

國立交通大學

電子工程系

碩士論文

里德所羅門解碼器之通用型架構設計  
Universal Architectures for Reed-Solomon  
Error-and-Erasure Decoder



研究生:張富科

指導教授:張錫嘉

中華民國九十四年八月

里德所羅門解碼器之通用型架構設計  
Universal Architectures for Reed-Solomon  
Error-and-Erasure Decoder

學生：張富科

Student : Fuke Chang

指導教授：張錫嘉

Advisor : Hsie-Chia Chang

國立交通大學

電子工程系

碩士論文



A Thesis

Submitted to Department of Electronics  
College of Electrical Engineering and Computer Science  
National Chiao Tung University  
In Partial Fulfillment of the Requirements  
For the Degree of Master  
In  
Electronics Engineering

August 2005

Hsinchu, Taiwan, R.O.C.


# 里德所羅門解碼器之通用型架構設計

學生：張富科

指導教授：張錫嘉

國立交通大學電機工程學系（研究所）碩士班

## 摘 要



里德所羅門碼主要用來保護資料來避免在傳輸中可能發生的錯誤，它的數學演算主要是根據有限場(finite field)的運算。里德所羅門碼在許多應用上都有例子，譬如 CD, DVD 光碟機，cable modem 以及 DVB-T 的系統。然而在各種應用裡，因應不同的規格要求，每種里德所羅門碼有著完全不同的參數以及不同的有限場的定義和  $p(x)$ 。而以往的多模式設計，總需要花上許多的硬體和週期來處理不同有限場定義的問題。因此本論文提出一個完全多模式的里德所羅門碼解碼器，它可以同時處理不同的參數包含可更正的錯誤和有現場的定義。我們總共提出兩種的架構，第一種架構主要支援最高有限場次方到 10，第二種架構有限場次方到 8。除此之外，我們還應用一些小面積的設計考量於次方為八的架構，使得能夠達到小面積的設計。這兩種架構都以 0.13 1P8M 的製程來實現，分別需要 110K 和 53K 個邏輯閘。根據模擬的結果，最快可以達到 220MHz 以及 250MHz 的工作頻率。

# Universal Architectures for Reed Solomon Error-and-Erasure Decoder

Student: Fuke Chang

Advisor: Hsie-Chia Chang

Institute of Electronics  
National Chiao Tung University

## ABSTRACT

Due to protecting the data form random error and burst error during transmission, Reed Solomon (RS) code has been widely accepted as the forward error correction scheme, such as xDSL, cable modem, and DVB-T. Because of the different RS specific parameters, a cost efficient RS decoder that can support various applications has practical importance to reduce the time-to-market and design costs. This thesis presents two universal architectures for Reed Solomon (RS) error-and-erasure decoder that can accommodate any codeword with different code parameters and finite field definitions. The first architecture can support the maximum degree to 10, and the second architecture can support to 8. The area efficient design approach is also considered in second architecture. Implemented with 1.2V 0.13 $\mu$ m 1P8M technology, the two decoders can operate at 220 MHz and 300MHz and reach 2.2Gb/s and 2.4Gb/s data rate, respectively. The total gate counts of two decoders are 110K with core size 0.78mm<sup>2</sup> and 54K with the core size 0.36mm<sup>2</sup>.

## 誌 謝

二年的研究所生活很快就過去了，在這兩年中學到許多作學問的方法以及一些處世的道理。當然要感謝的人非常多，首先最要感謝的當然是我的指導教授張錫嘉博士，這兩年的指導不但能夠讓我有一些研究成果，在遇到一些停頓時都能夠指導一個正確的研究方向，當然老師的好相處最讓我印像深刻。再來要感謝的就是幫助我最多的林建青學長，他讓我在研究上能夠有一個很可靠的支柱，包括研究的方向以及 IC 設計上所碰到的種種問題，可以讓我很順利的的完成各項進度。當然也感謝每個 FEC group 成員，以及最棒的 oasis 實驗室同學和學弟，這兩年我過的真的很快樂也很充實。最後再一次跟每個人說一聲謝謝。



# CONTENTS

中文摘要.....	ii
英文摘要.....	iii
誌謝.....	iv
目錄.....	v
圖目錄.....	vii
表目錄.....	ix
Chapter 1 .....	- 1 -
Introduction .....	- 1 -
1.1 Background.....	- 1 -
1.2 Motivation .....	- 2 -
1.3 Thesis Organization .....	- 4 -
Chapter 2 .....	- 5 -
Introduction to Reed-Solomon code.....	- 5 -
2.1 Reed Solomon Encoding .....	- 5 -
2.2 RS Code Decoding with Erasure Correction.....	- 7 -
Berlekamp Massey algorithm.....	- 9 -
Euclidean Algorithm.....	- 10 -
Chapter 3 .....	- 14 -
Universal Finite Field Operator.....	- 14 -
3.1 Montgomery Multiplication Algorithm.....	- 14 -
3.2 Universal Finite Field Multiplier.....	- 17 -
3.3 Universal Finite Field Inverter .....	- 19 -
Fermat's algorithm.....	- 20 -
On-the-fly Inversion Table .....	- 21 -
Chapter 4 .....	- 22 -
Proposed Universal Architectures .....	- 22 -
4.1 Universal Syndrome and Erasure Value Calculator .....	- 23 -
4.2 Erasure Locator Polynomial Expansion and Key Equation Solve block .....	- 25 -
Decomposed Berlekamp-Massey Architecture .....	- 25 -
Expansion Hardware of Erasure Locator Polynomial .....	- 26 -
Computation for Errata Evaluator Polynomial.....	- 29 -

4.3 Chien search and Error Evaluator Block .....	- 29 -
Error value evaluator .....	- 31 -
4.4 Summary.....	- 32 -
Chapter 5 .....	- 34 -
Area Efficient Design Approach.....	- 34 -
5.1 Universal Syndrome and Erasure Value Calculator .....	- 34 -
5.2 Chien search and Error Evaluator Block.....	- 36 -
5.3 $8 \leq t \leq 16$ Error-only Correction.....	- 39 -
5.5 Summary.....	- 40 -
Chapter 6 .....	- 42 -
Chip Implementation Result.....	- 42 -
6.1 Design and Test Consideration .....	- 42 -
6.2 CHIP Implementation for Proposed Architecture 1 .....	- 44 -
6.3 CHIP Implementation for Proposed Architecture II.....	- 46 -
6.4 Comparison.....	- 48 -
Chapter 7 .....	- 50 -
Conclusion.....	- 50 -
APPENDIX .....	- 51 -
Hardware Sharing Design for (528, 518) RS codec IP .....	- 51 -
Proposed Hardware Sharing Architecture .....	- 51 -
Composite Field Inverter .....	- 55 -
Implementation Result.....	- 57 -
Conclusion.....	- 58 -
BIBLIOGRAPHY .....	- 59 -



# List of Figures

Figure 1.1: Block diagram of communication system.....	- 1 -
Figure 2.1: The systematic RS encoding architecture .....	- 7 -
Figure 2.2: The systematic RS decoding scheme .....	- 8 -
Figure 3.1: Montgomery multiplier structure for $GF(2^m)$ while $m \leq 4$ .....	- 18 -
Figure 3.2: The finite field inverter based on Fermat's algorithm.....	- 20 -
Figure 3.3: On-the-fly inversion table .....	- 21 -
Figure 4.1: The syndrome cell of syndrome calculator .....	- 24 -
Figure 4.2: Universal Syndrome Block.....	- 24 -
Figure 4.3: The decomposed key equation architecture for calculate the error locator polynomial.....	- 26 -
Figure 4.4: Using decomposed architecture to compute the erasure locator polynomial....	- 28 -
Figure 4.5: (a) the double check Chien search cell. (b) Chien search architecture with correctable erasure is $n-k$ .....	- 30 -
Figure 4.6: The serial error evaluator architecture with one FFM. ....	- 32 -
Figure 4.7: The Timing Diagram for propose I architecture. ....	- 32 -
Figure 4.8: Block diagram of proposed I RS decoder. ....	- 33 -
Figure 5.1: The universal Syndrome and Erasure Value Calculator.....	- 36 -
Figure 5.2: The parallel Chien-search block with constant UFFM.....	- 37 -
Figure 5.3: The parallel error evaluator block with constant UFFM.....	- 38 -
Figure 5.4: Block diagram of 16 errors correction. ....	- 39 -



Figure 5.5: Decoding timing diagram for 16 errors-only correction.....	- 40 -
Figure 5.6: The Timing Diagram for propose II architecture.....	- 40 -
Figure 5.7: Block diagram of proposed II RS decoder.....	- 41 -
Figure 6.1: The entire design flow.....	- 43 -
Figure 6.2: The simulation environment .....	- 44 -
Figure 6.3: The die photo of proposed I architecture .....	- 45 -
Figure 6.4: The layout view of proposed II architecture .....	- 47 -
Figure A.1: (a) Encoder/Syndrome calculator block, (b) Syndrome cell (SCi).....	- 52 -
Figure A.2: The decomposed Berlekamp-Massey architecture with finite field inverter....	- 53 -
Figure A.3 : Chien-search and error-value evaluator block.....	- 53 -
Figure A.4: The hardware sharing architecture.....	- 54 -
Figure A.5 : Composite field inverter over $GF(2^{10})$ .....	- 56 -



# List of Tables

Table 1.1: Some application of Reed Solomon decoder and its finite field definition.....	- 2 -
Table 1.2: The number of primitive polynomial with different field degree.....	- 2 -
Table 3.1: The comparison of universal finite field multiplier.....	- 19 -
Table 6.1: The chip summary of proposed I universal RS decoder.....	- 46 -
Table 6.2: The chip summary of proposed II architecture.....	- 48 -
Table 6.3: The comparison of various mode RS decoder.....	- 49 -
Table A.1 : Comparison of required registers and finite-field function units. The C-S and E-E represent the Chien search block and error evaluator, respectively.....	- 55 -
Table A.2 : The comparison table for (528,518) RS codec with different key-equation block..	- 57 -

# CHAPTER 1

## Introduction

### 1.1 Background

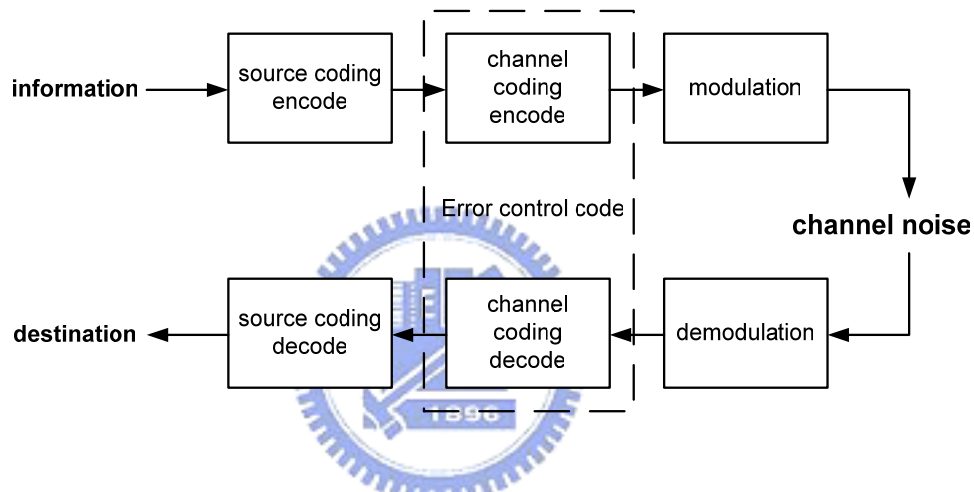


Figure 1.1: Block diagram of communication system

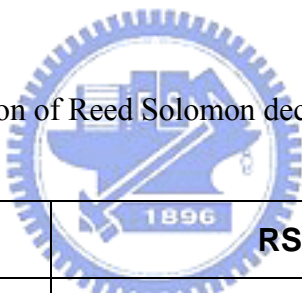
The importance of efficient and reliable data transmission in communication system is required in recent years. Fig 1.1 shows the typical communication system which composed of source coding, channel coding and modulation [1]. However, we only focus on the channel coding block or be named as well as error control coding which is used to resist the channel noise during data transmission. As shown of figure, the error control code is composed of channel encoder and channel decoder. The channel encoder is used to encode the information symbol with additional redundancy bits. The channel decoder can decode the encoded codeword and has capable of correcting the errors. The error control code also can be separate form different encoding arithmetic, one is block code and the other is convoluitional code. The Reed Solomon (RS) code which belongs to block code and has

cyclic structure [2] will be described in this thesis that includes algorithm research and hardware implementation.

## 1.2 Motivation

In recent years, the Reed Solomon code is used in many applications, such as xDSL, cable modem, DVD, blue-ray disc, and DVB-T systems. Table 1.1 shows a list of RS code applications and the finite field (FF) definition, and the Table 1.2 indicated the number of primitive polynomial with different field degree [3]. From table 1.1, we know that there are many different RS specifications in single systems. For example, the ITU J.83 system which includes of two different finite field definitions and the correctable error number has 3 different modes.

Table 1.1: Some application of Reed Solomon decoder and its finite field definition



Applications		RS code specifications
Blue-ray DISC	LDC	(248,216) RS code for $GF(2^8)$ , $t=16$
	BIS	(62,30) RS code for $GF(2^8)$ , $t=16$
Flash		(526,518) RS code for $GF(2^{10})$ , $t=4$
ITU J.83	A,B	(204, 188) RS code for $GF(2^8)$ , $t=8$
	C	(128,122) RS code for $GF(2^7)$ , $t=3$
	D	(207,187) RS code for $GF(2^8)$ , $t=10$
DVB-T		(204, 188) RS code for $GF(2^8)$ , $t=8$

Table 1.2: The number of primitive polynomial with different field degree

<b>Finite field degree m</b>	<b>Primitive polynomial number</b>
5	6
6	12
8	34
10	106

Because of the different RS specific parameters, a cost efficient RS decoder that can support various applications has practical importance to reduce the time-to-market and design costs. There are many similarities between various applications and the hardware can be shared for lower cost design. This thesis proposes two universal architectures for Reed Solomon error-and-erasure decoder that can accommodate any codeword with different code parameters and finite field definitions. The proposed I supports the maximum field degree to ten, and the corrective error is eight, and the proposed II can support the maximum field degree to eight, and the corrective error is sixteen. The area efficient approach is adopted for implementing the proposed II architecture. Furthermore, the proposed decoders can support erasure correction without increasing any finite field multipliers.

The design challenge is how to realize a dedicated RS decoder that is suitable for different finite field definitions. The Montgomery multiplication algorithm will be used to deal with the relation between different finite field definitions. In comparison with other reconfigurable RS decoders, the proposed design, based on the Montgomery multiplication algorithm, can support various finite field degrees, different primitive polynomials, and erasure decoding functions.

### ***1.3 Thesis Organization***

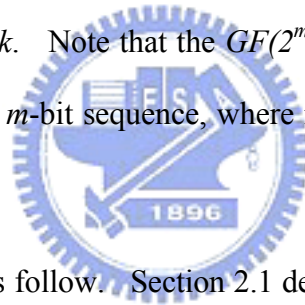
The organization of this thesis is described as follows. In chapter 2, the Reed Solomon code algorithm includes encoding and decoding will be introduced. Chapter 3 shows the Montgomery multiplication algorithm [8], universal finite field multiplier and universal finite field inverter. Additionally, the on-the-fly look-up table is described in subsection 3.3. The proposed universal RS decoder architecture will be addressed in chapter 4. Each block and its design methodology of proposed decoder will be described in detail. In chapter 5, the each subsection will show another area efficient design in proposed II architecture. The design and test consideration and chip implementation is shown in Chapter 6. We also compare the chip performance and size with others architecture in Chapter 6. Finally, Chapter 7 is the conclusion and future work.



# CHAPTER 2

## Introduction to Reed-Solomon code

Reed Solomon (RS) code which is used to protect the data during transmission has been widely accepted as the forward error correction scheme for various optical storage systems and communication systems,. The fundamental arithmetic of RS code is built on the Galois field which denoted with  $GF$  [9]. A RS code over  $GF(2^m)$  can be represented  $(n, k, t)$  code which has block length  $n$  and  $n-k$  parity symbols. The number of maximum correctable errors is  $t$  and the number of parity symbols is  $n-k$ . Note that the  $GF(2^m)$  indicates that RS code are non-binary code with symbols made up of  $m$ -bit sequence, where  $m$  is any integer having a value greater than two.



This chapter is organized as follow. Section 2.1 describes the RS encoding procedure and its mathematical arithmetic. The RS decoding scheme is presented in Section 2.2, and the Berlekamp-Massey algorithm and Euclidean algorithm which are used to solve the key equation will be introduced [3, 4].

### ***2.1 Reed Solomon Encoding***

It has been know that the  $GF(p^m)$  which  $p$  is a prime number can be represented using  $0$  and  $(p-1)$  consecutive powers of a primitive field element  $a \in GF(p^m)$ . Symbols from the field  $GF(2^m)$  are used in the construction of Reed Solomon code. Each of the  $2^m$  elements of the finite field  $GF(2^m)$  can be represented as a distinct polynomial of degree  $m-1$ .

$$\alpha^j = \alpha_i(x) = \alpha_{i,0} + \alpha_{i,1}x + \alpha_{i,2}x^2 + \dots + \alpha_{i,m-1}x^{m-1} \quad , \text{ for } i = 0 \sim 2m-2 \quad (2.1)$$

Let  $M(x)$  represented as  $(m_{k-1}, m_{k-2}, \dots, m_1, m_0)$  be the information symbols with  $k$  symbols. And the  $G(x)$  is the generator polynomial which is the product of the associated minimal polynomial.

$$G(x) = (x + \alpha^b)(x + \alpha^{b+1}) \dots (x + \alpha^{b+2t-2})(x + \alpha^{b+2t-1}) \quad (2.2)$$

Where the degree of  $G(x)$  is equal to the number of parity symbols, and the  $b$  is a constant. Therefore, for an  $(n, k, t)$  RS code, the nonsystematic encoding procedure can be expressed as follow:

$$\begin{aligned} C(x) &= G(x) \cdot M(x) \\ &= (x + \alpha^b)(x + \alpha^{b+1}) \dots (x + \alpha^{b+2t-1}) * M(x) \end{aligned} \quad (2.3)$$

Where the  $C(x)$  is the codeword that has  $2t$  roots of  $\alpha^{b+1} \sim \alpha^{b+2t}$ .

