國立交通大學

電機資訊學院 電子與光電學程

碩士論文

應用於低成本及高速化的多種類記憶體晶片系統之
新型可程式化控制(n，k，m)編碼參數的
錯誤更正編解碼器

A New Programmable Control (n, k, m) ECC Encoder-Decoder for Low-cost, High-speed Various Memory-Chips System Applications

研究生：陳彛梓

指導教授：李鎮宜 教授

中華民國九十四年六月

應用於低成本及高速化的多種類記憶體晶片系統之新型可程式化控制(n，k，m)編碼參數的錯誤更正編解碼器

A New Programmable Control (n, k, m) ECC Encoder-Decoder for Low-cost, High-speed Various Memory-Chips System Applications

研究生：陳彝梓　　Student：Yi-Tzu Chen

指導教授：李鎮宜　　Advisor：Chen-Yi Lee

國立交通大學

電機資訊學院 電子與光電學程

碩士論文

A Thesis

Submitted to Degree Program of Electrical Engineering and Computer Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electronics and Electro-Optical Engineering

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

應用於低成本及高速化的多種類記憶體晶片系統之
新型可程式化控制(n, k, m)編碼參數的錯誤更正編解碼器

學生：陳彝梓　　　　　　　　　　　　　　　指導教授：李鎮宜 教授

國立交通大學電機資訊學院 電子與光電學程（研究所）碩士班

摘　　要

本篇論文使用新的錯誤更正碼的建構方式去實現了一個新型可程式化控制的錯誤更正編解碼器之架構，這個被提議的錯誤更正碼具有「單一位元的錯誤更正，單一位元組內有奇數位元的錯誤更正及單一位元組的錯誤偵測，兩個位元的錯誤偵測」之錯誤更正及偵測的能力，並命名為SEC-S$_{odd}$EC-SBED-DED codes。 此外一個關鍵的重點是它很適合用於可程式化(n, k, m)編碼參數之控制，此處的n表示整個ECC編碼長度、k表示被編碼的資料長度、m表示被編碼的資料的寬度，因此這個被提議的SEC-S$_{odd}$EC-SBED-DED codes具有非常彈性化的資料編碼長度及寬度，可進行任何的(n, k)系統體系上區塊編碼。

本篇論文主要的目的是利用這個被提議的錯誤更正碼，去完成一個具有高速化及低複雜度的可程式化之順向錯誤更正編碼與解碼電路，能符合多種類記憶晶片系統應用上所需的高性能、低成本及適當的可靠度之需求。 另外地，我們也提出了交錯式SEC-S$_{odd}$EC-SBED-DED codes方法，可使得順向錯誤更正編解碼器具有多個位元組的錯誤更正及偵測的能力，大幅提高了整體錯誤控制系統的可靠度，進而可朝向更廣泛與多樣化的串列資料傳輸上的錯誤更正編碼之應用。

A New Programmable Control (n, k, m) ECC Encoder-Decoder for Low-cost, High-speed Various Memory-Chips System Applications

Student：Yi-Tzu Chen　　　　　　　　Advisors：Prof. Chen-Yi Lee

Degree Program of Electrical Engineering Computer Science
National Chiao Tung University

ABSTRACT

This paper utilizes new error-correcting-codes constructing approaches to present a new programmable control (n, k, m) error-correcting encoder-decoder architecture. The proposed ECC is named SEC-S_{odd}EC-SBED-DED codes to have these capabilities of random Single bit Error Correction－Single odd-bit Error Correction within a single byte－Single Byte Error Detection－random Double bits Error Detection. An important key point is that the proposed error-correcting code/circuit (ECC) is very well to these programmable or variable (n, k, m) parameters, where n=an ECC codeword length, k=an encoded information length, m= data-I/O wide. In other words, the proposed SEC-S_{odd}EC-SBED- DED code has a very flexible code-length and code-width to any type of a (n, k) systematic block-code without restriction.

Main purpose of the thesis is to show that the proposed error-correcting codes can finish a high-speed, low-complexity, programmable forward ECC encoding and decoding circuits to meet the high-performance, low-cost and moderate reliability demands for various memory-chips system applications. In addition, we propose also interleaving SEC-S_{odd}EC-SBED-DED codes for the FEC-codec system which reaches to multiple bytes error correcting-detecting accomplishment. Hence the reliability of whole error control system is enhanced in order to drive toward the wide varieties of serial error control coding systems applications.

Contents

List of Tables

List of Figures

Symbol Descriptions

Symbol	***Meaning***
n	Code-length, or the length of an ECC codeword
k	information/data/message length
M, m	Data-I/O wide, or a byte/symbol in bits
r	parity check bit length, or redundant-bit length
N	the number of an ECC codeword bit, where $N = n \times m$
K	the number of Information/data/message bit, where $K = k \times m$
R	the number of redundant-bit/parity check bit, where $R = r \times m$
$\lfloor x \rfloor$	largest integer less than or equal to x
$\lceil x \rceil$	the smallest integer greater than or equal to x
+ or $\oplus$	an XOR logic-operation

Keywords and Proper Nouns Explanations

Keywords	***Meaning***
ECC, FEC	Error-Correcting-Code, Forward Error-Correcting-Code
MLC memory	Multi-Level-Cell memory indicate that a 2^q level cell has q bits storage unit
BLC memory	Binary-Level-Cell memory indicate that a two level cell only has 1 bits storage unit
MCP	(Stacked) Multi-Chip-Package
MCM	Multi-Chip-Module package
PSRAM	Pseudo Static RAM is equivalent to 1T-SRAM
SDRAM	Synchronous Dynamic RAM
FCRAM	(Double/Single Data Rate) Fast Cycle RAM
DDR/SDR SDRAM	Double/Single Data Rate Synchronous Dynamic RAM
SEC-DED	Single-Error-Correction—Double-Error-Detection
SBED	Single-Byte-Error-Detection
DEC-TED	Double-Error-Correction—Triple-Error-Detection
TEC-QED	Triple-Error-Correction—Quadruple-Error-Detection
RS-code	Reed-Solomon Code
BCH-code	Bose-Chaudhuri-Hocquenghem Codes
SbEC-DbED	Single b-bit byte Error Correction—Double b-bit byte Error Detection
SEC-S_{odd}EC	Single-bit Error-Correction—Single odd-bit Error Correction
FEC-Codec	Forward Error Correcting Encoder-Decoder
GF(q)	Galois-Field, where q is the order of Galois-Field
BER	Bit Error Rate
FIT	Failure In Time is equivalent to 1-failure per billion device-hours

Chapter 1
Introduction

1.1 Motivations

Recently, more and more kinds of memory chips are applied to portable devices, such as digital cameras, digital audio and video player, mobile phone, and personal multi-media assistants, etc. In general, many varieties of high capacity memory chips are integrated into a memory card, a multi-chips module package (MCM), a product in package (PIP), or stacked multi-chip package (stacked MCP) for reducing chip-to-chip dimension toward the portable device applications. These varieties of integrated memory-chips modules are almost based on dram and flash memory to organize a specific combo-memory module. Furthermore, more and more the large-capacity memory chips using a deep submicron fabrication process and multi-level cell (MLC, i.e. multiple-bit in a cell unit) device technology is to induce more critical reliability issues, such as disturbs and data retention, and radiation induced soft errors [3]. So the memory control system must guarantee the reliability problems of these varieties of combo memory-modules during the operational life time of memory-chips. Therefore it can be said that memory control system need to adopt some form of error detecting and correcting codes in order to enhance the reliability of memory-chips. In most applications to semiconductor memory systems, errors that occur in semiconductor memory systems can be regarded as either random errors or byte error. The speed of the operations for semiconductor memory systems is very high, and therefore the error-correcting codes (ECC) decoders must be extremely fast. Also the number of redundant overhead cannot be too large [32].

Though many flash and dram memory-chips with on-chip ECC have been presented in the papers [1]-[9], [15]-[19], these ECC-memory chips must pay the access time penalty about 5~25ns and an additional ECC area overhead about 10%~35%. Thus memory-chips with on-chip ECC have high cost and poor performance in access-speed. For the both low-cost and high performance factors of the varieties of integrated memory-chips modules in serial page access operations to compare the figure 1.1 (a) with (b), and figure 1.1 (a), (b) show multiple memory-chips application that the system-level cost of a memory interface controller with ECC like figure (a) may be lower than that figure (b) with ECC-memory chips. In figure 1.1, we know a host memory-controller employ an error correcting code circuits that it will be more efficient to improve the whole memory system performance, reliability and minimize the cost for multiple memory-chips integration.

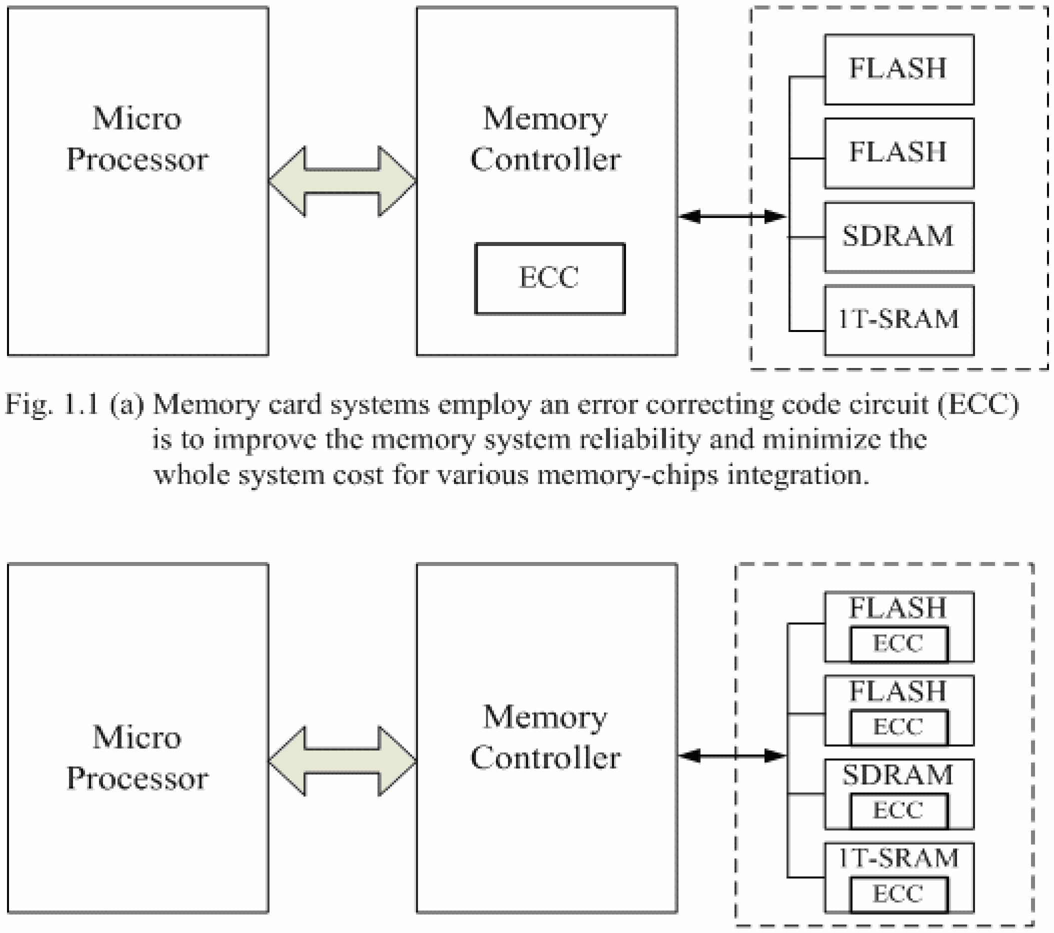


Fig. 1.1 (a) Memory card systems employ an error correcting code circuit (ECC) is to improve the memory system reliability and minimize the whole system cost for various memory-chips integration.

Fig. 1.1 (b) In various or multi-memory chips applications, the whole system cost of each memory with on-chip ECC will be higher than that with the memory-controller performing the error correction.

Fig. 1.1 Two different kinds of memory control systems with ECC.

Therefore memory interface controller built-in ECC will be a better choices as shown in figure 1.1 (a).

Now most of memory-chips have a high speed clock rate or page access-time, such as a 100~200Mhz clock-rate for mobile SDRAM, 200~400Mhz clock-rate for DDR FCRAM, SDRAM, 20ns~40ns page-access PSRAM, 50~100Mhz for page-mode flash. Besides, most of memory-chips have different page or burst-length such as 8bytes~512bytes burst-length for mobile-SDRAM/PSRAM, and specific 264bytes, 528bytes, 2112bytes page-length for NAND-flash, and different data I/O wide such as 1, 4, 8, 16, 18-bits….etc. So we need a simpler, faster, flexible and programmable error-correcting-coding technology to reach to system-user defined low-cost and high performance demands, and we must also think about the error-correcting-detecting capabilities and parity check-bits overhead.

By the above reasons, for the varieties of integrated memory-chips system, it is needed that the forward error-correcting encoder-decoder can provide a wide programmable coding length, and data-I/O width, minimum decoding latency, acceptable redundancy overhead and error-correcting-detecting ability, high throughput and high speed encoding-decoding operation with real time mapping-out process.

1.2 Outlines

In chapter 2, the basic concept of dram and flash reliability problems, some key features of error-correcting code for various memory-chips system are described, and the fundamentals of the existing error-correcting code circuits are reviewed briefly.

In chapter 3, the constructed methods and interleaved mechanisms of the proposed SEC-S_{odd}EC-SBED-DED code and Multi-Bit-Layer SEC-S_{odd}EC-SBED-DED interleaved-codes are clarified for fast and flexible programmable issues due to the different page size of the varieties of memory-chips.

In chapter 4, this section mainly describes hardware implementation for the programmable architecture and circuit design of the proposed FEC codec. Furthermore, the proposed ECC codes have been implemented in C-language software design for any (n, k, m) parameters. In addition, many performances comparisons with the existing ECC also are listed, such as throughput rate (maximum operating transfer rate), complexity (area overhead), decoded error rate...etc.

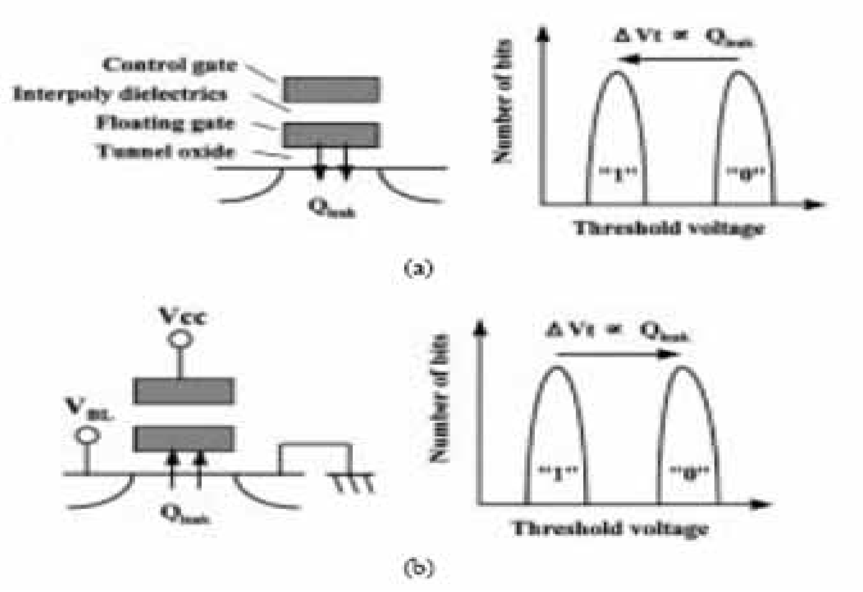
In chapter 5, the hardware and software simulation results are described, such as encoding-decoding waveform, read-write data flow of the proposed FEC-Codec, decoded error-rate...etc.

In chapter 6, a summary to our error correcting codec is given in this section.

Chapter 2
Basic Concepts for Memory Reliability Issues and the Existing ECC Codes

2.1 The DRAM and Flash memory reliability issues

Firstly, we introduce the common reliability problems to both dram and flash memory. The common reliability problems on flash memory have generally two types of errors [4]:
(1) After write and erase cycles, stored electrons can leak away from the floating gate through tunnel oxide during aging. The charge loss causes a decrease in the memory transistor threshold voltage, which may result in random 0 to1 errors.
(2) During read operation, the floating gate slowly gains electrons with the control gate held at Vcc. The charge gain causes an increase in the memory transistor threshold voltage, which may result in random 1 to 0 errors. The above (1) and (2) reliability problems are shown in Fig. 2.1.



Another reliability problems for dram memories mainly have also two types of errors [5]:
(1) One is called the memory cell error-upset that the associated cell or node capacitance in deep submicron process is scale-down, hence the capacitor is highly susceptible to being

discharge by noise electrons.

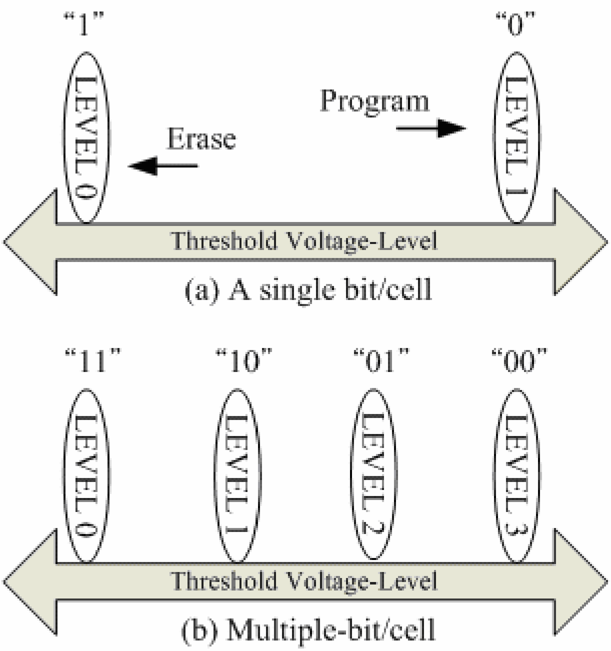
(2) Another is called bit-line error-upset that the sensing margin of sense amplifier is a very small signals, thus the bit-line differential voltage may degrade due to noise-couple, and hence the resulting read operation may be erroneous.

The foregoing flash and dram reliability will become a significant concern in deep sub-micron MLC (multi-level-cell, a 2^q level cell has q bits storage unit) technology. a bi-level single memory cell must distinguish between two voltage states, whereas a multiple-bit MLC-cell uses a voltage window with similar structure size, the distance between adjacent bit-to-bit threshold voltage levels in MLC is much smaller than traditional binary-level memory, which makes the reliability problems of MLC-memories more critical than conventional bi-level cell (BLC) memories [3], [8], as shown in Fig. 2.2 (a), (b).

The most of foregoing reliability issues are caused mainly by soft error due to alpha particles and soft errors are defined widely such as transient errors, power-supply noise spikes, thermal effects, and man-made states. These errors are called soft, because they do not damage the physical functions of a cell permanently, and they can easily corrected by complementing the data in the faulty cells [2], [5]. In a DRAM chip more than 98% of single-bit failures are radiation induced soft-errors [20]-[21], and In NAND-flash memory Over 99% of failures are attributed to single-bit soft errors [22]. Because dram storage unit is a trench or stacked capacitor and flash storage unit is by using floating-gate, which is a solid-state memory so the influence of the alpha particle induced soft error rate on dram memory is more significant than flash memory. About DRAM and flash memory reliability testing results are shown in the papers [23]-[27], we can know the average FIT (Failure in Time) and Bit-Error-Rate (BER) under different process, chip-size or different conditions. The soft error rate of different memory-chips is listed as follows [23]-[27], where 1-FIT = 1 failure per billion device-hours.

Type	BLC NOR-Flash	MLC NOR-Flash	BLC NAND-Flash	MLC NAND-Flash	BLC DRAM
# bits	16M/64M	64M	256M	256M	~512M
Process	0.23/0.17um	0.23um	0.16um	0.16um	0.25~0.13um
FITs/Mbit (Sea Level)	6E-9/3E-6	3E-7	1E-3 (read) 1E-4 (program)	1.0 (read) 1E-2 (program)	500~1000
FITs/Mbit (aircraft)	2E-6/1E-5	1E-4	-	-	-

As a consequence of these issues, the use of error correcting code techniques can help to reach adequate reliability of the deep sub-micron process, high-capacity, MLC-memories for immunity to soft-errors.



(a) A single bit/cell

(b) Multiple-bit/cell

Fig. 2.2 Threshold-voltage distribution of single-bit cell and multi-bit cell for flash memory.

2.2 A discussion on the existing ECC codes

Many ECC schemes have been widely proposed to enhance the reliability of dynamic-RAM, NAND-type flash and solid-state disk [1]-[9], [15]-[19]. In these [1]-[19] papers, it was understood that applying ECC to a memory control system requires a moderate balance between performance (access time penalty, operation frequency, throughput rate, encoding-decoding cycle count, error correcting ability, other features such as interleaving function, etc), chip-size overhead (circuitry complexity and parity check-bit overhead), and reliability enhancement (low decoded error rate or error probability, high detected error rate, soft-error-rate or yield improvement, reducing mean time to failure). Based on the above reasons, the proposed error-correcting code circuit must satisfy the following conditions for the most of various memory chips.

1) For a reliability issue of memory-chips in page-oriented memory-system application, because the memory chips usually can't have built-in error correcting code circuits

(ECC-circuits) due to the limitations of access time penalty and an additional area cost of ECC-circuits, i.e. the non-ECC commodity memory chips have NAND-type flash and specific mobile-DRAM. So the external memory control systems need a system/board-level ECC to ensure the validity of received data of the page/sector-oriented memories. In general, memory reliability depends on the both error correcting-detecting capability and the soft-error rate or failure rate of memory-chips.

2) For a high throughput data rate, the memory control systems need a high-speed FEC Codec hardware to minimize access latency and maximum operating clock speed. In order to demands of execute in place, the error-correcting code circuit can correct any error-bit of reading random address immediately after serial download program-code procedure from external memory as shown in Fig. 2.3. In other words, after the received n-bytes serial program-code data, the ECC circuit must be to look for the error-address and error-value instantly so that the CPU can execute the program-code right now for real-time application requirements. In general, a high-speed page access time is about 5ns~15ns based on DDR/SDR SDRAM memory, 10ns~70ns for NOR-Flash, 50ns for NAND-memory.

