applied to reduce the number of iterations for interpolation [4].

Many researchers have proposed a number of the interpolation architectures for the soft-decision RS decoder [5][6][7][13][14]. Most of the architectures proposed so far try to increase the decoding performance by parallelizing the processes for the candidate polynomials. This requires considerable hardware with memory modules that may be hard to apply in some applications. Ahmed, Koetter, and Shanbhag proposed the point-serial algorithm that calculates all the discrepancy coefficients corresponding to a particular interpolation point in parallel [5]. Wang and Ma represent the finite field numbers in both regular and power formats, i.e., hybrid-format, that can reduce the hardware complexity for the DCC block and parallelize the decoder architecture [6]. The parallel architecture [7] proposed by Gross, Kschischang, and Gulak embeds both a binary tree and a linear array in a 2–D array processor, enabling fast polynomial evaluation operations. Zhu *et. al.* have proposed backward interpolation, which eliminates interpolation points or reduces interpolation multiplicities [13]. The proposed architectures share computational units with forward interpolation architectures to reduce the hardware complexity. In [14], new techniques are employed to achieve high-speed interpolation for the iterative bit-level generalized minimum distance (BGMD) algebraic soft-decision decoding. They also proposed architectures to efficiently integrate the combined and backward interpolation techniques.

In this paper, we propose an efficient architecture with low hardware complexity for interpolation in a soft-decision list decoding of Reed-Solomon codes. To reduce hardware cost, the proposed architecture processes the terms of X degrees in the candidate polynomial serially, whereas it processes the terms of Y degrees in the candidate

This work was partially supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2008-313-D00743)

This work was partially supported by the Research Fund, 2011 of The Catholic University of Korea.

polynomial in parallel. During the polynomial update in the interpolation process, the appropriate polynomial coefficients should be read efficiently from memory. The processing order of candidate polynomials adaptively changes to increase the efficiency of memory access for coefficients. This scheduling minimizes the usage of internal registers and the number of memory accesses and simplifies the memory structure by combining and storing data in memory. Also, the proposed architecture shows high hardware efficiency, since each module is balanced in terms of latency and the modules are maximally overlapped in the schedule.

II. Soft-decision List Decoding of RS Codes

An (n,k) RS codes defined in the Galois field $GF(2^m)$ have codewords of length $n = 2^m - 1$, where m is a positive integer and k is the number of information symbols in the codeword. RS codes can be obtained by evaluating certain subspaces of $\mathbb{F}_q(X)$ in a set of points $\boldsymbol{\mathcal{P}} = \{x_0, x_1, \cdots, x_{n-1} \subseteq \mathbb{F}_q(X)\}$. Therefore, (n,k) RS codes $\mathbb{C}_q(n,k)$ of length n and dimension k is defined as

$$\mathbb{C}_q(n,k) = \{(f(x_0), f(x_1), \cdots, f(x_{n-1})) : x_0, x_1, \cdots, x_{n-1} \subseteq \boldsymbol{\mathcal{P}}, f(x) \in \mathbb{F}_{\boldsymbol{q}}(\boldsymbol{X}), \deg f(X) < k\} \quad (1)$$

Guruswami and Sudan verified that the generalized Reed-Solomon decoding problem reduces to the polynomial reconstruction problem [2]. Now, we define the essential elements of the soft decision list decoding algorithm. The bivariate polynomial $Q(X,Y)$ over $\mathbb{F}_q$ is defined as in [3].

$$Q(X,Y) = \sum_t^r \sum_s^{w_v} q_{s,t} X^s Y^t = \theta_0(X) + \theta_1(X)Y + \theta_2(X)Y^2 + \cdots + \theta_v(X)Y^v + \cdots + \theta_r(X)Y^r,$$

where $\theta_v(X) = q_{0,v} + q_{1,v}X + q_{2,v}X^2 + \cdots + q_{w_v,v}X^{w_v}$, $0 \le v \le r$. (2)

We define the weighted degree of a polynomial, as follows.

Definition 1: Let $Q(X,Y) = \sum_i^\infty \sum_j^\infty q_{i,j} X^i Y^j$ be a bivariate polynomial over $GF(2^q)$, and let w_x, w_y be real numbers. Then, the (w_x, w_y)-weighted degree of $Q(X,Y)$, denoted $\deg_{w_x,w_y} Q(X,Y)$, is the maximum over all numbers $iw_x + jw_y$, such that $q_{i,j} \neq 0$.

