논문 2012-49SD-1-4

기가비트 WPAN용 고성능 가변길이 리드-솔로몬 복호기 구조

(High-Performance Variable-Length Reed-Solomon Decoder Architecture for Gigabit WPAN Applications)

최 창 석*, 이 한 호**

(Chang-Seok Choi and Hanho Lee)

요 약

본 논문은 고속 WPAN 시스템에 대한 가변 길이 8-병렬 리드-솔로몬(RS) 복호기에 관한 일반적인 구조를 제안한다. 제안된 구조는 RS(255,239) 코드뿐만 아니라 다양한 단축화 RS 부호들을 지원 할 수 있다. 특히, 가변길이 구조는 다양한 단축화 RS 부호에 대해 가변적인 낮은 지연을 제공하며, 8-병렬 구조를 적용하여 높은 데이터 처리율을 제공한다. 제안된 RS 복호기는 90-*nm* CMOS 표준 셀 기술을 사용하여 성능 분석을 수행하였고, 클록 주파수 300*MHz*에서 19-*Gbps* 데이터 처리율을 제공한다.

Abstract

This paper presents a universal architecture for variable-length eight-parallel Reed-Solomon (RS) decoder for high-rate WPAN systems. The proposed architecture can support not only RS(255,239) code but various shortened RS codes. Moreover, variable-length architecture provides variable low latency for various shortened RS codes and the eight-parallel design also provides high data processing rate. Using 90-*nm* CMOS standard cell technology, the proposed RS decoder has been synthesized and measured for performance. The proposed RS decoder can provide a maximum 19-*Gbps* data rate at clock frequency 300 *MHz*.

Keywords : Reed-Solomon code, forward error correction, decoder, variable-length, wireless personal area network.

Ⅰ. Introduction

Wireless Personal Area Network (WPAN) is the wireless networking technology that supports the communication between physical layer and data link layer. The IEEE 802.15.3 Task Group 3c and ECMA TC-48 are developing a millimeter-wave (mmWAVE) based alternative physical layer (PHY) for WPAN standard[1~2]. mmWAVE is generally 30~300 GHz band and has high throughput for the near field communication. The mmWAVE WPAN will support high data rate applications such as high speed internet access and streaming content download (video on demand, home theater etc.). Also, the goal of mmWAVE WPAN replaces the cable with real-time streaming or wireless data bus.

Due to the nature of mmWAVE, the system requires high throughput and Forward Error Correction (FEC) architecture that can reduce Bit Error Rate (BER). The standard of IEEE 802.15.3c included Reed-Solomon (RS) decoder as the FEC standard. According to ECMA-387 standard, various shortened RS codes are used for header and payload, which are PHY header, ATIF, scrambled MAC

* 학생회원, ** 정회원, 인하대학교 정보통신공학부
(School of Information and Communication Engineering, Inha University)
※ 이 논문은 인하대학교의 지원에 의하여 연구되었음.
접수일자: 2011년5월12일, 수정완료일 2012년1월5일

표 1. 에러 정정 능력이 t=8인 다양한 단축화 RS 부호의 종류

Table 1. Classification of various shortened RS codes with error correction capability t=8 (p=1,…,28).

Type	RS code	# of zero symbol	RS(n,k)
1	RS(8[p+2], 8p)	0	(240,224)
2	RS(8[p+2]-1, 8p-1)	1	(239,223)
3	RS(8[p+2]-2, 8p-2)	2	(238,222)
4	RS(8[p+2]-3, 8p-3)	3	(237,221)
5	RS(8[p+2]-4, 8p-4)	4	(236,220)
6	RS(8[p+2]-5, 8p-5)	5	(235,219)
7	RS(8[p+2]-6, 8p-6)	6	(235,218)
8	RS(8[p+2]-7, 8p-7)	7	(233,217)

header and scrambled HCS by adding parity bytes.

The RS codes have been widely used in a variety of communication systems. The very high-speed data transmission techniques that have been developed for the mmWAVE~WPAN systems have necessitated the high-speed FEC architectures to meet the continuing demands for ever higher data rates[3~6].

The RS decoder architecture consists of three blocks, which are syndrome computation block, key equation solver (KES) block and error correction block that consists of Chien search and Forney algorithm blocks[6~11].

In this paper, we present the variable-length eight-parallel RS encoder/decoder architectures, which are designed to correct all of shortened RS codes with error correction capability t=8, for 19-*Gbps* mmWAVE WPAN systems. We will describe the key ideas applied to variable-length eight-parallel RS decoder design, especially those for achieving low-latency and high throughput. Moreover, the proposed RS encoder/decoder can support not only RS(255,239) code but also various shortened RS codes. Table 1 shows the various shortened RS codes with error correction capability t=8 specified in ECMA-387 standard[2]. The proposed architecture can manipulate different shortened RS code size to provide variable latency. Compared with fixed-length RS(255,239) decoder with fixed latency, the proposed variable-length RS decoder can provide the reduced latency.