Another encoding approach to encode the information symbols is the systematic encoding [4] which uses the parity check symbols to form the codeword. Firstly, the message polynomial  $M(x)$  is multiplied by  $x^{2t}$  and then modular by the generator polynomial  $G(x)$  to obtain the remainder polynomial  $R(x)$ .

$$R(x) = M(x) \cdot x^{2t} + Q(x) \cdot G(x) \quad (2.4)$$

Where the  $Q(x)$  is the quotient polynomial of the divided polynomial  $M(x) \cdot x^{2t}$  and the divisor polynomial  $G(x)$  which has degree less than  $2t-1$ . The systematic polynomial can be expressed as follow:

$$\begin{aligned} C(x) &= M(x) x^{2t} + R(x) \\ &= G(x) Q(x) \end{aligned} \quad (2.5)$$



We can think of shifting a message polynomial  $M(x)$  into the rightmost  $k$  location of a codeword and appending  $2t$  parity check symbols in the leftmost location. Fig. 2.1 shows the typical systematic encoding circuit with  $2t$  register, where the  $g_0, g_1, \dots, g_{2t-1}$  is the coefficient of generator polynomial. The output symbols are the message  $M(x)$  during the first  $k$  clock cycles. The remaining  $n-k$  cycles, the parity symbols  $R(x)$  are moved to output. The total number of required clock cycles is equal to  $n$ .

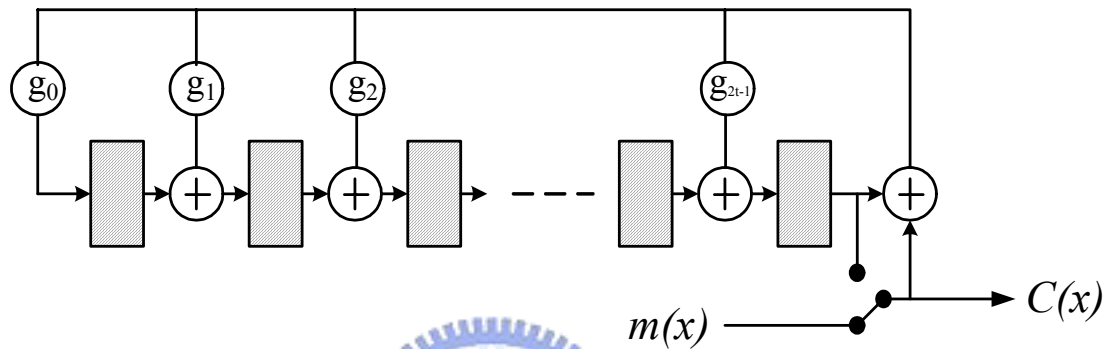


Figure 2.1: The systematic RS encoding architecture



## 2.2 RS Code Decoding with Erasure Correction

As mentioned early, the codeword polynomial is  $C(x)$  and the error polynomial is  $e(x)$ . The received polynomial  $r(x)$  can be expressed as follow:

$$r(x) = C(x) + e(x) \quad (2.6)$$

Fig. 2.2 shows the error-only RS decoding flow chart which can be divided into four steps: 1) calculation of the syndrome  $S(x)$  form the received codeword, 2) computation of the error locator polynomial  $\sigma(x)$  and the key equation  $\Omega(x)$  with Berlekamp-Massey algorithm [5, 6] or Euclidean algorithm, 3) search of the error location by Chien search approach, and 4) evaluation of error value.

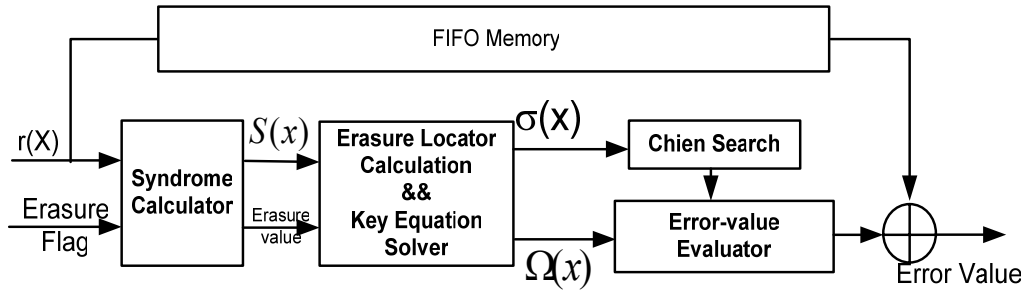


Figure 2.2: The systematic RS decoding scheme

The syndrome is the result of a parity check performed on received polynomial  $r(x)$  to determine whether  $r(x)$  is a valid member of the codeword set. If the received polynomial has no errors, then the syndrome polynomial  $S(x)$  is always 0. On the other hands, any nonzero value of syndrome indicates the presence of errors. The computation of a syndrome symbols can be describes as follows:

$$S(x) = \sum_{i=1}^{2t} r(\alpha^i) x^{i-1} \quad (2.7)$$

$$S_1 = r(\alpha^1) = e(\alpha^1) = e_1\chi_1 + e_2\chi_2 + \dots + e_v\chi_v$$

$$S_2 = r(\alpha^2) = e(\alpha^2) = e_1\chi_1^2 + e_2\chi_2^2 + \dots + e_v\chi_v^2$$

...

$$S_{2t} = r(\alpha^{2t}) = e(\alpha^{2t}) = e_1\chi_1^{2t} + e_2\chi_2^{2t} + \dots + e_v\chi_v^{2t} \quad (2.8)$$

Where the  $e_i$  represents the  $i$ -th error value and the  $v$  is the occurred error number, and the  $\chi_i$  represents the error location number. When a nonzero syndrome vector has been computed, it signifies that an error has occurred. Then, the error locator polynomial  $\sigma(x)$  and the key equation  $\Omega(x)$  will be computed. An error locator polynomial and key equation are defined as

$$\begin{aligned}\sigma(x) &= (1+\beta_1x)(1+\beta_2x)\cdots(1+\beta_vx) \\ &= \sigma_0 + \sigma_1x + \sigma_2x^2 + \sigma_3x^3 + \cdots + \sigma_vx^v\end{aligned}\quad (2.9)$$

$$\begin{aligned}\Omega(x) &= S(x) \sigma(x) \bmod x^{n-k} \\ &= e_1\chi_1(1-\chi_2x)(1-\chi_3x)\cdots(1-\chi_vx) \\ &\quad + e_2\chi_2(1-\chi_1x)(1-\chi_3x)\cdots(1-\chi_vx) \\ &\quad + e_3\chi_3(1-\chi_1x)(1-\chi_2x)\cdots(1-\chi_vx) \\ &\quad + \dots\end{aligned}\quad (2.10)$$

### Berlekamp Massey algorithm

In 1960, Peterson provided the first explicit description of a decoding algorithm for binary BCH code. He uses the relation of error locator polynomial and syndrome vector to solve the key equation. The relation can be rewritten as a matrix form:

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_v \\ S_2 & S_3 & S_4 & \cdots & S_{v+1} \\ S_3 & S_4 & S_5 & \cdots & S_{v+2} \\ & & \cdots & & \\ S_v & S_{v+1} & S_{v+2} & \cdots & S_{2v-1} \end{bmatrix} \begin{bmatrix} \sigma_v \\ \sigma_{v-1} \\ \cdot \\ \cdot \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ \cdot \\ \cdot \\ -S_{2v} \end{bmatrix}\quad (2.11)$$

But this algorithm is inefficient for large correctable error number code. Consequently, the error locator polynomial and error evaluate polynomial are always computed by Berlekamp-Massey algorithm or modified Euclidean algorithm in the past. The Berlekamp-Massey algorithm which is developed by Berlekamp and explained by Massey with linear feedback register has regular property for decoding key equation. The entire Berlekamp-Massey algorithm is shown as:

**1) Initially**  $\Lambda^{(b)}(x) = 1, \Lambda^{(a)}(x) = 1, l = 0, k = 1, \gamma^{(k)} = 1$

**2) Compute (a)**  $\Lambda^{(a)}(x) \leftarrow x\Lambda^{(a)}(x)$  and  $\delta = \sum_{j=0}^l \Lambda_j^{(b)} S_{k-j}$

$$(b) \Lambda^{(c)}(x) = \Lambda^{(b)}(x) + \frac{\delta}{\gamma} \Lambda^{(a)}(x)$$

**(c) If**  $\delta \neq 0$  and  $2l \leq k - 1$

$$\text{Set } \Lambda^{(a)}(x) = \Lambda^{(b)}(x), \quad l = k - l, \quad \gamma = \delta$$

$$(d) \Lambda^{(b)}(x) = \Lambda^{(c)}(x)$$

**3) Set**  $k = k + 1$ . If  $k < d$ , then go step 2.

**4) Stop**



The  $\delta$  is the discrepancy which is the convolution of syndrome vector and error locator polynomial and the  $\gamma$  is the dummy nonzero discrepancy that keeps the value of previous discrepancy. The discrepancy is used to verify that the linear feedback shift register generates corresponding the given syndrome sequence at each step. If the discrepancy is equal to zero, it represents that we don't update the error locator polynomial and the dummy discrepancy. For operating the Berlekamp-Massey algorithm, it totally costs  $2t$  iteration.

## Euclidean Algorithm

In 1975 Sugiyama et al. showed that Euclidean algorithm can decode Reed Solomon code. The Euclidean algorithm originally is used to calculating the greatest common divisor of two

polynomials. For rewriting the key equation, the Euclidean algorithm can be applied to produce the correct sets of solutions  $(\sigma(x), \Omega(x))$  that satisfy as

$$\begin{aligned}\Omega(x) &= S(x) \cdot \sigma(x) \bmod x^{n-k} \\ \Rightarrow Q(x) \cdot x^{n-k} + S(x) \cdot \sigma(x) &= \Omega(x)\end{aligned}\tag{2.12}$$

Where the  $Q(x)$  is the quotient polynomial of the dividend polynomial  $S(x)\sigma(x)$  and divisor polynomial  $x^{n-k}$ . The  $Q(x)$  is not available for us, but the pair  $(\sigma(x), \Omega(x))$  is the interested solution. The  $\Omega(x)$  computation is similar as calculating the GCD polynomial of  $x^{n-k}$  and  $S(x)$ .

$$\begin{aligned}R_1(x) &= x^d + S(x)Q_1(x) \\ R_2(x) &= S(x) + R_1(x)Q_2(x) \\ R_3(x) &= R_1(x) + R_2(x)Q_3(x) \\ &\dots \\ \Omega(x) &= R_{n-2}(x) + R_{n-1}(x)Q_n(x)\end{aligned}\tag{2.12}$$

where the  $Q_{(i)}(x)$  is  $i$ -steps quotient polynomial and  $R_{(i)}(x)$  is the  $i$ -steps remainder polynomial. At each step, the division operation of polynomial is performed. According to the Euclidean algorithm, the computation of error locator polynomial is shown as

$$\begin{aligned}\sigma_1(x) &= 0 + 1Q_1(x) \\ \sigma_2(x) &= 1 + \sigma_1Q_2(x) \\ \sigma_3(x) &= \sigma_1(x) + \sigma_2(x)Q_3(x) \\ &\dots \\ \sigma(x) &= \sigma_{n-2}(x) + \sigma_{n-1}(x)Q_n(x)\end{aligned}\tag{2.13}$$

Where the  $Q(x)$  is same as the result of computing error evaluate polynomial. From the equation (2.12), it is known that the error locator polynomial can be calculated by given quotient polynomial and previous error locator polynomial. For performing Euclidean

algorithm, the degree increasing of  $\sigma(x)$  is in opposition to the degree of  $\Omega(x)$ . Hence, the Euclidean decoding procedure terminates when the degree of  $\sigma(x)$  is larger than the degree of  $\Omega(x)$ .

Because of the regularity of Berlekamp Massey algorithm, the proposed universal architecture is implemented by applying this algorithm. After the key equation and error equation have been calculating, the next step is finding the error location roots by Chien search approach. The methodology of Chien search is substitution of error locator polynomial with finite field elements to check the result equals zero or not.

$$\sigma(\alpha^i) = 0 \text{ for } i = 0, 1, 2, \dots, N. \quad (2.14)$$

Then, according to Forney algorithm [7], the error value can be computed as follow:

$$e_i = \frac{\Omega(x_i^{-1})}{\sigma'(x_i^{-1})} \quad (2.15)$$

The  $x_i$  and the  $\sigma'(x)$  are the location root at the Chien search step and the derivative of error locator polynomial  $\sigma(x)$  respectively.

Erasure is a type of error with the position information. A RS decoder with erasure correction will improve the performance in various systems. For a  $(n, k, t)$  RS code, the erasure correction capability of the code is

$$s = d_{min} - 1 = n - k \quad (2.16)$$

where the  $d_{min}$  is the minimum distance between any two codewords. For RS code the minimum distance is given by

$$d_{min} = n - k + 1 \quad (2.17)$$

Simultaneous error correction and erasure correction capability can be expressed by the requirement that

$$2v+s < d_{min} < n-k \quad (2.18)$$

where  $v$  is the number of symbol error that can correct, and the  $s$  is the number of symbol erasure that can be corrected. For decoding erasure, it is shown that the error and erasure locator polynomial (errata locator polynomial) can be obtained directly by initiating an inverse-free BM algorithm with the erasure locator polynomial. Consequently, we just consider the expansion of erasure locator polynomial. The erasure locator polynomial is computed by the following equation.

$$T(x) = \prod (1 + \alpha^i x) \text{ mod } x^{2t} \quad (2.19)$$

Hence, the error-erasure locator polynomial  $\Lambda(x)$  (or say errata locator polynomial) and key equation  $W(x)$  of erasure correction can be rewritten respectively as follows:

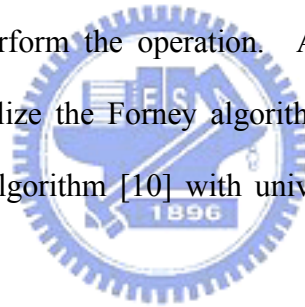
$$\Lambda(x) = \sigma(x) \cdot T(x) \text{ mod } x^{n-k} \quad (2.20)$$

$$W(x) = S(x) \cdot \Lambda(x) \text{ mod } x^{n-k} \quad (2.21)$$

# CHAPTER 3

## Universal Finite Field Operator

This Chapter describes the Montgomery multiplication algorithm [8] and indicates the implementation of universal finite field operators. The basic idea to achieve universal property is applying the universal finite field multiplier which can accommodate different finite field definition [20]. In comparison with others proposed approach, the universal finite field multiplier only cost two cycles to realize the finite field multiplication for various definitions. What if the input of universal FFM multiplies is replaced with the corrective factor at first, it only requires one cycle to perform the operation. Additionally, the universal finite field inverter is the last step to realize the Forney algorithm. Two approaches are presented in subsection 3.3, the Fermat's algorithm [10] with universal multiplier and on-the-fly lookup table with SRAM.



### *3.1 Montgomery Multiplication Algorithm*

An element  $A$  of the field  $GF(q^m)$  with a prime  $q$  can be interpreted as the polynomial representation. In the polynomial representation, multiplication in  $GF(q^m)$  corresponds to the multiplication of polynomials module an irreducible polynomial of degree  $m$ . Suppose  $A$  and  $B$  are two elements in  $GF(q^m)$ , and  $p(x)$  is the corresponding primitive polynomial of this field. Then, the multiplicative operation  $C=AB$  can be defined as follows:

$$C(x) = A(x)B(x) \text{ mod } p(x) \quad (3.1)$$

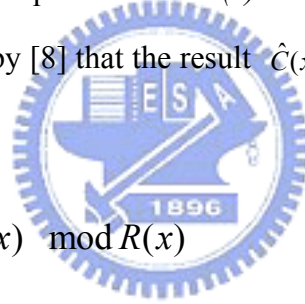
Where  $C$  is also an element of  $GF(q^m)$ .



Actually, the finite field addition and subtraction are just exclusive OR operations. Therefore, what we interested is the multiplication and division (or say, the inverse operation) in finite field. According to the modular multiplication property in (3.1), we can adopt Montgomery multiplication algorithm to calculate the product  $C(x)$ . The Montgomery multiplication algorithm has been proven that this algorithm can replace the modular operation with a series multiplication. The following equation defines the Montgomery product of A and B:

$$\hat{C}(x) = A(x)B(x)R^*(x) \pmod{p(x)} \quad (3.2)$$

The polynomial  $R^*(x)$  here is a fixed element of  $GF(q^m)$  satisfying  $R(x)R^*(x) = 1 \pmod{p(x)}$  while  $R(x) = x^m$ . Note that the requirement of  $R(x)$  and  $p(x)$  being relatively prime is always consistent. It has been proven by [8] that the result  $\hat{C}(x)$  of (3.2) can be obtained by following equations:



$$Q(x) = A(x)B(x)p^*(x) \pmod{R(x)} \quad (3.3)$$

$$\hat{C}(x) = [A(x)B(x) + Q(x)p(x)] / R(x) \quad (3.4)$$

The polynomial  $p^*(x)$  in (3.3) is defined as  $p(x)p^*(x) = 1 \pmod{R(x)}$ . As compared with (3.2), it is evident that modulo  $p(x)$  operation is replaced by modulo  $R(x)$  and division by  $R(x)$  operations. Since  $R(x) = x^m$ , implementation of (3.3) and (3.4) are much easier than that of (3.2). Furthermore, as A is interpreted in polynomial form and  $R^*(x) = x^{-m} \pmod{p(x)}$ , (3.2) can be rewritten as:

$$\begin{aligned} \hat{C}(x) = & [a_{m-1}B(x)x^{-1} \pmod{p(x)}] + [a_{m-2}B(x)x^{-2} \pmod{p(x)}] \\ & + \dots + [a_0B(x)x^{-m} \pmod{p(x)}] \end{aligned} \quad (3.5)$$

Rearrange this equation, an iterative representation comes out:

$$\hat{C}(x) = [a_{m-1}B(x) + [\dots [a_1B(x) + [a_0B(x)x^{-1} \bmod p(x)]]x^{-1} \bmod p(x)] \dots]x^{-1} \bmod p(x) \quad (3.6)$$

Based on this equation and the transformation from (3.4) to (3.6), the Montgomery multiplication algorithm is derived as:

**Montgomery multiplication algorithm:**

$$\begin{aligned} S_0(x) &= 0; \\ \text{for}(i = 0; i < m; i++) \{ \\ &\quad \rho_i(x) = [(S_i(x) + a_iB(x))p^*(x)] \bmod x; \\ &\quad S_{i+1}(x) = [S_i(x) + a_iB(x) + \rho_i(x)p(x)] / x; \\ \} \\ \hat{C}(x) &= S_m(x); \end{aligned} \quad (3.7)$$

The term  $p^*(x)$  is the multiplicative inverse of  $p(x)$  under modulo  $x$  multiplication.