Definition 2: A bivariate polynomial $Q(X,Y)$ is said to pass through a point (x_i, y_i) with multiplicity $m_{x_i y_i}$, if the shifted polynomial $Q(X - x_i, Y - y_i)$ contains a monomial of degree $m_{x_i y_i}$ and does not contain a monomial of degree less than $m_{x_i y_i}$. Equivalently, the point (x_i, y_i) is said to be a zero of multiplicity $m_{x_i y_i}$ of the polynomial $Q(X,Y)$.

When $P(X,Y)$ is the shifted version of the polynomial $Q(X,Y)$ by (x_i, y_i), the equation below holds.

$$P(X,Y) = \sum_\beta^r \sum_\alpha^{w_v} p_{\alpha,\beta} X^\alpha Y^\beta = Q(X - x_i, Y - y_i) = \sum_\beta^r \sum_\alpha^{w_v} q_{\alpha,\beta} (X - x_i)^\alpha (Y - y_i)^\beta.$$

$$coef(Q(X - x_i, Y - y_i), X^\alpha Y^\beta) = p_{\alpha,\beta}, \quad p_{\alpha,\beta} = 0, \forall\, \alpha + \beta < m. \quad (3)$$

The soft-decision RS list decoding can be considered as a "curve-fitting" problem. In the first phase, the algorithm finds a polynomial $Q(x,y)$ of low degree that fits the points (x_i, y_i). Next, it finds all small degree roots of $Q(x,y)$; and each factor of $Q(x,y)$ forms possible candidates of the message polynomial.

We now briefly describe the list decoding algorithm. Figure 1 shows the block diagram of the soft-decision RS list decoder. The block diagram consists of three steps: multiplicity computation, interpolation, and factorization. The multiplicity computation step calculates the multiplicity matrix that has reliability information from the channel.

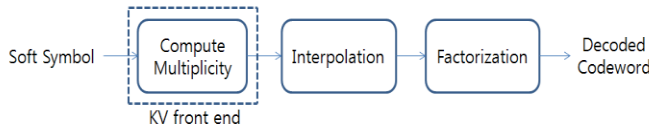
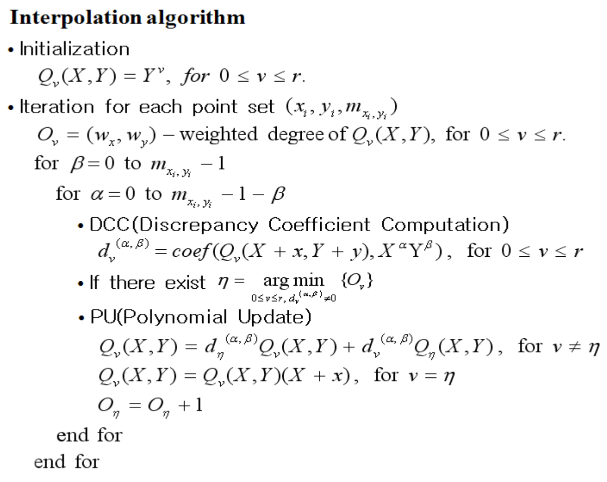


Figure 1. Block diagram of soft-decision RS list decoder.

Let us suppose that we have the set of interpolation points $(x_i, y_{i,j}), 1 \le i \le 2^q, 1 \le j \le n$ with corresponding multiplicities $m_{i,j}$. The interpolation step forms the nontrivial polynomial $Q(x,y)$ of minimal $(1, k-1)$-weighted degree that passes each interpolation point $(x_i, y_{i,j})$ with multiplicity at least $m_{i,j}$. This bivariate polynomial $Q(x,y)$ may contain the message polynomial as a root. After the bivariate polynomial $Q(x,y)$ is found, the factorization step determines all the factors of $Q(x,y)$ in the form of $Y - f(X)$, where the degree of $f(X)$ is at most k. Each root polynomial $f(X)$ is the candidate of the message polynomial. Finally, the polynomial with the highest probability among the candidates is selected as a message polynomial.