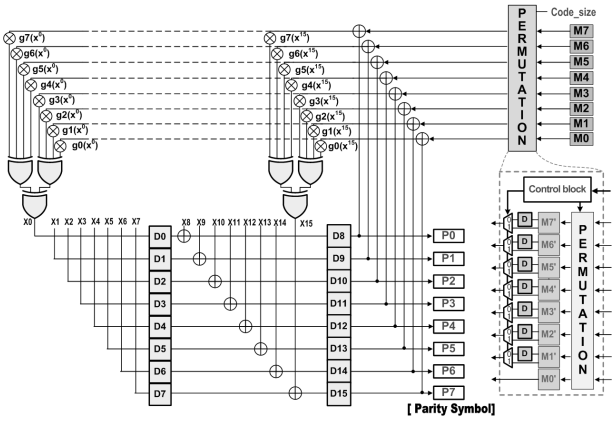


그림 1. 가변길이 8-병렬 RS 부호기

Fig. 1. Variable-length eight-parallel RS encoder.

This paper is organized as follows. Section II shows the proposed variable-length eight-parallel RS encoder architecture. Section III describes the proposed variable-length eight-parallel RS decoder architecture. A latency and timing issues are described in Section IV. Section V describes implementation results and performance comparison. Finally, conclusions are provided in Section VI.

Ⅱ. Variable-Length Eight-Parallel Reed-Solomon Encoder

The systematic RS encoding can be described as equation (1), where $U(x)$ is encoded polynomial. The message polynomial $M(x)$ is multiplied by X^{n-k} and then added parity polynomial $P(x)$. The following $P(x)$ can be written as equation (2). To apply eight-parallel structure, the equation (2) is needed to reformulate. The $M(x)$ consists of variable-length symbol, which is multiple of eight. As a result, eight-parallel based $P(x)$ can be described equation (3) and we can derive the following partial generator polynomial as shown in equation (4).

$$U(x) = x^{n-k}m(x) + p(x) \tag{1}$$

$$p(x) = x^{n-k}m(x) \bmod g(x) \tag{2}$$

$$P(x) = [[\cdots[m_{8p-1}x^{23} + m_{8p-2}x^{22} + m_{8p-3}x^{21} + m_{8p-4}x^{20} + m_{8p-5}x^{19} + m_{8p-6}x^{18} + m_{8p-7}x^{17} + m_{8p-8}x^{16}] \bmod g(x) \cdot x^8 +$$

$$\vdots$$

$$[m_7x^{23}+m_6x^{22}+m_5x^{21}+m_4x^{20}+ m_3x^{19}+m_2x^{18}+m_1x^{17}+m_0x^{16}]\bmod g(x) \quad (3)$$

$$\begin{aligned} g_0 &= x^{16}\bmod g(x) \quad g_1 = x^{17}\bmod g(x) \\ g_2 &= x^{18}\bmod g(x) \quad g_3 = x^{19}\bmod g(x) \\ g_4 &= x^{20}\bmod g(x) \quad g_5 = x^{21}\bmod g(x) \\ g_6 &= x^{22}\bmod g(x) \quad g_7 = x^{23}\bmod g(x) \end{aligned} \quad (4)$$

The proposed variable-length eight-parallel RS encoder architecture is shown in Fig. 1. Eight-parallel message symbols [M0, M1, M2, M3, M4, M5, M6, M7] are inputted to permutation block during the specific clock cycles decided by 'code_size' input, where 'code_size' means the codeword symbol size except parity symbol. The "code size" represents the data length information of the message symbols, which is generated from the previous block.

If the count of message symbols is not the multiple of eight, the message symbol must be permutated by adding the zero-padding. Eight types of message symbols should be generated for various shortened RS codes, as shown in Table 1. For example, the encoding of RS(237,221) code is

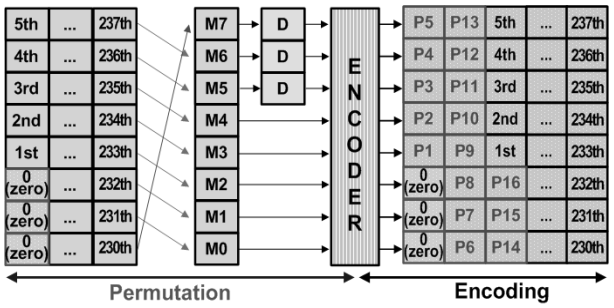


그림 2. RS(237,221) 부호의 코드 순환과정 예제
Fig. 2. Permutation process example for RS(237,221) code.

