A Fast Reed-Solomon Product-Code Decoder Without Redundant Computations

Hyun-Yong Lee and In-Cheol Park

Department of Electrical Engineering and Computer Science, KAIST
373-1 Guseong-Dong Yuseong-Gu, Deajeon 305-701, Republic of Korea

ABSTRACT

This paper presents a fast and low-power decoding scheme for Reed-Solomon Product-Code (RS-PC). To increase error correction capability of RS-PC, we need to repeat Reed-Solomon (RS) decoding several times for rows and columns. As the number of RS decodings increases, the execution time and the energy consumption also increase as well. In order to reduce unnecessary operations and memory accesses, we added two small memories to store data on whether the row or column is updated or not. In the next iteration, only the updated rows and columns are recalculated instead of the whole rows and columns. Based on the proposed scheme, we implemented a RS-PC decoder using UMC 0.25μm standard and memory cells. The working frequency is 133MHz at 2.5V. Experimental results show that the execution time is reduced by 48% in case of four iterations, and by 66% in case of six iterations.

1. INTRODUCTION

As Reed-Solomon (RS) code is strong to both burst errors and random errors, RS code is being widely used in the applications of communication and data storages. RS codes can be denoted by (N,K), where N and K represent the length of a block and the length of the information symbols, respectively. It can correct up to $t = \lfloor (N-K)/2 \rfloor$ symbols [1].

Recently, more advanced Reed-Solomon Product-Code (RS-PC) has been adopted for DVD, which is now being widely used for high-quality video and data storage. RS-PC is processed by a two-dimensional frame structure illustrated in Fig. 1. It is a matrix composed of a (182,172) RS code in row direction and a (208,192) RS code in column direction. The two-dimensional structure has a greater error correction capability and higher code efficiency, but it eventually increases the size of buffer memory and the number of operations and memory accesses. As a result, the execution time and power consumption of decoding increases significantly.

In RS-PC decoding, we have to process RS decoding for every row and column. Generally, the more we repeat RS decodings for rows and columns, the more the error correction capability of RS-PC increases. In short, the number of errors will decrease in proportion to the number of decoding of rows and columns. On the other hand, execution time and energy consumption will increase as a result because more operations are needed.

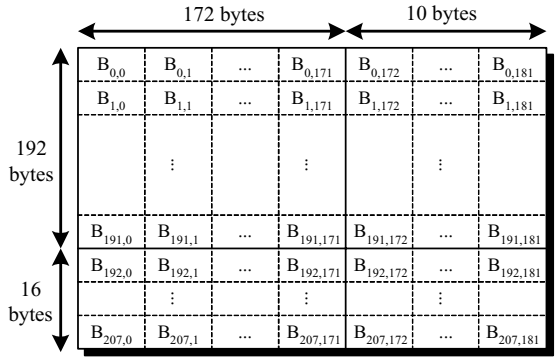

Figure 1. DVD RS-PC Frame Structure

There has been a lot of research that achieved an efficient RS decoder [2][3][4][5][6][10][11]. However, few researches have been done on RS-PC decoder. In a previous research related to RS-PC decoding, an area-efficient RS-PC decoder for DVD applications was presented by Hsie-Chia Chang [7]. In the decoder, the row and column decodings are parallelized by two dedicated RS decoders, and the parity symbols of frames are not corrected to reduce the frame buffer size and minimize the timing constraint. But this architecture is applicable only when rows and columns are decoded each once.

In this paper, we present a scheme to reduce execution time and energy consumption by eliminating unnecessary RS decoding in the process of repetitious decoding of rows and columns. We also estimate the performance of the proposed scheme by comparing its energy consumption with that of the conventional one.

2. OBSERVATION

As the performance of RS-PC decoding varies depending on the decoding method, it is difficult to estimate it accurately. Therefore, a software model written in C-language is used to simulate the performance when a random symbol error rate is given to figure out how many decodings are needed to make 20 frames completely error-free.