In  $GF(2^m)$ , elements are often represented in binary digits, and the coefficients  $a_i$  are referred to the bits of  $A$ . The binary representation will cause some reduction to Montgomery multiplication algorithm. Since  $p(x)$  is irreducible, the results of  $p(x) \bmod x$  and  $p^*(x) \bmod x$  are both equal to 1. The  $p^*(x)$  term in the Montgomery multiplication algorithm can be eliminated, which leads  $\rho_i(x)$  to equal the least significant bit of the sum  $S_i(x) + a_iB(x)$ .

The number of recursive operation in Montgomery multiplication depends on the field degree  $m$ . However, some modification can be proposed to remove the effect of unexpected variable  $m$ . In equation (3.3) and (3.4),  $R(x)$  is modified to be  $R_d(x) = x^d$ , and  $d$  is a constant integer such that  $d \geq m$ . Since the result of  $R_d^*(x) \bmod p(x)$  is an element of  $GF(q^m)$ , there exists an element  $R_d^*(x)$  in the field  $GF(q^m)$  that satisfies  $R_d(x)R_d^*(x) = 1 \bmod x$ . Therefore, the modified Montgomery multiplication algorithm for  $GF(2^m)$  with  $m \leq d$  is constructed:

### **Modified Montgomery multiplication algorithm:**

$$\begin{aligned} & MM(A(x), B(x), p(x)) \{ \\ & \quad S_0(x) = 0; \\ & \quad \text{for}(i = 0; i < d; i++) \{ \\ & \quad \quad \text{if}(i \geq m) \quad a_i = 0; \\ & \quad \quad T(x) = S_i(x) + a_i B(x); \\ & \quad \quad S_{i+1}(x) = [T(x) + t_0 p(x)] / x; \\ & \quad \quad \} \\ & \quad \hat{C}(x) = S_d(x); \\ & \quad \} \end{aligned} \tag{3.8}$$

The term  $t_0$  is the least significant bit of the temporal element  $T(x)$ . If the field degree is less than  $d$ , the most significant bits of  $A$  is set to zero. The final result will be multiplying the normal finite field product  $A(x)B(x)$  by a constant element  $R_d^*(x)$  of  $GF(2^m)$ . The output of Montgomery multiplier involves a constant factor  $R_d^*(x) \bmod p(x)$  with the standard product. Such constant factor can be canceled by applying one additional Montgomery multiplier. Calculation of the product  $C(x)=A(x)B(x)$  is completed using:

$$K(x) = x^{2d} \bmod p(x) \tag{3.9}$$

$$\hat{C}(x) = MM(A(x), B(x), p(x)) \tag{3.10}$$

$$C(x) = MM(\hat{C}(x), K(x), p(x)) \tag{3.11}$$

where  $K(x)$  is treated as a constant value for a given  $p(x)$ .

## ***3.2 Universal Finite Field Multiplier***

As mentioned in chapter 2, to design a universal finite field multiplier, the circuit complexity mainly depends on the module operation for different primitive polynomial. To achieve universal finite field operation, the methodology proposed in the past is using a series

shift and multiplication operations to replace the modular operation. However, this approach costs two or more cycles to operate than original dedicated finite field operation. This section presents a new multiplier architecture that can accommodate different finite field definition. The proposed universal finite field multiplier is built on the Montgomery multiplication and only cost two cycles to operate the finite field multiplication.

According to this modified algorithm, the bit-level multiplier architecture can be implemented easily. The  $t_0$  indicate the LSB bits of  $T(x)$ , and the division of  $x$  replaces as a left shift operation. The Montgomery multiplier architecture for  $GF(2^m)$  with  $m \leq 4$  is shown in Fig. 3.1. Fig. 3.1(a) and Fig. 3.1(b) indicate the function unit and the Fig.3.1(c) illustrates the overall architecture in  $GF(2^4)$ . The signal  $a_i$  and  $b_i$  are the bits of two input element A and B, which can be expressed as  $A=(a_3a_2a_1a_0)$  and  $B=(b_3b_2b_1b_0)$  respectively. Besides,  $m_i$  is used to indicate the  $i$ -th bits of the primitive polynomial and  $S_i$  is the  $i$ -th output bits.

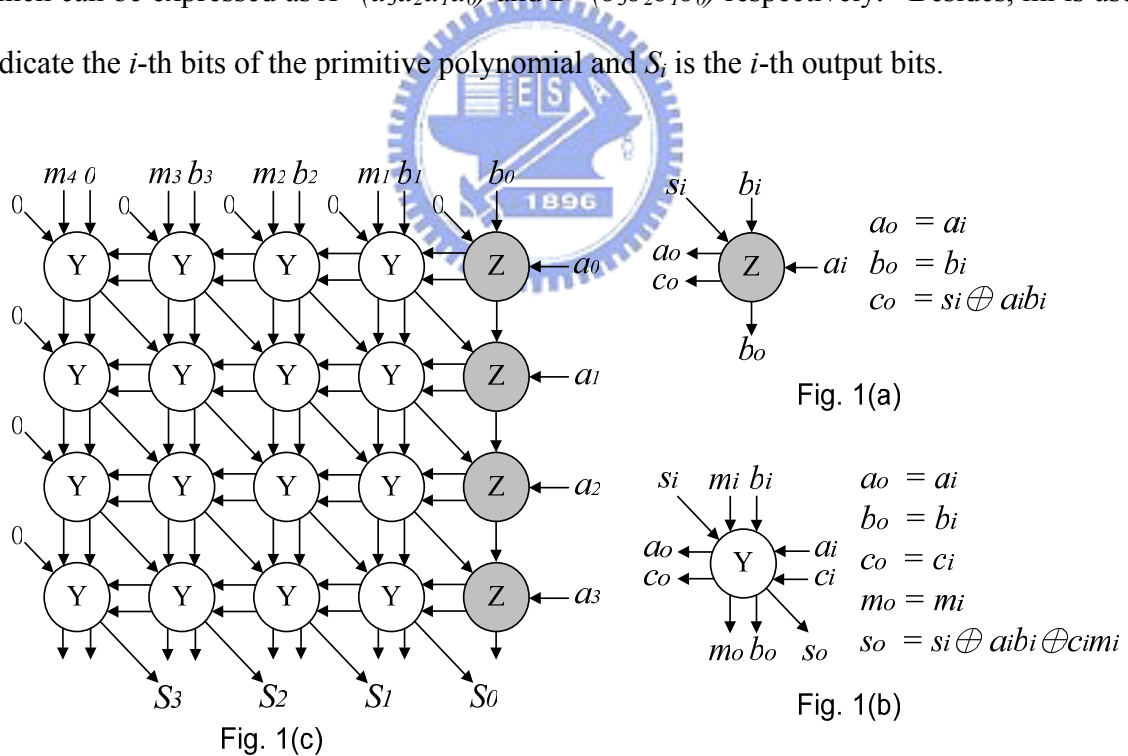


Figure 3.1: Montgomery multiplier structure for  $GF(2^m)$  while  $m \leq 4$

As the multiplier for maximum field degree  $d$  has been implemented, any multiplication of  $GF(2^m)$  with field degree less than  $d$  and corresponding primitive polynomial is applicable. As

shown in Fig. 3.1(c), the proposed bit-parallel multiplier dispenses with additional control circuit due to the regular structure.

Table 3.1: The comparison of universal finite field multiplier

	Critical path	Instruction cycle		
		$C=AB$	$C=A/B$	$C = \sum_{i=0}^{n-1} A_i B_i$
<b>L. Song [11]</b>	$8T_{AND}$ $+11T_{XOR}$	3	$4m-4$	$3n-2$
<b>Proposed [20]</b>	$9T_{AND}$ $+15T_{XOR}$	2	m	$2n$

As compared to another universal finite field multiplier proposed by [11], our approach needs no additional pre and post-shifting circuit. Table 3.1 compares the required instruction cycle between the proposed universal finite field multiplier [20] and the multiplier of [11] while operating over  $GF(2^m)$  with a multiplier that supporting maximum field degree of 8. Note that one instruction cycle here indicates a single shift operation, multiplication, or addition. And the finite field division in Table 3.1 is based on Fermat's algorithm. In this table, it is clear that our proposed multiplier cost less cycles for calculating the finite field operation.

### 3.3 Universal Finite Field Inverter

The implementation of Forney algorithm requires a universal finite field inverse operation. Two methods for realizing the inverse operation are generally used, one is using Fermat

algorithm which replaces inversion with a series of square and multiply operations [10], and the other is the looking up table.

## Fermat's algorithm

### Fermat's algorithm

$$\begin{aligned}
 \beta^{-1} &= \beta^{2^m - 2} \\
 &= \beta^{2+2^2+\dots+2^{m-1}} \\
 &= \beta^{2(1+2+\dots)} \\
 &= (\beta \dots (\beta(\beta * \beta^2)^2) \dots)^2
 \end{aligned} \tag{3.12}$$

Based on this algorithm, the inversion in  $GF(2^m)$  can be replaced by serial square and multiply operations. In addition, the Fermat algorithm shows us that inverse operation needs  $m-1$  cycles which include two Montgomery multiplications in each cycle. For example, in  $GF(16)$ ,  $\beta^{-1} = \beta^{16-2} = \beta^{2+4+8} = \beta^{2(1+2+4)} = (\beta(\beta\beta^2)^2)^2$ , three cycles are required.

Fig. 3.2 shows the architecture of inverter which is composed with two universal FFM. The control unit is added to realize the inverse operation, the Montgomery multiplier A is taken as a squarer and the multiplier B perform the finite field multiplication.

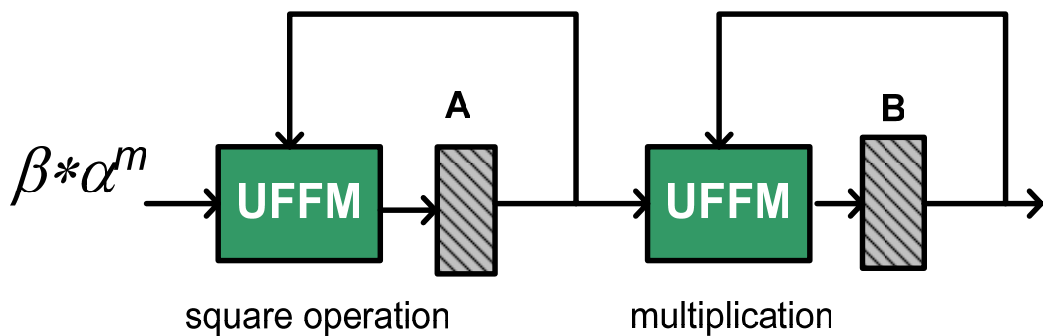


Figure 3.2: The finite field inverter based on Fermat's algorithm.

## On-the-fly Inversion Table

Since the Fermat algorithm needs many cycles to calculate the error value, leading to a larger FIFO buffer. Therefore, for high speed computation, the on-the-fly look-up table composed of  $2^m * m$  SRAM, universal  $\alpha$  generator and universal  $\alpha^{-1}$  generator, is proposed as shown in Fig. 3.3.

According to different finite field definition, the universal  $\alpha$  generator and  $\alpha^{-1}$  generator will update the finite field element and its corresponding inverse value respectively at syndrome calculating stage. At error evaluator stage, the inversion table is available for Forney algorithm. Hence, the total decoding stage can be kept on 4 and can decrease the length of FIFO buffer.

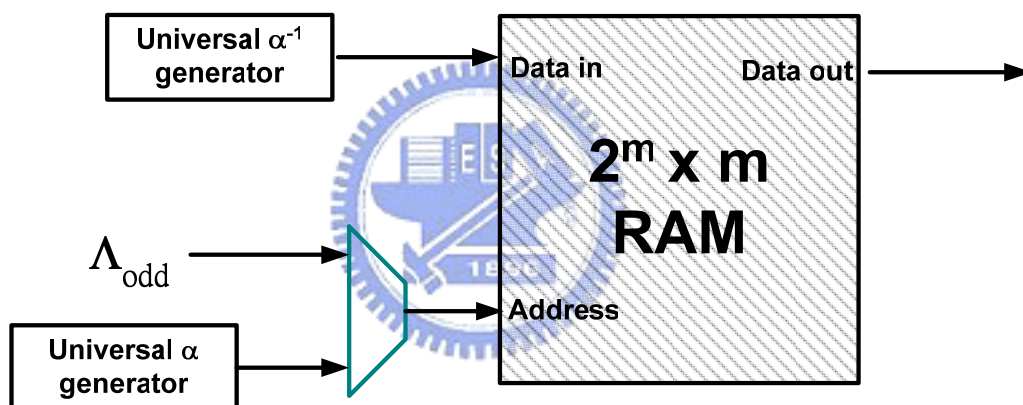


Figure 3.3: On-the-fly inversion table

# CHAPTER 4

## Proposed Universal Architectures

As mentioned in Chapter 2, the RS erasure decoder consists of syndrome calculator, erasure locator polynomial expansion, key-equation solver, Chien-search block and error-value evaluator, and a finite field inverter. The syndrome calculator computes the syndrome vector to key equation block. When the syndrome value is equal zero, the following decoding procedure will be terminated. If not, the erasure locator polynomial will be computed and the key equation block will calculate the error locator polynomial based on inverse Berlekamp-Massey algorithm [12]. The error location and the location roots will be known at Chien search step. Finally, according to the Forney algorithm, the error evaluate block calculate the error and erasure values. Besides, the FIFO buffer is used to keeps the received codeword which size is increasing with code block length.

The proposed architecture of universal RS erasure decoder is presented in this chapter. All of these components implementation mentioned early will be detailed in the following subsections. In subsection 4.1, the universal syndrome and erasure value calculator is addressed. For erasure correction, the corresponding erasure value must be kept to compute erasure locator polynomial at syndrome stage and transmit the erasure value to next stage, key equation block. For key equation block design, the authors present a decomposed inversionless BM architecture that can reduce the complexity significantly in paper [13, 14]. The proposal in [15, 16] requires  $2t\sim 3t$  finite field multiplier. However, the decomposed architecture only requires 3 finite field multipliers without any finite field inverter to implement. For decoding erasure, the key equation must replace the initial condition with the erasure locator polynomial. Therefore, the expansion hardware of erasure locator polynomial must work before operating



the Berlekamp Massey algorithm. For area efficient design, the combination of erasure locator polynomial expansion and decomposed Berlekamp-Massey architecture is presented in subsection 4.2. In subsection 4.3, the Chien search and error evaluator architecture is shown. This architecture can search the error and erasure roots of errata locator polynomial with any variable parameters.

### 4.1 Universal Syndrome and Erasure Value Calculator

This section presents universal syndrome architecture design. For design syndrome calculator in the past, the Horner's rule is applied to reduce the substitution hardware area. Let the  $R(x)$  be the received polynomial, and the syndrome value can be obtained by substituting the finite field elements  $\alpha^1, \alpha^2, \dots, \alpha^{2t}$ . This substitution of syndrome value can be expressed as follows:

$$S_i = R(\alpha^i) = ((R_{n-1}\alpha^i + R_{n-2})\alpha^i + R_{n-3})\alpha^i + \dots + R_2)\alpha^i + R_1 \quad \text{for } i=1 \sim 2t \quad (4.1)$$

For adopting the property of universal finite field multiplier, the syndrome has to be modified as:

$$\alpha^m S_i = \alpha^m R(\alpha^i) \quad \text{for } i=1 \sim 2t \quad (4.2)$$

,where the  $\alpha^m$  is the corrective factor for universal finite field multiplier. Hence, for implementing a syndrome calculator cell, a universal finite field multiplier is required. This cell architecture is shown in fig. 4.1. Assuming the correctable errors are equal eight, sixteen syndrome cells are needed.

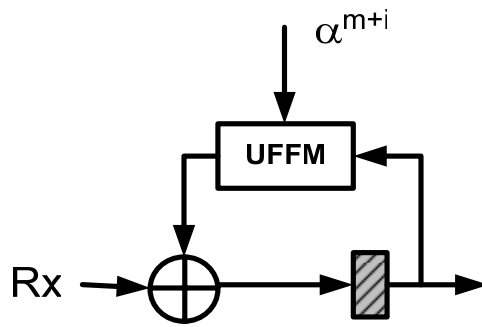


Figure 4.1: The syndrome cell of syndrome calculator

Fig. 4.2 shows the entire universal syndrome and erasure value calculator block with correctable erasure is 16. Except calculating the syndrome value, this stage also calculates erasure vectors. If the erasure flag is valid, the erasure occurs, and the corresponding erasure value must be saved. According to different correctable error and erasure number, the syndrome selector has to transmit the appropriate syndrome vector to key equation block.