Interpolation algorithm

```
• Initialization
    $Q_v(X,Y) = Y^v$, $for\ 0 \le v \le r$.
• Iteration for each point set $(x_i, y_i, m_{x_i,y_i})$
    $O_v = (w_x, w_y)$ – weighted degree of $Q_v(X,Y)$, for $0 \le v \le r$.
    for $\beta = 0$ to $m_{x_i,y_i} - 1$
      for $\alpha = 0$ to $m_{x_i,y_i} - 1 - \beta$
        • DCC(Discrepancy Coefficient Computation)
          $d_v^{(\alpha,\beta)} = coef(Q_v(X + x, Y + y), X^\alpha Y^\beta)$, for $0 \le v \le r$
        • If there exist $\eta = \arg\min_{0 \le v \le r,\, d_v^{(\alpha,\beta)} \neq 0} \{O_v\}$
        • PU(Polynomial Update)
          $Q_v(X,Y) = d_\eta^{(\alpha,\beta)} Q_v(X,Y) + d_v^{(\alpha,\beta)} Q_\eta(X,Y)$, for $v \neq \eta$
          $Q_v(X,Y) = Q_v(X,Y)(X + x)$, for $v = \eta$
          $O_\eta = O_\eta + 1$
      end for
    end for
```

Figure 2. Interpolation algorithm.

III. Proposed Architecture

Now, we will explain the interpolation algorithm in more detail. The interpolation step finds the bivariate polynomial that fits the set of points with the corresponding multiplicities. Two main interpolation algorithms have been proposed so far: a constrained-serial interpolation algorithm [6][7][8] and a point-serial interpolation algorithm [5]. The point-serial interpolation algorithm is usually less efficient and less flexible in architecture than the constraint-serial interpolation algorithm [6]. Figure 2 shows the interpolation algorithm based on the Fundamental Iterative Algorithm (FIA) [10].

The algorithm consists of two major operations, namely the Discrepancy Coefficient Computation (DCC) and the Polynomial Update (PU). As we explained earlier, the interpolation process finds a bivariate polynomial $Q(x,y)$ that passes a point (x_i, y_j) with a multiplicity $m_{i,j}$. In the algorithm, the DCC operation computes the discrepancy coefficients corresponding to a particular constraint for each candidate polynomial and the PU operation updates the polynomial by reducing the corresponding discrepancy coefficients to zero.

A. Proposed interpolation architecture and its scheduling

Figure 3 shows the block diagram of the proposed interpolation architecture. The proposed architecture consists of the Discrepancy Coefficient Computation Unit (DCCU) that calculates the discrepancy coefficients (DC) using the Hasse Derivative (HD), the Polynomial Update Unit (PUU) that updates the candidate polynomials, the Polynomial Order Sorting Unit (POSU) that decides the processing order of data by storing and aligning the weighted degrees of the candidate polynomials, and a few memory blocks and control logic. The DCCU also consists of the HD computation block and the Y-generator that calculates the power of Y for the HD computation. The DCCU takes the stream of interpolation points and multiplicities as inputs.

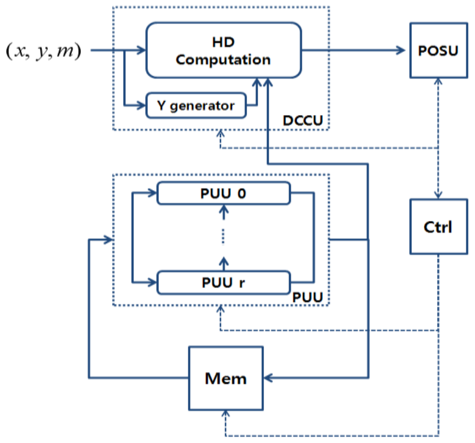


Figure 3. Proposed interpolation architecture.

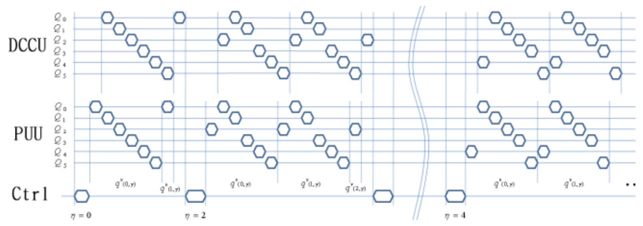


Figure 4. Timing diagram showing overlap between the DCCU and the PUU when $r = 5$.

The proposed architecture parallelizes the computation for the terms in Y and computes each candidate polynomial and the monomials in X sequentially and thus the required hardware is reduced. When the polynomials are updated sequentially, the "pivot" polynomial $Q_\eta(X,Y)$, which has the smallest weighted degree and a non-zero DC value, is used to update the other polynomials and is updated itself last.

