

國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

應用於 IEEE 802.15.3c 規格之高面積效率萃式
(224, 216) RS 解碼器



**An Area-Efficient Chase-type (224,216) RS Decoder for
IEEE 802.15.3c Applications**

學生：黃裕淳

指導教授：張錫嘉教授

中華民國九十八年十月

應用於 IEEE 802.15.3c 規格之高面積效率萃式
(224,216) RS 解碼器

**An Area-Efficient Chase-type (224,216) RS Decoder
for IEEE 802.15.3c Applications**

研 究 生：黃裕淳

Student：Yu-Chun Huang

指導教授：張錫嘉教授

Advisor：Hsie-Chia Chang

國立交通大學
電子工程學系 電子研究所 碩士班
碩 士 論 文

A Thesis

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

Electronics Engineering
October 2009
Hsinchu, Taiwan, Republic of China

中華民國九十八年十月

應用於 IEEE 802.15.3c 規格之高面積效率萃式 (224, 216) RS 解碼器

學生：黃裕淳

指導教授：張錫嘉 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘 要

隨著資料傳輸的頻寬不斷增加，未壓縮高品質影像傳輸的通訊系統設計成為了一個有挑戰性的主題。根據 IEEE 802.15.3c 規格，在這樣的通訊系統必須使用 (224, 216) 的 RS 解碼器。訊號在高速傳輸之下更顯脆弱，即使目前有少許可增加錯誤更正能力的軟性解碼的方式例如 Guruswami-Sudan 演算法、Koetter-Vardy 演算法以及低複雜度萃式演算法被提出。這些方法仍然因為高硬體複雜度以及速度過慢等問題而無法實現。

在論文中，我們提出了一個高速且高面積效率的萃式 (224, 216) 的 RS 解碼器。在錯誤更正能力上，相對於傳統 RS 解碼器能在 $BER=10^{-5}$ 得到 0.5dB 的效能改善。根據在 90nm 製程下的實驗結果，所提出的解碼器包含 27.5k 的運算邏輯閘。此解碼器的工作頻率最高為 312.5MHz 並且能達到 2.41Gb/s 的傳輸速度。

An Area-Efficient Chase-type (224,216) RS Decoder for IEEE 802.15.3c Applications

Student : Yu-Chun Huang

Advisor : Dr. Hsie-Chia Chang

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

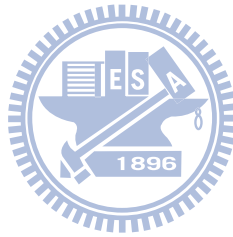
Abstract



As the bandwidth of data transmission advanced day by day, the communication system for uncompressed high-definition(HD) video streaming which requires high data rate is a challenging topic now. According to the IEEE 802.15.3c specification, a RS(224,216) decoder is specified in the uncompressed HD video streaming system. Since the signals under high-speed transmission will be more fragile, several techniques such as Guruswami-Sudan algorithm, Koetter-Vardy algorithm and Low-Complexity Chase algorithm are carried out to improve error-correcting ability of Reed-Solomon code. However, most of those method are still not practical because of huge hardware cost and large critical path due to high complexity.

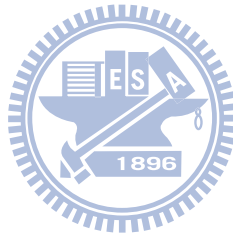
In this thesis, we proposed an high-speed and area-efficient Chase-type (224,216) Reed-Solomon decoder that has a performance gain of 0.5dB at $BER = 10^{-5}$ in contrast with conventional hard-decision RS decoder. Implemented in UMC CMOS

90nm 1P9M process, the proposed decoder can reach 2.41Gb/s at 312.5MHz operation frequency with 27.5K gate count.



誌 謝

兩年多的碩士班生活一晃眼就過了，這兩年經歷也學到了許多研究學習方法以及待人處世的道理。首先最該感謝的當然是指導教授張錫嘉老師，除了論文的耐心指導外，老師總是能在研究遇到瓶頸或問題時提出精闢的意見。此外，老師對於日常生活上的問題也給予了不少建議。接著要感謝對於研究上給予我最多幫助的林義閔學長，在這兩年間，無論是研究方向或是 IC 設計方面的問題，都仰賴學長的支持和幫助才能順利解決。當然，還必須要感謝 OCEAN group 和 OASIS Lab 的所有成員，以及其他在研究期間不斷關心幫助我的家人和朋友們。最後，僅向各位致上最高的謝意。



An Area-Efficient Chase-type (224,216) RS Decoder for IEEE 802.15.3c Applications

Student: Yu-Chun Huang

Advisor: Dr. Hsie-Chia Chang



Department of Electronics Engineering
National Chiao Tung University

Abstract

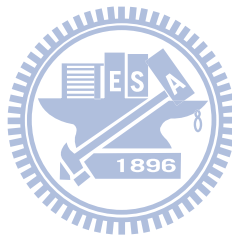
As the bandwidth of data transmission advanced day by day, the communication system for uncompressed high-definition(HD) video streaming which requires high data rate is a challenging topic now. According to the IEEE 802.15.3c specification, a RS(224,216) decoder is specified in the uncompressed HD video streaming system. Since the signals under high-speed transmission will be more fragile, several techniques such as Guruswami-Sudan algorithm, Koetter-Vardy algorithm and Low-Complexity Chase algorithm are carried out to improve error-correcting ability of Reed-Solomon code. However, most of those method are still not practical because of huge hardware cost and large critical path due to high complexity.

In this thesis, we proposed an high-speed and area-efficient Chase-type (224,216) Reed-Solomon decoder that has a performance gain of 0.5dB at $BER = 10^{-5}$ compared with conventional hard-decision RS decoder. Implemented in UMC CMOS 90nm 1P9M process, the proposed decoder can reach 2.41Gb/s at 312.5MHz operation frequency with 27.5K gate count.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Thesis organization | 2 |
| 2 | Hard-decision Reed-Solomon codes | 3 |
| 2.1 | Encoder of hard-decision Reed-Solomon codes | 3 |
| 2.2 | Decoder of hard-decision RS codes | 4 |
| 2.2.1 | Syndrome calculator | 4 |
| 2.2.2 | BM algorithm | 5 |
| 2.2.3 | Chien search | 7 |
| 2.2.4 | Error value evaluator | 7 |
| 3 | Soft-Decision Reed-Solomon Codes | 8 |
| 3.1 | Guruswami-Sudan algorithm | 9 |
| 3.1.1 | Interpolation | 10 |
| 3.1.2 | Factorization | 11 |
| 3.2 | Koetter-Vardy algorithm | 12 |
| 3.3 | Low-complexity Chase algorithm | 14 |
| 4 | Proposed Chase-type RS decoder | 15 |
| 4.1 | Design Parameters Analysis | 15 |
| 4.2 | Chase-Type Decoder Architecture | 17 |
| 4.2.1 | Syndrome calculator and Chase calculator | 20 |
| 4.2.2 | BM algorithm | 21 |
| 4.2.3 | Chien search and error value evaluator | 22 |

| | | |
|----------|---|-----------|
| 4.2.4 | Decision making unit | 26 |
| 5 | Implementation result | 28 |
| 5.1 | RTL design flow | 28 |
| 5.2 | Implementation result | 29 |
| 5.3 | Comparison of different RS decoders | 30 |
| 6 | Conclusion and Future Work | 34 |
| 6.1 | Conclusion | 34 |
| 6.2 | Future work | 34 |



List of Figures

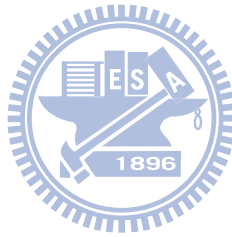
| | | |
|------|--|----|
| 1.1 | Block diagram of digital communication system. | 2 |
| 2.1 | The RS systematic encoder | 4 |
| 2.2 | Flow chart of the RS decoder | 4 |
| 2.3 | Berleykamp-Massey algorithm | 6 |
| 3.1 | Flow chart of Chase algorithm | 9 |
| 3.2 | Comparison for BD and list decoding | 10 |
| 3.3 | GS encoder and decoder with re-encoding technique | 11 |
| 3.4 | Brief interpolation process | 12 |
| 3.5 | The Koetter-Vardy algorithm | 13 |
| 3.6 | Algorithm for calculating multiplicity matrix | 13 |
| 4.1 | Performance of Chase-type decoding with different η | 16 |
| 4.2 | Performance of Chase-type decoding with different precision of soft-information | 17 |
| 4.3 | Performance of Chase-type decoding using Hamming distance or Euclidean distance | 18 |
| 4.4 | Block Diagram of a soft-decision Chase-type RS decoder design | 19 |
| 4.5 | Timing schedule of a soft-decision Chase-type RS decoder design | 19 |
| 4.6 | Syndrome Calculator design | 20 |
| 4.7 | Gray code example used in the design | 21 |
| 4.8 | Chase Calculator design | 21 |
| 4.9 | Berleykamp-Massey architecture design | 23 |
| 4.10 | Chien search architecture design | 24 |
| 4.11 | Serial shifter for error location record | 25 |
| 4.12 | Bjorck-Pereyra algorithm | 25 |

| | | |
|------|---|----|
| 4.13 | Bjorck-Pereyra architecture design | 26 |
| 4.14 | Flow chart for decison making block | 27 |
| 5.1 | The entire design flow | 29 |
| 5.2 | The layout of proposed RS decoder | 31 |
| 5.3 | Performance curve for comparison | 32 |



List of Tables

| | | |
|-----|--|----|
| 4.1 | Comparison over BM architecture | 22 |
| 5.1 | Summary of proposed design | 30 |
| 5.2 | Comparison of hardware requirements (a) LCC (b) UiBM | 33 |



Chapter 1

Introduction

1.1 Motivation

The fundamental block diagram of traditional digital communication system is showed in Fig. 1.1. The system transmits data from an information source to an information destination through an unknown channel. Generally, the communication system is simplified to three component parts which consists of transmitter, receiver, and channel. The transmitter includes source encoder, channel encoder, and modulator, which is used to transmit the information more effectively and more reliably. Furthermore, the receiver will reverse the converted-signal received by demodulator, channel decoder, and source decoder. Since the channel involved such as noise, interference and distortion may cause error in the received signal, the channel encoder is added to the system to reduce the transmission errors by adding certain redundancy message to the source codeword. These redundant message can be used for algebraic error correcting. Thus, the channel coding eliminates the effects of noise disturbances comparing with an simple uncoded communication system.

Nowaday, wireless has become a prominent technology of consumer communication and networking. For example, Bluetooth and WiFi has been widely used in mobile and multimedia network. Among those multimedia applications like PCs and digital TVs, high-definition(HD) video streaming [1] is always in the spotlight. A wireless video transmission system for supporting uncompressed HD video requires a high-speed system. According to IEEE 802.15.3c [2] standard, the millimeter-wave WPAN transmission over 2Gb/s is allowed.

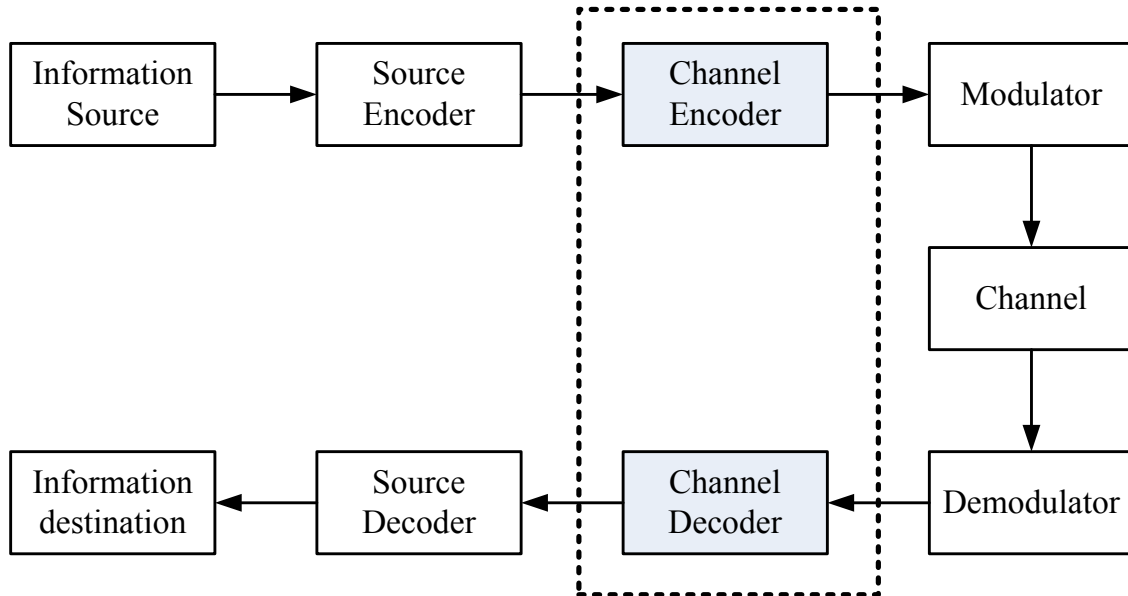


Figure 1.1: Block diagram of digital communication system.