3) For low-cost considerations as Fig. 1.1. We need a compact, flexible FEC Codec to minimize the ECC Codec complexity, parity check-bits overhead, and furthermore programmable code-length feature that applied to the different page sizes demands of various memory-chips, i.e. a page or sector in a single memory-chip is organized as m-bits data width (m-bit is one byte length) and an n-bytes data-length, where a page or sector has the number of $n \times m$ bits. Programmable (n, k, m) parameters are necessary so that the users of memory-chips can define an arbitrary data length with ECC parity check-bits.

In general, memory-chip data-wide m is a multiple of 4, such as 4, 8, 16, 32 bits, but some special memories have a specific data-wide. A page length usually is a multiple of 8, such as 8, 16, 32, 64, 128 Bytes, and so on. Furthermore, a page size of NAND-flash is 528, or 2112-Bytes.

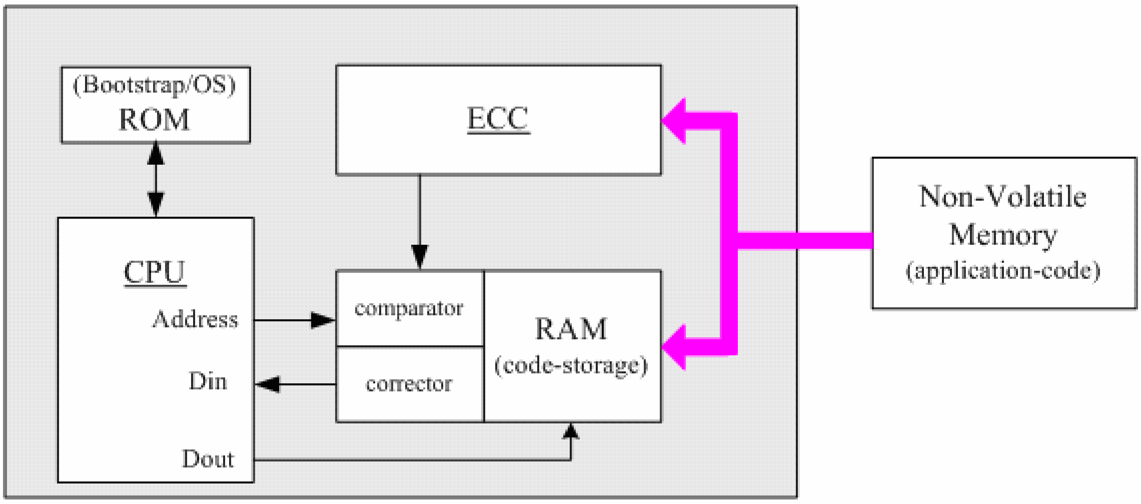


Fig. 2.3 Download application-code from external memory to execute application-program.

Among [1]-[19] literatures, we try to compare these error control code for finding the optimal coding style and to investigate the range of the page-sizes, data-width and an acceptable Bit-Error-Rate to correspond to the transition error probability of both DRAM and FLASH memories in practical conditions, so that our proposed FEC-Codec has low-cost, low-complexity and high-speed to provide a good performance and moderate reliability meet with the foregoing 1 to 3 ideas. Basically, the existing ECC generation methods have still some restrictions to the programmable coding length and width, and we propose ECC-generation methods that have almost no restrictions to coding length and width. Here we analyze the existing ECC codes in order to apply for the programmable (n, k, m), where n=code-length, k=data, message or information length, m=data I/O wide or a byte/symbol size in bit. We known parity check-bit length $r = (n-k)$, then the number of parity-check bits $R = (n-k)\times m$, the number of information-bits $K = k\times m$, and the total number of coding-bits are $N = K + R = n\times m$ (bits). N is user-defined memory block-size with both parity check-bits and information bits equal to a page memory-space, and k >> m in general memory-chips applications.

SEC/SEC-DED Hamming-codes or odd-weight column modified-Hamming-codes were presents in [4], [8], [15], [16] that they are suitable for on-chip, fixed code-length ECC design. It has a proper number of parity-bits $R = \lceil \log_2 k \rceil + 2$, and $R + K = N$ for SEC-DED and suits to serial data-bit coding by using a Hamming cyclic-code. It is hard and complex to design a variable code-length n and data-wide m in the modular decoding-circuit unless a multi-SEC-DED code using multiple decoding circuits can solve this problem.

However the cost overhead will be obviously increased, i.e. the larger parity check-bits is about $R = (\lceil \log_2 k \rceil + 2) \times m$.

Another traditional SEC-DED codes are bidirectional cross-parity/product codes that the type of code have been present in [1], [19] for on-chip ECC applications. Though it is suitable for programmable n and m parameters due to a simple encoding-decoding circuit, it also has a large number of parity check bits in proportion to k and m, such as $R = k + m + 1$.

Some DEC-TED codes and TEC-QED codes are presents in [2], [5] and [7] respectively. They have a good correcting capability, and programmable (n, k, m) circuits are feasible, but that's necessary to pay a largest number of parity check-bits, i.e. the TEC-QED research [7] was designed by combining odd-weight-column SEC-DED with vertical parity bit technique for a memory array, i.e. all word-lines of memory-array are along column direction, bit-lines of array are along row direction. Each column employs odd-weight-column SEC-DED codes, and each row employs a parity bits. So we get a parity check bits $R = m \times (\log_2 k + 2) + k$, or $R = k \times (\log_2 m + 2) + m$. The DEC-TED researches [2], [5] were designed using orthogonal Latin-square code which belong a majority-logic decoding code. For a square arrangement of the m^2 data array, it has also a large parity check-bit $R = 3m + 1$.

The Reed-Solomon code (RS-code), or BCH-code have a powerful multiple bytes error correcting and detecting capability, and a small number of check-bits for single/double byte correction, but have a complex decoding hardware and a longer decoding time. The RS-code defined in $GF(2^m)$ for programmable (n, k, m) are feasible but have some coding limitations by $n = 2^m - 1$, $r = n - k = 2t$, t = the number of error-correcting bytes. In these RS-code researches [9]-[14], the versatility of RS-decoders could be achieved by changing only the information length k with the block length n and symbol size m fixed [10], [11], [14], in order to change the error correcting capability t. Another type is to fix symbol-size m in order to the both n and k are variable [12]-[13]. These architectures pay a largest area cost for decoding circuit, and clock time and decoding latency is also bigger.

A class of multiple bits error correcting and detecting code were presented in [28]-[31] that these codes are based on Fujiwara codes, i.e. An odd-weight-column-matrix code over GF(2^b) is an SbEC-DbED code, where b denotes the number of bits in a byte and equal to m. the kinds of code may have arbitrary code and byte length, and the researches have a proper number of parity check-bits as follows. $R = \frac{1}{2}(3m + \log_2(k+2))$ in [28], and $R = \log_2 \lceil R + m \times k + m - 1 \rceil + m$ in [29], and $R = \log_2 \left| (m \times k)^2 + 2^{m-1} - 1 \right| - 1$ in [30]. The based-on Fujiwara codes for programmable (n, k, m) are feasible but still pay a complex decoding circuit though these codes have a proper error correcting capability, such as [30] has random double bit within a block error correction- single byte error detecting capability, [29] has single bit error correction-double bit error detection and fixed b-bit byte error detection capability, [28] has t-bit error correction within a single b-bit byte and single b-bit byte error detection capability, where t=3 and b=8, but they still have a poor error detection capability of random double/triple bit failures.

Chapter 3
The Proposed ECC Codes Constructed Methods and Interleaved Mechanisms

3.1 Constructed method of the proposed SEC-S$_{odd}$EC-SBED-DED ECC code

We propose a systematic error-correcting code by modified bidirectional cross-parity code which called SEC-S$_{odd}$EC-SBED-DED codes to have the kinds of capabilities, such as random Single bit Error Correction－Single odd-bit Error Correction within a single byte－Single Byte Error Detection－random Double bits Error Detection. Traditional bidirectional cross-parity/product SEC-DED codes need parity check-bits $r = k + m + 1$, that we utilize hierarchical structure to reduce the number of parity check bit. The proposed (n, k, m) systematic code is constructed as shown Fig. 3.1, and the following steps are performed:

Step1: To define an encoded data page or sector size as $m \times k$ bits for m-bit data wide and the information length k, where $0 \le i \le m-1$, and $0 \le j \le k-1$.

Step2: Each *i*-column is to perform a vertical-direction parity-bit for all $0 \le j \le k-1$ bits. This will result in generating m-bit column-parity-bits as the expression:

$C_i = \sum_{j=0}^{k-1} b_{ij}$ where addition is equal to XOR logic-operation and b$_{ij}$ indicates coordinate of one bit position.

Step3: Each row is to perform horizontal-direction parity bits using a hierarchical method for these m-bit bytes of k rows. This will result in generating ($2 \times \lceil \log_2 k \rceil$) row parity-bits. These parity check-bit generating expression is as follows.

$$R_1 = \sum_{i=0}^{m-1} b_{ij} \quad \text{for } j = odd \text{ ,i.e. (j mod 2)=1, and } \quad R_1' = \sum_{i=0}^{m-1} b_{ij} \quad \text{for } j = even \text{,i.e. (j mod 2)= 0.}$$

$$R_2 = \sum_{i=0}^{m-1} b_{ij} \quad \text{for (j mod 4) = 2 or 3, and } \quad R_2' = \sum_{i=0}^{m-1} b_{ij} \quad \text{for (j mod 4) = 0 or 1.}$$

$$R_3 = \sum_{i=0}^{m-1} b_{ij} \quad \text{for (j mod 8) = 4,5,6, or 7, and } \quad R_3' = \sum_{i=0}^{m-1} b_{ij} \quad \text{for (j mod 8) = 0,1,2 or 3.}$$

As the same computing form, we can continue to prove the formula:

If j=0, 1, 2….k-1, let $x = \lceil \log_2 k \rceil$ is the number of a pair of row parity-bits,

then $R_X' = \sum_{i=0}^{m-1} b_{ij}$ for (j mod 2^X) = 0 ~ ($2^{(X-1)} - 1$), and

$$R_X = \sum_{i=0}^{m-1} b_{ij} \quad \text{for (j mod } 2^X \text{)} = 2^{(X-1)} \sim (2^X - 1)\text{, where mod} \equiv \text{modulo-operator.}$$

Hence we can utilize $x = 1 \sim \lceil \log_2 k \rceil$ to compute all pairs of row parity-bits.

Step4: By the step1 to step3, we can get the total parity check bit: $(2 \times \lceil \log_2 k \rceil + m)$ for a data page size of $(m \times k)$ bits, hence the bits number of code-length n = k (the number of data-bits) + r (the number of parity check-bits) $= (k \times m) + (2 \times \lceil \log_2 k \rceil + m)$ bits. If m-bit data-wide usually represents a byte wide, let n $= k + (\frac{2 \times \lceil \log_2 k \rceil}{m} + 1)$ bytes.

In a write operation, firstly we write k data-bytes to external-memory in sequence, and then the encoded r parity check-bytes by step1~step3 continue to write in memory after k data-bytes, in order to finish the proposed systematic error-correcting-code. Hence a serial access operation, it just needs $n = k + r$ clock-cycle counts.

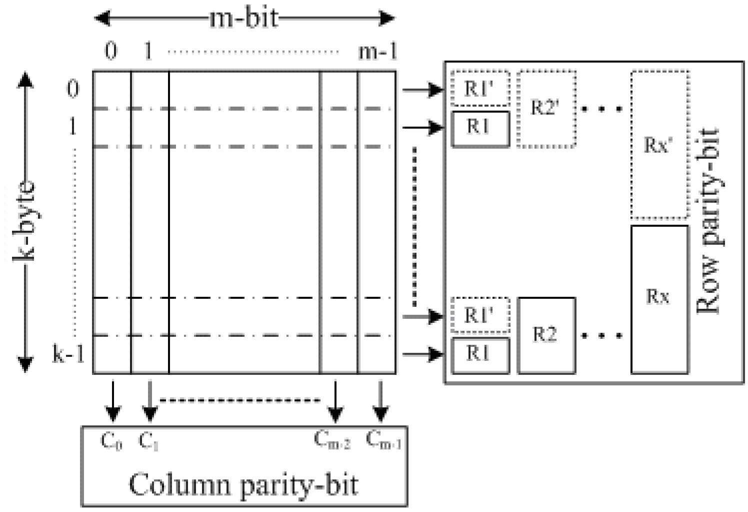


Fig. 3.1 The proposed error-correcting code structure for a (m×k) page size.

In a read operation, firstly we read n data-bytes from external-memory in sequence, and then the encoded r parity check-bytes by step1~step3 will gain new column parity check-bits C_i for $0 \le i \le m-1$ and new row parity check pair-bits (R_X, R'_X) for $1 \le x \le \lceil \log_2 k \rceil$ after the received k data-bytes. The decoding process is as following steps.

Step5: Syndrome generation methods are expressed as follows.

During the 0~(k-1) cycles for reading information bytes, and this k-th cycle for reading old column parity check-byte, and (k+1)~(n-1) cycles for reading old row parity check-bytes, so that we generate column syndrome bits at k-th cycle, and row syndrome bits during (k+1)~(n-1) cycles as the following expression:

Let b_{ik} indicates one bit position of old column parity check-byte on the read k-th byte for $0 \le i \le m-1$. Here addition is equal to an XOR logic-operation.

Column syndrome-bits： $S_{col}(i) = b_{ik} + C_i$

Let b_{ij} indicates one bit position of old row parity check-byte for $(k+1) \le j \le (n-1)$, $0 \le i \le m-1$, and $1 \le x \le \lceil \log_2 k \rceil$.

Row syndrome-bits： $S_{row}(x) = \sum_{i=1}^{m-1} b_{ij} + R_X$, for i= odd integer.

$S'_{row}(x) = \sum_{i=0}^{m-1} b_{ij} + R'_X$, for i= even integer.

Step6: Error correcting and detecting methods are analyzed as follows.

(a) No error: all $S_{col}(i) = S_{row}(x) = S'_{row}(x) = 0$, for $0 \le i \le m-1$, and $1 \le x \le \lceil \log_2 k \rceil$.

Another type of error is that it has a single bit error falling in the ECC-area, and we assume that it's no error occurring in information-area when the three $S_{col}(i), S_{row}(x), S'_{row}(x)$ results has only 1 bit equal to logic-1.

(b) SEC-S$_{odd}$EC: there are odd-bit errors occurring on a single-byte that these error-bits can be corrected. When the both $\sum_{i=0}^{i=m-1} S_{col}(i) = 1$, and $\{S_{row}(x) + S'_{row}(x)\} = 1$ for $1 \le x \le \lceil \log_2 k \rceil$ are existence, where $S_{col}(i)$ indicate error-bits position as an error value and $S_{row}(x)$ indicate error-address as an error location. We know the error value and error location that we can invert the error-bit data in order to correct it when the error address is read.

(c) SBED: if (a), (b) are inexistence, but $\sum_{i=0}^{i=m-1} S_{col}(i) = 0$ and $S_{row}(x) = S'_{row}(x) = 0$

for $0 \le x \le \lceil \log_2 k \rceil$, indicate that a single-byte error at least are detected.

In the other words, there are some even-bits errors occurring in a single-byte or multiple-bytes.

(d) DED: The type of error is assumed that it has the case of double errors or larger than double errors. If (a), (b), (c) are inexistence, then any $S_{col}(i) \neq 0$, $S_{row}(x) \neq 0$ or $S'_{row}(x) \neq 0$ indicates at least a double error existence.

The above (a), (b) both may be correctable, and the both (c) and (d) may be detectable and not to be correctable.

The foregoing error correcting code generation methods are very suitable for software or hardware implementation, programmable (n, k, m) ECC especially. By the above constructed method, we present a low-complexity and high-speed hardware in chapter 4.

Fig. 3.2 shows four capabilities of the proposed code for the16-bits memory page-size of information length k=4, and data wide m=4. The "X" denotes one fail-bit position.

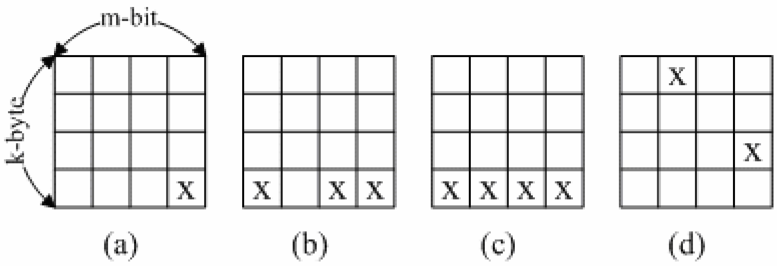


Fig. 3.2 The construct coding approach has these capabilities as (a)~(d).

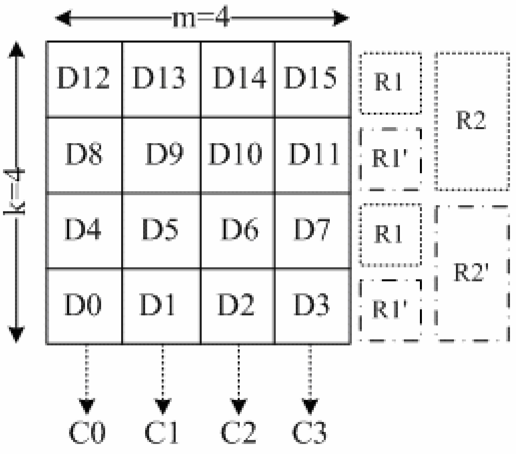
(a) SEC type: random single-bit error correction.

(b) S_{odd}EC type: odd-bit error correction within a single byte.

(c) SBED type: Single byte error detection.

(d) DED type: random double bit error detection.

Fig. 3.3 shows logical-scheme of the SEC-S_{odd}EC-SBED-DED code for the16-bits memory page-size of information length k=4, and data wide m=4. We can generate the parity check-bits by the foregoing step1~3.



Example: C0=D0+D4+D8+D12,…………,C3=D3+D7+D11+D15

R1'=D0+D1+D2+D3+D8+D9+D10+D11

⋮

R2=D8+D9+D10+D11+D12+D13+D14+D15

Fig. 3.3 the logical scheme of SEC-S_{odd}EC-SBED-DED code, where + indicates XOR logic-operation.

3.2 Constructed methods of Multi Bit-Layer SEC-S_{odd}EC-SBED-DED ECC code

Most interleaved techniques mainly can be used to solve the burst errors problems [32], and error patterns involving two or more adjacent cells are generally recovered by a proper physical interleaving of cells belong to the same codeword, thereby increasing overall memory reliability [18]. Some multi-SEC/DED codes are interleaved for each word-line of on-chip ECC or each data I/O of off-chip ECC are presented in [3], [4], [7], [16], [18]. The papers [3], [18] present an on-chip ECC scheme for MLC-flash memories, based on a binary code providing single-bit correction, are organized in different bit-layer. The paper [4] is a (522,512) SEC hamming cyclic code for each data I/O, that this multi-ECC (n, k) codes are optimized in consideration of balance between the reliability improvement and redundant-cells area overhead, but its weakness is that has a fixed 2n decoding latency even if the data is no error. The paper [7] is a TEC-QED ECC code which was designed by combining odd-weight-column SEC-DED hamming-code with the vertical parity bit techniques, but it has a large redundant cell overhead as parity-check bit $R = m \times (\log_2 k + 2) + k$. The paper [16] is a multi SEC-DED (n, k) Hamming code that a k-bit information data was split up into two SEC-DED hamming codeword so that it able to correct a two-bit error in two-bits-per-cell MLC-DRAM. The foregoing paper [3], [7],[16], [18] are only suitable to specific on-chip ECC coding way, and in practical, we need a compact, flexible and quick interleaved coding method in order to reach programmable coding and real-time mapping-out operation. So we propose an interleaved method which is called Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code. The principle of the proposed interleaved-code is to encode a $(k \times m)$ -bits block-data, and generate respective (n_l, k_l, m_l) SEC-S$_{odd}$EC-SBED-DED code on each data I/O so m-bit data I/O perform m-number of (n_l, k_l, m_l) SEC-S$_{odd}$EC-SBED-DED code which is called (n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code.

For a programmable (n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code, each data I/O code-length has n_l-bytes, the user-defined coding-data wide on each data-I/O has m_l-bit, the number of data I/O wide is m-bit, and the encoded information length on each data-I/O has also k_l-bytes correspond to $n_l = \left\lceil \frac{k+R}{m_l} \right\rceil$, where R is the number of parity check bits on each data-I/O equals $(2 \times \left\lceil \log_2\left(\frac{k}{m_l}\right) \right\rceil + m_l)$ -bit, and $k_l = \frac{k}{m_l}$ must be an integer.

When m=1 (only one data I/O), and k is equal to 64-bit, 512bit, 4096-bit respectively, the dependence of user-defined coding-data wide m_l and the number of parity check bit R is shown in Fig. 3.4.

For a Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code, if the number of data I/O wide is m, the number of information-length is k and the number of total information-bit is k×m, then the total number of parity check bit is

$$R = (2 \times \left\lceil \log_2\left(\frac{k}{m_l}\right) \right\rceil + m_l) \times m = (2 \times \lceil \log_2(k_l) \rceil + m_l) \times m \quad \text{(bits).}$$

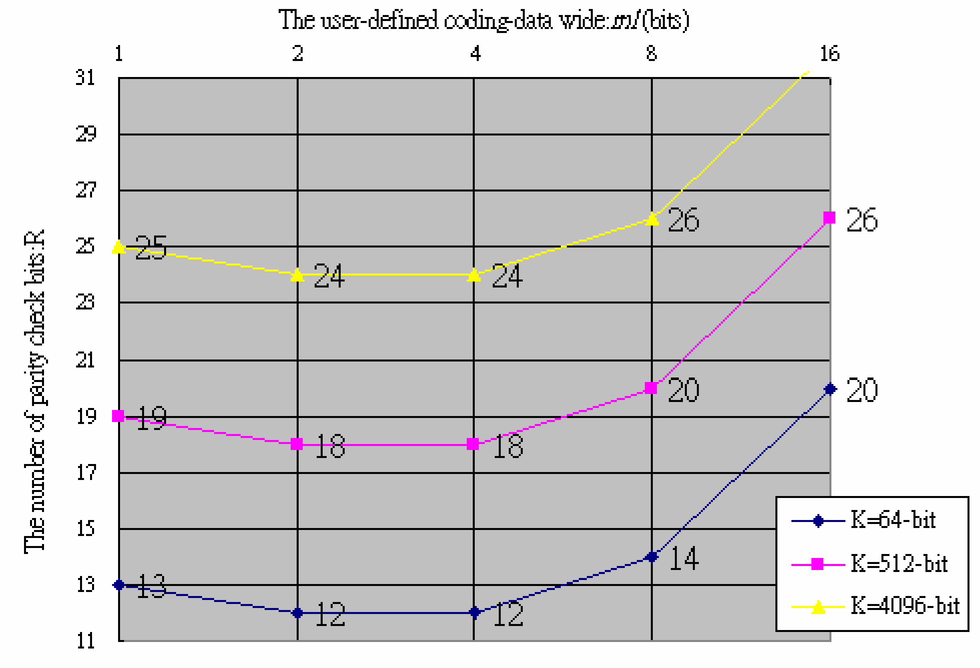


Fig 3.4 Dependence of the user-defined coding-data wide m_l and the number of parity-check-bits R for the proposed Multi-Bit-Layer codes when m=1

The above (n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S$_{\text{odd}}$EC-SBED-DED code can be constructed as shown in figure 3.5, and the generating method is described as following steps.

