A VLSI DESIGN OF A HIGH-SPEED REED-SOLOMON DECODER

Hanho Lee

Agere Systems, Platform IP Lab,
1247 S. Cedar Crest Blvd., Allentown, PA 18103, USA

ABSTRACT

Reed-Solomon (RS) codes have been widely used in a variety of communication systems to protect digital data against errors occurred in the transmission process. This paper presents a VLSI implementation of a high-speed 8-error correcting RS(255,239) decoder architecture using modified Euclidian algorithm for the communication systems. The RS decoder has been designed and implemented with the 0.16-μm CMOS standard cell technology with a supply voltage of 1.5V. The results show that the proposed RS decoder operates at a clock frequency of 670 MHz and has a data processing rate of 3.36 Gbits/s.

I. INTRODUCTION

Reed-Solomon (RS) codes have been widely used in a variety of communication systems such as space communication link, digital subscriber loops, and wireless systems as well as in networking communications [1]. RS decoders can be used to protect digital data against errors occurred and reduce the signal to noise ratio in the transmission process. The high-speed implementations of RS decoders are often necessary to meet continual demand for ever higher data rates. The RS decoder can be implemented using modified Euclidian (ME) algorithm or Berlekamp-Massey (BM) algorithm to solve a key equation. For either algorithm, a finite-field (also called Galois-field) is a mathematical structure which plays a crucial role in the theory of RS codes and the finite-field arithmetic operations are the fundamental building blocks for RS decoder [2],[3].

A syndrome-based RS decoder consists of three components. First part is a syndrome computation block. It generates a syndrome polynomial $S(x)$ that is used in the second component for solving a key equation. Either ME algorithm or BM algorithm can be used to solve the key equation $S(x)σ(x) = ω(x) \mod x^{2t}$ for an error-locator polynomial $σ(x)$ and an error-value polynomial $ω(x)$. Then in the third component, these two polynomials are used to find out the error locations and the corresponding error values according to Chien search and Forney algorithm, and corrects the errors as the received word is being read out of the decoder. In addition, a FIFO memory is used in order to buffer the received symbols according to the latency of these components. A $t$-error correcting primitive RS($n,k$) code with symbols from $GF(2^m)$ has codewords of length $n = 2^m - 1$ and satisfies $n - k = 2t$.

In this paper, we present a VLSI design and implementation of a high-speed 8-error correcting RS(255,239) decoder using ME algorithm, which is used for submarine system [4], as shown in Fig. 1. A key advantage of RS decoders based upon the ME algorithm is regularity. In addition, their regular data flow structure can be easily pipelined and implemented in VLSI [5]-[7].

This paper begins with brief introduction of the RS encoding algorithm in Section II. Sections III-VI describe the proposed architectures for the decoder components and illustrate the decoding process. The implementation and results of the proposed RS decoder are described in Section VII. Finally, the conclusions are given in Section VIII.

II. ENCODING OF REED-SOLOMON CODES

Consider an RS($n,k$) code of block length $n = 2^m - 1$ with each block containing $k$-bit information symbols over the finite-field $GF(2^m)$. The number of error symbols that can be corrected by such an RS code is $t = \lfloor \frac{n-k}{2} \rfloor$. Since RS codes belong to the category of cyclic codes, they can be completely specified by their generator polynomials. A generator polynomial of a $t$-error correcting RS code takes the following form: $g(x) = \prod_{i=0}^{t-1} (x + \alpha^{i}) = (x + \alpha^{t_0})(x + \alpha^{t_0 + 1}) \cdots (x + \alpha^{t_0 + 2^{m-1} - 1})$, where $α$ is a primitive element in $GF(2^m)$ and $t_0$ is an arbitrary number between zero and $n-1$. Different generator polynomials are formed with different values of $t_0$; by carefully choosing the constant $t_0$, the circuit complexity of the encoder and decoder can be reduced. The primitive polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ is used for multiplication and division in $GF(2^8)$. The $k$ information
symbols \( D = (d_{n-1}, \cdots, d_{n-k+1}, d_{n-k}) \) are consecutively fed into the encoder, and the \( n-k \) parity symbols \( P = (p_{n-k-1}, \cdots, p_1, p_0) \) are obtained by \( P(x) = D(x) \mod G(x) \), where \( D(x) = d_{n-1}x^{n-1} + \cdots + d_{n-k+1}x^{n-k+1} + d_{n-k}x^{n-k} \) and \( P(x) = p_{n-k-1}x^{n-k-1} + \cdots + p_1x + p_0 \). The parity symbols can be obtained one clock cycle after the last information symbol has been fed into the encoder, i.e., \( k \) clock cycles are needed to compute the parity symbols. The codeword symbols of length \( n, C = (d_{n-1}, \cdots, d_{n-k}, p_{n-k-1}, \cdots, p_0) \), are outputted serially.

