AN ULTRA-FAST REED-SOLOMON DECODER SOFT-IP WITH 8-ERROR CORRECTING CAPABILITY

T. Yamane and Y.Katayama

IBM Research, Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato, Kanagawa 242-8502 Japan
{tyamane, yasunaok}@jp.ibm.com

ABSTRACT

We present algorithm and IP design of a parallel Reed-Solomon decoder with up to 8-byte random error correcting capability. The decoder soft-IP consists of a core that can be designed as parallel combinational circuits of around 62K primitive gates and a peripheral that can be arranged flexibly depending on codeword configurations. The technology mapping results with even commercial FPGA demonstrates that a single core can achieve a throughput well over 40 Gbps when it is 4-stage pipelined. A single decoder design can process N-interleaved codewords efficiently if the core is operated in a time division multiplexing manner.

1. INTRODUCTION

Reed-Solomon codes are known for their advanced error-correcting capabilities and used extensively in a number of practical applications such as storage systems, communications and fault tolerant computing [1]. In recent applications to the optical communications, Reed-Solomon decoders are required to have a capability of 8-error correction and adaptability to interleaved codewords, and to operate at a speed of 40 Gbps and beyond.

Historically, there have been a number of fast and/or compact Reed-Solomon decoder designs, most of which are serial input/output sequential circuits shown in Fig. 1 (a). See, for example, [2] and references therein. These decoders, however, cannot achieve 40 Gbps data rate as a single decoder because of their direct implementation of conventional sequential algorithms with conditional branches, such as Berlekamp-Massey or (extended) Euclid algorithms. Therefore, it is indispensable to duplicate multiple sequential Reed-Solomon decoders in parallel as to achieve the necessary processing speed, which increases the total circuit size proportionally as the speed of the transmission increases. Various attempts have been made by [3], [4] and [5] to improve the hardware implementation of the conventional Reed-Solomon decoding algorithms, but they still require multiple decoders to achieve 40 Gbps.

On the other hand, an algorithm and architecture of high-performance 4-error-correcting Reed-Solomon decoders and design optimization methodology were presented in [6] [7]. Our decoder architecture is shown in Fig. 1 (b). The algorithm proposed to use a new error calculation polynomial of degree up to $t-1$ (t is the maximum number of correctable symbols) that can give error values at error locations without costly symbol-by-symbol divisions. A research prototype (40-34,32)Reed-Solomon decoder in [7] achieved 45ns latency, corresponding to 7 Gbps throughput, with 0.35 μm CMOS technology. A systematic and efficient extension of our previous approach to 8-error-correcting Reed-Solomon codes were recently given from the viewpoint of decoding algorithm [8], [9].

The aim of the present paper is to show that an ultra high-performance and flexible Reed-Solomon decoder with a compact circuit size, low decoding latency and low power consumption can be achieved for up to $t = 8$ Reed-Solomon codes, based on this improved algorithm. The performance can be further improved at the same time compared to the previous research prototype. In particular, our soft-IP decoder design with a commercial FPGA can give a throughput well beyond 40 Gbps when the core is 4-stage pipelined. Furthermore, the decoder can process interleaved codewords efficiently.

2. DECODING ALGORITHM OVERVIEW

Since our decoding algorithm for $t = 8$ is given for error locations [8] and error values [9] separately, let us give an overview of how the entire decoding process works, here. Throughout the present paper, we denote the codeword length of Reed-Solomon codes by n and the maximum number of correctable errors by t. We assume Reed-Solomon codes are defined over GF(2^m).

Once the syndromes are calculated, we shall start from Peterson approach, which is direct computation of Yule-Walker equation. The difference between previous $t = 4$ algorithm [6] and $t = 8$ algorithm is that the former needs