Figure 4 shows the processing schedule of DCCU and PUU when $r = 5$, where Q_v denotes a candidate polynomial and $q^v_{(i,j)}$ denotes the coefficient of X^iY^j in Q_v. All the coefficients of X, with the same degree in Q_v, $(q^v_{(x,0)}, q^v_{(x,1)}, q^v_{(x,2)}, \ldots, q^v_{(x,r)})$, are processed simultaneously, since the proposed architecture processes the terms in Y in parallel. We can overlap the computation of the DCCU and the PUU by sending the output coefficient of the PUU directly to the DCCU, as depicted in Fig.4. The updated coefficients in candidate polynomials are stored and sent to the DCCU to calculate the next DC simultaneously. Fig.4 shows the timing diagram when $\eta = 0, 2, 4$. We assume that the candidate polynomials initially have the constant terms only at the first iteration ($\eta = 0$). The value $q^v_{(1,r)}$ at the first iteration $\eta = 0$ denotes the updated coefficient in $Q_\eta(X,Y)$.

B. Discrepancy Coefficient Computation Unit (DCCU)

The DCC defined in equation (4) calculates the coefficient of the monomial $X^\alpha Y^\beta$ in $Q(X+x, Y+y)$ that is the shifted version of $Q(X,Y)$ by x and y in X, Y direction respectively, where α, β are non-negative integers that satisfy $\alpha + \beta < m$. The following equation shows the Hasse derivative [11] to find the DC.

$$d_v^{(\alpha,\beta)} = coef\left(Q_v(X+x, Y+y), X^\alpha Y^\beta\right) = \sum_{t=\beta}^{r} \sum_{s=\alpha}^{w_{v,t}} \binom{s}{\alpha}\binom{t}{\beta} q^v_{(s,t)} x^{s-\alpha} y^{t-\beta}, for\ v = 0,1,2,\ldots,r \quad (7)$$

The hardware to solve equation (7) may suffer from latency, because it consists of a double loop of addition. The architecture proposed by Wang and Ma utilized the finite field additions, instead of multiplications, by representing the symbol by its exponent [6]. The proposed architecture calculates the DCs with respect to Y in parallel, whereas it calculates the DCs with respect to X and the candidate polynomials in serial. Equation (7) can be expanded as follows.

$$d_v^{(\alpha,\beta)} = \sum_{s=\alpha}^{w_{v,t}} \binom{s}{\alpha} x^{s-\alpha} \left\{ \binom{0}{\beta} q^v_{s,0} y^{0-\beta} + \binom{1}{\beta} q^v_{s,1} y^{1-\beta} + \cdots + \binom{r}{\beta} q^v_{s,r} y^{r-\beta} \right\} \quad (8)$$

The values $s-\alpha, t-\beta, (0 \le t \le r)$ are meaningful when $s > \alpha, t > \beta$. Fig.5 shows the block diagram of the proposed DCCU to compute equation (8). The multiplication at the input computes $x^{s-\alpha}$ and the DCC is performed monomially and in increasing order of X. Once the multiple of x, $x^{s-\alpha}$ is calculated, it is stored for the next computation and distributed to compute the other DCs in candidate polynomial simultaneously. In Fig.5, $y^{0-\beta} y^{1-\beta}, \ldots, y^{r-\beta}$ denotes the output of the Y generator, to be explained later, and $q^v_{s,t}$, which is the output of the PUU, is the coefficient of monomial X^sY^t in the v-th candidate polynomial $Q_v(X,Y)$. The value $\binom{s}{\alpha}\binom{t}{\beta}$ can be easily implemented by Lucas's theorem [11]. $\binom{s}{\alpha}$ is the common term like $x^{s-\alpha}$ that will be distributed and $\binom{t}{\beta}$ can be further simplified according to t. The registers on the right in Figure 5 store the intermediate DC values for each candidate polynomial.

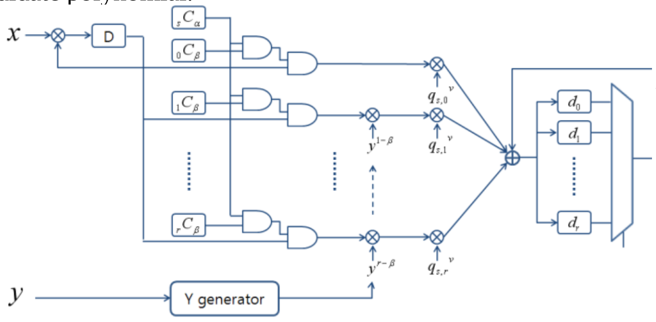


Figure 5. DCCU structure.