III. A PIPELINED FULLY-PARALLEL MULTIPLIER

The design of finite-field multipliers depends on the choice of basis for the representation. Here, we consider only the standard polynomial basis in which the \( m \)-bit byte \((a_{m-1}, a_{m-2}, \cdots, a_1, a_0)\) represents the finite-field elements \( A = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0 \). The product \( W \) of two \( GF(2^m) \) elements \( A \) and \( B \) is computed as follows:

\[
W = A \cdot B = \sum_{k=m}^{2m-2} d_k \lambda^k + \sum_{k=0}^{m-1} d_k \lambda^k \tag{1}
\]

where \( d_k = \sum_{i=0}^{k} a_i b_{k-i} \). Since \( \lambda^k \in GF(2^m) \), there exist unique \( g_i^{(k)} \) \((0 \leq i \leq m-1)\), such that \( \lambda^k = \sum_{i=0}^{m-1} g_i^{(k)} \lambda^i \) for \( m \leq k \leq 2m - 2 \). In this way and by changing notations, the equation (1) becomes

\[
W = \sum_{k=0}^{m-1} w_k \lambda^k \tag{2}
\]

where \( w_k = d_k + \sum_{j=m}^{2m-2} d_j g_{j-k} \). The equation (2) represents the final result for the product \( W \). The multiplication consists of two procedures, i.e., the multiplication as in equation (1) and the modular reduction as in equation (2). There are no data dependences in both procedures and they can be parallelly computed, respectively. Using a polynomial to represent a field element, the finite-field multiplication of \( A \) and \( B \) can be more concisely formatted as \( W(x) = A(x)B(x) \mod p(x) \). Fig. 2 shows the logic diagram of a pipelined fully-parallel multiplier over \( GF(2^8) \). The upper part is for parallel multiplication and the lower part is for parallel modular reduction, where \( g_i^{(k)} \) \((0 \leq i \leq 7, 8 \leq k \leq 14)\) are pre-computed values. This fully-parallel multiplier is pipelined between upper part and lower part. Thus, the critical path delay is \( T_{\text{pipe, mult}} = T_{DFF} + T_{\text{snd2}} + T_{\text{xor2}} + T_{\text{xor3}} \), where \( T_{DFF}, T_{\text{snd2}}, T_{\text{xor2}} \) and \( T_{\text{xor3}} \) are the delays of D Flip-flop, 2-input AND, XOR and 3-input XOR gates. The pipelined multiplier structure will provide significant gains for critical path delay.

IV. SYNDROME COMPUTATION BLOCK

Let \( C(x) \) and \( R(x) \) be the code word polynomial and the received polynomial, respectively. The received polynomial can be corrupted by channel noise during transmission. This can be described as \( R(x) = C(x) + E(x) = R_{n-1}x^{n-1} + \cdots + R_1x + R_0 \), where \( E(x) \) is the error polynomial.

The first step in the decoding algorithm is to calculate 2t syndromes \( S_i, 1 \leq i \leq 2t \), which are used to correct correctable errors. The syndrome polynomial \( S(x) \) is defined as \( S(x) = \sum_{i=0}^{2t-1} S_i \cdot x^i \) with \( S_i = \sum_{j=0}^{n-1} r_j \cdot \alpha^{ij} \), where \( \alpha \) is a root of primitive polynomial \( p(x) = x^8 + x^4 + x^3 + x^2 + 1 \) and \( t = 8 \), hence a primitive element in \( GF(2^8) \). For RS(255,239)
code, $\alpha^j$'s ($0 \leq j \leq 254$) denote the possible error locations. The received polynomial $R(x)$ is a valid codeword when and only when all the syndromes $S_i$ ($1 \leq i \leq 16$) are zero.