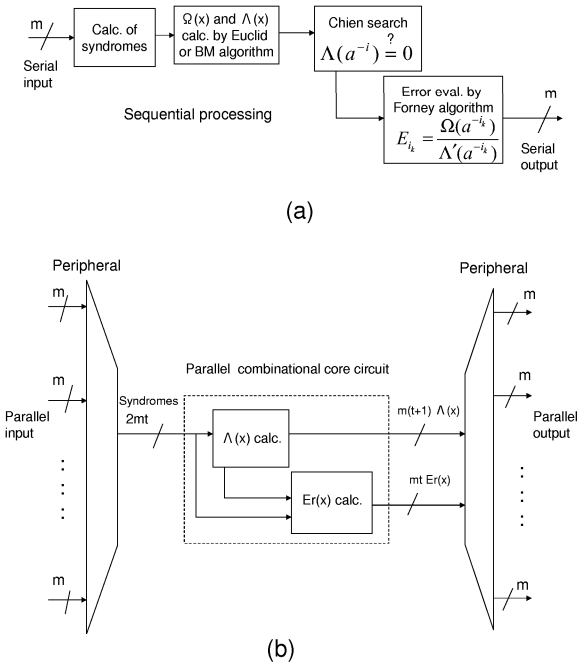

Fig. 1. Block diagram of Reed-Solomon decoders; (a) Conventional approaches. (b) Our approach.

to deal with the following asymmetric determinants

$$\Lambda_i^{(l)} = \frac{\tilde{\Lambda}_i^{(l)}}{\tilde{\Lambda}_0^{(l)}}, \quad \tilde{\Lambda}_i^{(l)} = \begin{vmatrix} S_0 & \cdots & S_{l-1} \\ \vdots & \ddots & \vdots \\ S_{l-i-1} & \cdots & S_{2l-i-2} \\ S_{l-i+1} & \cdots & S_{2l-i} \\ \vdots & \ddots & \vdots \\ S_l & \cdots & S_{2l-1} \end{vmatrix},$$

while the latter covert the calculation of the above asymmetric determinants into symmetric ones using the following Jacobi's formula :

$$\Gamma_i^{(l+1)} \tilde{\Lambda}_0^{(l)} + (\tilde{\Lambda}_i^{(l)})^2 = \tilde{\Lambda}_0^{(l+1)} \Gamma_{i-1}^{(l)},$$

Here, S_i are syndromes, $\Lambda_i^{(l)}$ are the coefficients of the l-th error locator polynomial and $\Gamma_i^{(l+1)}$ is the $(l+1-i, l+1-i)$ cofactor of $\tilde{\Lambda}_0^{(l+1)}$ given by

$$\Gamma_i^{(l+1)} = \begin{vmatrix} S_0 & \cdots & S_{l-1-i} & S_{l+1-i} & \cdots & S_l \\ \vdots & & \vdots & \vdots & & \vdots \\ S_{l-1-i} & \cdots & S_{2(l-1-i)} & S_{2(l-i)} & \cdots & S_{2l+1-i} \\ S_{l+1-i} & \cdots & S_{2(l-i)} & S_{2(l+1-i)} & \cdots & S_{2l+1-i} \\ \vdots & & \vdots & \vdots & & \vdots \\ S_l & \cdots & S_{2l-1-i} & S_{2l+1-i} & \cdots & S_{2l} \end{vmatrix}.$$

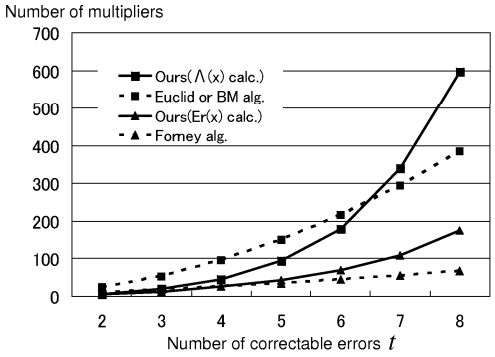

Fig. 2. Comparison for $\Lambda(x)$ calculation and $Er(x)$ calculation

This branchless algorithm is suitable for a combinational circuit and can reduce the number of multipliers thanks to the symmetry of $\Gamma_i^{(l+1)}$. In addition, the latency of this algorithm is equal to t steps in a unit of a multiplier.