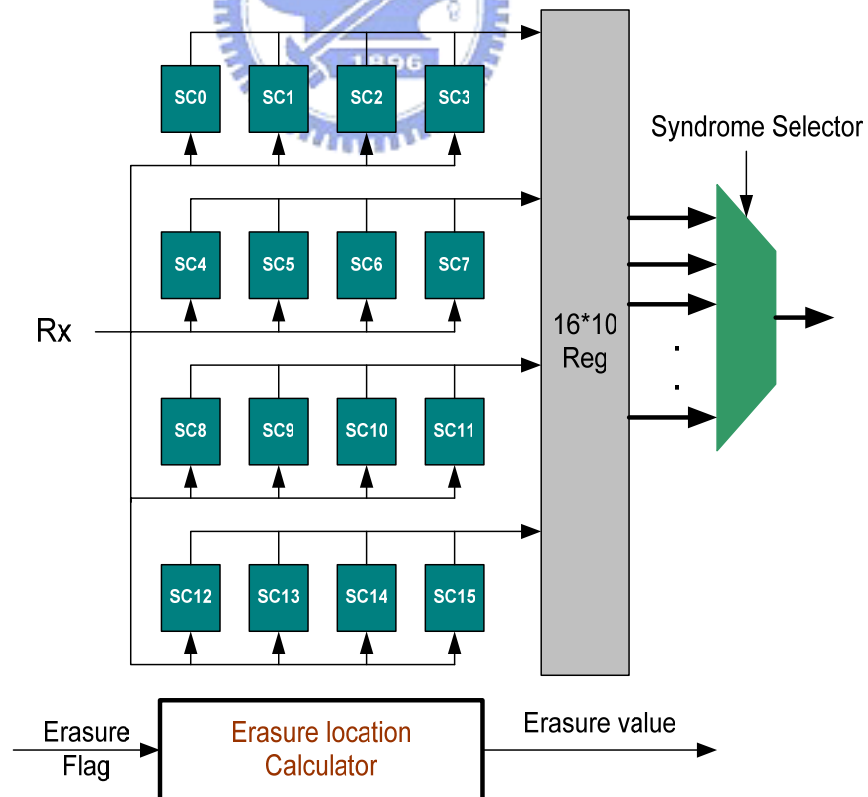


Figure 4.2: Universal Syndrome Block

## 4.2 Erasure Locator Polynomial Expansion and Key Equation Solve block

### Decomposed Berlekamp-Massey Architecture

As has been mentioned in chapter 2, the key-equation block can be implemented by two algorithms, Euclidean algorithm and Berlekamp-Massey algorithm. For implementing Berlekamp Massey algorithm, a lot of parallel architectures have realized in the past which required  $2t \sim 3t$  finite field multiplier. However, a decomposed architecture which only three finite field multipliers required has proposed to reduce the circuit complexity significantly in [13], and this architecture is based on the inversionless Berlekamp-Massey algorithm.

In the inversionless Berlekamp-Massey algorithm, the finite field inverter is replaced by a multiplier and doesn't have any influence on computing the correct result. The decomposed architecture slows down the key equation without impacting the decoding speed, and the each iteration of equation can be decomposed as following:

$$\Lambda_j^{(c)} = \begin{cases} \gamma \cdot \Lambda_0^{(b)} & , \quad \text{for } j = 0 \\ \gamma \cdot \Lambda_j^{(b)} + \delta^{(i)} \cdot \Lambda_{j-1}^{(a)} & , \quad \text{for } 1 \leq j \leq s + v_i \end{cases}$$

$$\delta_j^{(i+1)} = \begin{cases} 0 & , \quad \text{for } j = 0 \\ \delta_{j-1}^{(i+1)} + S_{i-j+3} \cdot \Lambda_{j-1}^{(c)} & , \quad \text{for } 1 \leq j \leq s + v_i \end{cases} \quad (4.5)$$

where  $\Lambda_j^{(a)}$  is the coefficient of  $\Lambda^{(a)}(x)$  and  $\delta_j^{(i)}$  is the i-steps partial result in computing the discrepancy. From above equation, only two finite field multipliers are used to computing the error locator polynomial  $\Lambda(x)$  and one finite field multiplier is needed to calculating the product of syndrome value and the coefficient of error locator polynomial.

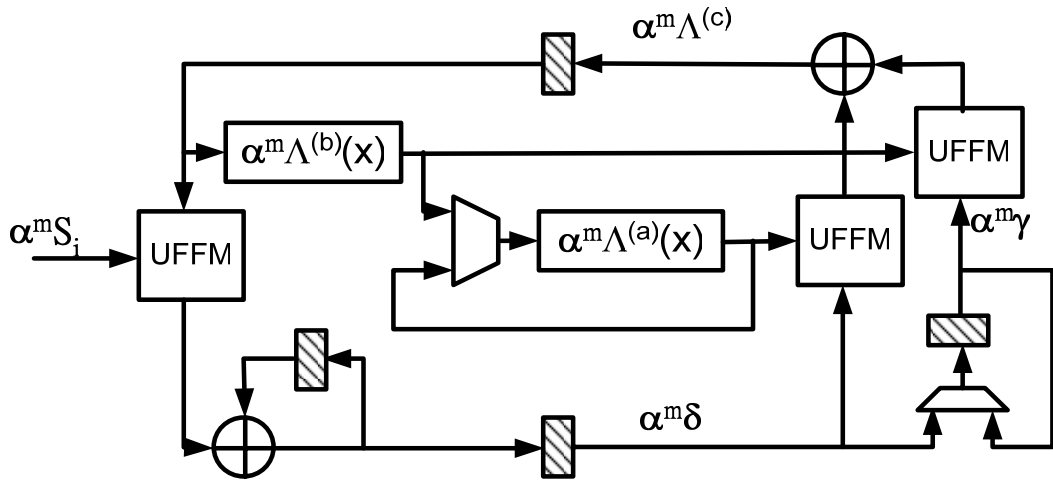


Figure 4.3: The decomposed key equation architecture for calculate the error locator polynomial.

Because of the regularity of Berlekamp Massey algorithm, the universal key equation block can be realized easily by replacing the dedicated finite field multiplier with universal finite field multiplier. Fig. 4.3 shows the error decoding steps of decomposed key equation architecture with three finite field multipliers. The register buffer with length  $(4t+1)*m$  bits is used to keep the latest error locator polynomial  $\Lambda^{(b)}(x)$  and previously error locator polynomial  $\Lambda^{(a)}(x)$ . Each initial coefficient such as discrepancy  $\delta$ , previous discrepancy  $\gamma$ , and error locator polynomial  $\Lambda^{(a)}(x)$  and  $\Lambda^{(b)}(x)$  must multiply the corresponding corrective factor  $\alpha^m$  respectively to obtain the correct result.

### Expansion Hardware of Erasure Locator Polynomial

The paper [17] shows that the error-erasure locator polynomial (errata locator polynomial) can be obtained directly by initiating an inverse-free BM algorithm with the erasure locator polynomial. Hence, for implementation of erasure correction, only the erasure polynomial expansion hardware must be considered.

In paper [18, 19], it shows two approach to implement the expansion hardware. One is the parallel architecture which costs  $n-k$  finite field multipliers and another is the serial architecture which needs two finite field multipliers. Additionally, after calculating of erasure locator polynomial, the initial discrepancy of erasure locator polynomial and syndrome vector must be computed as the inversionless Berlekamp-Massey algorithm coefficient. This step also needs additional penalty to realize. Therefore, to achieve a regular and minimum area design, the modified inversionless Berlekamp-Massey algorithm with erasure locator polynomial expansion is shown as follows.

$\Lambda^{(b)}(x) = 1, \quad \Lambda^{(a)}(x) = 1,$

**1) Initially**  $l = 0, \quad k = 1, \quad \gamma^{(k)} = 1, \quad decode = 0$

**2) If  $k < s$  decoder=0, set**  $\delta = 1, \quad \gamma = Z_k, \quad \Lambda^{(a)}(x) = x\Lambda^{(b)}(x) = x\Lambda(x)$

**Compute**  $\Lambda^{(c)}(x) = \gamma\Lambda^{(b)}(x) + \delta\Lambda^{(a)}(x) = (1 + Z_k x)\Lambda(x)$

**and**  $\delta = \sum_{j=0}^l \Lambda_j^{(c)} S_{k-j}$

**Set  $k=k+1,$**

**If  $k < s$  repeat step (2), Else set decoder =1 and go step (3)**

**3) Compute (a)**  $\Lambda^{(a)}(x) \leftarrow x\Lambda^{(a)}(x)$  **and**  $\delta = \sum_{j=0}^l \Lambda_j^{(b)} S_{k-j}$

**(b)**  $\Lambda^{(c)}(x) = \gamma\Lambda^{(b)}(x) + \delta\Lambda^{(a)}(x)$

**(c) If  $\delta \neq 0$  and  $2l \leq k-1$**

**Set**  $\Lambda^{(a)}(x) = \Lambda^{(b)}(x), \quad l = k-l, \quad \gamma = \delta$

**(d)**  $\Lambda^{(b)}(x) = \Lambda^{(c)}(x)$

**Set  $k = k+1.$  If  $k < d,$  then go step 3.**

**4) Stop**



## Computation for Errata Evaluator Polynomial

The paper [13] has also indicated that the errata evaluator polynomial can be computed by decomposed architecture. After the errata locator polynomial is obtained, the errata evaluator polynomial can be derived as following:

$$\begin{aligned}
 W(x) &= S(x)\Lambda(x) \bmod x^{2t} \\
 &= W^{(0)} + W^{(1)}x + \dots + W^{(v-1)}x^{v-1} \\
 W^{(i)} &= S_{i+1}\Lambda_0 + S_i\Lambda_1 + \dots + S_1\Lambda_i
 \end{aligned} \tag{4.6}$$

where the  $v$  is the degree of the errata locator polynomial and the  $W^{(i)}$  represents the coefficient of the errata evaluator polynomial. To compute the errata evaluator polynomial is similar to compute the discrepancy, which also requires a multiply-and-addition hardware to implement. The errata evaluator polynomial also can be decomposed like calculating discrepancy, which is show as follows:

$$W_j^{(i)} = \begin{cases} S_{i+1}\Lambda_0, & \text{for } j = 0 \\ W_{j-1}^{(i)} + S_{i-j+1} \cdot \Lambda_j, & \text{for } 1 \leq j \leq i \end{cases} \tag{4.7}$$

Obviously, this decomposed format is same to compute the discrepancy (equation (4.7)). Hence, the same hardware is used to solving the error evaluator polynomial after obtaining the errata locator polynomial.

### 4.3 Chien search and Error Evaluator Block

#### Chien search block

The Chien search is used to check the roots of errata locator polynomial which equals to zero or not. If  $\Lambda(\alpha^{-i})=0$ , this is represent that there is an error or erasure at the  $i$ -th location of received codeword. Similar to the syndrome block, the Chien search and error evaluator block

are also implemented through the Horner's rule. However, for universal Chien-search architecture design, the dedicated FFM is replaced with universal FFM for each cell. Fig. 4.5(a) shows the circuit of the Chien search cell. Because of the maximum field of our proposed design is ten, the one stage FIFO buffer must have length 1024x10 bits and costs large area. For reducing the FIFO buffer length, the double check Chien search is used to find the roots twice at a time. Fig. 4.5(b) shows the entire Chien structure with  $n-k$  Chien search cells.

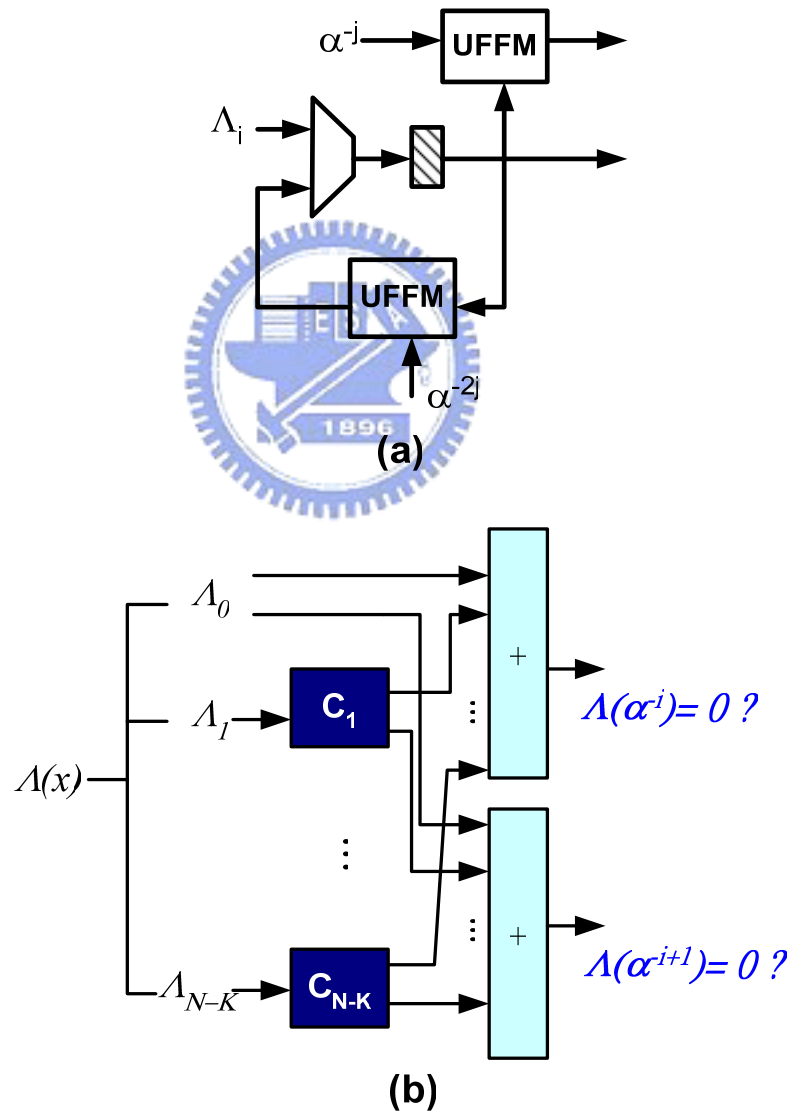


Figure 4.5: (a) the double check Chien search cell. (b) Chien search architecture with correctable erasure is  $n-k$ .

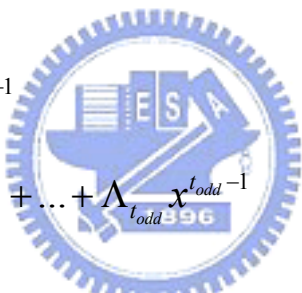


## Error value evaluator

The Foney algorithm mentioned in Chapter 2 is used to evaluate the error value. The Foney algorithm can be expressed as follows:

$$e_l = \frac{W(\beta_j)}{\Lambda'(\beta_j)} \quad (4.8)$$

where the  $\beta_j$  indicates the root of errata location polynomial  $\Lambda(x)$  and the  $\Lambda'(x)$  is represent the first derivative of  $\Lambda(x)$ . In finite field arithmetic, the derivative can be replaced with simple format which is composed of original odd coefficient. It is shown as follows:

$$\begin{aligned} \Lambda'(x) &= \left( \sum_{k=1}^l \Lambda_k x^k \right)' \\ &= \sum_{k=1}^l k \Lambda_k x^{k-1} \\ &= \Lambda_0 + \Lambda_3 x^2 + \dots + \Lambda_{t_{\text{odd}}} x^{t_{\text{odd}}-1} \\ &= \frac{1}{x} \Lambda_{\text{odd}}(x) \end{aligned} \quad (4.9)$$


where the  $t_{\text{odd}}$  represents the maximum degree of  $\Lambda(x)$ . Hence, the Foney algorithm can be rewritten as:

$$e_l = \frac{W(\beta_j)}{\Lambda'(\beta_j)} = \frac{W(\beta_j) \cdot \beta_j}{\Lambda_{\text{odd}}(\beta_j)}, \quad \text{for } j = 1 \sim t \quad (4.10)$$

There are two solutions to realize the error evaluator block. One is parallel approach which is similar as the Chien search architecture, and another serial structure is using one FFM to implement. However, the large FIFO buffer length is the penalty of serial architecture. Fig. 4.6 shows the serial error evaluator architecture. Where the UFFI represents a universal finite field inverter and  $\beta_j$  indicates the roots of errata locator polynomial. The finite field

inverter is used to calculate the inversion of finite field elements. This architecture can calculate the  $W(\beta_j)$  and  $1/\Lambda(\beta_j)$  at same time.

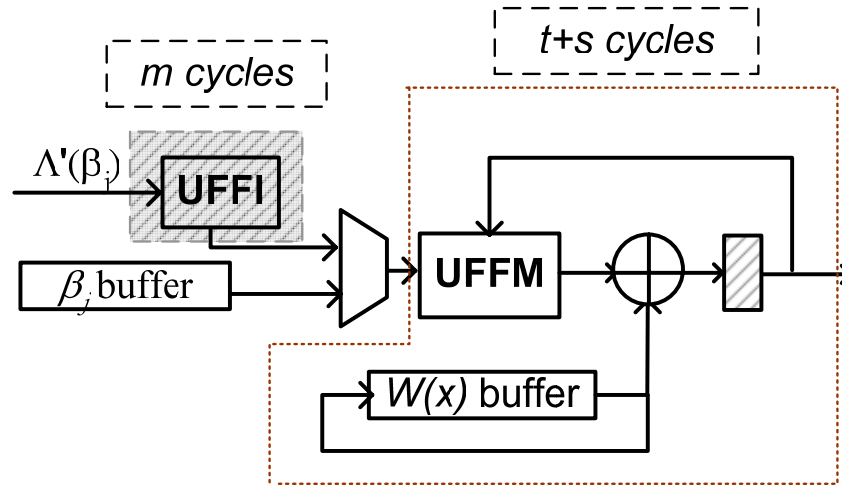


Figure 4.6: The serial error evaluator architecture with one FFM.

#### 4.4 Summary



In this chapter, the universal RS erasure decoder is proposed. If a  $(n, k, t, m)$  universal erasure RS decoder is designed. The any  $(n', k', t', m')$  RS code with  $n' \leq n, k' \leq k, t' \leq t, m' \leq m$  parameters can be decoded by our proposed architecture.