The syndrome computation block shown in Fig. 3 accepts the received symbols which are transmitted through noisy channel. It considers the symbol values as polynomial coefficients and determines if the series of symbols contained in a data block form a valid code word for the particular RS code chosen. It evaluates the polynomial for $2t$ syndrome values and detects whether the evaluations are zero (the data block is a code word) or non-zero (the data block is not a code word). Any block that is not a code word is corrupted by errors. $2t$ simple recursive multiply-accumulate circuits render it possible to compute the syndromes within $n$ symbol periods. The syndrome symbols $(S_1, S_2, \cdots, S_{15})$ are outputted serially using parallel-serial converter.

V. MODIFIED EUCLIDEAN ALGORITHM BLOCK

We need to obtain the error-locator $\sigma(x)$ and error-value $\omega(x)$ polynomials by solving the key equations $S(x)\sigma(x) = \omega(x) \mod x^{2t}$. The error-locator and error-value polynomials can be obtained by using ME algorithm. The ME algorithm is summarized as follows. Initially, $R_0(x) = x^{2t}$, $Q_0(x) = S(x)$, $L_0(x) = 0$, $U_0(x) = 1$. In $i$-th iteration,

$$R_i(x) = [\sigma_{i-1} b_{i-1} R_{i-1}(x) + \bar{\sigma}_{i-1} a_{i-1} Q_{i-1}(x)] - x^{[i-1]}[\sigma_{i-1} a_{i-1} Q_{i-1}(x) + \bar{\sigma}_{i-1} b_{i-1} R_{i-1}(x)],$$

$$Q_i(x) = \sigma_{i-1} Q_{i-1}(x) + \bar{\sigma}_{i-1} b_{i-1} R_{i-1}(x),$$

$$L_i(x) = [\sigma_{i-1} b_{i-1} L_{i-1}(x) + \bar{\sigma}_{i-1} a_{i-1} U_{i-1}(x)] - x^{[i-1]}[\sigma_{i-1} a_{i-1} U_{i-1}(x) + \bar{\sigma}_{i-1} b_{i-1} L_{i-1}(x)],$$

$$U_i(x) = \sigma_{i-1} U_{i-1}(x) + \bar{\sigma}_{i-1} b_{i-1} L_{i-1}(x),$$

where $a_{i-1}$ and $b_{i-1}$ are the leading coefficients of $R_{i-1}(x)$ and $Q_{i-1}(x)$, respectively, and

$$l_{i-1} = \text{deg}(R_{i-1}(x)) - \text{deg}(Q_{i-1}(x)), \quad \sigma_{i-1} = \begin{cases} 1 & \text{if } l_{i-1} \geq 0 \\ 0 & \text{if } l_{i-1} < 0. \end{cases}$$

The algorithm stops when $\text{deg}(R_i(x)) < t$. If the stop condition is satisfied, then $\omega(x) = R_i(x)$ and $\sigma(x) = L_i(x)$.

Fig. 4(a) shows the ME algorithm processing element (PE). A systolic array of $2t$ PEs shown in Fig. 4(b) computes the error-locator polynomial $\sigma(x)$ and error value polynomial $\omega(x)$, and it is capable of performing the ME algorithm continuously. The basic PE consists of a degree computation (DC) block and a polynomial arithmetic (PA) block. DC block has the following two roles. First, it performs the control to determine when two polynomials are to be exchanged and when the initial polynomial and product polynomial are to be exchanged. Thus, exchange control circuits control whether the first ($R_i(x)$) and second data lines ($xQ_i(x)$) are exchanged or not, and also control whether the third ($L_i(x)$) and fourth data lines ($xU_i(x)$) are exchanged or not. That is, exchange control circuit computes $i = \text{deg}(R_i(x)) - \text{deg}(Q_i(x))$, if $\text{deg}(R_i(x)) < \text{deg}(Q_i(x))$, then $sw = 1$, otherwise $sw = 0$. Second, it detects if the arithmetic oper-
Figure 5: Chien search block.

Figure 6: Forney algorithm and error correction block.