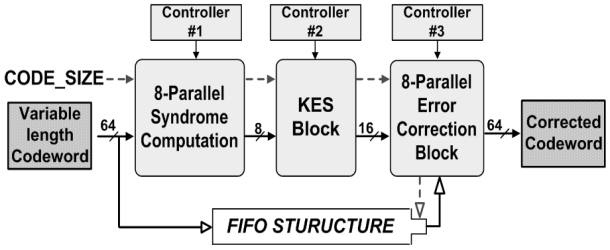


그림 3. 가변길이 8-병렬 RS 복호기
Fig. 3. Variable-length eight-parallel RS decoder.

represented by equation (5), and Fig. 2 illustrates the permutation process of RS(237,221) code.

$$\begin{aligned} P(x) = [[\bullet\bullet\bullet[0\bullet x^{23}+0\bullet x^{22}+0\bullet x^{21}+m_{236}x^{20}+ \\ m_{235}x^{19}+m_{234}x^{18}+m_{233}x^{17}+m_{232}x^{16}]\bmod g(x)\bullet x^8+ \\ \vdots \\ [m_7x^{23}+m_6x^{22}+m_5x^{21}+m_4x^{20}+ \\ m_3x^{19}+m_2x^{18}+m_1x^{17}+m_0x^{16}]\bmod g(x) \end{aligned} \quad (5)$$

Eight message symbols that are permutated by permutation module are fed into each partial generator polynomial and then the results of each partial generator polynomial are stored in register D1,…,D16. After all message symbols are received in RS encoder, the parity symbols are outputted.

Ⅲ. Variable-Length Eight-Parallel Reed-Solomon Decoder

The proposed variable-length RS decoder architecture consists of syndrome computation block, KES block, and error correction block as shown in Fig. 3. The syndrome computation block and error correction block are reformulated for the eight-parallel processing. To support the variable-length shortened RS codes, the syndrome computation block includes permutation module, and error correction block includes ROMs which store the first roots. This root value will be used to generate the error locator polynomial and error value polynomial. The root values outputted from the ROMs are increased by multiple of eight through constant multiplier and again will be used for the error locator polynomial and error value polynomial. In case of RS(255,239) code, the first roots are $[\alpha^1,\cdots,\alpha^8]$ and then the roots are increased up to $[\alpha^{248},\cdots,\alpha^{255}]$. But in case of RS(240,224) code, the first roots are $[\alpha^{16},\cdots,\alpha^{23}]$ and the roots are increased up to $[\alpha^{248},\cdots,\alpha^{255}]$. The first roots depend on codeword size. Therefore, the first roots of variable-length RS code which are controlled by 'code_size' are inputted to the

ROMs. The details of each sub-block are described as follows.

1. Variable-Length Eight-Parallel Syndrome Computation Block

The syndrome computation block calculates all the syndromes S_i $(0 \le i \le 15)$ by putting the roots of generator polynomial $G(x)$ into the received codeword polynomial $R(x)$. The equation (6),(7) are reformulated for eight parallel processing. As shown in Fig. 4, the proposed variable-length eight-parallel syndrome computation block is implemented by following equation (8). Eight-parallel codeword symbols [A, B, C, D, E, F, G, H] are inputted during the specific clock that is decided by 'code_size' signal.

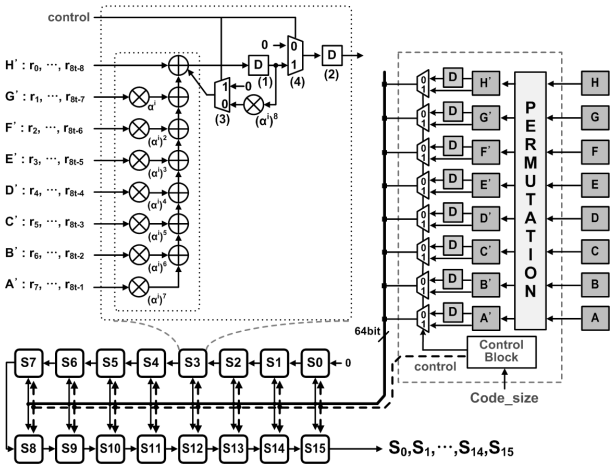


그림 4. 신드롬 계산 블록
Fig. 4. Syndrome computation block.

The codeword size is a variable length. If the count of codeword symbol is not multiple of eight, the codeword symbol should be permutated by adding zero-symbol. The permutation is decided by 'code_size' signal and the zero is added as shown in Table 1. The permutation process is equal to the case of RS encoder.

$$R(x) = r_{239}x^{239} + r_{238}x^{238} + r_{237}x^{237} + \cdots + r_1 x + r_0 \tag{6}$$

$$S_i = R(\alpha^i) = (r^{239}(\alpha^i) + r^{238})(\alpha^i) + \cdots)(\alpha^i) + r_0 \tag{7}$$

$$\begin{aligned} S_i &= R(\alpha^i) \\ &= [[\cdots[r_{8p-1}(\alpha^i)^7 + r_{8p-2}(\alpha^i)^6 + r_{8p-3}(\alpha^i)^5 + r_{8p-4}(\alpha^i)^4 + \\ &\quad r_{8p-5}(\alpha^i)^3 + r_{8p-6}(\alpha^i)^2 + r_{8p-7}(\alpha^i)^1 + r_{8p-8}](\alpha^i)^8 + \\ &\quad \vdots \\ &\quad [r_7(\alpha^i)^7 + r_6(\alpha^i)^6 + r_5(\alpha^i)^5 + r_4(\alpha^i)^4 + \\ &\quad r_3(\alpha^i)^3 + r_2(\alpha^i)^2 + r_1(\alpha^i)^1 + r_0] \end{aligned} \tag{8}$$

$$S_i = R(\alpha^i) = e(\alpha^i) = \sum_{l=1}^{v} e_{m_l} \alpha^{m_l \cdot i} \tag{9}$$