In case when the channel environment is not good enough because of many errors, we can enhance the error-correction capability by increasing the number of row and column decodings (as shown in Fig. 2). Many decodings are required to recover the original data in high BER, and it seems that the reasonable number of decodings is four to six.

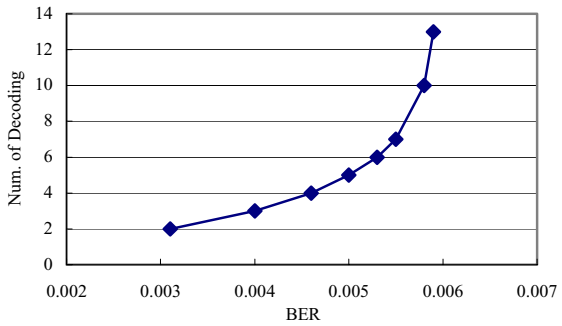

Figure 2. Number of Iterations for Error-free Frames

With the improved semiconductor technologies developed in recent years, we can repeat RS decodings several times for high accuracy while still satisfying a required data processing rate.

However, as the number of decodings increases, the number of computations and memory references also increase as well. In addition, the memories to store RS-PC frames should be significantly larger and be more frequently accessed. Therefore, the memory access power is critical in the energy consumption of the entire RS-PC decoding system.

3. PROPOSED SCHEME

If row and column decodings are processed alternatively, there is no need to decode the rows (columns) that are not modified during the recent column (row) decoding. This is because the RS decodings of the rows (columns) were already carried out on the identical codewords and the errors of the rows (columns) were corrected during the previous row (column) decoding process.

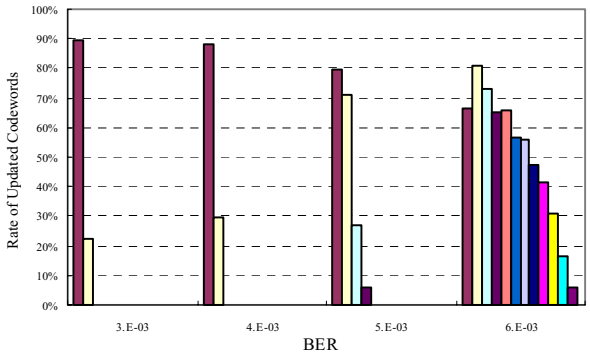

Figure 3. Rate of Updated Codewords

To estimate our proposed scheme, we simulate how the rate of updated codewords changes while the row and column decodings are repeated. As shown in Fig. 3, the updated codewords rate gradually decreases in most cases.

With this in mind, we designed a RS-PC decoder that does not repeat redundant decodings. We added two small memories to store row and column dirty bits. These memories need only 182 bits and 208 bits, each of which shows whether each row and column is modified or not.

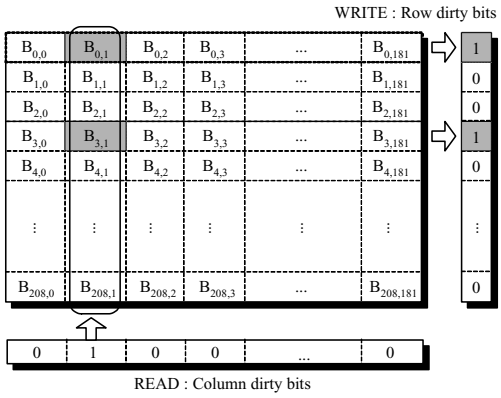

Figure 4. Proposed Scheme

Fig. 4 shows an example of column decoding procedure in the above-mentioned method. If a column decoding corrects a row, the dirty bit of the corresponding row is set to 1. Let us assume that we are conducting row decoding after completing the decoding of all columns. Before starting to decode a row, we check whether the dirty bit of the row is set to 1. If the bit is 1, the row should be decoded again, and if not, we can simply skip the decoding procedure. As it is not necessary to read from the frame memory the rows that have no symbols corrected in the previous column decodings, we can reduce the number of memory accesses and save the processing power needed for RS decoding.