To implement a uncompressed HD video transmission system mentioned above, we integrated a high-speed Reed-Solomon channel decoder. Furthermore, we are interested in improving the error-correcting ability of the decoder. In this thesis, also we tried to apply KV algorithm [3] and some other techniques to achieve better error protection. Finally, we proposed a high-speed and low complexity RS(224,216) decoder.

1.2 Thesis organization

This thesis consists of 5 chapters. In chapter 2, the concepts of several improving decoding algorithms of Reed-Solomon codes will be introduced. In chapter 3, the decoding procedure and hardware architecture of high speed design will be introduced. In chapter 4, the hardware implementation result will be shown. Finally, the conclusions and future works are given in chapter 5.

Chapter 2

Hard-decision Reed-Solomon codes

The Reed-Solomon (RS) code, a variant of BCH code, was invented in 1960 by Irving S. Reed and Gustave Solomon [4]. RS codes are efficient of treating burst error, and are used in hard disk driver, DVDs, telecommunication, and digital protocols. RS encoding and decoding systems used nowadays are mostly the hard-decision ones. A (n,k,t) RS code over $GF(2^m)$ represents that the codeword of the system has a block length n and $n - k$ parity symbols, which is composed of m bits. And the t means the maximum number of correctable error of the system.

In this chapter, the encoding and decoding procedure of a conventional RS system will be introduced. Each part of decoder, such as BM algorithm for solving key equation and Forney's algorithm for evaluating error value, will be described in detail.

2.1 Encoder of hard-decision Reed-Solomon codes

Reed-Solomon (RS) encoders typically use a recursive systematic convolutional (RSC) encoders. The block diagram of a RS encoder is illustrated in Fig. 2.1. In the Fig. 2.1, message is encoded based on the generator polynomial $g(x)$, and the information bits are encoded to the systematic part $m(x)$ and the parity part $r(x)$.

$$c(x) = m(x) \cdot x^{2t} + r(x) \quad (2.1)$$

$$r(x) = m(x) \cdot x^{n-k} \bmod g(x) \quad (2.2)$$

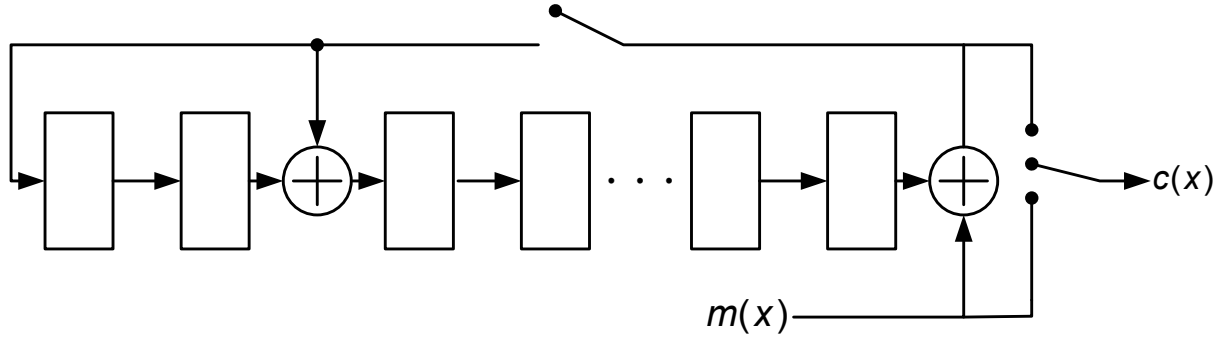


Figure 2.1: The RS systematic encoder

2.2 Decoder of hard-decision RS codes

A common hard-decision RS decoder flow chart is shown in Fig. 2.2. Where $r(x)$ is the received systematic information, $S(x)$ is the $2t$ syndromes calculated from the received vector $r(x)$. The key equation, defined by $\Omega(x) = S(x)\sigma(x) \bmod x^{2t}$, to be found by Berlekamp-Massey or Euclidean algorithm. The RS decoder is then divided into two parts, the error location calculator and error value evaluator. The error location solver finds out the position of symbols that are tainted by noise from channel, then the error value evaluator solves out the error value for each error location.

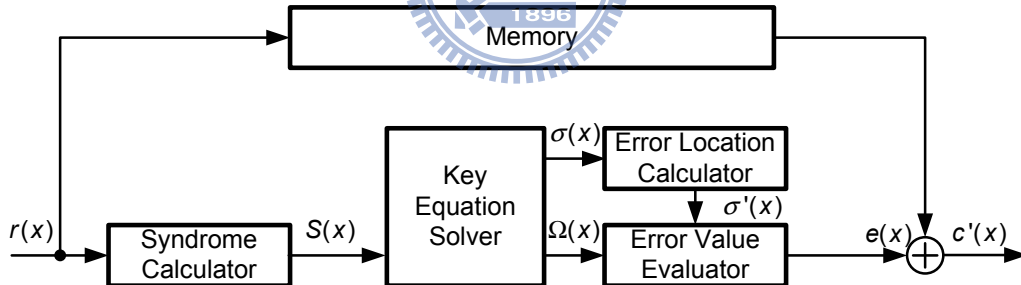


Figure 2.2: Flow chart of the RS decoder

2.2.1 Syndrome calculator

In RS decoders, the message polynomial $m(x)$ is encoded by dividing the generator polynomial $g(x)$. The generator polynomial has the property

$$g(\alpha) = g(\alpha^2) = \dots = g(\alpha^{2^t}) = 0 \quad (2.3)$$

Thus, the polynomial of encoded codeword $c(x)$ has the same property

$$c(\alpha) = c(\alpha^2) = \dots = c(\alpha^{2^t}) \quad (2.4)$$

After transmitted through the channel, the recieved polynomial $r(x) = c(x) + e(x)$ generates a series of information by substituting $\alpha \sim \alpha^{2^t}$

$$S_i = r(\alpha^i) = \sum_{j=1}^v e_j X_j^i \quad i = 1, 2, \dots, 2t \quad (2.5)$$

where

$$X_j = \alpha^{i_j} \quad (2.6)$$

The value S_1, S_2, \dots, S_{2t} are called the syndrome of the recieved sequence. By solving the syndrome equations, we can get the error value e_j and error location X_j .

2.2.2 BM algorithm

In RS decoding algorithm, Berleykamp-Massey (BM) algorithm is commonly applied to solve key equation from error locator polynomial. Compared with Euclidean algorithm, it's easier for practical hardware implementation since it has a fixed computation time. BM algorithm was developed in 1969 by Elwyn R. Berlekamp [5] and James Massey [6]. BM algorithm works recursively modifying the minimal polynomial to fit the sequences of syndrome

$$\begin{aligned} S_1 &= e_1 X_1 + e_2 X_2 + \dots + e_v X_v \\ S_2 &= e_1 X_1^2 + e_2 X_2^2 + \dots + e_v X_v^2 \\ &\vdots \\ S_{2t} &= e_1 X_1^{2t} + e_2 X_2^{2t} + \dots + e_v X_v^{2t} \end{aligned}$$

Define the key equation $\sigma(x)$ as follow

$$\sigma(x) = (1 - X_1 x)(1 - X_2 x) \dots (1 - X_v x) = \sigma_0 + \sigma_1 x + \dots + \sigma_v x^v \quad (2.7)$$

$$\begin{aligned}
\sigma_0 &= 1 \\
\sigma_1 &= X_1 + X_2 + \dots + X_v \\
\sigma_2 &= X_1X_2 + X_2X_3 + \dots + X_{v-1}X_v \\
&\vdots \\
\sigma_v &= X_1X_2 \cdots X_v
\end{aligned}$$

According to the Newton's identities, rewrite the relation in matrix form

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_v \\ S_2 & S_3 & \cdots & S_{v+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_v & S_{v+1} & \cdots & S_{2v-1} \end{bmatrix} \begin{bmatrix} \sigma_v \\ \sigma_{v-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} S_{v+1} \\ S_{v+2} \\ \vdots \\ S_{2v} \end{bmatrix}$$

The psuedo code for the decoding process can be express in Fig 2.3. After the $2t$ times of update, the minimal polynomial result will be the error locator polynomial that fulfill all the syndrome equations.

Initial: $L_{-1} = 0, \delta = 1, \tau^{(-1)}(x) = 1, \sigma^{(-1)}(x) = 1, \Delta_0 = S_1$
for $i = 0$ **to** $2t - 1$
 $\sigma^{(i)}(x) = \sigma^{(i-1)}(x) + \frac{\Delta_i}{\delta} \cdot x \tau^{(i-1)}(x)$
 $\Delta_{i+1} = S_{i+2} + S_{i+1} \sigma_1^{(i)} + \dots + S_1 \sigma_{i+1}^{(i)}$
if $\Delta_i = 0$ **or** $2L_{i-1} \geq i + 1$ **{**
 $L_i = L_{i-1}$
 $\tau^{(i)}(x) = x \tau^{(i-1)}(x)$
}else{
 $L_i = i + 1 - L_{i-1}$
 $\tau^{(i)}(x) = \sigma^{(i-1)}(x)$
 $\delta = \Delta_i$
}

Figure 2.3: Berleykamp-Massey algorithm

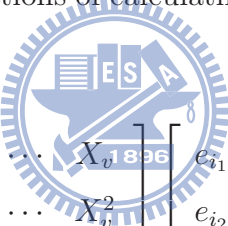
2.2.3 Chien search

Once the error locator polynomial $\sigma(x)$ of the decoding sequence is known, the following task is to find out error locations and error values. Chien search is an efficient and simple way that searching all the roots of error locator polynomial $\sigma(x)$. By evaluating each nonzero element in $GF(2^m)$, all X_i^{-1} of $\sigma(x)$ will be found.

If the roots are all distinct and lie in the appropriate location, that is, the error locator polynomial of degree v have exactly v different roots in which the operations take in. Then the results can be applied to error value evaluator and the error values of each location will be determined. Otherwise, if the error locator have repeated roots or illegal root, then the received word is not within distance t of any codeword. In this case, the error pattern of the received sequence is uncorrectable.

2.2.4 Error value evaluator

For RS codes, solving error values of each error is the last step required for error correction. Observing that the equations of calculating syndrome formed a Vandermonde matrix



$$\begin{bmatrix} X_1 & X_2 & \cdots & X_v \\ X_1^2 & X_2^2 & \cdots & X_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{2t} & X_2^{2t} & \cdots & X_v^{2t} \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_v} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{2t} \end{bmatrix}$$

Forney's algorithm is a fast way of solving Vandermonde system, the error values of each error location can be easily formulated as

$$e_{ik} = \frac{\Omega(X_k^{-1})}{\sigma'(X_k^{-1})}$$

where $\Omega(x)$ is the key equation defined as

$$\Omega(x) = S(x)\sigma(x) \bmod x^{2t}$$

Chapter 3

Soft-Decision Reed-Solomon Codes

Conventional hard-decision RS codes are employed in many practical systems now. However, the RS decoder using BM algorithm can only correct errors up to half the minimum distance. In order to improve the error correction limit of RS codes, in earlier years, reliability based algorithms such as Chase algorithm were published.

Chase algorithm [7] is a simple decoding strategy which generated and decoded several possible sequences around the original sequence. It functioned by applying bit-flipping to the original received sequence c_0 , and a set of candidate sequences $c_0, c_1, \dots, c_{\eta-1}$ are generated in this way. To do the bit-flipping, Chase algorithm chooses the least reliable independent positions [8](LRIPs), which are determined by applying the soft information from channel. Fig. 3.1 shows the decoding flow of the Chase algorithm. Note that η is the number of LRIPs to be bit-flipped.

However, the improvement of Chase algorithm is limited and remains unsatisfying. Thus, Guruswami-Sudan (GS) algorithms [9] were developed, which can extend the error correction ability to exceed the distance of t . Based on this algorithm, other extensions which make use of the soft information from channel were carried out, trying to reduce the error rate by "suspect" those unreliable data than those reliable ones. In this chapter, both traditional hard-decision RS decoding and soft-decision RS decoding methods will be briefly introduced.