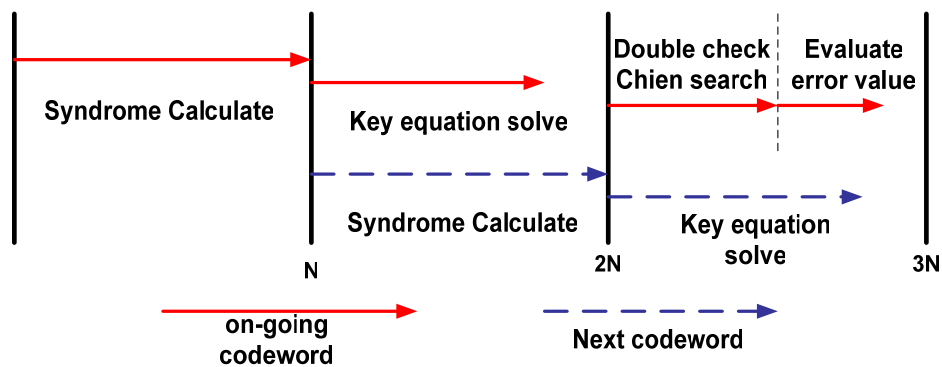


Figure 4.7: The Timing Diagram for propose I architecture.

The decoding timing scheme of proposed RS decoders is shown in fig. 4.7. As the fig.4.7 shows, the double check Chien search is used to reduce the search cycles and serial error evaluator architecture can be applied. And, the finite field inverter of proposed I architecture is based on Fermat's algorithm.

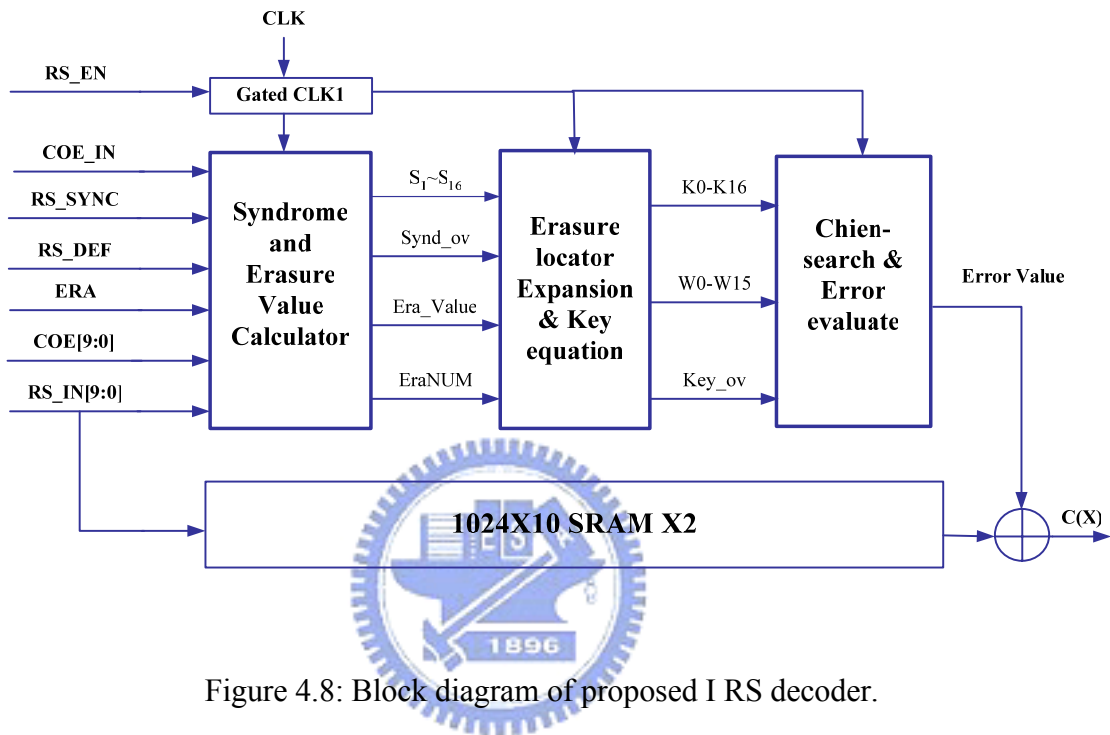


Figure 4.8: Block diagram of proposed I RS decoder.

Fig 4.8 shows the block diagram of proposed architecture I. The proposed architecture I can support the maximum field degree to 10, and the corrective error is 8. Two 2048x10 SRAMs are used to store the received codeword. Because the syndrome cell and Chien search cell are implemented by universal FFM, the total gate count of proposed architecture I is large. Hence, for implementing the error evaluator block and finite field inverter, the serial computation architectures is used.

# CHAPTER 5

## Area Efficient Design Approach

The RS erasure decoder consists of syndrome calculator, erasure locator polynomial expansion, key-equation solver, Chien-search block and error-value evaluator. However, in proposed architecture I, the design has larger design-cost than typical single mode RS decoder. In proposed architecture II, constant universal finite field multiplier will be used to reduce the gate count. In this chapter, the modified syndrome calculator is introduced in subsection 5.1 and the parallel Chien search and error evaluator architecture is shown in subsection 5.2. The parallel Chien search and error evaluate architecture can search the error and erasure roots and calculate the error value simultaneously. The key equation architecture is same as proposed architecture 1. Because of the implementation of constant universal FFM, the total area cost of syndrome block and Chien search block is improved obviously. Finally, the improving of the decoder function that can correct 16 errors is described in subsection 5.3.

### 5.1 Universal Syndrome and Erasure Value Calculator

The syndrome value represents the error information of received codeword. The Horner's rule is applied to compute the syndrome value. Hence, the typical substitution form of syndrome value is shown as following:

$$\begin{aligned} S_i &= R(\alpha^i) \\ &= (..((R_{n-1}\alpha^i + R_{n-2})\alpha^i + R_{n-3})\alpha^i + \dots R_2)\alpha^i + R_1 \quad \text{for } i=1\sim 2t \end{aligned} \quad (5.1)$$

It is because that the universal finite field multiplier cost larger area than a dedicated constant multiplier. In order to achieving area efficient design, a constant UFFM (CUFFM)

can be constructed by replacing one input value of UFFM with fixed finite field element  $x^j$ . It can be expressed as following:

$$\begin{aligned}
\overline{C}(x) &= A(x)B(x)r^*(x) \bmod p(x) \Big|_{A(x)=x^j} \\
&= x^j B(x)x^{-m} \bmod p(x) \\
&= [0 + [..[0 + [1 \cdot B(x)x^{-1} + [..[0B(x)x^{-1}]...]]x^{-1} \\
&\quad \bmod p(x)]]x^{-1} \bmod p(x)]...]x^{-1} \bmod p(x) \Big|_{x=\alpha} \\
&= \alpha^{(j-m)} B(x) \bmod p(x)
\end{aligned} \tag{5.2}$$

It is shown that the  $\alpha^{(i-m)}$  represents a constant UFFM (CUFFM) function.

However, for adapting to constant UFFM function, the original substitution of syndrome polynomial must be modified. According the Horner's rule, each codeword symbols must multiply the constant  $\alpha^{m*n}$  before entering the syndrome cell, where  $n$  represents the location of codeword symbols. The following equation indicates the detail modified syndrome substitution procedure.

$$\begin{aligned}
\alpha^m R(\alpha^i) &= \alpha^m R_{n-1} \alpha^{i*n-1} + \alpha^m R_{n-2} \alpha^{i*n-2} + \dots + \alpha^m R_0 \\
&= \alpha^m R_{n-1} \alpha^{(m+(i-m))*n-1} + \alpha^m R_{n-2} \alpha^{(m+(i-m))*n-2} + \dots + \alpha^m R_0 \\
&= (\dots((R_{n-1} \alpha^{m*n} \alpha^{(i-m)} + R_{n-2} \alpha^{m*n-1}) \alpha^{(i-m)} + \dots)) \alpha^{(i-m)} + R_0 \alpha^m
\end{aligned} \tag{4.4}$$

Based on above equation form, the modified syndrome calculator is constructed in Fig 4.1. Fig. 5.1(a) indicates each cell of syndrome architecture and Fig. 5.1(b) shows the entire universal syndrome and erasure value calculator block with correctable erasure is 16. Since the area and critical path of CUFFM increase in proportion to the minus degree of  $\alpha^{(i-m)}$ , the maximum minus degree of CUFFM is kept at eight.

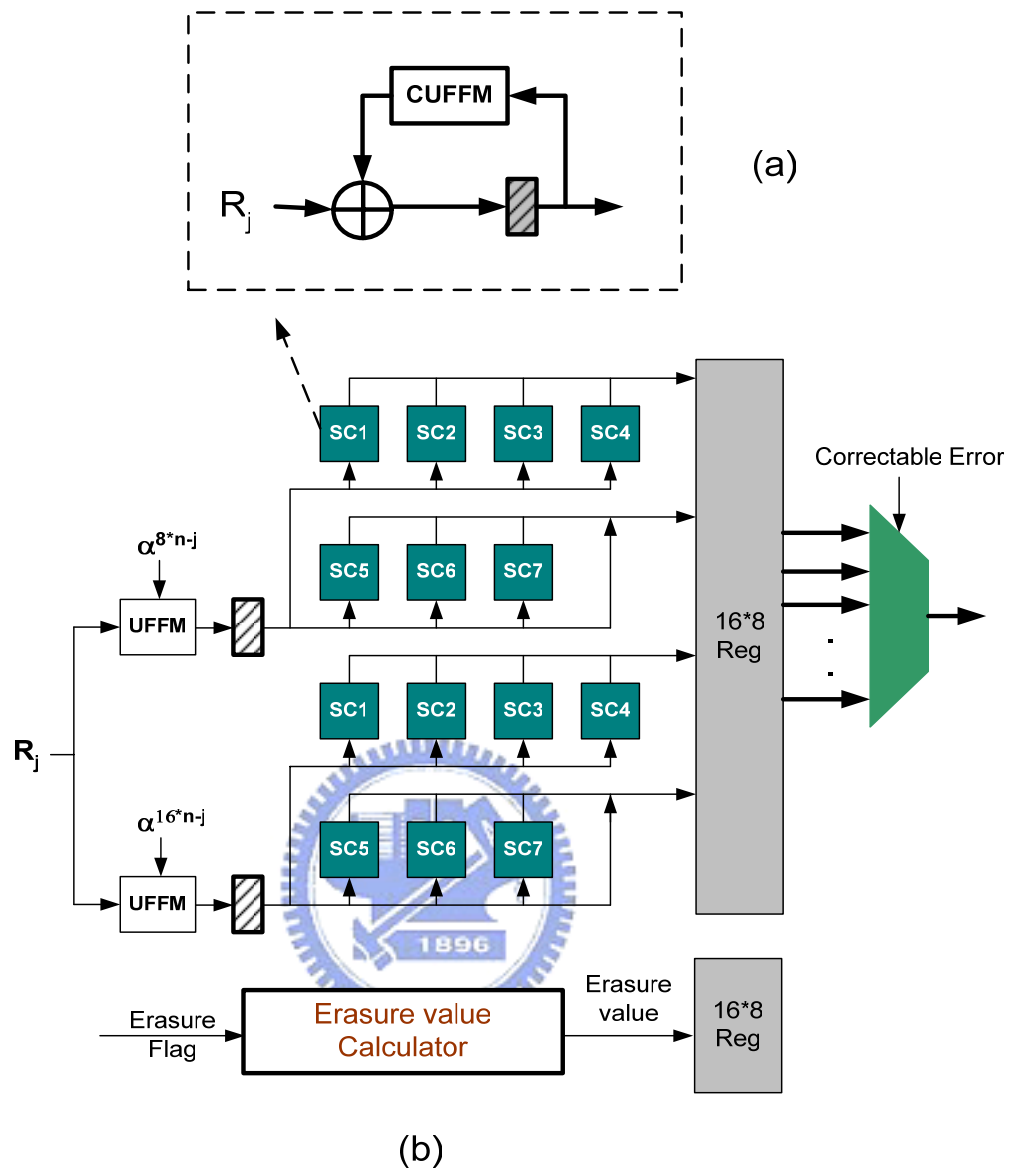


Figure 5.1: The universal Syndrome and Erasure Value Calculator

## 5.2 Chien search and Error Evaluator Block

For area efficient design, the universal FFM can be replaced with constant universal FFM in each cell of Chien search and error evaluator block. Since the area and critical path of CUFFM increase with the minus degree of  $\alpha$ , the errata polynomial form must be modified to

avoid larger minus degree. Assume the correctable erasure is 16, the modified errata polynomial form is shown as follows;

$$\begin{aligned}
 \Lambda(\alpha^{-i}) &= \Lambda_0 + \Lambda_1(\alpha^{-1})^i + \Lambda_2(\alpha^{-2})^i + \dots + \Lambda_8(\alpha^{-8})^i + \\
 &\quad \Lambda_9(\alpha^{-9})^i + \Lambda_{10}(\alpha^{-10})^i + \dots + \Lambda_{16}(\alpha^{-16})^i \\
 &= \Lambda_0 + \Lambda_1(\alpha^{-1})^i + \dots + \Lambda_8(\alpha^{-8})^i + \\
 &\quad (\alpha^{-8})^i \{ \Lambda_9(\alpha^{-1})^i + \Lambda_{10}(\alpha^{-2})^i + \dots + \Lambda_{16}(\alpha^{-8})^i \}
 \end{aligned} \tag{5.2}$$

From the above modified equation format, the maximum  $\alpha$  minus degree is always 8 and the Chien search block can be implemented easily based on this polynomial form. Fig. 5.2 shows the area efficient Chien-search architecture for  $t=8$ . The cell's output value whose alpha degree is large than 8 will multiply the corrective factor  $\alpha^{-8}$ .

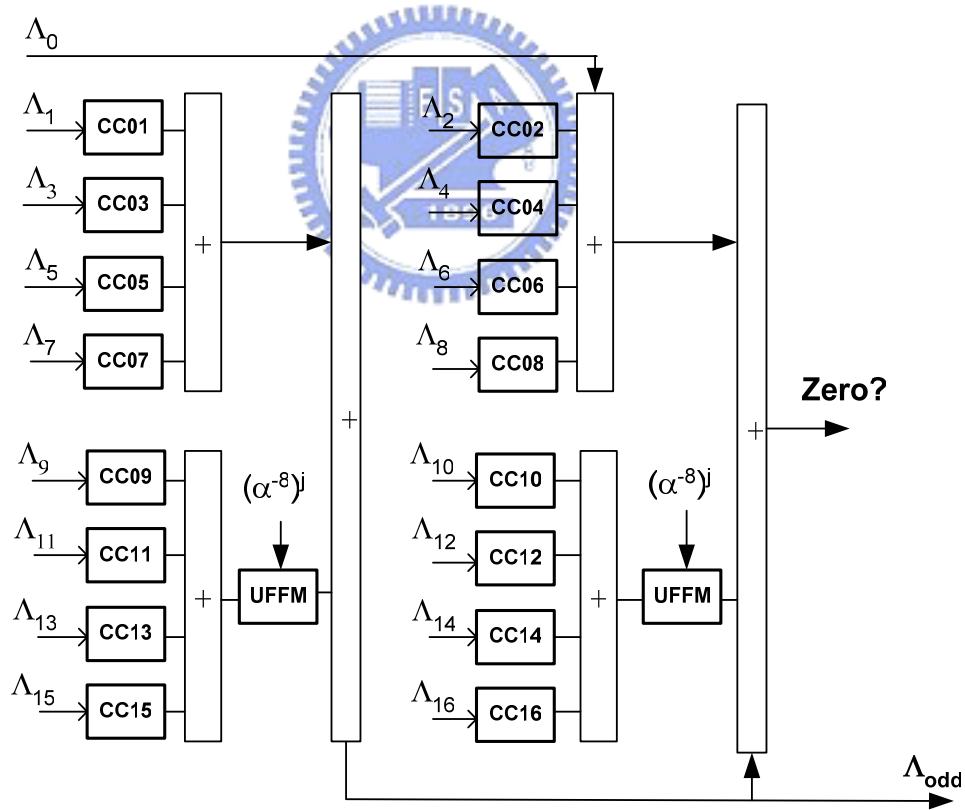


Figure 5.2: The parallel Chien-search block with constant UFFM

For the error evaluator polynomial, its implementation is same as the error locator polynomial that is shown as follows:

$$\begin{aligned}
 \Omega(\alpha^i) &= \Omega_0 + \Omega_1(\alpha^{-1})^i + \Omega_2(\alpha^{-2})^i + \dots + \Omega_8(\alpha^{-8})^i + \\
 &\quad \Omega_9(\alpha^{-9})^i + \Omega_{10}(\alpha^{-10})^i + \dots + \Omega_{16}(\alpha^{-16})^i \\
 &= \Omega_0 + \Omega_1(\alpha^{-1})^i + \dots + \Omega_8(\alpha^{-8})^i + \\
 &\quad (\alpha^{-8})^i \{ \Omega_9(\alpha^{-1})^i + \Omega_{10}(\alpha^{-2})^i + \dots + \Omega_{16}(\alpha^{-8})^i \}
 \end{aligned} \tag{5.3}$$

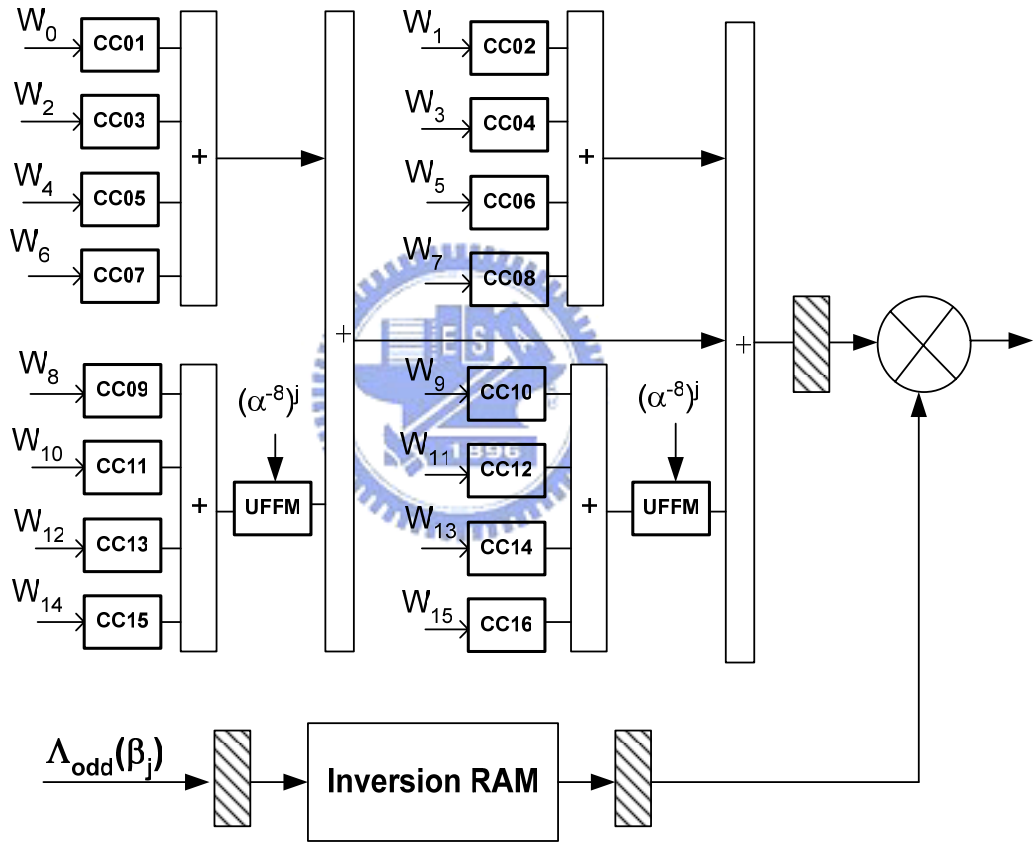


Figure 5.3: The parallel error evaluator block with constant UFFM

Figure 5.3 shows the parallel error evaluator architecture. The inversion RAM which will be described in next subsection is used to store the corresponding finite field inverse. Each cell is constructed by one const universal FFM. As compared with serial architecture in Fig. 4.6, because the function of Chien search and error evaluate can be performed at same time,



this architecture will reduce one stage FIFO buffer. However, it totally costs  $N$  cycles to operate the error evaluator function.