Step1: To define an encoded data page or sector size as $m \times k$ bits for m-bit data wide and the information length k, the user-defined coding-data wide on each data-I/O has m_l-bit, Firstly let $0 \le i \le m-1$, $0 \le j \le k-1$, and $0 \le h \le m_l - 1$ in order to construct multi-bit-layer ECC code, and we can gain $k_l = \dfrac{k}{m_l}$ =integer,

$$R = (2 \times \left\lceil \log_2\left(\frac{k}{m_l}\right) \right\rceil + m_l), \quad n_l = \left\lceil \frac{k+R}{m_l} \right\rceil$$

on each data-I/O. Thus each data I/O is to perform a single (n_l, k_l, m_l) Bit-Layer SEC-S$_{\text{odd}}$EC-SBED-DED code, total data-I/O number is m-bit to perform a (n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S$_{\text{odd}}$EC-SBED-DED code.

Step2: Each data I/O is to perform column-parity check-bits and row-parity check-bits, where the number of column-parity check-bits equal to m_l-bits and the number of row-parity check-bits equal to $(2 \times \lceil \log_2(k_l) \rceil)$ -bits. Let i=0, 1,till (m-1), and the both j, h are variable for $0 \le j \le k-1$, and $0 \le h \le m_l - 1$.

Hence each data I/O generate m_l -bit column parity bits as the expression:

$C_{i,h} = \oplus b_{i,j}$ if and only if (j mod m_l = h).

Each data I/O generate row parity bits as the expression:

$R_{i,1} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{m_l} \right\rfloor$ mod 2)=1, and $R^{'}_{i,1} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{m_l} \right\rfloor$ mod 2)= 0.

$R_{i,2} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{m_l} \right\rfloor$ mod 4) = 2 or 3, and $R^{'}_{i,2} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{m_l} \right\rfloor$ mod 4) = 0 or 1.

As the same computing form, we can continue to prove the formula:

If j is a variable equal to 0, 1, 2…till k-1, let $X = \left\lceil \log_2\left(\frac{k}{m_l}\right) \right\rceil = \lceil \log_2(k_l) \rceil$ is the number of a pair of row parity-bits, then $R^{'}_{i,X} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{m_l} \right\rfloor$ mod 2^X) = 0 ~ ($2^{(X-1)} - 1$), and $R_{i,X} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{m_l} \right\rfloor$ mod 2^X) $= 2^{(X-1)}$ ~ ($2^X - 1$), where mod $\equiv$ modulo-operator, the sign $\oplus$ equals to an XOR logic-operation, $\lfloor x \rfloor$ denotes the largest integer less than or equal to *x*, and $\lceil x \rceil$ denotes the smallest integer greater than or equal to *x*.

Step3: By the above step1 and step2, the $m \times k$ block-data is encoded into $m \times n = m \times (k + R)$ multi-bit-layer ECC code, and we can generate the parity-check bit on each data I/O:

$R = 2 \times \left\lceil \log_2\left(\frac{k}{m_l}\right) \right\rceil + m_l = 2 \times \lceil \log_2(k_l) \rceil + m_l$ (bits).

Total parity-check-bit overhead is $R \times m = (2 \times \lceil \log_2(k_l) \rceil + m_l) \times m$ (bit).

Thus an overall encoded page size equals to $(k \times m) + (2 \times \lceil \log_2(k_l) \rceil + m_l) \times m$ (bits).

An example of logical scheme is shown in Fig. 3.6, and the logical expressions of parity check-bits are also list in figure 3.6 by using the above step1~tep3.

In the figure 3.6, we know a (14, 8, 2, 4) Multi-Bit-Layer SEC-S$_{\text{odd}}$EC-SBED-DED code has 2-bit column-parity-check-bit, and 4-bit row-parity-check-bit on each data-I/O to encode an information area $(k \times m) = (8 \times 4)$ -bit into an ECC-codeword $(n_l \times m) = (12 \times 4) bit$.

Here we propose a powerful and programmable interleaved-code which is called

(n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code owns the capability of Single-Byte Error Correction－Odd-bytes Error Correction within m_l consecutive bytes－m_l consecutive Bytes Error Detection－random Double bits Error Detection on each data-I/O. Because of the constructed method of Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code has the simpler and faster generation steps, hence this interleaved code can be implemented in a compact and high-speed hardware circuit, and it has a very large programmable (n_l, k_l, m_l, m) range.

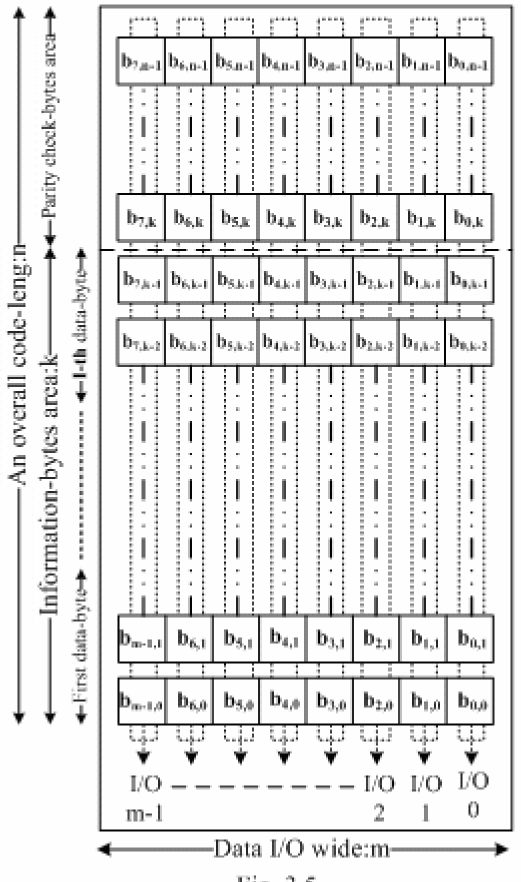
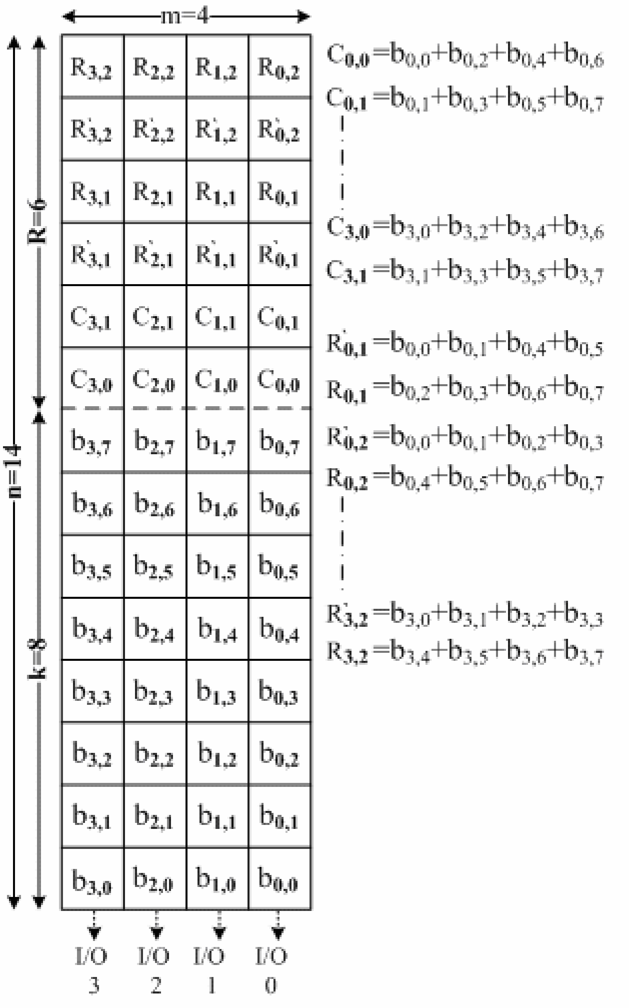

An overall code-leng:n
Information-bytes area:k
Parity check-bytes area
First data-byte
1-th data-byte
$b_{7,n-1}$ $b_{6,n-1}$ $b_{5,n-1}$ $b_{4,n-1}$ $b_{3,n-1}$ $b_{2,n-1}$ $b_{1,n-1}$ $b_{0,n-1}$
$b_{7,k}$ $b_{6,k}$ $b_{5,k}$ $b_{4,k}$ $b_{3,k}$ $b_{2,k}$ $b_{1,k}$ $b_{0,k}$
$b_{7,k-1}$ $b_{6,k-1}$ $b_{5,k-1}$ $b_{4,k-1}$ $b_{3,k-1}$ $b_{2,k-1}$ $b_{1,k-1}$ $b_{0,k-1}$
$b_{7,k-2}$ $b_{6,k-2}$ $b_{5,k-2}$ $b_{4,k-2}$ $b_{3,k-2}$ $b_{2,k-2}$ $b_{1,k-2}$ $b_{0,k-2}$
$b_{m-1,1}$ $b_{6,1}$ $b_{5,1}$ $b_{4,1}$ $b_{3,1}$ $b_{2,1}$ $b_{1,1}$ $b_{0,1}$
$b_{m-1,0}$ $b_{6,0}$ $b_{5,0}$ $b_{4,0}$ $b_{3,0}$ $b_{2,0}$ $b_{1,0}$ $b_{0,0}$
I/O m-1
I/O 2
I/O 1
I/O 0
Data I/O wide:m

Fig. 3.5



$R_{3,2}$	$R_{2,2}$	$R_{1,2}$	$R_{0,2}$
$R'_{3,2}$	$R'_{2,2}$	$R'_{1,2}$	$R'_{0,2}$
$R_{3,1}$	$R_{2,1}$	$R_{1,1}$	$R_{0,1}$
$R'_{3,1}$	$R'_{2,1}$	$R'_{1,1}$	$R'_{0,1}$
$C_{3,1}$	$C_{2,1}$	$C_{1,1}$	$C_{0,1}$
$C_{3,0}$	$C_{2,0}$	$C_{1,0}$	$C_{0,0}$
$b_{3,7}$	$b_{2,7}$	$b_{1,7}$	$b_{0,7}$
$b_{3,6}$	$b_{2,6}$	$b_{1,6}$	$b_{0,6}$
$b_{3,5}$	$b_{2,5}$	$b_{1,5}$	$b_{0,5}$
$b_{3,4}$	$b_{2,4}$	$b_{1,4}$	$b_{0,4}$
$b_{3,3}$	$b_{2,3}$	$b_{1,3}$	$b_{0,3}$
$b_{3,2}$	$b_{2,2}$	$b_{1,2}$	$b_{0,2}$
$b_{3,1}$	$b_{2,1}$	$b_{1,1}$	$b_{0,1}$
$b_{3,0}$	$b_{2,0}$	$b_{1,0}$	$b_{0,0}$

$C_{0,0}=b_{0,0}+b_{0,2}+b_{0,4}+b_{0,6}$
$C_{0,1}=b_{0,1}+b_{0,3}+b_{0,5}+b_{0,7}$

$C_{3,0}=b_{3,0}+b_{3,2}+b_{3,4}+b_{3,6}$
$C_{3,1}=b_{3,1}+b_{3,3}+b_{3,5}+b_{3,7}$

$R'_{0,1}=b_{0,0}+b_{0,1}+b_{0,4}+b_{0,5}$
$R_{0,1}=b_{0,2}+b_{0,3}+b_{0,6}+b_{0,7}$
$R'_{0,2}=b_{0,0}+b_{0,1}+b_{0,2}+b_{0,3}$
$R_{0,2}=b_{0,4}+b_{0,5}+b_{0,6}+b_{0,7}$

$R'_{3,2}=b_{3,0}+b_{3,1}+b_{3,2}+b_{3,3}$
$R_{3,2}=b_{3,4}+b_{3,5}+b_{3,6}+b_{3,7}$

Fig. 3.6

Fig. 3.5 Constructed scheme of the proposed (n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code.
Fig. 3.6 Logical scheme of a (n_l=7, k_l=4, m_l=2, m=4) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code for mxk=4x8 block-data.

3.3 Application Examples of Interleaved Mechanisms using Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code

Many types of interleaved methods using the proposed code can apply to different page size for various memory-chips system. Here we utilize Multi-Bit-Layer SEC-S$_{odd}$EC-SBED -DED codes to form two interleaved-code mechanisms, which are mainly applied to the NAND-type flash memory that its memory array consists of many pages unit. A page size equals 528-bytes which is used to page-program or page-read operation, and a block size are 32 pages which is used to block-erase operation.

The first interleaved method using Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code is described as follows:

Basically, the conventional multiple ECC codes utilize the proposed (n, k, m)=(66, 63, 8) SEC-S$_{odd}$EC-SBED-DED code to let a 528-bytes page-size is divided into 8 segments, and every segment has 63 data-bytes and additional 3 parity check bytes. The conventional multiple ECC codes organization are shown in Fig. 3.7 (a), and every segment has only the default error-correcting and detecting capability of SEC-S$_{odd}$EC-SBED-DED code, therefore it can not correcting single-byte error or detection double-byte errors. For improving this weakness of the conventional multiple ECC code, we may utilize the foregoing interleaved code in section 3.2 that the (n_l, k_l, m_l, m) =(66, 63, 8, 8) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED coding method to let this interleaved code is capable of Single-Byte Error Correction－Odd-bytes Error Correction within eight consecutive bytes－Eight consecutive Bytes Error Detection－random Double bits Error Detection on each data I/O as shown in Fig. 3.7 (b). According to figure 3.7 (a) and (b), we know segment-1 has 63 data-bytes from $D_{1,0}$, $D_{1,1}$ until $D_{1,62}$, and 3 parity check-bytes from $C_{1,0}$ to $C_{1,2}$, where each $D_{m,k}$ denotes a data byte position, each $C_{m,k}$ denotes a parity check-byte position, m=data wide=segment number for $1 \le m \le 8$, and k=the information length for $0 \le k \le 62$. Every segment stores 63 data-bytes with 3 parity check-bytes on each horizontal row-direction as figure 3.7 (a), so we change the arrangement of 66-bytes codeword to let 66-bytes(528-bits) lie on each vertical column-position in sequence as figure 3.7 (b). An n-bytes (n×m bits) codeword is stored in the same bit-layer data-I/O, i.e. the segement-1 codeword is stored in the first bit-layer data-I/O of the 528-bits ($b_{0,0}$~ $b_{0,511}$) memory space, the segement-2 codeword is stored in the second bit-layer data-I/O of the 528-bits ($b_{1,0}$~ $b_{1,511}$) memory space, and so on. Therefore the above way can create a (n_l, k_l, m_l, m) =(66, 63, 8, 8) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED code to fit into a 528-bytes page size.

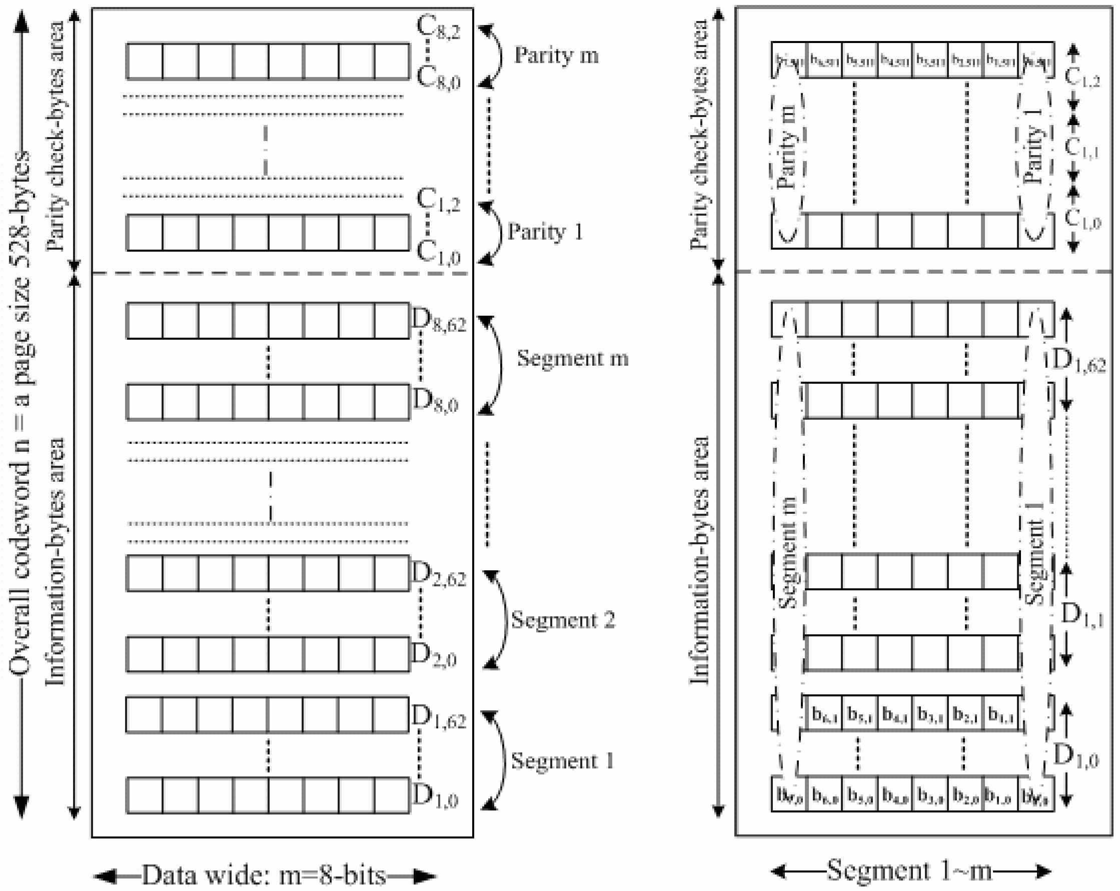


Fig. 3.7 (a) Fig. 3.7 (b)

Fig. 3.7 The (66, 63, 8, 8) SEC-S$_{odd}$EC-SBED-DED code is used for a 528-bytes page size.
(a) Conventional multiple ECC codes organization for a 528-byte page size.
(b) The proposed ECC code using interleaved techniques for a 528-bytes page size.

For a specific page size like shown in figure 3.7(b), we can construct the (66, 63, 8, 8) Multi-Bit-Layer SEC-S$_{odd}$EC-SBED-DED coding mechanism within a single-page by using section 3.2 constructing methods is described as following steps.

Step1: We have known a page size equal to $n \times m$ =528×8-bit. Let $m = m_l$, $n_l = \frac{n}{m_l}$ =integer, then a page has m number of bit-layer ECC-codeword, and each ECC-codeword is said that $R_{\max} = 2 \times \left\lceil \log_2\left(\frac{n}{m_l}\right) \right\rceil + m_l$ (bits), $k_l = \left\lfloor \frac{n - R_{\max}}{m_l} \right\rfloor$ (bytes), $k = k_l \times m_l$

(bits), and the real $R = 2\times\lceil \log_2(k_l)\rceil + m_l$ (bits), and $n_l = \frac{n}{m_l} = k_l + \left\lceil \frac{R}{m_l} \right\rceil$ (bytes).

By the above descriptions, we can solve the values: When $n = 528, m = m_l = 8,$ then

$k_l = 63, k = 504, R = 20, n_l = 66$.

Step2: Each data I/O is to perform an ECC-codeword with column-parity check-bits and row-parity check-bits. The number of column-parity check-bits is equal to m_l-bits.

Let *i*=0, 1, till (m-1), and j is a variable for $0 \le j \le k-1$.

Thus each data I/O generate *8*-bit column parity bits as the expression:

$C_{i,0} = \oplus b_{i,j}$ if and only if (j mod 8 = 0); $C_{i,1} = \oplus b_{i,j}$ if and only if (j mod 8 = 1);

$C_{i,2} = \oplus b_{i,j}$ if and only if (j mod 8 = 2); $C_{i,3} = \oplus b_{i,j}$ if and only if (j mod 8 = 3);

$C_{i,4} = \oplus b_{i,j}$ if and only if (j mod 8 = 4); $C_{i,5} = \oplus b_{i,j}$ if and only if (j mod 8 = 5);

$C_{i,6} = \oplus b_{i,j}$ if and only if (j mod 8 = 6); $C_{i,7} = \oplus b_{i,j}$ if and only if (j mod 8 = 7);

Each data I/O generate row parity bits as the expression:

$R_{i,1} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{8} \right\rfloor$ mod 2)=1, and $R'_{i,1} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{8} \right\rfloor$ mod 2)= 0.

$R_{i,2} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{8} \right\rfloor$ mod 4) = 2 or 3, and $R'_{i,2} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{8} \right\rfloor$ mod 4) = 0 or 1.

As the same computing form, we can continue to prove the formula:

If j is a variable equal to 0, 1, 2…till k-1, let $X = \lceil \log_2(k_l)\rceil = 6$ is the number of a pair of row parity-bits, then $R'_{i,6} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{8} \right\rfloor$ mod 2^6) = 0 ~ ($2^5 - 1$), and

$R_{i,6} = \oplus b_{i,j}$ iff ($\left\lfloor \frac{j}{8} \right\rfloor$ mod 2^6) = 2^5 ~ ($2^6 - 1$), where mod ≡ modulo-operator, the sign $\oplus$ equals to an XOR logic-operation, $\lfloor x \rfloor$ denotes the largest integer less than or equal to *x*, and $\lceil x \rceil$ denotes the smallest integer greater than or equal to *x*.