The proposed syndrome computation block can process the syndromes during a specific clock cycles provided by codeword size information. At the first clock, the received codeword (r_{8p-1} ,r_{8p-2}, r_{8p-3} ,r_{8p-4}, r_{8p-5}, r_{8p-6}, r_{8p-7}, r_{8p-8}) are inputted in parallel. If the 'code_size' signal is not multiple of eight, the codeword symbols are permutated by permutation module as shown in Fig. 4. The codeword symbols that are passed through the permutation module compute the following partial syndromes $r_{8p-1}(\alpha^i)^7 + r_{8p-2}(\alpha^i)^6 + \cdots + r_{8p-7}(\alpha^i)^1 + r_{8p-8}$ stored in the flip-flop (1). At the next, the flip-flop (1) is multiplied by $(\alpha^i)^8$ and then added with $r_{8p-9}(\alpha^i)^7 + r_{8p-10}(\alpha^i)^6 + \cdots + r_{8p-15}(\alpha^i)^1 + r_{8p-16}$. This iterative process will be performed during the specific clock cycles decided by 'code_size'. Multiplexer (3) and (4) are selected '1' at a specific clock cycle to output the syndrome polynomial. As a result, the syndrome S_0, S_1, ... , S_{15} are outputted serially for the KES block, so that new syndromes can be computed in the flip-flop (1).

2. Key Equation Solver Block

Most conventional high-speed RS decoders have adopted ME algorithm[3, 6~7] for KES block, because an ME algorithm can be easily implemented by a fully pipelined systolic-array architecture. The pipelined degree-computation less modified Euclidean (pDCME) algorithm and architecture has been used to obtain the error locator polynomial $\sigma(x)$ and the error value polynomial $\omega(x)$ by solving the key equation $\omega(x)=S(x)\sigma(x) \bmod x^{2t}$. The detailed

explanation of pDCME algorithm and architecture was addressed in our previous paper[8]. The three-stage pipelined KES block provided the reduced latency, so that $\sigma(x)$ and $\omega(x)$ value can be outputted after 48 clock cycles.

3. Variable-Length Eight-Parallel Chien Search and Error Correction Block

After the KES block, the error locator polynomial $\sigma(x)$ and the error value polynomial $\omega(x)$ are fed into the Chien search block, which calculates the roots of the error locator polynomial. The Forney algorithm lock works in parallel with the Chien search block to calculate the magnitude of the error symbol at each error location. The decoder can find the error locations by checking whether $\sigma(\alpha^{-j})=0$ for each j, $0 \le j \le n-1$. 'α' ($\alpha^{255}=0$)' means that r^0 is corrupted by error. In equation(13), $\sigma'(x)$ is the derivative of $\sigma(x)$. Rewriting $\sigma(x)$ as the sum of the even terms $\sigma_{even}(x)$ and the odd terms $\sigma_{odd}(x)$, we have $\sigma_{odd}(x) = x \cdot \sigma'(x)$. Therefore, the Chien search and error correction block is implemented as shown in Fig. 5.

$$\sigma(x) = \prod_{j \neq 1}(1 - X_j x) \tag{10}$$

$$\sigma'(z) = \lambda_1 + \lambda_3 x^2 + \cdots + \lambda_v x^{v-1} \tag{11}$$

$$\omega(X_l^{-1}) = Y_l \prod_{j \neq i}(1 - X_j x) \tag{12}$$

$$Y_i = -\frac{\omega(X_i^{-1})}{\sigma'(X_i^{-1}) X_i^{-1}} \tag{13}$$

In equation (13), dividing operation is implemented by 256×8 ROM in which the inverse of field elements are stored. As shown in Fig. 5(a), serial Chien search cell[6] was expanded into eight-parallel Chien search cell, because the following Chien search and Forney algorithm block should calculate eight locations of error at each clock cycle. The first roots of $\sigma(x)$ are decided by'code_size', because the variable length RS codes are shortened RS codes that are based on the RS(255,239). For example, if received codeword is RS(240,224), the first roots that $\sigma(\alpha^{16})$, $\sigma(\alpha^{17})$, $\sigma(\alpha^{18})$, $\sigma(\alpha^{19})$, $\sigma(\alpha^{20})$, $\sigma(\alpha^{21})$, $\sigma(\alpha^{22})$, $\sigma(\alpha^{23})$ are inserted in $\sigma(x)$. At the last, $\sigma(\alpha^{248})$, $\sigma(\alpha^{249})$, $\sigma(\alpha^{250})$, $\sigma(\alpha^{251})$, $\sigma(\alpha^{252})$, $\sigma(\alpha^{253})$, $\sigma(\alpha^{254})$, $\sigma(\alpha^{255})$ are calculated consecutively.