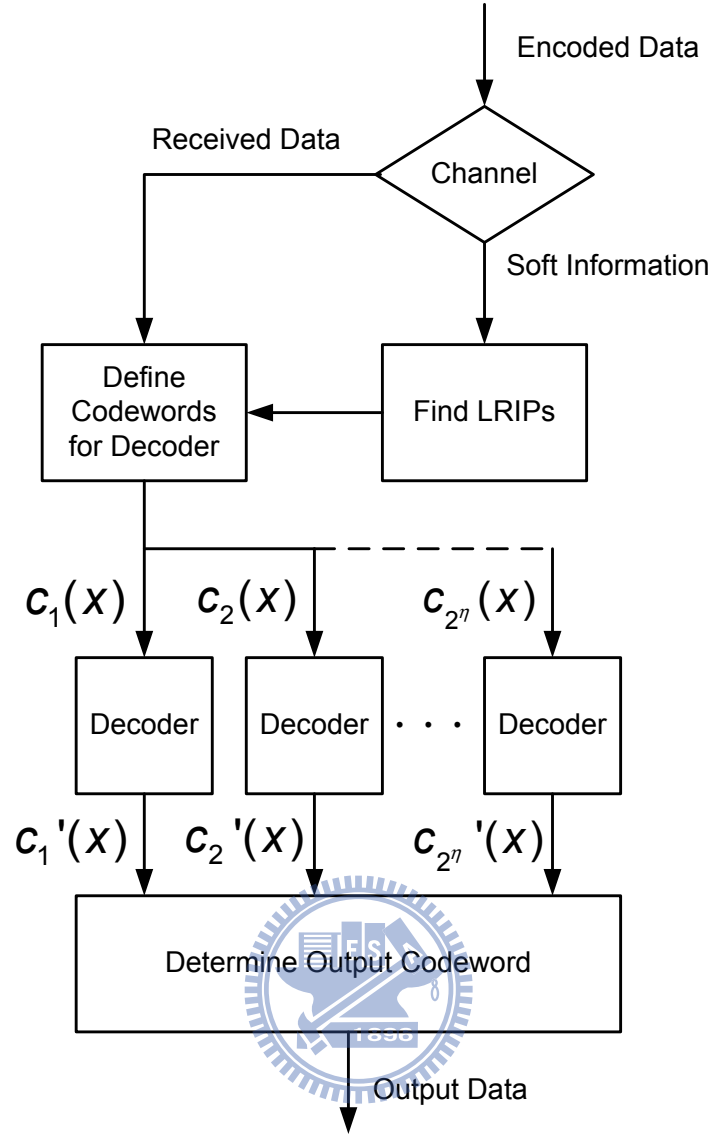


Figure 3.1: Flow chart of Chase algorithm

3.1 Guruswami-Sudan algorithm

In 1999, Madhu Sudan and Venkatesan Guruswami [9] introduced an alternative list decoding algorithm for RS codes, which can stretch the error correction ability and find out all the probable codewords within its extended decoding sphere. Compared with the original bounded distance (BD) decoding, the list decoding is able to solve some situations that would be uncorrectable for BD decoding. Guruswami-Sudan (GS) algorithm generates several possible decoded codewords, although in most situations, the list size of codewords is small. This algorithm provides a means for algebraic way of soft decision

decoding of RS codes.

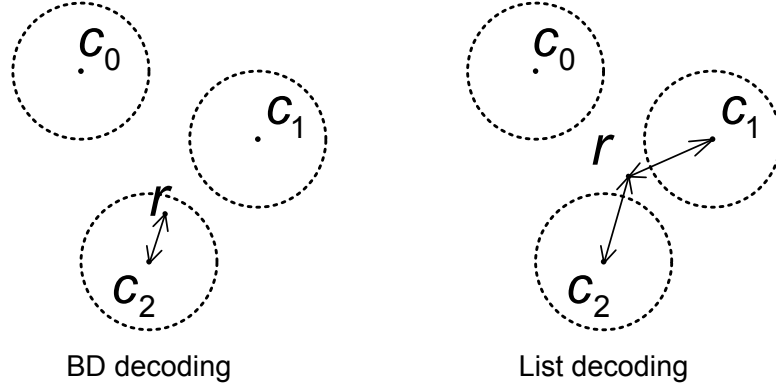


Figure 3.2: Comparison for BD and list decoding

GS encoder is different from the typical RSC encoder, the original message polynomial $m(x)$ is encoded by substituting with each element value of $GF(2^m)$ by an evaluate map encoder, which is a non-systematic encoder. However, it can be transformed into a systematic encoder by adding a transforming matrix. Since it has been proved that the codeword set is from the same span of space, the transformed codeword can be decoded with the same decoder.

Block diagram for a GS encoding and decoding system is showed in Fig 3.3, the recieved message $y(x)$ is re-encoded [10] to reduce the computation time of GS decoder. As in the Fig. 3.3, by re-encode the recieved pattern R and added to itself, the systematic part of the new pattern R' will be all zero. Thus, the number of vectors for GS decoder to "interpolate" is highly reduced at a rate of $\frac{K}{N}$.

The decoder processing units can be mainly separated into two parts: interpolation and factorization. We will briefly introduce these two parts in the following paragraphs.

3.1.1 Interpolation

In the GS encoding, a set of data points (x_i, c_i) , $i = 1, \dots, n$ are generated, which means the original codeword polynomial "interpolates" each point of the set. However, since some points are suffered from noise, we seek polynomials that can match at least $n - e$ data points. By finding those suitable polynomials, the errors are corrected as we choose one of the polynomial to be the result.

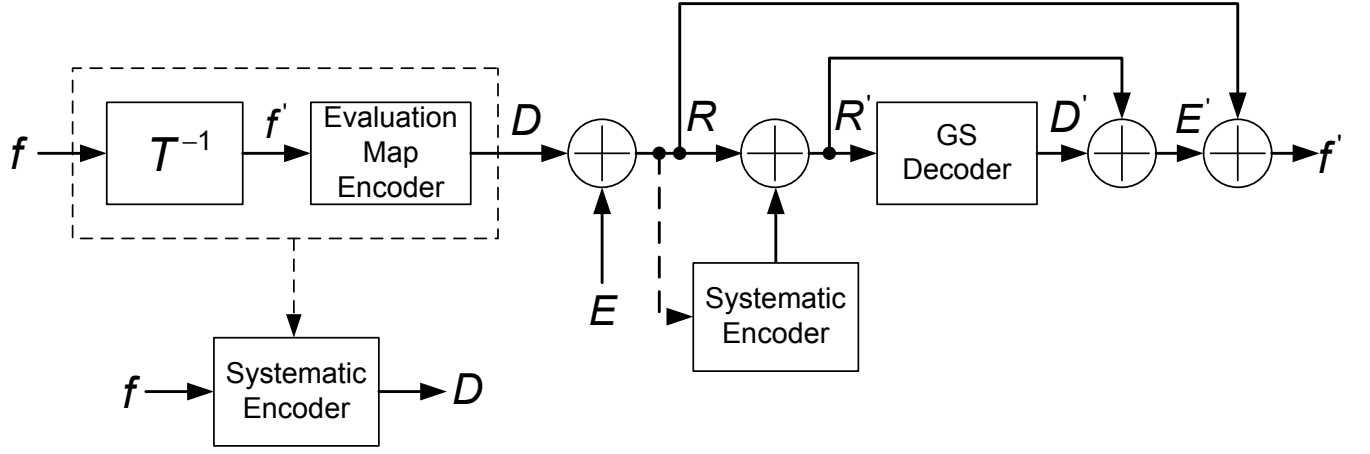


Figure 3.3: GS encoder and decoder with re-encoding technique

The decoder constructs several non-zero bivariate candidate polynomials, which is built up by interpolating each point by constituting recieved data points (x_i, y_i) , $i = 1, \dots, n$ and update the candidate polynomials by times ,the constructed polynomial is of the form

$$Q(x, y) = \sum_{j=0}^C a_j \phi_j(x, y) \quad (3.1)$$

$$C = n \cdot \frac{(m+1)!}{2!(m-1)!} \quad (3.2)$$

where $\phi_j(x, y)$ is of the form $x^p y^q$.

In addition to simple interpolating, an interpolation parameter, multiplicity m , is introduced to define the order of the interpolation at each point. Larger multiplicity m_i increases the error correction ability of GS decoder. However, the computation complexity of GS decoder would be highly raised ,which has a time complexity of $O(n^2 m^4)$. The GS algorithm also has a maximum error correction limit. For a (N, K) RS code applying GS algorithm, the maximum decoding sphere limit is $t_\infty = N - 1 - \left\lfloor \sqrt{N(K-1)} \right\rfloor$.

Algorithms such as Feng-Tzeng [11] algorithm and Koetter algorithm [12] are carried out for interpolation, and Koetter algorithm is commonly used.

3.1.2 Factorization

According the interpolation part, $Q(x, y)$ generated this way would contain all factors polynomial of the form $y - p(x)$, where $p(x)$ is a polynomial of degree $K - 1$ or less, and


```

Initial:  $g_0 = y^0, g_1 = y^1, \dots, g_\ell = y^\ell$ 
for  $i = 1$  to  $n$ 
   $C = \frac{(m_i+1)m_i}{2}$ 
  for  $(r, s) = 0$  to  $C$ 
    for  $j = 0$  to  $\ell$ 
       $\Delta_j = D_{r,s}g_j(x_i, y_i)$  (compute discrepancy of each points)
       $J = \{j : \Delta_j \neq 0\}$  (set of non - zero discrepancy)
      for all  $j \in J$ 
        let  $f = g_j^*, \Delta = \Delta_j^*$  ( $g_j^*$  is the smallest degree in set J)
         $g_j^* = (x - x_i) \cdot f$ 
         $g_j = \Delta \cdot g_j - \Delta_j \cdot f$ 
      end
    end
  end
 $Q(x, y) = \text{minimum degree } g_j$ 

```

Figure 3.4: Brief interpolation process

is one of the decoded result of the candidates.

$$L = \{p_1(x), p_2(x), \dots, p_\ell(x)\} \quad (3.3)$$

Those candidates in the list L match with (x_i, y_i) in at least t_m places. Factorization usually performed by applying Roth and Ruckenstein algorithm. After the decoded list is generated, normally the result with the minimum Euclidean distance is chosen.

3.2 Koetter-Vardy algorithm

GS algorithm provides a new point of view over RS codes by breaking the error correction limit with a extendable correcting distance. To make use of this advantage and further improve the performance of RS systems, Koetter-Vardy (KV) algorithm adds a multiplicity calculating process right ahead the GS algorithm, which translates soft-information from channel. By applying those information proportional to the reliability of each individual point, computation time for interpolation can be reduced on some level,

and the error correction performance is improved.

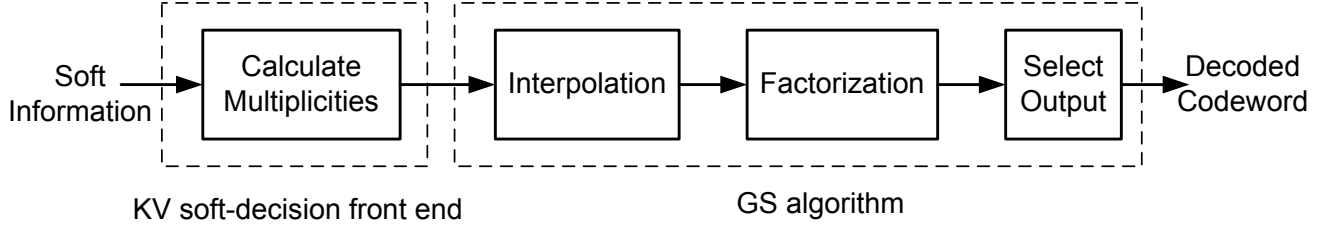


Figure 3.5: The Koetter-Vardy algorithm

Consider a codeword of symbols from $GF(q) = \{\alpha_0, \alpha_1, \dots, \alpha_{q-1}\}$ is transmitted. The reliability for a recieved value β_j given the symbol α_j was sent, is called the a posteriori probability (APP):

$$\pi_{i,j} = \Pr(\alpha_i \text{ send} | \beta_j \text{ recieved}) \quad (3.4)$$

The soft information can be calculated from the received channel information if the noise model is acknowledged. Then, a $(q \times n)$ reliability matrix Π can be constructed. Each entry $\pi_{i,j}$ represents the reliability of a point (x_j, y_i) . By translating the reliability matrix Π as the algorithm shown below, the multiplicity matrix M is calculated. In the multiplicity matrix M , each element corresponds to the multiplicity m to be use while calculating the information point in the interpolation part of decoder.

Choose the limit for $s = \sum_{i=0}^{q-1} \sum_{j=0}^{n-1} m_{i,j}$
 While $s > 0$
 Find largest entry $\pi_{i,j}$ in M
 $\pi_{i,j} = \frac{\pi_{i,j}}{m_{i,j}+2}$
 $m_{i,j} = m_{i,j} + 1$
 $s = s - 1$

Figure 3.6: Algorithm for calculating multiplicity matrix

3.3 Low-complexity Chase algorithm

KV algorithm calculates and processes overall soft information from channel, which enables the back-end GS decoder to achieve a better result with less complexity. However, the calculation required for interpolation during high multiplicity m is still too high for practical implementation.

Low-complexity Chase (LCC) algorithm [13] is a Chase-type decoding which cooperates with GS decoder at low multiplicity m . By determine the hard-decision (x_i, y_i^{HD}) and secondary hard-decision (x_i, y_i^{2HD}) symbols of each data, the probability γ_i is then given. A high probability that the hard-decision is wrong and the secondary hard-decision is correct if γ_i of the symbol is close to 1.

$$\gamma_i = \frac{p(c_i = y_i^{2HD} | \gamma_i)}{p(c_i = y_i^{HD} | \gamma_i)} \leq 1$$

Finding out the η symbols which have the largest γ_i , we interpolates 2^η times for each transmission. However, interpolation results of the $n - \eta$ data is the same, by taking the advantage of the interpolation property that each point is interpolated individually, only the η data has to be interpolated repeatedly. Some other improvements about LCC like [14] are also carried out, which further reduces the memory use for the multi-sequence generated by Chase algorithm.