Y generator that is basically the same as that in [9] computes the multiple of y. When $\alpha=\beta=0$, $y^{0-0}=y^0=1, y^{1-0}=y, \ldots, y^{r-0}=y^r$. At the first iteration, the value when $\beta=0$ will be used immediately. As β increases, the values stored in the registers are shifted down and we can compute $y^{t-\beta}$ for the DCCU without any additional hardware. $r-1$ finite multipliers are required for the Y generator and the number of latency to get the first output is $\lceil log_2\, r\rceil$.

C. Polynomial Update Unit (PUU)

The PU stage updates each candidate polynomial $Q_v(X,Y)$ using the selected "pivot" polynomial $Q_\eta(X,Y)$, where η is the index of the minimum weighted degree polynomial among all polynomials with non-zero DCs. As explained in Fig.2, the "non-pivot" polynomials are usually updated first and the "pivot" polynomial is updated last.

$$Q_v(X,Y)=\begin{cases} d_\eta^{(\alpha,\beta)}Q_v(X,Y)+d_v^{(\alpha,\beta)}Q_\eta(X,Y), & for\ \ v\neq\eta \\ Q_\eta(X,Y)(X+x), & for\ \ v=\eta \end{cases}$$

Here, $d_\eta^{(\alpha,\beta)}$ and $d_v^{(\alpha,\beta)}$ denotes the DCs of $Q_\eta(X,Y)$ and $Q_v(X,Y)$ respectively. The candidate polynomials for $v\neq\eta$, are updated by adding two polynomials: $Q_v(X,Y)$ multiplied by $d_\eta^{(\alpha,\beta)}$ and $Q_\eta(X,Y)$ multiplied by $d_v^{(\alpha,\beta)}$. This deletes the monomial $X^\alpha Y^\beta$ in $P(X,Y)$ in equation (3) that is the shifted version of the polynomial $Q(X,Y)$ by (x_i,y_i). Last, the polynomial $Q_\eta(X,Y)$ will be updated by multiplying $(X+x)$.

The proposed architecture processes the polynomials in serial but with an efficient schedule. The update for the "non-pivot" polynomials in case of $v\neq\eta$ requires the information of both $Q_v(X,Y)$ and $Q_\eta(X,Y)$ before update. The monomials with the same X degree in the polynomial are computed simultaneously and the update is preceded from the constant terms to higher order of monomials, since the proposed architecture processes the Y degrees in parallel.

We can serialize and rewrite the update procedure, as in equation (11).

$$q'_{i,y}=q_{i-1,y}+xq_{i,y}, 0\le i\le d_x,\ \ 0\le y\le r,\ \ q_{-1,y}=0 \tag{11}$$

Figure 6 depicts the structure of one PUU element that can update both $Q_v(X,Y)$ and $Q_\eta(X,Y)$. When the corresponding coefficient $q^\eta_{i,y}$ in the polynomial $Q_\eta(X,Y)$ comes in as an input, the multiplexer select signal '*sel*' is set to '0' and the computation of $q^{\eta\ \prime}_{i,y}=xq^\eta_{i,y}+q^\eta_{i-1,y}$ is implemented. In the register, the coefficient $q^\eta_{i-1,y}$ will be stored to update the other polynomials $Q_v(X,Y)$. Then, this structure is used to update the polynomials $Q_v(X,Y)$ by setting the multiplexer select signal '*sel*' to '1', so the computation $q^{v\ \prime}_{i,y}=d_vq^\eta_{i,y}+d_\eta q^v_{i,y}$ will be implemented.

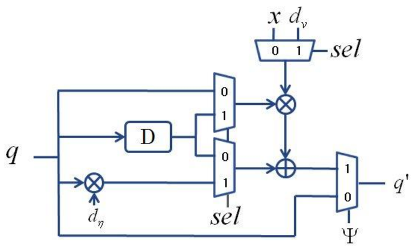


Figure 6. Structure of one PUU element.