Step3: By the above step1 and step2, we can generate the parity-check bit on each data I/O: R= 20-bits, total parity-check-bit overhead is $R \times m = 160$ -bits and an overall encoded page size equals $(k_l \times m_l + R) \times m = (k + R) \times m$ =524×8-bit. So we must make sure that the overall encoded page size is smaller than a page size $n \times m$ =528×8-bit.

The result is that an interleaved $(n_l, k_l, m_l = m)$ SEC-S$_{\text{odd}}$EC-SBED-DED code for a single $(n \times m)$ page-size owns the capability of Single-Byte Error Correction—Odd-bytes Error Correction within *m* consecutive bytes—*m* consecutive Bytes Error Detection—random Double bits Error Detection on each data I/O.

The second interleaved method using Multi-Bit-Layer SEC-S$_{\text{odd}}$EC-SBED-DED code is described as follows:

Furthermore, we can continue to expand the error-correcting and detecting capable of an interleaved (66, 63, 8, 8) SEC-S$_{\text{odd}}$EC-SBED-DED code for four consecutive pages that this code owns these capability of four consecutive Bytes Error Correction—Odd-bytes Error Correction within (8×4) consecutive bytes—(8×4) consecutive Byte Error Detection—four random Double bits Error Detection on each data I/O. The skill is shown in Fig. 3.8, where $b_{f,i,j}$ indicates a data bit position, f= page number from 0 to 3, i=data wide=segment number from 0 to 7, j=the bit number of each segment from 0 to 2015. The segment of each page has 63 data-bytes equal to 504-bits length and 3 parity-check bytes equal to 24-bits length that we expand the overall interleaved codeword over four pages, e.g., the four-bits $b_{1,0,0}, b_{2,0,1}, b_{3,0,2}, b_{4,0,3}$ are included in the first group, where the first-bit $b_{1,0,0}$ is as the first data-bit of segment-1 in page1, the second-bit $b_{2,0,1}$ is as the first data-bit of segment-1 in page2, the third-bit $b_{3,0,2}$ is as the first data-bit of segment-1 in page3, the fourth-bit $b_{4,0,3}$ is as the first data-bit of segment-1 in page4, respectively. Therefore a four page has (63×8)=504 groups as 2016-bits data-length on each segment, it implies that every column or each segment has four (66, 63, 8, 8) SEC-S$_{\text{odd}}$EC-SBED-DED interleaved code over four consecutive-page for reaching to the four consecutive bytes error correcting ability.

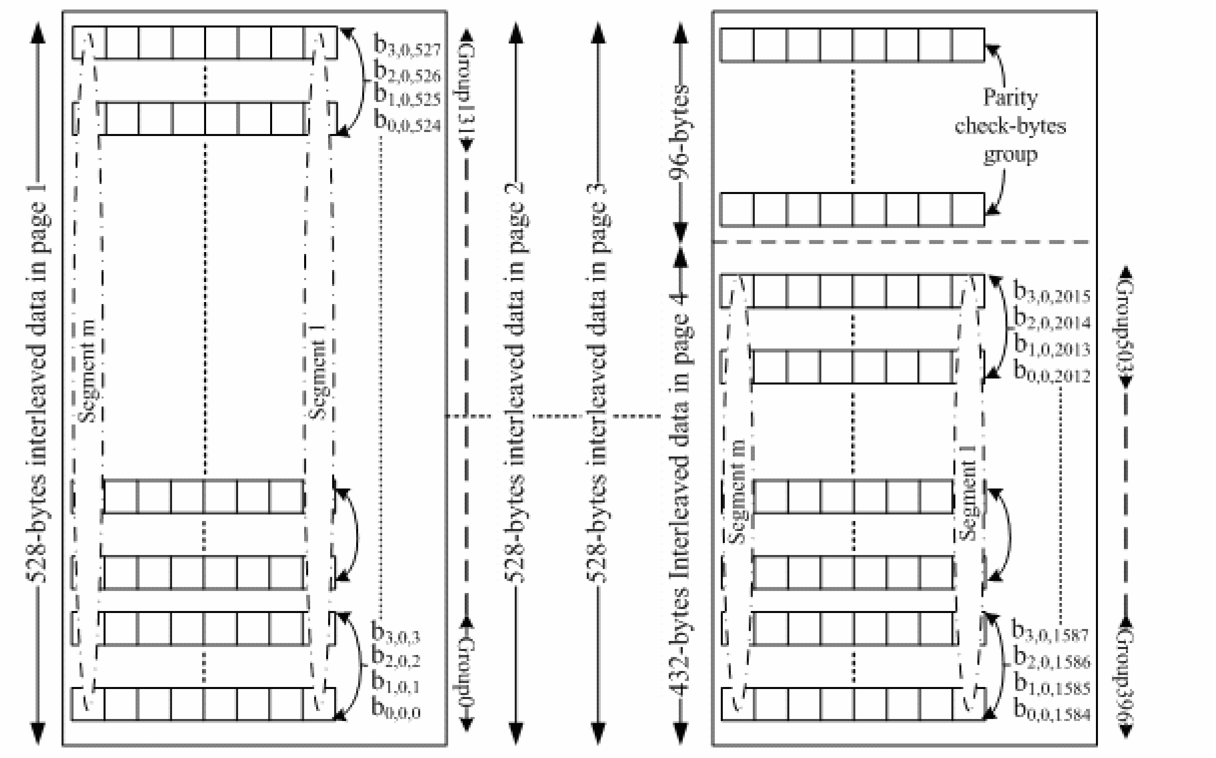


Fig. 3.8 Four (66, 63, 8, 8) SEC-S$_{\text{odd}}$EC-SBED-DED code is interleaved over four-page memory space.

Thus the specific consecutive-pages like shown in figure 3.8, we can construct the interleaved (n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S$_{\text{odd}}$EC-SBED-DED coding mechanism over *l*-page is described as following steps.

Step1: We have known a page size is equal to $(n \times m)$-bits, and need to encode l consecutive-pages. A single page has m number of bit-layer ECC-codeword, and each ECC-codeword is said that $R_{\max} = 2 \times \left\lceil \log_2\left(\frac{n}{m_l}\right) \right\rceil + m_l$ (bits), $k_l = \left\lfloor \frac{n - R_{\max}}{m_l} \right\rfloor$ (bytes), $k = k_l \times m_l$ (bits), and the real $R = 2 \times \lceil \log_2(k_l) \rceil + m_l$ (bits), and $n_l = \frac{n}{m_l} = k_l + \left\lceil \frac{R}{m_l} \right\rceil$ (bytes). The $b_{f,i,j}$ denotes the bit position of each data-I/O over *l*-page, where f denotes the number of page for $0 \le f \le l-1$, *i* denotes the number of data-I/O for $0 \le i \le m-1$, j denotes the number of the encoded information length over *l*-page for $0 \le j \le (k \times l) - 1$.

Step2: Each data I/O is to perform the interleaved ECC-codeword over *l*-page, the number of column-parity check-bits is equal to $(m_l \times l)$ bits, and the number of row-parity check-bits is equal to $\{(2\times\lceil\log_2(k_l)\rceil)\times l\}$ bits.

Let *i*=0~(m-1), f=0~$(l-1)$, and the both j, h are variable for $0 \le j \le (k\times l)-1$, $0 \le h \le m_l - 1$.

Hence each data I/O generate column parity bits as the expression:

$C_{f,i,h} = \oplus b_{f,i,j}$ if and only if { j $\bmod (m_l \times l) = (h\times l + f)$ }.

For example as shown in figure 3.8, the first data-I/O is to perform the interleaved ECC-codeword over *4*-page, the number of column-parity check-bits is equal to $(m_l \times l) = 8\times 4 = 32$ bits. Therefore we know $0 \le j \le 2015$, $0 \le h \le 7$ then

$C_{0,0,h} = \oplus b_{0,0,j}$ if and only if { j $\bmod 32 = (h\times 4)$ },

$C_{1,0,h} = \oplus b_{1,0,j}$ if and only if { j $\bmod 32 = (h\times 4+1)$ },

$C_{2,0,h} = \oplus b_{2,0,j}$ if and only if { j $\bmod 32 = (h\times 4+2)$ },

$C_{3,0,h} = \oplus b_{3,0,j}$ if and only if { j $\bmod 32 = (h\times 4+3)$ }.

Each data I/O generate row parity bits as the expression:

$R_{f,i,1} = \oplus b_{f,i,j}$ iff ($\left\lfloor \frac{j}{m_l \times l} \right\rfloor$ mod 2)=1, and $R'_{f,i,1} = \oplus b_{f,i,j}$ iff ($\left\lfloor \frac{j}{m_l \times l} \right\rfloor$ mod 2)= 0.

$R_{f,i,2} = \oplus b_{f,i,j}$ iff ($\left\lfloor \frac{j}{m_l \times l} \right\rfloor$ mod 4) = 2 or 3, and $R'_{f,i,2} = \oplus b_{f,i,j}$ iff ($\left\lfloor \frac{j}{m_l \times l} \right\rfloor$ mod 4) = 0 or 1. As the same computing form, we can continue to prove the formula:

Let $X = \left\lceil \log_2\left(\frac{k}{m_l}\right) \right\rceil = \lceil \log_2(k_l) \rceil$ is the number of a pair of row parity-bits,

then $R'_{f,i,X} = \oplus b_{f,i,j}$ iff ($\left\lfloor \frac{j}{m_l \times l} \right\rfloor$ mod 2^X) = 0 ~ ($2^{(X-1)} - 1$), and

$R_{f,i,X} = \oplus b_{f,i,j}$ iff ($\left\lfloor \frac{j}{m_l \times l} \right\rfloor$ mod 2^X) = $2^{(X-1)}$~ ($2^X - 1$).

For example as shown in figure 3.8, the first data-I/O is to perform the interleaved ECC-codeword over *4*-page, the number of row-parity check-bits is equal to $\{(2\times\lceil\log_2(k_l)\rceil)\times l\} = \{(2\times\lceil\log_2(63)\rceil)\times 4\} = 48$ bits, and the number of a pair of row parity-bits $X = \lceil \log_2(k_l) \rceil = 6$. When $0 \le j \le 2015$, $0 \le h \le 7$, and f=1, we know that

$R_{0,0,1} = \oplus b_{0,0,j}$ iff ($\left\lfloor \frac{j}{32} \right\rfloor$ mod 2)=1, and $R'_{0,0,1} = \oplus b_{0,0,j}$ iff ($\left\lfloor \frac{j}{32} \right\rfloor$ mod 2)= 0, $R_{0,0,2}$

and $R'_{0,0,2}$,…., till $R_{0,0,6} = \oplus b_{0,0,j}$ iff ($\left\lfloor \frac{j}{32} \right\rfloor \bmod 2^6) = 2^5 \sim (2^6 - 1)$,

and $R'_{0,0,6} = \oplus b_{0,0,j}$ iff ($\left\lfloor \frac{j}{32} \right\rfloor \bmod 2^6) = 0 \sim (2^5 - 1)$.

Step3: By the above step1 and step2, we can generate the parity-check bit of each data-I/O over *l*-page: $R = (2 \times \lceil \log_2(k_l) \rceil + m_l) \times l$ (bits).

Total parity-check-bit overhead over *l*-page is $R = (2 \times \lceil \log_2(k_l) \rceil + m_l) \times l \times m$ (bit).

Thus an overall encoded page size is equal to $\{k + (2 \times \lceil \log_2(k_l) \rceil + m_l) \times l\} \times m$ (bits).

Therefore, the consequence is that a $(n \times m)$ page-size and *l*-consecutive-pages using an interleaved (n_l, k_l, m_l) SEC-S$_{\text{odd}}$EC-SBED-DED code which owns the capability of l-Byte Error Correction－Odd-bytes Error Correction within $(l \times m_l)$ consecutive bytes－$(l \times m_l)$ consecutive Bytes Error Detection－*l*-random Double bits Error Detection on each data I/O.

Chapter 4
Programmable Architecture, Circuit and Software Program Design for the proposed FEC Codec

4.1 Programmable architecture of the proposed FEC Codec

For a various page or sector size to different memory-chips, The proposed error-correcting methods are designed into a programmable encoder-decoder for (n, k, m)= (4100~4, 4096~2, 8~1), where information-length k may be changed from 2, 3 or 4, until 4096-bytes and m may be changed from 1, 2, or 3 until 8-bits, and $n = k + r$ in byte notation, or $n \times m = k \times m + r \times m$ in bit notation as shown in table 4.1.

Table 4.1 When data-wide m = 8-bit as one byte, the number of parity check bits for different information length.

Information (k-bytes)	**2~16Bytes**	**17~256Bytes**	**257~4096Bytes**	**Above 4097Bytes**
Column Parity-bits	1Byte(8bit)	1Byte(8bit)	1Byte(8bit)	1Byte(8bit)
Row Parity-bits	1Byte(2~8bit)	2Byte(10~16bit)	3Byte(18~24bit)	4Byte(26bit)
Total parity-bit r-bytes, (R-bits)	2Byte(10~16bit)	3Byte(18~24bit)	4Byte(26~32bit)	5Byte
Codeword n-bytes	N=K+2 (4~18Byte)	N=K+3 (20~259Byte)	N=K+4 (261~4100Byte)	N=K+5 (4102Byte≦)
Code-rate (k/n)	0.5~0.889	0.85~0.988	0.985~0.999	0.9988~

The block diagrams of the proposed programmable encoder-decoder architectures are shown in Fig. 4.1. These functional sub-blocks are described as follows.

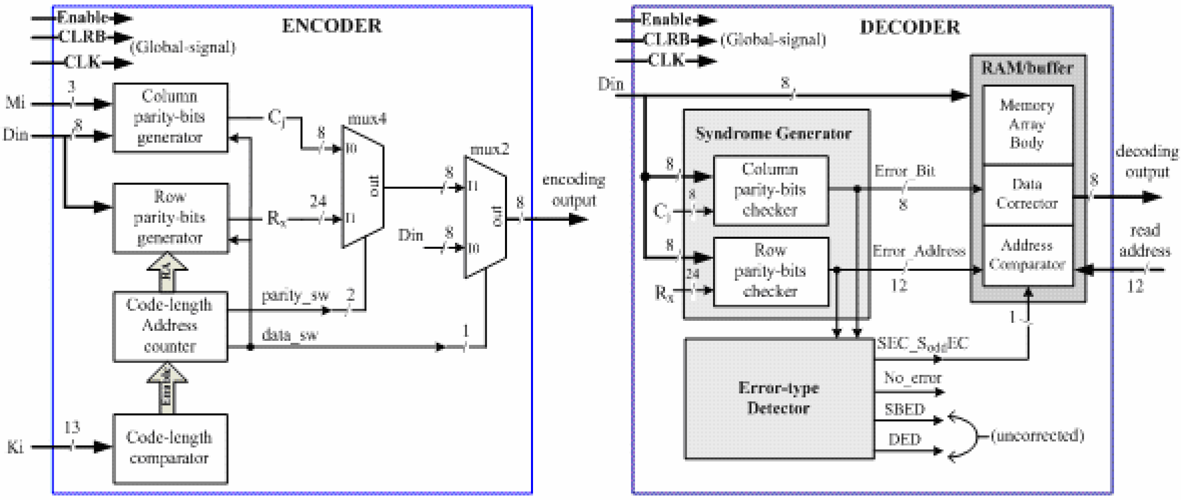


Fig. 4.1 Block diagram of the proposed programmable ECC Codec.

4.1.1 Sub-block Functions of Encoder

Column parity-bits generator: By the foregoing step2, this generator uses XOR-operation from the first data-bit of data-width to the final data-bit of data-width that the results of XOR-operation store in m column parity-bit registers, where m=1~8bit is decided by 3-bit value of m_i as below.

Mi (setup of data wide)	000/b	001/b	010/b	011/b	100/b	101/b	110/b	111/b
m (data-I/O wide)	1-bit	2-bit	3-bit	4-bit	5-bit	6-bit	7-bit	8-bit

Row parity-bits generator: By the foregoing step3, this generator also uses XOR-operation from the first data byte of page-length to the final data-byte of page-length that the results of XOR-operation store in x-pairs of row parity-bit registers, where $x = \lceil \log_2 k \rceil$ is decided by 13-bit value of k_i as below.

Ki (setup of data length)	0~1/h	2/h	3/h	4/h	…	0fff/h	1000/h	above
k (data-byte length)	unused	2B	3B	4B	…	4095B	4096B	unused

Code-length control unit: It may setup the different page-sizes from 2-bytes to 4096-bytes that it consists of Code-Length Address Counter and Code-Length Comparator.

Code-length comparator decides mainly the number of parity check-bytes by the comparison with k_i value.

Code-length address counter executes the work of target address counter so that row parity-bits generator can carry out each byte XOR-operation, and switch the row parity-bytes, column parity-bytes, and information/data bytes to encoding output terminal.

4.1.2 Sub-block Functions of Decoder

Syndrome unit: The functional block mainly consists of column parity-bits checker and row parity-bits checker. The syndrome bits may be gained by the foregoing step5.

Column parity-bits checker: In read operation, the old parity check bytes are compared with new generation parity-bytes in order to generate the column syndrome bits which indicate the error-bits positions.

Row parity-bits checker: the old parity check bytes are compared with new generation parity-bytes in order to generate the row syndrome bits which indicate the error address.

Error-type Detector: By the foregoing step6. It mainly shows that odd-bits error in a single byte as SEC-S_{odd}EC type, or no error existence, or an uncorrectable error existence as SBED-DED type.

The error correction mechanism is that the read-out data-byte as an error byte can corrected simultaneously when the error address is read from buffer memory. In other words, after the n-bytes data are received, the CPU can execute program code in any place of buffer memory. If the error address is read, the error-bits will be output inversely.

Finally, the programmable architecture of the proposed FEC codec has the characteristics as shown in table 4.2 and table 4.3. Pin descriptions of the FEC codec are listed as shown in table 4.4.

Table 4.2 Operating clock-cycle counts of each functional block during encoding process for a page write.

Encoder Block-name	Code-length Control unit	Column parity-bits generator	Row parity-bits generator	Clock-gating multiplexer
Cycle count	n	n	n	n
Decoder Block-name	Error-type generator	Column parity-bits checker	Row parity-bits checker	Internal SRAM memory
Cycle count	0	0	0	0

Table 4.3 Operating clock-cycle counts of each functional block during decoding process for a page read

Encoder Block-name	Code-length Control unit	Column parity-bits generator	Row parity-bits generator	Clock-gating multiplexer
Cycle count	n	n	n	n
Decoder Block-name	Error-type generator	Column parity-bits checker	Row parity-bits checker	Internal SRAM memory
Cycle count	n-k+1	n-k	n-k	k

Table 4.4 Pin descriptions of the FEC codec

Pin name	I/O(bit-number)	Description
DIN	I(8bit)	Data-input of Codec for write/read-operation
CLK	I(1bit)	Codec clock (positive-edge latch)
CLRB	I(1bit)	Codec reset (active low)
EN_ENC	I(1bit)	Encoder enable (active high)
EN_DEC	I(1bit)	Decoder enable (active high)
Mi	I(3bit)	The number of Data-I/O wide
Ki	I(13bit)	The number of Data-length
read_addr	I(10bit)	Address-input of decoder for the read-operation from system-request
buf_dout	I(8bit)	Data-input of decoder from buffer for the read-operation.
buf_din	I(8bit)	Data-input of memory-buffer
buf_wr	I(1bit)	Write or read control-signal of memory-buffer
enc_dout	O(8bit)	Data-output of encoder in order to write data to external-memory
dec_dout	O(8bit)	Data-output of decoder for the read-operation of memory-buffer.
end	O(1bit)	Response of the end of a page access
*one_err	O(1bit)	Response of single-error detection
*two_err	O(1bit)	Response of double-error detection
*no_err	O(1bit)	Response of no-error detection
*err_addr	O(12bit)	Response of error address location
*err_bit	O(8bit)	Response of error bit location

4.2 Circuit Design of the proposed FEC Codec

The section introduce the circuit implementation that the proposed FEC-codec has a low-complexity logic circuit (FEC-Codec gate-count = 883+628=1511), high-speed operation (clock frequency larger than 400Mhz). The column and row parity-check-bit generators are shown in Fig. 4.2 and Fig. 4.3 respectively.

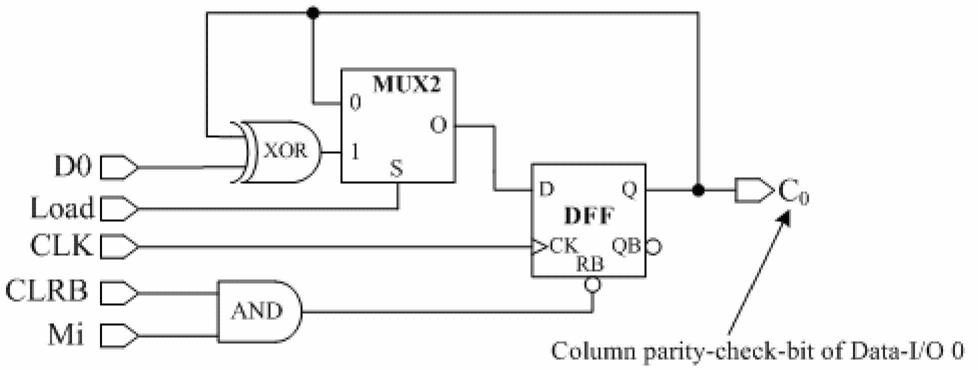


Fig. 4.2 Schematic of one-bit column parity-check-bit generator.