4. RS-PC DECODER IMPLEMENTATION

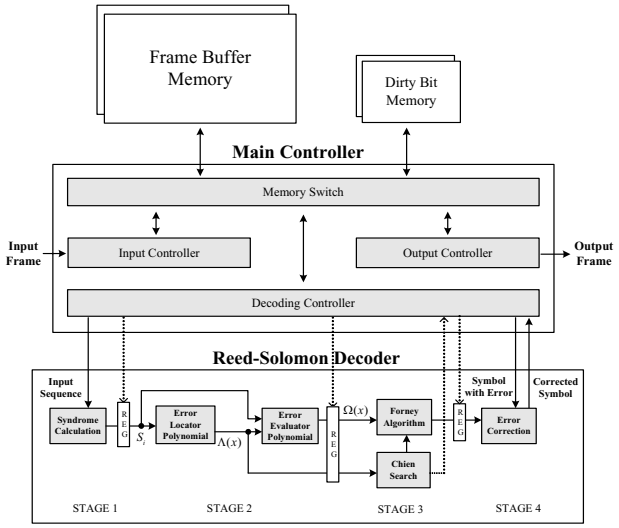

Figure 5. Block Diagram of RS-PC Decoder

Fig. 5 shows the entire architecture of the RS-PC decoder we designed. Following the structure of the most popular RS decoder architecture, we designed the architecture based on algorithms which have high efficiency and low complexity, such as serial inversionless Berlekamp-Massey (BM) algorithm and Forney algorithm [2][3][4][8][9]. Our RS decoder can support

both (182, 172) and (208, 192) RS codes. To control the pipeline flow in the RS decoder, the main controller generates control signals for the each pipeline register in the RS decoder.

Our decoder has two large frame memories. One memory is to store the input stream and send the decoded output. After a decoded symbol is sent out, an input symbol occupies the empty space. The other is used for decoding operation. As these two separated memories work independently, the input stream can be received continuously while the previous frame is being decoded.

We use a dual-port memory for frame buffering. If an error symbol is found, the controller reads the symbol data from the frame memory in the next cycle. In the second cycle, the error is corrected by the error evaluation value. And finally, in the third cycle the corrected symbol data is rewritten over the original symbol data in the frame memory.

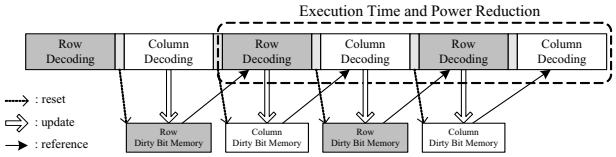

Figure 6. Timing Diagram of Proposed Scheme

Fig. 6 shows the overall timing diagram related to dirty bits in case of six iteration decoding. In the first stage of row and column decodings, we have to decode all rows and columns. However, from the second stages, we may decode only updated rows and columns that can be determined by looking at their dirty bits. And the execution time for decoding a frame can be shorter because we can skip the row or column decoding by adding only one additional cycle to read the dirty bit.

As the word size of a memory is 8 bits, we design the decoder to access 8 dirty bits from the memory at one time. Since the dirty bit requires only 1 bit to represent each row and column, it is smaller in size compared to that of the frame memory. Therefore, the dirty bit memory consumes much less access power than does the frame memory.

Before starting to decode rows and columns, the dirty bit memories should be reset. It takes 26 cycles for rows and 23 cycles for columns to reset the memory. And if an error is found while decoding a row, the dirty bit of the relevant column should be set to 1. These dirty bits of columns are used in the next column decoding. If no error is found in the frame, DONE output signal will be set and the frame in another frame memory will start to be decoded from the next cycle.

5. EXPERIMENTAL RESULT

We designed a DVD RS-PC decoder using Verilog and verified its operation by RTL simulation. During the verification, the RS-PC frames were constructed and some errors were added in C-description. Then we confirmed that the errors were corrected while being processed in RTL simulation. Verilog and C language processes communicate with each other through Program Language Interface (PLI).

The decoder designed by Verilog was synthesized using UMC 0.25μm standard and memory cells. The maximum operating frequency of the decoder is 133MHz at 2.5V.