Figure 7 shows the PUU structure that updates the polynomials in Y in parallel. PUU consists of $(r+1)$ PUU elements and each PUU element computes for the polynomial $\theta_v(X)$ in equation (2). The updated coefficients will be stored in the registers at the output and will be sent to the DCCU simultaneously to overlap the operations of the PUU and the DCCU. After finishing update in the registers, the data in the registers will be stored in the memory immediately, so the memory access occurs once every $(r+1)$ clock cycles. The number of memory accesses is reduced and the memory structure is simplified by applying the proposed scheduling.

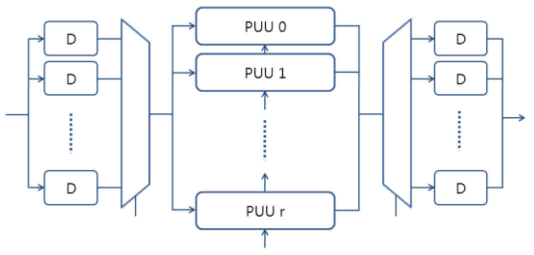


Figure 7. PUU overall structure.

D. Polynomial Order Sorting Unit (POSU)

In each iteration of the interpolation algorithm, the polynomial $Q_\eta(X,Y)$ needs to be selected by the weighted degree and the DC computed in the DCCU. Fig.8 shows the structure of the proposed POSU. Instead of computing the weighted degrees of the candidate polynomials to select $Q_\eta(X,Y)$ each iteration, we save the candidate polynomials with their weighted degrees once in the internal memory and update the weighted degrees every iteration. The proposed POSU has $(r+1)$ registers that store the weight degree and its index in one word. As shown in equation (10), the degree of the polynomial $Q_\eta(X,Y)$ will be increased by one due to the multiplication by $(X+x)$, whereas the degrees of the other polynomials remain the same.

Figure 8 shows the POSU structure that reorders the polynomials by their weighted degrees. The registers consist of the $(r+1)$ shift registers and each register is partitioned to the part for the weighted degrees (O_v) and the part for the index of the polynomials (v), where O_v is initialized to $1,(k-1),2(k-1),\cdots,r(k-1)$ (k is message symbol, r is the number of the candidate polynomials) and v is initialized to $0,1,2,\dots,r$. The weighted degree O_v is increased by one in case that $v=\eta$ and a bubble sort is performed to reorder the weighted degrees using comparator and shift registers.

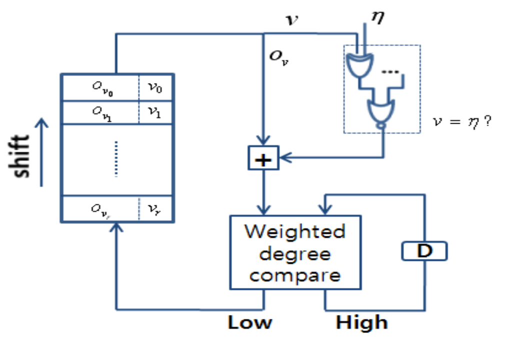


Figure 8. POSU structure.

IV. Design and Performance Analysis

In this section, we apply the proposed architecture to the $RS\ code\ (n,k)=(255,239)$ defined on $GF(2^8)$. We analyze the proposed architecture in terms of hardware cost, latency, and performance and compare it to existing architectures. The primitive polynomial used in this paper is $p(x)=1+x^2+x^3+x^4+x^5$. For fair comparison, we use the same experimental condition as that used in [6]. In the KV front-end, the maximum multiplicity, m_{max}, equals 5, using the low complexity method in the case of $\lambda=5$. The simulation shows the soft-decision decoder with an interpolation cost, C = 3,800, can provide more than 0.5dB of coding gain at a codeword error rate of 10^{-5} compared to the hard-decision decoder [11]. The following equation can be applied to estimate the number of bivariate polynomials [6].

$$r=min\left\{t\in Z : (t+1)\left(\frac{t(k-1)}{2}+k\right)>C\right\},$$

where Z denotes the set of all integers and C is the cost of the interpolation. By applying these parameters to this equation, r equals to 5. The re-encoding technique to reduce the number of iterations is used to achieve higher throughput. The number of iterations when the re-encoding is applied can be computed to $C'=C\times(n-k)/n=3{,}800\times16/255\cong239$. When C' is applied to the interpolation algorithm, the number of degrees of X in the polynomials can be computed to $C'/(r+1)=239/6\cong40$. The total number of latencies is $\left(\left\lceil\frac{C}{2}\right\rceil+\varepsilon\right)C'$, where ε is the number of clocks to select and control the "pivot" polynomial, $Q_\eta(X,Y)$ in Figure 4.