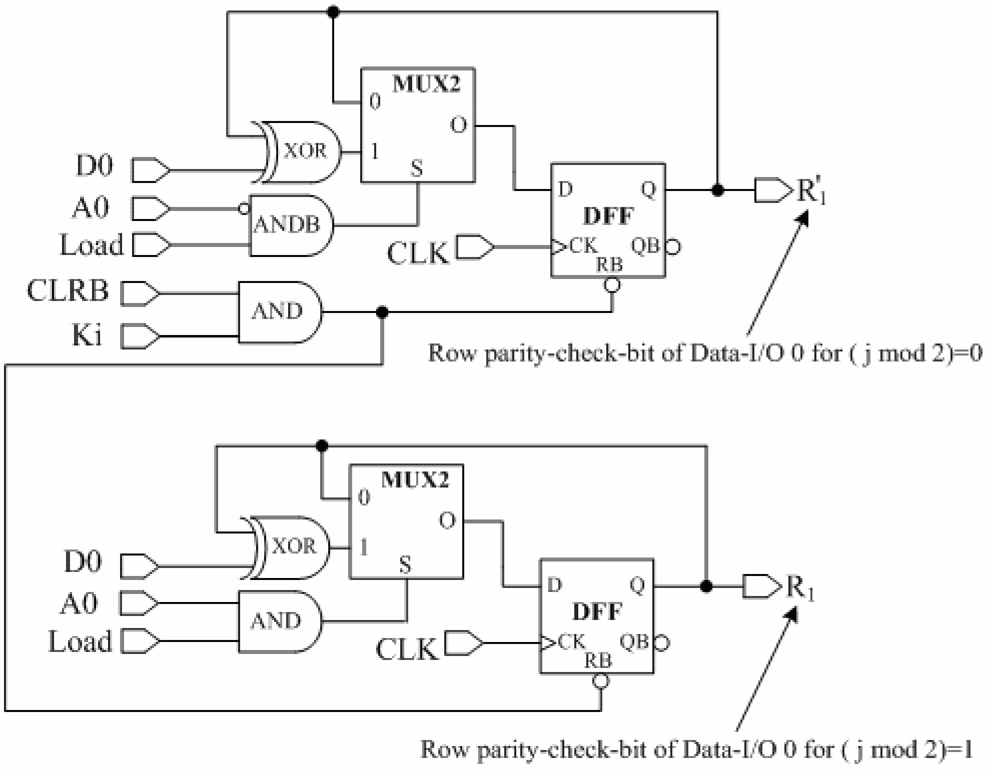

MUX2
0
O
1
S
XOR
D0
A0
Load
ANDB
CLK
CLRB
Ki
AND
D
Q
DFF
CK
QB
RB
R'_1
Row parity-check-bit of Data-I/O 0 for (j mod 2)=0
MUX2
0
O
1
S
XOR
D0
A0
Load
AND
CLK
D
Q
DFF
CK
QB
RB
R_1
Row parity-check-bit of Data-I/O 0 for (j mod 2)=1

Fig. 4.3 Schematic of one-pair of row parity-check-bit generator, where j is information-length from 0 to k-1.

The critical path of FEC-codec is created by the code-length address counter and code-length comparator.

Here we use 0.35um logic-process under high-temperature=90℃, and VDD=2.5V to estimate critical path delay of the proposed FEC-codec as shown in the following table 4.5.

Table 4.5 Critical path delay for each stage

stage	Gate name of each-stage	Gate-delay of each stage
1	CK to Q delay	0.8ns
2	Two-input XOR	0.35
3	Three-input NOR	0.35
4	Three-input NAND	0.25
5	Two-input NOR	0.2
6	Pass-gate + Ts(setup-time)	0.5ns
total	1+2+3+4+5+6	2.45ns (408Mhz)

Hence we know that the clock speed of the FEC-codec can be operated larger than 400-Mhz. The operating speed of FEC-codec can be cover all-kinds of high-speed memory-chip in real-time operation. Furthermore, timing flow-diagram of encoder-decoder logic-behavior are shown in Fig. 4.4, and 4.5 separately.

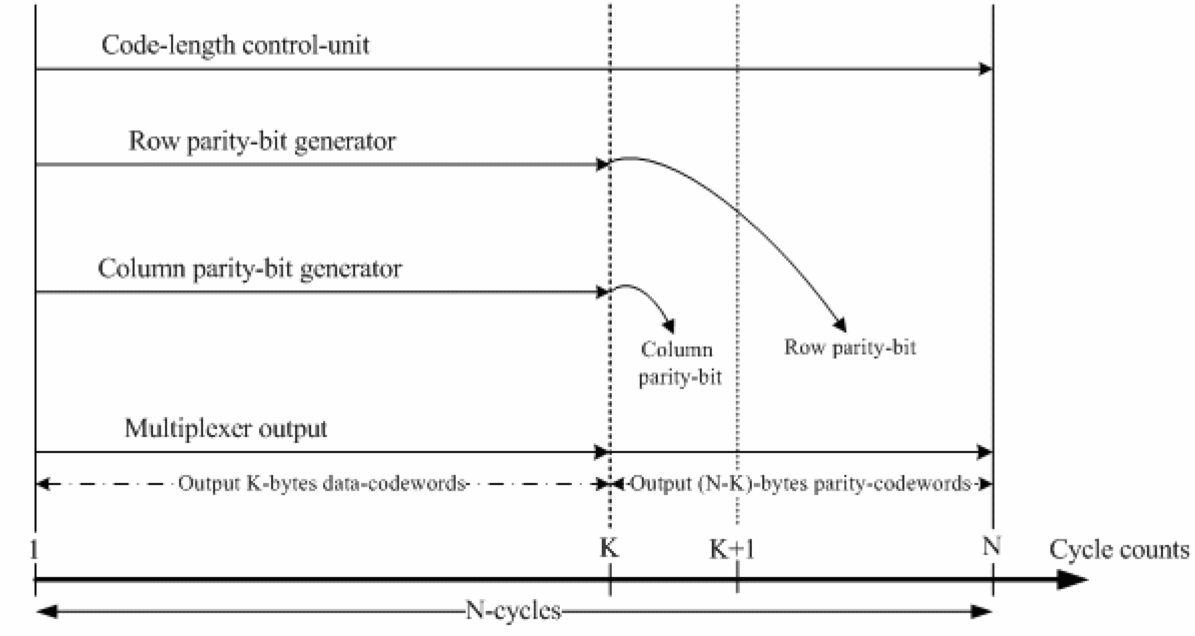


Fig. 4.4 An example of encoding procedure of the proposed FEC-Codec, where K=information-length, N=an ECC code-length, R=parity-bit length=N-K.

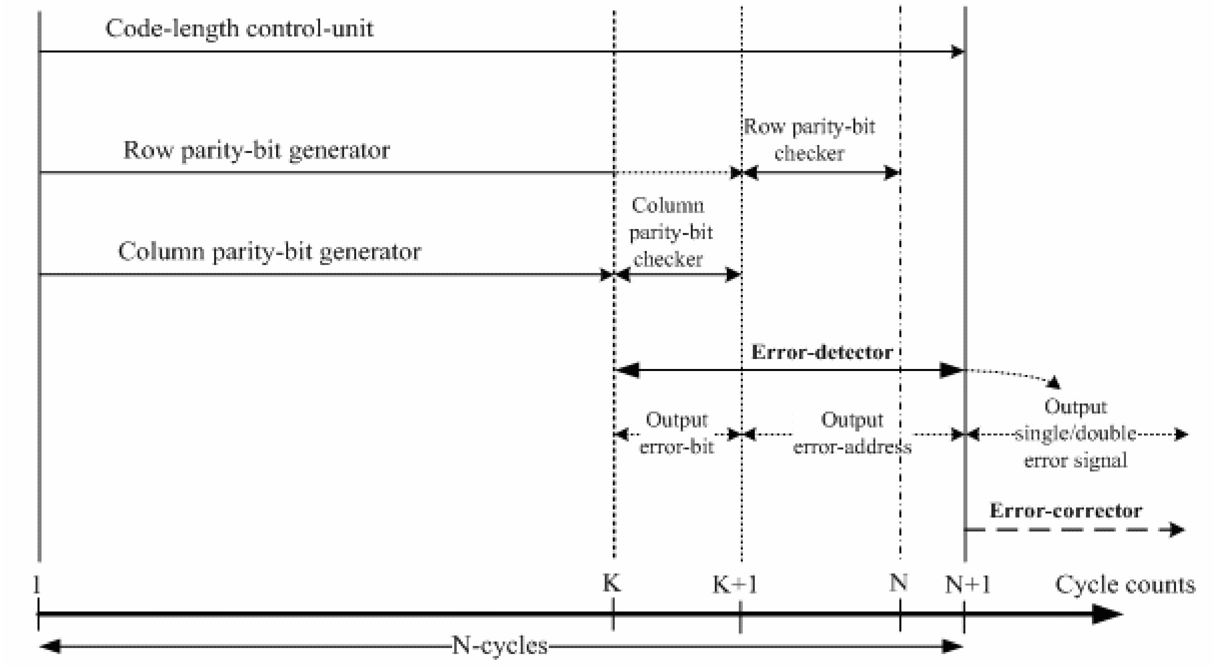


Fig. 4.5 An example of decoding procedure of the proposed FEC-Codec, where K=information-length, N=an ECC code-length, R=parity-bit length=N-K.

The programmable (n, k, m)=(4100~4, 4096~2, 8~1) FEC-codec logic gate count can be estimated without k-bytes RAM-buffer as table 4.6.

Table 4.6 Estimated counts for the proposed FEC-Codec

no	Cell name	Equivalent gate-counts	The number of cells for encoder	The number of cells for decoder
1	DFF with reset	6.6	57	41
2	Two-input NAND	1	63	35
3	Three-input NAND	1.4	24	6
4	Two-input NOR	1.2	4	18
5	Three-input NOR	1.5	32	8
6	Two-input multiplexer	2.2	85	61
7	Two-input XOR	2.5	52	46
8	Inverter	0.6	52	35
9	Latch	4.4	2	2
10	Equivalent gate-counts (1+2+3+4+5+6+7+8+9)	-	883	628

The proposed FEC codec can gain low complexity logic circuits, and the number of gate-counts are 1511 and only slightly incremental to code-length and data-wide width. In the other words, the number of gate-counts is slight increment by the code-length and data-wide, and RAM buffer size is proportional to code-length and data-wide. Basically, this FEC-codec increases 1-bit data-wide which only need to increase about 50 gate-counts, and increases a multiple of code-length which only need to increase also about 50 gate-counts, i.e. the programmable (n, k, m) equal to (65540~3, 65536~2, 16~1), only need the additional gate counts about 1200, so the total gate-counts of FEC-codec are 2711.

Moreover, the modular design methods are very suitable for the proposed FEC-codec to keep a constant high-speed operation over 400Mhz clock rate, regardless of the number of the both code-length and data wide.

4.3 Software-Program Design for an arbitrary (n, k, m) SEC-S_{odd}EC-SBED-DED Codec

Here we use C/C++ language program to finish the software design for an arbitrary (n, k, m) SEC-S_{odd}EC-SBED-DED code. The C-language application program is designed in order to simulate error-correcting and detecting behavior model of the proposed (n, k, m) FEC-codec that this program design architecture is shown in Fig. 4.6.

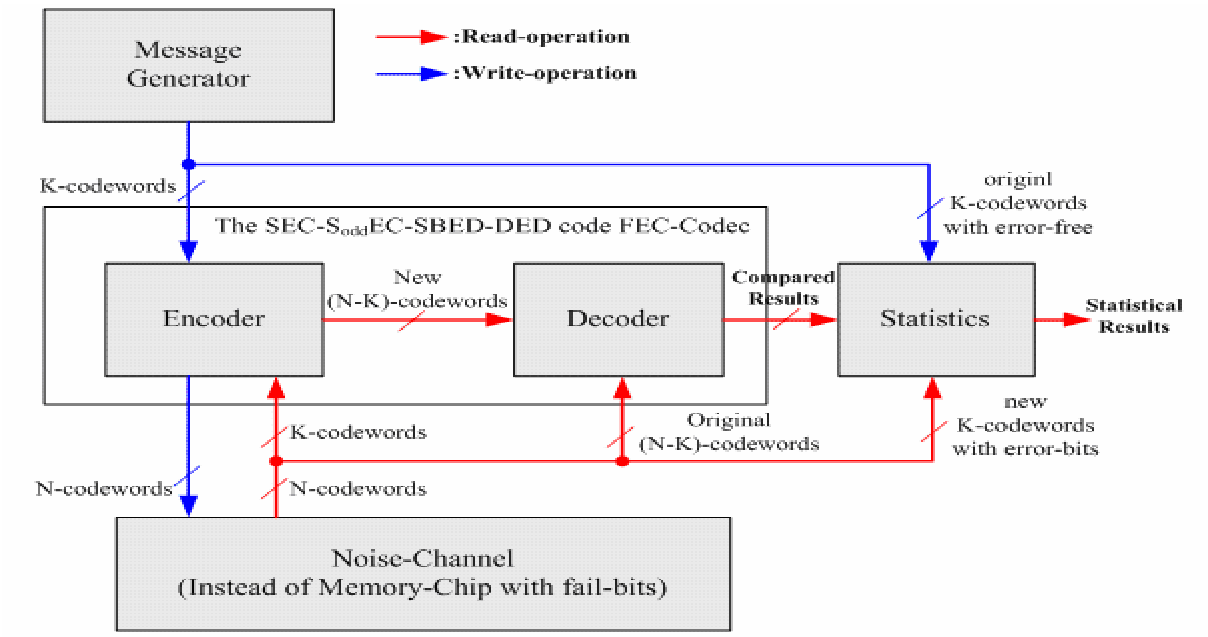


Fig. 4.6 Design architecture of C-language program for the proposed SEC-S_{odd}EC-SBED-DED code FEC-Codec.

This program has five function-block is described as follows.

(1) Message-Generator：To generate k message or information-codewords by using a random generator and every codewords has m-bits the, i.e. codewords value is among $0 \sim (2^m - 1)$. The number of messages can be set as a parameter, and n codeword-length including k-message or information-length and n-k parity-check codewords that these n, k, m value can be set as parameters, too.

(2) Encoder：To encode k message-codewords into an ECC codeword that owns n codewords including k message or information-codewords and (n-k) parity-check codewords.

(3) Noise-Channel：To receive n codewords and fail-bits are introduced into n codewords by Noise-mode. The noise channel instead of memory-chip with fail-bits, and we create 6 type of noise mode as (a) ~ (e).

(a) Error-Free channel.

(b) Random bit error for a BSC (Binary Symmetric Channel) with a bit error transition probability p. Hence each bit of overall codewords has the same error probability p and p can be set manually.

(c) Burst bit errors for a BSC (Binary Symmetric Channel) with a bit error transition probability p. The burst error length L and p can be defined manually, and we have some ‘tricks’ on it to guarantee that no two sequential burst errors will be connected together.

(d) Fixed p-bit random-bit errors on each (n, k, m) code, i.e. every (n, k, m) code can be induced with p-bit error by manual input, where p=1~ n.

(e) Fixed p-bit random bit errors within a single byte of each (n, k, m) code, where p=1~m.

(4) Decoder：To receive the original n-k parity-check codewords, compared with new generating parity-check codewords. The decoder will show 5 error types, such as a single-bit error, odd-bit error within a single codeword, a codeword (byte) error, two random bit errors at least, and a single-bit error in parity check codewords.

(5) Statistics：To receive the both original and new codewords that compare the both and analyze these error type so that we can make a statistical results for the number of error bit, the number of error codeword, the number of error sector.

4.4 Comparisons and Summary

4.4.1 Comparisons of Random multiple-bit errors Detecting Capability

The comparison of error-detection capabilities of existing ECC codes [28]-[31] and our proposed SEC-S_{odd}EC-SBED-DED code are listed as following table 4.7, that use the same information-bit length, such as K_1=64-bits with parity-check R_1 bits, K_2=128-bits with parity-check R_2 bits, and this table mainly shows the comparison results for random double-bit errors, and random triple-bit errors detecting capabilities, and mis-correction capability. The existing code [28]-[32] has specific error-correcting and detecting capabilities, such as [28] has 3-bit error correction within a single 8-bit byte and single 8-bit byte error detection capability, and [29] has single bit error correction-plus fixed 8-bit byte error correction capability, and [30] has random double bit error correction within 16-bytes - single 4-bit byte error detecting capability, and [31] has single bit error correction and a single 4-bit byte error detecting capability within each 32-bit block-length. The [32] has a single bit error correction and double bit error detection, and Odd-weight-column Hamming-code has better triple error-bits detecting capabilities than traditional Hamming-code. Furthermore, our proposed code has single bit error correction and a single odd-bit Error Correction within a single byte single byte error detection and random double bits error detection capability for any (n, k, m) value. The error-detection capabilities results of our proposed code are simulated by the parameters:

The k information or message codewords has 10^6 pattern-numbers by random-generator, which are executed on noise-channel (d) item for p=2, or 3, i.e. 2-bit or 3-bit random errors on every (N, K,M) block-code.

Table 4.7 Comparisons of random-bit errors detecting capabilities and mis-correction

ECC-codes	[28]		[29]		[30],[31]		[32]		**ours**	
ECC-code capability	$S_{3/8}$EC-S_8ED, For (76,64) and (141,128) code.		(S+F8)EC, For (79,64) and (144,128) code.		DEC$_{16}$-S_4ED, for (270,256) code. SEC-$S_{4/32}$EL, for (72,64) code.		K1:Odd-weight-column Hamming-code K2:Non-Odd-weight column Hamming-code		**SEC**/S_{odd}EC--SBED-DED, For (78,64,8) and (144,128,8) code.	
Data-bit and Parity-bit	K_1=64 R_1=12	K_2=128 R_2=13	K_1=64 R_1=15	K_2=128 R_2=16	K_1=64 R_1=8	K_2=256 R_2=14	K_1=64 R_1=8	K_2=64 R_2=8	K_1=64 R_1=14	K_2=128 R_2=16
Detection capabilities of double-bit errors (%)	44.52	46.16	≦64.9 (a)	≦55.8 (a)	54.0	53.31	99~100	99~100	81.34 **(81.34)**	90.54 **(90.54)**
Detection capabilities of Triple-bit errors (%)	44.60	47.23	≦74.2 (a)	≦63.9 (a)	-	88.20	43.72	24~43.5	57.60 **(83.65)**	39.47 **(78.85)**
Double-bit error miscor-rection or mislocate (%)	≦55.5 (a)	≦53.8 (a)	35.15	44.26	46.0	≦46.7 (a)	-	-	17.53 **(17.53)**	9.23 **(9.23)**
Triple-bit error miscor-rection or mislocate (%)	≦55.4 (a)	≦52.8 (a)	25.85	36.15	-	≦11.8 (a)	-	-	42.18 **(16.13)**	60.48 **(21.11)**

Note that (a): denotes an estimated bound-value, though the papers don't refer to these values.

(b) - : the papers don't refer to the value.

(c) If our proposed code is the (78, 64, 8) and (144, 128, 8) SEC-S_{odd}EC-SBED-DED code, triple-bit miscorrection is 42.18% and 60.48%, but we don't use odd-bit errors correction within a single-byte, i.e. the (78, 64, 8) and (144, 128, 8) SEC-SBED-DED code only has single-bit error correction capability, then the percentages of triple-bit miscorrection are reduced to 16.13% and 21.11% respectively.

The summary of the compared results in the table 4.7 are that the proposed (78, 64, 8) and (144, 128, 8) SEC-S_{odd}EC-SBED-DED code has a better double-bit errors detection capability about 1.9, 1.6, 1.7, and 1.5 times than [28], [29], [30], and [31] respectively, and a better triple-bit errors detection capability about 1.29, and 1.32 times than [28], and [32] respectively. Furthermore, it also has a smaller double-bit errors miscorrection about 0.31, 0.5, 0.2, and 0.38 times than [28], [29], [30] and [31] respectively.

4.4.2 Comparisons and Analysis of Sector Error Rate for NAND-flash memory

In Flash memory, the erase operation is performed in sector units, and a sector is 8k-bytes equal to 16-pages, where a page is (512+16)-bytes. Hence we introduce sector error rate, which is the probability that an error occurs in a sector [4]. The sector error rate of the flash-memory can denote the endurance level in program and erase cycles.

Part Ⅰ：Principle for the error probability of no alignment

A memory has **W** pages, we can assign the number of error-correcting-code words has also **W** ECC-codeword on the memory-chip, and an ECC-codeword as a page. For example, a 512Mb (64Mx8bit) memory-chip has totally 131072 pages, i.e. W=131072 and a page size organized as 512x8bit (512-Bytes). If we start with a fault-free chip, the chances that the first failing cell will not occur coincident with another failing cell is absolute certainty. However, for the second faulty cell not to occur in the same ECC-codeword as the first one, the probability is 131071÷131072. Similarly for the third random failing cell can't to occur in the same ECC-codeword as 131070÷131072. Multiplication of these three probabilities results in the probability of no alignment between the first, second, and third fault in the same ECC codeword [15-17]. Assume that a memory-chip has **e** single-cell faults on an ECC codeword which the probability of no alignment can be expressed as the following formula

$$P_w\,(\text{no-alignment}) = \prod_{i=1}^{e}\left(\frac{W-i+1}{W}\right) = \frac{1}{W^e}\prod_{i=0}^{e-1}(W-i) = \frac{W!}{W^e(W-e)!} \qquad (4\text{-}1)$$

Part Ⅱ：Comparison and Analysis of sector error rate

Assume that error probability of an error occur on a sector is **p** without ECC, and a sector has **w** ECC-codewords with **e** single-cell faults on the same ECC codeword. Then the sector error probability with ECC instead of the formula (4-1) can be written into (4-2) as the follows.

$$P_{SER} = (1 - P_w) \times p^e \qquad (4\text{-}2)$$

If the sector rate without ECC is ***p=10***$^{-4}$ after one million write and erase cycles [4]. Since the sector size is constant to be 8kB, the number of ECC codewords is inversely proportional to data length, i.e. more and more the number of ECC-words in a sector indicate that sector error rate is becoming more and more low value. For the proposed SEC-S$_{odd}$EC-SBED-DED code, Fig. 4.7 and table 4.8 show the dependence of the sector error rate with ECC and cell area overhead on the ECC code-length in an ECC codeword when the sector error rate without ECC is 10^{-4}. When two random bit errors occur in the same ECC codeword of a sector, the sector is uncorrectable that sector error rate of the proposed code is as $P_{SER2} = (1 - P_w) \times p^2$. As the same way, when three random bit errors occur in the same ECC codeword of a sector, the sector error rate of the SEC Hamming-code is as the formula (4-2), $P_{SER3} = (1 - P_w) \times p^3$. Our ECC code has odd-bit errors correcting capability within a single byte, and a sector has 8kB (8192-bytes) so we rewrite formula (4-2) into (4-3) as the following expression:

$$P_{SER3-} = \{(1 - P_w) - (1 - P_{8192})\} \times p^3 = (P_{8192} - P_w) \times p^3 \qquad (4\text{-}3)$$

Therefore the P$_{SER3-}$ of our ECC-code is lower than that of the existing SEC code as shown in table 4.8. Besides, a sector is not counted as a failure sector until a double bit error occurs in one of ECC words in the sector. Therefore the P$_{SER2}$ of sector error rate with ECC is improved about 6~7 order than that without ECC when code-length is from 528-bytes to 8-bytes.