To evaluate the error values E_{i_k}, we use the same error calculation polynomial $Er^{(l)}(x)$ as the one used in $t = 4$

$$\begin{aligned} Er^{(l)}(x) &\equiv \frac{\sum_{k=0}^{l-1} E_{i_k} \prod_{j \neq k} (x + a^{i_j}) \prod_{h,l \neq k, h > l} (a^{i_h} + a^{i_l})}{\prod_{h>l} (a^{i_h} + a^{i_l})} \\ &= \frac{1}{f^{(l)}} \sum_{j=0}^{l-1} \left\{ \sum_{h=0}^{l-1} \sum_{i=0}^{l-j-1} S_{h+l} \Lambda_{l-j-i-1}^{(l)} f_h^{(l-1)} \right\} x^j, \end{aligned}$$

where a is a primitive element of the finite field GF(2^m) and a^{-i_k} are error locators. Both $f^{(l)}$ and newly introduced $f_h^{(l-1)}$ are functions of $\Lambda_1^{(l)}, \ldots, \Lambda_l^{(l)}$ and are given by

$$f^{(l)} = \begin{vmatrix} \Lambda_1^{(l)} & \Lambda_3^{(l)} & \cdots & \cdots & \cdots & 0 \\ 1 & \Lambda_2^{(l)} & \Lambda_4^{(l)} & \cdots & \cdots & 0 \\ 0 & \Lambda_1^{(l)} & \Lambda_3^{(l)} & \Lambda_5^{(l)} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \Lambda_{l-4}^{(l)} & \Lambda_{l-2}^{(l)} & \Lambda_l^{(l)} \\ 0 & \cdots & \cdots & \cdots & \Lambda_{l-3}^{(l)} & \Lambda_{l-1}^{(l)} \end{vmatrix},$$

$$f_h^{(l-1)} = \begin{vmatrix} \Lambda_1^{(l)} & \Lambda_3^{(l)} & \cdots & \cdots & 0 \\ 1 & \Lambda_2^{(l)} & \Lambda_4^{(l)} & \cdots & 0 \\ 0 & \Lambda_1^{(l)} & \Lambda_3^{(l)} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \Lambda_{l-5}^{(l)} & \Lambda_{l-3}^{(l)} & \Lambda_{l-1}^{(l)} \\ 0 & \cdots & \cdots & \Lambda_{l-h-4}^{(l)} & \Lambda_{l-h-2}^{(l)} \end{vmatrix},$$

where $\Lambda_0^{(l)} = 1, \Lambda_i^{(l)} = 0, i < 0$. These determinants calculation becomes a major step in calculating coefficients of

$Er(x)$ and improved algorithm is used for up to $t = 8$ decoding. They are systematically computed in a closed and shared manner based on their regular structure [9]. Note that the second expression of $Er^{(l)}(x)$ contains only S_i and $\Lambda_k^{(l)}$, neither a^{i_k} nor E_{i_k} themselves explicitly. This makes the computation of $Er^{(l)}(x)$ independent of the search of the error locators and leads to a parallel computation of them. Furthermore, the $Er^{(l)}(x)$ defined here needs only one inverter since the division in the definition is common to all symbols. This allows much cost saving compared to the conventional Forney algorithm where costly divisions are necessary at multiple error locations. The branchless computation of $Er(x)$ is suitable for combinational circuits and the latency of the critical path is $t-2$ in a unit of a multiplier and one division.

Figure 2 compares our algorithms with the conventional Euclid or Berlekamp-Massey(BM) algorithm and Forney algorithm with respect to the number of necessary multiplications. Our approach is comparable to those of the conventional schemes at the algorithm level even though it remains as a closed form algorithm. When our algorithm is mapped to combinational circuits, it can become more efficient in terms of circuit size, since additional circuit elements such as clocks, registers, branch related circuit, and controllers are not required. Furthermore, our algorithm enables the completely parallel decoder architecture shown in Fig. 1 (b). As a result, the decoder circuit can achieve ultra-high performance without duplicating conventional decoders shown in Fig. 1 (a).

3. SOFT-IP DECODER DESIGN

3.1. Core architecture

Closed-form decoding algorithms suitable for parallel combinational circuits can be mapped directly to the combinational circuit. The algorithm is described using Mathematica and translated into VHDL code after symbolic verification is performed. The gate size is further reduced by using a logic optimization technique. As a result, the size of the parallel combinational core circuit is around 62K primitive gates in case of $t = 8$ and $m = 8$ (XOR and AND are counted as 1 gate).