### 5.3 $8 \leq t \leq 16$ Error-only Correction

Since the proposed design supports the maximum 16 correctable erasure, it can be configured to correct 9~16 errors without any erasure. The basic idea is calculating the syndrome twice which costs  $2n$  ( $n$  is the block length) cycles. At first  $N$  cycles, the syndrome  $S_1 \sim S_{16}$  will be calculated. However, if the first half of syndrome  $S_1 \sim S_{16}$  are equal zero, the  $S_{17} \sim S_{32}$  will all equal zero, and the following decoding process, includes 16 syndrome calculation, key equation solve, and Chen search block can be terminated. Based on this property, the power consumption can be reduced significantly. If syndrome  $S_1 \sim S_{16}$  are not equal zero, the syndrome  $S_{17} \sim S_{32}$  will be executed. The syndrome block will read the received codeword again from the FIFO buffer, and the next codeword will be hold. Fig 5.4 shows the hardware structure between syndrome block and FIFO buffer, and fig. 5.5 indicates the decoding procedure of 16 error-only correction.

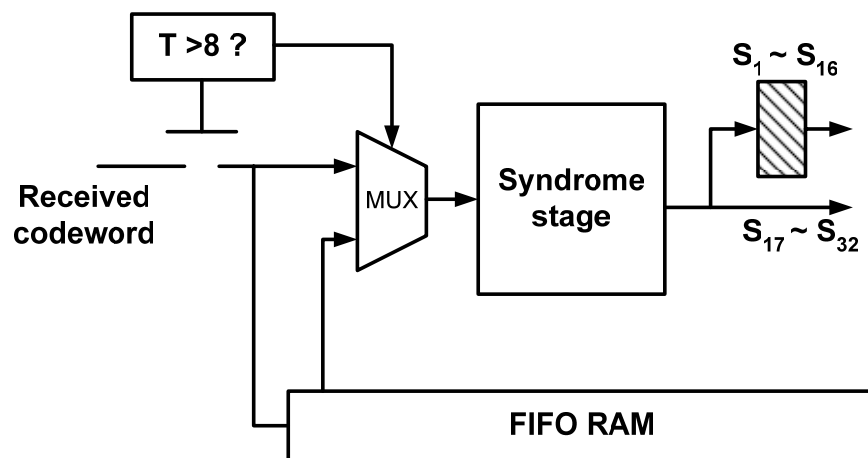


Figure 5.4: Block diagram of 16 errors correction.

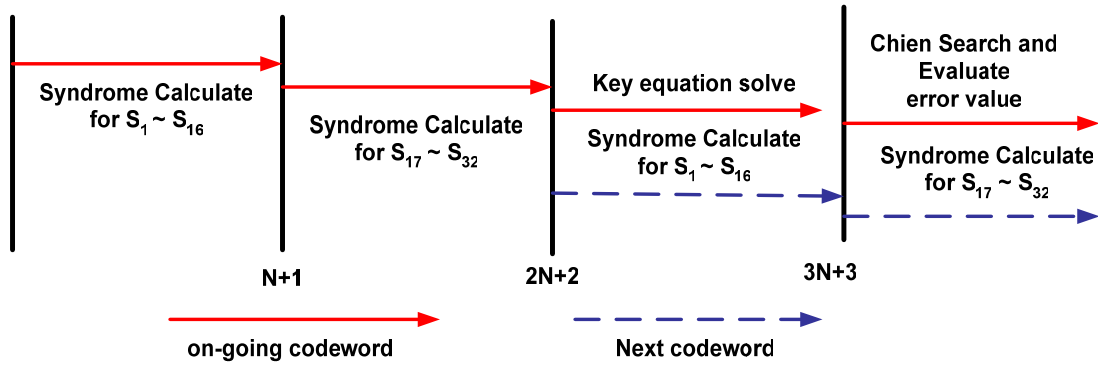


Figure 5.5: Decoding timing diagram for 16 errors-only correction.

### 5.5 Summary

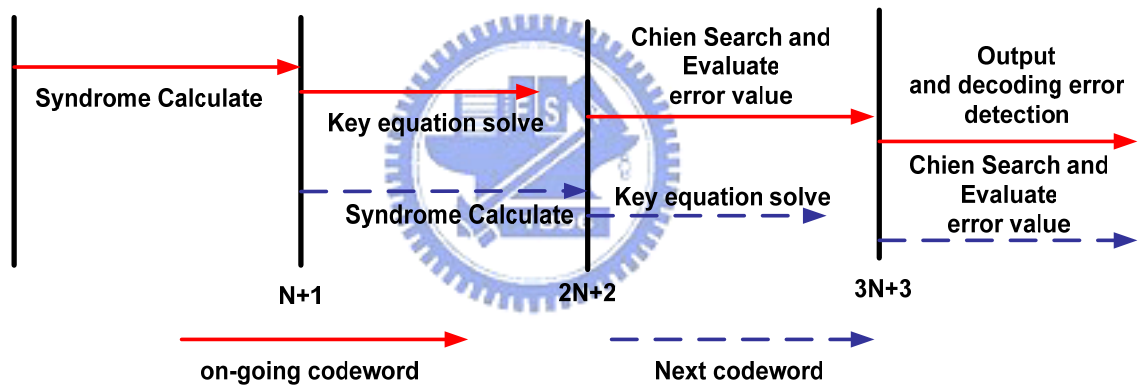


Figure 5.6: The Timing Diagram for propose II architecture.

In this chapter, the area efficient architecture of universal RS erasure decoder is introduced. The proposed architecture II can support the maximum field degree to 8, and the corrective error is 16. The decoding timing scheme of proposed decoder II is shown in fig. 5.6. As this figure shows, the maximum latency is  $4N+4$  and two  $512 \times 8$  SRAMs are applied.

Fig. 5.7 shows the block diagram of proposed architecture II. The area efficient approach is adopted for implementing the proposed II architecture. A  $256 \times 8$  SRAM is used to realize the on-the-fly inversion table, and parallel Chien-search and error evaluate block is adopted for

high speed computation. Besides, some methodology like gated CLK circuit is adopted to reduce the power consumption. The on-the-fly SRAM can support the parallel chien search and error evaluate block with high speed computation. Additionally, more power consumption issue is considered in proposed architecture II.

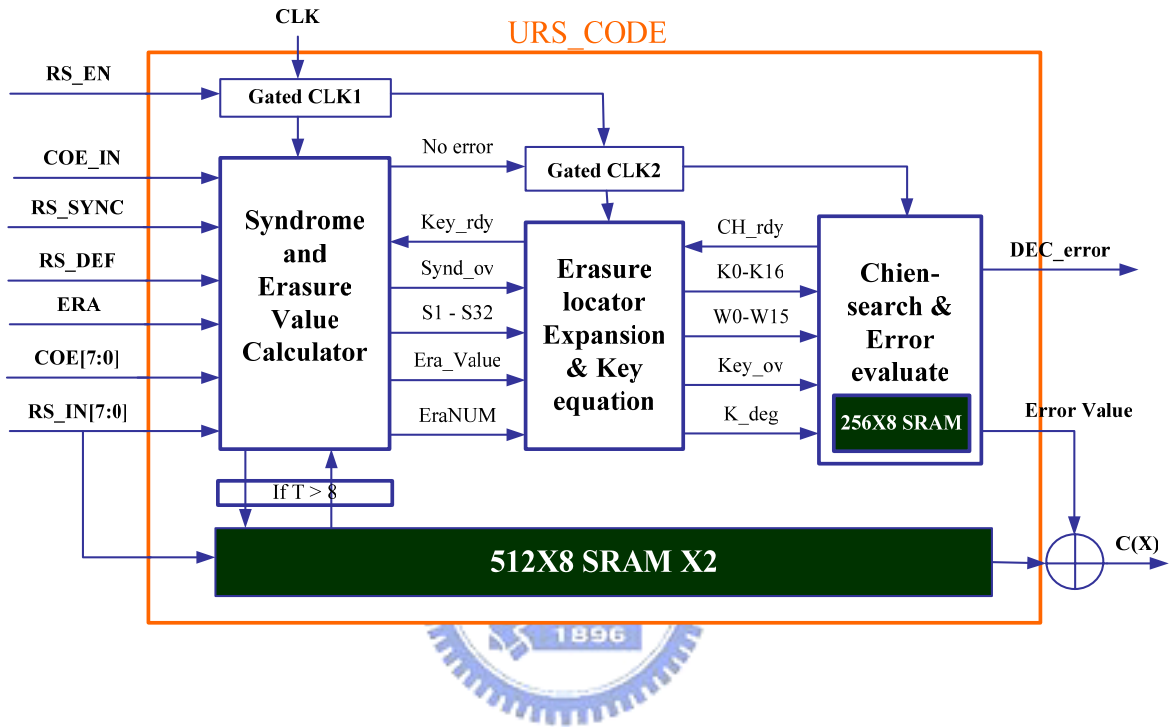
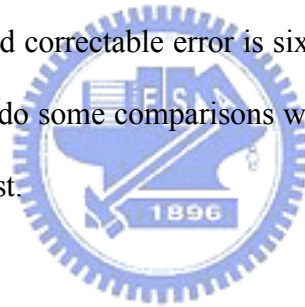


Figure 5.7: Block diagram of proposed II RS decoder.

# CHAPTER 6

## Chip Implementation Result

This chapter will describe the CHIP implementation and its design methodology. In subsection 6.1, we will describe the design and test consideration. Then, there are two implementations of proposed universal RS erasure decoders are shown in subsection 6.2 and 6.3. The proposed I architecture can support the field degree to ten and correctable errors to eight. The proposed II architecture implemented by area efficient approach can support the maximum field degree eight and correctable error is sixteen. Besides, the simulation result of two proposed architecture will do some comparisons with other single-mode or reconfigurable RS decoder published in the past.



### *6.1 Design and Test Consideration*

Fig. 6.1 presents the entire design and testing flow with various CAD tools. At first, we can use the high level language like C/C++ or MATLAB to construct the software simulation environment and generate a lot of random codewords with AWGN noise. Hence, after the RTL coding, the hardware-software co-simulation ensure the correction of behavior model. Fig 6.2 shows the relation of hardware-software co-simulation.

The verilog description language is chosen as the RTL implementation. After the RTL level, the gate level implementation will be performed by Synopsys Design Analyzer synthesis tools. And, the synthesis standard library of proposed architecture is 0.13mm 1P8M CMOS

technology. The clock rate and the performance in  $0.13\mu m$  technology are improving significantly. And, the memory size of FIFO buffer and inversion table decreases obviously. After the gate level synthesis, the pre-layout simulation will be performed to verify the gate level performance. In deep submicron process, the wire delay plays an important role of circuit speed. Hence, the pre-layout simulation can not calculate the circuit speed precisely. Besides for pre-layout simulation with nc-verilog complier, the primetime is also an effective CAD tool to calculate the critical path.

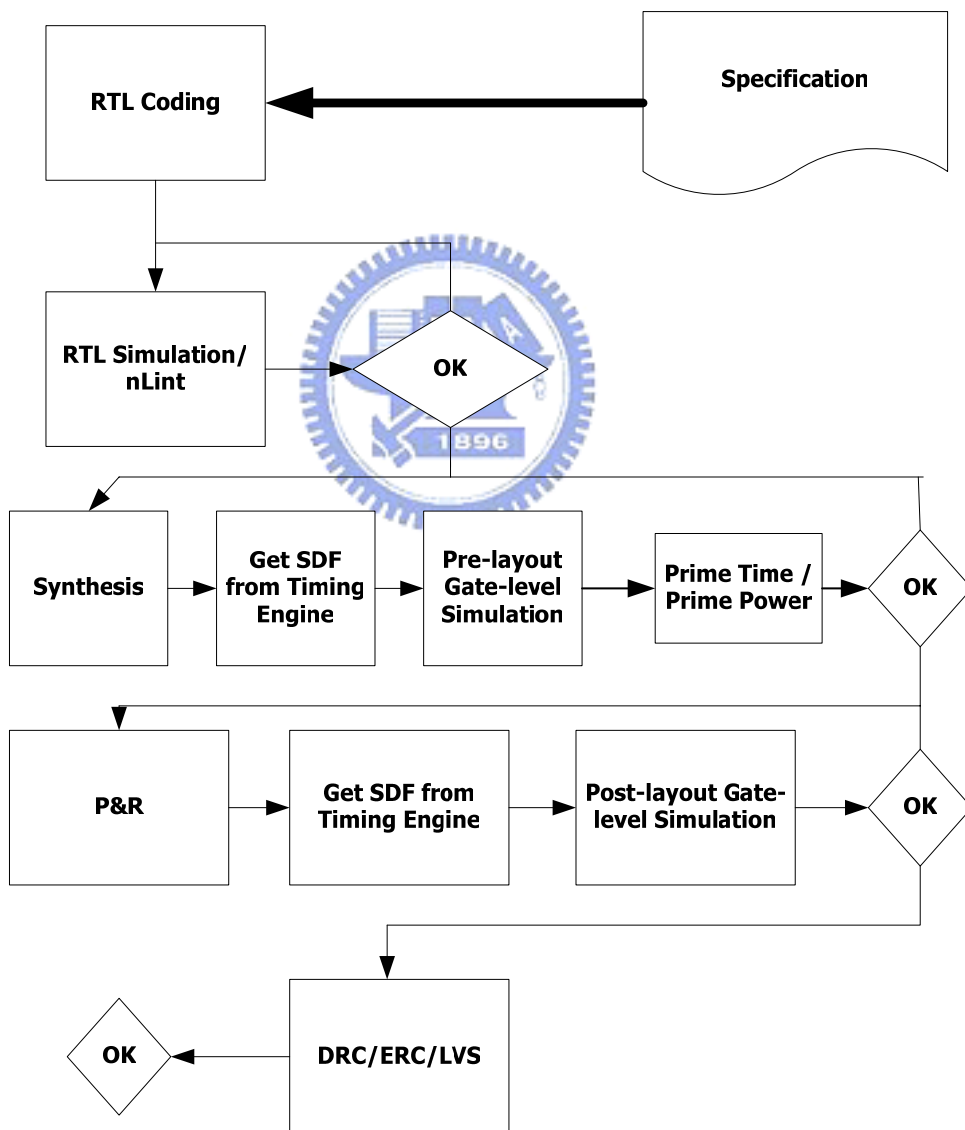


Figure 6.1: The entire design flow

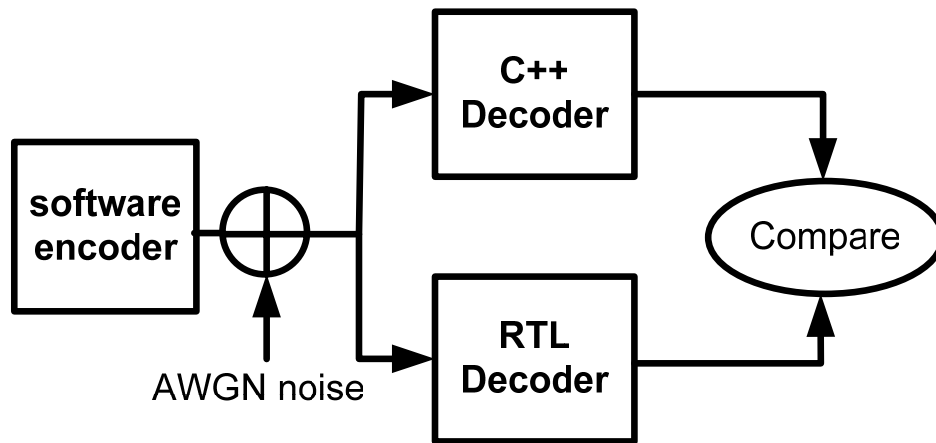


Figure 6.2: The simulation environment

For successful pre-layout simulation, the place and route will be performed through the SOC Encounter tool. In deep-submicron design, many problems like signal integrity (SI), IR drop, and wire delay must be considered carefully. The power consumption, RC extraction, and timing estimation will be computed exactly at place and route procedure. Finally, the post layout simulation includes DRC (design rule check) and LVS (layout versus schematic) can verify the chip layout integrity.