TABLE I.
HARDWARE COMPLEXITY AND LATENCY OF DCCU AND PUU

Module		HW complexity	Latency
DCCU	$GF(2^q)$ multipliers	$3r+1$	$\left\lceil\frac{C}{2}\right\rceil$
	$GF(2^q)$ adders	$r+1$	
	registers	$(2r+3)+\lfloor log_2 r\rfloor$	
PUU	$GF(2^q)$ multipliers	$2(r+1)$	$\left\lceil\frac{C}{2}\right\rceil$
	$GF(2^q)$ adders	$(r+1)$	
	Registers	$(r+1)+2(r+1)^2$	
Total			$(\left\lceil\frac{C}{2}\right\rceil+\varepsilon)C$

Table I shows the hardware cost and latency of the DCCU and the PUU. The architecture in [5] uses the relatively complex dual-port memory. Also, the architectures in [5] and [6] divide the memory into $(r+1)^2$ of small memory modules that require a bigger area and more controls. We expect that the benefit of getting rid of complex access control is more than the degradation of power consumption and memory access speed. All the required coefficients can be read out and stored simultaneously and they are fed to the DCC and the PU in an efficient way. Also, by utilizing one large memory module with one-port, instead of multiple memory banks with dual-ports, the memory structure is simplified and efficient.

TABLE II.
HARDWARE COMPLEXITY AND PERFORMANCE COMPARISON

Design	Area (# of XOR gates	Critical Path (# of gates)	Latency	Throughput (normalized)	Efficiency (normalized)
[5]	8535	12	1437	1	1
[6]	11726	4	1775	2.43	1.77
[13]	7872	10	916	1.88	2.04
[14]	10718	12	454	3.17	2.52
Proposed	1321	4	10650	0.4	2.62

Table II compares the hardware complexity and the performance with the existing architectures. Ahmed, Koetter, and Shanbhag use a point-serial algorithm for the interpolation and apply a parallelism on polynomials and Y degrees [5]. The point-serial algorithm usually improves the performance when the interpolation points have high multiplicities. However, the hardware cost of the DCCU also increases due to τ $(\tau=2^{\lceil log_2 m\rceil})$, which is normally greater than r. The architecture proposed by Wang and Ma [6] also applies to a parallelism on polynomials and Y degrees and uses a hybrid data format for conversion between normal and power representations, so the computation complexity between symbols on finite field is dramatically reduced. However they need hardware for pre- and post-processing for the format conversion, such as a look-up table (LUT).

A finite field multiplier can be implemented by 64 XOR gates and 48 AND gates by employing composite field arithmetic, whereas a finite field adder simply requires 8 XOR gates. As analyzed in [6], the hardware complexity of the interpolation architecture grows linearly with $(m_{max}+1)^2$. For fair comparison, the proposed architecture including the architectures [5], [6] are scaled down with $m_{max}=2$ since m_{max} is equal to 2 in the BGMD architectures [13], [14]. All possible optimizations have been applied to the architectures in [5], [6]. Also, we can apply the pipelining to the proposed architecture for further speedup like the architecture in [6]. Based on that, the critical path can be reduced to the delay of 4 XOR gates. The hardware cost is analyzed based on the following assumption. Each AND gate or OR gate requires 3/4 of the area of an XOR, each MUX or memory cell has the same area as an XOR, and each register occupies about 3 times of the area of an XOR. According to Table I, the hardware complexity of the proposed architecture when $m_{max}=5$ is 5284 in equivalent XOR gates including memory. In case of $m_{max}=2$, the area requirement of the proposed architecture is equivalent to that of 1321 XOR gates. Since the terms of X degrees are processed in serial, the latency of the proposed architecture will be increased to the order of $(\mathrm{r}+1)$, compared to the architecture in [6]. The analysis results for the existing architectures in Table II can be found in the paper [14]. Even though the throughput is relatively low, the efficiency is highest among the architectures in Table II.