Table 4.8 Sector error rate with ECC for different code-length when the sector error rate without ECC is p=10^{-4}

Code-length for our (n, k, m) code	The number of ECC-codewords	Redundant-cells Overhead: $\frac{(n-k)}{k}$	Sector Error Rate (P$_{SER2}$)	Sector Error Rate (P$_{SER3,}$ P$_{SER3-}$)
528-Byte, (528,524,8)	16	0.76%	6.25E-10	1.797E-13, 1.793E-13
256-Byte, (256,253,8)	33	1.19%	3.03E-10	8.907E-14, 8.871E-14
128-Byte, (128,125,8)	66	2.40%	1.52E-10	4.500E-14, 4.463E-14
64-Byte, (64,61,8)	132	4.92%	7.58E-11	2.261E-14, 2.225E-14
32-Byte, (32,29,8)	264	10.34%	3.79E-11	1.133E-14, 1.097E-14
16-Byte, (16,14,8)	528	14.28%	1.89E-11	5.675E-15, 5.309E-15
8-Byte, (8,6,8)	1056	25.0%	9.47E-12	2.839E-15, 2.473E-15

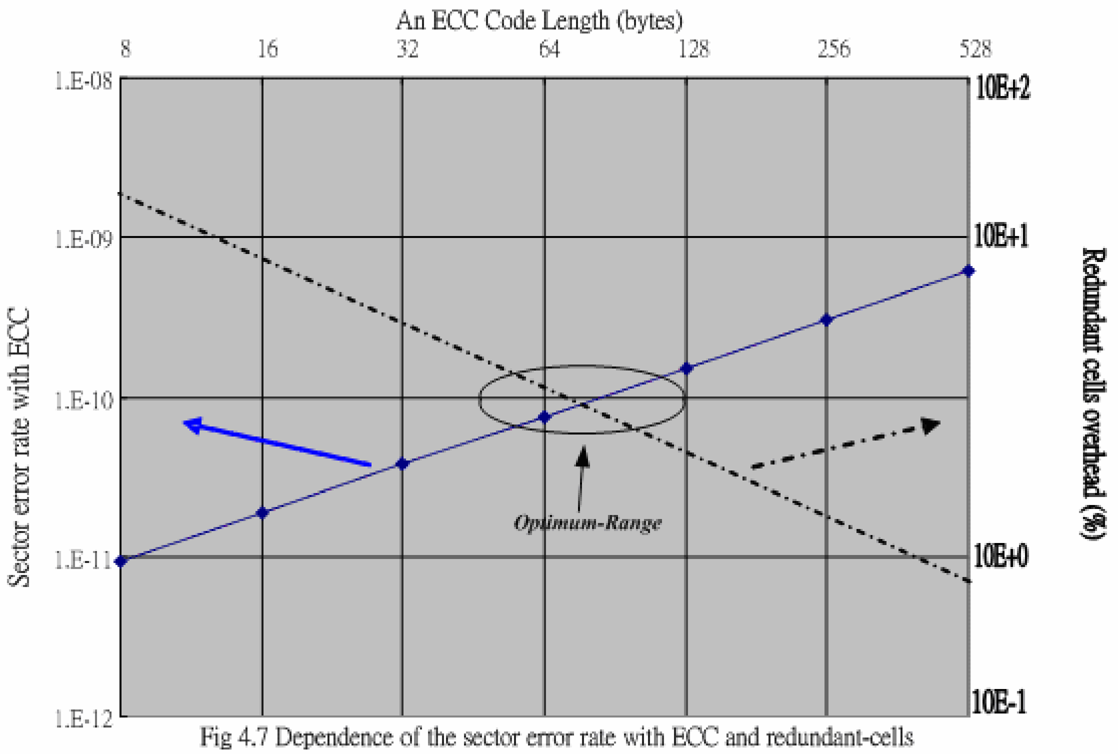


Fig 4.7 Dependence of the sector error rate with ECC and redundant-cells overhead for different ECC code-length

The summary is as follows.

In figure 4.7, we found that an optimum ECC code-length is close to 66-bytes in order to balance the trade-off of the sector error rate and redundant cells overhead. Hence a (66, 63, 8) SEC-S_{odd}EC-SBED-DED code just fits a page size (528-bytes) because of 528 equal (66×8). The (66, 63, 8) SEC-S_{odd}EC-SBED-DED code has a redundant cells overhead of 4.76%, and a sector-error-rate of 7.81E-11. Thus, the reliability target of flash memory endurance can be expressed by the sector error rate which is improved by six orders than that without ECC after one million write and erase cycles.

4.4.3 Performance Comparisons between the existing ECC-codes and the proposed ECC-code

Here we will compare ours ECC-code with the existing code for the programmable hardware implementation of FEC-codec. The compared results are list table 4.9.

Table 4.9 Performance comparisons of different programmable FEC Codec

Reference papers	**[4]**	**[6]**	**[35]**
ECC-Code type ***<note 1>***	**Multi-bit-layer** **SEC Hamming-code,** A fixed (N,K,M)=(522, 512,8) codec	**RS-code,** A fixed (n, k, m)=(72,64,8) decoder	**RS-code,** A programmable codec (n, k, m)=(255,239,8) for n=255~(23+10t), t=1~10, m=8
Correcting/Detecting capability	8 random-bits column-error-correction or only a single-byte error correction for m=8	4 random-bytes error correction (t=4)	8 random-bytes error correction (t=8)
The number of redundant check bit	$R = (\lceil \log_2 k \rceil + 1) \times m$	$R = 2t \times m$	$R = 2t \times m$
Ratio of redundant overhead: (R/K)% ***<note 2>***	2% for (522, 512)	12.5% for (72, 64, 8),t=4	6.7% for (255, 239, 8),t=8
Data transfer wide (bit/clock)	8	8	8
Clock rate (Mhz)	Larger than 100-Mhz	33	10
Throughput Bit rate (Mbit/sec)	Larger than 800	80	80
Total gate-counts of Codec	About 1000	16,000 (only decoder)	about 50,000~70,000 (200,000 transistors)
Data output Latency (clock)	2n	3n	2n+10t+34
Random access after Decoding finish (clock)	No	Yes (after 2n clock)	Yes (after n+10t+34 clock)

Table 4.9 (To be continued)

Reference papers	**[12]**	**[13]**	**[10]**
ECC-Code type	**RS-code,** A programmable (n, k, m)=(255, 239, 8) decoder for n=255~13, t=1~10,m=8	**RS-code,** A programmable (n, k, m)= (255, 239, 8) decoder for $n = 2^m - 1, m \le 8, t \le 8$	**RS-code,** A programmable (n, k, m)= (255, 239, 8) decoder for n=255, any k ,m=8
Correcting/Detecting capability	8 random-bytes error correction (t=8)	8 random-bytes error correction (t=8)	8 random-bytes error correction (t=8)
The number of redundant check bit	$R = 2t \times m$	$R = 2t \times m$	$R = 2t \times m$
Ratio of redundant overhead: (R/K)%	6.7% for (255, 239, 8),t=8	6.7% for (255, 239, 8),t=8	6.7% for (255, 239, 8),t=8
Data transfer wide (bit/clock)	8	1	8
Clock rate (Mhz)	30	48	About 11
Throughput Bit rate (Mbit/sec)	240	48	83
Total gate-counts of Codec	211,296 (only decoder)	43987 (only decoder)	340,500 (only decoder)
Data output latency (clock) ***<note 4>***	$3n-1+(4t^2+t+20)/pk$ ***<note 3>***	3mn+4mt+4m	n(n+1)
Random access after Decoding finish (clock) ***<note 5>***	Yes (after n+10t+34 clock)	Yes (after 3mn+4mt+4m-n clock)	Yes (after n^2 clock)

Note 1: (N, K, M) denotes N-bit code-length, K-bit information-length, M-bit data-wide, and the (n, k, m) denotes n-byte length, k-byte information-length, m-bit data-wide as a byte.

Note 2: The ratio of Redundant overhead equals the number of redundant check-bit divided by the number of information-bit.

Note 3: the larger product value of p and k is the smaller block length.

Note 4: This column of “data output latency” indicates that the decoded operations is finished, and we will read-out all data sequentially from the k-bytes data-buffer RAM, and it usually needs k or n clock cycles to finish the read action.

Note 5: This column of “random-access after decoding finish” indicates that we want to read-out data in random access from the k-bytes data-buffer RAM after the decoding finish, i.e. the decoder must be can compute the error address and error-bits value in order to correct the fail data immediately when read-out the error address.

Table 4.9 (To be continued)

Reference papers	[9]	Ours-1	Ours-2
ECC-Code type	**RS-code** One-Shot RS-decoder, A programmable (n, k, m)=(40,32,8) for n=40~34, k=32, m=8	**The proposed code:** **SEC-S_{odd}EC-SBED-DED,** A programmable Codec (n,k,m)=(4100,4096,8) for n=4100~4, k=4096~2, m=1~8.	**The interleaved code:** **multi-bit-layer** **SEC-S_{odd}EC-SBED-DED,** A programmable Interleaved (*nl*,*kl*,*ml*,m) Codec for *nl*=4100~4, *kl*=4096~2, *ml*=m=1~8.
Correcting/Detecting capability	4 random-bytes error-correction (t=4)	Single bit error correction	8 random-bits column-error-correction or only a single-byte error correction for m=8
The number of redundant check bit	$R = 2t \times m$	$2\times\lceil \log_2 k \rceil + m$	$(2\times\left\lceil \log_2\left(\frac{k}{m_l}\right)\right\rceil + m_l)\times m$
Ratio of redundant overhead: (R/K)%	25% for (40, 32, 8),t=4	0.1% for (4100,4096,8) 1.2% for (256,253,8)	0.6% for (4122,4096,8) 7.6% for (256,238,8)
Data transfer wide (bit/clock)	320bit (40x8)	8	8
Clock rate (Mhz)	22.2 (45ns)	Over 400	Over 400
Throughput Bit rate (Mbit/sec)	7Gb/s for 320bit-inpu 178Mb/s for 8bit-input	Over 3.2Gb/s for m=8, Over 6.4Gb/s for m=16,	Over 3.2Gb/s for m=8, Over 6.4Gb/s for m=16,
Total gate-counts of Codec	24,000 (only decoder)	1511 (without k-bytes RAM)	About 9000 (without k-bytes RAM)
Data output Latency (clock)	2n+(45ns)	2n+1	2n+1
Random access after Decoding finish (clock)	Yes (after n-clock+45ns)	Yes (after n+1 clock)	Yes (after n+1 clock)

From Table 4.9, the proposed ours-1 and ours-2 is compared to the RS-code type of Codec that our proposed codec gains 8~40 times faster in clock rate, a more wide programmable range for (n, k, m) parameters, area cost-down about 12~170 times, a high throughput bit rate or data rate of 13~67 times for m=8, 26~134 times for m=16 due to the data-wide m and code length n are almost irrelative to clock speed. Furthermore, data output latency is reduced about (10t+34)~2n clock, and only need n+1 decoding clock to finish the computations of

error address and error-bit value. The ours-1 only has a poor SEC-S$_{odd}$EC error correcting capabilities but its overhead also only pay a very little redundancy about 1.2%.
The ours-2 has a good error correcting capability to multiple random-bits error, and only has an acceptable redundancy overhead about 7.6%.

In additional, advantages and disadvantages between the existing ECC codes and the proposed ECC codes are listed in Table 4.10. We can also clearly know these ECC-codes performance from table 4.10.

Table 4.10 Advantages and disadvantages between the existing ECC codes and the proposed ECC codes

ECC-Code type	**Multi-bit-layer SEC Hamming-code**	**Bi-directional Product Code**	**RS or BCH Code**
Correcting/Detecting capability	Multi-SEC [4]	SEC-DED [1],[19]	Multiple-bits correcting and detecting by t. [9-14]
The number of redundant check bit	$R = (\lceil \log_2 k \rceil + 1) \times m$	$R = k + m + 1$	$R = 2t \times m$
Ratio of redundant overhead: (R/K)% (When k=16~1024-bytes, m=8)	**Medium--Low** (31.3~1.1%)	**Medium** (19.5~12.6%)	**Low** (12.5~0.2% for t=1) **High--Low** (100~1.6% for t=8)
Circuit complexity of Programmable (n, k, m)	**<Medium>** About several thousand gate-counts	**<Low>** About several thousand gate-counts	**<High>** Several ten-thousand gate-counts
A programmable (n, k, m) range	**Large**	**Larger**	**Small**
Limitations of a (n, k, m) coding parameter (r=redundant check-bit)	Just for m=1, and k is a regular variable $n=2^r$, $k=2^r$-r-1, n-k=r+1,	k and m are any variable, and n=k+m.	n or m is fixed by the formulas $n=2^m$-1 and n-k=2t, n and k is a regular variable m=a symbol size in bit.
Operating clock-speed	**High--Medium**	**High**	**Medium--Low**
Decoding-Latency	**Medium**	**Medium--Low**	**High--Medium**

Table 4.10 (To be continued)

ECC-Code type	**DEC/TEC Code**	**SbEC-DbED Code**	**The proposed Code (Ours-1 and Ours-2)**
Correcting/Detecting capability	DEC-TED [2], TEC-QED [7].	S3EC-SBED [28], SEC-SBED [29], DEC-SBED [30].	Ours1: SEC-S_{odd}EC-SBED-DED Or Ours2: Multi-bit-layer ECC
The number of redundant check bit	$R = 3m + 1$ [2] $R = m \times (\log_2 k + 2) + k$ [7]	$R = \frac{1}{2}(3m + \log_2(k+2))$ [28]	$R = 2 \times \lceil \log_2 k \rceil + m$ or $(2 \times \left\lceil \log_2\left(\frac{k}{m_l}\right) \right\rceil + m_l) \times m$
Ratio of redundant overhead: (R/K)% (When k=16~1024 bytes, m=8)	**High--Medium** (50~13.7%) for [7]	**Low** (11~0.4%) for [28]	**Low** (12.5~0.34% for ours-1) **High--Low** (60~2.1% for ours-2)
Circuit complexity of Programmable (n, k, m)	**<High>** About several thousand gate-counts	**<High>** About several thousand gate-counts	**<Low>** Less than two thousand gate-counts
A programmable (n, k, m) range	**Medium**	**Small**	**Larger**

Limitations of a (n, k, m) coding parameter
(r=redundant check-bit)
	K=m^2,m=odd integer [2]
	n=2^r, k=2^r-r-1, [7]
		Any $(b \times b)$ matrix,
		$t < b, N = b \times (2^{R-b} - 1) + R$
			Any k and m, and
			n=k+r

Operating clock-speed	**Medium**	**Medium--Low**	**High**
Decoding-Latency	**Medium**	**High--Medium**	**Medium--Low**

Chapter 5
Hardware and Software Simulation Results

This section mainly describes the simulation results of the both proposed FEC-Codec hardware and software design. ECC-capabilities of our proposed SEC-S$_{\text{odd}}$EC-SBED-DED code will be simulated by using the foregoing C-language program in the chapter 4.3. The programmable hardware circuits of the proposed FEC-Codec for (n, k, m)=(4100~4, 4096~2, 8~1) are also simulated to show timing-diagram of encoding-decoding behavior.

5.1 Hardware Simulation Results

Here we will show the operation behavior and waveform simulations results of the proposed programmable (n, k, m) FEC-Codec hardware.

Memory-control-unit writes data to external-memory through ours FEC-Codec, A write flow-chart is shown in Fig. 5.1. Memory-control-unit reads data from external-memory through ours FEC-Codec, A read flow-chart is described in Fig. 5.2. Therefore we can know the control flow of the proposed programmable FEC-Codec that it’s how to access data from external memory.

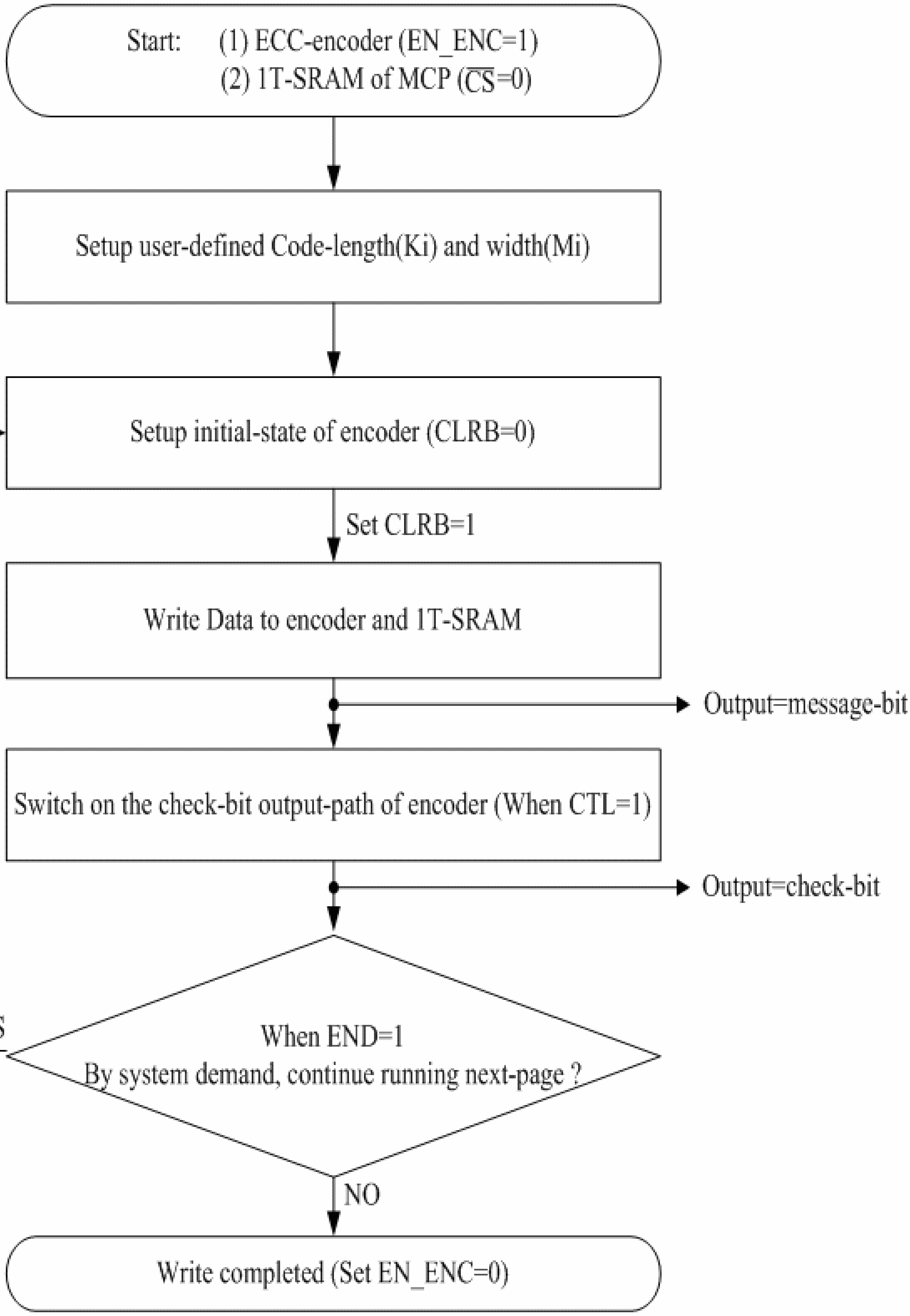

Start: (1) ECC-encoder (EN_ENC=1)
(2) 1T-SRAM of MCP ($\overline{CS}$=0)
Setup user-defined Code-length(Ki) and width(Mi)
Setup initial-state of encoder (CLRB=0)
Set CLRB=1
Write Data to encoder and 1T-SRAM
Output=message-bit
Switch on the check-bit output-path of encoder (When CTL=1)
Output=check-bit
YES
When END=1
By system demand, continue running next-page ?
NO
Write completed (Set EN_ENC=0)

Fig. 5.1 Write flow-chart of memory through ours FEC-Codec.

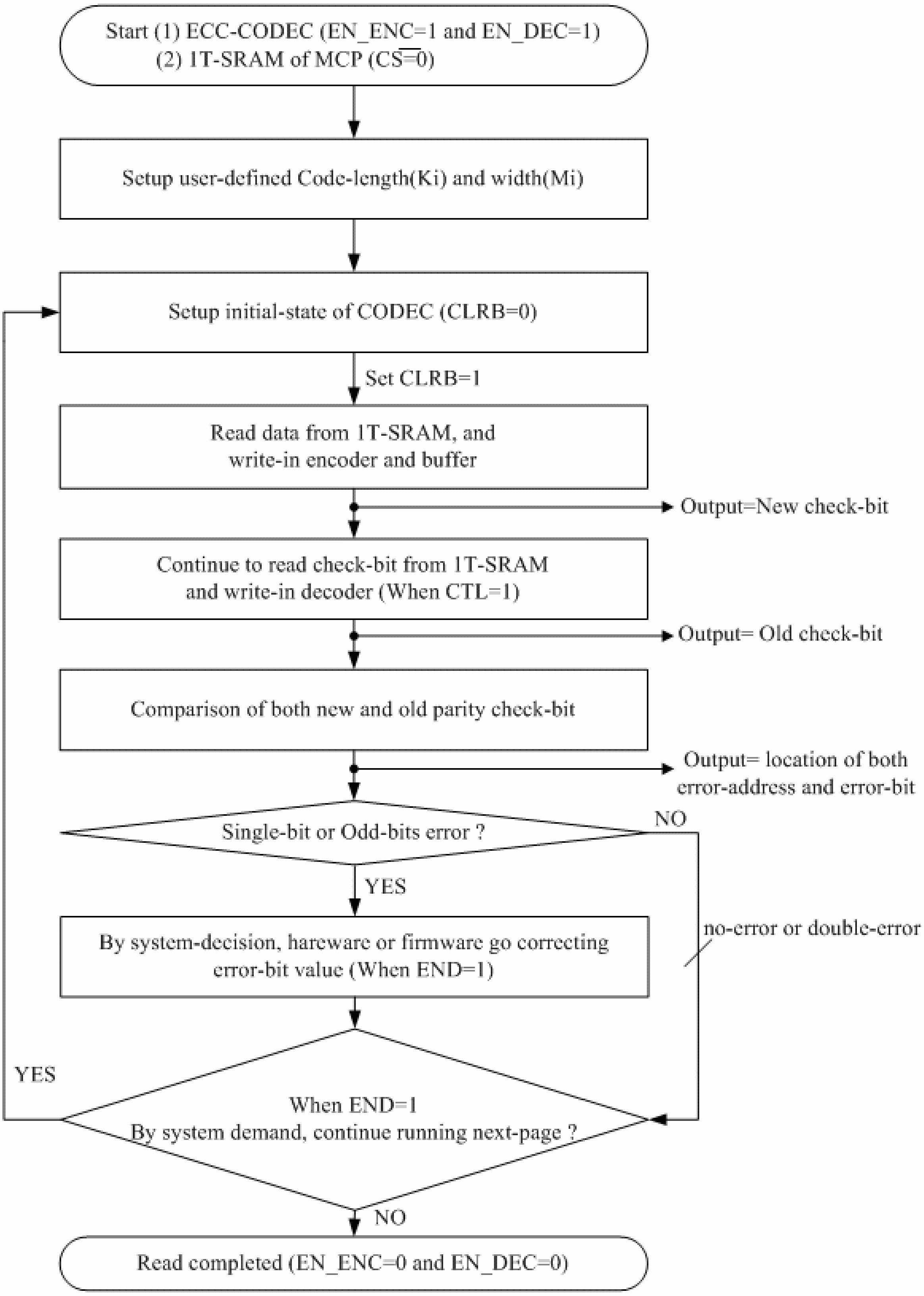

Start (1) ECC-CODEC (EN_ENC=1 and EN_DEC=1)
(2) 1T-SRAM of MCP ($\overline{CS}$=0)
Setup user-defined Code-length(Ki) and width(Mi)
Setup initial-state of CODEC (CLRB=0)
Set CLRB=1
Read data from 1T-SRAM, and
write-in encoder and buffer
Output=New check-bit
Continue to read check-bit from 1T-SRAM
and write-in decoder (When CTL=1)
Output= Old check-bit
Comparison of both new and old parity check-bit
Output= location of both
error-address and error-bit
Single-bit or Odd-bits error ?
NO
YES
By system-decision, hareware or firmware go correcting
error-bit value (When END=1)
no-error or double-error
YES
When END=1
By system demand, continue running next-page ?
NO
Read completed (EN_ENC=0 and EN_DEC=0)

Fig. 5.2 Read flow-chart of memory through ours FEC-Codec.