For $i = 1$ to $255$

If $(\sigma(\alpha^i) == 0)$ then

$$cc_{255-i} = R_{255-i} + \frac{\omega(\alpha^i)}{\sigma(\alpha^i)}$$

End If

End For

where $\sigma'(x)$ is the derivative of $\sigma(x)$; $R_i$ and $cc_i$ denote the $i$-th symbol in the received polynomial and the corrected codeword polynomial, respectively. Note that the computation of error locations and values involves evaluating three polynomials $\sigma(\alpha^i)$ (of degree $t$), $\omega(\alpha^i)$ (of degree $t-1$) and $\sigma'(\alpha^{-i})$ (of degree $t-1$). Rewriting $\sigma(x)$ as the sum of the even terms $\sigma_{even}(x)$ and the odd terms $\sigma_{odd}(x)$, we have $\sigma_{odd}(x) = x^t \cdot \sigma'(x)$. Therefore, first computing $\sigma_{even}(\alpha^i)$ and $\sigma_{odd}(\alpha^i)$ and then $\sigma(\alpha^i)$ can eliminate the computation of $\sigma'(\alpha^i)$.

The block diagram for Chien search is shown in Fig. 5. Fig. 6 shows the block diagram of the Forney algorithm and error correction block in which Forney algorithm generates the error value. A straightforward approach for computation of the inverse of a nonzero element in $GF(2^8)$ is to use a 256x8 ROM in which inverse of the field elements are stored. As each error value is computed, the corresponding received symbol is fetched from a FIFO memory, which buffers the received symbols during the decoding process. Each error value is simply added to the received symbol to produce the correct symbol.

VII. VLSI IMPLEMENTATION

We have implemented the proposed RS decoder using Verilog HDL and performed logic synthesis using SYNOPSYS design tool with 0.16-μm, 1.5V CMOS technology. Table 1 shows the implementation results for the proposed RS(255,239) decoder using ME algorithm. The total number of gates is 77,030 from the synthesized results excluding the FIFO memory and the clock frequency is 670 MHz. The pipelined multiplier structure in ME algorithm block was used to reduce the critical path delay and has provide significant gains for clock frequency. Furthermore, DC block was
Table 1: Implementation results for RS(255,239) decoder

<table>
<thead>
<tr>
<th>Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syndrome computation</td>
<td>2,870 gates</td>
</tr>
<tr>
<td>ME algorithm block</td>
<td>70,680 gates</td>
</tr>
<tr>
<td>Chien search, Forney</td>
<td>3,480 gates</td>
</tr>
<tr>
<td>FIFO memory</td>
<td>355×8 bits</td>
</tr>
<tr>
<td>Total number of Gates</td>
<td>77,030 gates</td>
</tr>
<tr>
<td>Latency</td>
<td>355 clocks</td>
</tr>
<tr>
<td>Critical path delay</td>
<td>1.5 ns</td>
</tr>
<tr>
<td>Clock frequency</td>
<td>670 MHz</td>
</tr>
<tr>
<td>Data processing rate</td>
<td>5.36 Gbits/s</td>
</tr>
</tbody>
</table>

Fig. 7 shows the timing chart for RS decoder using ME algorithm. The syndrome computation block provides 2t syndromes simultaneously for the next blocks one clock cycle after the last received symbol has been fed into the block. The ME algorithm processing block accepts syndromes and outputs the coefficients of σ(x) and ω(x) serially. The proposed RS decoder takes in code blocks consecutively, performs the appropriate coding operation and outputs the data with a fixed latency of n + 10t + 20 clock cycles, where n is the processing delay required for computing syndromes and 10t + 8 clock cycles are the delay required for the ME algorithm processing block including serial-parallel converter and 12 clock cycles are the delay required for the Chien-search, Forney algorithm and error correction blocks.

VIII. CONCLUSIONS

We have presented a high-speed RS decoder using ME algorithm. Their regular data flow structure can be highly pipelined and easily implemented in VLSI. The pipelining the multiplications result in an order of magnitude reduction in the critical path delay. This RS decoder has been implemented with the 0.16-μm CMOS technology with a supply voltage of 1.5V. The clock frequency and maximum data processing rate are 670 MHz and 5.36 Gbit/s, respectively, under the worst case condition.

IX. REFERENCES