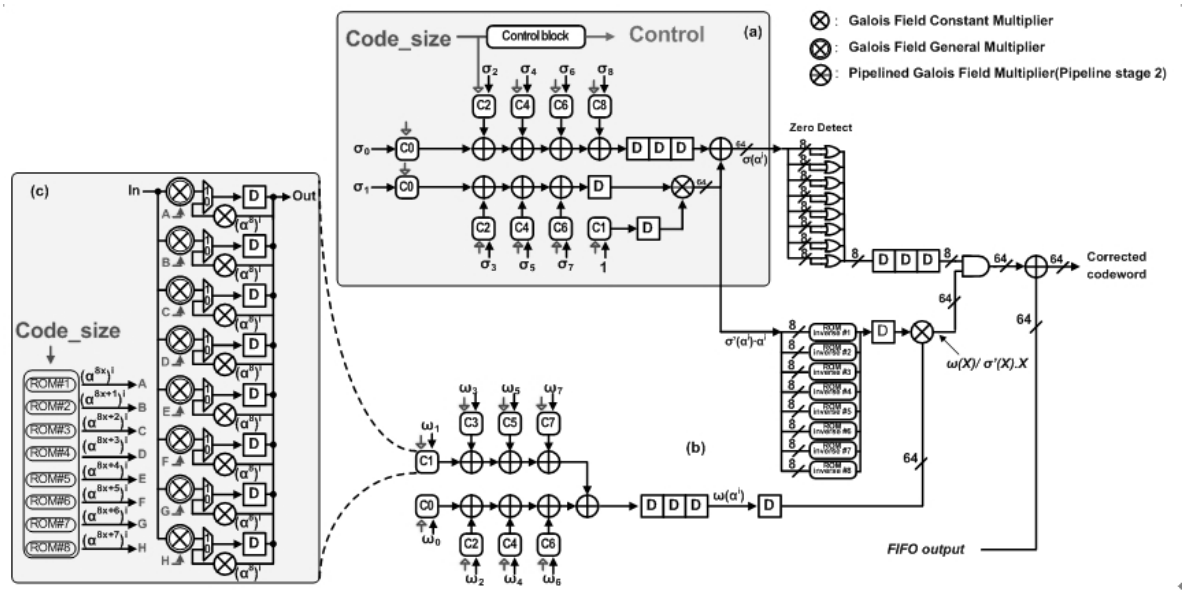


그림 5. (a) Chien-Search 블록, (b) Forney-algorithm 블록, (c) 에러 정정 기본 셀의 블록도
Fig. 5. Block diagram of (a) Chien search block, (b) Forney algorithm block, and (c) error correction cell.

4. FIFO Memory and Control Block

As each error value is calculated, 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 a corrected symbol. At the locations where no errors have occurred, the error values are zero and there is no change in the received polynomial at those locations when added.

The memory size of FIFO is 4,928 bits (64×77 bit). The latency of various shortened RS codes in FIFO must be changed by the received codeword size. As shown in Fig. 6, the received data that are stored in memory from 47 to 77 address are connected to MUX and then the output of MUX is connected to control block. The 'code_size' signal controls MUX and control block. The output of FIFO is corrected by the output of error correction block.

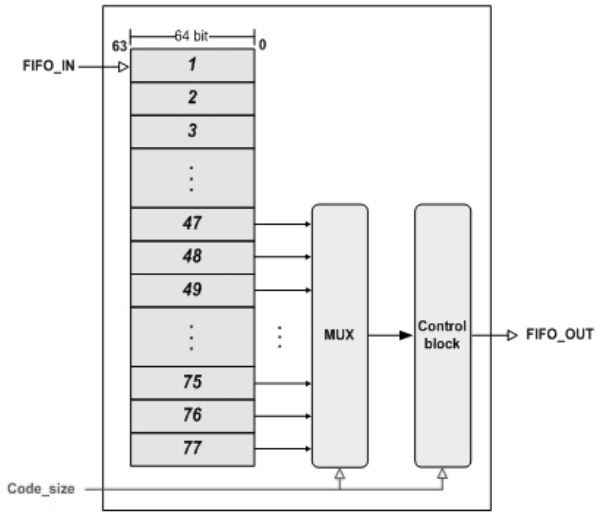


그림 6. FIFO 메모리 구조 및 제어 블록
Fig. 6. FIFO memory and control block.

Ⅳ. Timing and Latency

The decoding processing time and latency are changed by 'code_size'. The fixed-length RS decoder with the fixed-length codeword size receives the codewords continuously, but the variable-length RS

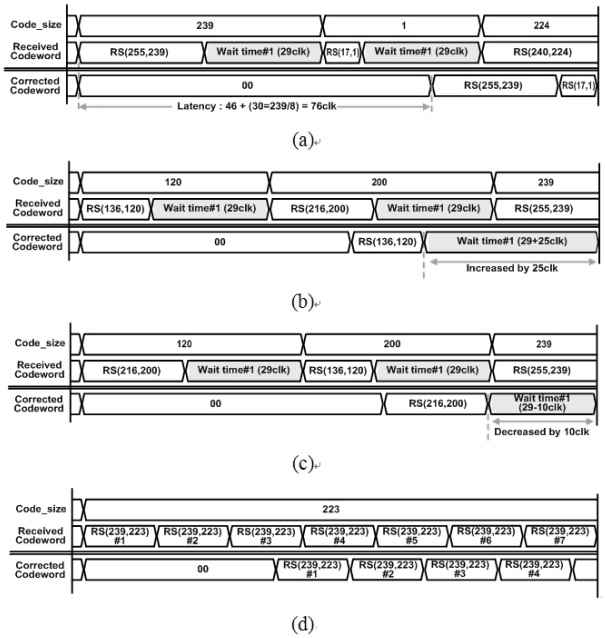


그림 7. 가변길이 8-병렬 RS 부호기의 데이터 흐름도
Fig. 7. Data flow of variable-length eight-parallel RS decoder.