Figure 3 shows the diagram of an enhanced version of our decoder core, which exploits 4-stage pipelined design. In the first stage of $\Lambda(x)$ calculation, the 2, 3, 4 and 5 dimensional determinants are calculated. In the second stage, the 6, 7 and 8 dimensional determinants are calculated and the number of errors e is determined according to the values of $\tilde{\Lambda}_0^{(1)}, \ldots, \tilde{\Lambda}_0^{(8)}$. The coefficients of error locator polynomial $\Lambda^{(e)}(x)$ are also selected here. In the first stage of $Er(x)$ calculation, $f^{(l)}$ and $f_h^{(l-1)}$ stated in Section 2 are calculated for $l = 1, \ldots, 8$. In the second stage, $f^{(e)}$ and $f_h^{(e-1)}$

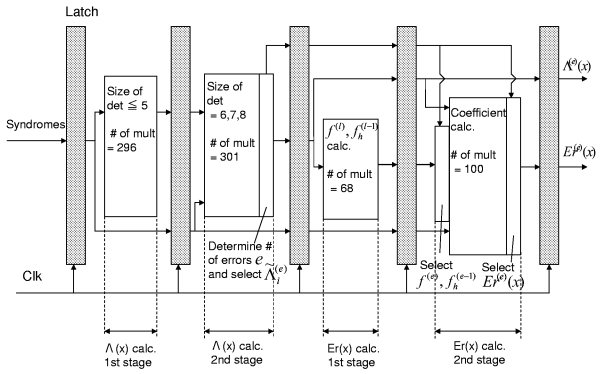

Fig. 3. The 4-stage pipelined parallel combinational core circuit

Table 1. FPGA synthesis results after 4-stage pipelining of the core

	Delay[ns]	Size [slices]
$\Lambda(x)$ calc. 1st stage	24.2	4732
$\Lambda(x)$ calc. 2nd stage	26.2	4648
$Er(x)$ calc. 1st stage	26.0	1211
$Er(x)$ calc. 2nd stage	25.5	1839

are selected according to e and the coefficients of $Er^{(e)}(x)$ are computed.

Table 1 shows the FPGA synthesis result of each stage of the pipelined core circuit with a commercial FPGA (Xilinx Virtex-II Pro). The result indicates that the pipelined core circuit has a potential throughput as high as ∼80 Gbps. By operating the core according to the diagram shown in Fig. 4, the core can process codewords at a speed of 43 Gbps, which is required for 40 Gbps optical communications with the (255,239)Reed-Solomon out-of-band error correction scheme. Still, since only a slow clock speed of around 21MHz is required for the core operation, the power consumption can be kept small. Our Reed-Solomon decoder is advantageous over the recent implementation of 40 Gbps decoders with 0.16 μm CMOS technology [5] because the 40 Gbps throughput is achieved even with a commercial FPGA. Some commercial (255,239)Reed-Solomon soft IPs are now available, for example [10], but they are limited to 10 Gbps applications.

3.2. Peripheral design

We can adapt the parallel combinational decoder to process the N-interleaved codewords by arranging the N peripheral circuits only and operating the core in a TDM manner with the help of mux/demux and registers. Figure 5 shows an example of block diagram for the (255,239) Reed-Solomon

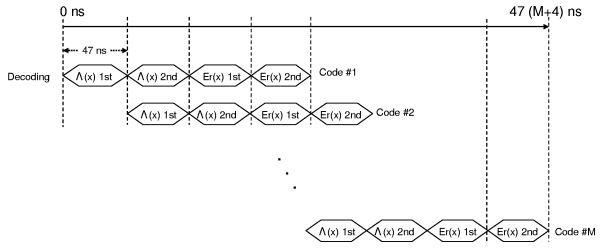

Fig. 4. Timing diagram of the 4-stage pipelined core circuit for 43 Gbps operation

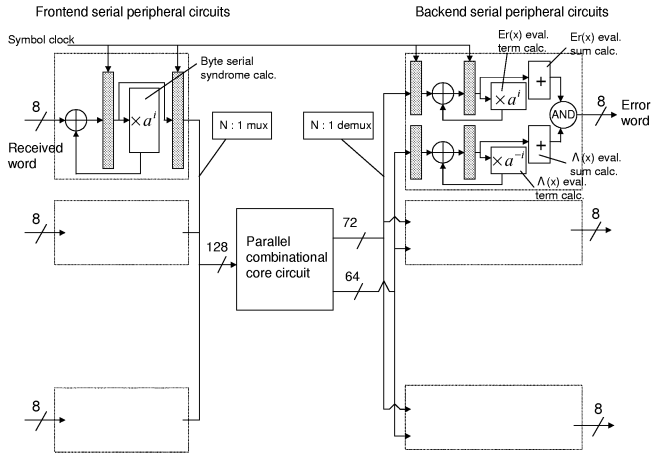

Fig. 5. Blockdiagram of an N-interleaved $(255, 239)$ Reed-Solomon decoder on GF(2^8)