Chapter 4

Proposed Chase-type RS decoder

In this chapter, the architecture of an area-efficient Chase-type RS decoder design will be introduced. To implement an area-efficient RS decoder, algebraic RS algorithm such as KV algorithm, is not seemed to be practical for use. Therefore, to improve the error correction ability, we change our focus on Chase algorithm.

4.1 Design Parameters Analysis

Chase algorithm involves a parameter η , which represents the number of bits to be changed. It also affects the numbers of candidate sequences to be decoded, so the value of η should be chosen carefully since the number of candidate sequences 2^η doubled once the η increased by 1. As shown in Fig. 4.1, Chase algorithm has a performance gain of 0.6dB while bit error rate (BER) is at 10^{-5} . According to the result, the performance gain at 10^{-5} is 0.6dB while $\eta = 5$ and is 0.5dB while $\eta = 3$, then we choose $\eta = 3$ for our design to make the computation time fixed with little performance loss.

To implement a soft-decision decoder, the number of bits for the soft-information input is an important issue since the precision of the information may affect the result of comparison. In Fig. 4.2, the performance will have a loss of 0.1dB at 10^{-5} while using 5 decimal bits and no performance loss while 6 or more decimal bits are used. Thus, we choose the number of floating point to be 7, which composed of 1 signed bit and 6 decimal bits.

The last problem for the implementation of proposed decoder is on determining output

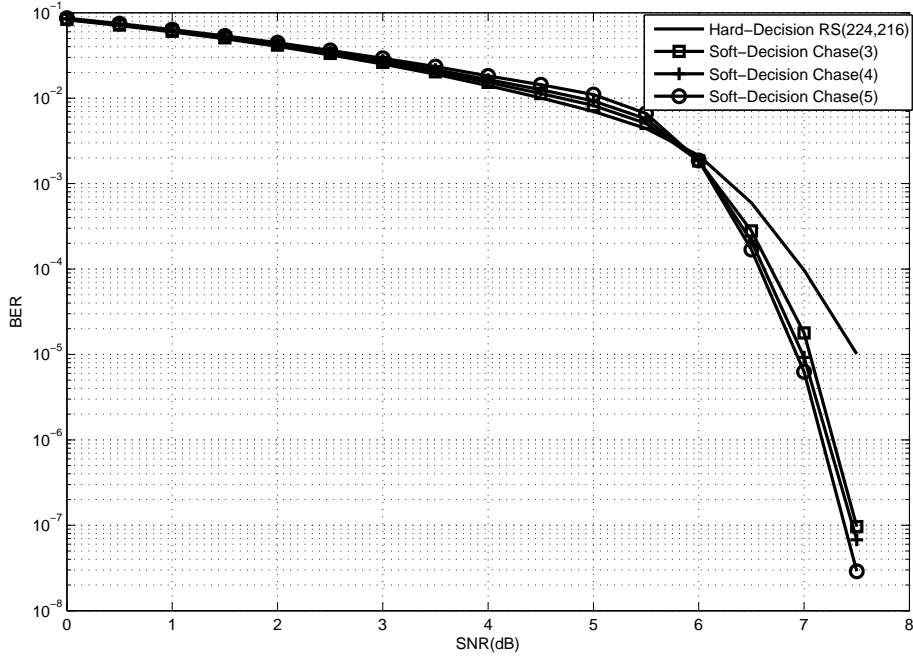


Figure 4.1: Performance of Chase-type decoding with different η

data from each decoded candidate codewords. Normally, the Euclidean distance is used.

$$R = \sum_{j=1}^n |c_j - r_j|^2$$

where c_j is the decoded symbol and r_j is the recieved value from channel. However, in Fig. 4.3, using Hamming distance as the determination method can get the same performance as Euclidean distance.

$$R = \sum_{j=1}^n |c_j - x_j|^2$$

If the BPSK modulation is used, the x_j is defined as

$$\begin{cases} x_j = +1 & \text{if } r_j < 0 \\ x_j = -1 & \text{if } r_j \geq 0 \end{cases}$$

Thus, we choose Hamming distance determination since it's simpler in calculating, which does not has a decimal part. Since the Hamming distance calculation only needs

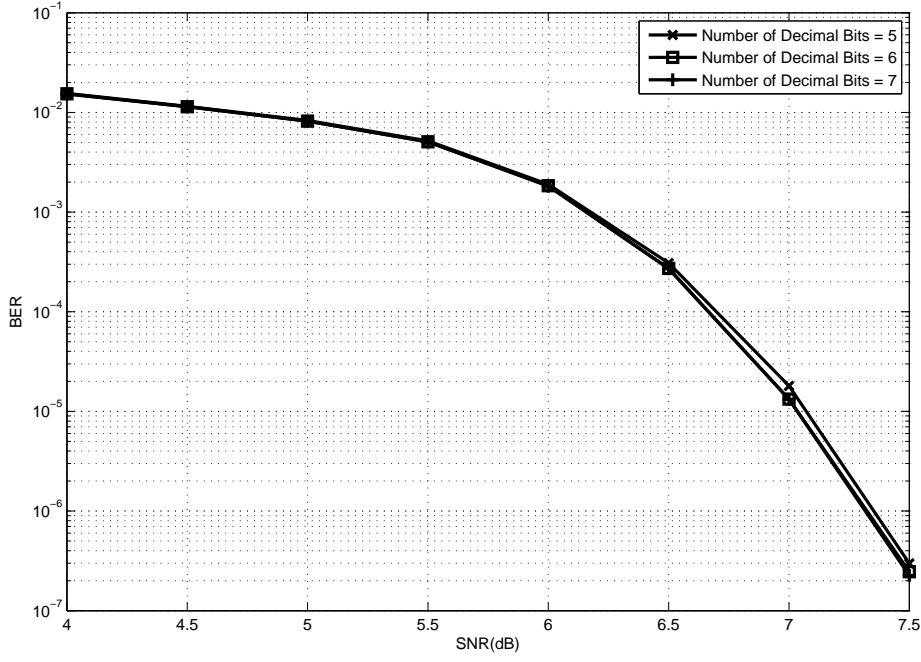


Figure 4.2: Performance of Chase-type decoding with different precision of soft-information

simple integer addition, this result is meaningful to our design which reduce the hardware cost.



4.2 Chase-Type Decoder Architecture

For a Chase-type RS decoder design, the main challenge is that the decoder has to deal with multi-sequence and determine which result to be output. In our case, there are 8 different codewords to process. Thus, a well-designed pipeline schedule is needed. According to section 2.1.2, a conventional RS decoder mainly composed of those operating units: syndrome calculator, key equation solver, Chien search and error value evaluator. In addition, a decision making unit which count for Hamming weight of each decoded codeword should be added to the backend of the decoder.

Fig. 4.4 above shows a Chase-type decoder design. Compared with Fig. 2.2, a Chase calculator and a decision making unit is added into the system. The Chase calculator recieves the transmitted data $r(x)$ and the soft information of each bits $i_1 \sim i_m$, then we

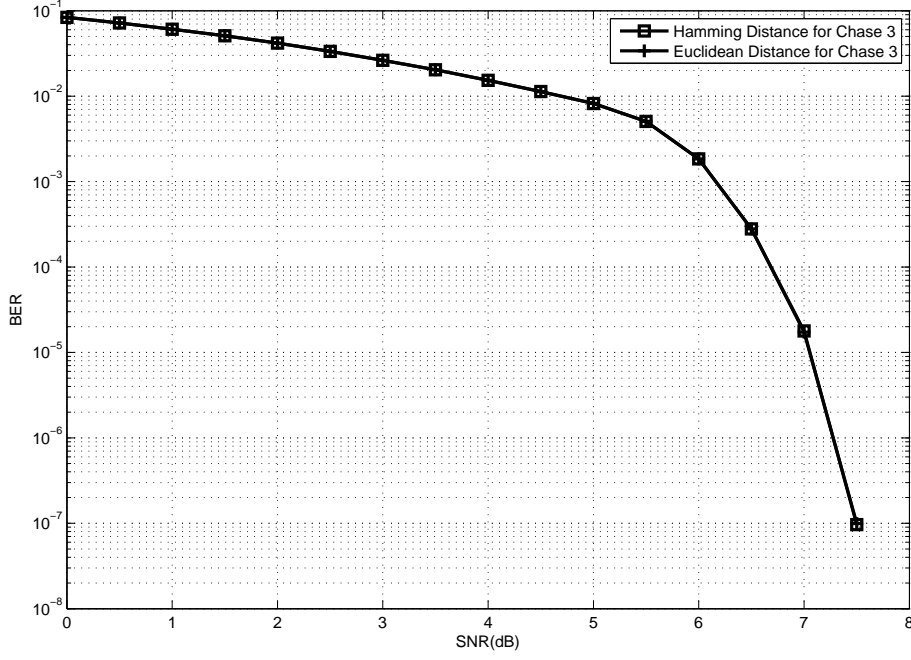


Figure 4.3: Performance of Chase-type decoding using Hamming distance or Euclidean distance

get η least reliable bits. By bit-flipping those bits and re-calculate the value of syndrome $e'(x)$ to be added to the original syndrome $S(x)$, the new syndromes $S'(x)$ are send to original RS decoder. The modified LRIPs can be considered as an error pattern we added to the syndrome.

$$e'(x) = e'_1 x^{j'_1} + e'_2 x^{j'_2} + \dots + e'_\eta x^{j'_\eta} \quad (4.1)$$

$$S'(x) = S(x) + e'(x) \quad (4.2)$$

As the same decoding procedure as typical RS decoder, we use BM algorithm to solve key equation and Chien search to find out error location. However, to evaluate the error value, we use Bjorck-Pereyra (BP) algorithm as an alternative algorithm to solve the Vandermonde matrix. Once the decoding process ends, the corrected codeword is sent to the decision making unit, storing the codeword closest to the recieved data and outputting the result.

Fig. 4.5 is the timing schedule of our design. As in Fig. 4.5, syndrome $S(x)$ is reck-

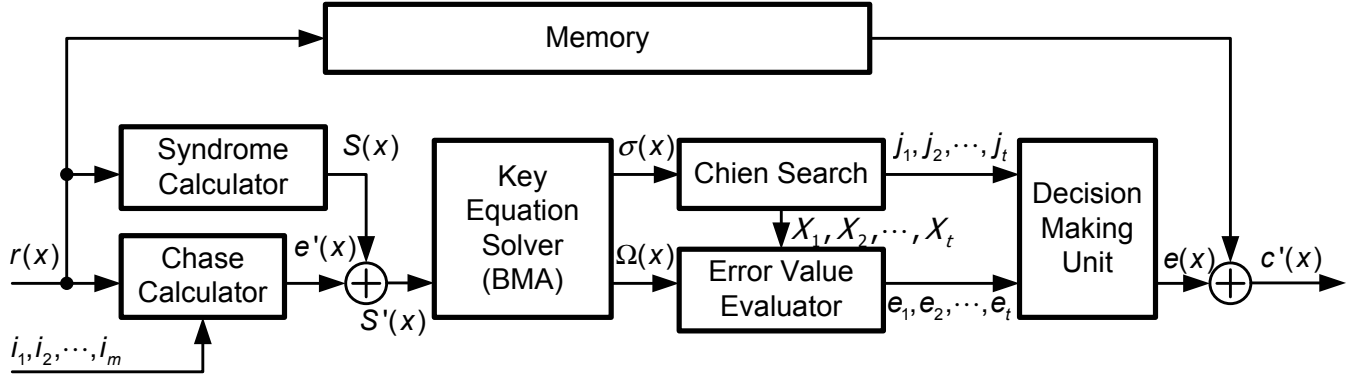


Figure 4.4: Block Diagram of a soft-decision Chase-type RS decoder design

oned at data input and first syndrome pattern is available while the first input pattern ends. Then, BM algorithm as the key equation solver start working, note that the Chase Calculator begins to count for the syndrome pattern to be modified $e'(x)$ at the same time, which ensures that the next syndrome $S'(x)$ is ready while BM algorithm finishes its previous work. After a series of calculation of Chien search and error value evaluation, the decision making unit refresh the correction $e(x)$ for the output when each decoded codeword arrives. When the 8th pattern arrives, the final codeword is then determined and outputted.