TABLE 1. Execution Time (Cycles)

BER	1.0E-03	2.0E-03	3.0E-03	4.0E-03	5.0E-03	6.0E-03
Four Iterations	77454	77636	85826	88556	114400	135512
Six Iterations	77454	77636	85826	88556	117364	185822

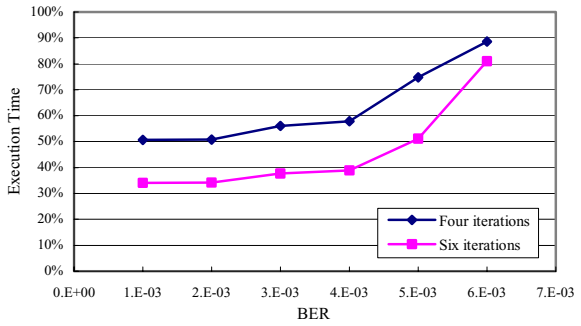

Figure 7. Execution Time with Proposed Scheme

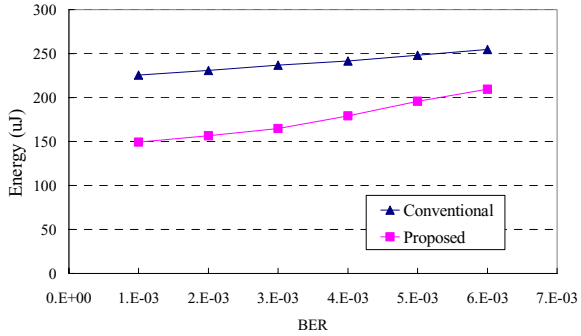

Figure 8. Energy Consumption (Four Iterations)

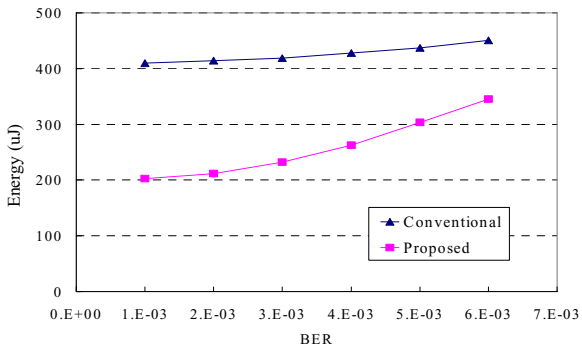

Figure 9. Energy Consumption (Six Iterations)

We repeated the simulation 10 times and averaged the energy consumption results to estimate the energy consumption in case of several error probabilities. Table. 1 is execution times in the proposed RS-PC decoder, and Fig. 7 shows the execution times of the proposed RS-PC decoder ratioed to those of a conventional one. As the BER becomes lower and the number of iterations increases, the execution time decreases.

The execution times of our decoder are shorter than the conventional one. Therefore, the energy consumption for one

frame decoding is more suitable unit to compare than power consumption.

First, we simulated the case of four iterations. The result is shown in Fig. 8, which suggests that we can reduce 19% to 34% of energy with four iterations. In case of six iterations, the energy saving rate is higher than that of four iterations as shown in Fig. 9. The energy saving rate in case of six iterations is 21% to 52%.