decoder architecture handling N-interleaved codewords. Since the algorithm for error value evaluation stated in Section 2 eliminates costly symbol-by-symbol divisions, all the computations both in the frontend and in the backend are only linear operations. Therefore, it is easy to modify according to the I/O data path configurations.

In the frontend peripheral circuits, input digital signals are interleaved and input in parallel. A syndrome S_i is calculated for each input signal by a sequential circuit and the syndromes for each serial stream are stored in a register. After that, they are multiplexed and sequentially transmitted to the core. The coefficients of $\Lambda(x)$ and $Er(x)$ are demultiplexed according to the number of interleaves, and the results are transmitted to the backend peripheral circuits. The backend peripheral circuits include those for $\Lambda(x)$ evaluation, $Er(x)$ evaluation and AND gates. They are arranged in a number equivalent to the number of interleaves of input digital signals. The $\Lambda(x)$ evaluation circuits search the zero points of $\Lambda(x)$ and output "1" for an error location and "0" otherwise. The $Er(x)$ evaluation circuits compute $Er(a^i), i = 0, \ldots, 2^m - 1$ and give error values at error locations and arbitrary values otherwise. The AND gates multiply these two outputs and compute the error word.

The multiplexers and demultiplexers need to handle not nm bit codewords themselves but $2mt$ bit syndromes and $m(2t+1)$ bit coefficients of $\Lambda(x), Er(x)$, respectively. Thus, the number of necessary multiplexers, demultiplexers and buffers are reduced significantly.

4. CONCLUSION

We have presented an ultra-fast Reed-Solomon decoder soft-IP for 8-error correcting Reed-Solomon codes. Our approach is based on an improved decoding algorithm suitable for combinational circuits up to $t = 8$. The presented soft-IP combines, compact circuit size, low decoding latency, and flexible codeword configuration. The FPGA synthesis results of the pipelined core achieve well beyond 40 Gbps throughput.

5. REFERENCES

[1] Stephen B. Wicker and Vijay K. Bhargava (eds.), Reed-Solomon Codes and their applications, IEEE Press, 1994.

[2] W. Wilhelm, "A New Scalable VLSI Architecture for Reed-Solomon Decoders," IEEE Journal of Solid-State Circuits, Vol.34, No.3, pp.388-396, 1999.

[3] H.M. Ji, "An Optimized Processor for Fast Reed-Solomon Encoding and Decoding," IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3097-3100, 2002.

[4] H-J. Kang and I-C. Park, "A high-speed and low latency Reed-Solomon decoder based on a dual-line structure," IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3180-3183, 2002.

[5] L. Song, M-L. Yu and M.S. Shaffer, "10- and 40-Gb/s Forward Error Correction Devices for Optical Communications," IEEE Journal of Solid-State Circuits, Vol.37, No.11, pp.1565-1573, 2002.

[6] Y. Katayama and S. Morioka, "One-shot Reed-Solomon decoder," Proc. of Conference on Information Science and Systems, pp.700-705, 1999.

[7] S. Morioka and Y. Katayama, "Design Methodology for a One-Shot Reed-Solomon Decoder and Encoder," Proc. of IEEE International Conference on Computer Design : VLSI in Computers and Processors, pp.60-67, 1999.

[8] T. Yamane, Y. Katayama and S. Morioka, "A closed-form calculation of Yule-Walker equation based on Jacobi's formula and its application to decoding of Reed-Solomon codes," Proc. of Conference on Information Science and Systems, pp.124-128, 2001.

[9] Y. Katayama and T. Yamane, "On Calculating Interpolation Polynomials for Error Values in Reed-Solomon Decoding Algorithm, " Proc. of IEEE International Symposium on Information Theory, pp.90, 2002.

[10] http://www.xilinx.com/products/logicore/alliance/cselt/tilab_rs_decoder.pdf