## 6.2 CHIP Implemenation for Proposed Architecture 1

The proposed I architecture can support the maximum field degree to 10, and the maximum correctable error is 8 (maximum correctable erasure is 16). In syndrome block, the universal FFM is applied in 16 syndrome cell. In Chien search block, the double check architecture is used to reduce the search cycles. And, the error evaluator block is designed by the serial architecture. Two 2048x10 SRAMs is used to realize the FIFO buffer, and the inverter is implemented based on Fermat algorithm. In this design, the circuit complexity

isn't considered that syndrome cell and Chien cell are implemented by universal FFM. This architecture is implemented by 0.13 $\mu\text{m}$  1P8M standard cell technology. The critical path of synthesized gate level model exists in the key equation block. Fig 6.3 shows chip die photo of proposed I architecture.

Table 6.1 shows the chip summary of proposed decoder I. The total gate count is about 110K and the maximum clock rate is 222 MHz. And, the core size is 1.25 x 0.63 mm<sup>2</sup>. The maximum power consumption is 23mW at clock rate 222 MHz. The chip is packaged in a 84 CLCC package.

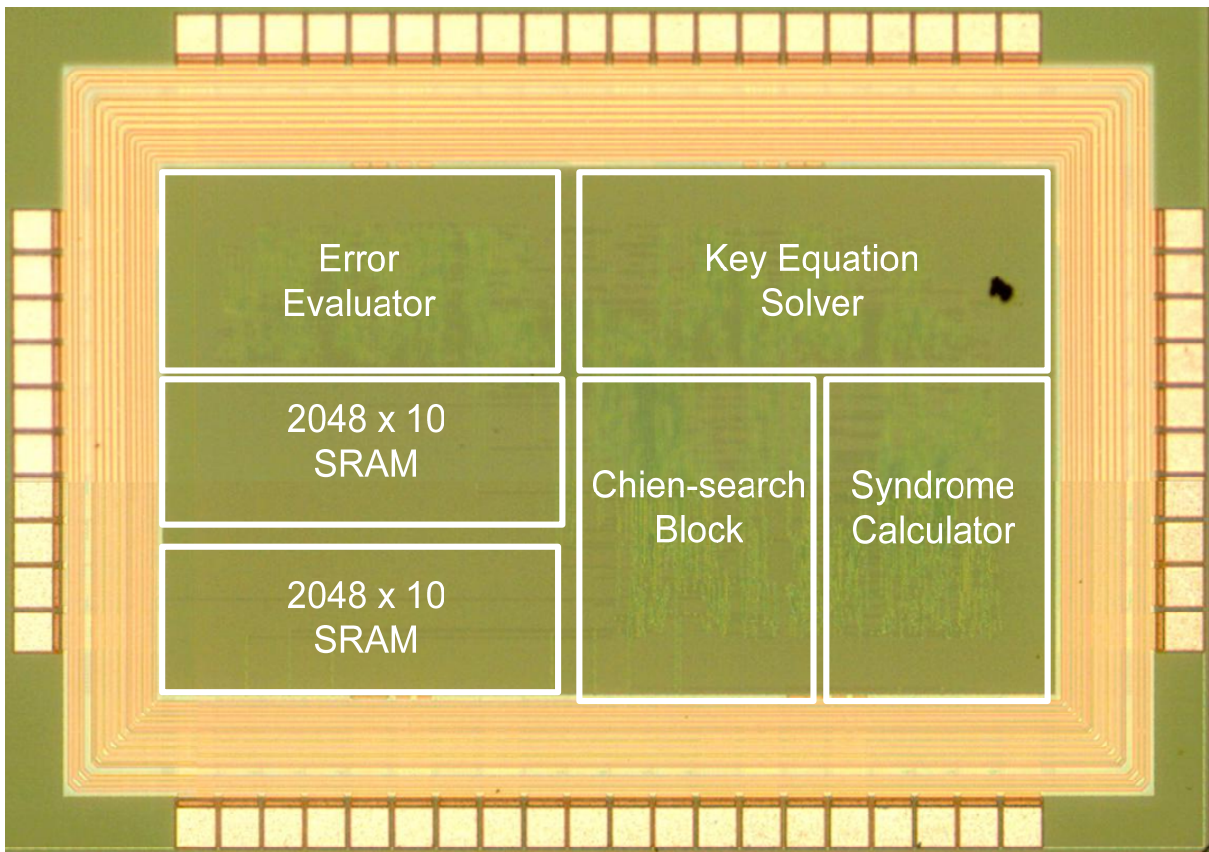


Figure 6.3: The die photo of proposed I architecture

Table 6.1: The chip summary of proposed I universal RS decoder.

Design	Universal RS Erasure Decoder
maximum field degree	10
Corrective error	1 ~ 8
Memory size	40 K bits
Core area (mm <sup>2</sup> )	0.78
Total gate count	75K + 35K FIFO RAM
Maximum Operating Frequency	220 MHz
Date rate (M bits/s)	2200
Average Power (supply voltage)	23.2 (1.2V)

### ***6.3 CHIP Implemenation for Proposed Architecture II***

The proposed II universal RS erasure decoder is implemented by area efficient design. This architecture can support the maximum field degree to 8, and the maximum correctable error is 16 as well as maximum correctable erasure. Two 512x8 SRAMs is used to realize the FIFO buffer, and a 256x8 SRAM is used to construct the finite field inversion table. The error evaluator block is designed by the parallel architecture which performs the Forney



algorithm at Chien search stage. For circuit complexity consideration, that syndrome cell and Chien cell are implemented by constant UFFM. Therefore, the total area of entire RS code has smaller overhead than a single mode RS decoder.

This decoder is also implemented by 0.13mm 1P8M standard cell technology. Fig. 6.4 shows layout view of proposed decoder II. The critical path of synthesized gate level model also exists in the key equation block.

Table 6.2 shows the chip summary of proposed II decoder. The total gate count is about 54K with FIFO buffer 14K, and the maximum clock rate is 300 MHz. And, the core size is 0.36 mm<sup>2</sup>. The maximum power consumption is 20.2mW at clock rate 222 MHz. The chip is packaged in a 68 CLCC package.

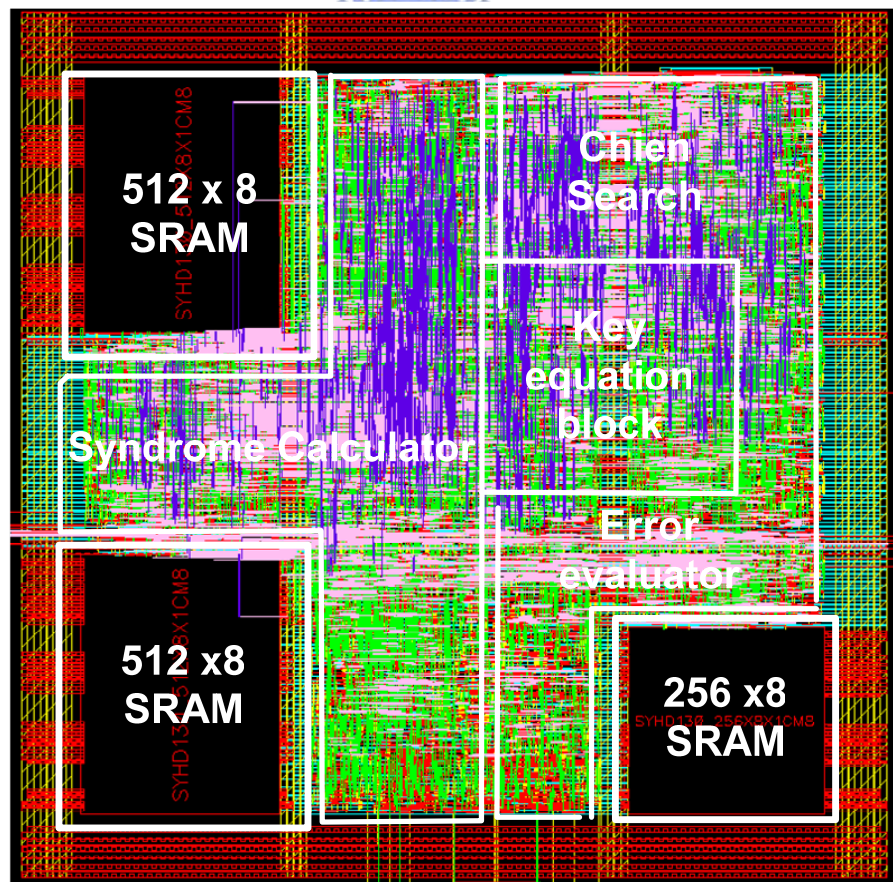


Figure 6.4: The layout view of proposed II architecture

Table 6.2: The chip summary of proposed II architecture

Design	Universal RS Erasure Decoder
maximum field degree	8
Corrective error	1 ~ 16
Memory size	10 K bits
Core area (mm <sup>2</sup> )	0.36
Total gate count	39K + 14K FIFO
Maximum Operating Frequency	300 MHz
Date rate (M bits/s)	2400
Average Power (supply voltage)	20.2 (1.2V)

## 6.4 Comparison

Table 6.3 lists various mode RS comparison. From this table, it is obviously that our proposed architecture can support the maximum correctable errors, erasure correction, and the complete reconfigurable capability. As compared the same universal decoder proposed in [23], our proposed decoder can improve about 50 times decoding speed with parallel decoding scheme. Additionally, our proposed design has more flexibility and much higher decoding data rate.


Table 6.3: The comparison of various mode RS decoder

	[21]	[22]	[23]	Proposed I	Propose II
Mode	single	Variable (n, t)	Universal (n, t, m p(x))	Universal (n, t, m p(x))	Universal (n, t, m p(x))
M	8	8	1~8	1~10	1~8
T	8	1~8	1~8	1~8	1~16
Erasure	No	No	No	Yes	Yes
P(x)	Single	Single	Variable	Variable	Variable
Data rate	1600 (200MHz) (parallel)	800 (100MHz) (parallel)	48 (serial)	2200 (220MHz) (parallel)	2400 (300MHz) (parallel)
Gate count	21 K	34K	44K	75K + 35K FIFO	39K + 14K FIFO
Technology	0.25	0.35	0.25	0.13	0.13

# CHAPTER 7

## Conclusion

In this paper, two universal architectures for RS error-and-erasure decoder are presented. The proposed architecture can accommodate variable codeword length, correctable errors, different finite field degrees, and different primitive polynomials. The proposed I architecture can support the maximum field degree to ten, and the corrective error is eight. The proposed II architecture can support the maximum field degree to eight, and the corrective error is sixteen.



To achieve the universal property, the design challenge is to realize a dedicated RS decoder that can accommodate different finite field definition. Hence, the main solution is applying Montgomery multiplication algorithm which described in section 2.1. Based on this algorithm, the universal finite field operator includes multiplier and inverter can be implemented. In consideration of complexity, a universal constant multiplier will be applied in syndrome block and Chien search block to reduce the area size. Besides, we combine the erasure locator expansion and Berlekamp-Massey algorithm to achieve the erasure correction with increasing additional universal FFM.

In design approach view, the software simulation is built first, and then the RTL code can be verified in according to software result. The Verilog description language is chosen as the RTL implementation. After the RTL level, the gate level implementation will be performed with the synthesis standard library of proposed architecture is **0.13 $\mu$ m 1P8M** CMOS technology. Finally, the layout will be constructed by the SOC Encounter software.

# APPENDIX

## Hardware Sharing Design for (528, 518) RS codec IP

In this chapter, an area-efficient Reed-Solomon (RS) codec IP with composite-field inverter is presented. For some specific applications such as flash memory controller, the RS decoder will stop receiving any new codeword until the on-going erroneous codeword to be corrected. It is that the circuit complexity can be reduced by sharing the registers and finite-field operation units. The proposed hardware sharing architecture also includes the RS encoder function. Moreover, for area consideration, the composite field inverter is constructed in error evaluator.

### *Proposed Hardware Sharing Architecture*

In flash memory controller, the RS (528, 518) code over  $GF(2^{10})$  is used to mitigate the errors that may be introduced during manufacturing or by user damage. Note that there are totally 518 message bytes in each codeword of 528 coded bytes. Since the specified RS code is constructed over  $GF(2^{10})$ , these 10 parity-checking bytes imply that the number of correctable errors is 4.

In this section, firstly the RS encoder & syndrome calculator, key-equation solver, as well as Chien-search & error-value evaluator are introduced in following subsections. Then the hardware sharing architecture will be addressed to optimize the usage of registers and operation units.

By means of linear system theory transformations, Fettweis proposed a combined methodology to implement both the RS encoder block and the syndrome calculator [24]. In key equation solver, the decomposed inversionless Berlekamp-Massey architecture uses 3

finite-field multipliers (FFM) without any finite-field inverter (FFI). However, one FFI is always needed in the error-value evaluator block. Thus the key-equation solver is implemented according to the Berlekamp-Massey algorithm within two FFMs and one FFI in our hardware sharing architecture. Furthermore, the composite-field is introduced to realize the FFI since the look-up table for  $GF(2^{10})$  cost too much circuit complexity [26].

#### A. RS Encoder & Syndrome Calculator

The RS encoder & syndrome calculator block are combined according to [24]. Figure A.1(a) shows the combined circuit of RS encoder & syndrome calculator for  $t=4$ , and figure A.1(b) is the function cell( $SC_i$ ) of figure A.1(a). Additionally, this combined circuit uses eight syndrome registers  $s_1 \sim s_8$ .

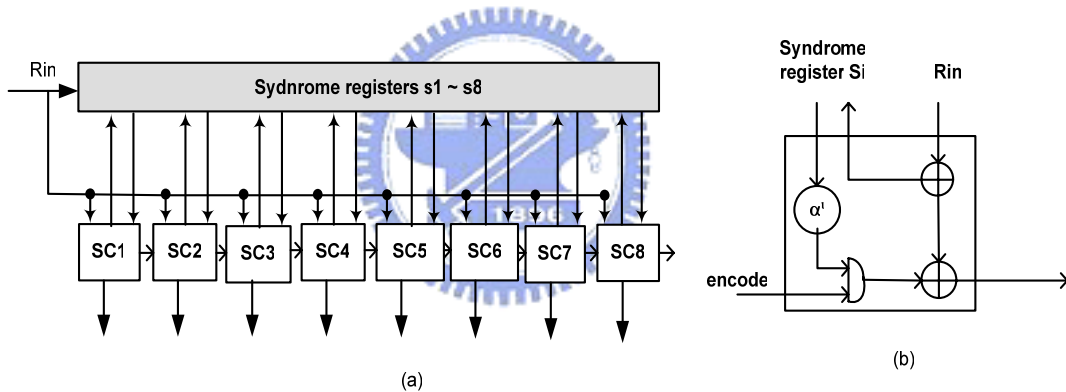


Figure A.1: (a) Encoder/Syndrome calculator block, (b) Syndrome cell ( $SC_i$ ).

#### B. Key-Equation Solver

The key-equation solver is used to calculate the error-locator polynomial  $\Lambda(x)$  and error-evaluate polynomial  $\Omega(x)$ . The Berlekamp-Massey algorithm has been mentioned in chapter 2. It is because that the error-value evaluator block needs one finite field inverter. Therefore, for achieving the hardware-sharing design, this inverter can be merged into decomposed key architecture base on Berlekamp-Massey algorithm. Figure A.3 shows the

original decomposed BM architecture with two FFMs [25]. When implementing the original BM algorithm, two FFMs and one FFI are needed.

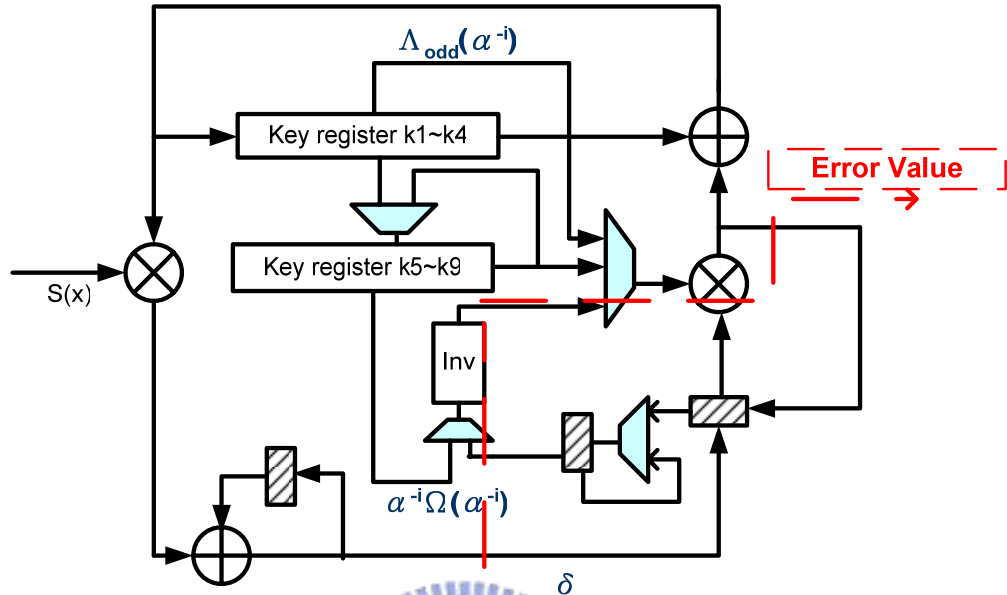


Figure A.2: The decomposed Berlekamp-Massey architecture with finite field inverter.