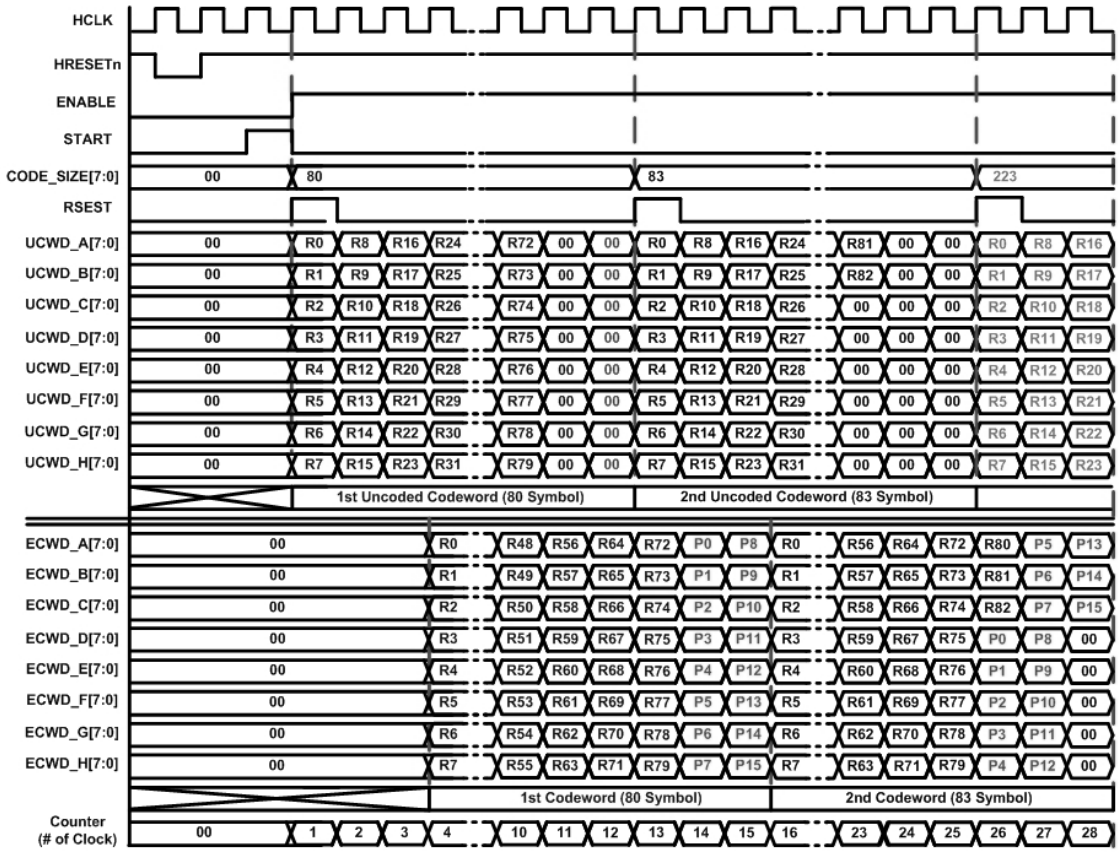
decoder with variable codeword size needs the wait-time to process the various RS codes. That is, the wait-time is needed during the decoding process. The wait-time is decided by the longest and the shortest code size as shown in equation (14).

$$\text{Wait-time (clock)} = [\text{the longest code\_size}/8] - [\text{the shortest code\_size}/8] \quad (14)$$