TABLE III. RESULTS OF DESIGN AND SYNTHESIS

parameters				performance	
C	m_{ma}	r	ε	total latency	Max clock freq.
239	5	5	4	29636	200 Mhz

	DCCU	PUU	control	total
gate count	7K	11K	7.1K	25.1K

The proposed interpolation architecture for the (255, 239) RS list decoder is designed with VerilogHDL and synthesized using a DongbuAnam 0.18 ㎛ standard cell library. Table III shows the synthesized gate count for the functional blocks in the proposed interpolation architecture. We use $C=239$, $m_{max}=5$, $r=5$, and $\varepsilon=4$ as the design parameters. The maximum operating clock frequency is 200MHz and the synthesized gate count is about 25.1K gates in two-input equivalent NAND gates.

V. CONCLUSIONS

In this paper, we proposed an efficient architecture with low hardware complexity for interpolation in soft-decision list decoding of Reed-Solomon codes. The proposed architecture has several advantages over the existing architectures in the following view points: 1) it employs parallel processing only for Y degrees in bivariate polynomial $Q(X,Y)$ and shares hardware modules, thus reducing the hardware complexity; 2) the schedule is adaptively adjusted according to the “pivot” polynomial computed at each iteration, so the irregular memory access problem is resolved; 3) the number of internal registers is reduced by processing the polynomial monomially; 4) scheduling minimizes the number of memory accesses and simplifies the memory structure by combining and storing data in memory, and the proposed architecture consists of one-port memory and one bank of memory and is efficient in area; 5) the DCCU and the PUU in the proposed architecture are overlapped in schedule, so the total latency is reduced. The proposed interpolation architecture for the (255, 239) RS list decoder is designed with VerilogHDL in a ModelSim environment. After logic synthesis, using the DonbuAnam 0.18um standard cell library, the maximum operating clock frequency is 200MHz and the synthesized gate count is about 25.1K gates in two-input equivalent NAND gates.

ACKNOWLEDGMENT

We are grateful to the IC Design Education Center that provided us with a design environment.

REFERENCES

[1] R. E. Blahut, Theory and practice of Error Control Codes, Addison-Wesley, Reading MA, 1983.

[2] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," IEEE Trans. Inf. Theory, vol. 45, no. 6, pp. 1755-1764, Sep. 1999.

[3] 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.

[4] R. Koetter, J. Ma, A. Vardy and A. Ahmed, "Efficient interpolation and factorization in algebraic soft decision decoding of Reed-Solomon codes," in Proc. of IEEE Symp. On Info. Theory, 2003.

[5] A. Ahmed, R. Koetter, and N. Shanbhag, "VLSI architectures for soft-decision decoding of Reed-Solomon codes," in Proc. ICC, pp. 2584-2590, 2004.

[6] Z. Wang and J. Ma, "High-speed interpolation architecture for soft-decision decoding of Reed-Solomon codes," IEEE Trans. VLSI systems, vol. 14, no. 9, pp. 937-950, Sep. 2006.

[7] W. J. Gross, F. R. Kschischang, and P. Gulak, "Architecture and implementation of an interpolation processor for soft-decision Reed-solomon decoding," IEEE Trans. VLSI systems, vol. 15, no. 3, pp. 309-318, Mar. 2007.

[8] W. J. Gross, F. R. Kschsichang, R. Koetter, and P. G. Gulak, "A VLSI architecture for interpolation in soft decision list decoding of Reed-Solomon codes," in Proc. of IEEE Workshop on Signal Processing Systems, 2002.

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

[10] 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 Trans. Inf. Theory, vol. 37, no. 5, pp. 1274-1287, Sep. 1991.

[11] H. Hasse, "Theorie der Hoheren differentiale in einem algebraischen funktionenkorper mit vollkommenem konstantenkorper bei beliebiger charakteristik," J. Reine. Ang. Math., vol. 175, pp. 50-54, 1936.

[12] V. C. da Rocha Jr., "Digital sequences and the Hasse derivative," in Communications Coding and Signal Processing, B. Honary, M. Darnell, and P. Farrell (Eds.), Communication Theory and Applications, John Wiley and Sons Inc., vol. 4, pp. 256-268, 1997.

[13] J. Zhu, X. Zhang, and Z. Wang "Backward interpolation architecture for algebraic soft-decision Reed–Solomon decoding," IEEE Trans. VLSI systems, vol. 17, no. 11, pp. 1602-1615, Nov. 2009.

[14] X. Zhang and J. Zhu, "High-throughput interpolation architecture for algebraic soft-decision Reed–Solomon decoding," IEEE Trans. Circuits and systems, vol. 57, no. 3, pp. 581-591, Mar. 2010.