### C. Chien-search Block and Error-value Evaluator

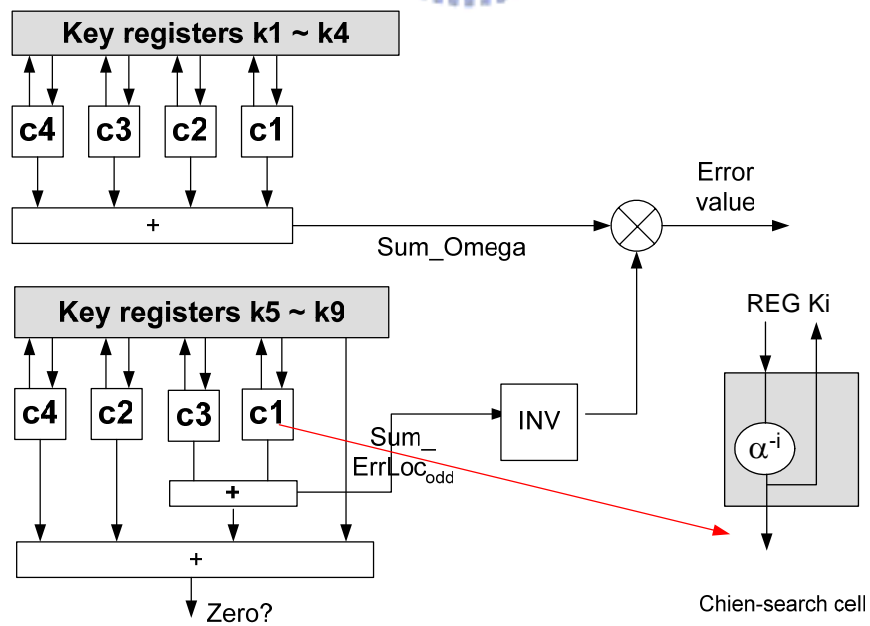


Figure A.3 : Chien-search and error-value evaluator block.

Similar to the syndrome calculator, the Chien-search & error-value evaluator block can also be realized through the Horner's rule. Figure A.3 is the Chien-search and error-value evaluator block architecture for  $t=4$ .

Through using one FFI and one FFM, the Forney algorithm, which is used to obtain the error value, can be realized. When performing the Chien-search and error-value evaluator, the data flow of inverter and multiplier is similar to the dotted line shown in Figure A.2.

*D. The Hardware Sharing Architecture*

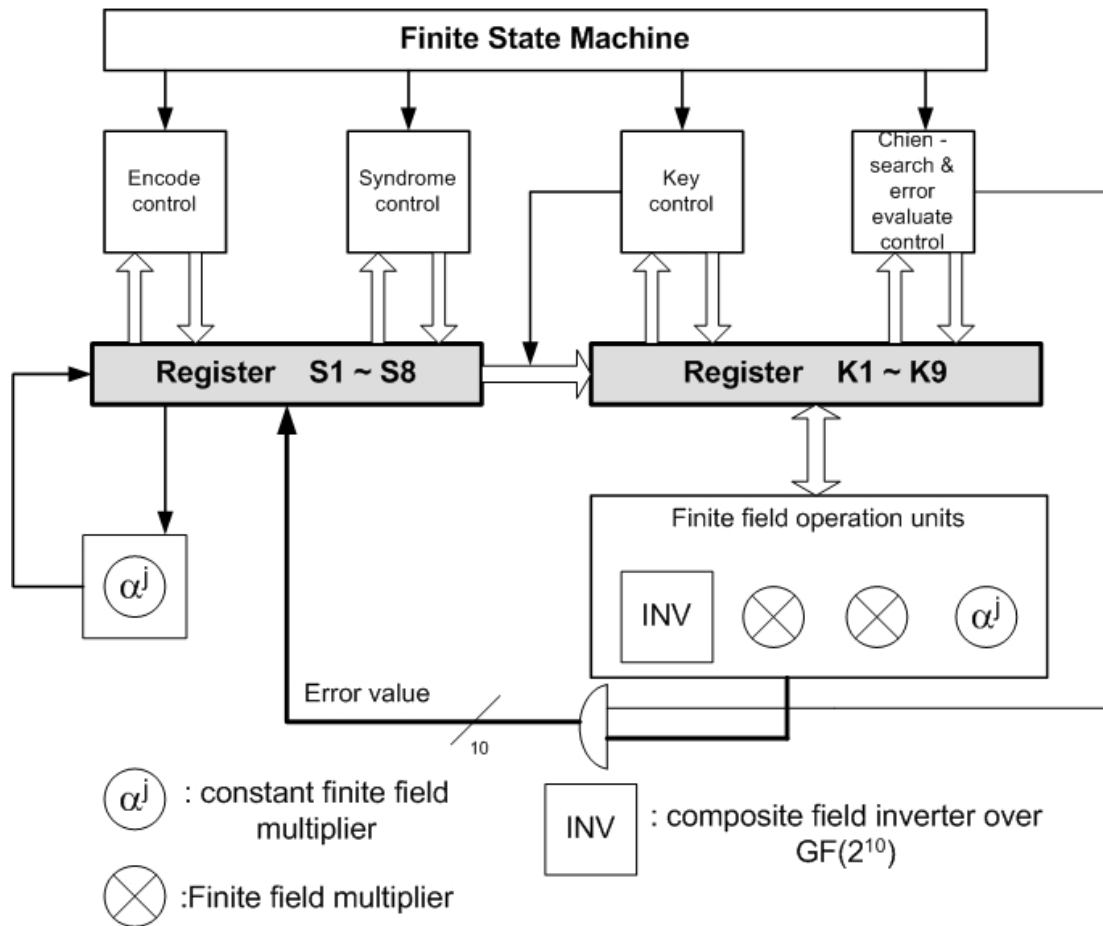


Figure A.4: The hardware sharing architecture.

Figure A.4 shows the entire hardware-sharing block diagram. This architecture integrates the blocks from Figure A.1 to figure A.3 to realize hardware-sharing design. The syndrome registers  $s1 \sim s8$  are used to calculate the parity-checking symbols and syndrome symbols, and



the key registers  $k_1 \sim k_9$  are used to compute the key equation, Chien search and error evaluate operation. The finite field operation units are accessed through the finite state machine controlling. Additionally, the 8 syndrome registers also store the 4 error values and 4 error locations at Chien-search and error evaluate step.

Table A.1 : Comparison of required registers and finite-field function units. The C-S and E-E represent the Chien search block and error evaluator, respectively.

	Syndrome	Key for BM	C-S and E-E	Output Register	Original Design	Our proposed
Register	$2t$	$2t+1$	$2t$	$2t$	$8t+1$	$4t+1$
FFM	0	3	1	0	4	2
FFI	0	0	1	0	1	1

Table A.1 shows the information of register numbers and finite field operation units. Through using hardware sharing architecture, we reduce  $4t$  registers and two finite field multipliers.

### ***Composite Field Inverter***

The composite field is a type of extension field whose subfield is defined over  $GF(2^n)$  rather than  $GF(2)$ . Given a finite field  $GF(2^k)$  where  $k=nm$ , we can construct a isomorphic composite field over  $GF((2^n)^m)$  by introducing a monic primitive polynomial which has order  $m$  and coefficients from  $GF(2^n)$ . Composite field arithmetic is a combination of subfield calculations.

As a result, we can apply composite field to the field  $GF(2^k)$  and implement the  $GF(2^k)$  arithmetic architecture with subfield arithmetic circuits. The complexity of the subfield arithmetic circuits is small so that efficient hardware implementation is achievable.

Finite field inverse over  $GF(2^{10})$  requires a large amount of hardware area in case implemented with a lookup table. Our design employed the composite field  $GF((2^5)^2)$  to the field  $GF(2^{10})$ . This allows us to realize the inverse function with a smaller lookup table over  $GF(2^5)$  [27]. After being mapped to the composite field, every element of  $GF(2^{10})$  can be written as a polynomial of the first degree with coefficients from  $GF(2^5)$ , i.e.,  $bx+c$ ,  $b, c \in GF(2^5)$ . Denoting the primitive polynomial as  $x^2+Ax+B$ , the multiplicative inverse for an arbitrary polynomial  $bx + c$  is given by

$$(bx+c)^{-1}=b(b^2B+bcA+c^2)^{-1}x+(c+bA)(b^2B+bcA+c^2)^{-1}$$

The problem of calculating the inverse in  $GF(2^{10})$  is now translated to calculating the inverse in  $GF(2^5)$  and performing some multiplications, squaring, and additions in  $GF(2^5)$ . The inverse in  $GF(2^5)$  can be stored in a small table. We use a standard basis for the subfield  $GF(2^5)$  computations and the primitive polynomial defining  $GF(2^5)$  is  $x^5+x^2+1$ . Moreover, we select  $A$  equal to the unity (denoted 00001) to further simplify the operation. Follow gives a schematic representation of the required calculations.

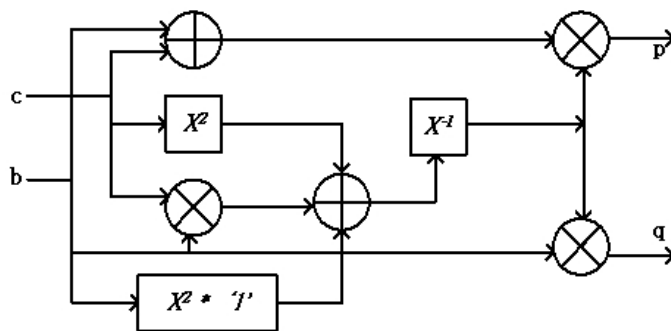


Figure A.5 : Composite field inverter over  $GF(2^{10})$

Finally, in order to transform  $GF(2^{10})$  into  $GF((2^5)^2)$ , the transform matrix and the monic primitive polynomial have to be determined. There exist effective algorithms for resolving both problems [27]. In our case, the monic primitive polynomial of  $GF((2^5)^2)$  is  $x^2+x+w^3$  ( $w^3$  denoted 01000, where  $w$  is the primitive root with respect to  $GF(2^5)$ ).

## ***Implementation Result***

After implementing by  $0.18\mu m$  1P6M standard cell slow library, the RS (528, 518) codec IP totally requires 2 finite-field multiplier, 1 composite-field inverter and  $17(=4t+1)$  registers, where  $t$  is the number of correctable errors. In contrast with other architectures, at least 42% circuit complexity can be reduced in our proposal.

Table A.2 compares the implementation for various key algorithms at  $0.18\mu m$  cell library process. In combination circuit, the hardware sharing circuit uses 2 finite field multipliers (FFM), 1 composite field inverter (CFI) and additional finite state machine control circuit. In synchronous circuit, the proposed architecture has only 17 register. Therefore, at least 42% circuit complexity can be reduced by our proposed hardware sharing architecture.

Table A.2 : The comparison table for (528,518) RS codec with different key-equation block

<b><i>Process (0.18um)</i></b>	<b>Components</b>	<b>Combinational gates count</b>	<b>Synchronous gates count</b>	<b>Total gates count</b>
<b>Original with BM</b>	4 FFM+1 CFI 33 REGs	6K	4.8K	10.8K <b>(1.42)</b>
<b>Original with Euclidean</b>	4 FFM+1 CFI 44 REGs	7.2K	5.6K	12.8 K <b>(1.68)</b>
<b>Our proposal</b>	2 FFM+1 CFI 17 REGs	4.8K	2.8K	7.6K <b>(1)</b>

## ***Conclusion***

In this chapter, an area-efficient methodology is presented to reduce RS codec area for some applications. The circuit complexity can be reduced through sharing registers and finite-field operation units. And the RS encoder can be also combined with the syndrome calculator. Furthermore, the composite-inverter over  $GF(2^{10})$  is implemented to replace the original look-up table. As a result, it can be achieved to reduce at least 42% circuit complex. If the number of correctable errors is large, the reducing area will raise linearly.



# BIBLIOGRAPHY

- [1] R. Blahut, Theory and Practice of Error control Codes, Boston: Addison-Wesley, 1983
- [2] W. Peterson “Encoding and error-correction procedures for the BCH codes,” IEEE Trans. on Inform. Theory, vol IT-6, pp459-470, Sep. 1960.
- [3] B. Wicker Error Control Systems for Digital Communication and storage, Georgia Institute of Technology
- [4] S. Lin and D. Costello, Error Control Coding: Fundamentals and Applications, Englewood Cliffs., NJ: Prentice-Hall, 1983.
- [5] E. Berlekamp, Algebraic Coding Theory, New York: McGraw-Hill, 1968.
- [6] J. Massey, “Shift-register synthesis and BCH decoding,” IEEE Trans. Inform. Theory, vol. IT-15, pp122-127, Jan. 1969
- [7] G. Forney, “On decoding BCH codes,” IEEE Trans. Inform. Theory, vol. IT-11, pp. 549–557, Oct. 1965
- [8] C.K. Koc and T Acar, “Fast Software Exponentiation in  $GF(2^k)$ ”, 13th IEEE Symposium on Computer Arithmetic, pp. 225-231, 1997.
- [9] R.J. McEliece, Finite Fields for Computer Scientists and Engineers, Kluwer Academic Publishers, Boston, MA, 1987.
- [10] T. Itoh and S. Tsujii, “A Fast Algorithm for Computing Multiplicative Inverses in  $GF(2^m)$  Using Normal Bases”, Journal of Information and Computation J., vol 78, pp.171-177, 1988.
- [11] L. Song, K.K. Parhi, I. Kuroda, and T. Nishitani, “Hardware/Software Codesign of Finite Field Datapath for Low Energy Reed-Solomon codecs”, IEEE Transactions on Very Large Scale Integration Systems, Vol. 8, No. 2, pp. 160-172, Apr. 2000

- [12] T.-K. T. J.-H Jeng, "On decoding of both errors and erasures of a Reed-Solomon code using an inverse-free Berlekamp-Massey algorithm," IEEE Trans. Comput., vol. 47, pp. 1488–1494, Oct. 1999.
- [13] Hsie-Chia Chang, and Shung, C.B, "New serial architecture for the Berlekamp-Massey algorithm," IEEE Trans. on Commun., Page(s):481 – 483, April, 1999.
- [14] H.C. Chang, C.B. Shung and C.Y. Lee, "A Reed–Solomon Product-Code (RS-PC) Decoder Chip for DVD Applications", IEEE Journal of Solid-State Circuits, Vol. 1, No. 2, pp. 229-236, Feb. 2001
- [15] I. Reed, M. Shih, and T.-K. Truong, "VLSI design of inverse-free Berlekamp-Massey algorithm," Proc. Inst. Elect. Eng. pt. E, vol. 138, pp. 295–298, Sept. 1991.
- [16] K.-Y. Liu, "Architecture for VLSI Design of Reed-Solomon decoders," IEEE Trans. Comput., vol. C-33, pp. 178–189, Feb. 1984.
- [17] Jyh-Hong Jeng and Trieu-Kien, Truong, "On Decoding of Both Errors and Erasures of a Reed-Solomon Code Using an Inversion-Free Berlekamp-Massey Algorithm," IEEE Trans. on Communication, vol.47, no. 10, October, 1999.
- [18] Kyutaeg Oh and Wonyong Sung, "An efficient Reed-Solomon decoder VLSI with erasure correction," IEEE Trans. on Communication, Page(s):193 – 201, Nov. 1997
- [19] Tong Zhang and K. K. Parhi "On the High-Speed VLSI Implementation of Errors-and-Erasures correcting RS decoders" GLSVLSI 02, April 18~19, 2002
- [20] Chien-Ching Lin, Fuh-Ke Chang, Hsie-Chia Chang, and Chen-Yi Lee, "An Universal VLSI Architecture for Bit-parallel computation in  $GF(2^m)$ ," IEEE Asia-Pacific Conference Circuits and Systems, 6-9 Dec, 2004.

- [21] Huai-Yi Hsu, and An-Yeu Wu, "VLSI design of a reconfigurable multi-mode Reed-Solomon codec for high-speed communication systems," IEEE Asia-Pacific Conference, Page(s):359 – 362, 6-8 Aug, 2002.
- [22] A.G.M. Strollo, N.petra, D.De Caro, and E. Napoli "An Area-Efficient High-Speed Reed-Solomon Decoder in 0.25um CMOS," IEEE 30 th Eur. Solid State Circuits Conf. (ESSCIRC), 21-23 Sept. 2004.
- [23] Jin-Chuan Huang, Chien-Ming Wu, and Ming-Der Shieh, Chien-Hsing Wu, "An area-efficient versatile Reed-Solomon decoder for ADSL," IEEE Int. Conf. Circuits and Systems(ISCAS), Page(s):517 – 520, 30 May-2 June, 1999
- [24] G. Fettweis, M. Hassner, "A Combined Reed-Solomon Encoder and Syndrome Generator with Small Hardware Complexity," IEEE ISCAS, vol. 4 , pp.1871~1874, May 1992.
- [25] H.C. Chang and C.B. Shung. "New Serial Architecture for the Berlekamp-Massey Algorithm," IEEE Trans. on Communications, vol. 47, no. 4, April 1999.
- [26] V. Rijmen, "Efficient Implementation of the Rijndael S-box," the Rijndael page, <http://www.esat.kuleuven.ac.be/~rijmen/rijndael/>
- [27] C. Paar. "Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields," VDI Verlag, Dusseldorf, 1994.

## 作者簡介

姓名：張富科

出生：桃園縣

學歷：武漢國小 復旦國中 中壢高中

88.9 ~ 92.6 國立成功大學電機工程學系

92.9 ~ 94.7 國立交通大學 電子研究所 (oasis lab)

## 得獎

92 學年度大專院校 FPGA 設計競賽 Xilinx 組優等



## 已發表論文

Chien-Ching Lin, Fuh-Ke Chang, Hsie-Chia Chang, and Chen-Yi Lee, "An Universal VLSI Architecture for Bit-parallel computation in  $GF(2^m)$ ," *IEEE Asia-Pacific Conference Circuits and Systems*, 6-9 Dec, 2004

Fuh-Ke Chang, Wei-Chun Hsu Chien-Ching Lin, and Hsie-Chia Chang, "Design and Implementation of an Reconfigurable Architecture for (528, 518) RS Codec IP," *IEEE NEWCAS 05*, 22-24 June, 2005

Fuh-Ke Chang, Chien-Ching Lin, Hsie-Chia Chang, and Chen-Yi Lee, "An Universal Architecture for Reed-Solomon Erasure Decoder," *IEEE ASSCC*, 1-3 Nov, 2005