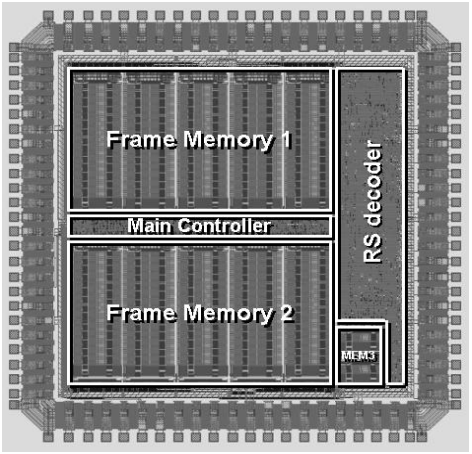

Figure 10. Layout of RS-PC Decoder

The maximum operating frequency of our RS-PC decoder is 133MHz. As the symbol rate of DVD is 4Mbytes/s [7], this chip can support up to 9x DVD with four iterations, and up to 6x DVD with six iterations. Fig. 10 shows the layout that was synthesized using Milkyway and Apollo tool. The chip size is 4mm x 4mm. The memory space of the frame buffer is larger than the non-memory space. The frame memory 1 and frame memory 2 in Fig.10 are the buffers for RS-PC frames, and MEM3 stores row and column dirty bits. The main controller is attached to frame memory 1 and 2 because it should access data from the frame buffer memories for RS decoding.

6. CONCLUSION

In this paper, we have presented a fast and low-power scheme for RS-PC decoding and estimated the execution time and energy consumption in various conditions. For higher accuracy, we repeated more RS decodings. However, as the number of decoding iterations increases, the number of computations and the memory references also increase. To reduce them, we added two small memories to store the dirty bits, each of which indicates whether a row or column is changed or not.

The execution time was measured by RTL simulation and the energy consumption was simulated in the UMC 0.25μm standard cells library using the toggle information of synthesized gate-level circuit. The execution time and energy-saving rate vary significantly according to the BER. In most cases, the iterative row and column RS decoding gradually reduces errors in RS-PC. Therefore, we can expect the execution time and energy significantly can be greatly reduced with the proposed scheme.

In conclusion, we can reduce the execution time and energy consumption of Reed-Solomon Product-Code decoding at the cost of a small additional area.

7. ACKNOWLEDGEMENT

This work was supported by Institute of Information Technology Assessment through the ITRC, by the Korea Science and Engineering Foundation through MICROS center and by IC Design Education Center (IDEC).

8. REFERENCES

[1] Shu Lin and Daniel J. Costello, Jr., *Error Control Coding : Fundamentals and Applications*, IEEE PRESS, 1994.

[2] Hsie-Chia Chang and C. Bernard Shung, "New Serial Architecture for the Berlekamp-Massey Algoritm," *IEEE Trans. Comm.*, vol. 47, No. 4, 1999.

[3] I. Reed, M. Shih, and T. Truong, "VLSI Design of Inverse-Free Berlekamp-Massey Algorithm," *Proc. IEE*, pt. E, vol. 138, pp. 295~298, Sept. 1991.

[4] J.H. Jeng and T. K. Truong, "On Decoding of Both Errors and Erasures of a Reed-Solomon Code Using an Inverse-Free Berlekamp-Massey Algorithm," *IEEE Trans. Commun*., vol.47, pp. 1488~1494, Oct. 1999.

[5] S. Kwon and H. Shin, "An Area-Efficient VLSI Architecture of a Reed-Solomon Decoder-Encoder for Digital VCRs," *IEEE Trans. Consumer Electron.,* vol. 43, pp. 1019~1027, Nov. 1997.

[6] Hyeong-Ju Kang, and In-Cheol Park, "A High-Speed and Low-Latency Reed-Solomon Decoder Based on a Dual-Line Structure," *IEEE ICASSP*, pp. 3180 –3183, 2002.

[7] Hsie-Chia Chang, C. Bernard Shung, and Chen-Yi Lee, "A Reed-Solomon Product-Code (RS-PC) Decoder Chip for DVD applications," *IEEE Journal of Solid-State Circuits*, Vol. 36, No. 2, February 2001.

[8] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Application.* New York: IEEE PRESS, 1994.

[9] E. Berlekamp, *Algebraic Coding Theory*. New York: MacGraw-Hill, 1968.

[10] T. Tuong, W. Eastman, I. Reed, and I. Hsu, "Simplified Procedure for Correcting Both Errors and Erasures of Reed-Solomon Code Using Euclidean Algorithm," *Proc. IEE*, pt. E, vol. 135, no. 6, pp. 318~324, 1988.

[11] K. Liu, "Architecture for VLSI Design of Reed-Solomon Decoders," *IEEE Trans. Computers*, vol C-33, pp. 178~189, Feb. 1984.