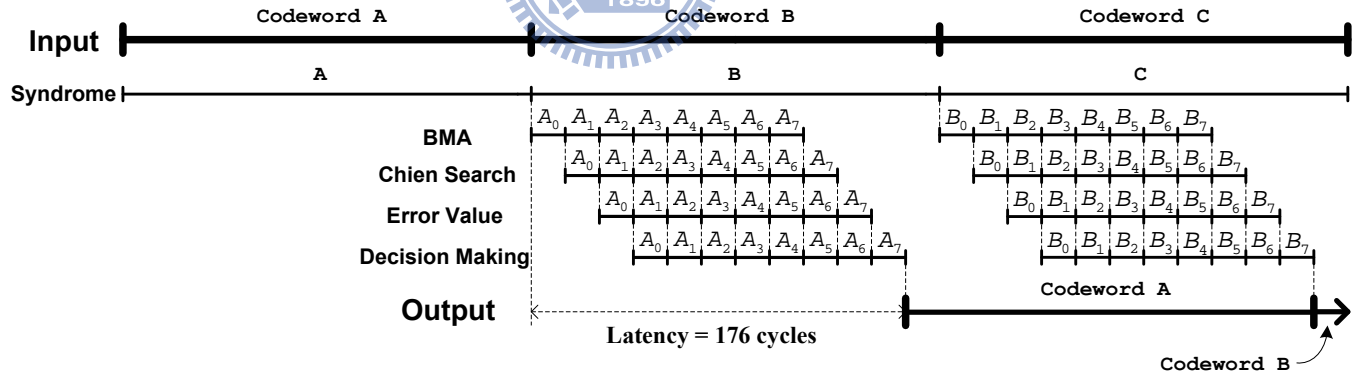


Figure 4.5: Timing schedule of a soft-decision Chase-type RS decoder design

To limit the memory usage, we expect to make the 1st output pattern ready before the 3rd pattern inputted. Thus, according to our schedule, at most $\frac{224}{11} \approx 20$ cycles are available for each pipeline stage. Consider the critical path and hardware cost of each blocks, each blocks are well-arranged and takes only 16 cycles, which reduces the latency

of output to 176 cycles.

4.2.1 Syndrome calculator and Chase calculator

To calculate the syndrome, an efficient architecture is use. In Figure 4.6, the syndrome $S_1 \sim S_8$ are refreshing iteratively and will be valid once the input finished.

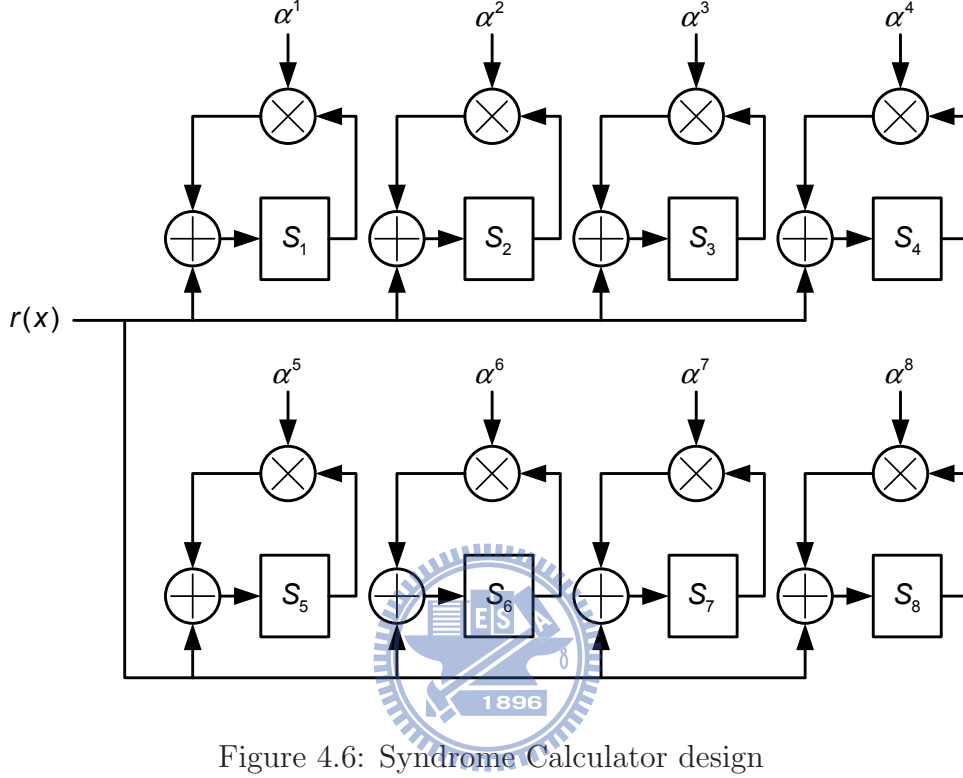


Figure 4.6: Syndrome Calculator design

Chase Calculator also involves syndrome calculation. To reduce the number of multiplier use, modifying sequence of Chase algorithm is design as the gray code system, Fig. 4.7 shows the sequence of the gray code permutation. Note that each 1's represents bit-flipping at corresponding error position. For example, the LRIPs are at 3rd, 50th and 67th bits of the recieved data, then a "011" in the Fig. 4.7 means to flip the 50rd and 67th bits and do nothing to the 3th bit. But we only have to flip the 50th bit in the design since it is permuted from the "001".

By this way, the candidate sequences will be modified exactly one bit each time. Thus, it only requires one multiplier but one look-up tables to get specific number of power of $GF(2^8)$. In Fig. 4.8, the j_i represents the error location of the symbol that bit-flipping occured, which e_i represents the error value of that symbol. And the $e'(x)$ is the calculated

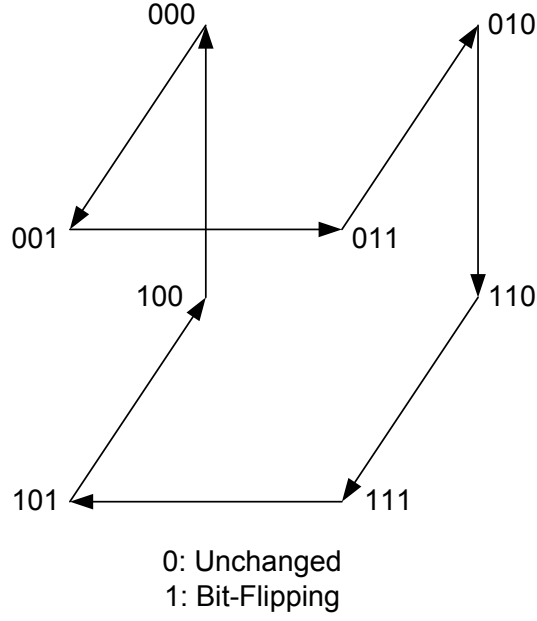


Figure 4.7: Gray code example used in the design

syndrome, which will be added to the syndrome to be sent to the BM algorithm.

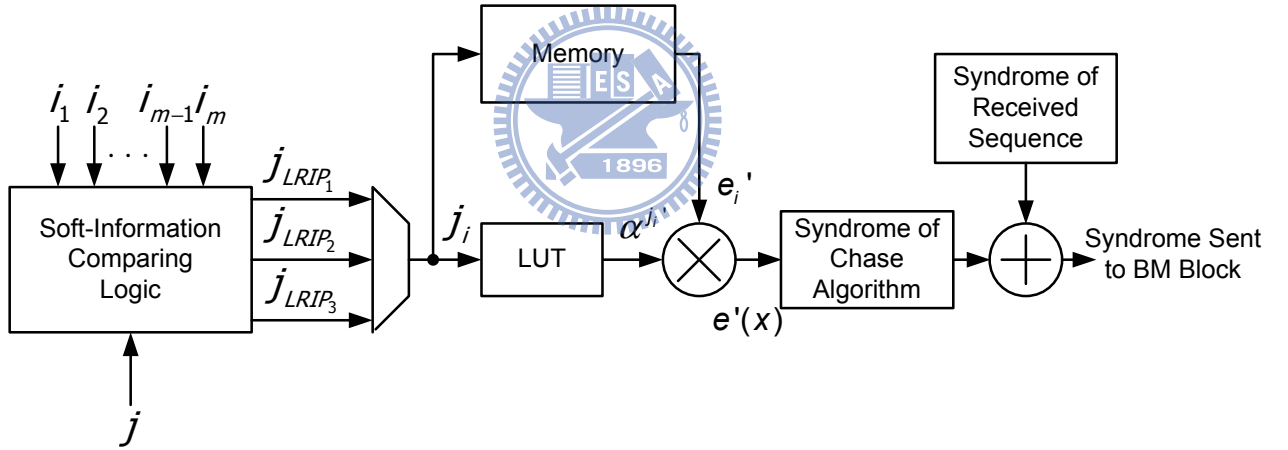


Figure 4.8: Chase Calculator design

4.2.2 BM algorithm

According to [15], riBM and RiBM are efficient architecture for BM algorithm implementation. These two architectures have shorter critical path, and most important, they figure out the key equation $\Omega(x)$ required for error value evaluation step. However, it has

some draw back in hardware usage.

Table 4.1: Comparison over BM architecture

| Architecture | Adders | Mult. | Reg. | Muxes | Clock cycles | Critical path |
|-------------------|----------|----------|----------|----------|--------------|-------------------------------------|
| iBM(1) | $2t + 1$ | $3t + 3$ | $4t + 2$ | $t + 1$ | $3t$ | $T_{mult} + \log_2 t \cdot T_{add}$ |
| riBM | $3t + 1$ | $6t + 2$ | $6t + 2$ | $3t + 1$ | $2t$ | $T_{mult} + T_{add}$ |
| RiBM | $3t + 1$ | $6t + 2$ | $6t + 2$ | $3t + 1$ | $2t$ | $T_{mult} + T_{add}$ |
| iBM (proposed) | $2t$ | $2t + 1$ | $2t + 4$ | $t + 3$ | $4t$ | $T_{mult} + \log_2 t \cdot T_{add}$ |

For our design, consider the clock time require and critical path of Chien Search block since it has to search all $n = 224$ roots. From a viewpoint of pipeline, architecture for BM algorithm does not have to be so fast. And, because the critical path of BM algorithm is relatively shorter than Chien Search block, the most important goal for the BM algorithm design is to reduce the hardware cost. In our design, a traditional inversionless Berleykamp-Messay (iBM) [16] architecture is used. Moreover, we further reduce the multipliers usage from $3t + 3$ to $2t + 1$, with a trade off on clock cycle which is doubled from $2t$ to $4t$.

In the Fig. 4.9, the computation can be separated into two parts. At first cycle $i = 0, 2, \dots, 2t - 2$, calculate the $\sigma(x)$ as following:

$$\sigma^{(\frac{i}{2})}(x) = \delta \cdot \sigma^{(\frac{i}{2}-1)}(x) + \Delta_{\frac{i}{2}} \cdot x\tau^{(\frac{i}{2}-1)}(x) \quad (4.3)$$

At the next cycle $i = 1, 3, \dots, 2t - 1$, calculate the remaining parameters:

4.2.3 Chien search and error value evaluator

When the error locator polynomial $\sigma(x)$ is obtained, the Chien search block start searching each position j_i , determining whether it's in error by substituting the $\alpha^{(-j_i)}$ to $\sigma(x)$. According to the pipeline schedule in Figure 4.5, we duplicate the Chien search series into two parallel lines, in this case, we're able to search 16 positions each cycle. By checking the C_{j_i} , the positions and total number of error can be determined. Note

$$\begin{aligned}
\Delta_{\frac{i+1}{2}} &= S_{\frac{i+1}{2}+1} + S_{\frac{i+1}{2}}\sigma_1^{(\frac{i-1}{2})} + \cdots + S_1\sigma_{\frac{i+1}{2}}^{(\frac{i-1}{2})} \\
\text{if}(\Delta_{\frac{i-1}{2}} = 0 \quad \text{or} \quad 2L_{\frac{i-3}{2}} \geq \frac{i+1}{2})\{ \\
&L_{\frac{i-1}{2}} = L_{\frac{i-3}{2}} \\
&\tau^{(\frac{i-1}{2})}(x) = x\tau^{(\frac{i-3}{2})}(x) \\
&\} \text{ else}\{ \\
&L_{\frac{i-1}{2}} = \frac{i+1}{2} - L_{\frac{i-3}{2}} \\
&\tau^{(\frac{i-1}{2})}(x) = \sigma^{(\frac{i-3}{2})}(x) \\
&\delta = \Delta_{\frac{i-1}{2}} \\
&\}
\end{aligned}$$