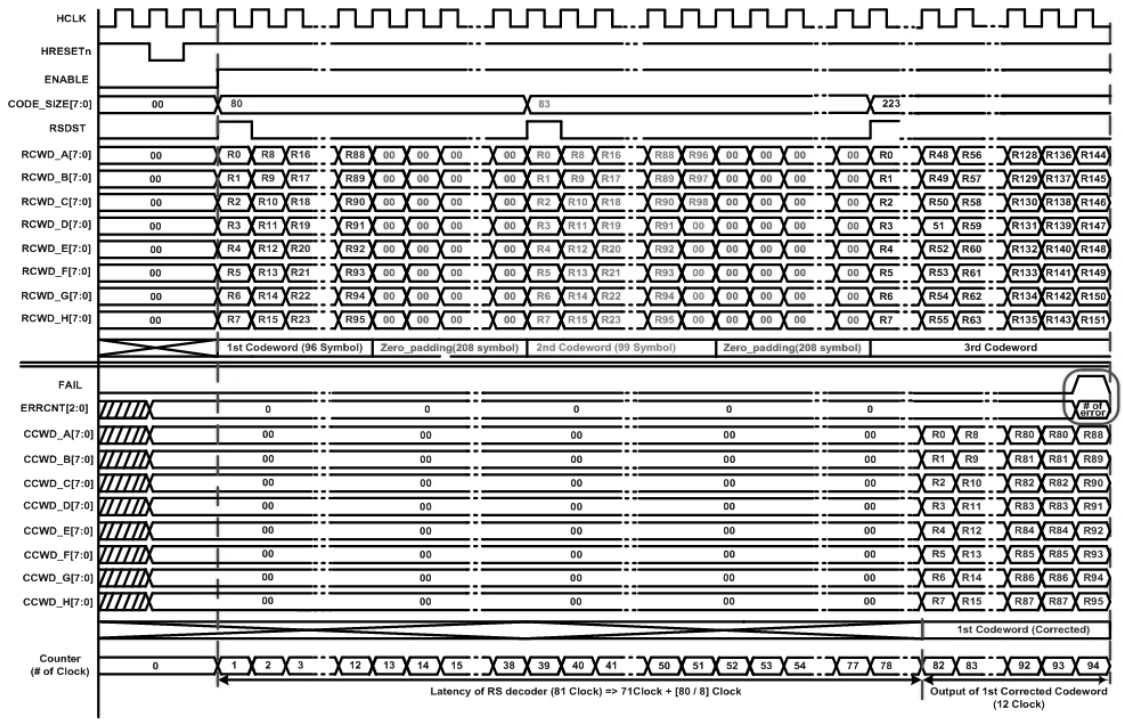
In our design, the longest code size is RS(255,239) code and the shortest code size is RS(17,1) code. Therefore, the wait-time of the proposed architecture is 29 clock cycles. Fig. 7 shows the data flow of proposed RS decoder. As shown in Fig. 7(a), RS(17,1) code must be inputted after wait-time 29 clock cycles, and two corrected codeword are outputted continuously without delay between each decoding. As shown in Fig. 7(b), if RS decoder receives shorter codeword than the previous codeword, the wait-time is increased by

$$[\text{the previous code\_size}/8] - [\text{the current code\_size}/8]] \quad (15)$$

As shown in Fig. 7(c), if RS decoder receives



(a)



(b)

그림 8. (a) 가변길이 8-병렬 RS 부호기 및 (b) 가변길이 8-병렬 RS 복호기의 타이밍도

Fig. 8. Timing chart of (a) variable-length eight-parallel RS encoder and (b) variable-length eight-parallel RS decoder.

longer codeword than the previous codeword, the wait-time is decreased by

[the current code_size/8]
- [the previous code_size/8] (16)

As shown in Fig. 7(d), if RS decoder receives the codewords with same size, it does not need the wait-time. Thus, the latency of received codeword can be estimated by the 'code_size'. The latency of variable-length RS decoder is the sum of the decoding operation time and [code_size/8] The decoding operation time of variable RS code is 71 clocks.

For example, the latency of RS(255,239) code is 71 clocks + [239/8] clocks = 101 clocks. The latency of RS(17,1) code is 71 clocks + [1/8]clocks = 72 clocks.

The timing charts of proposed RS encoder and decoder are shown in Fig. 8(a) and (b), respectively.

The proposed RS encoder has three clock latency, and generates encoded codeword (ECWD_A~ ECWD_H). The encoder start signal (RSEST) is needed during 1 clock and after then the effective codeword symbols are entered (UCWD_A~ UCWD_H). As shown in Fig. 8(b), the RS decoder accepts the received codeword (RCWD_A~ RCWD_H) at the same time. The latency of RS outputted. decoder depends on 'code_size' of received codeword. When the proposed RS decoder outputs the corrected codeword (CCWD_A~CCWD_H), error count signal (ERRCNT) for the first codeword is outputted with the last codeword at the same time. If more than 9 errors were occurred the FAIL signal is outputted.

표 2. 가변길이 8-병렬 RS 부호기와 일반 8-병렬 부호기의 구현결과 비교
Table 2. Implementation results of the variable-length eight-parallel RS encoder and fixed-length eight-parallel RS encoder.

Design	Proposed variable-length RS(255,239)	Fixed-length RS(255,239)
Total # of Gates	9,250	5,810
Clock Rate (*MHz*)	300	400
Latency (*clocks*)	3	1
Throughput (*Gbps*)	19.2	25.6
Technology	90-*nm* CMOS 1.2 *V*	90-*nm* CMOS 1.2 *V*

표 3. 가변길이 8-병렬 RS 복호기와 일반 8-병렬 복호기의 구현결과 비교
Table 3. Implementation results of the variable-length eight-parallel RS decoder and fixed-length eight-parallel RS decoder.

Design	Proposed Variable-length RS(255,239)	Fixed-length RS(255,239)
Syndrome	18,070	6,590
KES (pDCME)	32,000	31,870
Chien, Error	179,160	38,100
Total # of Gates	229,230	76,560
Clock Rate (*MHz*)	300	400
Latency (*clocks*)	73~102	101
Throughput (*Gbps*)	19.2	25.6
Technology	90-*nm* CMOS 1.2 *V*	90-*nm* CMOS 1.2 *V*

V. Results and Comparison

The proposed variable-length eight-parallel RS encoder and decoder architecture was modeled in Verilog HDL and simulated to verify its functionality.