We simulate the programmable FEC-Codec hardware when we set the data-length equal to 256 (Ki=100\h) and data-wide equal to 8-bit (Mi=7\h), i.e. we set the FEC-Codec which is a programmable (n, k, m)=(259, 256, 8) ECC-encoder-decoder, and the simulation waveforms are shown in Fig. 5.3, 5.4, 5.5, 5.6.

Figure 5.3 shows the encoding timing waveforms, and figure 5.4 shows decoding timing waveforms when a single error is occurred. Figure 5.5 shows decoding timing waveforms when no errors. Figure 5.6 shows decoding timing waveforms when double errors are occurred. This FEC-hardware only needs 260 clock-cycle times at most to compute the error-address and error-bit value, and it has a low decoded latency, hence a high throughput is gained to access the consecutive pages.

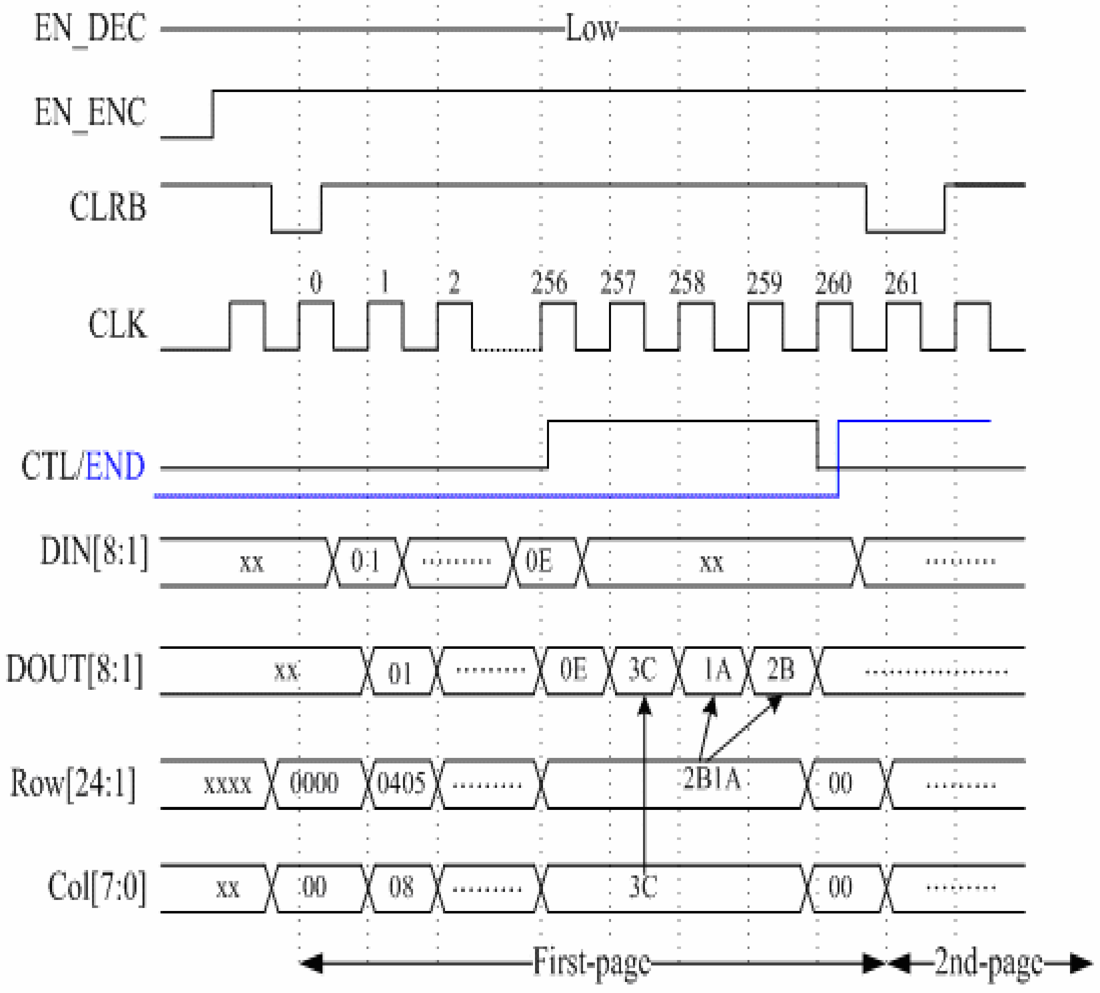


Fig. 5.3 Encoding timing diagram of FEC-codec when control-parameters (n, k, m)=(259, 256, 8)

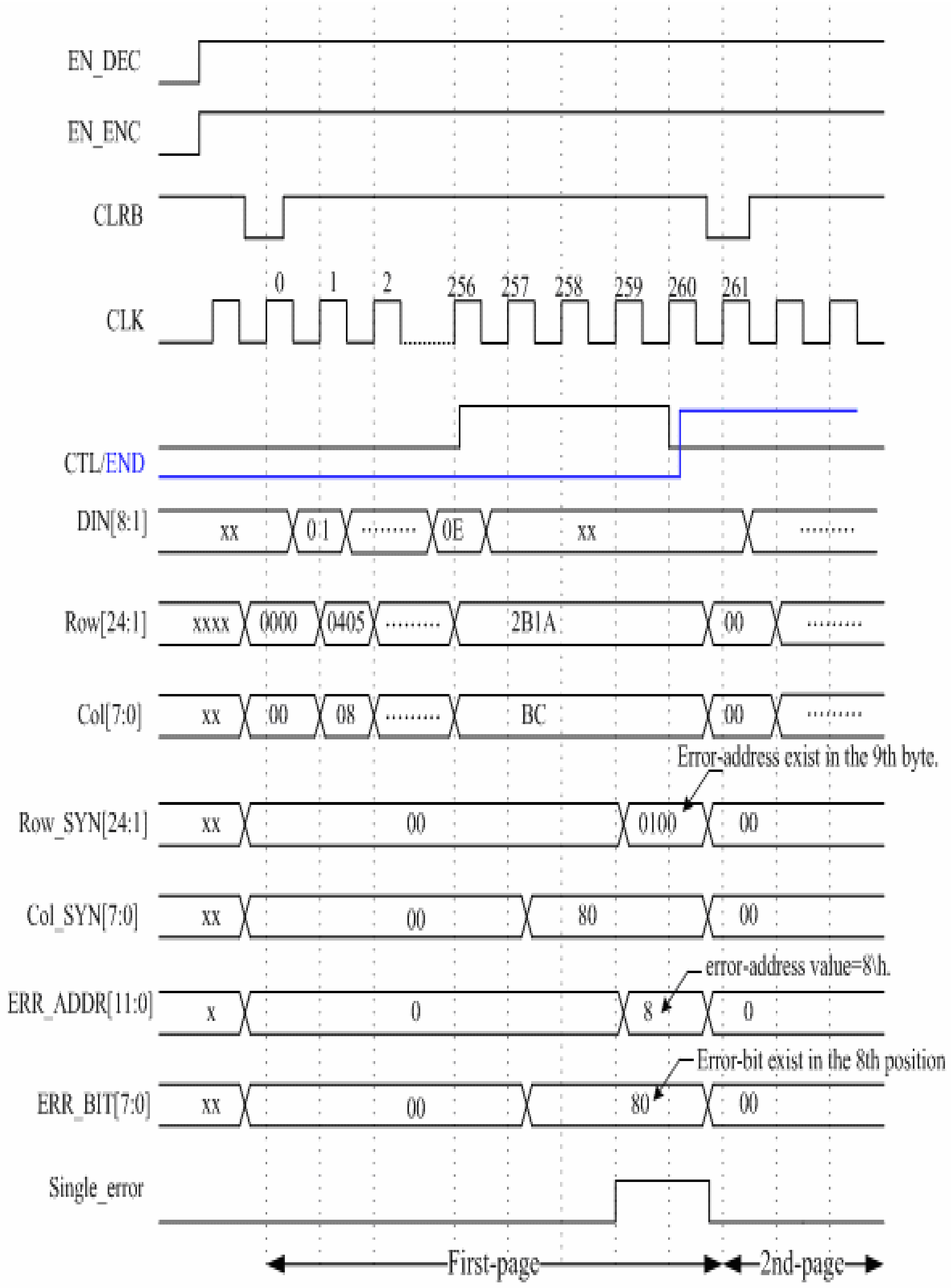


Fig. 5.4 Decoding timing diagram of FEC-Codec when the occurrence of a single-bit error

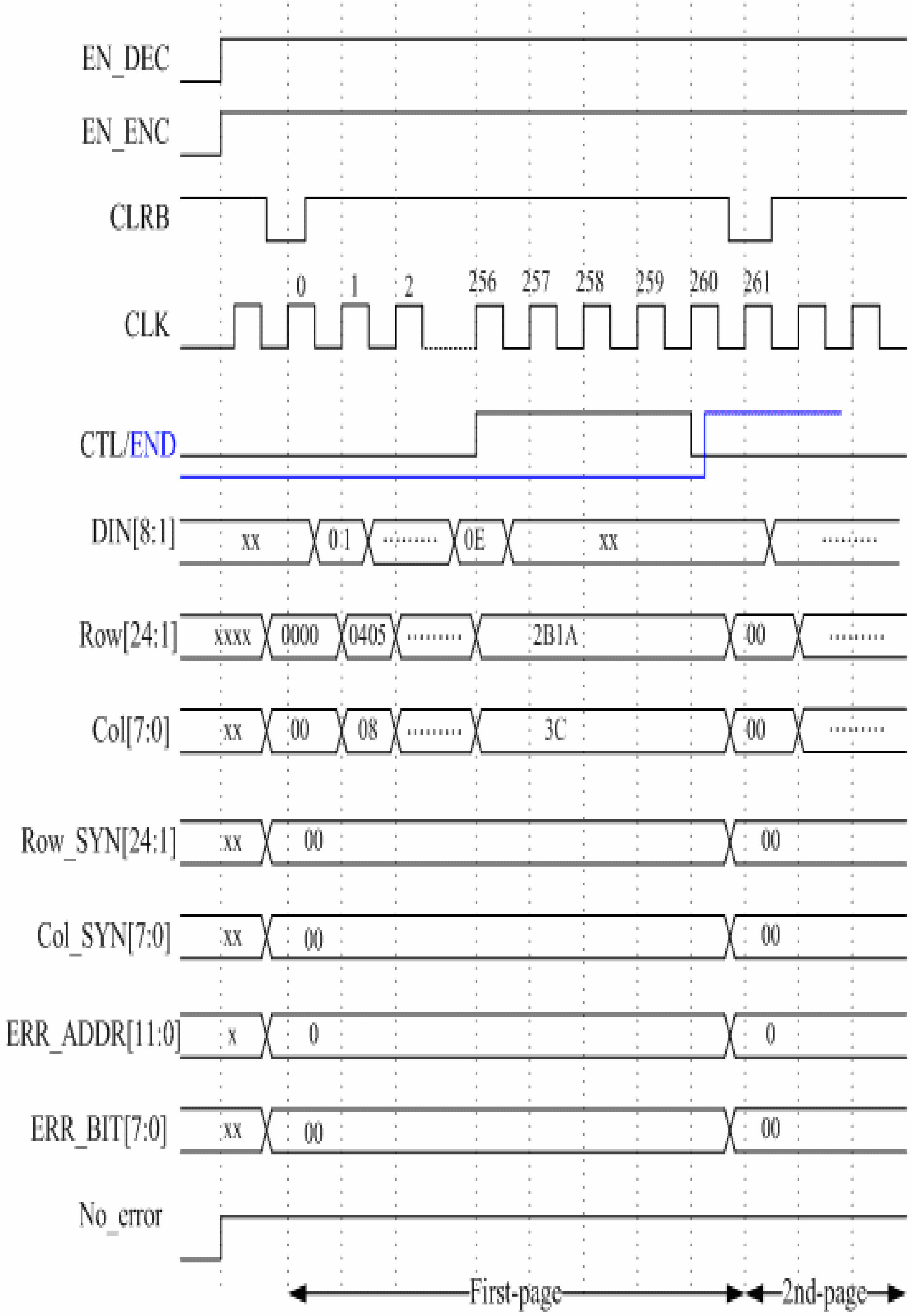


Fig. 5.5 Decoding timing diagram of FEC-Codec when the occurrence of no-error

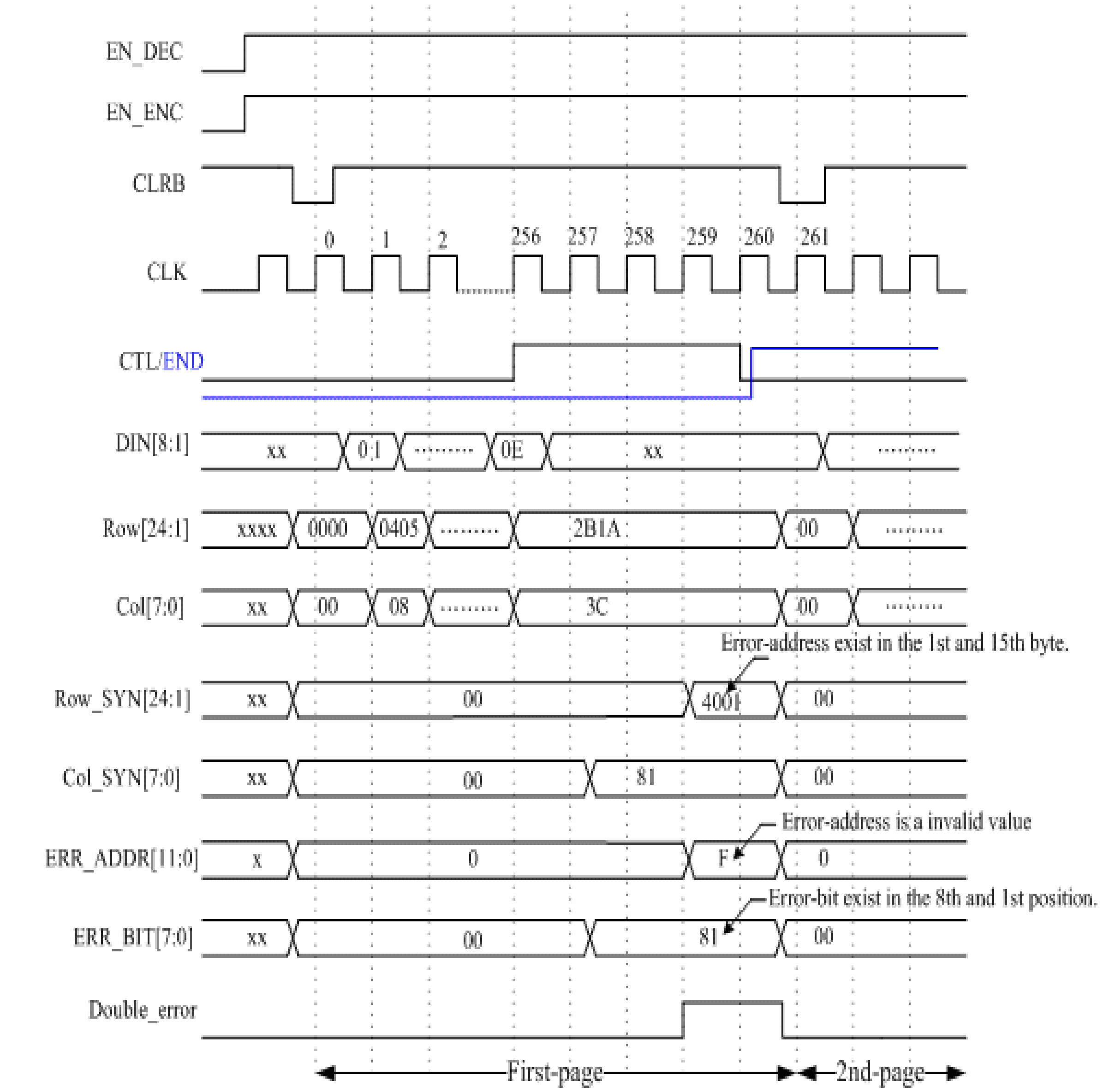


Fig. 5.6 Decoding timing diagram of FEC-Codec when the occurrence of double error

5.2 Software Simulation Results

We use the above C-language application program to simulate the error detection and correction capabilities of SEC-S$_{odd}$EC-SBED-DED code that execute 10^6 pattern-numbers of k information codewords (also called data codewords or message codewords).

Some symbols definitions of SEC-S$_{odd}$EC-SBED-DED code as follows:

(1) The (n, k, m) parameters: n=code-length in bytes, k=information or data or message-length in bytes, m= a byte wide or the number of a byte, or a codeword.

(2) The (N, K, M) parameters: N=$n \times m$ =code-length in bits, K=$k \times m$ =information or data or

message-length in bits, M=m= a data I/O wide, a byte wide or the number of a byte, or a codeword.

(3) r is parity check length in bytes, and R=$r \times m$ = parity check length in bits.

Part I: *Error-correcting-detecting capabilities simulation results by using multiples random bit errors happening on each SEC-S_{odd}EC-SBED-DED (n, k, m) code.*

The software program parameters is set to select noise-channel (d) with 1~8 random bit errors on each (n, k, m) code. In table 5.1 and table 5.3, we know the proposed SEC-S_{odd}EC-SBED-DED code has an 80~90% detecting error capability for random even-bit errors, and a 40~81% detecting error capability for random odd-bit errors.

Furthermore, the software program parameters is set to select noise-channel (e) with 1~8 random bit errors on a single codeword of each (n, k, m) code. In table 5.2 and table 5.4, we know the proposed SEC-S_{odd}EC-SBED-DED code has a 100% correcting error capability for a single odd-bit errors byte, and a 93~99% detecting error capability for a single even-bit errors byte. In table 5.1 and 5.5, we found that (78, 64, 8) and (76, 64, 4) SEC-S_{odd}EC-SBED-DED codes have the same information length, but have some different features that the (76, 64, 8) code has a better detecting error capability about 99.5~100% for random even-bit errors, and a poor detecting error capability about 40~66% for random odd-bit errors than (78, 64, 8) code.

In table 5.1 and table 5.2, the software program parameters is set to select noise-channel (e) with 1~8 random bit errors on each (N, K, M) code.

Table 5.1 Simulation results for multiple random-bit errors when the SEC-S_{odd}EC-SBED-DED (N, K, M)=(78, 64, 8) code.

Random-bit errors-numbers	**1bit**	**2bit**	**3bit**	**4bit**	**5bit**	**6bit**	**7bit**	**8bit**
Mis-correction	0%	17.53% (17.53%)	42.18% (16.13%)	19.83% (6.98%)	28.47% (6.01%)	18.16% (3.48%)	21.44% (3.05%)	16.14% (2.09%)
Correction	100%	0%	0%	0%	0%	0%	0%	0%
Detectable	0%	81.34% (81.34%)	57.60% (83.65%)	79.95% (92.8%)	71.42% (93.08%)	81.75% (96.43%)	78.51% (96.9%)	83.82% (97.87%)
Un-detectable	0%	1.13%	0.23%	0.22%	0.11%	0.09%	0.05%	0.04%

Note that () value indicates simulation results only for the both miscorrection and detection of SEC-SBED-DED (N, K, M)=(78, 64, 8) code.

Table 5.2 Simulation results for multiple random-bit errors on a single-byte when the SEC-S$_{odd}$EC-SBED-DED (N, K, M)= (78, 64, 8) code.

Random-bit errors-numbers	**1bit**	**2bit**	**3bit**	**4bit**	**5bit**	**6bit**	**7bit**	**8bit**
Mis-correction	0%	0%	0%	0%	0%	0%	0%	0%
Correction	100%	0%	100%	0%	100%	0%	100%	0%
Detectable	0%	93.62%	0%	94.8%	0%	96.17%	0%	96.71%
Un-detectable	0%	6.37%	0%	5.2%	0%	3.83%	0%	3.29%

Table 5.3 Simulation results for multiple random-bit errors when the SEC-S$_{odd}$EC-SBED-DED (N, K, M)=(144, 128, 8) code.