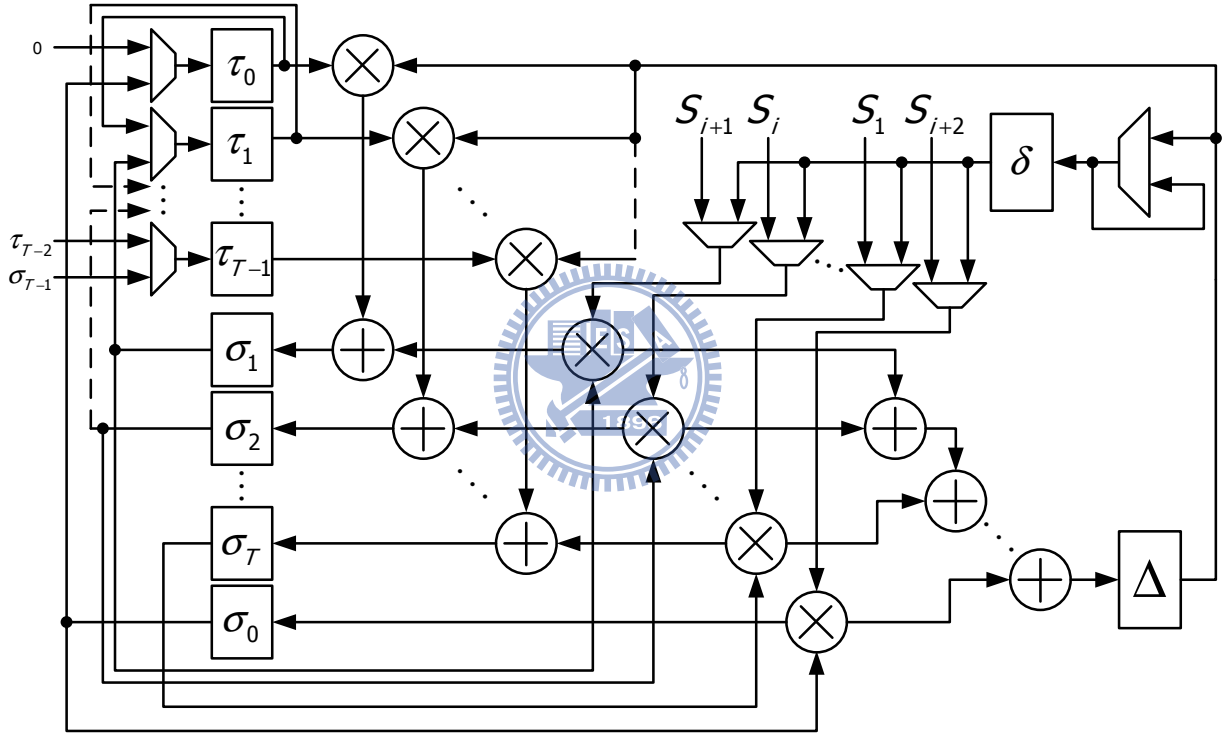


Figure 4.9: Berlekamp-Massey architecture design

that the sequence may be uncorrectable if the degree of error locator polynomial $\sigma(x)$ is inconsistent with number of roots found.

However, while the Chien search architecture in Fig. 4.10 enables us to search the roots of $\sigma(x)$ quickly, another fast hardware architecture is required to record the error locations searched by Chien search in each cycles. In our design, two parallel serial

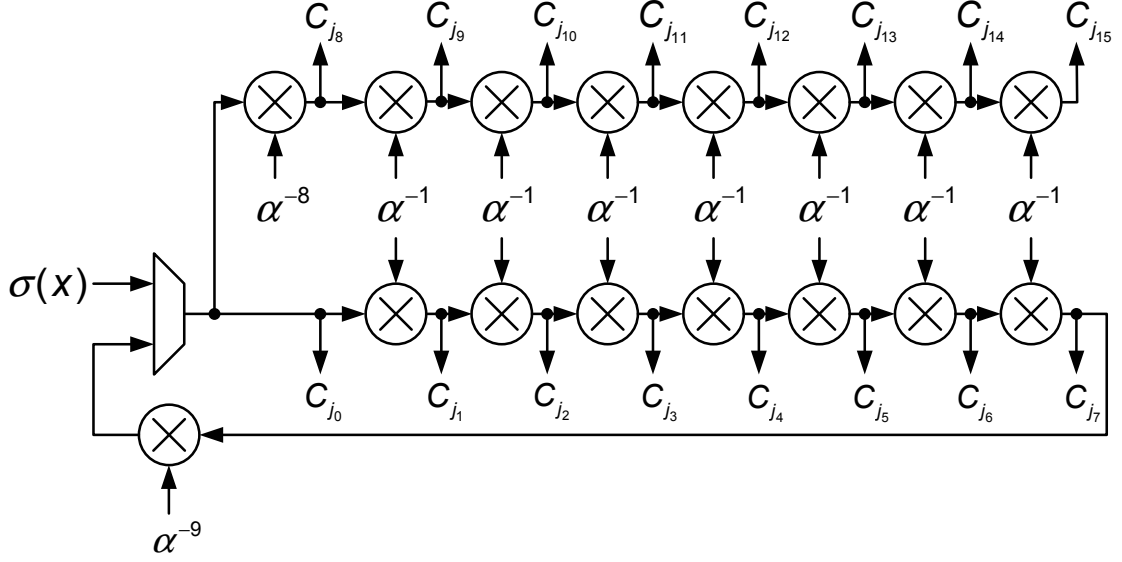


Figure 4.10: Chien search architecture design

shifters are used to record the information from Chien Search. The β_0 and β_1 in the figure represent the signal used to store error location. At each multiplexers in Fig. 4.11, if the C_{j_i} is zero, then the β added the information j_i into itself, where the j_1 represents the error location which C_{j_1} stands for. Note that the order of error positions being stored does not affect the correctness of following BP algorithm. Thus, the β_0 and β_1 can store the error location information respectively and independently, and these two signal will be merged together at the next cycle. By using the serial shifter, the number of roots and the value of each roots can be recorded properly.

In most high-speed RS decoder design, Forney's algorithm is implemented and used as error value evaluator. Cooperating with some fast architecture, Forney's algorithm and Chien search can be processing at the same time once the key equation $\Omega(x)$ is known. However, in our design, it will require a great number of Chien search parallelism to reach such a high-speed design, or the speed and latency of the decoder will be pulled down.

Bjorck-Pereyra (BP) Algorithm is an efficient algorithm for solving Vandemonde matrix. The matrix shown below is the matrix to solve in RS system, where the e_i are the error value corresponding to each X_i . BP algorithm has low computation complexity, and it's hardware cost is even less than that of Forney's algorithm, as in Fig. 4.13, only 1 divider, 1 multipliers and 2 adders are used. Although BP algorithm takes t^2 cycles time to work, in our design, it match our timing schedule perfectly.

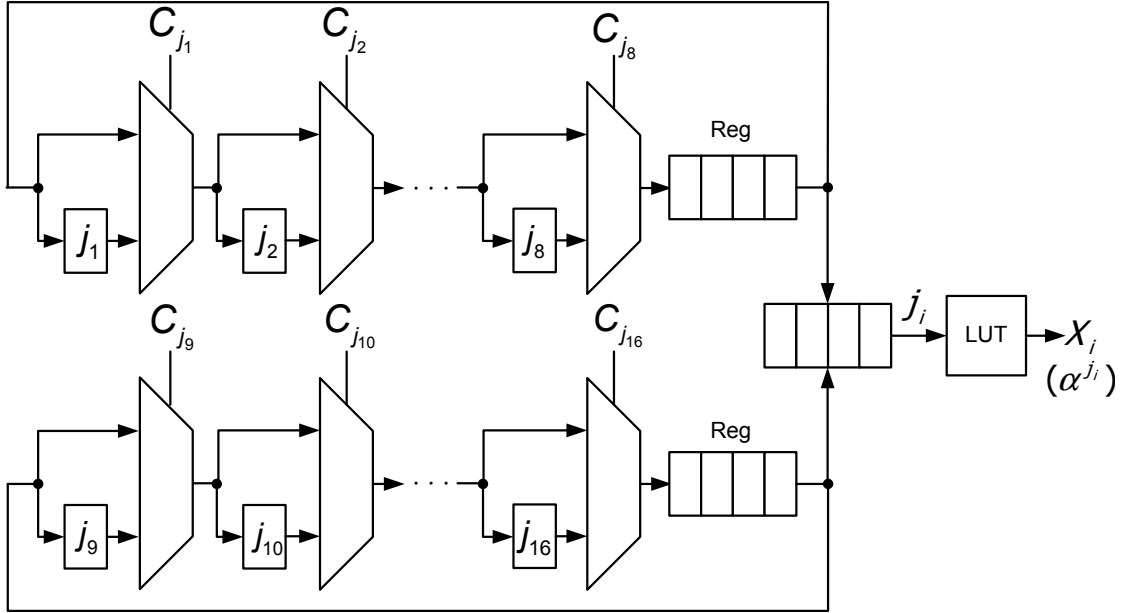


Figure 4.11: Serial shifter for error location record

$$\begin{bmatrix} X_1 & X_2 & X_3 & X_4 \\ X_1^2 & X_2^2 & X_3^2 & X_4^2 \\ X_1^3 & X_2^3 & X_3^3 & X_4^3 \\ X_1^4 & X_2^4 & X_3^4 & X_4^4 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} \quad (4.4)$$

```

For  $k = 1$  to  $t - 1$ 
  For  $i = t$  to  $k + 1$ 
     $S_i = S_i - X_k S_{i-1}$ 
  For  $k = t - 1$  to  $1$ 
    For  $i = k + 1$  to  $e$ 
       $S_i = \frac{S_i}{X_i - X_{i-k}}$ 
       $S_{i-1} = S_{i-1} - S_i$ 
  For  $k = 1$  to  $t$ 
     $S_k = \frac{S_k}{X_k}$ 

```

Figure 4.12: Bjorck-Pereyra algorithm

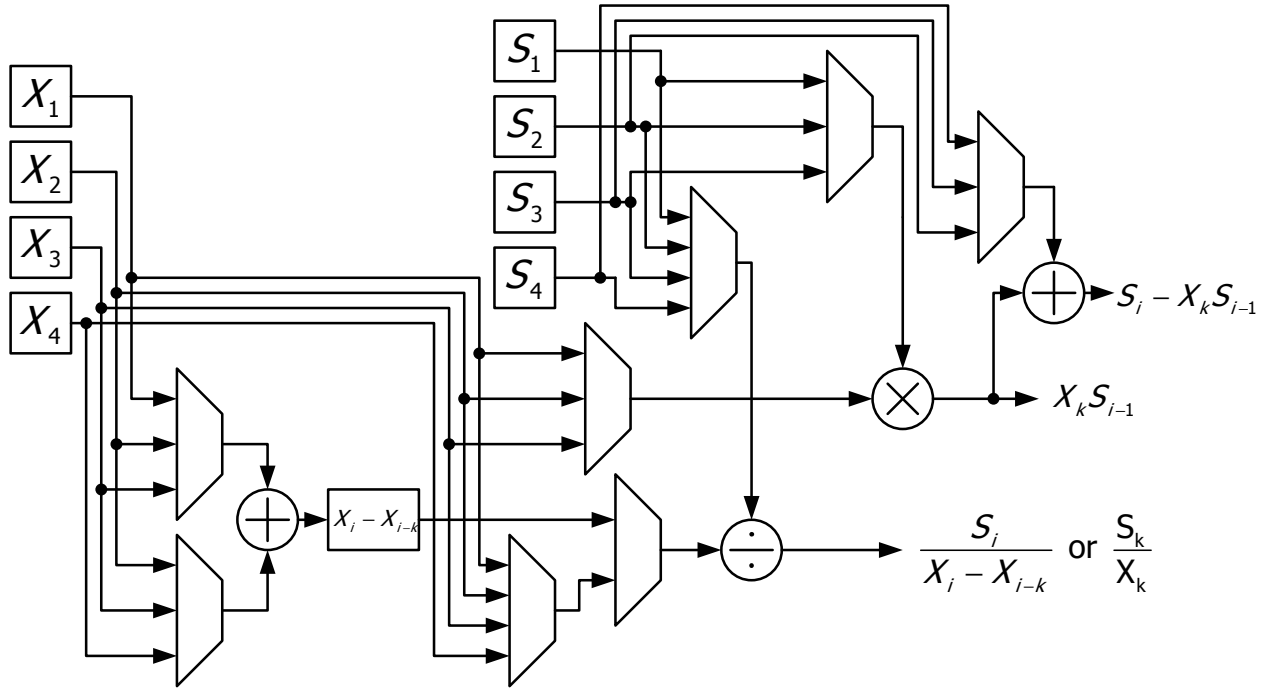


Figure 4.13: Bjorck-Pereyra architecture design

4.2.4 Decision making unit

Decision making unit is the last stop of our decoding process. Since the error values and locations have been found, the only task remain is to determine which codeword to output. As mentioned in Section 3.1, even the Hamming distance of each codeword $c_i(x)$ to the recieved pattern $r(x)$ is sufficient to choose the desired output. To calculated the distance, we only have to calculate the difference made by Chase algorithm and error correction. Which means at most 7 symbols are in different while counting the distance of codeword.

$$c_i(x) = r(x) + e_{ErrorCorrect}(x) + e_{ChaseModify}(x) \quad (4.5)$$

As in Fig. 4.14, there are two situation for each candidate codeword: correctable and uncorrectable. If the codeword is uncorrectable, the Hamming distance of the codeword does not have to be calculated. However, a uncorrectable counter is required. If all codewords are uncorrectable, the decoder outputs the recieved pattern directly according to the uncorrectable counter. If the codeword is correctable, calculating the Hamming distance d_i and then comparing it with current minimum distance d_{min} . If d_i is smaller,