After complete verification of the design functionality, it was then synthesized using appropriate time and area constraints. Both simulation and synthesis steps were carried out using SYNOPSYS synthesis tool and 90-*nm* CMOS technology optimized for a 1.2 *V* supply voltage. For the purpose of comparison, the fixed-length eight-parallel RS encoder/decoder was also modeled in Verilog HDL and synthesized using the same 90-*nm* CMOS technology. The gate count of variable-length eight-parallel RS encoder and decoder is 9,250 and 229,230 gates, respectively, excluding FIFO memory. From the pre-layout simulation, the

proposed variable-length eight-parallel RS decoder architecture can operate at clock frequency of 300 *MHz* and has data processing rate of 19-*Gbps*. Table 2 and 3 compare the performance of the proposed variable-length RS encoder/decoder architectures. The table shows that the proposed RS encoder/decoder operates at a clock rate of 300 *MHz* and a throughput of 19.2-*Gbps*. Although the hardware complexity of the proposed RS decoder is approximately three time larger than that of fixed-length RS decoder, the proposed RS decoder has lower latency than that of fixed-length RS decoder. Thus, the proposed RS decoder is very useful for high-rate WPAN systems requiring low latency.

VI. Conclusions

This paper presents the design and implementation of variable-length eight-parallel 19-*Gbps* RS encoder and decoder for high-rate WPAN systems. Eight-parallel processing is used to achieve high data throughput. The permutation method enables variable-length RS decoder to have low latency for shortened RS code. Using 90-*nm* CMOS standard cell technology, the proposed RS decoder can provide a maximum 19-*Gbps* data rate at clock frequency 300 *MHz*. Therefore, it can be applied in the FEC devices for next-generation high-rate WPAN systems with a data rate of 10-*Gbps* and beyond.

References

[1] "Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for High Rate Wireless Personal Area Networks (WPANs): Amendment 2: Millimeter-wave based Alternative Physical layer Extension," IEEE P802.15.3c/D00. 2008.

[2] "High Rate 60GHz PHY, MAC and HDMI PAL" Standard ECMA-387, 1st Edition, Dec.2008. vol. 37, no. 11, pp. 1565-1573, Nov. 2002.

[4] S. B. Wicker, "Error Control Systems for Digital Communication and Storage," Prentice Hall,1995.

[5] D. V. Sarwate and N. R. Shanbhag, "High-speed Architecture for Reed-Solomon decoders" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 9, no. 5, pp. 641-655, Oct. 2001.

[6] H. Lee, "High-Speed VLSI Architecture for Parallel Reed-Solomon Decoder," IEEE Trans. on VLSI Systems, vol. 11, no.2 pp. 288-294, April 2003.

[7] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen and I. S. Reed, "A VLSI Design of a Pipeline Reed-Solomon Decoder," IEEE Trans. on Computers, vol. C-34, no.5, pp. 393-403, May 1985.

[8] S. Lee, H. Lee, J-Y. Shin, and J-S. Ko, "A High-Speed Pipelined Degree-Computationless Modified Euclidean Algorithm Architecture for Reed-Solomon Decoders," 2007 IEEE International Symposium on Circuits and Systems (ISCAS2007), pp. 901-904, May 2007.

[9] S. Lee, C-S. Choi and H. Lee, "Two-parallel Reed-Solomon based FEC Architecture for Optical Communications," IEICE Electronics Express, vol. 5, no.10, pp.374-380, May 2008.

[10] C-S. Choi and H. Lee, "High-Speed Low-Complexity Three-Parallel Reed-Solomon Decoder for 6-Gbps mmWAVE WPAN Systems," European Conference on Circuit theory and Design 2009 (ECCTD'09), pp. 515-518, Aug. 2009.

[11] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multilevel cell NAND flash memories," in Proc. IEEE Workshop Signal Process. Syst. (SiPS): Des. Imple., pp. 248-253, Oct. 2006.

저 자 소 개

최 창 석(학생회원)

2005년 한신대학교 정보통신공학 학사 졸업
2007년 인하대학교 정보통신공학 석사 졸업
2007년～현재 인하대학교 정보공학과 박사 재학
<주관심분야 : 통신용 VLSI 및 SoC설계>

이 한 호(정회원)

1993년 충북대학교 전자공학과 학사 졸업
1996년 Univ. of Minnesota 전기컴퓨터공학 석사 졸업
2000년 Univ. of Minnesota 전기컴퓨터공학 박사 졸업
2000년～2002년 Member of Technical Staff, Lucent Technologies (Bell Labs.), USA.
2002년～2004년 Assistant Prof. Dept. of Electrical and Computer Engineering, Univ. of Connecticut, USA.
2004년～현재 인하대학교 정보통신공학부 교수
<주관심분야 : 통신용 VLSI 및 SoC설계>