Random-bit errors-numbers	**1bit**	**2bit**	**3bit**	**4bit**	**5bit**	**6bit**	**7bit**	**8bit**
Mis-correction	0%	9.23% (9.23%)	60.48% (21.1%)	13.32% (4.58%)	45.1% (8.4%)	14.69% (2.66%)	34.8% (4.35%)	14.7% (1.77%)
Correction	100%	0%	0%	0%	0%	0%	0%	0%
Detectable	0%	90.54% (90.54%)	39.47% (78.85%)	86.54% (95.28%)	54.86% (91.56%)	85.25% (97.28%)	65.18% (95.63%)	85.27% (98.2%)
Un-detectable	0%	0.23%	0.04%	0.14%	0.04%	0.06%	0.02%	0.03%

Note that () value indicates simulation results only for the mis-correction of SEC-SBED-DED (N, K, M)= (144, 128, 8) code.

Table 5.4 Simulation results for multiple random-bit errors on a single-byte when the SEC-S$_{odd}$EC-SBED-DED (N, K, M)= (144, 128, 8) code.

Random-bit errors-numbers	**1bit**	**2bit**	**3bit**	**4bit**	**5bit**	**6bit**	**7bit**	**8bit**
Mis-correction	0%	0%	0%	0%	0%	0%	0%	0%
Correction	100%	0%	100%	0%	100%	0%	100%	0%
Detectable	0%	97.79%	0%	97.04%	0%	98.02%	0%	98.37%
Un-detectable	0%	2.21%	0%	2.96%	0%	1.98%	0%	1.63%

Table 5.5 Simulation results for multiple random-bit errors when the SEC-S$_{odd}$EC-SBED-DED (N, K, M)= (76, 64, 4) code.

Random-bit errors-numbers	**1bit**	**2bit**	**3bit**	**4bit**	**5bit**	**6bit**	**7bit**	**8bit**
Mis-correction	0%	0%	59.97%	0%	44.19%	0%	33.68%	0%
Correction	100%	0%	0%	0%	0%	0%	0%	0%
Detectable	0%	100%	40.03%	99.5%	55.81%	99.68%	66.32%	99.76%
Un-detectable	0%	0%	0.09%	0.5%	0%	0.32%	0%	0.3%

Part II: *Error decoding probability simulation results by using a bit error probability p happening on consecutive SEC-S$_{odd}$EC-SBED-DED (n, k, m) codewords.*

From the references [3],[33],[34], we can define error probability and bit error rate as below.

(1) The error probability P_e without error correction for a (N, K) block code is define as the following formula:

$P_e = 1-(1-p)^K$ .... (5-1), where p= single cell error probability, and k indicates k-bits data length or K symbols data length.

(2) The expected number of decoded errors with e errors in a (n, k) SEC-DED block-code is as the formula:

E[X(e)]=0 for e=0,1; E[X(e)]=$\frac{(k-e)}{n}(e+1)+\frac{(n-k)}{n}e+\frac{e}{n}(e-1)$ for e>1 ....(5-2)

We can rewrite formula 5-2 into formula 5-3:

E[X(e)]= $\frac{k+(n-2)\times e}{2} \geq e$ for $\frac{k}{2} \geq e > 1$................(5-3)

(3) The probability that the number i of errors exceed the (n, k) block-code recovery capability t can be expressed as an erroneous decoding probability is upper bounded by the formula: P$_{ECC}$ ≦ $\sum_{i=t+1}^{n}\binom{n}{i}p^i(1-p)^{n-i}$ .... (5-4), where the expression $\binom{n}{i}=\frac{n!}{i!(n-i)!}$.

Therefore the system BER (Bit-Error-Rate) of a t-error-correcting (n, k) block-code after decoding can be written into the formula 5-5:

BER$_{out}$ $=\frac{E[X(e)]}{k}\{\sum_{e=t+1}^{n}\binom{n}{e}\times(BER_{in}{}^{i})\times(1-BER_{in})^{n-e}\}$ .... (5-5)

So a SEC (n, k) code has the formula 5-6:

BER$_{out}$ $=\frac{k+(n-2)e}{2k}\{\sum_{e=2}^{n}\binom{n}{e}\times(BER_{in}{}^{i})\times(1-BER_{in})^{n-e}\}$ .... (5-6)

The BER without error correction for a (N, K) block code is the formula 5-7:

BER$_{out}$ $=\frac{e}{k}\{1-(1-BER_{in})^k\}$ .... (5-7), where e=the number of error-bits,

BER$_{in}$=input bit error rate or bit error probability before encoding-decoding procedure.

An example of different (n, k)=(76, 64) 4-ary block-codes for bit-error-rate after decoding are shown in Fig. 5.7, where “Pne” is the decoded BER without ECC by the formula 5-7, “SSC” is a Single Symbol-error-correcting Code which only can correct a single q-ary symbol, where q=4, and n=19, k=16. “DSC” is a Double Symbol-error-correcting Code which only can correct two q-ary symbol, where q=4, and n=21, k=16. We use formula 5-5, and 5-6 to compute “DSC”, and “SSC” respectively. The blue curve is a theoretical value for a SEC block-code, n=76, k=64 by formula 5-6. Ours curve is drawn by the software program parameters which is set to select noise-channel (b), and n=76, k=64, m=4 for random bit error on a BSC (Binary Symmetric Channel) with a bit error transition probability p=10^{-1}~10^{-7}. Ours (76, 64,) curve has a lower BER than a SEC (76, 64) curve.

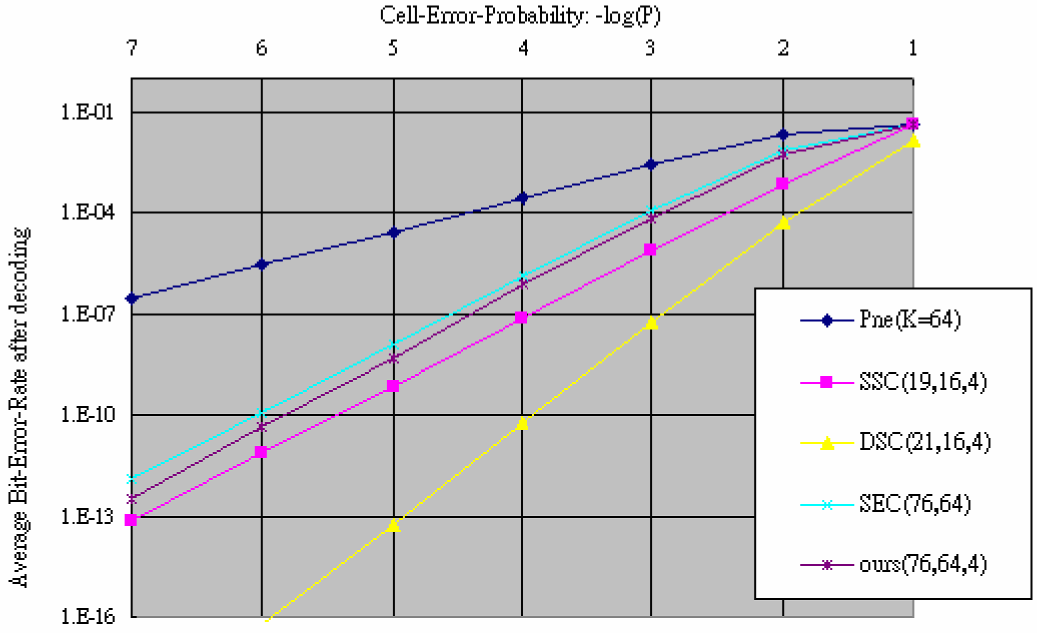


Fig. 5.7 Error-correcting capability comparison for different type of ECC-codes with K=64

The decoding Bit-error-rate of the proposed SEC-S_{odd}EC-SBED-DED codes for different data-length from 4-byte to 4096bytes is shown in P1~P4 curve of figure 5.8 when a bit error transition probability p=10^{-3}~10^{-6}. Furthermore, the number of parity check bit divided by the number of information-bit equals redundancy-rate as shown in the purple curve of figure 5.8, the rate is about 0.1%~27.27%.

Table 5.6 shows the simulations value and redundancy rate for information length=4096~4 as the figure 5.8.

Table 5.6 Simulations value and redundancy rate for different information length in figure 5.8

Data-length	4096	2048	1024	512	256	64	16	4
$p=10^{-3}$	1.43E-3	1.37E-3	1.29E-3	1.21E-3	1.14E-3	4.71E-4	1.39E-4	4.19E-5
$p=10^{-4}$	1.27E-4	1.05E-4	6.83E-5	3.77E-5	1.97E-5	5.25E-6	1.45E-6	5.08E-7
$p=10^{-5}$	3.14E-6	1.52E-6	9.85E-7	2.93E-7	2.05E-7	6.97E-8	2.24E-8	3.91E-9
$p=10^{-6}$	3.62E-8	2.5E-8	1.95E-8	3.7E-9	1.40E-9	7.00E-10	2.30E-10	3.50E-11
Red. Rate (N-K)/K	0.001	0.0018	0.0034	0.0063	0.0116	0.0376	0.1111	0.2727

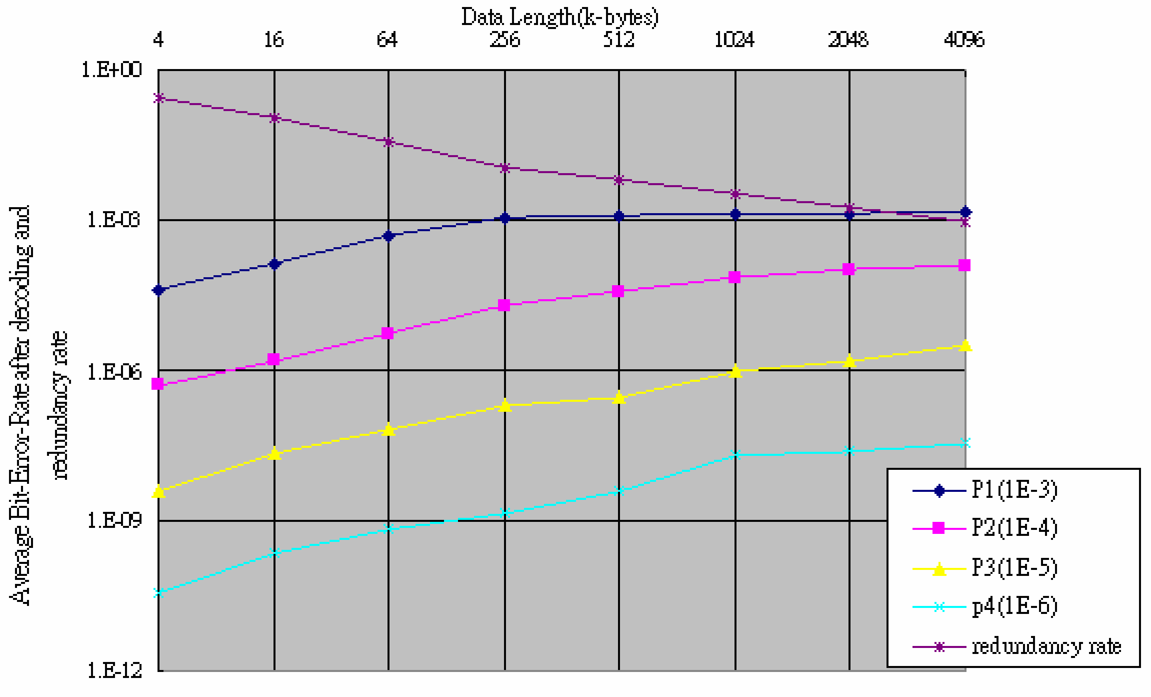


Fig. 5.8 Error-correcting capability comparison for different data-length and cell error probability

The software program parameters is set to select noise-channel (c) for 2~7 burst-bit errors on a BSC (Binary Symmetric Channel) with a bit error transition probability $p=10^{-2}\sim10^{-8}$.

An example of (N,K,M)=(78,64,8) SEC-S_{odd}EC-SBED-DED code is shown as the following table 5.7.

Table 5.7 Simulation results for burst-bit errors when the SEC-S_{odd}EC-SBED-DED (N, K, M)= (78, 64, 8) code.

Burst-bit errors	2-bit	3-bit	4-bit	5-bit	6-bit	7-bit
$p=10^{-2}$ with ECC	1.94E-2	2.03E-2	3.83E-2	3.84E-2	5.67E-2	6.05E-2
$p=10^{-4}$ with ECC	1.98E-4	9.90E-5	3.91E-4	3.02E-4	5.97E-4	5.85E-4
$p=10^{-6}$ with ECC	1.88E-6	8.10E-7	3.84E-6	3.16E-6	5.96E-6	5.81E-6
$p=10^{-8}$ with ECC	1.91E-8	8.90E-9	3.90E-8	3.05E-8	5.97E-8	5.80E-8
$p=10^{-8}$ without ECC	2E-8	3E-8	4E-8	5E-8	6E-8	7E-8

The decoding bit error rate of burst-errors is shown that the proposed code has a little error correcting capability and odd-bit error rate is less than even-bit error rate, burst-error equal to 3-bit especially. In general, SEC-DED code has a worst decoding bit error rate under burst error conditions, because of un-detecting error or mis-correction, hence decoding bit error rate of the SEC-DED code is larger or equal to that without ECC [32]. For burst-error rate, the SEC-S_{odd}EC-SBED-DED code has a stable decoding bit error-rate than the SEC-DED codes as shown in the last two-column of the table 5.7.

Chapter 6
Conclusions

Because most of memory-chips have a high-speed clock rate, such as a 133Mhz clock-rate for mobile SDRAM, 200Mhz for DDR FCRAM, 400Mhz for RDRAM, 100Mhz for page-mode flash. Furthermore, most of memory-chips have different page or burst-length such as 8bytes~512bytes burst-length for mobile-SDRAM/1T-sram, and specific 264bytes, 528bytes, 2112bytes page-length for NAND-flash, and different data I/O wide such as 1, 4, 8, 16, 18-bits….etc. Hence we propose a class of systematic ECC codes called the (n, k, m) SEC-S_{odd}EC-SBED-DED code which is capable of correcting a random single-error or odd-bit error occurring within a single byte, detecting double-bit error or a single-byte error, and another extended code of SEC-S_{odd}EC-SBED-DED codes is an interleaved code which called the (n_l, k_l, m_l, m) Multi-Bit-Layer SEC-S_{odd}EC-SBED-DED code which is capable of correcting Single-Byte Error Correction—Odd-bytes Error Correction within m_l consecutive bytes— m_l consecutive Bytes Error Detection—random Double bits Error Detection on each data-I/O. The number of parity-check-bit overhead of the proposed interleaved code is smaller than DEC-TED [2] and TEC-QED [7] ; the error-correcting-detecting capability is better than DEC [2], TEC [7], SbEC [28-30], and other multi-bit-layer SEC-DED code [1], [4], [19]; the programmable (n_l, k_l, m_l, m) range is very large than the existing ECC-codes. Furthermore, the hardware implementation is more compact, flexible and faster than the existing ECC codec.

Finally, we developed a new programmable SEC-S_{odd}EC-SBED-DED encoder-decoder, and Multi-Bit-Layer SEC-S_{odd}EC-SBED-DED generating methods suitable for different type of memory-chips due to the better features, such as high speed clock-rate and high throughput data-rate (over 400Mhz, and 3.2Gbit/sec), cost-effective (less than two-thousand logic gated counts), a shorter decoding latency (n+1 clock-cycles), low redundancy, and a very large programmable range for code-length and data-wide . A compact, modular, and parallel pipelined hardware design makes the FEC-codec keeping a high-speed operation over 400Mhz, and is almost regardless of the code-length and data-wide.

Reference

[1] T. Mano, J. YAMADA, J. Inoue, and S. Nakajima, "Circuit Techniques for a VLSI memory," IEEE J. Solid-State Circuits, vol. SC-18, pp. 463-470, Oct. 1983.

[2] P. Mazumder, "Design of a Fault-Tolerant Three-Dimensional Dynamic Random-Access Memory with On-Chip Error-Correcting Circuit," IEEE Trans. on Computers, vol. 42, pp. 1453-1468, Dec. 1993.

[3] S. Gregori, A. Cabrini, O. Khouri, G. Torelli, "On-Chip Error Correcting Techniques for New Generation Flash memories," IEEE Proceedings, vol. 91, pp. 602-616, Apr. 2003.

[4] T. Tanzawa, T. Tanaka, K. Takeuchi, R. Shirota, "A Compact On-Chip ECC for Low Cost Flash Memories," IEEE J. Solid-State Circuits, vol. 32, pp. 662-669, May. 1997.

[5] P. Mazumder, "An On-Chip ECC Circuit for Correcting Soft Errors in DRAM's with Trench Capacitors," IEEE J. Solid-State Circuits, vol. 27, pp. 1623-1633, May. 1992.

[6] G. Cardarilli, M. Dizenzo, P. Pistilli, "A High Speed Reed-Solomon Encoder-Decoder for Fault Tolerant Solid-State Disks," IEEE workshop on defect and fault tolerance in VLSI systems. pp. 33-40, Oct. 1993.

[7] F. Alzahrani, T. Chen "On-Chip TEC-QED ECC for Ultra-Large, Single-Chip Memory Systems," IEEE VLSI in Computers and Processors, pp. 132-137, Dec. 1994.

[8] W. Gao and S. Simmons, "A Study On The VLSI Implementation of ECC for Embedded DRAM," IEEE Electrical and Computer Engineering, pp. 203-206, May. 2003.

[9] Y. Katayama, and S. Morioka, "One-Shot Reed-Solomon Decoding for High-Performance Dependable Systems," IEEE Dependable Systems and Networks, pp. 390-399, Jun. 2000.

[10] Yousf R. Shayan, and Tho Le-Ngoc, "A Cellular Structure for a Versatile Reed-Solomon Decoder," IEEE Trans. on Computers, vol. 46, pp. 80-85, Jan. 1997.

[11] M. K. Song, M. H. Kong, H. S. Won, "An Area-Efficient Versatile Reed-Solomon Decoder for ADSL," IEEE Circuits and Systems, vol. 1, pp. 517-520, Jun. 1999.

[12] M. Song, M. E. Kim, H. S. Won, etc, "Architecture for Decoding Adaptive Reed-Solomon Codes with Variable Block Length," IEEE Consumer Electronics, Vol. 48, pp. 631-637, Aug. 2002.

[13] J. Chang, C. Shung, "A High Speed Reed-Solomon Codec Chip using Look-forward Architecture," IEEE Trans. on Computers, pp. 212-217, 1994.

[14] Y. R. Shayan, and T. Le-Ngoc, "A Versatile Time-Domain Reed-Solomon Decoder," IEEE J. Selected Areas in Comm., pp. 1535-1542, Oct. 1990.

[15] J. A. Fifield, and C. H. Stapper, “High-Speed On-Chip ECC for Synergistic Fault Tolerant Memory Chips,” IEEE J. Solid-State Circuits, vol. 26, pp. 1449-1452, Oct. 1991.

[16] B. Polianskikh and Z. Zilic, “Design and Implementation of Error Detection and Correction Circuitry for Multilevel Memory Protection,” IEEE Proceedings, Multiple-Valued Logic, Symposium, pp. 89-95, May. 2002.

[17] D. Rossi, C. Metra, and B. Ricco “Fast and Compact Error Correcting Scheme for Reliable Multilevel Flash Memories,” IEEE On-Line Testing workshop, pp. 221-225, Jul. 2002.

[18] S. Gregori, O. Khouri, etc, “An Error Control Code Scheme for Multilevel Flash Memories,” IEEE Memory Technology, Design and Testing Workshops, pp.45-49, Aug. 2001.

[19] J. YAMADA, “Selector-Line Merged Built-In ECC Technique for DRAM’s,” IEEE J. Solid-State Circuits, vol. sc-22, pp. 868-873, Oct. 1987.

[20] R. J. McPartland, “Circuit Simulations of alpha-particle-induced soft error in MOS dynamic RAM’s,” IEEE J. Solid-State Circuits, vol. sc-16, pp. 31-34, Feb. 1981.

[21] G. A. Sai-Halasz, etc, “Alpha-particle-induced soft error rate in VLSI circuits,” IEEE J. Solid-State Circuits, vol. sc-17, pp. 355-361, Apr. 1982.

[22] J. Chao, etc, “Cost Savings with NAND shadowing reference Design,” application-report from Toshiba, 2002.

[23] P. Gillingham, etc, “A 768k Embedded DRAM for 1.244Gb/s ATM Switch in a 0.8um Logic Process,” IEEE ISSCC, pp. 262-263, 1996.

[24] “A DRAM reliability report,” PSC corp. , 2002.

[25] W. Leung, etc, “The Ideal SoC Memory:1T-SRAM,” IEEE ASIC/SOC conference., pp. 32-36, Sep. 2000.

[26] A. D. Fogle, etc, “Flash Memory under Cosmic & Alpha Irradiation,” IEEE Trans. Device and Materials Reliability, pp. 371-276, Sep. 2004.

[27] “Implementing MLC NAND Flash for Cost-effective, High-Capacity Memory,” white paper from M-system company, Sep. 2003.

[28] G.. Umanesan, E. Fujiwara, “A Class of Random Multiple Bits in a Byte Error Correcting and Single Byte Error Detecting (St/bEC-SbED) Codes,” IEEE Trans. on computer, vol. 52. , pp. 835-847, Jul. 2003.

[29] T. Ritthongpital, E. Fujiwara, M. Kitakami, “Optimal Two-Level Unequal Error Control Codes for Computer Systems,” IEEET proceedings of FTCS, pp. 190-199, Jun. 1996.

[30] G.. Umanesan, E. Fujiwara, “Single Byte Error Control Codes with Double Bit Within a

Block Error Correcting Capability for Semiconductor Memory Systems," IEEE Symp. On Defect and Fault Tolerance in VLSI Systems, pp. 192-200, Oct. 2000.

[31] E. Fujiwara, Masato Kitakami "A Class of Error-Locating Codes for Byte-Organized Memory Systems," IEEE Trans. on information theory vol. 40, pp. 1857-1865, Nov. 1994.

[32] T. R. N. RAO. and E. Fujiwara, Error Control Coding for Computer Systems, Prentice Hall, 1989.

[33] S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice Hall, 1983.

[34] WAYEN D. GROVER, THOMAS E. MOORE "Design and Characterization of an Error-Correcting Code for SONET STS-1 Tributary," IEEE Trans. on communication, vol 38. , pp. 467-476, APR. 1990.

[35] Sterling R., etc, "Reed-Solomon VLSI Codec for Advanced Television," IEEE Trans. on Circuits and Systems, pp. 230-236, June. 1991.