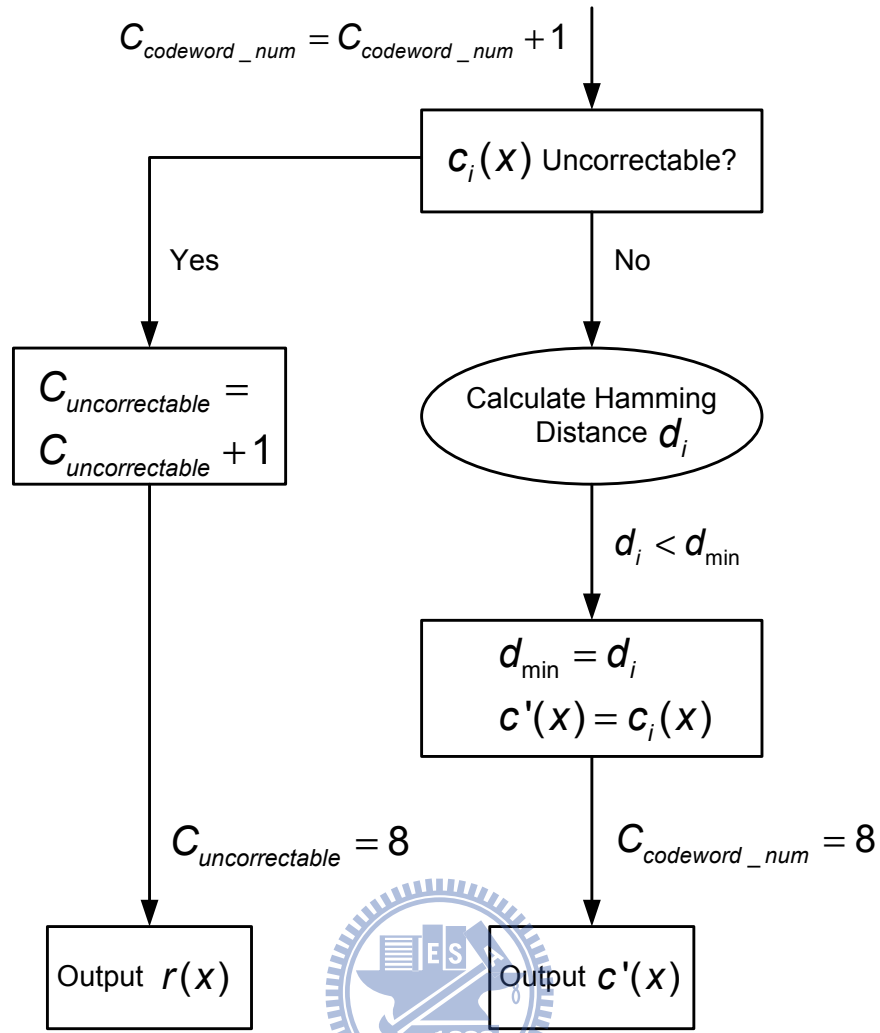


Figure 4.14: Flow chart for decision making block

replacing the d_{min} with d_i and changing the candidate codeword $c'(x)$ to $c_i(x)$. While 2^n candidate codeword are all processed, the result is determined and outputed.

Chapter 5

Implementation result

In this chapter, the design flow for the decoder will be introduced, and implementation results of the Chase-type soft-decision RS decoder will be shown. Furthermore, comparisons with other RS decoder is illustrated in this chapter.

5.1 RTL design flow

Fig. 5.1 shows the entire design flow with a various type of CAD tools. To get started, the first task is to verify our design in algorithm level with high level programming language, such as C/C++ or MATLAB. These tools help us to generate random noise from AWGN and test pattern for hardware verification. It is important to construct the environment carefully. Once we are heading for hardware simulation, the software simulation result is almost the only efficient information to check the correction of our hardware design.

Verilog is chosen as the RTL implementation language. After RTL level design and verification, the RTL-to-Gate synthesis applied with Synopsys Design Compiler, the standard library for synthesis is 90nm 1P9M CMOS technology. Gate area and circuit timing is improved significantly in 90nm. After the gate level synthesis, a gate-level simulation and check is performed for the performance and correctness of the design.

After successful gate-level simulation, the place and route procedure is performed with the SoC Encounter CAD tool. In deep-submicron design, problems like design rule violation, signal integrity (SI) and IR drop are to be considered and await to be solved.

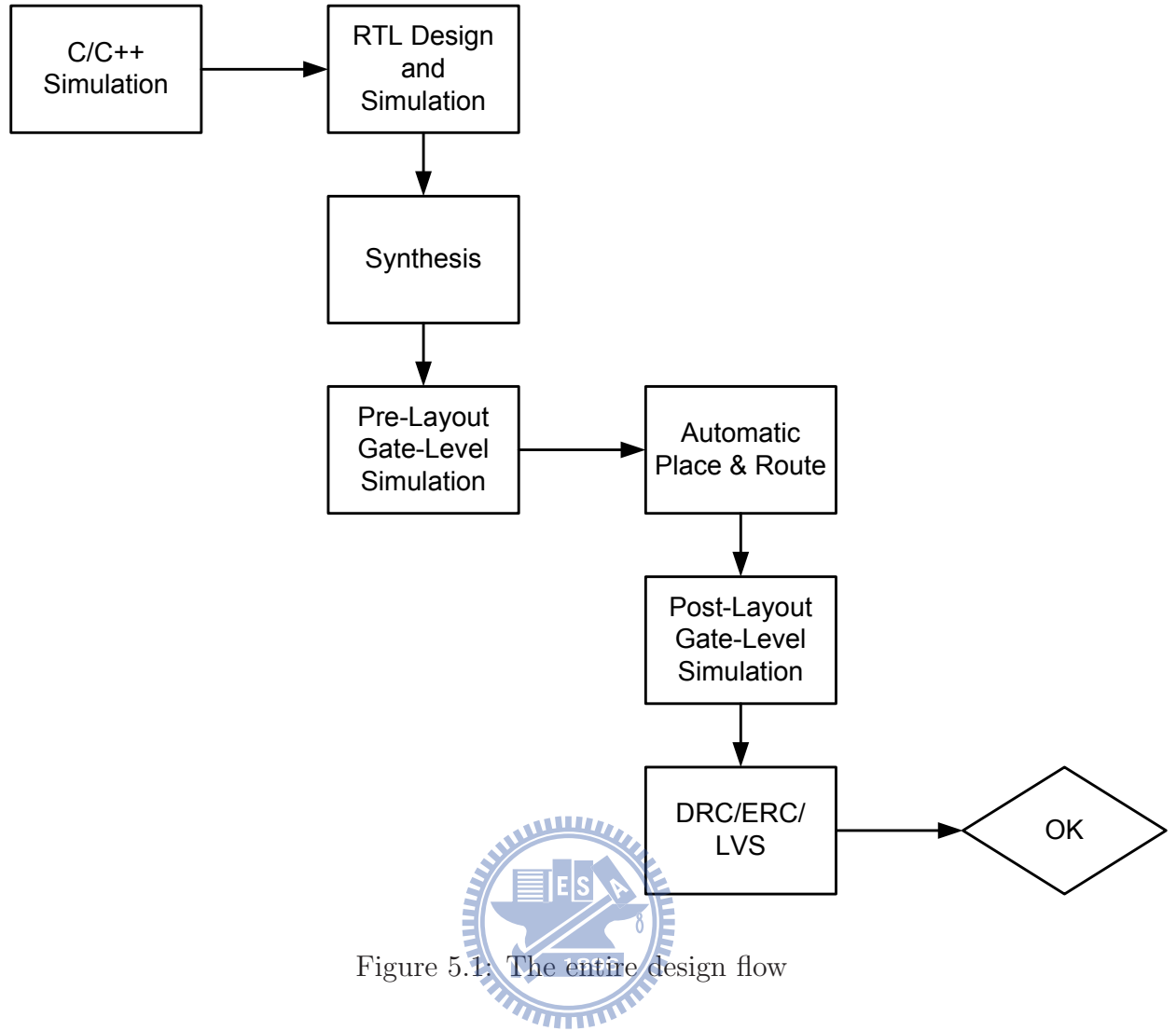


Figure 5.1: The entire design flow

The timing check, RC extraction, DRC (design rule check), LVS(layout versus schematic) and power consumption will be calculated in the place and route procedure. Finally, the post-layout simulation is carried out to check the correctness of the chip integrated.

5.2 Implementation result

The final layout of the designed decoder is shown in Fig. 5.2. The proposed soft-decision Chase-type RS(224,216) decoder can work at a minimum cycle time of $3.2ns$, that is , the decoder running at a frequency of $312.5MHz$. By applying the following equation, the throughput of the decoder is $2.41G$.

$$Throughput(bits/s) = frequency \times bit \text{ per symbol} \times code \text{ rate} \quad (5.1)$$

The gate count of proposed decoder is 27.5k exclusive memory. To estimate actual gate count of the the computation block. A detail summary table is listed as Table. 5.1.

Table 5.1: Summary of proposed design

| Design | Gate Count (exclusive memory) | Core Area (μm^2) | Utilization | Avg. Power (mw) |
|------------------|----------------------------------|----------------------------|-------------|--------------------|
| Proposed Decoder | 27499 | 471.56×468.72 | 82.1% | 23.848 |

5.3 Comparison of different RS decoders

To the best knowledge of the author, there was no decoders on RS(224,216) has been published. Thus, to make the comparison, we simply transform and shorten the decoders for RS(255,239) in [17] and [18] to RS(224,216). The comparison result is listed below.

In the Table. 5.2(a), we specified and separated the variable multiplier and constant multiplier. According to the real implementation result, the gate count of a variable multiplier is 6 ~ 20 times of a constant multiplier. Compared with the Low-Complexity Chase (LCC) RS decoder, our proposed decoder is much more efficient in the total gate count of multipliers. And our register and memory usage are also much smaller in number. Though there are some overhead in the use of adders and multiplexers. The proposed RS decoder is still area-efficient. The latency of the proposed decoder is also smaller than that of LCC RS decoder.

According to Table. 5.2(b), our design costs more hardware than the Ultra-folded inversionless Berlekamp-Massey (UiBM) RS decoder on almost all the components listed. Note that the UiBM is the hard-decision RS decoder, our proposed decoder is better in the error-correction ability. For the UiBM system to achieve the same performance in the same latency, it will has to be parallelized with several copies. In that way, the proposed decoder is frugal in contrast with direct parallism since it only 2 ~ 3 times of the gate count of the hard-decision RS decoder.

The performance curve of each decoder is shown in Fig. 5.3. According to the simulated performance curve, the LCC decoder has the same error correction ability as proposed Chase-type decoder.

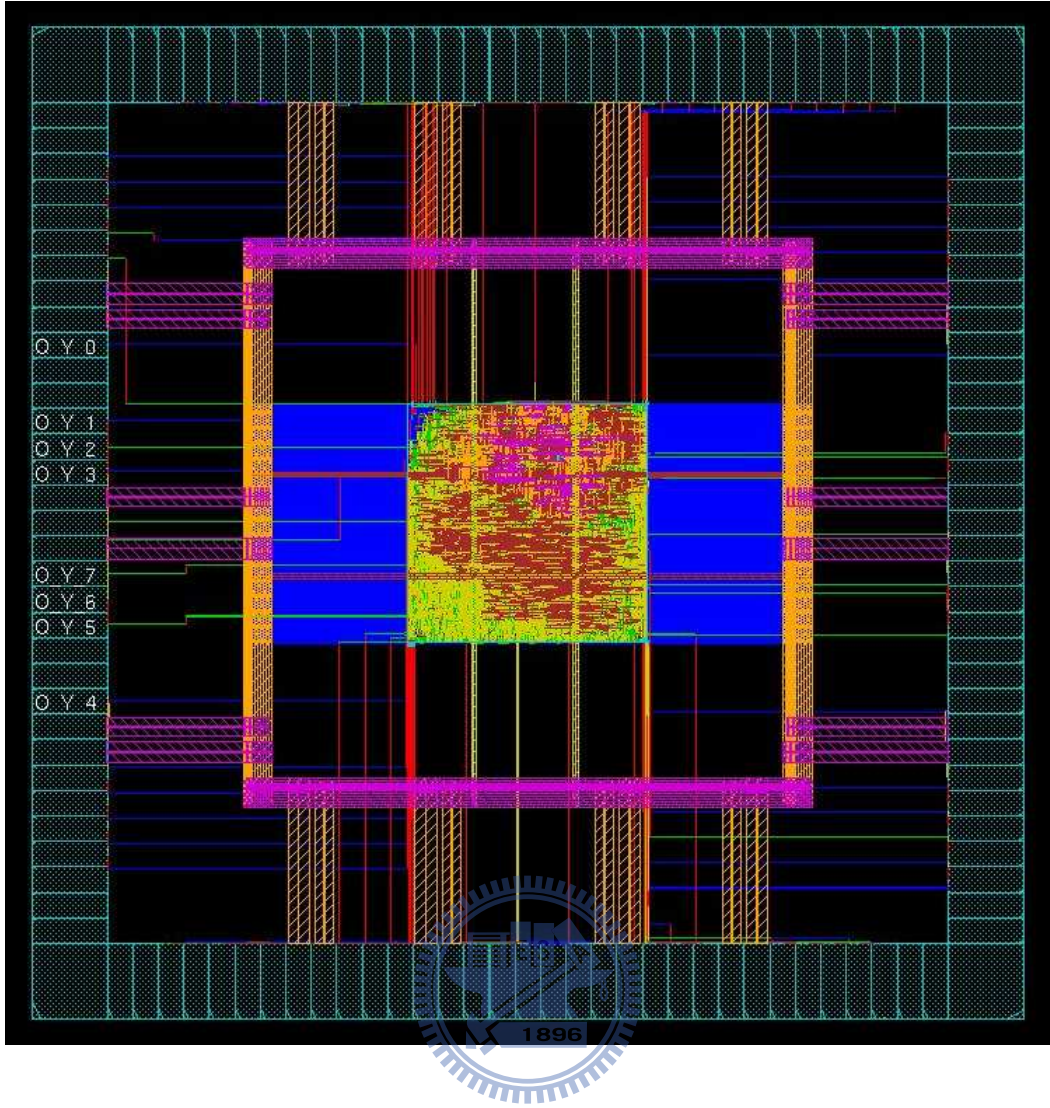


Figure 5.2: The layout of proposed RS decoder

In conclusion, the proposed design is relatively efficient in hardware cost comparing with the other soft-decision RS decoder, and it has a better performance in error correction compared with conventional RS decoder at an acceptable cost.

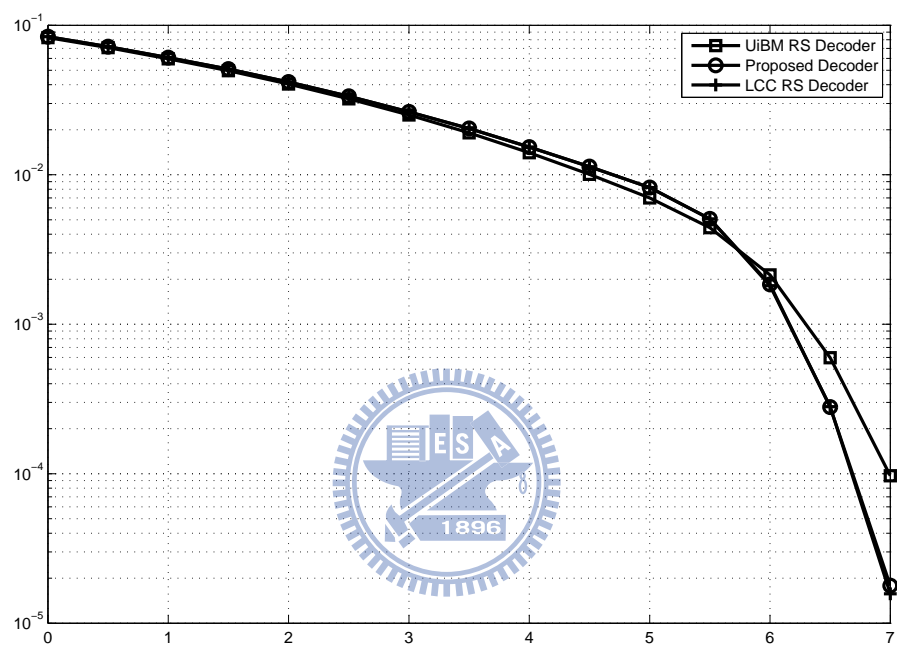


Figure 5.3: Performance curve for comparison

Table 5.2: Comparison of hardware requirements (a) LCC (b) UiBM

(a)

| Architecture | Galois Field Variable Multiplier | Galois Field Constant Multiplier | Galois Field Adder | 2-to-1 MUX | Register (bit) | LUT (byte) | FIFO (byte) | Latency (cycle time) |
|---------------------|--|--|-----------------------|---------------|-------------------|---------------|---------------------|----------------------------|
| Proposed | | | | | | | | |
| Syndrome Calculator | 0 | 8 | 8 | 0 | 64 | 0 | 0 | 224 |
| Chase Calculator | 1 | 0 | 1 | 56 | 216 | 224 | 0 | 0 |
| BM Algorithm | 9 | 0 | 8 | 93 | 136 | 0 | 0 | 16 |
| Chien Search | 0 | 64 | 64 | 519 | 355 | 0 | 0 | 16 |
| BP Algorithm | 2 | 0 | 3 | 168 | 64 | 224 | 0 | 16 |
| Decision Making | 0 | 0 | 2 | 268 | 177 | 0 | 0 | 16 |
| Total | 12 | 72 | 86 | 1104 | 835 | 448 | $0 + 224 \times 2$ | 224 |
| LCC(255,239) [13] | | | | | | | | |
| Re-encode | 21 | 0 | 39 | 448 | 600 | 512 | 0 | 528 |
| Interpolation | 14 | 0 | 12 | 87 | 166 | 0 | 68 | 525 |
| Polynomial Select | 8 | 0 | 8 | 139 | 264 | 0 | 0 | 23 |
| Chien Search | 0 | 8 | 8 | 0 | 128 | 0 | 0 | 239 |
| Forney Algorithm | 2 | 0 | 2 | 136 | 24 | 256 | 0 | 152 |
| Erasur Decoder | 21 | 0 | 39 | 299 | 424 | 256 | 0 | 528 |
| Total | 66 | 8 | 108 | 1109 | 1606 | 1024 | $68 + 256 \times 8$ | 528 |
| LCC(224,216) | | | | | | | | |
| Re-encode | 13 | 0 | 23 | 392 | 344 | 448 | 0 | 464 |
| Interpolation | 14 | 0 | 12 | 87 | 166 | 0 | 68 | 461 |
| Polynomial Select | 8 | 0 | 8 | 139 | 264 | 0 | 0 | 23 |
| Chien Search | 0 | 4 | 4 | 0 | 64 | 0 | 0 | 216 |
| Forney Algorithm | 2 | 0 | 2 | 136 | 24 | 224 | 0 | 76 |
| Erasur Decoder | 13 | 0 | 23 | 243 | 168 | 224 | 0 | 464 |
| Total | 50 | 4 | 72 | 997 | 1430 | 896 | $68 + 224 \times 8$ | 464 |

(b)

| | | | | | | | | |
|---------------------|----------|-----------|-----------|------------|------------|-------------|--------------------|------------|
| UiBM(255,239) [17] | | | | | | | | |
| Syndrome Calculator | 0 | 8 | 8 | 64 | 128 | 0 | 0 | 510 |
| BM Algorithm | 2 | 0 | 1 | 21 | 413 | 0 | 0 | 400 |
| Chien Search | 0 | 4 | 4 | 48 | 128 | 0 | 0 | 510 |
| Forney Algorithm | 1 | 4 | 4 | 32 | 64 | 256 | 0 | 0 |
| Total | 3 | 16 | 17 | 165 | 733 | $256 + 256$ | $0 + 256 \times 3$ | 510 |
| UiBM(224,216) | | | | | | | | |
| Syndrome Calculator | 0 | 8 | 8 | 64 | 128 | 0 | 0 | 224 |
| BM Algorithm | 2 | 0 | 1 | 21 | 277 | 0 | 0 | 104 |
| Chien Search | 0 | 4 | 4 | 48 | 128 | 0 | 0 | 224 |
| Forney Algorithm | 1 | 4 | 4 | 32 | 64 | 224 | 0 | 0 |
| Total | 3 | 16 | 17 | 109 | 597 | $224 + 224$ | $0 + 224 \times 3$ | 224 |

Chapter 6

Conclusion and Future Work

6.1 Conclusion

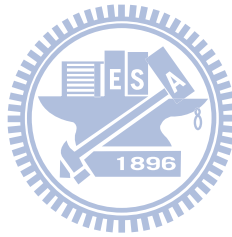
In this thesis, we proposed an area-efficient Chase-type RS(224,216) decoder that the performance gain is 0.5dB comparing with hard-decision RS at $BER = 10^{-5}$. The throughput of the design is $2.41Gbit/s$ at the operation frequency 312.5MHz. The core area is $471.56 \times 468.72\mu m^2$ with the power consumption of $2.38mW$. The gate count of the proposed decoder is $27.5k$. Our design is efficient and superior in comparison with Low-Complexity Chase (LCC) RS decoder. The proposed decoder composed of less variable multipliers, which is the most complex hardware block of the system. And the LCC decoder only has similar performance gain as our proposed decoder. Our proposed design can be used to realize the uncompressed HD video transmission system specified by IEEE 802.15.3c specification.

6.2 Future work

In our design, there are still some issues to be concerned. First, the storage unit use in the design is using register storage so far, a new version of the design using RAM memory to replace those block is neccessary. Besides, the throughput of the design is still unsatisfied. The next goal for us is to design and implement a RS(224,216) decoder with improved performance to have a throughput exceed $3Gbit/s$.

Besides, for transmission specified by IEEE 802.15.3c [2], a convolutional decoder is

required to deal with the inner code while RS decoder deal with the outer code. More experiments on the combination of our RS decoder and convolutional decoder such as Viterbi decoder are necessary to the goal of realizing the transmission system.



Bibliography

- [1] H. Singh, H. Niu, X. Qin, H. Shao, C. Kwon, G. Fan, S. Kim, and C. Ngo, “Supporting uncompressed HD video streaming without retransmissions over 60GHz wireless networks,” *IEEE Wireless Communications and Networking Conference, WCNC*, pp. 1939–1944, 2008.
- [2] *Millimeter-wave based Alternative Physical Layer Extension*, IEEE Std. P802.15.3c/D02, 2008.
- [3] R. Koetter and A. Vardy, “Algebraic soft-decision decoding of Reed-Solomon codes,” *IEEE Trans Inform. Theory*, vol. 49, no. 11, pp. 2809–2825, 2003.
- [4] I. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *J. Soc. Indust. and Appl. Math*, vol. 8, no. 2, pp. 300–304, June 1960.
- [5] E. R. Berlekamp, *Algebraic Coding Theory*. McGraw-Hill, 1968.
- [6] J. Massey, “Shift-register synthesis and BCH decoding,” *IEEE Trans Inform. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.
- [7] D. Chase, “A class of algorithms for decoding block codes with channel measurement information,” *IEEE Trans. Inform. Theory*, vol. 18, no. 1, pp. 170–182, Jan. 1972.
- [8] M. Lalam, K. Amis, and D. Leroux, “On the use of Reed-Solomon codes in space-time coding,” *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, pp. 31–35, Sept. 2005.
- [9] V. Guruswami and M. Sudan, “Improved decoding of Reed-Solomon and algebraic-geometry codes,” *IEEE Trans Inform. Theory*, vol. 45, pp. 1757–1767, 1999.

- [10] W. Gross, F. Kschischang, R. Koetter, and P. Gulak, "Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders," *Journal of VLSI Signal Processing*, vol. 39, pp. 93–111, 2005.
- [11] G. Feng and K. Tzeng, "A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes," *IEEE Trans Inform. Theory*, vol. 37, pp. 1274–1287, Sept. 1991.
- [12] W. Gross, F. Kschischang, R. Koetter, and M. Sudan, "A VLSI architecture for interpolation in soft-decision list decoding of Reed-Solomon code," *IEEE Workshop on Signal Processing Systems*, pp. 39–44, Oct. 2002.
- [13] J. Bellorado and A. Kavcic, "A low-complexity method for Chase-type decoding of Reed-Solomon codes," *Proc. ISIT*, pp. 2037–2041, Jul. 2006.
- [14] J. Zhu, X. Zhang, and Z. Wang, "Novel interpolation architecture for low-complexity Chase soft-decision decoding of Reed-Solomon codes," *IEEE International Symposium on Circuits and Systems, ISCAS*, pp. 3078–3081, May 2008.
- [15] D. Sarwate and N. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Trans. VLSI System*, vol. 9, no. 5, pp. 641–654, OCT. 2001.
- [16] I. S. Reed, M. T. Shih, and T. K. Truong, "VLSI design of inverse-free Berlekamp-Massey algorithm," *Proc. Inst. Elect. Eng*, vol. 138, pp. 295–298, Sept. 1991.
- [17] K. Seth, K. Srinivasan, and S. Kamakoti, "Ultra folded high-speed architectures for Reed-Solomon decoders," *Proc. of the 19th International Conference on VLSI Design*, Jan. 2006.
- [18] X. Zhang, "High-speed VLSI architecture for low-complexity Chase soft-decision Reed-Solomon decoding," *IEEE Inform. Theory and Application Workshop*, pp. 422–430, Feb. 2009.

作者簡歷

姓名：黃裕淳

出生地：台灣 高雄市

出生日期：1985.05.14

學歷：1999.9~2002.6 高雄市立高雄高級中學

2003.9~2007.6 國立交通大學 電機學院與資訊學院學士班 學士

2007.9~2009.10 國立交通大學 電子研究所 系統組 碩士

